Universidad de La Laguna

Escuela Técnica Superior de Ingeniería y Tecnología
Dept. de Ingeniería Informática y de Sistemas

A dissertation submited in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

Dynamic state estimation for mobile robots

Estimación dinámica de estados para robots móviles

Antonio Luis Morell González

June 2017

Supervisors: *Leopoldo Acosta Sánchez*
Jonay Tomás Toledo Carrillo

D. Leopoldo Acosta Sánchez, Doctor en Física y Catedrático del Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna.

D. Jonay Tomás Toledo Carrillo, Doctor en Informática y Profesor Titular del Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna.

CERTIFICAN:

que D. Antonio Luis Morell González, Ingeniero en Automática y Electrónica Industrial, ha realizado bajo nuestra dirección la presente Tesis, titulada "Dynamic state estimation for mobile robots (Estimación dinámica de estados para robots móviles)", para optar al grado de Doctor en Física e Ingeniería por la Universidad de La Laguna.

Con esta fecha, autorizo la presentación de la misma.

En San Cristóbal de La Laguna, a 13 de Junio de 2017.

Los Directores,

Leopoldo Acosta Sánchez

Jonay Tomás Toledo Carrillo

A Valeria y a Daniela,
porque el futuro es para ellas.

Agradecimientos

Empezar una aventura como la que este manuscrito intenta plasmar supone tomar varias deciciones nada fáciles de explicar a familiares y amigos. En primer lugar, hace falta el aliento, confianza e inspiracíon que tanto he recibido de mi director, el Dr. D. Leopoldo Acosta Sánchez y co–director, el Dr. D. Jonay Tomás Toledo Carrillo, a quienes debo mi más sincera gratitud por su dedicación y paciencia. Gracias por depositar en mi su apoyo, tanto antes como después de que tomara la decisión de perseguir una vocación tan exigente como la investigadora.

Hacen falta comprensión, empatía y hombros sobre los que apoyarse. He tenido la inmensa suerte de recibirlos con creces de parte de D. Francisco J. Fumero Batista, el Dr. D. José Daniel Perea Ström y la Dra. Dña Yanira González González, con quienes no puedo estar más agradecido. Por permitirme aprender a dar la mejor versión de mi mismo y por compartir este largo peregrinaje hasta su mismo desenlace. Del mismo modo, no es fácil encontrar a personas como el Dr. D. Juan J. Trujillo, el Dr. D. Javier Jesús Sánchez Medina y el Dr. D. Carmelo Militello Militello, que han sido una inagotable fuente de inspiración, y que sin cuyas aportaciones, tanto personales como profesionales, este trabajo no habría sido posible. A la Dra. Dña. Silvia Alayón Miranda, por demostrar esa gran paciencia, empatía y corazón de forma siempre desinteresada. Al Dr. D. Juan Albino Méndez Pérez, que ha sido más que un compañero y amigo, por saber escucharme en los momentos más difíciles y por estar ahí cuando lo he necesitado. Al Dr. D. Jesús Miguel Torres Jorge, por su diligencia y dedicación, por dedicar su tiempo desinteresadamente y ser un modelo de integridad y coherencia. Al Dr. D. Pedro Antonio Toledo Delgado al que gran admiro, por esas largas y tardías discusiones que tanta luz han arrojado durante este camino.

También hay que saber rodearse de las mejores personas, para que podamos aspirar a superarnos cada día. No puedo sino agradecer al gran equipo humano que ha compartido conmigo tantos momentos desarrollando a *Verdino*, como parte del Grupo de Robótica de la Universidad de La Laguna. Al Dr. D. Rafael Arnay del Arco, Dr. D. Néstor Morales Hernández, Dr. D. Javier Hernández Aceituno, Dr. D. Jesús Javier Espelosín Ortega, D. Jonatán Felipe García y D. Juan Esteban Rodríguez Rosales, por hacer inolvidables tanto los momentos buenos y los no tan buenos, los maratones y las satisfacciones, tanto dentro como fuera de ese nuestro querido cochito.

Del mismo modo, no sabría como expresar la enorme deuda que tengo con la entrega que a diario me han brindado, dentro y fuera del laboratorio, la Dr. Dña. Ángela Hernández López, Dr. Dña Kelin Victoria Zuñiga Meneses, Dr. Dña. Mariana Cairós González, Dña Elisenda Eva Espino Espino, Dña. Elena Santos Hernández, Dña. Silvia Vera González, Dr. D. Iván Castilla Rodríguez, D. Pablo Vicente Torres Carrion, D. Dailos Reyes Díaz, D. Yeray Del Cristo Barrios Fleitas, Dr. D. Omar Núñez Regalado, Dr. D. Ayoze Marrero Ramos y D. Yeray Callero de León, con los que he podido trabajar, reír y hasta cantar.

También, a quienes han contribuído a que mi etapa como investigador en formación se desarrollara en el mejor de los ambientes: Dr. D. Lorenzo Moreno Ruiz, Dr. D. José Demetrio Piñeiro Vera, Dr. D. Alberto Hamilton Castro, Dra. Dña. Rosa María Aguilar Chinea, Dr. D. Santiago Torres Álvarez, Dr. D. José Francisco Sigut Saavedra, Dra. Dña Marta Sigut Saavedra, Dra. Dña. Vanesa Muñoz Cruz, Dr. D. Evelio José González González, Dr. D. José Luis Sánchez de la Rosa, Dr. D. Graciliano Nicolás Marichal Plasencia, Dr. D. José Ignacio Estévez Damas, Dr. D. Roberto Luis Marichal Plasencia, Dra. Dña. Carina Soledad González González, Dr. D. Juan Julián Merino Rubio, D. Eladio Hernández Díaz, D. Germán Carlos González Rodríguez, D. Carlos Alberto Martín Galán, Dr. D. Héc-

tor Javier Reboso Morales, Dr. D. Sid Ahmed Ould Sidha, Dr. D. Ginés Coll Barbuzano y D. Manuel Fernández Vera.

Finally, I cannot avoid giving a very special mention to Professor Atsuo Takanishi, head of the Department of Modern Mechanical Engineering, Waseda University, and President of the Robotics Society of Japan. I will be always in great debt with him for being my host, encouraging me and supporting my research project during my internship at Takanishi Lab, Waseda University. I would like to also send all my gratitude to the head of the Biped Humanoid Group, Kenji Hashimoto, as well as to all the members and collaborators of the team, postdoctoral researchers and Japanese and foreign students like me, specially for Yukitoshi Minami Shiguematsu, Gabriele Trovato, Martim Brandão, Matthieu Destephe, Gan Ma, Bryan Deliencourt, Ashley Edwards and other students and researchers I briefly met during my stay. These great group of great human beings contributed to open my mind on many things, both technical and mundane alike, allowing me to feel less like a foreigner during my stay in Tokyo. Thank you from the bottom of my heart.

「最後になりますが、早稲田大学高西研究室でのインターンシップのホストになって下さった総合機械工学科高西淳夫教授に心から感謝の気持ちと御礼を申し上げます。私の研究テーマに関する高西教授の指導とサポート、そしてこんな貴重な機会を下さった恩情に心より感謝しております。また、高西研究室2 足班のリーダー、橋本健二氏をはじめ、インターンシップを成功に導いて下さった皆さんに感謝の意を表します。このインターンシップで出会い、技術的な有意義な議論だけではなく、日常生活のなかで私が外国にいながらアットホームな気分にさせていただいた2 足班の南重松行紀氏、トロヴァト・ガブリエレ氏、ブランダン・マルティム氏、デステフ・マチウ氏、馬淦氏、そして私と同じくインターンシップ生のデリアンクール・ブライアン氏、エドワーズ・アシュリー氏、当時2 足班の皆さんに心より感謝いたします。」

Gracias a todas las personas que de un modo u otro han contribuído a que esta etapa que ahora concluye haya sido una experiencia estimulante e inolvidable, y en especial a toda la gente que ama la investigación y que están dispuestas a sacrificar su tiempo y su vida por ello. En especial a todas las personas involucradas en la Asociación de Jóvenes Investigadores de Tenerife —JINTE — por saber entender que la investigación no es solo acumular méritos, y ser un catalizador para motivarnos en la dura tarea de la divulgación y la reivindicación de una carrera investigadora justa y sostenible. Porque han dado lo mejor de sí mismas y me han permitido conseguir como investigador más de lo que hubiera podido imaginar.

Por último, he de agradecer a toda mi familia, en especial a mi madre, Encarnación y mi hermano, Eusebio, por hacer de este camino un sacrificio asumible, por haberme transmitido su apoyo y cariño constantemente, por haber entendido y compartido los mejores y peores momentos de esta etapa.

Y a mi padre, Eusebio, por ser un referente permamente antes, ahora, y en todo aquello que me quede por recorrer.

Abstract

The scientific goal of this thesis is to tackle different approaches for effective state estimation and modelling of relevant problems in the context of mobile robots. The starting point of this dissertation is the concept of *probabilistic robotics*, an emerging paradigm that combines state–of–the–art methods with the classic probabilistic theory, developing stochastic frameworks for understanding the uncertain nature of the interaction between a robot and its environment. This allows introducing relevant concepts which are the foundation of the localisation system implemented on the main experimental platform used on this dissertation. An accurate estimation of the position of a robot with respect to a fixed frame is fundamental for building navigation systems that can work in dynamic unstructured environments. This development also allows introducing additional contributions related with global localisation, dynamic obstacle avoidance, path planning and position tracking problems.

Kinematics on generalised manipulators are characterised for dealing with complex nonlinear systems. Nonlinear formulations are needed to properly model these systems, which are not always suitable for real–time realisation, lacking analytic formulations in most cases. In this context, this thesis tackles the serial–parallel dual kinematic problem with a novel approach, demonstrating state–of–the–art accuracy and real–time performance. With a spatial decomposition method, the forward kinematics problem on parallel robots and the inverse kinematics problem on serial manipulators is solved modelling the nonlinear behaviour of the pose space using Support Vector Machines. The results are validated on different topologies with the analytic solution for such manipulators, which demonstrates the applicability of the proposed method.

Modelling and control of complex dynamical systems is another re-

levant field with applications on mobile robots. Nonlinear techniques are usually applied to tackle problems like feature or object tracking. However, some nonlinear integer techniques applied for tasks like position tracking in mobile robots with complex dynamics have limited success when modelling such systems. Fractional calculus has demonstrated to be suitable to model complex processes like viscoelasticity or super diffusion. These tools, that take advantage of the generalization of the derivative and integral operators to a fractional order, have been applied to model and control different topics related with robotics in recent years with remarkable success.

With the proposal of a fractional–order PI^{λ} controller, a suitable controller design method is presented to solve the position tracking problem. This is applied to control the distance of a self–driving car with respect to an objective, which can also be applied to other tracking applications like following a navigation path. Furthermore, this thesis introduces a novel fractional–order hyperchaotic system, stabilised with a full–pseudo–state–feedback controller and a located feedback method. This theoretical contribution of a chaotic system is introduced hoping to be useful in this context. Chaos theory has recently started to be applied to study manipulators, biped robots and autonomous navigation, achieving new and promising results, highlighting the uncertain and chaotic nature which also has been found on robots.

All together, this thesis is devoted to different problems related with dynamic state estimation for mobile robots, proposing specific contributions related with modelling and control of complex nonlinear systems. These findings are presented in the context of a self–driving electric car, Verdino, jointly developed in collaboration with the Robotics Group of *Universidad de La Laguna* (GRULL).

Contents

List of Figures

List of Algorithms

Acronyms

AI	Artificial Intelligence	34
CTRW	Continuous Time Random Walk	73
CKF	Cubature Kalman Filter	16
DC	Direct Current	78
DOF	Degrees of Freedom	90
EKF	Extended Kalman Filter	15
EM	Expectation Maximization	69
ERM	Empirical Risk Minimization	36
FC	Fractional Calculus	72
FDE	Fractional Differential Equation	72
FK	Forward Kinematics	33
GPS	Global Positioning System	16
GRULL	Robotics Group of *Universidad de La Laguna*	137
IK	Inverse Kinematics	34
IMU	Inertial Measurement Unit	26
KF	Kalman Filter	15
LE	Lyapunov Exponent	93

Part I

Dynamic state estimation for mobile robots

"The last thing one discovers in composing a work is what to put first."

— Blaise Pascal, Pensées

1

State estimation, Uncertainty and Robots

SCIENTISTS and practitioners aim to understand all physical processes around us, either to mimic and improve them or to take advantage of such knowledge in many different ways and fields. This idea extends to robotics, as an universal effort to keep improving all techniques and methods that allow robots and unmanned systems to seamlessly interact with our world. However, as happens with any physical systems, robots can only be understood by taking into account uncertainties and nonlinear dynamics. Why are we interested in applications for mobile robots? The main reason is to emphasize the dynamic nature of the systems and processes involved. Robotics is a multidisciplinary field, where problems of different nature have to be tackled taking into account such nonlinear dynamics and uncertainties.

In the context of state estimation, modelling processes with complex dynamics is an open research topic in almost all scientific fields. In robotics, such uncertainties, complex systems and nonlinear dynamics

appear in almost all aspects related with the interaction with the environment and when estimating internal states of the system. Due to strict performance and real–time requirements, the physical realization of methods applied to solve state estimation and modelling problems on robotics needs to yield solutions with good enough accuracy and performance. Depending on the complexity of the target task, the type of environment —structured such as industrial plants, or unstructured like a pedestrian street — and the approach applied, models and actions are described as nonlinear systems, in an attempt to capture such complex dynamics.

The challenges on approaching these problems from the perspective of a system or device with complex dynamics are two–fold, since both the internal and external states of the system have to be considered. However, robotics is not the only domain on which these observations and results can be applied, since the techniques described can be generalised for a wide range of filtering, control and modelling applications. At the same time, this understanding intends to help achieving better design methods and algorithms that increase the accuracy on real–time systems, useful not only to mobile robots or even robotics.

Stochastic frameworks and uncertainty–driven models that emerged during the last few decades have been applied to mobile robots with great success. Also called Statistical Machine Learning (ML) research, the application of these methods reached a milestone with *Stanley*. This self–driven car was the first mobile robot that won the Defense Advanced Research Projects Agency (DARPA) Grand Challenge (Thrun et al., 2006). This competition consisted on a high–speed desert driving without manual intervention. The team behind Stanley managed to complete this task for the first time using state–of–the–art Artificial Intelligence (AI) technologies, specifically ML and probabilistic reasoning.

Another set of tools that has started to be applied during the last decades to systems modelling and control theory on several fields, including robotics, comes from the theory of fractional calculus. It has become a branch of mathematical analysis which deals with integro–differential operators, allowing integrals and derivatives of any positive real order. Even though the first monograph describing these tools was published by Oldham and Spanier (1974), the topic is considered to have started from a discussion between G. W. Leibniz and the Marquis de L'Hospital in 1695 (Debnath and Bhatta, 2014). This class of techniques has gained remarkable popularity due to its demonstrated applications in a wide range of fields of science and engineering. It has been used on problems where variations on the dynamics, as a consequence of changes in the payload carried by a robot, invalidate the models and controllers designed, decreasing the closed–loop positioning performance (Monje et al., 2010).

Manipulators play an important role as the most relevant mechanisms for effectively interact with the environment. The ability to grab, move, touch or *sense* objects and processes using a manipulator is of great importance for several types of robots, such as exploration rovers, unmanned rescue vehicles, humanoid and anthropomorphic robots or spacial and submarine vehicles, to name a few. On general purpose manipulators, solving kinematics problems in real–time is challenging when the mechanism can be only described as a nonlinear system. This imposes several limitations on the methods that can be used to yield accurate solutions for these problems, since approximate methods and assumptions are typically applied, render the solutions non accurate enough or incomplete.

Considering the above, this dissertation presents different contributions that cover the application of nonlinear models obtained both from data and using nonlinear frameworks and tools.

1.1 Problem statement

Dynamic state estimation problems in mobile robots are tackled by the application of nonlinear models and frameworks. These models are tailored for a wide variety of contexts, such as kinematics, dynamic control, localization, path planning and obstacle avoidance, to name a few. While these methodologies try to capture the complex dynamics that affects both internal and external states, several difficulties arise depending on the chosen approach. Complex dynamics present challenging behaviours which are not straightforward to model, requiring different nonlinear models than classic ordinal approaches.

For instance, some nonlinear formulations are very difficult to be used to obtain closed–form solutions for generalised morphologies. Furthermore, other methods are very difficult to implement or are unable to meet real–time performance, requiring complex representations for general purpose applications. Numerical methods are sometimes used to yield approximate solutions. However, they tend to have convergence problems, they are not general enough and they might not yield all solutions for a given problem, e.g., on kinematics problems. Aside from that, simplifications and assumptions add modelling noise, affecting accuracy and suitability of such models and methods.

The interaction between a mobile robot and its environment is considered as a coupled dynamical systems. In general, robots are subject to uncertainties affecting their internal states and the interaction with external processes and events. To deal with these effects, stochastic approaches are employed, to take into account the uncertainties affecting sensors and actuators. Dealing with sensors of different nature is challenging, requiring to describe suitable models that take advantage of the information extracted from the environment and the state of the robot.

1.2 Research goals

Based on these observations, this dissertation addresses the following theses:

- Machine learning tools can be applied to model the kinematics of a manipulator, yielding accurate estimations on a given parameter space and meeting real–time requirements.
- When used for modelling complex nonlinear systems such as kinematics on manipulators, machine learning methods can be used to yield all solutions in contrast to numerical methods.
- A fractional–order controller of order $\lambda > 1$ is suitable for solving the position tracking problem, even with a reduced stability region.
- Fractional–order incommensurate hyperchaotic systems can be stabilized with a simple located feedback and a full–pseudo–state–feedback controller.

1.3 Originality and contributions

The following four contributions can be identified as the main originalities of this thesis:

- Application of a machine learning method with a spatial decomposition approach to model the behaviour of parallel and serial manipulators on each region of the pose space.
- Proposal of a fractional–order PI^{λ} controller to solve the position tracking problem on a self–driving electric car, with the definition of a suitable design method for a fractional–order $1 < \lambda < 2$.

- Introduction of a new incommensurate hyperchaotic fractional–order system, controlled with a located feedback and full–pseudo–state–feedback controller that can stabilize any unstable fixed point.
- Contributions to the development of the control, localization and navigation systems of the self–driving autonomous vehicle Verdino.

1.4 Methodology

An analysis of the different fields and related publications has been performed to support the claims described on this thesis. It has also served to select proper references and comparable approaches, for achieving a better understanding of the novelty and suitability of those claims. Reliable numerical simulations have been performed for all proposed methods and contributions. Results have been validated using similar methods and compared with known results, when they were available.

Some of the contributions presented in this thesis are related with an experimental platform developed as a joint collaboration with the Robotics Group of *Universidad de La Laguna* (GRULL), a self–driving electric vehicle —Verdino. It is built on top of a low–cost EZ–GO TXT–2 golf cart, an Ackerman platform, as shown in figure 1.1. This prototype has been designed for passenger transportation and surveillance in outdoor unstructured environments. The vehicle has been modified with several sensors and actuators, allowing it to perform autonomous navigation tasks through urban areas. The localization and navigation system is robust enough to allow dynamic obstacle avoidance and to estimate its position in GPS–denied environments. The modifications include a speed controller commanded by an on–board computer. A low–level controller for both steering wheel and Direct Current (DC) motor allows sending motion commands following kinematics based on an Acker-

Figure 1.1 Verdino, experimental platform based on a self–driving car.

man platform. It is also equipped with sensors and extra actuators in order to assist the top–level software in many tasks, including trajectory tracking.

Verdino is equipped with two frontal and one rear Sick LMS111 Light Detection and Ranging (LiDAR), as horizontal depth sensors for localization; a Microsoft Kinect sensor in the front, yielding 3–D point clouds of the near front; a differential Global Positioning System (GPS) Javad Triumph–1; a Xsens MTi Inertial Measurement Unit (IMU) which aids during the estimation of the vehicle's orientation; a mechanical odometry system of the rear wheels; and two frontal high resolution cameras for obstacle avoidance. Some of the contributions of this thesis are demonstrated with this vehicle.

1.5 Thesis outline

This thesis is presented as a compendium of publications. The dissertation is comprised of 5 chapters, presenting the background of the problems addressed and their relationship with the research topic.

Chapter 1 is the introduction of this dissertation, presenting the motivation and the goals of the research. It introduces the topic of state estimation form the different perspectives tackled on the thesis. Considering these, the problem is highlighted and the research goals are described. Then, a summary of the contributions is presented, followed by a brief description of the methodology and the experimental platform developed, the self–driving car Verdino.

Chapter 2 presents the stochastic background of the thesis. The paradigm of probabilistic robotics is described, with emphasis in the stochastic nature of a mobile robot and its environment, as a coupled dynamic system. It also describes the global localization problem and the sequential Monte Carlo algorithm, as the foundation of the localization system developed on Verdino. The chapter is concluded with the introduction of the heterogeneous sensor fusion approach developed.

Chapter 3 briefly introduces the machine learning field and the tools relevant to the thesis, describing support vector methods and how they can be used for state estimation. Then, the serial–parallel duality of the kinematics problems is presented, describing a spacial decomposition method that solves both the forward kinematics problem in parallel robots and the inverse kinematics problem in serial

robots using support vector machines. Three contributions are presented, describing the application of the proposed methodology to parallel and serial manipulators. The presented approach yield approximate but accurate solutions with simple and fast operations, suitable to real–time applications. For each contribution, several experiments are performed which validate the approach.

Chapter 4 starts with the description of the foundations of the fractional calculus and its application, followed by a summary of relevant fractional operators. The position tracking problem is described, focusing on the definition of the problem when the input is a ramp function. In this case, a integer order PI controller is demonstrated to be not suitable for solving this problem. In contrast, the proposed fractional–order PI^{λ} controller with $\lambda > 1$ demonstrates to yield a zero steady–state error. A suitable design method is described, which is applied to control the speed of the DC motor of Verdino. Stability and robustness conditions are also verified theoretically and experimentally. The attention is turned towards the application of fractional calculus to model a novel fractional–order hyperchaotic system. Chaos theory is introduced, with emphasis on its relation to robots in general and mobile robots in particular. The application of both analysis and synthesis of chaotic behaviours is discussed. Then, contribution is presented, where a simple located feedback and a full–pseudo–state–feedback controller is applied to the incommensurate system introduced.

Chapter 5 concludes with a summary of the observations, findings and contributions of this thesis, discussing about further research fields and application which could be opened to extend the presented results.

"You'd be amazed how much research you can get done when you have no life whatsoever."

— Ernest Cline, Ready Player One

2

Stochastic Processes and Nonlinear Dynamics

FROM the perspective of a robot, real–world environments can be highly unpredictable, including stochastic processes in the general case. People doing unexpected movements, objects not perfectly placed, constructive and mechanical imperfections, simplifications and assumptions, noisy sensors and actuators, are some examples of uncertainties and uncontrolled events that may affect any model or estimation method applied on a real robot. In general, unstructured environments —in contrast with well–structured ones, such as an industrial plant — are highly dynamic, like an urban area or a private house.

Furthermore, a robot also influences its environment when it executes *control actions* —through its actuators —, often introducing more uncertainty. A mobile robot performing any task in an unstructured environment will certainly fail at some extent if it is not designed to cope with the uncertainty that stochastic processes introduce. Self–awareness of internal and external states —which are affected by uncertainties at

different levels — is needed to properly plan and perform any task.

The performance of any sensor is tightly related with the physical principle it is based on and unwanted deviations due to its physical realization. Both affect their accuracy and sensitivity to noise, adding even more uncertainty into the loop. Actuators, at the same time, suffer from mechanical disturbances and other unwanted effects due to slippery and uneven ground or control noise. Moreover, internal models are always approximations of physical processes. Model and algorithmic noise suppose other relevant sources of uncertainties on any robotics system.

The estimation of internal and external states has real–time requirements when implemented on a mobile robot. This is needed to solve some of the core features on any mobile robot: know its location with respect to an external reference frame and effectively plan and perform navigation tasks to meet certain goals. Active approaches for localization and navigation are designed to work on the move, in contrast to algorithms that require off–line processing. On active localization, algorithms that estimate the position iteratively improve these estimates in real–time while the robot moves and interacts with the environment. Similarly, algorithms designed for active navigation update and improve the yielded path as the robot moves across the navigable area, considering both internal and external states —the robot and observable external features.

Moreover, kinematics and dynamics on mobile robots tend to be highly nonlinear, often requiring complex theoretical models to capture their real behaviour. These models have in rare cases analytic solutions, so approximations, assumptions and simplifications are applied, affecting the resulting estimation. The challenge is to develop methods and techniques to solve these problems achieving real–time performance and accuracy close enough to the real process or system.

In this context, we present in the following a general background for this dissertation: the stochastic framework developed for a self–driving car. Some of the contributions of this compendium have been developed for this mobile robot, along with other contributions also described in this dissertation.

2.1 Probabilistic robotics

Probabilistic robotics is an emerging field based on tools from the Bayesian probability theory. This paradigm has experienced a growing interest during the past 20 years. It builds stochastic frameworks where the interaction between a robot and its environment is modelled as a coupled dynamical system. In other words, the environment is a dynamical system that possesses internal state, from which a mobile robot maintains a *belief*, built from the information captured by its sensors. Many current state–of–the–art systems for robot localization, navigation and mapping are built on top of these techniques, demonstrating outstanding performance (Thrun et al., 2005).

In order to cope with these effects, classical tools designed for the filtering and prediction problem have been applied to robotics, which started with the well–known and celebrated Kalman Filter (KF). Presented in the seminal paper by R. E. Kalman (1960) as a new solution to the *Wiener problem*[1] (Wiener, 1949), the KF has been followed by many extensions and generalizations, improving the original approach for linear systems, such us the Extended Kalman Filter (EKF) (Anderson and Moore, 1979; Einicke and White, 1999), the Unscented Kalman Filter (UKF) (Julier and

[1] The classic filtering problem, as stated by Norbert Wiener, is three–fold: (i) Prediction of random signals; (ii) Separation of random signals from random noise; (iii) Detection of signals of known form (pulses, sinusoids) in the presence of random noise.

Uhlmann, 1997) and more recently, the Cubature Kalman Filter (CKF) (Arasaratnam and Haykin, 2009). The EKF is derived via a linearisation procedure for applications on nonlinear processes with multivariate Gaussian noise. It has been applied to active localization and navigation systems for mobile robots on environments with Global Positioning System (GPS) denials (Lynen et al., 2013), tackling the problem of sensor fusion or sensor–feed integration within state estimation.

However, if the system model is inaccurate or not well known, KF derivations start to have problems and will eventually fail because of excessive drift, for instance. These methods tend to work well only if the position uncertainty is small. Localization systems based on the KF are better suited for position tracking problems, where the linearised model follow a Gaussian probability distribution. In spite of this, they have been found to be very robust to violations of linearity assumptions, mainly due to robust data association, and specifically when the number of landmarks or features grows.

The Bayes filter is a probabilistic framework based on the Bayes rule, where the posterior $p(x \mid z)$ is computed using the inverse *conditional probability* $p(z \mid x)$ along with the *prior* probability $p(x)$, where x and z are values drawn from random variables X and Z, respectively. The solution is represented by a *belief* —the posterior — on a given point in time t, defined as a Probability Density Function (PDF) over the space of all locations. Thus, the *posterior probability distribution* is given by

$$p(x \mid z) = \eta \, p(z \mid x) \, p(x) \tag{2.1}$$

where $\eta = p(z)^{-1}$, a constant for any value x in the posterior $p(x \mid z)$. Probabilistic models for control actions and sensors need to be defined in order to define the stochastic evolution of the system.

Generative model

Future states for a mobile robot can be generated stochastically from the prior state and current knowledge, obtained from sensors and the actions performed. Let $\mathbf{x}_t$ be the state vector that contains the state variables of the *pose* for a mobile robot at time t. For instance, a mobile robot on a planar space is represented by the three–dimensional state vector $\mathbf{x}_t = (x_t, y_t, \theta_t)$, where x_t, y_t and θ_t are the position coordinates and the orientation of the pose with respect to a reference frame, respectively.

Control actions are the commands which the robot performs, defining its interaction with the environment. These are usually modelled as a sequence of control data, with a vector $\mathbf{u}_t$ representing all actions at time t. On a mobile robot, these measurements can be received from a mechanical or *visual* odometric system. Similarly, data from the environment —gathered from the sensors — are available as another vector $\mathbf{z}_t$, for all data corresponding to a specific time t. We assume that data from all sensors are available at the same time.

The Bayes filter is a recursive technique where the state $\mathbf{x}_t$ is calculated from the previously computed state, $\mathbf{x}_{t-1}$. These algorithms are built on top of the *Markov assumption*: the stochastic evolution of future states from $\mathbf{x}_t$ only depends on the conditions and variables at time t. A temporal process that meets this condition is know as a *Markov chain*.

Thus, the solution to the filtering problem is the posterior or belief of the pose at time t, defined as

$$bel(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{z}_t, \mathbf{u}_t) \tag{2.2}$$

It is convenient to define the belief just before adding the latest update from the sensors — considering $\mathbf{z}_{t-1}$ instead — to make sure that control actions $\mathbf{u}_t$ are executed first.

Then, this additional posterior will be given by

$$\overline{bel}(\mathbf{x}_t) = p(\mathbf{x}_t \mid \mathbf{z}_{t-1}, \mathbf{u}_t) \tag{2.3}$$

which is said to represent the *prediction* step at time t before incorporating the measurement. Obtaining $bel(\mathbf{x}_t)$ from $\overline{bel}(\mathbf{x}_t)$ is called *correction* or *measurement update*, as the second step following probabilistic filtering theory.

These are the core steps performed by methods based on the Bayesian filtering theory, such as *particle filters*.

2.2 Particle filters and the global localization problem

Particle Filter (PF) methods solve state estimation problems based on the Bayes filter. The term was coined by (Del Moral, 1996) in reference to mean field interacting particle methods used in fluid mechanics. Liu and Chen (1998) proposed a general framework for on–line Monte Carlo methods for dynamic systems, setting the foundations and unifying the application to *hidden Markov chains*.

The first application to a computer vision problem was presented by Isard and Blake (1998), where the object tracking problem on real–time is solved using an algorithm named CONDENSATION, from a derived technique called *CONditional DENsity propagATION*. This technique was adapted and applied for the first time to solve the localization problem on a mobile robot by Dellaert et al. (1999), as one of the first applications of probabilistic robotics.

The global localization problem

In contrast with the position tracking problem, the solution to the global localization problem is an estimate of the position of a mobile robot considering a specific search space, like a prior map. The solution depends on the knowledge available initially and at run–time. In general, the estimate can be obtained without any prior knowledge about its position, only based on the map —built off–line or on–line — and the information from its sensors.

The initial knowledge is typically a map —$\mathcal{M}$ —, from which the localization system extracts a set of features and compares them with a set of observations $\mathbf{z}_t = \{z_t^1, z_t^2, \dots\}$, obtained from its sensors, where z_t^i is the i^{th} sensor. One of the most popular approaches for representing a localization map is by the definition of an *occupancy grid*, where a set of one or more possible values —representing levels of occupancy — is assigned to discrete locations. A map $\mathcal{M}$ can be composed by an arbitrary number of dimensions, depending on the application.

In general, on a localization framework based on Particle filters, this knowledge is used to compute the posterior probability distribution of the belief, given by

$$bel(\mathbf{x}_{1:t}) = p(\mathbf{x}_{1:t} \mid \mathbf{x}_{1:t-1}, \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}, \mathcal{M}) \tag{2.4}$$

where $\mathbf{x}_{1:t} = \mathbf{x}_1, \dots, \mathbf{x}_t$ is the complete trajectory of the mobile robot, with respect to an external reference frame.

The basic Bayesian approximation is shown on algorithm 2.1, with the steps for the probabilistic solution of the Markov localization problem. PF algorithms are tractable approximations of the Bayes filter. Efficient implementations have affordable computational complexity and accuracy of the approximations to complex nonlinear problems, like the

Algorithm 2.1 Markov localization

Function *markov_localization*($bel(\mathbf{x}_{t-1})$, $\mathbf{u}_t$, $\mathbf{z}_t$, $\mathcal{M}$) **is**
  for all $\mathbf{x}_t$ **do**
    $\overline{bel}(\mathbf{x}_t) = \int p(\mathbf{x}_t \mid \mathbf{u}_t, \mathbf{x}_{t-1}, \mathcal{M})\ bel(\mathbf{x}_{t-1})\ d\mathbf{x}_{t-1}$
    $bel(\mathbf{x}_t) = \eta\ p(\mathbf{z}_t \mid \mathbf{x}_t, \mathcal{M})\ \overline{bel}(\mathbf{x}_t)$
  return $bel(\mathbf{x}_t)$

global localization. The sequential implementation of this problem is called Monte Carlo Localization (MCL), a method that solves the Bayes *importance sampling*.

2.2.1 Monte Carlo Localization (MCL)

The MCL algorithm is based on an approximation of the PDF that represents the posterior, by means of a set of *particles*. These particles are the stochastic evolution of the state of the mobile robot at a time t, based on its previous state and knowledge. The sequential version of the Bayes filter, the Sequential Importance Sampling (SIS), introduces a probability distribution known as the *importance function* (Doucet, 1998), $\boldsymbol{\pi}(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1})$. This importance function depends on the observations $\mathbf{z}_{1:t}$, yielding a unit–less weight or score for each particle.

Unfortunately, as particles evolve, the SIS algorithm suffers from a degeneracy problem in the steady–state: all but one of the normalized *importance weights* are very close to zero. In other words, particles which are not representative or close enough to the real state of the mobile robot have a weight close to zero. In order to cope with this degeneracy problem, *resampling* methods are introduced. Resampling techniques eliminate trajectories with weak normalized importance weights and multiply those closer to the real state.

The sequential algorithm

The most popular resampling scheme used to solve this problem is the Sampling Importance Resampling (SIR) algorithm (Rubin, 1988; Smith and Gelfand, 1992). It is based on the following steps:

1. *Prediction step (sampling):* On each sampling step, a new set of k particles $\{\mathbf{x}_t\}_k$ is drawn from a proposal distribution $\boldsymbol{\pi}$, described by a probabilistic motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1})$, based on the previous generation of n particles $\{\mathbf{x}_{t-1}\}_n$

2. *Importance weighting:* Then, importance weights are defined as follows (Doucet, 1998):

$$w_t^{(i)} = \frac{p(\mathbf{x}_{1:t}^{(i)} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1})}{\boldsymbol{\pi}(\mathbf{x}_{1:t}^{(i)} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1})} \tag{2.5}$$

3. *Resampling:* Each particle survives with a probability proportional to the likelihood of the observations relative to the map. This step allows maintaining a discrete set of particles which approximate a continuous distribution.

Applied to MCL, the next generation of samples —on the sampling step — is typically drawn from a multivariate Gaussian approximation $\mathcal{N}$, as follows

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}) \simeq \mathcal{N}(\boldsymbol{\mu}_t^{(i)}, \boldsymbol{\Sigma}_t^{(i)}) \tag{2.6}$$

where $\boldsymbol{\mu}_t^{(i)}$ is the mean vector and $\boldsymbol{\Sigma}_t^{(i)}$ is the covariance matrix, both with respect to the particle i. These parameters are defined as part of the motion model of a mobile robot. It is assumed that all continuous random variables are measurable and all continuous distributions possess densities.

Grisetti et al. (2005, 2007) observed that almost in all cases the motion model $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1})$ has a limited number of maxima, mostly a single one, within an interval

$$L^{(i)} = \left\{ \mathbf{x} \;\middle|\; p(\mathbf{z}_t \mid \mathcal{M}, \mathbf{x}) > \epsilon \right\} \tag{2.7}$$

This allows computing less sample positions since the surrounding area of this interval is small.

For each particle i, the parameters $\boldsymbol{\mu}_t^{(i)}$ and $\boldsymbol{\Sigma}_t^{(i)}$ of the Gaussian approximation $\mathcal{N}$ are computed from K sampled points in the interval given by equation 2.7. Those parameters are:

$$\boldsymbol{\mu}_t^{(i)} = \frac{1}{\boldsymbol{\eta}^{(i)}} \sum_{j=1}^{K} p(\mathbf{z}_t \mid \mathcal{M}, \mathbf{x}_j) p(\mathbf{x}_j \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}) \mathbf{x}_j \tag{2.8}$$

$$\boldsymbol{\Sigma}_t^{(i)} = \frac{1}{\boldsymbol{\eta}^{(i)}} \sum_{j=1}^{K} p(\mathbf{z}_t \mid \mathcal{M}, \mathbf{x}_j) p(\mathbf{x}_j \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}) \cdot (\mathbf{x}_j - \mu_t^{(i)})(\mathbf{x}_j - \mu_t^{(i)})^{\mathsf{T}} \tag{2.9}$$

where $\boldsymbol{\eta}$ is a normalization factor given by

$$\boldsymbol{\eta}^{(i)} = \sum_{j=1}^{K} p(\mathbf{z}_t \mid \mathcal{M}, \mathbf{x}_j) p(\mathbf{x}_j \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}) \tag{2.10}$$

This leads to a closed form approximation of the optimal proposal distribution.

The importance weights can be computed according to the observation model $p(\mathbf{z}_t \mid \mathcal{M}, \mathbf{x}_t)$, replacing $\boldsymbol{\pi}$ by the motion model that propa-

gates each particle i, given by $p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1})$, as follows

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \cdot p(\mathbf{z}_t \mid \mathcal{M}, \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}) \\ &\simeq w_{t-1}^{(i)} \cdot \sum_{j=1}^{K} p(\mathbf{z}_t \mid \mathcal{M}, \mathbf{x}_j) p(\mathbf{x}_j \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1}) \\ &= w_{t-1}^{(i)} \cdot \boldsymbol{\eta}^{(i)} \end{aligned} \tag{2.11}$$

where $\boldsymbol{\eta}$ is the same as in equation 2.10.

Finally, a resampling procedure is applied to the set of particles, in order to discard those with a lower likelihood (Hahnel et al., 2003). The performance of the algorithm is greatly affected by both the number of particles and the frequency at which the resampling step happens. The *effective sample size* N_{eff}, introduced by Liu (1996), estimates the quality of a sample set $\mathcal{S}_t$ as a representation of the posterior probability, at a certain time step. It can be obtained as (Doucet et al., 2001)

$$N_{\text{eff}} = \frac{1}{\sum_{i=1}^{|\mathcal{S}_t|} \left(\tilde{w}_t^{(i)}\right)^2} \tag{2.12}$$

where $|\mathcal{S}_t|$ is the cardinality of the current sample set, and $\tilde{w}_t^{(i)}$ is the normalized weight of the i^{th} particle at time t.

These steps are the foundation of a wide class of algorithms based on PFs, following the core idea of recursive state estimation from sensor data. A summary of the sequential Monte Carlo (MC) method for localization is shown on algorithm 2.2. The result is a new particle set after applying the prediction and measurement update steps, resampling the set if the quality of the posterior obtained drops according to equation 2.12. Then, the new position estimate can be determined according to the particle with the maximum weight in $\mathcal{S}_t$.

Algorithm 2.2 MC sequential algorithm for localization with sensor fusion

Data: Latest computed set of particles $\mathcal{S}_{t-1}$, current control actions $\mathbf{u}_t$, measurements $\mathbf{z}_t$ from sensors and the map $\mathcal{M}$.

Result: Updated set of particles $\mathcal{S}_t$, weighted according to the current state of the mobile robot, resampled if needed.

$T_{\text{res}} =$ resampling threshold // controls resampling rate
if $\|\mathbf{u}_t\| \simeq 0$ **then** // the robot has not moved
  $\mathcal{S}_t \leftarrow \mathcal{S}_{t-1}$
else
  $\mathcal{S}_t \leftarrow \emptyset$
  for all $\left\langle \mathbf{x}_{t-1}^{(i)}, w_{t-1}^{(i)} \right\rangle \in \mathcal{S}_{t-1}$ **do**
    $\bar{\mathbf{x}}_t \leftarrow p(\mathbf{x}_t \mid \mathbf{x}_{t-1}^{(i)}, \mathbf{u}_{t-1})$ // particle sampling
    $w_t^{(i)} \leftarrow \prod_{j}^{|\mathbf{z}_t|} p(\mathbf{z}_t^{(j)} \mid \bar{\mathbf{x}}_t, \mathcal{M})$ // importance weighting
    $\mathcal{S}_t \leftarrow \mathcal{S}_t \cup \left\{ \left\langle \mathbf{x}_t^{(i)}, w_t^{(i)} \right\rangle \right\}$ // update weighted particles
  if $N_{\text{eff}}(\mathcal{S}_t) < |\mathcal{S}_t| \cdot T_{\text{res}}$ **then** // quality of particle set
    $\bar{w} \leftarrow \frac{1}{|\mathcal{S}_t|} \sum_{i=1}^{|\mathcal{S}_t|} w_t^{(i)}$ // normalise weights
    $\dot{w} \leftarrow$ **random** $\in [0, \bar{w}]$ // naive discrete event sampler
    $\mathcal{S}_{\text{temp}} \leftarrow \emptyset$
    for $i \in \{1, \cdots, |\mathcal{S}_t|\}$ **do** // start resampling
      while $\sum_{j=1}^{i} w_t^{(j)} > \dot{w}$ **do**
        $\mathcal{S}_{\text{temp}} \leftarrow \mathcal{S}_{\text{temp}} \cup \left\{ \left\langle w_t^{(i)} \right\rangle \right\}$ // add sample
        $\dot{w} \leftarrow \dot{w} + \bar{w}$ // update weight
    $\mathcal{S}_t \leftarrow \mathcal{S}_{\text{temp}}$

Extension for simultaneous mapping

The Simultaneous Localization And Mapping (SLAM) problem is a directly related application of Bayes filters, where the map is recursively built with a sequence of pose estimates. In this context, the simultaneous problem has been solved applying both Gaussian (Martinez-Cantin and Castellanos, 2005; Paz et al., 2007; Shojaie and Shahri, 2008) and MC (Grisetti et al., 2005, 2007; Hahnel et al., 2003) approaches. To solve this problem, the joint posterior obtained is $p(\mathbf{x}_{1:t}, \mathcal{M} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1})$ about both the map $\mathcal{M}$ and trajectory $\mathbf{x}_{1:t} = x_1, \ldots, x_t$.

A Rao–Blackwellized Particle Filter (RBPF) (Doucet et al., 2000; Murphy, 1999; Mustière et al., 2006) is built in order to solve this new posterior, applying the following factorization

$$p(\mathbf{x}_{1:t}, \mathcal{M} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) = p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1}) \cdot p(\mathcal{M} \mid \mathbf{x}_{1:t}, \mathbf{z}_{1:t}) \tag{2.13}$$

obtained from both estimates.

The term $p(\mathbf{x}_{1:t} \mid \mathbf{z}_{1:t}, \mathbf{u}_{1:t-1})$ can be estimated applying a Bayesian PF, where each particle maintains its own map, since the trajectory is stored as a sequence of robot poses. The second term can be computed efficiently since the position over maps can be obtained analytically (Moravec, 1988), i.e., poses $\mathbf{x}_{1:t}$ and observations $\mathbf{z}_{1:t}$ are known.

The factorization is based on the *Rao–Blackwell* theorem (Blackwell, 1947), a technique that can be applied to a PF in order to obtain an optimal estimator, obtaining a sampling distribution closer to the target distribution (Liu and Chen, 1998).

In order to solve the simultaneous problem when creating a map, the implementation of these algorithms handle partial maps for each particle, that need to be integrated as new and reliable data is retrieved.

2.2.2 Heterogeneous sensor fusion

If the noise that affects the sensors can be assumed to be statistically independent, *conditional independence* from Bayes theory can be applied. Thus, on a probabilistic approach, this means that, for each particle, measurements from each sensor $\mathbf{z}_t = \{z_t^{(i)}\}$ can be added during the weighting step to equation 2.8 as follows

$$p(\mathbf{z}_t \mid \mathbf{x}_t, \mathcal{M}) = \prod_i p(z_t^{(i)} \mid \mathbf{x}_t, \mathcal{M}) \tag{2.14}$$

This assumption allows fusing information from the environment gathered from sensors of different nature, such as depth sensors —e.g. Light Detection and Ranging (LiDAR) —, absolute orientation —e.g. Inertial Measurement Unit (IMU) — or absolute position —e.g. GPS sensor —, along with the information from an odometric system, which is taken into account during the propagation step.

However, when fusing data from sensors subject to unpredictable errors, such as a GPS sensor, other considerations have to be taken into account. This kind of sensors yields absolute measurements that, while very accurate in the general case, are prone to noticeable errors due to different reasons, such as atmospheric instability, occlusions or temporary disruptions on the satellite constellation, to name a few. GPS sensors are often modelled with a Gaussian PDF following equation 2.6. During these events, the covariance matrix $\mathbf{\Sigma}_t$ —calculated or received from the sensor — if properly obtained, may help filtering them during the weighting step. These effects have to be taken into account when adding information from GPS sensors to MCL algorithms, to reduce the negative impact on the accuracy of the resulting implementation. Thus, robust integration of these measurements is challenging.

Assuming conditional independence on position and orientation, the

likelihood of a GPS measurement $\mathbf{z}_t^{\text{GPS}}$, subject to a pose $\mathbf{x}_t^{(i)}$, can be obtained as the product of

$$p(\mathbf{z}_t^{\text{GPS}} \mid \mathbf{x}_t^{(i)}) = p(\langle \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t \rangle_{x,y} \mid \mathbf{x}_t^{(i)}) \cdot p(\langle \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t \rangle_{\theta} \mid \mathbf{x}_t^{(i)}) \tag{2.15}$$

where the PDF for the position —assuming a 2–D case — is

$$p(\langle \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t \rangle_{x,y} \mid \mathbf{x}_t^{(i)}) = \frac{1}{2\pi \left|\boldsymbol{\Sigma}\right|^{1/2}} \exp\left\{ -\frac{1}{2} \left(\begin{bmatrix} x - \mu_x \\ y - \mu_y \end{bmatrix}^{\intercal} \boldsymbol{\Sigma}^{-1} \begin{bmatrix} x - \mu_x \\ y - \mu_y \end{bmatrix} \right) \right\} \tag{2.16}$$

and the PDF corresponding to the orientation angle θ, which follows a wrapped normal distribution, is

$$p(\langle \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t \rangle_{\theta} \mid \mathbf{x}_t^{(i)}) = \frac{1}{2\pi \left|\boldsymbol{\Sigma}\right|^{1/2}} \sum_{k=-\infty}^{\infty} \exp\left\{ -\frac{1}{2\sigma_\theta^2} (\theta - \mu_\theta + 2\pi k)^2 \right\} \tag{2.17}$$

Some GPS devices can provide the orientation as a *course–over–ground* measurement. When this is not the case, it can be approximated based on the latest position measurements, along with its covariance. If the measurement is not reliable —wide covariances, for instance —, IMU sensors can be used as an alternative source to obtain an estimate of the orientation. These sensors can provide an absolute measurement, typically fusing —with KF based approaches in general — the earth magnetic field vector and measurements from gyroscopic and accelerometer sensors. While these sensors yield good estimates in general, they are specially sensible to magnetic fields that distort the measurement. Thus, when adding absolute position measurements, special care is needed in order to reduce the impact of unreliable measurements.

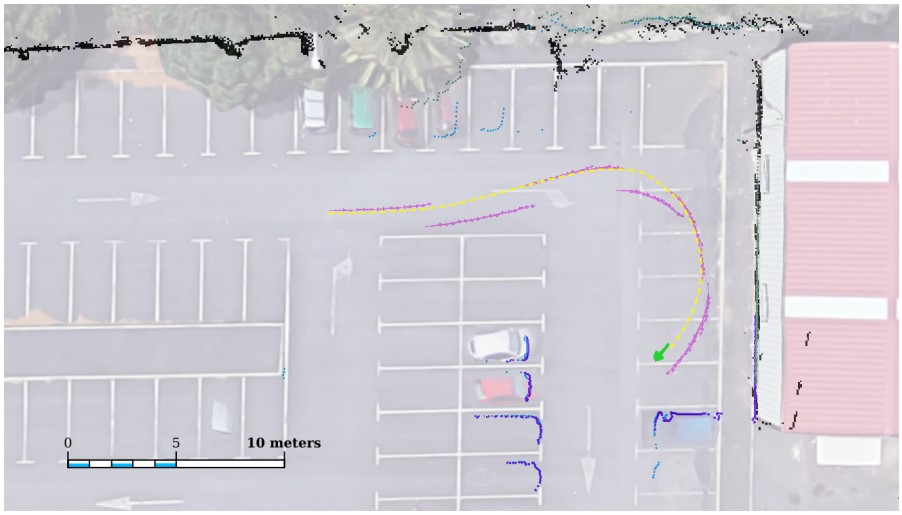
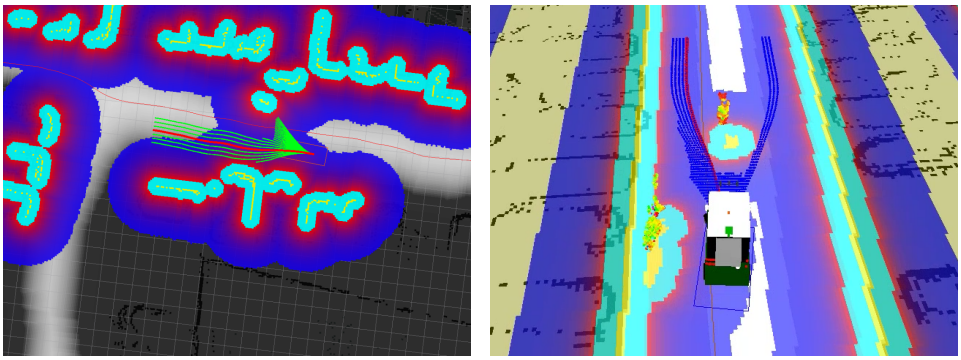


Figure 2.1 Localization with particle filters. Green arrows: set of particles from the localization hypotheses estimated by the PF; magenta arrows: GPS readings; green, blue and purple dots: features from LiDAR scans; yellow dashed line: ground truth of the trajectory; black: obstacles represented in the map.

2.3 Results and discussion

The techniques described in this chapter are the foundation of the localization and navigation systems implemented for the self–driving car Verdino. The contributions described here have only been possible with the invaluable collaboration from all current and past members of the Robotics Group of *Universidad de La Laguna* (GRULL), specially Dr. Daniel Perea, Dr. Jonay Toledo, Dr. Néstor Morales, Dr. Rafael Arnay, Dr. Javier Hernández–Aceituno, Dr. Alberto Hamilton and Prof. Leopoldo Acosta, with the additional support of the Biped Humanoid Group of Waseda University, specially Yukitoshi Minami Shiguematsu, Dr. Kenji Hashimoto and Prof. Atsuo Takanishi. The related contributions are included as the Part II of this dissertation.

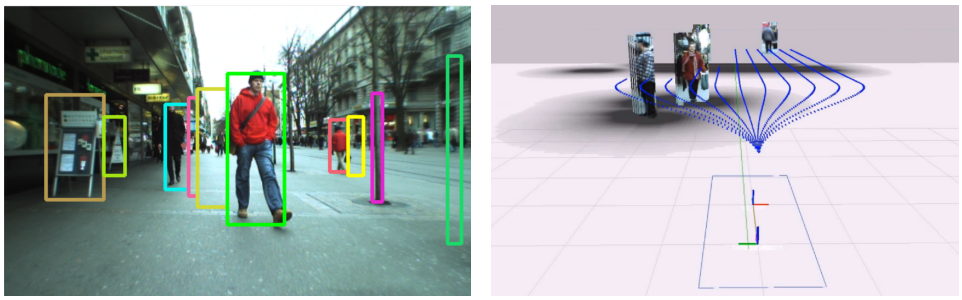


(a) Static costmap **(b)** Dynamic obstacles

Figure 2.2 Costmap based path planning, obstacle detection and avoidance. The map is represented as an occupancy grid, with discrete values representing the likelihood of having an obstacle for each cell. Black: cell with a high occupancy likelihood; white: cell with a low occupancy likelihood; yellow: the highest cost for a grid cell; cyan to navy blue: cost for a grid cell, from highest to lowest; blue and green lines: tentative paths generated by the local planner; thick red lines: the winning path of the local planner, after a weighting process.

Perea et al. (2013) compared the weighting mechanism described in the previous section with an alternative approach where new particles are generated, drawn from the PDF defined for each GPS and IMU measurement during the prediction step. We demonstrated that the weighting mechanism is better suited in some scenarios where adding particles around a wrong absolute measurement can lead to localization errors if measurements from other sensors are poor or misleading. Figure 2.1 shows an experiment with the implemented localization system, where a multipath event occurred. In this experiment, the particle filter is able to reject the noisy measurements from the GPS, keeping a narrow localization hypothesis thanks to additional measurements from the LiDARs, IMU and wheeled mechanical odometry.

Figure 2.2 shows two examples of navigation using the local and

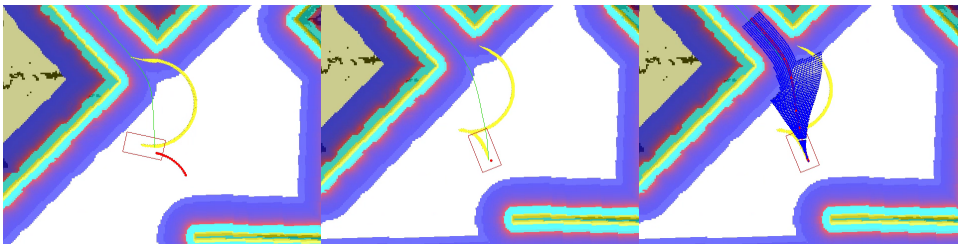


(a) Stixels extracted **(b)** Avoidance of detected stixels

Figure 2.3 Dynamic obstacle detection and avoidance with stixels. The system implemented in Verdino adds the costs due to detection of obstacles as stixels to the layered costmap.

global path planners implemented in Verdino. Morales et al. (2016a) and Arnay et al. (2016) described the navigation system, based on the generation of costmaps for path planning and obstacle avoidance. We developed a navigation system which takes advantage of the paradigm of layered costmaps, where each subsystem for obstacle detection yield the proper cost for each discrete partition of the environment. Then, a primitive–based global planner constructs a path starting from the position of the vehicle —obtained by applying the approach described by Perea et al. (2013) — to a desired goal. The path is generated using short primitives, representing feasible manoeuvres according to the kinematic model of the mobile robot (Espelosín et al., 2013).

The integration of an additional system for obstacle extraction as presented by Morales et al. (2016b) allows for a better prediction and avoidance of dynamic obstacles. These obstacles are detected with the vision system and added to the layered costmap as a simplified representation of the obstacles as vertical instances called *stixels*. They require modest computational effort compared to other techniques, reducing the resolution of the yielded obstacles. This vision–based obstacle detection



(a) Backwards 1 **(b)** Backwards 2 **(c)** New path

Figure 2.4 Recovery manoeuvres. Backward manoeuvres can be chained. Yellow line: traversed path; red line: winning path

system is also implemented in Verdino, allowing for a richer representation of the surrounding costmap, as can be seen on figure 2.3. Additional obstacles which could be located on the navigable frontal area of the vehicle are detected using a detection system based on Microsoft Kinect (Hernández-Aceituno et al., 2016), a time of flight sensor. These obstacles are also included in the layered costmap.

The navigation system also includes recovery behaviours, performed when motion is impossible due to kinematic restrictions, if no feasible path plan exists. These manoeuvres, as shown in figure 2.4, allow the vehicle to recover from challenging positions, performing forward or backward movements. The navigation pipeline is completely described in appendix C.

Humanoid robots are specially complex dynamical systems, with challenging actuation and sensory systems. In the context of the research plan of this thesis, we collaborated developing a sensorimotor system based on that of humans, towards a study of its effects on walking stabilization (Minami Shiguematsu et al., 2015). This collaboration was performed during a research internship in the Biped Humanoid Group[2]

[2]Takanishi Laboratory, Graduate School of Advanced Science and Engineering, Waseda University, Tokyo, Japan.

of Waseda University. We applied a stochastic sensor fusion approach based on the EKF (Lynen et al., 2013), for real–time state estimation of loosely coupled sensors. These include mechanical odometry, a visual odometry system and a IMU sensor. This sensory system will be used for developing a robust online walking pattern generator, taking into account the dynamics of the robot and the information yield by the proposed sensor fusion system. A description of the sensor system can be seen in appendix B.

"The purpose of a storyteller is not to tell you how to think, but to give you questions to think upon."

— Brandon Sanderson, The Way of Kings

3

Machine Learning and Nonlinear Systems

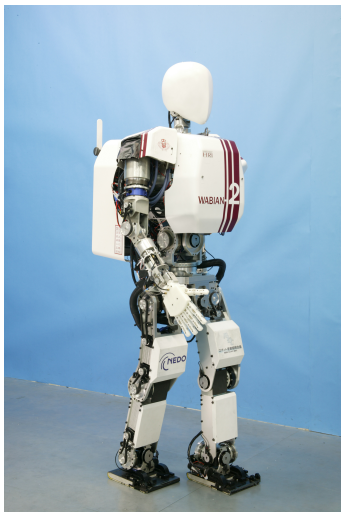
HUMANOID and walking robots are examples in which serial and parallel mechanisms are employed as manipulators and legs. These kinematic chains play a relevant role when these kind of mobile robots perform complex tasks, such as walking or interacting with human beings and the environment. Figure 3.1 show two examples of both serial and parallel kinematics chains used for building anthropomorphic arms and legs for two biped robots, WABIAN (Narang et al., 2013) and the Waseda Leg WL–16RV biped walker (Hashimoto et al., 2008).

Kinematics of generalised morphologies for serial and parallel manipulators are challenging, requiring in the most general case the applications of methods that yield approximate solutions, not always suitable for real–time applications. In this context, this chapter presents three contributions for two well–known problems related to manipulators and kinematic chains: the *Direct* or Forward Kinematics (FK) problem in par-

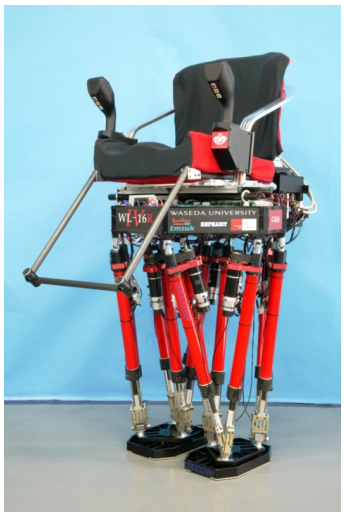
allel manipulators and the Inverse Kinematics (IK) problem for the class of serial kinematic chains. These two classic kinematics problems are also known as the serial–parallel duality or *Jacobian symmetry* (Waldron and Hunt, 1991), still attracting the interest of many researchers.

These contributions take advantage of Machine Learning (ML) methods for state estimation problems, creating models from data for solving complex nonlinear kinematics problems. ML has been one of the key contributors to the Artificial Intelligence (AI) surge in the past few decades, becoming one of its relevant sub–areas. It is being applied on a vast variety of problems and fields, ranging from speech recognition to search engines or image understanding. Several relevant applications are related with search and planning, knowledge representation and reasoning, multi–agent systems, machine perception, robotics and human robot interaction, to name a few. It is considered that the field of AI was officially born at a workshop organized by John McCarthy et al. (1956) at Dartmouth College in Hanover, New Hampshire. The workshop consisted on a proposal for a summer research project, intended to describe how every aspect of learning or any other feature of intelligence can be so precisely described that a machine can be made to simulate it.

In recent years, the field of AI has experience a renewed interest not only in the academic field, being social networks and internet search engines some of the most prominent applications. It started appealing again in part due to an increasing number of web–based applications and the popularization of *deep learning* techniques. With the *perceptron* (Rosenblatt, 1957) as one of the first tools developed, deep learning techniques employ a form of adaptive artificial neural network, trained using the *backpropagation* (or backward propagation of errors) method. A recent report from Stanford University (Stone et al., 2016) summarizes and sets a long–term investigation and expert review on this topic, highlighting what potential advancements lie ahead related to the field, as



(a) WABIAN humanoid robot



(b) Waseda biped vehicle

Figure 3.1 Walking robots using kinematic chains, developed at the Takanishi Laboratory, Waseda University, Tokyo, Japan. Left: WABIAN–2R, (WAseda BIpedal humANoid–2 Refined), an anthropomorphic bipedal humanoid (Narang et al., 2013); right: WL–16RV (Waseda Leg–16 Refined V), a biped walking vehicle (Hashimoto et al., 2008).

well as their influence on society.

Developed originally as classification tools, some ML methods have been extended to solve nonlinear problems, yielding in some cases more accurate solutions. Well known examples are Nonlinear Logistic Regression and Support Vector Machine (SVM) Regression. SVMs, introduced in the next section, are one of the most popular ML methods, used some times in combination with other algorithms, such as Decision Trees (Wang et al., 2008), Naive Bayes (Feng et al., 2016) or Logistic Regression (Liu et al., 2017), to name a few.

A brief introduction of the main methods applied in these contributions is presented in the following.

3.1 Support Vector methods for state estimation

In the context of statistical learning, Support Vector (SV) methods have been considered useful tools since the 60's, before their formal introduction on a paper published by Vapnik and his co–workers, Boser et al. (1992). Nowadays, these methods has been widely used in ML and AI in different shapes, originally as a tool based on Supervised Learning (SL). The general term SVM is used to refer to this class of methods. The formalization introduced by Vapnik (1995) initially described these methods as classification tools. Later, Vapnik et al. (1996) extended these methods to solve regression problems. The frameworks introduced here are called Support Vector Classification (SVC) and Support Vector Regression (SVR), respectively. Whereas the former was developed as a pattern recognition tool by Boser et al. (1997), the latter was proposed for estimating regressions, constructing multidimensional splines, solving linear operator equations and developing models for nonlinear processes.

As a statistical learning tool, SVMs have a great ability to generalize, thanks to the Structural Risk Minimization (SRM) principle instead of the traditional Empirical Risk Minimization (ERM), very common in other systems, e.g., neural networks. When solving pattern recognition problems, a SVM is able to find a decision rule with enough generalization by selecting a small subset of the training data, the SV. It can be shown that optimal separation of the SV is equivalent to the optimal separation of the entire data (Vapnik et al., 1996), by means of an optimal separating hyperplane in the feature space. This enables SVM to summarize the information contained in a data set, making them optimal estimators for several classes of problems.

When used as a SL tool for classifying, a training algorithm is defined

as first step. The training data require a set of p pairs $(\mathbf{x}_i, y_i)$, with a sample vector $\mathbf{x}_i \in \Re^n$ and a class label $y_i \in \{-1, 1\}$. Such algorithm finds the parameters of a decision function $D(\mathbf{x})$ using these training samples. The direct space decision function is defined as:

$$D(\mathbf{x}_i) = \langle w, \mathbf{x}_i \rangle + b\,. \tag{3.1}$$

Therefore, the purpose is to find the parameter w and the bias b that define the optimal separating hyperplane. In order to classify an unknown pattern this function is evaluated. The sample will belong to the first class if $D(\mathbf{x}) > 0$ and to the second otherwise.

The formulation can be extended to the *dual space* (or *form*) representation:

$$D(\mathbf{x}) = \sum_{i=1}^{p} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b \tag{3.2}$$

where coefficients α_i are the parameters found by the algorithm and $\mathbf{x}_i$ are the training samples. This mathematical sleight, named the *kernel trick*, allows SV methods to be used to model nonlinear processes, where the problem to be solved is not linearly regressable. Using a kernel function $K(\mathbf{x}_i, \mathbf{x})$ enables operations to be performed in the input space rather than in the feature space, which typically has a higher dimensionality. Furthermore, the only operation that needs to be performed is the dot product, which reduces the computational complexity.

As a regression tool, a SVM employs an alternate loss function, which includes a distance measurement. Vapnik (1995) proposed the *ε–insensitive loss function*, a more sophisticated penalty tool, which defines a region between y_i, the predicted value, and t_i, the actual value. Only when $|t_i - y_i| \geq \varepsilon$ the function add one of two slack variable penalties, depending on whether they lie *above* (ξ^+) or *below* (ξ^-) the region called

the *ε–insensitive tube*. Similarly to equation 3.1, the data to be modelled is a vector $\mathbf{x}_i \in \Re^n$ and a value $y \in \Re$, with a linear function:

$$f(x) = \langle w, \mathbf{x} \rangle + b \tag{3.3}$$

For this class of problems, the optimal regression function is given by the minimum of the functional:

$$\Phi(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_i (\xi_i^- + \xi_i^+) \tag{3.4}$$

with a given value C.

Similarly to the classification problem, a regression model can be defined as a function that can be recast into a higher dimensionality space by means of the same *kernel trick*. These models can easily fit complex nonlinear functions, such as the nonlinear mapping of complex kinematics problems in manipulators. In order to achieve this, the workspace is partitioned into smaller regions. Then, a nonlinear model is obtained by training a SVR using randomly generated data, representing the pose and the related joint configuration. This approach allows to model the behaviour of the manipulator for that particular region. In a latter phase, each regression model is retrieved to predict the value of the forward kinematics on parallel manipulators —or the inverse for a serial one.

3.2 Parallel robots and the forward kinematics problem

This class of robots can be classified as closed–loop mechanisms, in contrast with serial robots, which are usually open–loop kinematic chains. The basic morphology consists of two frames connected by means of

active links, such as cylindrical —C for short, including a non–actuated rotational degree of freedom along the axis — or prismatic actuators, and joins that can be spherical —S for short — or rotational —R for short, a non–actuated rotational degree of freedom —, in general.

The kinematic structure generally has several advantages, such as high stiffness, rigidity, accuracy, and damping. However, kinematic dexterity is usually limited. In spite of that, their good dynamic performance and high load carrying capacity make them very attractive in many applications and fields. Notable examples of parallel manipulators are the *Delta* robot (Clavel, 1988; Pierrot et al., 1990), the *Tricept* (Siciliano, 1999) and the Gough–Stewart Platform (Gough, 1956–1957; Stewart, 1965–1966). The latter has become the most popular morphology, also with different types of configurations, depending on the actuators used and other mechanical constraints. For instance, the general 6–SPS —Spherical–Prismatic–Spherical — is preferred for modelling purposes. The 6–RRCRR —with rotational and cylindrical joints, respectively — is a realistic and challenging configuration due to joint–offsets, used on applications like the one described by Dalvand and Shirinzadeh (2011).

A general 6–SPS configuration is shown in figure 3.2. Both platforms or frames A and B are modelled using 4×6 matrices, containing 3–D Cartesian homogeneous coordinates, corresponding to the points where each joint is connected, and related to coordinates frames xyz and XYZ, respectively. A position of the end–effector can be described as vector on a *pose space*, the 6–D space that represents the position of the frame, as a combination in 3–D Cartesian coordinates of position (x, y, z) and 3–D angular coordinates (α, β, γ) for the orientation. A *pose* is then defined by the vector $\mathbf{p} = (xyz\alpha\beta\gamma)^\intercal$. Similarly, we can define a position in the *link space* as a vector on a 6–D space, containing the length of each linear actuator. A *link vector* represents a position in the link space as a 6–coordinate vector $\mathbf{l} = (l_1 l_2 \cdots l_6)^\intercal$.

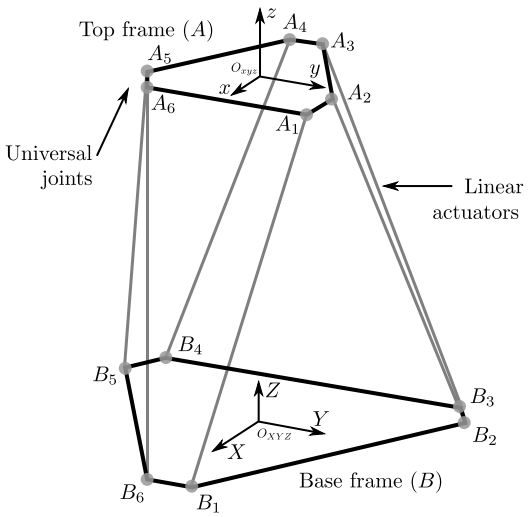


Figure 3.2 The 6–SPS configuration for a Stewart Platform.

Thus, IK can be described by the geometric relationship between both the fixed frame (B) and the end–effector (A). As shown on equation 3.5, the closed–form mathematical solution for the IK problem is the Euclidean distance or L^2–norm between the position where each joint is attached:

$$\begin{aligned} l_i &= \sqrt{\|Ra_i + D - b_i\|} \quad i = 1, 2, \cdots, 6 \\ &= f_i(\mathbf{p}) \end{aligned} \tag{3.5}$$

where a_i and b_i are the i–th row of the matrix that represents the end effector and fixed frame respectively, and D and R are the position vector and rotation matrix of the end–effector.

Similarly, the FK problem can be stated as: find the vector $\mathbf{p}$ that satisfies a given link vector $\mathbf{l}$. Solving this problem is very important for avoiding singular configurations, calibrating the zero configuration of

the mechanism, and evaluating IK solutions.

Closed–form solutions

Similarly to the serial case, analytical solutions are preferred, but for the case of parallel manipulators the problem is harder: for the general case it is an under–defined problem with a high–degree nonlinear formulation, in which the solution in most cases is not unique.

Liu et al. (1993) described an analytical solution for the general forward kinematics problem for a 6–SPS manipulator. Some of the most relevant analytical methods found in the literature are based on the Sylvester's Dialytic elimination method (Lee and Shim, 2001), the Gröbner's bases method (Huang and He, 2009) or combinations thereof (Dhingra et al., 1999). The main principle of such methods is to formulate a symbolic equation or equation set with a single unknown by means of certain elimination or induction techniques.

However, these kind of methods cannot be easily applied to more complex configurations, which also can show constructive constraints. For instance, the 6–RRCRR manipulator analysed by Dalvand and Shirinzadeh (2011, 2012) has offsets in the RR–joints, which makes the forward kinematics solution more complicated, involving a set of highly nonlinear and coupled equation, denoting the kinematics constraints of the parallel kinematics chains: it is described by three sets of equations —a total of 18 equations — with 30 unknown joint variables, as well as 6 unknown pose variables of the end effector, for a total of 36 unknown variables, making the derivation of an analytical solution nearly impossible. For these more complex cases, approximate solutions have to be applied. Numerical methods are prominent, as well as neural networks or polynomial fitting. Some of these approaches, also based on ML techniques, are able to yield approximate solutions suited for real–

time applications.

In this context, preliminary results of a spatial decomposition method for solving the forward kinematics problem on parallel robots are presented in the following section (Morell et al., 2012). This method involves a novel application of SVMs for regression to model the behaviour of the robot on a given partition of the workspace. A more detailed description and discussion of the methodology and alternative methods is also presented. Then, the method is applied to solve the forward kinematics problem on generalised parallel robots (Morell et al., 2013b).

3.3 An artificial intelligence approach to forward kinematics of Stewart platforms

This section includes the full text for the following article, one of the contributions of this thesis presented as a compendium of publications.

- Title: An artificial intelligence approach to forward kinematics of Stewart platforms
- Authors: Antonio Morell, Leopoldo Acosta and Jonay Toledo
- Publication: Proceedings of the 20^{th} IEEE Mediterranean Conference on Control & Automation (MED)
- Year: 2012
- ISBN: 978–1–4673–2531–8
- doi: 10.1109/MED.2012.6265676

2012 20th Mediterranean Conference on Control & Automation (MED)
Barcelona, Spain, July 3-6, 2012

An Artificial Intelligence Approach to Forward Kinematics of Stewart Platforms

Antonio Morell, Leopoldo Acosta and Jonay Toledo

Abstract— **The Stewart Platform, one of the most successful and popular parallel robots, has attracted the attention of many researchers in recent decades. The solution of the forward kinematics problem in real–time is one of the key aspects that continues to garner interest. In this paper we propose a new approach for solving this particular case using Support Vector Machines, a popular Machine Learning method for classification and regression. The algorithm involves a data generation and preprocessing off–line phase, and a fast on–line evaluation. The experiments show that this method is very accurate and suitable for use in real–time.**

Index Terms— **Parallel Robots, Stewart Platform, Forward Kinematics, real–time, Support Vector Machines, Support Vector Regression.**

I. INTRODUCTION

EXTENSIVELY used as manipulators in a wide variety of fields, e.g. medicine, optics, astronomy and many industries like aerospace, automotive and aviation, parallel robots have attracted the attention of many researchers in recent decades. They can be classified as closed–loop mechanisms, in contrast with serial robots, which are usually open–loop kinematics chains. Instead, they consist of two frames connected by means of active links, such as prismatic actuators, sometimes called *legs*. The advantages of this type of structure include a greater dexterity, rigidity and positioning capability, good dynamic performance and high load carrying capacity, which makes them very attractive in many applications and fields.

This paper considers a new approach for solving the forward kinematics of Parallel Robots, one of the relevant topics related to parallel manipulators in general, and to the Stewart Platform in particular [1]. Whereas the inverse kinematics problem has a closed–form mathematical solution, the forward case lacks one. It is an under–defined problem with a high–degree nonlinear formulation in which the solution in most cases is not unique. Although the kinematics is one of the most studied aspects of parallel robots, the forward problem continues to gain interest, especially in terms of those methods that can solve it in real–time, as it is essential for a platform characterization and its closed control loop to know the position and orientation (*pose*) by means of the length of the linear actuators attached to the joints.

A review of the literature shows different methods and strategies. One of the main groups are those that describe an analytical solution [2]. Some authors propose a simplification of the model [3], which is solved with numerical methods like Newton–Raphson, or use an interval analysis [4] to work the solution out. However, these kinds of methods lack generalization, since they are proposed for a particular type of platform. Other methods develop a solution to the forward kinematics problem by directly applying a numerical method [5] like Newton–Raphson. In general these kinds of techniques achieve accurate solutions with enough iterations of the algorithms, but may have convergence problems and high computational requirements. Furthermore, there are other solutions that add rotary type sensors or extra links [6] – [7] to obtain additional information, in order to aid and simplify the algorithms. However, these approaches may be difficult to generalize and in some cases will not be applicable due to structural or mechanical constrains. A second group of techniques are those which obtain approximate solutions using a wide variety of methods, from neural networks [8] to polynomial fitting [9]. These types of strategies are more suited to real–time applications and yield good enough approximations for a wide variety of fields.

In this context, we present a different approach for obtaining approximate but accurate solutions in real–time for the forward kinematics of Generalized Stewart Platforms using *Support Vector Machines* (SVMs). Whereas many of the main concepts and features of the SVMs have been present in Artificial Intelligence and Machine Learning since the 60's, they were first formally introduced in the 1992 paper published by Vapnik and his co–workers [10]. SVMs were developed initially as a classification tool, and a few years later were extended to solve regression problems [11]. More specific terms for both cases are *Support Vector Classification* (SVC) and *Support Vector Regression* (SVR). Whereas the former was developed as a pattern recognition tool [12], the latter was proposed for estimating regressions, constructing multidimensional splines and solving linear operator equations [13]. When solving pattern recognition problems, a SVM is able to find a decision rule with good generalization by selecting a small subset of the training data, called the Support Vectors (SVs). It can be shown that optimal separation of the SVs is equivalent to the optimal

Manuscript received January 30, 2012. The authors gratefully acknowledge the contribution of the Spanish Ministry of Science and Technology under Project SAGENIA DPI2010-18349.

A. Morell is with the *Departamento de Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores (ISAATC), Universidad de La Laguna, La Laguna 38203, Spain* (corresponding author to provide phone: +34–922–318287, e–mail: amorell@isaatc.ull.es

L. Acosta is with the *Departamento de Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores (ISAATC), Universidad de La Laguna, La Laguna 38203, Spain* (e–mail: leo@isaatc.ull.es

J. Toledo is with the *Departamento de Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores (ISAATC), Universidad de La Laguna, La Laguna 38203, Spain* (e–mail: jonay@isaatc.ull.es

978-1-4673-2531-8/12/$31.00 ©2012 IEEE

separation of the entire data [11]. Hence, SVMs can be used to summarize the information contained in a data set.

We have chosen the paper presented by M. Tarokh [9] as a reference to compare our results against the exact analytical solution for a Generalized Stewart Platform presented by Liu et al [2]. This paper is organized as follows. Support Vector Methods and Support Vector Regression are described in Section II. Then, after a detailed description of the proposed procedure presented in Section III, the results are discussed in Section IV. Finally, the main conclusions are summarized in Section V.

II. SUPPORT VECTOR METHODS

A. Support Vector Classification

The training algorithm requires a set of p pairs $(\mathbf{x}_i, y_i)$, with a sample vector $\mathbf{x}_i \in \Re^n$ and a class label $y_i \in \{-1, 1\}$. The algorithm finds the parameters of a decision function $D(\mathbf{x})$ using these training samples. An unknown pattern will belong to the first class if $D(\mathbf{x}) > 0$ and to the second otherwise. The direct space decision function is:

$$D(\mathbf{x}_i) = \langle w, \mathbf{x}_i \rangle + b \quad . \tag{1}$$

The goal of the algorithm is to find the parameter w and the bias b that represents the optimal separating hyperplane.

The decision function has a dual space representation, which reduces the number of computations required to train the classifier [10]:

$$D(\mathbf{x}) = \sum_{i=1}^{p} \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b \quad , \tag{2}$$

where the coefficients α_i are the parameters found by the algorithm and $\mathbf{x}_i$ are the training samples. $K(\mathbf{x}_i, \mathbf{x})$ is the kernel function, which allows operations to be performed in the input space rather than in a high dimensional feature space. The kernel performs the non–linear mapping into the feature space, and can be chosen from among the most commonly employed functions, such as polynomials, radial basis functions and certain sigmoid functions.

The samples are sparse over the input space and we can assume that they are linearly separable. Classes separated with a generalized optimal hyperplane are subject to:

$$y_i[\langle w, \mathbf{x}_i \rangle + b] \geq 1 - \xi_i \quad , \quad i = 1 \ldots, p \quad , \tag{3}$$

where w is normal to the hyperplane and ξ_i is a measure of the misclassification error. The perpendicular distance from the hyperplane to the origin is $\frac{b}{\|w\|}$, and the optimal hyperplane is the one that minimizes,

$$\Phi(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i \quad , \tag{4}$$

which can be solved with Quadratic Programming (QP) optimization. This optimization problem can be transformed by means of the Lagrangian, and the solution is given by the saddle point:

$$\Phi(w, b, \alpha, \xi, \beta) = \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_{j=1}^{p} \beta_j \xi_j - \\ - \sum_{i=1}^{p} \alpha_i (y_i [\langle w, \mathbf{x}_i \rangle + b] - 1 + \xi_i) \quad , \tag{5}$$

where α, β are the Lagrange multipliers. Thanks to the duality of the classical Lagrangian, (5) is minimized with respect to $w, b, \mathbf{x}$ and maximized with respect to α, β. The dual problem is given by:

$$\max_{\alpha} W(\alpha, \beta) = \max_{\alpha, \beta} (\min_{w, b, \xi} \Phi(w, b, \alpha, \xi, \beta)) \quad . \tag{6}$$

Differentiating (5) and setting the derivatives to zero leads to the dual problem formulation,

$$\max_{\alpha} W(\alpha) = \max_{\alpha} -\frac{1}{2} \sum_{i=1}^{p} \sum_{j=1}^{p} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{k=1}^{p} \alpha_k \quad , \tag{7}$$

subject to the constraints,

$$0 \leq \alpha_i \leq C \quad i = 1, \ldots, p \tag{8a}$$

$$\sum_{j=1}^{p} \alpha_j y_j = 0 \quad . \tag{8b}$$

The given value C is a regularization parameter, which must be chosen to reflect the noise in the data [14].

Maximizing the margin between classes and training samples is an alternative to other optimizing cost function methods, e.g., mean squared error. This provides some important features to SVMs, like automatic capacity tuning of the classification function, the representation of the data relevant for the classification by a small subset (data compression), and uniqueness of the solution.

B. Support Vector Regression

As a regression tool, a SVR employs an alternative loss function, which includes a distance measure. Vapnik proposed the *ε–insensitive loss function* [13], a more sophisticated penalty tool, which defines a region between y_i, the predicted value, and t_i, the actual value. Only when $|t_i - y_i| \geq \varepsilon$ does the function add one of two slack variable penalties, depending on whether they lie *above* (ξ^+) or *below* (ξ^-) the region called the *ε–insensitive tube*. The data to be modeled is a vector $\mathbf{x}_i \in \Re^n$ and a value $y \in \Re$, with a linear function:

$$f(x) = \langle w, \mathbf{x} \rangle + b \quad . \tag{9}$$

The optimal regression function is given by the minimum of the functional,

$$\Phi(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_i (\xi_i^- + \xi_i^+) \quad , \tag{10}$$

with a given value C.

978-1-4673-2531-8/12/$31.00 ©2012 IEEE

The primal form can be obtained by introducing the Lagrange multipliers $\alpha_i^+ \geq 0, \alpha_i^- \geq 0, \mu_i^+ \geq 0, \mu_i^- \geq 0 \quad \forall i$:

$$L_P = C\sum_{i=1}^{L}(\xi_i^+ + \xi_i^-) + \frac{1}{2}\|w\|^2 - \sum_{i=1}^{L}(\mu_i^+\xi_i^+ + \mu_i^-\xi_i^-) - \\ - \sum_{i=1}^{L}\alpha_i^+(\varepsilon + \xi_i^+ + y_i - t_i) - \sum_{i=1}^{L}\alpha_i^-(\varepsilon + \xi_i^- - y_i + t_i) \tag{11}$$

Differentiating (11), setting the derivatives to zero and substituting leads to the optimization problem:

$$\max_{\alpha^+,\alpha^-}\left[\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)t_i - \varepsilon\sum_{i=1}^{L}(\alpha_i^+ + \alpha_i^-) - \right. \\ \left. -\frac{1}{2}\sum_{i,j}(\alpha_i^+ - \alpha_i^-)(\alpha_j^+ - \alpha_j^-)\mathbf{x}_i \cdot \mathbf{x}_j\right] \quad , \tag{12}$$

subject to,

$$0 \leq \alpha_i^+ \leq C \tag{13a}$$

$$0 \leq \alpha_i^- \leq C \tag{13b}$$

$$\sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-) = 0 \quad \forall_i \quad . \tag{13c}$$

Predicted values can be found with:

$$y' = \sum_{i=1}^{L}(\alpha_i^+ - \alpha_i^-)\mathbf{x}_i \cdot \mathbf{x}' + b \quad . \tag{14}$$

Those samples with $0 < \alpha_i < C$ and $\xi_i^+ = 0$ (or $\xi_i^- = 0$) will be elements of the set S of Support Vectors $\mathbf{x}_s$.

Finally, the bias can be obtained as the average over all elements in S:

$$b = \frac{1}{N_s}\sum_{s\in S}\left[t_s - \varepsilon - \sum_{m\in S}^{L}(\alpha_m^+ - \alpha_m^-)\mathbf{x}_m \cdot \mathbf{x}_s\right] \quad . \tag{15}$$

C. SVM Implementation

There are several libraries that implement SVMs, and one of the most widely used is LIBSVM [15]. We have chosen it for its ease of use, outstanding performance and because it can interface with MATLAB, which allows working with a rapid prototyping and testing framework. It is a library under active development, and provides a set of tools that quickly yield acceptable results. It implements a set of SVM formulations, multi–class classification and probability estimates.

III. ALGORITHM

The algorithm consists of an *off–line* and *on–line*. In the former, the algorithm generates a SVM regression model from each division (or *cell*) of the link space, that is partitioned by a fixed amount in each dimension. In the latter phase, the model corresponding to the region whose pose is going to be predicted is recovered from a *look-up table*, and then the forward kinematics can be approximated with simple and fast operations, suitable to real–time applications.

A. Model Representation and Data Generation

As shown in Fig. 1, the generalized configuration is composed of two frames and a set of linear actuators, typically six. A platform is modeled with two 4×6 matrices, A and B, which represent the end effector and fixed frame, respectively, by the Cartesian homogeneous coordinates where each joint is connected. Thus, it is computationally easy to solve the inverse kinematics by simple mathematical operations (16) where needed.

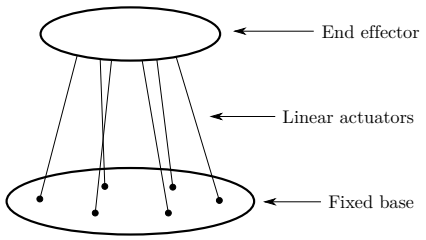


Fig. 1. A class of Parallel Robot.

Consider a fixed base B, with a coordinate frame XYZ attached to it, and another coordinate frame xyz fixed to the end effector A as well. Let us define the *link space* as the 6–D space consisting of the position (as length) of each linear actuator. A *link vector* represents a position in the link space, as a 6–coordinate vector $\mathbf{l} = (l_1 l_2 \cdots l_6)^\top$. Similarly, the *pose space* is the 6–D space that represents the position of the mobile plane (or end effector), as a combination in 3–D Cartesian coordinates of position (x, y, z) and orientation (α, β, γ). A *pose* is defined by the vector $\mathbf{p} = (xyz\alpha\beta\gamma)^\top$.

Given D and R as the position vector and rotation matrix of the end effector, the inverse kinematics can be solved as the Euclidean distance between the positions where each joint is attached:

$$l_i = \sqrt{\|Ra_i + D - b_i\|} \quad i = 1, 2, \cdots, 6 \\ = f_i(\mathbf{p}) \quad , \tag{16}$$

where a_i and b_i are the *i*–th row of the matrix that represents the end effector and fixed frame respectively.

In order to simplify the complexity of the approximation method, the link space needs to be partitioned. The method in the reference paper [9] involves fitting a set of polynomial equations to obtain a parametric model to approximate the forward kinematics solutions in each partition. In this paper we propose a different approach by using SVR machine models to represent the behavior of the platform in a given *region* or partition of the pose and link space. As we stated before, one of the advantages of SVMs is their ability to easily fit complex non–linear functions, but there is a tradeoff with the size of the dataset and the parameters used in terms of training time and accuracy: large training sets will not significantly improve the accuracy of the solutions and will add a time penalty. At the same time, more aggressive parameters will improve the solutions, but will worsen the training time as well. Therefore, the size of the divisions

978-1-4673-2531-8/12/$31.00 ©2012 IEEE

made to the link space balances the tradeoff, because it affects the complexity of the machines to be tuned.

Once the size of the partitions is established, each region needs to be populated with a meaningful number of poses so that the SVR models can be successfully trained. A generated pose is associated with its link space cell, thanks to the inverse kinematics definition (16). The number of divisions along an axis can be described by the equation:

$$N_i = \frac{l_{i,max} - l_{i,min}}{l_c} \quad , \quad i = 1, 2, \cdots, 6 \quad , \qquad (17)$$

where l_c defines the length of each cell. Each generated pose must be tested to discard those that are invalid. A pose is valid if its corresponding link lengths are within the feasible range, and if they do not violate any platform–specific mechanical constraints, e.g., joint limitations or link collisions.

Randomly generated poses are a better choice for yielding a valid dataset than regularly spaced intervals, which will not produce uniform link space coverage due to the nonlinear mapping of the forward kinematics. We have chosen a pseudorandom number generator that belongs to the TGFSR (*Twisted Generalized Feedback Shift Register*) family, introduced in 2006 by L'Ecuyer and Panneton [16], called WELL (*Well Equidistributed Long-period Linear*), a class of robust generators with very good equidistribution, periods, speed and other interesting properties.

A cell of the link space is mapped into a region of the pose space, that can be identified by a unique index, which resembles a change of numerical base. For a given link configuration $L_k = (l_{k1}, l_{k2}, l_{k3}, l_{k4}, l_{k5}, l_{k6})^T$, the index can be obtained by:

$$\text{cell}_k = \sum_{i=1}^{6} \left(\left\lfloor \frac{l_{ki} - l_{i,min}}{l_c} \right\rfloor (N_i)^{i-1} \right) + 1 \quad , \qquad (18)$$

where $l_{i,min}$ is the minimal length or position of the *i*–th link and N_i is the number of cells in each dimension, given by (17). Then, the valid generated poses can be associated by their region index and stored based on this index so they can be processed in the next step.

B. Classification

It should be noted that some regions will represent more than one forward kinematics solution, so before starting the SVR training, they need to be identified and classified. As shown in Fig. 2 (divided into two 3D representations for ease of visualization), the stored data are organized by clusters that need to be identified. In this case they appear overlapped because of the 2D representation, but in other regions they will look like a homogeneous cloud.

The number of clusters to identify, K_{max} partially depends on the structure of the parallel robot: more general configurations of the platform will have more solutions in a given region, i.e. 6–6 (base–end effector link connections) compared to more constrained versions like 6–3 or 3–3. For the considered platform we have set $K_{max} = 8$ as in [9].

The method chosen to estimate the number of clusters for each region starts by fitting a Gaussian Mixture model

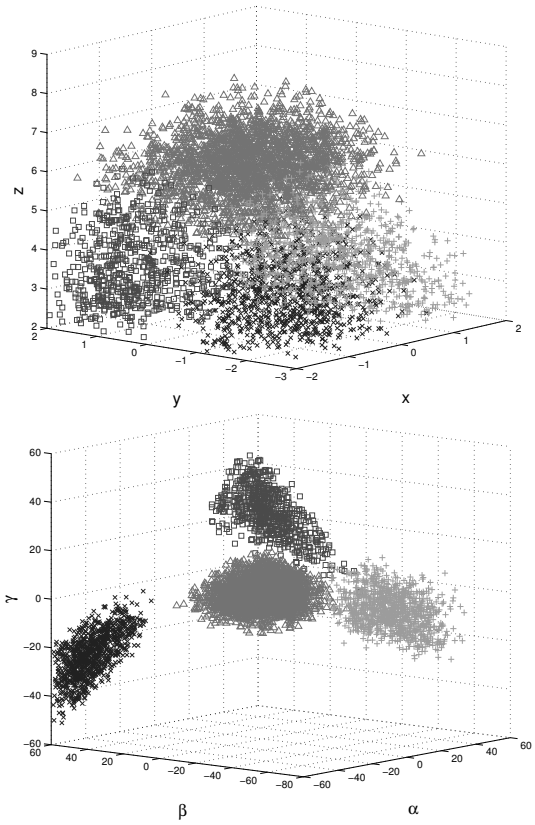


Fig. 2. 3D representation of region 1: top – position subspace; bottom – orientation subspace.

using an Expectation Maximization (EM) algorithm [17] with maximum likelihood estimates, and then applying a measure of the relative goodness of fit of a statistical model called *Akaike Information Criterion* (AIC). Developed by Akaike [18], it is widely used in model selection because it is an estimate of the expected Kullback–Leibler information of a fitted model. A simple way to apply this criterion is by an iterative process, where the Gaussian Mixture model is fitted with *k* components, and where *k* iterates from 1 to the previously set K_{max} value. In each iteration the AIC, a dimensionless quantity, is calculated by the following expression:

$$AIC = 2P - 2\ln(L) \quad , \qquad (19)$$

where P is the number of parameters in the statistical model and L is the maximized value of the likelihood function for the estimated model, obtained with the EM algorithm.

By means of a simple heuristic, it is possible to select a value for K_{max} using the AIC. A procedure like the *Elbow method* does not guarantee the most appropriate guess of the number of clusters: when the data cloud is homogeneous and the points are spread across the region, the presence of an *elbow* point between two iterations might not be a proper

978-1-4673-2531-8/12/$31.00 ©2012 IEEE

estimation of K_{max}. One possible approach to improve the estimation consists of setting a threshold to find a cutoff iteration from which increasing the number of clusters does not improve the fit significantly. If that iteration is greater than the elbow point we have found that setting K_{max} as the midpoint gives a good guess regarding the number of clusters.

The main advantage of this procedure for estimating K_{max} is that we can calculate the posterior probability of each component from the fitted Gaussian Mixture model. Hence, in order to classify each pose we only need to determine the cluster in which its posterior probability is a maximum.

C. Model Training and Lookup Table

After the clustering process, the data is ready to be used as training samples. An SVR is a MISO system, which supports a very large number of inputs as data features. The output or target value is predicted with a decision function, applied to a data instance. Since a trained machine will output one predicted value, then for every cluster we need to train one SVR model for each dimension $[\,xyz\alpha\beta\gamma\,]$ using the same data instances. Therefore, a given cluster will need a total of six trained SVR machines to represent an estimated pose. The training parameters will be described in Section IV.

Link data must be scaled to avoid numerical difficulties and the negative effect of greater numeric ranges against those which happen to be smaller. Both training and testing data must be scaled with the same method and range, e.g., $[\,-1, 1\,]$ or $[\,0, 1\,]$. In the experiments described in the next section we have used the former to adjust the input data.

Since the method produces a large number of models, we need to populate an indexed data structure or *lookup table* in the *off–line* phase to store them. The morphology of this lookup table should allow for fast indexation. Given that we can apply (18) to obtain the address of the *k*–th element of the input vector, the structure can be a sequence of elements that contains the trained model parameters. As the input links in the *on–line* phase must be scaled according to the training data to obtain a valid prediction, the scaling information used for training is also stored. This ends the *off–line* phase, which only needs to be executed once for a single parallel robot.

D. On–line pose parameter estimation

Given a link vector $\mathbf{l}$, the address of the lookup table where the model parameters are stored can be obtained by applying (18). Then, after scaling the link coordinates, each pose space parameter is predicted using $\mathbf{l}$ with the LIBSVM function `svmpredict`. The time to retrieve the six model parameters and generate the estimation is negligible.

The mean error obtained after the experiments is small enough to satisfy a wide variety of applications, but if needed, the accuracy of the approximation can be improved by a correction step. Let $\hat{\mathbf{p}}$ be the pose parameters estimated for the $\mathbf{l}$ input vector after the *on–line* phase. We then apply the inverse kinematics expression (16) to obtain the link lengths $\hat{\mathbf{l}}$ associated with the pose estimation $\hat{\mathbf{p}}$. The difference between the target and the estimated link lengths can be calculated as $\Delta\mathbf{l} = (\mathbf{l} - \hat{\mathbf{l}})$, and can be added to $\mathbf{l}$ to obtain a new target link configuration $\lambda = \mathbf{l} + \Delta\mathbf{l}$. Finally, solving the forward kinematics of the corrected link lengths λ will give an improved approximation of the initial link vector $\mathbf{l}$.

IV. RESULTS AND DISCUSSION

The experiments were made on a workstation with an Intel Xeon CPU E5440@2.83 GHz, 16 GB of RAM, with Linux kernel version 2.6.24–24–generic and MATLAB R2011B, both 64–bit versions. The most time–consuming steps of this method are the classification and the fitting with SVR machines. The data generation took 80 seconds, and the total training time was 11 255 seconds (about 3 hours). Since regions are independent, some steps like classification and SVR training processes can benefit from parallelization, but for the purpose of this paper we have implemented a non-parallel version of the method. Whereas the implementation was mainly written in MATLAB, the random pose generator was coded in C++ and compiled as a MATLAB EXecutable (*MEX*) binary file to reduce the total computation time. The LIBSVM library also provides a MATLAB implementation of its functions with MATLAB *MEX*–files. The resulting lookup table after the end of the *off–line* phase occupied about 57 MB of memory.

The method was tested with the experimental model proposed in [2], where the authors solved the forward kinematics of a parallel manipulator analytically. The link space was decomposed into cells of length $lc = 3.5\ m$, which gives, applying (17), $N_i = 2$ divisions along each dimension. Therefore, the number of cells and regions is $2^6 = 64$. The pose generation was carried out as described in Section III-A, with random data generated within the ranges

$$\begin{array}{llllll} x = [\,-2 & 2\,] & (m) & \alpha = [\,-60 & 60\,] & (^\circ) \\ y = [\,-2.5 & 2\,] & (m) & \beta = [\,-80 & 60\,] & (^\circ) \\ z = [\ \ 2 & 13\,] & (m) & \gamma = [\,-70 & 70\,] & (^\circ) \end{array} , \quad (20)$$

according to the extreme link configurations reported in [2].

When exploring the best training settings for SVRs, we found that the Radial Basis Function (RBF) kernel gave us the best accuracy/training time ratio, especially as compared to the sigmoid kernel. Given the cost of adjusting each one individually, all the SVRs were trained with the same parameters. In the model training function, `svmtrain`, we choose the ε–SVM method, with the parameter $\gamma = 0.1$ (the width of the Gaussian basis function) and the cost equal to 1000 (C parameter in loss function). Also, we set the tolerance of the termination criterion equal to 5×10^{-3} and the epsilon parameter of the loss function equal to 10^{-3}.

There are six configurations of the linear actuators that represent extreme positions of the end effector, relevant when designing a parallel manipulator, since they provide an insight into its workspace: the *lowest and highest position* of the mobile plane, where all of the actuators are at their minimum/maximum lengths; the *most tilted position* (with two asymmetric cases) where one pair of the linear actuators is at its maximum/minimum length, while the other two pairs

978-1-4673-2531-8/12/$31.00 ©2012 IEEE

TABLE I
COMPARISON OF BOTH METHODS AND THE ANALYTICAL SOLUTION

	$x\ (m)$	$y\ (m)$	$z\ (m)$	$\alpha\ (°)$	$\beta\ (°)$	$\gamma\ (°)$
			Lowest position			
A	0.0000	0.0000	2.6460	0.0000	0.0000	0.0000
P	0.0000	0.0000	2.6460	0.0000	0.0000	0.0000
S	0.0000	0.0000	2.6458	0.0006	0.0001	0.0000
			Highest position			
A	0.0000	0.0000	12.9600	0.0000	0.0000	0.0000
P	0.0000	0.0000	12.9610	0.0000	0.0000	0.0000
S	0.0005	−0.0008	12.9610	0.0005	0.0009	−0.0016
			Most Tilted position #1			
A	−1.2360	−2.1420	5.5030	58.9900	−73.8400	−46.0600
P	−1.2390	−2.1390	5.5060	58.9900	−74.0000	−46.0700
S	−1.2369	−2.1421	5.5036	58.9890	−73.8400	−46.0520
			Most Tilted position #2			
A	−0.6930	−1.2000	10.4950	−47.7200	39.4200	−18.0100
S	−0.6939	−1.2017	10.4940	−47.7280	39.4280	−18.0160
			Most Twisted position #1			
A	0.0000	0.0000	7.1920	0.0000	0.0000	68.3620
P	0.0000	0.0000	7.1940	0.0000	0.0000	68.3400
S	0.0001	0.0000	7.1924	0.0012	0.0000	68.3590
			Most Twisted position #2			
A	0.0000	0.0000	7.1920	0.0000	0.0000	−68.3620
S	0.0000	0.0001	7.1923	−0.0001	−0.0010	−68.3600

A : Analytical solution [2]
P : Polynomial method [9]
S : SVR machine method proposed

are at their minimum/maximum lengths; and the *most twisted position*, where the three odd actuators are at their minimum length with the even ones extended to their maximum (with a symmetric opposite case).

Table I shows a comparison of the proposed approach and the polynomial method described in [9] against the exact analytical solution presented in [2]. There were no data available to compare with the polynomial method for the most twisted and most tilted position symmetric and asymmetric cases. It can be seen that, while our method has in some cases a slight decimal error, it actually yields a very good approximation of the exact solution. All results are presented with the correction described in Section III-D. Finally, the experiments yielded an average *on–line* evaluation time of 4.87 milliseconds, enough for real–time operation.

V. CONCLUSION

In this paper we have proposed the use of Support Vector Regression Machines for solving a relevant topic within the parallel robotics field. Unlike other methods, the proposed algorithm is not dependent on the geometry of the platform. Also, the results have shown that it can be applied to problems that require high accuracy. Improvements can be made, however. Smaller cell sizes would provide a rich variety of linear actuators configurations, but the time needed to populate the regions with valid poses would increase as well. The *on–line* phase is fast and suitable for real–time applications, and like the accuracy, the computation and evaluation time can be improved with an implementation in a compiled language like C++. Furthermore, it does not have high memory requirements as it only needs to accommodate the lookup table. Finally, the accuracy of the solutions can be easily improved adjusting the parameters of the SVRs, at the cost of more *off–line* training time.

REFERENCES

[1] B. Dasgupta and T. S. Mruthyunjaya, "The Stewart platform manipulator: a review," *Mechanism and Machine Theory*, vol. 35, no. 1, pp. 15–40, 2000.

[2] K. Liu, J. M. Fitzgerald, and F. L. Lewis, "Kinematic analysis of a Stewart platform manipulator," *IEEE Transactions on Industrial Electronics*, vol. 40, no. 2, pp. 282–293, 1993.

[3] P. Nanua, K. Waldron, and V. Murthy, "Direct Kinematic solution of a Stewart Platform," *Robotics and Automation, IEEE Transactions on*, vol. 6, no. 4, pp. 438–444, August 1990.

[4] J.-P. Merlet, "Solving the Forward Kinematics of a Gough–Type Parallel Manipulator with Interval Analysis," *International Journal of Robotic Research*, vol. 23 (3), pp. 221–235, March 2004.

[5] J. E. Dieudonne, R. V. Parrish, and R. E. Bardusch, "An actuator extension transformation for a motion simulator and an inverse transformation applying Newton–Raphson's method," NASA Langley Research Center, Hampton, Va., Technical Report NASA-TN-D-7067, 1972.

[6] J.-P. Merlet, "Closed–form resolution of the direct kinematics of parallel manipulators using extra sensors data," in *Robotics and Automation, 1993. Proceedings., IEEE International Conference on*, vol. 1, may 1993, pp. 200–204.

[7] S.-H. Chen and L.-C. Fu, "The Forward Kinematics of the 6–6 Stewart Platform Using Extra Sensors," in *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, vol. 6, oct. 2006, pp. 4671–4676.

[8] P. J. Parikh and S. S. Lam, "Solving the forward kinematics problem in parallel manipulators using an iterative artificial neural network strategy," *The International Journal of Advanced Manufacturing Technology*, vol. 40, pp. 595–606, 2009, 10.1007/s00170-007-1360-x. [Online]. Available: http://dx.doi.org/10.1007/s00170-007-1360-x

[9] M. Tarokh, "Real Time Forward Kinematics Solutions for General Stewart Platforms," in *Robotics and Automation, 2007 IEEE International Conference on*, april 2007, pp. 901–906.

[10] B. E. Boser, I. Guyon, and V. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the 5th annual ACM Workshop on Computational Learning Theory*. ACM Press, 1992, pp. 144–152.

[11] V. Vapnik, S. E. Golowich, and A. Smola, "Support vector method for function approximation, regression estimation, and signal processing," in *Advances in Neural Information Processing Systems 9*. MIT Press, 1996, pp. 281–287.

[12] B. E. Boser, I. Guyon, and V. Vapnik, "Pattern recognition system using support vectors," U.S. Patent 5,649,068, July 15, 1997.

[13] V. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer–Verlag New York, Inc., 1995.

[14] S. R. Gunn, "Support vector machines for classification and regression," School of Electronics and Computer Science, University of Southampton, Southampton, U.K., Tech. Rep., 1998.

[15] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[16] F. Panneton, P. L'Ecuyer, and M. Matsumoto, "Improved long–period generators based on linear recurrences modulo 2," *ACM Trans. Math. Softw.*, vol. 32, pp. 1–16, March 2006. [Online]. Available: http://doi.acm.org/10.1145/1132973.1132974

[17] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood Estimation From Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[18] H. Akaike, "A New Look at the Statistical Model Identification," *IEEE Transactions on Automatic Control*, vol. 19, no. 6, pp. 716–723, 1974.

978-1-4673-2531-8/12/$31.00 ©2012 IEEE

3.4 Solving the forward kinematics problem in parallel robots using support vector regression

This section includes the full text for the following article, one of the contributions of this thesis presented as a compendium of publications.

- Title: Solving the forward kinematics problem in parallel robots using support vector regression
- Authors: Antonio Morell, Mahmoud Tarokh and Leopoldo Acosta
- Journal: Engineering Applications of Artificial Intelligence
- Year: 2013
- ISSN: 0952–1976
- doi: 10.1016/j.engappai.2013.03.011

Engineering Applications of Artificial Intelligence 26 (2013) 1698–1706

Contents lists available at SciVerse ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Brief paper

Solving the forward kinematics problem in parallel robots using Support Vector Regression

Antonio Morell [a,*], Mahmoud Tarokh [b], Leopoldo Acosta [a]

[a] Department of Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores (ISAATC), Universidad de La Laguna, 38203 La Laguna, Tenerife, Spain
[b] Department of Computer Science, San Diego State University, San Diego, CA 92182-7720, USA

ARTICLE INFO

Article history:
Received 14 June 2012
Received in revised form
28 February 2013
Accepted 18 March 2013
Available online 3 May 2013

Keywords:
Parallel robots
Stewart platform
Forward kinematics
Real-time
Support vector machines
Spatial decomposition

ABSTRACT

The Stewart platform, a representative of the class of parallel manipulators, has been successfully used in a wide variety of fields and industries, from medicine to automotive. Parallel robots have key benefits over serial structures regarding stability and positioning capability. At the same time, they present challenges and open problems which need to be addressed in order to take full advantage of their utility. In this paper, we propose a new approach for solving one of these key aspects: the solution to the forward kinematics in real-time, an under-defined problem with a high-degree nonlinear formulation, using a popular machine learning method for classification and regression, the Support Vector Machines. Instead of solving a numerical problem, the proposed method involves applying Support Vector Regression to model the behavior of a platform in a given region or partition of the pose space. It consists of two phases, an off-line preprocessing step and a fast on-line evaluation phase. The experiments made have yielded a good approximation to the analytical solution, and have shown its suitability for real-time application.

1. Introduction

A few years after the term *robot* was coined, the first industrial parallel robot, a spray painting machine, was patented by Pollard (1940). Extensively used as manipulators in a wide variety of fields, e.g. medicine, optics, astronomy and many industries like aerospace, automotive and aviation, parallel robots have attracted the attention of many researchers in recent decades. They can be classified as closed-loop mechanisms, in contrast to serial robots, which are usually open-loop kinematics chains. Instead, they consist of two frames connected by means of active links, such as prismatic actuators, sometimes called *legs*. The advantages of this type of structure include a greater rigidity and positioning capability, good dynamic performance and high load carrying capacity, which make them very attractive in many applications and fields. Notable examples of parallel manipulators are the *Delta* robot (Clavel, 1988), the *Tricept* (Siciliano, 1999) and the Gough–Stewart platform. Presented by Gough (1956–1957) as a tyre testing machine, it is one of the most popular parallel manipulators. It gained popularity as a flight simulator (Stewart, 1965–1966) and is commonly known as the *Stewart platform*. The first design as a manipulator system was presented by McCallion and Pham (1979) as an assembly workstation. A detailed and very informative review by Dasgupta and Mruthyunjaya (2000) provides an extensive account of some relevant aspects of the Stewart platform.

In this paper we present a novel method for solving the Forward Kinematics Problem (FKP), still a relevant topic for some types of parallel manipulators, e.g., those with joint offsets. For such robots, unlike the inverse kinematics problem, the FKP lacks a closed-form mathematical solution. It is an under-defined problem with a high-degree of nonlinearity in which the solution in most cases is not unique. Although kinematics is one of the most studied topics of parallel robots, the FKP continues to gain interest, especially in terms of those methods which can solve it in real-time. This is essential for a platform characterization and its closed-loop control to know the position and orientation (*pose*) by means of the length of the linear actuators attached to the joints.

A review of the literature shows that the FKP in parallel robots has been solved in recent years using numerical as well as approximate methods and strategies. Liu et al. (1993) describe an analytical solution for the generalized configuration, i.e., 6-SPS (Spherical–Prismatic–Spherical), whereas some authors propose a simplification of the model (Nanua et al., 1990), which is solved with numerical methods like Newton–Raphson, or use of an interval analysis (Merlet, 2004) to work out the solution. However, these methods lack generalization, since they are proposed for particular types of platform. Other methods develop a solution

* Corresponding author. Tel.: +34 922 31 82 87; fax: +34 922 31 82 88.
E-mail addresses: amorell@isaatc.ull.es (A. Morell), tarokh@cs.sdsu.edu (M. Tarokh), lacosta@ull.es (L. Acosta).

0952-1976/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.engappai.2013.03.011

to the FKP by directly applying a numerical method (Dieudonne et al., 1972) like Newton–Raphson. For instance, Dalvand and Shirinzadeh (2011, 2012) proposed an algorithm which yields solutions by means of an iterative method for a platform with joint offsets, a 6-RRCRR (Rotational–Rotational–Cylindrical–Rotational–Rotational) parallel manipulator, which does not have a closed-form solution for the FKP. In general, these techniques can achieve accurate solutions with enough iterations of the algorithms, but may have convergence problems and high computational requirements. Furthermore, there are other solutions that add rotary type sensors or extra links (Merlet, 1993; Chen and Fu, 2006) to obtain additional information, in order to aid and simplify the algorithms. However, these approaches may be difficult to generalize and in some cases will not be applicable due to structural or mechanical constraints. In addition, approximate solution strategies have been successfully applied, mainly by means of Artificial Intelligence methods like neural networks, in order to study kinematics of serial robots (Gao et al., 2010; Karlik and Aydin, 2000; Köker, 2005; Chiddarwar and Babu, 2010) and parallel robots (Parikh and Lam, 2009; Tarokh, 2007). These strategies are more suited to real-time applications and obtain good enough approximations for a wide variety of fields.

This paper presents a spatial decomposition method for obtaining accurate solutions in real-time for the FKP of Stewart platforms using a popular machine learning method, the *Support Vector Machines* (SVMs), as the regression model. Firstly, the method is applied to a 6-SPS parallel manipulator for which an analytical solution exists. In order to verify its correctness and efficiency, the yielded results are compared with the polynomial curve fitting method proposed by Tarokh (2007), and against the exact analytical solution for a generalized Stewart platform presented by Liu et al. (1993). Secondly, a similar experiment is conducted with a real parallel manipulator, the M-850 hexapod by *Physik Instrumente*.

This paper is organized as follows. Kinematics in parallel robots are discussed in Section 2. The spatial decomposition method is introduced in Section 3, followed by the description of the classification procedure in Section 4. Then, the forward kinematics modeling with SVR machines and the *on-line* evaluation procedure are detailed in Section 5. Finally, results of different experiments are discussed in Section 6, and the main conclusions are summarized in Section 7.

2. On kinematics of parallel robots

2.1. The 6-SPS general Gough–Stewart platform

The theory of serial–parallel duality, which highlights the qualitative distinctions between serial and parallel manipulators, states that in both position and velocity there is a symmetric relationship in the forward and inverse cases (Collins and Long, 1995). In contrast to the simple forward kinematics and complicated inverse kinematics of serial manipulators (requiring the solution of a system of nonlinear equations), parallel manipulators exhibit more or less straightforward inverse kinematics and a challenging solution for the forward kinematics problem.

The generalized Gough–Stewart platform is the most celebrated manipulator in the entire class of parallel robots. It has found a central status in the literature due to the fact that it exhibits the serial–parallel duality in the most prominent manner.

As shown in Fig. 1, the generalized configuration is composed of two platforms and a set of linear actuators, typically six, often called *legs*. Consider a fixed base *B*, with a coordinate frame *XYZ* attached to it, and another coordinate frame *xyz* fixed to the top platform *A*. Let us define the *link space* as the 6-D space consisting

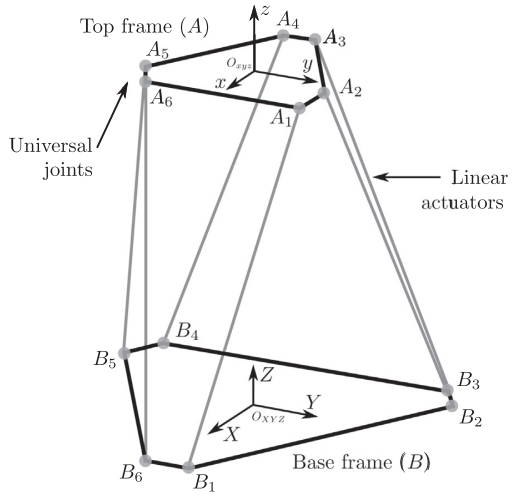


Fig. 1. The general Gough–Stewart platform, a class of parallel robots.

of the value of the length of each leg. A *link vector* represents a position in the link space as a 6-D vector $l=[l_1,l_2,\ldots,l_6]^{\mathrm{T}}$. Similarly, the *pose space* is the 6-D space that represents the position and orientation of the top platform (or end effector), as a combination in 3-D Cartesian coordinates of position $[x,y,z]$ and 3-D orientation angles $[\alpha,\beta,\gamma]$. A *pose* is defined by the vector $p=[x,y,z,\alpha,\beta,\gamma]^{\mathrm{T}}$.

Let d denote the displacement vector of the frame $[x,y,z]$ relative to the frame *XYZ*, let B_i be the position of the i-th link (*leg*) attached to the base relative to *XYZ*, and similarly let A_i be the position of the i-th link attached to the top platform with respect to $[x,y,z]$. Finally let R be the 3×3 rotation matrix that defines the rotating angles of the frame $[x,y,z]$ with respect to the frame *XYZ*. Since there is no joint offsets, the length of each link connecting the base to the top can be written as

$$l_i=\sqrt{\|RA_i+d-B_i\|}\quad i=1,2,\ldots,6$$
$$=f_i(p), \tag{1}$$

where $f_i(p)$ is a known function and p is the top platform pose. It is noted that A_i and B_i are known for a given hexapod. Furthermore, specification of the pose, i.e., position and orientation of the top platform, determines the rotation matrix R and displacement vector d.

For parallel robots, the forward kinematics problem can be stated as follows: Given a link vector l, and the position of link attachments to the top and base platforms, A_i and B_i (with $i=1,2,3,4,5,6$), respectively, find the set of all possible poses p that satisfy (1).

2.2. A platform with joint offsets: the 6-RRCRR case

This kind of hexapod is convenient in some situations, for example when there are manufacturing constraints, since universal joints are more complex to produce. Fig. 2 shows a diagram for a given leg. It is composed of two revolute joints, separated by an offset given by P_f, a prismatic joint, a passive cylindrical joint, and similarly, another two revolute joints with the same offset. For the i-th leg, the joint variables are q_{ki}, with $k=1,2,3,4,5,6$. This configuration leads to more complicated kinematics equations, since dependency between joint variables exists. An inverse kinematics solution for this parallel manipulator is the distance between joints q_{1i} and q_{6i}, i.e., the Euclidean norm of vector q_{3i},

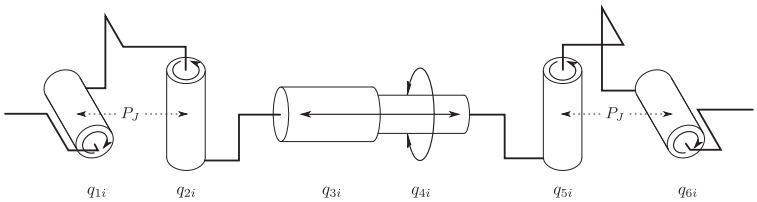


Fig. 2. Leg diagram of the 6-RRCRR model.

as follows:

$$q_{3i} = \sqrt{{}^{B_i}X_{q_{3i}}^{2B_i} + {}^{B_i}Y_{q_{3i}}^{2B_i} + {}^{B_i}Z_{q_{3i}}^{2B_i}} \quad (\forall i = 1, \ldots, 6), \tag{2}$$

where

$$\begin{aligned}
{}^{B_i}X_{q_{3i}}^{2B_i} = {} & X + P_A(C(\theta_A)(C(\beta)C(\gamma) + S(\alpha)S(\beta)S(\gamma)) - S(\theta_A)(C(\beta)S(\gamma) \\
& - C(\gamma)S(\alpha)S(\beta))) - P_B C(\theta_B) + P_J S(q_{6i})(C(\rho_A)(C(\theta_A)(C(\beta)S(\gamma) \\
& - C(\gamma)S(\alpha)S(\beta)) + S(\theta_A)(C(\beta)C(\gamma) + S(\alpha)S(\beta)S(\gamma))) \\
& + S(\rho_A)(C(\theta_A)(C(\beta)C(\gamma) + S(\alpha)S(\beta)S(\gamma)) - S(\theta_A)(C(\beta)S(\gamma) \\
& - C(\gamma)S(\alpha)S(\beta)))) - P_J S(q_{1i})(C(\rho_B)S(\theta_B) + C(\theta_B)S(\rho_B)) \\
& - P_J C(q_{1i})S(\phi_B)(C(\rho_B)C(\theta_B) - S(\rho_B)S(\theta_B)) + P_J C(\alpha)S(\beta)C(q_{6i})
\end{aligned}$$

$$\begin{aligned}
{}^{B_i}Y_{q_{3i}}^{2B_i} = {} & Y - P_B S(\theta_B) + P_A(C(\alpha)C(\gamma)S(\theta_A) + C(\alpha)C(\theta_A)S(\gamma)) \\
& + P_J S(q_{1i})(C(\rho_B)C(\theta_B) - S(\rho_B)S(\theta_B)) \\
& - P_J S(\alpha)C(q_{6i}) - P_J S(q_{6i})(C(\rho_A)(C(\alpha)C(\gamma)C(\theta_A) - C(\alpha)S(\gamma)S(\theta_A)) \\
& - S(\rho_A)(C(\alpha)C(\gamma)S(\theta_A) \\
& + C(\alpha)C(\theta_A)S(\gamma))) - P_J C(q_{1i})S(\phi_B)(C(\rho_B)S(\theta_B) + C(\theta_B)S(\rho_B))
\end{aligned}$$

$$\begin{aligned}
{}^{B_i}Z_{q_{3i}}^{2B_i} = {} & Z - H_B - P_A(C(\theta_A)(C(\gamma)S(\beta) - C(\beta)S(\alpha)S(\gamma)) - S(\theta_A)(S(\beta)S(\gamma) \\
& + C(\beta)C(\gamma)S(\alpha))) - P_J S(q_{6i})(C(\rho_A)(C(\theta_A)(S(\beta)S(\gamma) \\
& + C(\beta)C(\gamma)S(\alpha)) + S(\theta_A)(C(\gamma)S(\beta) - C(\beta)S(\alpha)S(\gamma))) \\
& + S(\rho_A)(C(\theta_A)(C(\gamma)S(\beta) - C(\beta)S(\alpha)S(\gamma)) - S(\theta_A)(S(\beta)S(\gamma) \\
& + C(\beta)C(\gamma)S(\alpha)))) - P_J C(\phi_B)C(q_{1i}) + P_J C(\alpha)C(\beta)C(q_{6i}).
\end{aligned} \tag{3}$$

As can be seen, there is a dependency with joint angles q_{1i} and q_{6i}, which leads to a set of nonlinear equations. For further details, please see Dalvand and Shirinzadeh (2011, 2012), where the kinematics of this hexapod was studied as a part of a skull-base surgery system. In Section 6 we compare the results yielded by our method with the results reported in that work.

3. Spatial decomposition and data generation

As described by Tarokh (2007), the spatial decomposition method consists of an *off-line* preprocessing phase and a fast *on-line* evaluation phase. In the off-line phase, the method decomposes the link space into small rectangloid *cells*. Using (1), it assigns random values within the range of interest to the pose p and computes the link values l_i. Data points that fall into a cell in the link space have corresponding points in a *region* of the pose space. The relationship between data points in a cell and its corresponding region must be modeled so that when a given set of link values is specified, the sets of all pose values can be computed during the on-line phase. The parameters of this relationship are stored in the form of a look-up table. The parameters are then retrieved during the on-line phase.

The method described by Tarokh (2007) for modeling the relationship involves fitting a set of polynomial equations to obtain a parametric model which approximates the forward kinematics solution. In this paper we propose a different approach using a Support Vector Regression (SVR) machine to relate the platform pose in each *region* to the link vector in the corresponding *cell* of the link space. One of the advantages of SVMs is their ability to easily fit complex nonlinear functions. However, there is a tradeoff between the size of the dataset and the parameters used in terms of training time and accuracy: large training sets will not significantly improve the accuracy of the solutions and will add a time penalty. At the same time, fine tuning the parameters will improve the solutions, but will worsen the training time as well. An example of this behavior is given in Section 6. Therefore, the size of the cells in the link space provides a tradeoff, because it affects the complexity and the number of the machines to be tuned. However, SVR machines allow for fewer link and pose space divisions, since they can model more complex data. Specifically, in the given numerical example described in Section 6, the link and pose space were divided by only 64 regions without accuracy loss, compared with the 15 625 needed by the polynomial method (Tarokh, 2007).

The number of divisions along an axis of the link space can be described by

$$N_i = \frac{l_{i,max} - l_{i,min}}{l_c}, \quad i = 1, 2, \ldots, 6, \tag{4}$$

where l_c is the side length of a cell, which determines its size.

Now each region in the pose space and its corresponding link cell must be populated with a meaningful number of pose and link data points, so that the SVR models can be successfully trained. Randomly generated pose is a better choice than regularly spaced intervals for yielding a valid dataset. The latter will not produce uniform link space coverage due to the nonlinear mapping of the forward kinematics. We have chosen a pseudorandom number generator that belongs to the TGFSR (*Twisted Generalized Feedback Shift Register*) family, introduced by Panneton et al. (2006), called WELL (*Well Equidistributed Long-period Linear*), a class of robust generators with very good equidistribution, periods, speed and other interesting properties. Each generated pose must be tested to discard those that are invalid. A pose is valid if its corresponding link lengths are within the feasible range, and if they do not violate any platform-specific mechanical constraints, e.g., joint limits or link collision.

A cell of the link space is mapped into a region of the pose space, that can be identified by a unique index, as follows. For a given link configuration $L_k = (l_{k1}, l_{k2}, l_{k3}, l_{k4}, l_{k5}, l_{k6})^T$, the index is obtained by

$$C_k = \left(\sum_{i=1}^{6} \left\lfloor \frac{l_{ki} - l_{i,min}}{l_c} \right\rfloor (N_i)^{i-1} \right) + 1, \tag{5}$$

where $l_{i,min}$ is the minimal length or position of the i-th link and N_i is the number of cells in each dimension, given by (4). Thus, the valid generated poses can be associated with a unique link space cell and stored based on their region index in order to be processed in the next step.

4. Classification

The generated data form clusters in the pose space, representing different solutions over a specific region. These clusters must be classified before starting the SVR training, in order to represent a more accurate model. In the example shown in Fig. 3 they appear partially overlapped because of the 2D representation, but in other cases they will look like a more homogeneous cloud.

Depending on the classification method, the number of clusters to identify, K_c, is a required input parameter. It partially depends on the structure of the parallel robot. Irregular shaped platforms with more leg attachment points will have more solutions (poses).

The method chosen to estimate the number of clusters for each region starts by fitting a Gaussian Mixture model using an *Expectation Maximization* (EM) algorithm (Dempster et al., 1977) with maximum likelihood estimates. A measure for the relative fitness of a statistical model called *Akaike Information Criterion* (AIC) is then applied. Developed by Akaike (1974), it is widely used in model selection because it is an estimate of the expected Kullback–Leibler information of a fitted model. A simple strategy to apply this criterion is by an iterative process, where the Gaussian Mixture model is fitted with k components, and where k iterates in $[1, \ldots, k, \ldots, K_{max}]$, i.e., the maximum expected value for K_c, which should be set according to the platform structure. In each iteration the AIC, a dimensionless quantity, is calculated by the following expression:

$$AIC = 2P - 2\ln(L), \tag{6}$$

where P is the number of parameters in the statistical model and L is the maximized value of the likelihood function for the estimated model, obtained with the EM algorithm.

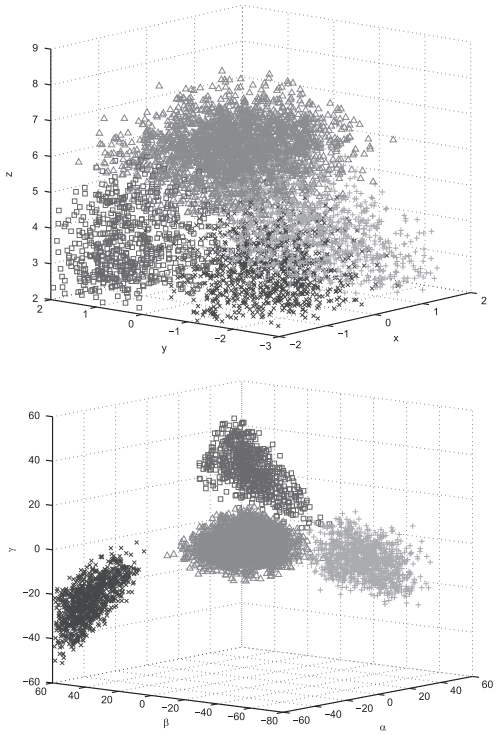


Fig. 3. 3D representation of region 1: first—position subspace; second—orientation subspace.

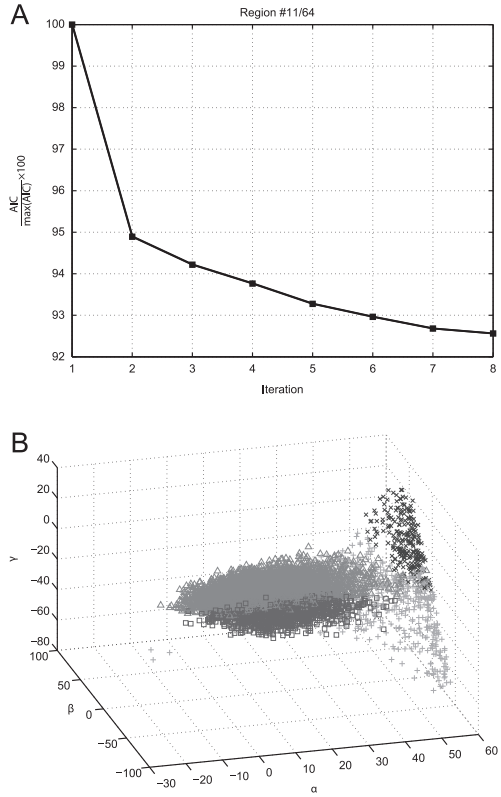


Fig. 4. First—AIC value for each iteration; second—orientation subspace. The cutoff is at the midpoint.

Given the values after each iteration, K_c can be heuristically selected. We have found that a procedure like the *Elbow method* does not guarantee the most appropriate guess of the number of clusters. For example, as illustrated in Fig. 4 (where we have plotted the orientation subspace for simplicity), the presence of an *elbow* point between the second and third iteration suggests a $K_c = 2$. The values are shown as a percentage, normalized by their maximum value, which happens at the first iteration. Whereas the cloud is homogeneous, the points are spread across the region, so it seems appropriate to divide them into more than the two clusters that the Elbow method suggests. One possible approach to automate the selection consists of setting a threshold to find a cutoff iteration from which increasing the number of clusters does not significantly improve the fit. If that iteration is greater than the elbow point, we have found that selecting the midpoint gives a good guess for K_c.

The main advantage of this procedure is that we can calculate the posterior probability of each region point for each cluster from the fitted Gaussian Mixture model. Hence, in order to classify each pose we only need to find the cluster in which its posterior probability is maximum. Furthermore, thanks to the SVR modeling capabilities, there are fewer regions to be classified, 64 instead of 15 625 in the example described in Section 6.

5. FK modeling and evaluation

After the clustering step, the data are ready to be used as training samples. However, the data must be preprocessed before

1702 A. Morell et al. / Engineering Applications of Artificial Intelligence 26 (2013) 1698–1706

starting the training process. First, removing outliers could significantly reduce the training time of the related SVR machine. To detect and remove them we have used the Multivariate Outlier Detection algorithm presented by Peña and Prieto (2001). Second, link data must be scaled (normalized) to avoid numerical difficulties and the adverse effect of greater numeric ranges against those which happen to be smaller. Both training and testing data must be scaled with the same method and range, e.g., [−1,1] or [0,1]. In the experiments described in the next section we have used the former to normalize the input data, which is now suitable for being used as SVR training samples.

Support Vector Machines were formally introduced as a pattern recognition tool by Vapnik and his co-workers (Boser et al., 1992), and a few years later they were extended to solve regression problems, constructing multidimensional splines and solving linear operator (Vapnik, 1995; Vapnik et al., 1996). As a statistical learning tool, SVMs were developed with a greater ability to generalize. When solving pattern recognition problems, a SVM is able to find in a feature space a decision rule with good generalization, i.e., the optimal separating hyperplane, by selecting a small subset of the training data, called the Support Vectors (SVs). It can be shown that optimal separation of the SVs is equivalent to the optimal separation of the entire data (Vapnik et al., 1996). Maximizing the margin between classes and training samples is an alternative to other optimizing cost function methods, e.g., mean square error. This provides some important features to SVMs, like automatic capacity tuning of the classification function, the representation of the data relevant for the classification by a small subset (data compression), and uniqueness of the solution. Hence, SVMs can be used to summarize the information contained in a dataset. The experiments carried out in this work prove that using SVR machines to model the behavior of the platform in a given region yields results good enough for a wide variety of applications.

A SVR is a multi-input single-output system, which supports a very large number of inputs as data features, while the output or target value is predicted with a decision function. Since one trained machine will output one predicted value, each pose prediction will need a trained SVR model for each dimension $[x, y, z, \alpha, \beta, \gamma]$ using the same data instances. Therefore, a given cluster will need a total of six trained SVR machines in order to obtain a full pose estimation. Fig. 5 shows a description of the implementation. The training parameters will be described in Section 6.

The training algorithm for a given SVR machine requires an input vector (p_i, l_k), where $l_k \in \Re^6$ is the k-th link configuration vector, with $k=1,\ldots,m$, and $p_i \in \Re$ is the corresponding value of the i-th pose vector parameter. The algorithm finds the parameters of a decision function $F(l)$ using these training samples

$$F(l) = \langle w, l_k \rangle + u. \tag{7}$$

The goal of the algorithm is to find the parameter w and the bias u that represent the optimal separating hyperplane, which model the pose parameter p_i in the given region.

The decision function has a dual space representation, which reduces the number of computations required to train the classifier (Boser et al., 1992)

$$F(l) = \sum_{k=1}^{m} \alpha_k K(l_k, l) + u, \tag{8}$$

where the coefficients α_k are the parameters found by the algorithm, l_k are the training samples, with m the number of samples. $K(l_k, l)$ is the kernel function, which allows operations to be performed in the input space rather than in a high dimensional feature space. Extending the formulation to the *dual form* allows the application of different kernel functions, such as polynomials, radial basis functions and certain sigmoid functions.

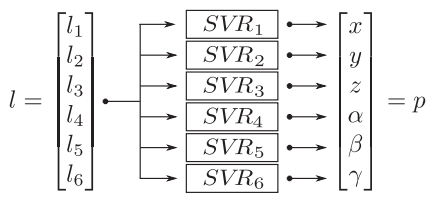


Fig. 5. SVR models corresponding to each pose space parameter for a given cluster.

As a regression tool, a SVR employs an alternative loss function, which includes a distance measure. Vapnik (1995) proposed the *ε-insensitive loss function*, a more sophisticated penalty tool, which defines a region between $\hat{p_i}$, the predicted value, and p_i, the actual value. Only when $|p_i - \hat{p_i}| \geq \varepsilon$ does the function add one of two slack variable penalties, depending on whether they lie *above* or *below* a certain region called the *ε-insensitive tube*. The ε parameter has an effect on the smoothness of the response and it affects the number of Support Vectors, i.e., the generalization capability.

To store the data of the trained models, the method needs to populate an indexed data structure or *lookup table* in the off-line phase, which should allow fast indexation. Given that we can apply (5) to obtain the address of the k-th element of the input vector, the structure can be a sequence of elements that contains the trained model parameters. The input link data in the on-line phase must be scaled according to the training data to obtain a valid prediction, so the scaling information used for training is also stored. This completes the off-line phase, which only needs to be executed once for a given parallel robot.

Given a link vector l, the address of the lookup table where the model parameters are stored can be obtained by applying (5). Then, after scaling the link coordinates, each pose space parameter is predicted using l. The time to retrieve the six model parameters and generate the estimation is negligible, as will be described in the next section.

The mean error obtained after the experiments is small enough to satisfy a wide variety of applications, but if needed, the accuracy of the approximation can be improved by a correction step. Let $\hat{p}$ be the pose parameters estimated for the l input vector after the on-line phase. Applying the inverse kinematics expression (1) yields an estimation of the link lengths $\hat{l}$ associated with the pose estimation $\hat{p}$. The difference between the target and the estimated link lengths can be calculated as $\Delta l = (l - \hat{l})$, and can be added to l to obtain a new target link configuration $\lambda = l + \Delta l$. Finally, solving the forward kinematics of the corrected link lengths λ will give an improved approximation of the initial link vector l.

6. Results and discussion

6.1. Experimental setup

The experiments were made on a workstation with an Intel Xeon CPU E5440@2.83 GHz, 16 GB of RAM, with Linux kernel version 2.6.24-24-generic and MATLAB R2011B, both 64-bit versions. The most time-consuming steps are the classification and the SVR machines training. For the 6-SPS (Spherical–Prismatic–Spherical) model, the total training time was approximately 3 h. The off-line time for the 6-RRCRR model was about 9 h. Whereas the implementation of the method was mainly written in MATLAB, the random pose generator was coded in C++ and compiled as a MATLAB EXecutable (*MEX*) binary file to reduce the total computation time.

There are several libraries that implement SVMs, and one of the most widely used is LIBSVM (Chang and Lin, 2011). We have

chosen it for its ease of use, outstanding performance and because it can interface with MATLAB, which allows working with a rapid prototyping and testing framework. It is a library under active development, and provides a set of tools that can quickly yield acceptable results. It implements a set of SVM formulations, multi-class classification and probability estimates. Also, four basic kernel functions are available: linear, polynomial, sigmoid and Radial Basis Function (RBF) kernel, defined as

$$K(l_i, l_j) = e^{-\gamma \|l_i - l_j\|^2}, \quad \gamma > 0, \tag{9}$$

where the γ parameter behaves as a scaling factor. The LIBSVM library also provides a MATLAB interface with its functions, based on MATLAB *MEX*-files.

When exploring the best training settings for the SVR machines, we found that the Radial Basis Function kernel (9) yielded the best accuracy/training time ratio, especially as compared to the sigmoid kernel. Given the cost of adjusting each one individually, all the SVRs were trained with the same parameters. In the model training function, `svmtrain`, we have chosen the ε-SVM method, with the parameter $\gamma = 0.1$ (the width of the Gaussian basis function) and the cost equal to 1000 (C parameter in loss function). The given value C is a regularization parameter, which must be chosen to reflect the noise in the data (Gunn, 1998). It controls the trade-off between training error and model complexity. Also, we set the tolerance of the termination criterion equal to 5×10^{-3} and the epsilon parameter of the loss function equal to 10^{-3}. The forward kinematics is solved with the function `svmpredict`.

The resulting lookup table after the completion of the off-line phase occupied about 57 MB of memory for the 6-SPS robot, and about 35 MB for the 6-RRCRR model.

6.2. Generalized 6-SPS model

The method was first tested with the experimental model proposed by Liu et al. (1993), where the forward kinematics of a parallel manipulator is solved analytically. This allows us to verify the correctness and efficiency of the method against known solutions for a given platform. The top frame is an equilateral triangle, as can be seen in Fig. 6.

The model is described with respect to a pair of matrices, A and B, as shown in Fig. 7, where the coordinates of each vertex are described. Thus, both matrices are as follows:

$$A = \begin{bmatrix} \frac{\sqrt{3}}{6}a & \frac{1}{2}a & 0 \\ \frac{\sqrt{3}}{6}a & \frac{1}{2}a & 0 \\ -\frac{\sqrt{3}}{3}a & 0 & 0 \\ -\frac{\sqrt{3}}{3}a & 0 & 0 \\ \frac{\sqrt{3}}{6}a & -\frac{1}{2}a & 0 \\ \frac{\sqrt{3}}{6}a & -\frac{1}{2}a & 0 \end{bmatrix}^{\top}, \tag{10}$$

$$B = \begin{bmatrix} \frac{\sqrt{3}}{6}(2b+c) & \frac{1}{2}c & 0 \\ -\frac{\sqrt{3}}{6}(b-c) & \frac{1}{2}(b+c) & 0 \\ -\frac{\sqrt{3}}{3}(b+2c) & \frac{1}{2}b & 0 \\ -\frac{\sqrt{3}}{3}(b+2c) & -\frac{1}{2}b & 0 \\ -\frac{\sqrt{3}}{6}(b-c) & \frac{1}{2}(b+c) & 0 \\ \frac{\sqrt{3}}{6}(2b+c) & -\frac{1}{2}c & 0 \end{bmatrix}^{\top}. \tag{11}$$

Matrix A (10) indicates the positions of the links at the connection to the top platform relative to frame xyz with respect to the global frame XYZ. Similarly, matrix B (11) represents the links positions at the connection to the base relative to frame XYZ. The experimental model is described with the parameters

$$\begin{aligned} &a = 10.0\ \text{m} \quad b = 15.0\ \text{m} \quad c = 1.0\ \text{m} \\ &L_{min} = 8.0\ \text{m} \quad L_{max} = 15.0\ \text{m}, \end{aligned} \tag{12}$$

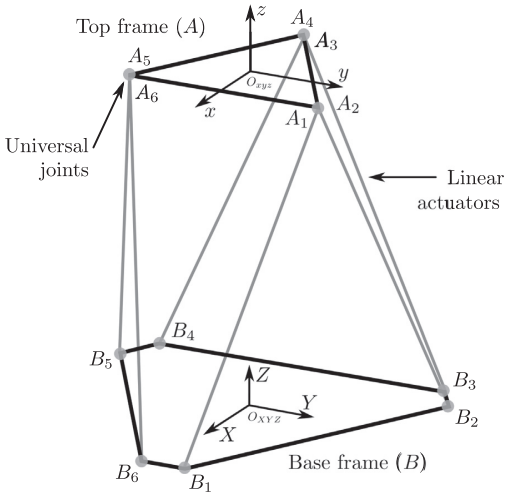


Fig. 6. 6-SPS model (Liu et al., 1993).

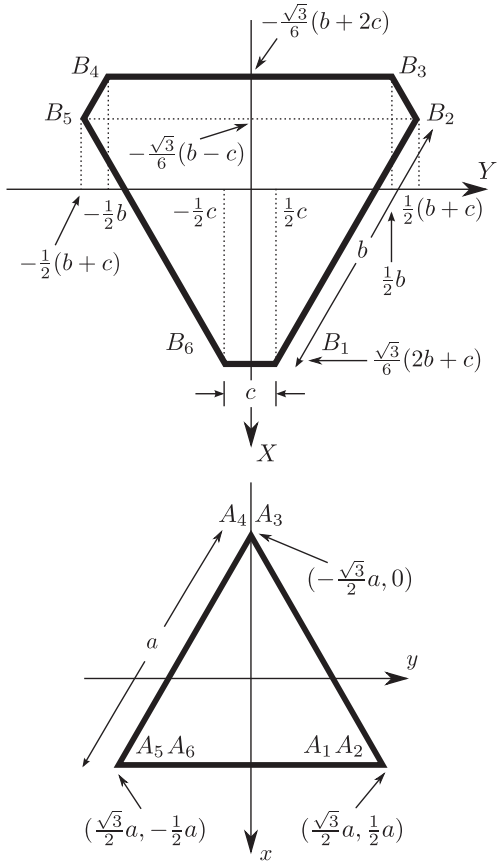


Fig. 7. 6-SPS model parameters: first—bottom frame; second—top frame.

where L_{min} and L_{max} are the minimum and maximum length of the linear actuators, respectively. It should be noted that Z coordinate on matrix A is zero because it describes the top frame w.r.t. the base coordinate system XYZ. For a given top platform posture, such plane is rotated and translated according to it.

The pose generation was carried out as described in Section 3, with random data generated within the ranges

$$\begin{array}{llll} x=[-2\ \ 2] & (\text{m}) & \alpha=[-60\ \ 60] & (^\circ) \\ y=[-2.5\ \ 2] & (\text{m}) & \beta=[-80\ \ 60] & (^\circ) \\ z=[2\ \ 13] & (\text{m}) & \gamma=[-70\ \ 70] & (^\circ), \end{array} \tag{13}$$

according to the extreme link configurations reported by Liu et al. (1993). The link space was decomposed into cells of length $lc=3.5$ m in which applying (4) gives $N_i=2$ divisions along each dimension. Therefore, the number of cells and regions is $2^6=64$. The number of poses generated was 320 000, about 4000 for each region.

There are six configurations of the linear actuators that correspond to representative positions of the top platform relevant when designing a parallel manipulator, since they provide an insight into its workspace. The *lowest and highest positions* of the end effector, with all the actuators at their minimum and maximum lengths, $l=[8,8,8,8,8,8]^\top$ and $l=[15,15,15,15,15,15]^\top$, respectively. The *most tilted position*, where one adjacent pair of linear actuators is at its maximum (or minimum) length, while the other two pairs are at their minimum (or maximum) lengths, with two asymmetric cases, $l=[15,15,8,8,8,8]^\top$ and $l=[9.9015,9.9015,15,15,15,15]^\top$. Finally, the *most twisted position*, where the three odd numbered actuators are at their minimum length with the even ones extended to their maximum, $l=[8,15,8,15,8,15]^\top$, with a symmetric case $l=[15,8,\ 15,8,15,8]^\top$. In the experiments conducted, these extreme configurations are evaluated using the proposed approach and are compared with the other methods.

6.3. 6-RRCRR experimental model

This type of parallel robot is a Gough–Stewart platform with orthogonal non-intersecting RR pairs or offset U-joints configurations. Fig. 8 shows both frames and leg attachments. First joint on both frames is rotated by ϕ_A and ϕ_B with respect to X-axis, as shown in that figure. Legs are distributed radially every 120°. Such radius is P_A and P_B. The gap between two legs of the same pair is given by the angles θ_A and θ_B. They are also rotated by ρ_A and ρ_B to make them parallel. Legs are connected to the same attachment index, i.e., A_i is connected with B_i, with $i=1,2,3,4,5,6$. Finally, points B_i are within a plane located H_B millimeters with respect to Z-axis of the world frame. Similarly, end effector is H_A millimeters above the plane formed by points A_i.

The parameters of the experimental model used correspond to the M-850 hexapod by *Physik Instrumente*, which are summarized in

$$\begin{array}{ll} H_B=35.99275\ \text{mm} & H_A=25.0\ \text{mm} \\ P_B=151.076\ \text{mm} & P_A=92.371\ \text{mm} \\ \theta_B=0.2389\ \text{rad} & \theta_A=0.3922\ \text{rad} \\ \rho_B=0.11945\ \text{rad} & \rho_A=0.1961\ \text{rad} \\ \phi_B=0.32167\ \text{rad} & \phi_A=0\ \text{rad} \\ P_J=13.0\ \text{mm} & \end{array} \tag{14}$$

For this model, random pose data was generated within the following ranges:

$$\begin{array}{llll} x=[-50\ \ 50] & (\text{mm}) & \alpha=[-15\ \ 15] & (^\circ) \\ y=[-50\ \ 50] & (\text{mm}) & \beta=[-15\ \ 15] & (^\circ), \\ z=[303.4\ \ 353.4] & (\text{mm}) & \gamma=[-30\ \ 30] & (^\circ) \end{array} \tag{15}$$

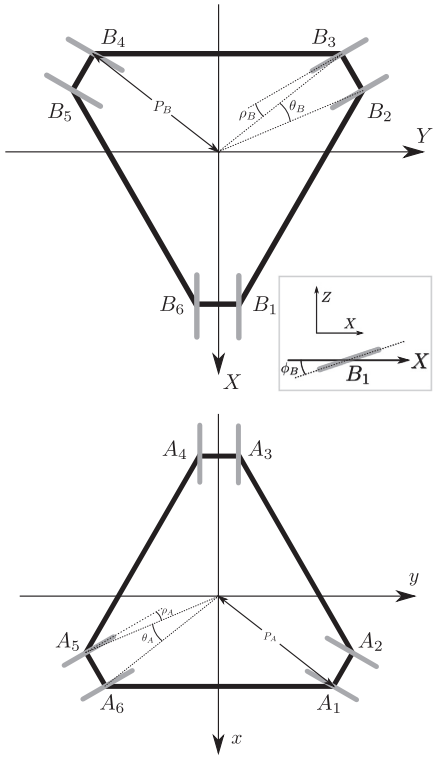


Fig. 8. 6-RRCRR model parameters: first—bottom frame; second—top frame.

as described by the datasheet and technical data, available at manufacturer's webpage (Physik Instrumente, PI). Similar to the 6-SPS model, the link space was decomposed into cells of size 120 mm, which gives $N_i=2$ divisions along each dimension. The number of cells and regions was also $2^6=64$. Two accuracy tests were made, where each region were populated with about 2000 and 18 000 random poses. The total off-line time for the latter was almost 2 days, which had to be done only once.

6.4. Experimental results

Table 1 shows a comparison of the proposed approach and the polynomial method described by Tarokh (2007) applied to the 6-SPS parallel platform. In this experiment we solved the forward kinematics of extreme positions, and compared the yielded results against the exact analytical solution presented by Liu et al. (1993). This procedure allows us to check the correctness of the proposed method with the known forward kinematics solution given by the analytical method. The estimated link lengths obtained with the polynomial method for these configurations are not available, so we cannot compare the absolute full-range errors for both estimations. However, it can be seen that, while our method has in some cases a slight error, it actually yields a very good approximation of the exact solution. The best approximation seems to be the Most Tilted position #1, and the worst, the Highest position. The others yield more accurate estimations than those reported with the polynomial method. Results are presented with the correction step described in Section 5. The absolute full-range error for each extreme configuration is shown in Table 3, with and without correction. Each of them is lower than the average error reported by Tarokh (2007).

Table 1
Comparison of both methods and the analytical solution: 6-SPS model.

Method	x (m)	y (m)	z (m)	α (°)	β (°)	γ (°)
Lowest position						
A	0.0000	0.0000	2.6460	0.0000	0.0000	0.0000
P	0.0000	0.0000	2.6460	0.0000	0.0000	0.0000
S	0.0000	0.0000	2.6458	0.0006	0.0001	0.0000
Highest position						
A	0.0000	0.0000	12.9600	0.0000	0.0000	0.0000
P	0.0000	0.0000	12.9610	0.0000	0.0000	0.0000
S	0.0005	−0.0008	12.9610	0.0005	0.0009	−0.0016
Most Tilted position #1						
A	−1.2360	−2.1420	5.5030	58.9900	−73.8400	−46.0600
P	−1.2390	−2.1390	5.5060	58.9900	−74.0000	−46.0700
S	−1.2369	−2.1421	5.5036	58.9890	−73.8400	−46.0520
Most Tilted position #2						
A	−0.6930	−1.2000	10.4950	−47.7200	39.4200	−18.0100
S	−0.6939	−1.2017	10.4940	−47.7280	39.4280	−18.0160
Most Twisted position #1						
A	0.0000	0.0000	7.1920	0.0000	0.0000	68.3620
P	0.0000	0.0000	7.1940	0.0000	0.0000	68.3400
S	0.0001	0.0000	7.1924	0.0012	0.0000	68.3590
Most Twisted position #2						
A	0.0000	0.0000	7.1920	0.0000	0.0000	−68.3620
S	0.0000	0.0001	7.1923	−0.0001	−0.0010	−68.3600

A: Exact method (Liu et al., 1993).
P: Polynomial method (Tarokh, 2007).
S: SVR machine method proposed.

Table 2
Comparison of both methods: 6-RRCRR model.

Method	x (mm)	y (mm)	z (mm)	α (°)	β (°)	γ (°)
Test position #1						
T	55.0000	−32.0000	316.4000	−0.2094	0.4363	−0.3316
N	55.0001	−32.0000	343.4000	−0.2094	0.4363	−0.3316
S	54.9903	−31.9882	316.4018	−0.2095	0.4358	−0.3312
Test position #2						
T	−36.0000	44.0000	310.4000	0.0873	−0.2967	0.3665
N	−36.0000	44.0000	310.4000	0.0873	−0.2967	0.3665
S	−36.0020	43.9994	310.3986	0.0872	−0.2966	0.3666

T: Test position.
N: Numerical method (Dalvand and Shirinzadeh, 2012).
S: SVR machine method proposed.

Table 3
Full-range errors for extreme configurations: 6-SPS model.

Correction	**Lowest**	**Highest**
w/o	0.0142%	0.0137%
w/	0.0007%	0.0048%
Correction	**Most Tilted #1**	**Most Tilted #2**
w/o	0.0105%	0.0195%
w/	0.0098%	0.0067%
Correction	**Most Twisted #1**	**Most Twisted #2**
w/o	0.0211%	0.0202%
w/	0.0007%	0.0009%

The next experiment is an accuracy test, where 1000 randomly generated link lengths are solved with the proposed method. Then, the inverse kinematics of those estimated poses is solved applying (1). The average absolute error of the estimated link lengths is obtained with $|l-\hat{l}|$. Also, this test is useful to measure the on-line evaluation time. Since it is very difficult to take a measure of the total evaluation time for a single configuration of the linear actuators, a more appropriate approach is to obtain the average from a sequence of tests. For the 6-SPS parallel platform, the average link length error and its standard deviation σ are lower after the correction step, as seen in Table 4. The total time taken to complete the accuracy test was 4.87 s, which means an average evaluation time of 4.87 ms for each test.

Table 4
Accuracy test with 1000 link configurations: 6-SPS model.

Method	**Error**	σ	
P	0.00478 m 0.6829%	0.0089 m 0.1271%	**w/o correction**
S	0.1929 m 2.7556%	0.5152 m 7.3597%	
P	0.00124 m 0.0177%	0.0038 m 0.0543%	**w/correction**
S	0.0008 m 0.0117%	0.0022 m 0.0319%	

P: Polynomial method (Tarokh, 2007).
S: SVR machine method proposed.

Table 5
Accuracy test with 1000 link configurations: 6-RRCRR model.

Experiment	**Error**	σ	
Test #1	1.7115 mm 0.7131%	1.8619 mm 0.7758%	**w/o correction**
Test #2	0.0841 mm 0.0350%	0.0357 mm 0.0149%	
Test #1	0.0083 mm 0.0035%	0.0125 mm 0.0052%	**w/correction**
Test #2	0.0011 mm 0.0005%	0.0010 mm 0.0004%	

SVR machine method proposed.
Test #1: About 2000 poses per region.
Test #2: About 18 000 poses per region.

Similarly, we solved the forward kinematics for the poses shown in Table 2, two end effector positions proposed by Dalvand and Shirinzadeh (2012). It can be seen that the proposed method found the solution with a slight error. However, it should be noted that our method is able to find more than one solution when it exists and if the SVR machines are trained with enough data points. Furthermore, the pose prediction using SVR machines is a fast and reliable process, i.e., it does not need an initial guess, and does not suffer from convergence problems. However, this method is able to yield very accurate solutions, which can be further improved with more data points and fine tuning the training parameters, which imply more off-line training time, as shown in Table 5. Unfortunately, there is no similar experiment for the accuracy test to compare with the method proposed by Dalvand and Shirinzadeh (2012) for the 6-RRCRR model. However, results showed a very low error, further improved by the correction step. It can be seen that highly populated regions, i.e., more training data for each SVR machine, yield more accurate results after this test, with a tradeoff of more training time.

Finally, a performance comparison has been conducted on the 6-SPS model. Table 6 shows that accuracy can be improved by increasing the parameter C, although the training time increases as well. However, there is a risk of overfitting when its value is too high. Furthermore, γ affects the accuracy more clearly, since it leads to a change in the model.

Table 6
Performance comparison with different parameters.

C	γ	Time (s)	F-R Error	σ
100	10^{-1}	3092.2	0.0191% 0.0013 m	0.0467% 0.0033 m
500	10^{-1}	7066.63	0.0126% 0.0009 m	0.0347% 0.0024 m
1000	10^{-1}	11255.1	0.0117% 0.0008 m	0.0319% 0.0022 m
2000	10^{-1}	18075.5	0.0117% 0.0008m	0.0320% 0.0022 m
500	10^{0}	2534.92	0.7864% 0.0550 m	1.8259% 0.1278 m
500	10^{-2}	2819.31	0.0324% 0.0023 m	0.0634% 0.0044 m
500	10^{-3}	2137.57	0.1565% 0.0110 m	0.1562% 0.0109 m
1000	10^{-3}	2196.1	0.1358% 0.0095 m	0.1402% 0.0098 m

7. Conclusion

In this paper we have proposed an alternative approach for solving the forward kinematics problem of parallel robots, a very popular class of manipulators that has gained interest in a wide variety of engineering applications. Unlike other methods, the proposed algorithm is not dependent on the geometry of the platform. It consists of a preprocessing phase and a fast evaluation step: random generated poses are validated and classified, in order to be used as training samples of a popular machine learning method, the Support Vector Machines. It can also be applied to problems that require high accuracy, without high memory requirements.

The main contributions of the proposed method are as follows. Firstly, since divisions of the link space (number of cells) are relatively few, the random pose generation is fast. Secondly, the classification takes advantage of fitted Gaussian Mixture Models, which allows simultaneous estimation of the number of clusters and classification. Finally, using Support Vector Regression machines yields a very good approximation of the analytical solution, which is further improved by a correction step. This also leads to an evaluation phase reasonably fast enough to be applied to real-time applications.

Some other considerations are also worthy of note. For example, a smaller cell size would provide higher accuracy, but the time needed to populate the regions with valid poses would increase as well. Finally, the accuracy of the solutions can also be easily improved by adjusting the parameters of the SVRs, at the cost of more off-line training time.

Acknowledgments

The authors gratefully acknowledge the contribution of the Spanish Ministry of Economy and Competitiveness under Project SAGENIA DPI2010-18349.

References

Akaike, H., 1974. A new look at the statistical model identification. IEEE Trans. Autom. Control 19, 716–723.

Boser, B.E., Guyon, I., Vapnik, V., 1992. A training algorithm for optimal margin classifiers. In: Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, ACM Press, pp. 144–152.

Chang, C.C., Lin, C.J., 2011. LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. 2, 27:1–27:27. Software available at ⟨http://www.csie.ntu.edu.tw/∼cjlin/libsvm⟩.

Chen, S.H., Fu, L.C., 2006. The forward kinematics of the 6–6 Stewart platform using extra sensors. In: IEEE International Conference on Systems, Man and Cybernetics, 2006 (SMC '06), pp. 4671–4676.

Chiddarwar, S.S., Babu, N.R., 2010. Comparison of RBF and MLP neural networks to solve inverse kinematic problem for 6R serial robot by a fusion approach. Eng. Appl. Artif. Intell. 23, 1083–1092.

Clavel, R., 1988. DELTA, a fast robot with parallel geometry. In: Burckhardt, C.W. (Ed.), Proceedings of the 18th International Symposium on Industrial Robots. Springer-Verlag, pp. 91–100.

Collins, C.L., Long, G.L., 1995. On the duality of twist/wrench distributions in serial and parallel chain robot manipulators. In: IEEE International Conference on Robotics and Automation, 1995, pp. 526–531.

Dalvand, M.M., Shirinzadeh, B., 2011. Forward kinematics analysis of offset 6-RRCRR parallel manipulators. Proc. Inst. Mech. Eng. C: J. Mech. Eng. Sci. 255, 3011–3018.

Dalvand, M.M., Shirinzadeh, B., 2012. Kinematics analysis of 6-DOF parallel micromanipulators with offset U-joints: a case study. Int. J. Intell. Mechatronics Robot. 2, 28–40.

Dasgupta, B., Mruthyunjaya, T.S., 2000. The Stewart platform manipulator: a review. Mech. Mach. Theory 35, 15–40.

Dempster, A.P., Laird, N.M., Rubin, D.B., 1977. Maximum likelihood estimation from incomplete data via the EM algorithm. J. R. Stat. Soc. Ser. B 39, 1–38.

Dieudonne, J.E., Parrish, R.V., Bardusch, R.E., 1972. An Actuator Extension Transformation for a Motion simulator and an inverse transformation applying Newton–Raphson's method. Technical Report NASA-TN-D-7067. NASA Langley Research Center, Hampton, VA.

Gao, Z., Zhang, D., Hu, X., Ge, Y., 2010. Design, analysis, and stiffness optimization of a three degree of freedom parallel manipulator. Robotica 28, 349–357.

Gough, V.E., 1956–1957. Contribution to discussion of papers on research in automobile stability, control and tyre performance. In: Proceedings of the Auto Divisional Institute of Mechanical Engineers, pp. 392–394.

Gunn, S.R., 1998. Support Vector Machines for Classification and Regression. Technical Report. School of Electronics and Computer Science, University of Southampton, Southampton, UK.

Karlik, B., Aydin, S., 2000. An improved approach to the solution of inverse kinematics problems for robot manipulators. Eng. Appl. Artif. Intell. 13, 159–164.

Köker, R., 2005. Reliability-based approach to the inverse kinematics solution of robots using Elman's networks. Eng. Appl. Artif. Intell. 18, 685–693.

Liu, K., Fitzgerald, J.M., Lewis, F.L., 1993. Kinematic analysis of a Stewart platform manipulator. IEEE Trans. Ind. Electron. 40, 282–293.

McCallion, H., Pham, D., 1979. The analysis of a six degrees of freedom work station for mechanized assembly. In: 5th World Congress on Theory of Machines and Mechanisms, Montréal, pp. 611–616.

Merlet, J.P., 1993. Closed-form resolution of the direct kinematics of parallel manipulators using extra sensors data. In: Proceedings of IEEE International Conference on Robotics and Automation, 1993, pp. 200–204.

Merlet, J.P., 2004. Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis. Int. J. Robot. Res. 23 (3), 221–235.

Nanua, P., Waldron, K., Murthy, V., 1990. Direct kinematic solution of a Stewart platform. IEEE Trans. Robot. Autom. 6, 438–444.

Panneton, F., L'Ecuyer, P., Matsumoto, M., 2006. Improved long-period generators based on linear recurrences modulo 2. ACM Trans. Math. Softw. 32, 1–16.

Parikh, P.J., Lam, S.S., 2009. Solving the forward kinematics problem in parallel manipulators using an iterative artificial neural network strategy. Int. J.Adv. Manuf. Technol. 40, 595–606.

Peña, D., Prieto, F.J., 2001. Multivariate outlier detection and robust covariance matrix estimation. Technometrics 43, 286–310.

Physik Instrumente, PI. M-850 Hexapod, Technical data available at ⟨http://www.physikinstrumente.com/micropositioningsystems/8_4t.html⟩.

Pollard, W.L.G., 1940. Spray painting machine.

Siciliano, B., 1999. The Tricept robot: inverse kinematics, manipulability analysis and closed-loop direct kinematics algorithm. Robotica 17, 437–445.

Stewart, D., 1965–1966. A platform with six degrees of freedom. Proc. Inst. Mech. Eng. 180, 171–186.

Tarokh, M., 2007. Real time forward kinematics solutions for general Stewart platforms. In: IEEE International Conference on Robotics and Automation, 2007, pp. 901–906.

Vapnik, V., 1995. The Nature of Statistical Learning Theory. Springer-Verlag New York, Inc., New York, NY, USA.

Vapnik, V., Golowich, S.E., Smola, A., 1996. Support vector method for function approximation, regression estimation, and signal processing. In: Advances in Neural Information Processing Systems 9. MIT Press, pp. 281–287.

3.5 Serial robots and the inverse kinematics problem

A serial manipulator can be described as a series of links connected by different types of joints. The most basic types of joints add a revolutive or prismatic degree of freedom, which can be mechanically combined to allow for more complex movements on a single point. The number of joints and links determines the degrees of freedom of a given manipulator, which at the same time, has a direct relationship in determining the capabilities of the serial robot. Generally speaking, the links are assumed to be rigid but flexible joints can also be possible with more complex control strategies (Benosman and Le Vey, 2004). One of the ends can freely move in space, with the other end attached to a reference frame, which can be a specific point on a given robot —fixed or mobile. The former is called the *end effector* and the latter the *ground* or base. In order to perform any task, the location of the end effector relative to the base should be known beforehand. This is called the *position analysis problem*, referring to both the FK and IK problems.

Manipulators of this class are usually modelled using the Denavit–Hartenberg matrices (Denavit and Hartenberg, 1955). Whereas the *direct* or forward problem is straightforward, the inverse problem can be more involved, specially for manipulators with complex geometry. Generally speaking, closed–form solutions can be found for manipulators with simple geometry, such as those with consecutive joint axes and parallel to one another. However, it is considered that the case of the 6–R manipulator of general geometry is the most difficult problem in kinematics, called the *Mount Everest* of the position analysis for this class of mechanisms (Duffy, 1980). For the case of a general 7–R manipulator, Roth et al. (1973) predicted up to 32 solutions, which was confirmed some

years later by Duffy and Crane (1980). Furthermore, it is worth noting than for a redundant serial manipulator there may be several infinitudes of inverse kinematics solutions for a given end effector location, which in some cases prevents some methods to yield proper or complete solutions. Redundancy has advantages when designing more versatile manipulators: it improves the flexibility and versatility of the robot and it allows implementing collisions–free motion in the robotic workspace by using redundant Degrees of Freedom (DOF). Spatial mechanism are one of the most prominent examples of this, where anthropomorphic robotic manipulators are one of the central mechanism where the inverse kinematics problem has been profusely studied.

Closed–form solutions

Analytical or closed–form solutions are always the preferred approach when solving the IK problem, where all possible solutions and manipulator postures can be accounted for. Some popular methods that can be found in the literature for this purpose are (Tsai, 1999):

- 4×4 matrix method (Denavit and Hartenberg, 1955),
- Vector algebra method (Chace, 1963),
- Quaternian algebra method (Yang and Freudenstein, 1964),
- Iterative method (Uicker et al., 1964),
- 3×3 dual matrix method (Yang, 1969),
- Screw algebra method (Yang and Freudenstein, 1971),
- Geometric method (Duffy and Rooney, 1975).

For instance, a simple planar 3–R manipulator, where the three revolutive joint axes are parallel and they are pointing out of the same reference plane, the FK problem can be described with the following system of

equations:

$$\begin{aligned} x &= l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) + l_3 \cos(\theta_1 + \theta_2 + \theta_3) \\ y &= l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) + l_3 \sin(\theta_1 + \theta_2 + \theta_3) \\ \phi &= \theta_1 + \theta_2 + \theta_3 \end{aligned} \tag{3.6}$$

where l_i is the length of the link i and θ_j the angle of the joint j, known values.

For this manipulator, the IK problem can be solved analytically by a simple substitution approach. To solve for θ_i, $i = 1, 2, 3$, given x, y and ϕ, one can define:

$$\begin{aligned} X &= x - l_3 \cos(\theta) \\ Y &= y - l_3 \sin(\theta) \end{aligned} \tag{3.7}$$

where X and Y are known. Squaring and adding, one can get:

$$X^2 + Y^2 = l_1^2 + l_2^2 + 2 l_1 l_2 \cos(\theta_2) \tag{3.8}$$

Then:

$$\theta_2 = \pm \cos^{-1}\left(\frac{X^2 + Y^2 - l_1^2 - l_2^2}{2 l_1 l_2}\right) \tag{3.9}$$

After solving for θ_2 , θ_1 can be found solving the four–quadrant inverse tangent:

$$\theta_1 = \text{arctan2}(Y,\, X) - \text{arctan2}(l_2 \sin(\theta_2)\,,\, l_1 + l_2 \cos(\theta_2)) \tag{3.10}$$

Finally, θ_3 can be obtained from:

$$\theta_3 = \phi - \theta_1 - \theta_2 \tag{3.11}$$

For manipulators with a more complex geometry, a solution like this is not straightforward or even possible. To address these cases, approximate methods are usually employed, specially those based on numerical approximations.

As described in the following contribution, the method presented in previous sections is tailored to be applied for serial kinematic chains (Morell et al., 2013a).

3.6 Inverse kinematics solutions for serial robots using support vector regression

This section includes the full text for the following article, one of the contributions of this thesis presented as a compendium of publications.

- Title: Inverse kinematics solutions for serial robots using support vector regression
- Authors: Antonio Morell, Mahmoud Tarokh and Leopoldo Acosta
- Publication: Proceedings of the 2013 IEEE International Conference on Robotics and Automation (ICRA)
- Year: 2013
- ISSN: 1050–4729
- doi: 10.1109/ICRA.2013.6631171

2013 IEEE International Conference on Robotics and Automation (ICRA)
Karlsruhe, Germany, May 6-10, 2013

Inverse Kinematics Solutions for Serial Robots using Support Vector Regression

Antonio Morell, Mahmoud Tarokh† and Leopoldo Acosta (*Member, IEEE*)
Department of Systems Engineering and Control and Computer Architecture
University of La Laguna, 38203 La Laguna, Tenerife, Spain
email: amorell@isaatc.ull.es, lacosta@ull.es
†Department of Computer Science, San Diego State University,
San Diego, CA 92182–7720, U.S.A.
email: tarokh@cs.sdsu.edu

***Abstract*—Serial kinematic chains are widely used in robotics and computer animation among other fields. Many manipulators do not have closed–form solutions to the inverse kinematics problem, which is of great importance for many applications. In this paper we introduce a fast and accurate procedure which yields all joint angle solutions for a given manipulator or limb posture (position and orientation) and certain swivel angle. By means of a spatial decomposition method, the procedure involves finding accurate models which represent the behavior of the robot or limb in a given workspace region. We propose Support Vector Machines, a very popular machine learning method, as the method that models such behaviors. The performance of the method is tested on the Robotic Research Arm K–1207. The results confirm that the method finds accurate solutions and can be used on real world applications with real–time requirements.**

I. Introduction

We define a *configuration* of a manipulator as a set of joints angle, and a *posture* as the position and orientation of its end effector (hand or foot in a human–like figure). The Inverse Kinematics Problem (IKP) is stated as the process of finding the set of all configurations that result in a given posture in the workspace. Many articulated manipulators with joint offsets do not have closed–form solutions, and the computation of their inverse kinematics is complex and time consuming and therefore unsuitable for real–time applications. Whereas some classes of serial manipulators actually have an algebraic inverse kinematics solution, the analytical expressions have to be defined for each specific morphology [1].

During last decades, several methods for solving the IKP have been proposed, such as Jacobian based methods (pseudoinverse, transpose and damped least squares) [2], genetic algorithms [3], continuation [4] and interval [5] methods. However, some of them do not guarantee all the possible solutions for a given manipulator pose. In addition, they might have convergence problems and high computational requirements, which usually makes them unsuitable for real–time applications. The spatial decomposition method has proven to be suitable for solving kinematics [6] and planning problems [7] which usually have non trivial and complex solutions, if any. It provides simple steps to obtain a model of the behavior of a given robot or limb through its workspace, in order to determine accurate solutions for the inverse kinematics problem for serial manipulators, and similarly, the forward kinematics problem for parallel robots [8]. The most important feature of this method is its ability to yield accurate solutions with a small evaluation time, which enables it to be used in real–time applications.

In this context, this paper presents a spatial decomposition method for obtaining accurate solutions in real–time for the IKP for serial manipulators and kinematic chains, using a popular machine learning method, the *Support Vector Machines* (SVMs), as the regression model. Using SVMs as the modeling tool allows to obtain accurate approximations of the solutions as well as small evaluation times, while the overall complexity of the method decreases. The yielded results are compared with the polynomial method proposed by [6] using the same case study.

This paper is organized as follows. Section II discusses the inverse kinematics problem for serial robots and kinematic chains. Section III describes the first steps of the method, where the workspace of a robot is decomposed into small cells, which are populated with a large amount of configuration and posture data point. Then, these datapoints are classified as described in Section IV, in order to obtain meaningful training data sets for the modeling step with SVMs, that is introduced in Section V. Finally, the evaluation step, which is done *on–line*, is illustrated in Section VI. Some experiments have been performed and are shown in Section VII, where we compare our results with those obtained by the very fast approximation method proposed in [6].

II. IKP on Serial Robots and Kinematic Chains

Serial kinematic chains are present in a wide variety of fields and applications in robotics and computer animation. They are often implemented as 7 Degrees of Freedom (DOF) kinematic chains, modeled similarly to the human arm (or leg), with a 3–DOF spherical joint as a shoulder (hip), another 3–DOF for the wrist (ankle), and a single DOF revolute joint for the elbow (knee) [9]. An example of a typical 7–DOF serial robot is the Robotic Research Arm K–1207 manipulator [10]. The model which represents the configurations and postures for this robot can be described by the Denavit–Hartenberg (D–H)

978-1-4673-5642-8/13/$31.00 ©2013 IEEE

TABLE I
D–H PARAMETERS & JOINT LIMITS OF THE 7–DOF ARM

θ_i	$\bar{\alpha}$	$\bar{a}$	$\bar{d}$	θ_{min}	θ_{max}
θ_1	$0°$	0	0	$-39°$	$+164°$
θ_2	$-90°$	$\bar{a}_1$	0	$-83°$	$+210°$
θ_3	$+90°$	$\bar{a}_2$	$\bar{d}_3$	$-40°$	$+61°$
θ_4	$-90°$	$\bar{a}_3$	0	$-78°$	$+94°$
θ_5	$+90°$	$\bar{a}_4$	$\bar{d}_5$	$-61°$	$+187°$
θ_6	$-90°$	$\bar{a}_5$	0	$-149°$	$0°$
θ_7	$+90°$	$\bar{a}_6$	0	$-59°$	$+78°$

link frame assignment criterion. Table I shows the particular parameters for this manipulator (a bar distinguishes them over other notations used in this work). The length of the upper and lower arm is given by $\bar{d}_3$ and $\bar{d}_5$, while $\bar{a}_1$ through $\bar{a}_6$ represents the shoulder ($\bar{a}_1, \bar{a}_2, \bar{a}_3, \bar{a}_4$), elbow ($\bar{a}_5$) and wrist ($\bar{a}_6$) offsets. The position of the end effector can be obtained from the transformation matrices [11] that solves the forward kinematics of this manipulator. The Cartesian coordinates are as follows:

$$\begin{aligned} x =& \bar{d}_3 \cos\theta_1 \sin\theta_2 + \\ & \bar{d}_5 \left[\sin\theta_4 \left(\cos\theta_1 \cos\theta_2 \cos\theta_3 - \sin\theta_1 \sin\theta_3 \right) + \right. \\ & \left. \cos\theta_1 \sin\theta_2 \cos\theta_4 \right] \\ y =& \bar{d}_3 \sin\theta_1 \sin\theta_2 + \\ & \bar{d}_5 \left[\sin\theta_4 \left(\sin\theta_1 \cos\theta_2 \cos\theta_3 + \cos\theta_1 \sin\theta_3 \right) + \right. \\ & \left. \sin\theta_1 \sin\theta_2 \cos\theta_4 \right] \\ z =& \bar{d}_3 \cos\theta_2 + \\ & \bar{d}_5 \left(\cos\theta_2 \cos\theta_4 - \sin\theta_2 \cos\theta_3 \sin\theta_4 \right) \quad , \end{aligned} \tag{1}$$

where the unknowns are the angles of the shoulder joints $\theta_1, \theta_2, \theta_3$ and the elbow angle θ_4. Since a hand position is defined by three Cartesian coordinates (x, y, z), there is an extra DOF which can be used to define a specific hand posture. This extra parameter can be defined to solve the redundancy, and it is commonly described as the *swivel angle* [9], [12], or aiming angle of the hand, denoted by ψ. It defines the angle of rotation of the plane containing shoulder, elbow and wrist origins about the shoulder–wrist line. For the zero–offset case, the swivel angle ψ [12] is given by

$$\psi = \arctan \frac{\sin\theta_2 \sin\theta_3 \sin\theta_4}{\bar{d}_5 \sin\theta_4 \left(\cos\theta_2 \sin\theta_4 + \sin\theta_2 \cos\theta_3 \left(1 + \cos\theta_4 \right) \right) / Q} \tag{2}$$

with $Q = \sqrt{\bar{d}_3{}^2 + \bar{d}_5{}^2 + 2\bar{d}_3 \bar{d}_5 \cos\theta_4}$. It is noted that with zero offset there is a closed form solution for (x, y, z, ψ). However, the method to be described is applicable to manipulators with non–zero offset.

While the hand position, defined by (x, y, z, ψ), is a function of the first four joint angles, i.e., $\theta_{arm} = (\theta_1, \theta_2, \theta_3, \theta_4)^\top$, the hand orientation (α, β, γ) is related to the wrist joint angles $\theta_{wrst} = (\theta_5, \theta_6, \theta_7)^\top$: there is a closed–form solution for the θ_{wrst} vector for a desired hand position and orientation, i.e., the wrist joint angles can be obtained analytically from the hand rotation angles [11]. Let $R_{ef}(\alpha, \beta, \gamma)$ be the rotation matrix corresponding to the rotation angles of the end effector. Similarly, let $R_{arm}(\theta_{arm})$ be the rotation matrix resulting from the first four joint angles. Then, the rotation matrix of the wrist joint angles can be expressed as

$$R_{wrst}(\theta_{wrst}) = {R_{arm}}^\top(\theta_{arm}) R_{ef}(\alpha, \beta, \gamma) \quad . \tag{3}$$

Using the ZYZ Euler angles convention, the wrist angles can be obtained with

$$\begin{aligned} \theta_5 =& \arctan \frac{r_{23}}{r_{13}} \\ \theta_6 =& \arctan \frac{\sqrt{{r_{31}}^2 + {r_{32}}^2}}{r_{33}} \\ \theta_7 =& \arctan \frac{r_{32}}{-r_{31}} \quad , \end{aligned} \tag{4}$$

where r_{ij} are elements of the known wrist rotation matrix $R_{wrst}(\theta_{wrst})$ defined in (3). Therefore, the IKP can be solved if a solution of the arm joint vector θ_{arm} is found.

In this work, we have solved the IKP by using a well known machine learning method, the Support Vector Regression, by means of a spatial decomposition method, in order to model the behaviour of arbitrary serial kinematic chains. This yields extremely fast solutions, accurate enough for a wide variety of applications.

III. SPATIAL DECOMPOSITION

As described by [6], [8], the spatial decomposition method consists of an *off–line* preprocessing phase and a fast *on–line* evaluation phase. In the off–line phase, the method decomposes the workspace (or pose space) of the robot into small *cells*. Using the known forward kinematics expression (1), it assigns random values within the ranges of the robot joints θ_{arm} and computes the end effector pose (x, y, z, ψ) (position and aiming angle). Data points that fall in a given cell in the pose space have corresponding points in a *region* of the joint space. The relationship between data points in a cell and its corresponding region must be modeled so that when an end effector posture is specified, the sets of all feasible joint angle values can be computed during the on–line phase. The parameters of this relationship are stored in the form of a look–up table, which are then retrieved during the on–line phase.

The method described in [6] for modeling the above mentioned relationship involves fitting a set of polynomial equations to obtain a parametric model which approximates the inverse kinematics solution. In this paper we propose a different approach using a Support Vector Regression (SVR) machine to relate the arm joint angles in each region to the end effector posture vector in the corresponding cell of the pose space. One of the advantages of SVMs is their ability to easily fit complex nonlinear functions. However, there is a tradeoff between the size of the dataset and the parameters

used in terms of training time and accuracy: large training sets will not significantly improve the accuracy of the solutions and will add a training time penalty.

At the same time, fine tunning the parameters will improve the solutions, but will worsen the training time as well. However, the preprocessing step is done only once, so training time is not as important as accuracy and how many SVR machines are stored. Therefore, the cell size in the pose space provides a tradeoff, because it affects the complexity and the number of the machines to be tuned. It is worth noting that using SVR machines allows fewer pose and joint space divisions, since they can model more complex data. Specifically, in the given numerical example described in Section VII, the joint and pose spaces were divided only into 407 regions without a significant accuracy loss, compared with the 23 469 regions needed by the polynomial method for the same level of accuracy.

The workspace or pose space (x, y, z, ψ) is decomposed into several cells in order to obtain accurate models for each one. For prismatic robots, the workspace is best described in Cartesian coordinates, but for revolute joint robots like the one considered in this work, a spherical coordinate system is better suited due to the topological nature of its kinematics. The four–dimensional spherical coordinates of the robot posture $(\rho, \delta, \varphi, \psi)$ are related to the Cartesian coordinates by

$$\begin{aligned} \rho &= \sqrt{x^2 + y^2 + z^2}\,, \\ \delta &= \arctan \frac{y}{x}\,, \\ \varphi &= \arccos \frac{z}{\rho}\,, \end{aligned} \tag{5}$$

where ρ is the radius or distance from the origin, $\delta \in [\,0, 2\pi\,]$ defines the azimuth angle in the $x-y$ plane measured from the x axis, and $\varphi \in [\,0, \pi\,]$ is the polar angle measured from the z axis. The swivel angle $\psi \in [\,0, 2\pi\,]$ is the same for both coordinates systems.

The workspace decomposition involves choosing the cell size for each dimension, denoted by $N_\rho, N_\delta, N_\varphi$ and N_ψ. Higher values on each dimension results in larger cells, i.e., the workspace is divided into less cells. The number of cells directly affects the number of SVR machines to be trained, as well as the accuracy of the solutions found, as previously stated. In addition, the off–line training time and size of the look–up table are also directly related. As shown in Section VII, smaller cell sizes allows for more accurate models of the inverse kinematics.

After the spatial decomposition is performed, each region needs to be populated with a meaningful number of data points, in order to be used as training samples for each SVR machine. Due to joint limits, not all cells within the workspace will contain arm posture data, because data points are generated within to the joint ranges, and hence some regions will not contain any points. Note that regularly spaced joint values will not guarantee a uniform workspace coverage due to the nonlinear mapping of the inverse kinematics. Hence, joint vectors θ_{arm} has to be generated randomly within joint limits. The forward kinematics (1) of each joint vector allows determining the associated posture vector in spherical coordinates, which is used to compute the cell index of the given joint vector. The cells indexes are encoded so that each cell can be addressed by a unique integer value. For a given end effector posture p, the index can be obtained as

$$\text{index}(p) = \left\lfloor \frac{\rho}{N_\rho} \right\rfloor W_\rho + \left\lfloor \frac{\delta}{N_\delta} \right\rfloor W_\delta + \left\lfloor \frac{\varphi}{N_\varphi} \right\rfloor W_\varphi + \left\lfloor \frac{\psi}{N_\psi} \right\rfloor W_\psi, \tag{6}$$

where $W_\rho, W_\delta, W_\varphi, W_\psi$ denotes the position weight of each dimension, as follows:

$$\begin{aligned} W_\rho &= 1 & M_1 &= \left\lceil \frac{\max_\rho}{N_\rho} \right\rceil \\ W_\delta &= M_1 & M_2 &= \left\lceil \frac{\max_\delta}{N_\delta} \right\rceil \\ W_\varphi &= M_1 M_2 & M_3 &= \left\lceil \frac{\max_\varphi}{N_\varphi} \right\rceil \\ W_\psi &= M_1 M_2 M_3 \end{aligned} \tag{7}$$

with $\max_\rho, \max_\delta, \max_\varphi$ being the maximum values for the spherical coordinates dimensions ρ, δ, φ, respectively.

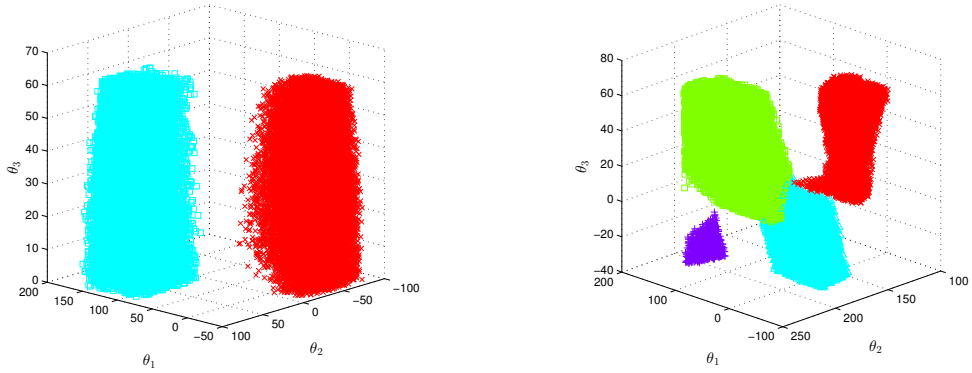
IV. Data Classification

The generated data form clusters in the joint space, which belong to different solutions clusters contained in a specific workspace region. These clusters must be classified before starting the SVR training in order to represent a more accurate model for each solution in a given region.

Regarding the number of solutions, with no joint limits and offsets, it can be shown from (1) and (2) that a given posture (x, y, z, ψ) can be reached with a maximum of four possible joint configurations: $(\theta_1, \theta_2, \theta_3, \theta_4)$, $(\theta_1 + \pi, -\theta_2, \theta_3, -\theta_4)$, $(\theta_1, \theta_2, \theta_3 + \pi, -\theta_4)$ and $(\theta_1 + \pi, -\theta_2, \theta_3 + \pi, \theta_4)$. Joint limits add restrictions to the serial robot movement, so that the number of solutions in some regions is reduced.

The classification procedure for each region starts by fitting a Gaussian Mixture model using an *Expectation Maximization* (EM) algorithm [13] with maximum likelihood estimates. With an iterative process, a Gaussian Mixture model is fitted with k components, and where k iterates in $[\,1, 4\,]$, i.e., the maximum expected number of clusters or solutions. Then, the distance between each cluster found is checked in order to select the value for k that maximazes it. When the best model is found, each datapoint is classified according to its posterior probability: a joint posture belongs to the cluster in which its posterior probability is maximum.

In Fig. 1a, two clusters are easily seen, while in Fig. 1b four solutions have been found within a more complex region. Only the first three joint angles are shown for ease of visualization.



(a) Region with 2 solutions easily separable

(b) Region with 4 solutions, more complex clusters

Fig. 1. Classification examples with multiple solution clusters

V. Inverse Kinematics modeling

Support Vector Machines were formally introduced as a pattern recognition tool by Vapnik and his co–workers [14], and a few years later they were extended to solve regression problems, constructing multidimensional splines and solving linear operator equations [15], [16]. As a statistical learning tool, SVMs were developed with a greater ability to generalize. When solving pattern recognition problems, a SVM is able to find in a feature space a decision rule with good generalization, i.e., the optimal separating hyperplane, by selecting a small subset of the training data, called the *Support Vectors* (SVs). It can be shown that optimal separation of the SVs is equivalent to the optimal separation of the entire data [16]. Maximizing the margin between classes and training samples is an alternative to other optimizing cost function methods, e.g., mean square error. This provides some important features to SVMs, like automatic capacity tuning of the classification function, the representation of the data relevant for the classification by a small subset (data compression), and uniqueness of the solution. Hence, SVMs can be used to summarize the information contained in a data set. The experiments carried out in this work prove that using SVR machines to model the behavior of the manipulator in a given region yields results good enough for a wide variety of applications.

An SVR is a multi–input single–output system, which supports a very large number of inputs as data features, while the output or target value is estimated with a decision function. Since a trained machine will output one estimated value, each joint angle estimation will need a trained SVR model for each arm configuration $\theta_{arm} = (\theta_1, \theta_2, \theta_3, \theta_4)^\top$ using the same data instances. Therefore, a given cluster will need a total of four trained SVR machines in order to obtain a full θ_{arm} estimation.

In other words, the training algorithm for a given SVR machine requires an input vector (θ_i, p), where $p \in \Re^4$ is a pose configuration vector and $\theta_i \in \Re$ is the corresponding value of the *i*–th joint angle vector parameter. The algorithm finds the parameters of a decision function $F(p)$ using these training samples,

$$F(p_i) = \langle w, p_i \rangle + b \quad . \tag{8}$$

The goal of the algorithm is to find the parameter w and the bias b that represents the optimal separating hyperplane, which models the joint parameter θ_i in the given region. The decision function has a dual space representation, which reduces the number of computations required to train the classifier [14]:

$$F(p) = \sum_{i=1}^{l} \alpha_i K(p_i, p) + b \quad , \tag{9}$$

where the coefficients α_i are the parameters found by the algorithm and p_i are the training samples. $K(p_i, p)$ is the kernel function, which allows operations to be performed in the input space rather than in a high dimensional feature space. Extending the formulation to the *dual form* allows the application of different kernel functions, such as polynomials, radial basis functions and certain sigmoid functions.

As a regression tool, a SVR employs an alternative loss function, which includes a distance measure. Vapnik proposed the *ε–insensitive loss function* [15], a more sophisticated penalty tool, which defines a region between $\hat{\theta}_i$, the estimated value, and θ_i, the actual value. Only when $|\theta_i - \hat{\theta}_i| \geq \varepsilon$ does the function add one of two slack variable penalties, depending on whether they lie *above* or *below* a certain region called the *ε–insensitive tube*. The ε parameter has an effect on the smoothness of the response and it affects the number of support vectors, i.e., the generalization capability.

One of the most widely used SVMs implementations is LIBSVM [17]. It is a library under active development, and provides a set of tools that can quickly yield acceptable results. It implements a set of SVM formulations, multi–class classification and probability estimates. Also, four basic kernel

functions are available, namely linear, polynomial, Radial Basis Function (RBF) and sigmoid kernels.

Cluster data must be preprocessed before starting the training process. Link and pose data must be scaled (normalized) to improve the traning time of the Support Vector Machines, as suggested in the libSVM documentation [17]. It also avoids numerical difficulties and the adverse effect of greater numeric ranges against those which happen to be smaller. Both training and testing data must be scaled with the same method and range, e.g., $[-1, 1]$ or $[0, 1]$. In the experiments described in the next Section data is normalized using the former, which is now suitable for being used as SVR training samples.

A simple method to store the trained models is to populate an indexed data structure or look–up table which should allow fast indexation. Given that we can apply (6) to obtain the address of a desired table entry, the structure can be a sequence of elements that contains the trained model parameters corresponding to all solutions. The input position data in the on–line phase must be scaled according to the training data to obtain a valid estimation, so the scaling information used for training is also stored. This completes the off–line phase, which only needs to be executed once for a particular robot.

VI. On–line Inverse Kinematics Evaluation

The models are sequentially stored in a look–up table, sorted by its index or address. Each entry also stores its corresponding cell address, which uniquely identifies it. To obtain the inverse kinematics solution(s) of a given full posture vector $(x, y, z, \psi, \alpha, \beta, \gamma)$, the first step is to apply (5) to transform it to spherical coordinates, $(\rho, \delta, \varphi, \psi)$. Then, its corresponding address can be obtained using (6). Since the stored table entries are sorted according to their index, a method like binary search can be applied. With a worst case of $\lceil \log_2 N \rceil$ at most, where N is the number of regions, looking for the model parameters to all solutions for a given region is extremely fast. Subsequently, the SVR model parameters are used to estimate the joint values of the arm vector $\theta_{arm} = (\theta_1, \theta_2, \theta_3, \theta_4)^\top$. When all solutions found for the given region are estimated, the last joint angles corresponding to the wrist vector, $\theta_{wrst} = (\theta_5, \theta_6, \theta_7)^\top$, are obtained analytically by applying equation (3) to get the associated rotation matrix $R_{wrst}(\theta_{wrst})$, and finally (4) for the joint angles.

It should be noted that the maximum number of solutions in each region for a given full position/orientation vector is eight, i.e., a combination of four due to position and two due to orientation, since a hand orientation $(\alpha, \beta.\gamma)$ can be achieved with two sets of wrist joint angles, $(\theta_5, \theta_6, \theta_7)$ and $\theta_5 + \pi, -\theta_6, \theta_7 + \pi)$, respectively. The most suitable solution depends on the application and the current status of the manipulator or animation character.

As will be described in the next Section, although the accuracy of the approximations obtained by this procedure is enough for most applications, it can be easily improved by a correction step, which involves another inverse kinematics estimation using the SVR models and simple operations, as described in the following steps:

TABLE II
Accuracy test with 1000 pose vectors

$(N_\rho, N_\delta, N_\varphi, N_\psi)$	N	$\lvert E \rvert$	$\lvert E_{cor} \rvert$	LuT size
$(105\,mm, 60^\circ, 30^\circ, 60^\circ)$	407	$0.4822\,mm$	$0.1543\,mm$	105 MB
$(60\,mm, 75^\circ, 45^\circ, 120^\circ)$	238	$0.6414\,mm$	$0.1553\,mm$	74 MB
$(105\,mm, 90^\circ, 60^\circ, 90^\circ)$	96	$0.8168\,mm$	$0.2571\,mm$	44 MB

1) Apply (1) to the estimated $\hat{\theta}_{arm}$ vector to yield its associated pose vector $\hat{p}$.
2) Calculate the difference between the given pose vector p and $\hat{p}$: $\Delta p = p - \hat{p}$.
3) Obtain a corrected pose vector with $p_c = p + \Delta p$.
4) Predict a new set of θ_{arm} joint angles from the corrected p_c, which is a more accurate arm configuration for the initial pose vector.

With these simple steps, joint vector estimation can be greatly improved without adding much overhead to the overall on–line time. The following section will show a numerical example demostrating the procedure.

VII. Results and Discussion

We have applied the proposed method to the model of the Robotic Research manipulator, a human–like arm with 7–DOF [10]. The manipulator model has zero shoulder, elbow and wrist offsets, i.e., $(\bar{a}_i = 0\,, i = 1, \cdots, 6)$, with upper and lower arm lengths $\bar{d}_3 = 334\,mm$ and $\bar{d}_5 = 288\,mm$ respectively, and the joint limits are given in Table I.

The experiments were conducted using a workstation with an Intel Core i5–3550 CPU @ 3.30 GHz, 16 GB of RAM, with Linux kernel version 3.2.0–30–generic and MATLAB R2012A, both 64–bit versions. The most time–consuming steps were the classification and the SVR machines training, which took several hours but only had to be done once. Whereas the implementation of the method was mainly written in MATLAB, the random pose generator was coded in C++ language. The LIBSVM library also provides a MATLAB implementation of its functions with MATLAB *MEX*–files.

Results of the accuracy tests are summarized in Table II. The model accuracy was obtained as the mean position error between the given arm posture vector and the estimated one yielded after applying a series of 1000 random inverse kinematics evaluations. The average evaluation time after these tests was $6.4\,ms$, which is reasonable considering that the correction step requires a second inverse kinematics estimation. If the application does not require such degree of accuracy, the evaluation time can be reduced to $3.1\,ms$. Furthermore, the tests were conducted using MATLAB, so there is considerable room for reducing the on–line time using another language such as C++.

With 407 regions, the resulting look–up table after the completion of the off–line phase occupied about 105 MB of memory. Small cell sizes have an impact on the number of SVR machines to be trained, i.e., the look–up table size will be increased accordingly. We tested the performance with cell

TABLE III
PERFORMANCE WITH DIFFERENT TRAINING PARAMETERS

N	C	γ	$\lvert E \rvert$	$\lvert E_{cor} \rvert$
96	1000	0.1	$1.2145\,mm$	$0.2592\,mm$
96	3000	0.1	$1.1703\,mm$	$0.2161\,mm$
96	2000	0.01	$1.3386\,mm$	$0.3941\,mm$
96	2000	0.5	$1.2080\,mm$	$0.2053\,mm$

sizes $(30\,mm, 15°, 9°, 10°)$. Such partition yielded a huge look–up table since the number of regions, N, was $73\,284$ after the random pose generation process. The average position error was similar to the one reported in [6], and it was not a real accuracy improvement, considering the size of the resulting look–up table. The average error without and with correction, $|E|$ and $|E_{cor}|$, are small enough to satisfy a wide variety of applications, and can be further improved by fine–tuning the parameters of the SVR training if needed. As the dimension of the workspace increases, the number of cells also increase in an exponential manner and more memory will be needed. Finally, it should be noted that there is no orientation error, due to the fact that the wrist joint angles are computed using the closed–form orientation expressions (3) and (4).

SVR machines were trained with the same kernel parameters, using an RBF kernel with parameters $C = 2000$ and $\gamma = 0.1$. The latter controls how spread the gaussians will be; big values yield better results as it increases the fit of irregular decision boundaries, while low values avoid overfitting the models as it implies very smooth boundaries. However, it is a problem dependant parameter, so in this work we have tried to find the value which yielded a better accuracy after a grid search. For the third partition scheme, we tested different combinations of C and γ parameters, which are summarized in Table III. The results show that large values for both parameters decrease accuracy (as can be seen when no correction is applied), as the SVR machines have overfitted the training data, while lower values also have an impact on accuracy, as regression function boundaries are smoother.

VIII. CONCLUSION

In this paper we have developed a new approach for computing fast and accurate inverse kinematics solutions. This approach, based on a spatial decomposition method, consists of taking advantage of the higher modeling capabilities of Support Vector Machines for regression. Experiments show that the behavior of a serial robot or limb can be accurately modeled with fewer regions compared to the previous method, since using SVR machines provides more complex models which accurately describe larger partitions of the manipulator's workspace. Since there are fewer entries in the look–up table, the binary search is faster. Furthermore, this method is not only well suited to manipulators with 7–DOF, but it can be applied to other types of kinematic chains in a straightforward manner. The complexity of the kinematics chain, e.g., existence of joint offset, does not affect the procedure since the SVM uses only the known forward kinematics equations.

Applications with high accuracy requirements can also benefit from using this method with a simple correction step. The estimation of the joint angles can be easily improved without adding much overhead on the computation, since it requires only one more joint vector estimation and few simple algebraic operations. The evaluation time in both cases is fast enough for real–time applications, using optimized code and a suitable language.

ACKNOWLEDGMENT

The authors gratefully acknowledge the contribution of the Spanish Ministry of Economy and Competitiveness under Project SAGENIA DPI2010-18349.

REFERENCES

[1] D. Manocha and Y. Zhu, "A Fast Algorithm and System for the Inverse Kinematics of General Serial Manipulators," in *IEEE Int. Conf. Robot. Autom.*, 1994, pp. 3348–3354.
[2] K. Tchon, "Optimal Extended Jacobian Inverse Kinematics Algorithms for Robotic Manipulators," *IEEE Trans. Rob.*, vol. 24, no. 6, pp. 1440–1445, Dec. 2008.
[3] F. Chapelle and P. Bidaud, "A Closed Form for Inverse Kinematics Approximation of General 6R Manipulators using Genetic Programming," in *IEEE Int. Conf. Robot. Autom.*, vol. 4, May 2001, pp. 3364–3369.
[4] L. W. Tsai and A. P. Morgan, "Solving the Kinematics of the Most General Six and Five Degree of Freedom Manipulators by Continuation Methods," in *Trans. ASME J. Mech., Transmission Autom. Design*, vol. 107, 1985, pp. 189–200.
[5] A. Castellet and F. Thomas, "Towards an Efficient Interval Method for Solving Inverse Kinematic Problems," in *IEEE Int. Conf. Robot. Autom.*, 1997, pp. 3615–3620.
[6] M. Tarokh and M. Kim, "Inverse Kinematics of 7–DOF Robots and Limbs by Decomposition and Approximation," *IEEE Trans. Robot. Autom.*, vol. 23, no. 3, pp. 595–600, Jun. 2007.
[7] A. Hourtash and M. Tarokh, "Manipulator Path Planning by Decomposition: Algorithm and Analysis," *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 842–856, Dec. 2001.
[8] M. Tarokh, "Real Time Forward Kinematics Solutions for General Stewart Platforms," in *IEEE Int. Conf. Robot. Autom.*, Apr. 2007, pp. 901–906.
[9] J. Zhao and N. I. Badler, "Inverse Kinematics Positioning Using Nonlinear Programming for Highly Articulated Figures," *ACM Trans. Graph.*, vol. 13, no. 4, pp. 313–336, Oct. 1994.
[10] K. Kreutz, M. Long, and H. Seraji, "Kinematic Functions for the 7–DOF Robotics Research Arm," in *Proceedings of the NASA Conference on Space Telerobotics*, vol. 1, Jan. 1989, Technical Report, pp. 39–48.
[11] J. J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed. Boston, MA, USA: Addison–Wesley Longman Publishing Co., Inc., 2005.
[12] Seraji, H. and Long, M. K. and Lee, T. S., "Motion Control of 7–DOF Arms: The Configuration Control Approach," *IEEE Trans. Robot. Autom.*, vol. 9, no. 2, pp. 125–139, Apr. 1993.
[13] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum Likelihood Estimation From Incomplete Data via the EM Algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.
[14] B. E. Boser, I. Guyon, and V. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Proceedings of the 5th annual ACM Workshop on Computational Learning Theory*. ACM Press, 1992, pp. 144–152.
[15] V. Vapnik, *The nature of statistical learning theory*. New York, NY, USA: Springer–Verlag New York, Inc., 1995.
[16] V. Vapnik, S. E. Golowich, and A. Smola, "Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing," in *Advances in Neural Information Processing Systems 9*. MIT Press, 1996, pp. 281–287.
[17] C.-C. Chang and C.-J. Lin, "LIBSVM: A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

3.7 Results and discussion

The proposed methodology improves some previous results in terms of accuracy of the solution. It also demonstrated to be more general in terms of the morphology of the manipulator. Additionally, it does not have high memory requirements to store the model parameters, tightly related with the implementation used to model SVR machines. Related with the implementation, even with the real–time performance obtained, there are room for optimizations on some steps. Combining this with a more suitable coverage of the pose space, training data generated should yield a better representation of the input space, potentially improving accuracy.

On the classification step, a classic Expectation Maximization (EM) algorithm was employed. While it demonstrated to be suitable for separating different solutions that may appear inside some regions, special attention has to be paid in order to increase the success rate of this approach. A heuristic was developed to select the number of solutions in these cases. Overestimating this value does not suppose a real issue, since this just supposes a fragmentation of models for the same solution. However, underestimating this number can yield a lower number of solutions than the proper one for a given configuration, which supposes a loss of information.

For the case of a serial manipulator, it is worth noting that a higher number of regions were needed to achieve similar accuracy levels than the reference method. This would indicate that the yielded nonlinear models are not properly capturing the complexity compared with the results obtained in the case of the parallel manipulators investigated. Aside from a higher training time and a bigger look–up table —twice as big as the one for the parallel case —, the impact on the methodology

is not critical. However, this also would indicate the possibility to find a different combination of training parameters, another kernel function or even another estimation method rather than a general SVR machine.

All together, I consider that this methodology can be applied to a wide variety of morphologies with just some fine tuning, yielding good enough performance and accuracy, which can be improved with alternative data generation approaches and a more suitable approach for training parameter selection.

"Deserves it! I daresay he does. Many that live deserve death. And some that die deserve life. Can you give it to them? Then do not be too eager to deal out death in judgement. For even the very wise cannot see all ends."

— J.R.R. Tolkien, The Fellowship of the Ring

4

Fractional Differential Equations and Dynamical Systems

EXPONENTIAL laws are the classical approach to study dynamical systems using Ordinary Differential Equations (ODEs) on a wide variety of fields. However, several natural processes and physical systems have been found to exhibit faster or slower dynamics than those provided by models based on pure exponential laws. In 1903, the *Mittag–Leffler function* E_α, a direct generalization of exponential functions, was introduced (Mittag-Leffler, 1903). The $E_\alpha(z)$ function is defined by

$$E_\alpha(z) := \sum_{k=0}^{\infty} \frac{z^k}{\Gamma(\alpha k + 1)} \quad (z \in \mathbb{C}; \quad \Re(\alpha) > 0) \tag{4.1}$$

At the same time, it satisfies the main properties of exponential functions. For instance, when $\alpha = 1$, $E_1(z) = e^z$. In spite of not being the only function that exhibit these properties, it has become one of the most

used by practitioners and scientists when modelling systems with a physical meaning.

This function has been one of the most important tools that enabled the application of Fractional Differential Equations (FDEs) on complex systems. FDEs are the foundation of Fractional Calculus (FC), a set of tools built around the generalization of derivatives and integrals to any arbitrary real or complex order (Oustaloup, 1995; Podlubny, 1999a). FC describes several useful tools for solving differential and integral equations, and various other problems involving special functions of mathematical physics. These tools have been used to model complex systems on a wide variety of applications, including super–diffusion processes, fluid flow, electrical networks, probability and statistics, control theory of dynamical systems, viscoelasticity, signal processing and many others. (Baleanu et al., 2010; Caponetto et al., 2010; Kilbas et al., 2006; Liu et al., 2003; Méhauté et al., 2005; Podlubny, 1999a; Samko et al., 1993; Tenreiro-Machado and Luo, 2008; Tenreiro-Machado, 2002; Tenreiro-Machado and Barbosa, 2008; Tenreiro-Machado et al., 2011).

Some of the first applications were developed by Abel and Boole during the 19^{th} century. Then, it started to become a popular topic during the first decades of the past century. FC has been applied to almost all areas of classical analysis, which is not surprising considering that it extends two of the most basics operators of mathematics —differentiation and integration.

Fractional–order models take advantage of being more suitable for describing nonlinear dynamics on complex systems (Hilfer, 2000; Kilbas et al., 2006; Oldham and Spanier, 1974). While classical operators are still suitable and powerful for modelling some important dynamical processes, complex systems with non–local dynamics and long–memory in time hugely benefit from the application of operators based on FDEs. In contrast, classical derivation and integration are local operators, not

depending on their past values. Specifically, some processes that exhibit anomalous diffusion —*particles* following nonlinear functions with respect of time — and macroscopic complex behaviours cannot be properly characterized by classical integer–order models. Furthermore, some complex systems have only been able to be physically explained after the application of FDEs.

One of the clear examples is the *Brownian motion*, which models the path of a particle subject to complex dynamics and serves as a model of the physical behaviour of normal and anomalous diffusion processes. Some systems based on natural processes can be modelled as a *random walk* on a two–dimensional plane, where each particle randomly jumps at an also random pace (Feller, 1968). When the jumps does not follow a Gaussian propagator, more complex processes can be described by using the so–called Continuous Time Random Walk (CTRW) (Metzler and Klafter, 2000).

Applications to other nonlinear dynamical systems like chaotic systems and robotics have also demonstrated some of the benefits that the fractional calculus introduces. Some of them will be introduced in the following sections.

4.1 Fractional operators

In this section we introduce the fractional operators and definitions relevant to this dissertation. An extended and more complete description of the fractional operators, FDE and fractional–order control has been developed by Samko et al. (1993), Podlubny (1999a) and Kilbas et al. (2006) and references therein.

The Riemann–Liouville fractional integral $I^{\lambda}_{a^+} f$ of order λ, with $\lambda \in \mathbb{R}^+$, of a suitable real function $f(t)$, probably one of the most used defi-

nitions, can be expressed as

$$(I^{\lambda}_{a^+} f)(t) \triangleq \frac{1}{\Gamma(\lambda)} \int_a^t \frac{f(\tau)}{(t-\tau)^{1-\lambda}} d\tau \tag{4.2}$$

where $a \in \mathbb{R}$, $t > a$ and $\Gamma(\cdot)$ is the Euler Gamma function. Conversely, the Riemann–Liouville fractional derivative $D^{\mu}_{a^+}$ of order μ, with $n-1 < \mu \leq n$, of a function $f(t)$, is defined as

$$D^{\mu}_{a^+} f(t) \triangleq D^n (I^{1-\mu}_{a^+} f)(t) \tag{4.3}$$

Another useful definition of the fractional derivative was given by Caputo, as follows

$${}^{c}D^{\mu}_{a^+} f(t) \triangleq (I^{1-\mu}_{a^+} (D^n f))(t) \tag{4.4}$$

where $D \equiv \frac{d}{dt}$. Furthermore, the Grünwald–Letnikov's definition of the fractional derivative is also relevant, and is given by

$$D^{\mu}_{a^+} f(t) = \lim_{h \to 0} \frac{1}{h^n} \sum_{j}^{(t-a)/h} (-1)^j \binom{\mu}{j} f(t - jh) \tag{4.5}$$

where

$$\binom{\mu}{j} = \frac{\Gamma(\mu+1)}{\Gamma(j+1)\Gamma(\mu-j+1)} \tag{4.6}$$

For simplicity, we will consider $D^{\mu}_{0^+} \equiv D^{\mu}$ and $I^{\lambda}_{0^+} \equiv I^{\lambda}$.

For a suitable real function $f(t)$, the Laplace transform of the fractional–order integral, as described by Podlubny (1999a) and Kilbas et al. (2006),

assuming zero initial conditions, is

$$\mathcal{L}\{I^{\lambda} f(t)\} = \int_0^{\infty} \mathrm{e}^{-st} \left(\frac{1}{\Gamma(\lambda)} \int_a^t \frac{f(\tau)}{(1-\tau)^{1-\lambda}} d\tau \right) dt = \\ = \frac{1}{s^{\lambda}} F(s) \tag{4.7}$$

Similarly, the Laplace transform of D^{μ}, also assuming zero initial conditions, is given by

$$\mathcal{L}\{D^{\mu} f(t)\} = s^{\mu} F(s) \tag{4.8}$$

where the integer n lies within $n-1 < \mu \leq n$.

With these fractional operators, we can introduce the definition of a system of FDEs with initial conditions as

$$\begin{cases} {}^{c}D^{\mu}_{a^+} x(t) = f\big(t, x(t)\big), \ m-1 < \mu < m \in \mathbb{Z}^+, \ t > 0 \\ {}^{c}D^{k}_{a^+} x(t)\Big|_{t=0} = x_0^k, \ k = 0, 1, 2, \ldots, (m-1) \end{cases} \tag{4.9}$$

where the fractional derivative is the Caputo's definition, $x \in \mathbb{R}^n$, the function $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$ and $\mu = (\mu_1 \mu_2 \ldots \mu_n)^T$.

Furthermore, using the Riemann–Liouville fractional derivative, a fractional–order Linear Time–Invariant (LTI) causal system is defined as

$$\sum_{k=0}^{n} a_k D^{\alpha_k} y(t) = \sum_{k=0}^{m} b_k D^{\beta_k} u(t) \ , \ n \geq m \tag{4.10}$$

where orders $\alpha_k, \beta_k \in \mathbb{R}$. If the orders of derivation α_k and β_k can be expressed as multiples of a given value $\lambda \in \mathbb{R}^+$, the system is said to be

of *commensurate* order, and $\alpha_k = \beta_k = k\lambda$. Thus, equation 4.10 becomes

$$\sum_{k=0}^{n} a_k D^{k\lambda} y(t) = \sum_{k=0}^{m} b_k D^{k\lambda} u(t) \ , \ n \geq m \tag{4.11}$$

with $k \in \mathbb{N}$. Applying the Laplace transform, the transfer–function of a fractional–order continuous–time system can be expressed as

$$G(s) = \frac{Y(s)}{U(s)} = \frac{\sum_{k=0}^{m} b_k (s^\lambda)^k}{\sum_{k=0}^{n} a_k (s^\lambda)^k} \tag{4.12}$$

assuming zero initial conditions.

4.2 Fractional–order control and the tracking problem

Dynamically tracking and following a given target is one of the basic tasks a mobile robot can typically perform. Depending on the environment and the dynamics of the robot, the solution to this problem involves modelling and controlling complex dynamics. Assuming a properly parametrized target, this task involves the control of the traction system for a self–driving car. For the case of electric motors, the system is controlled by an input voltage and the output is the rotational speed of the rotor. This is ultimately translated into the position of the mobile robot with respect to the target. The self–driving car Verdino is an example of a mobile robot that can benefit from an accurate control method in order to keep a fixed distance when tracking a target. In this case, the dynamic target or objective can be a pedestrian or another vehicle, which also can freely move. In the following, this context will be used to

describe a control method applied to this problem, for which a related contribution will be presented.

In Control Engineering (Ogata, 2010), the tracking problem can be described in terms of the *velocity* or *position* of the output. Whereas the physical form of the output can vary depending on the nature of the system which will be controlled, we will consider that the velocity and position of the output are directly related with the corresponding velocity and position of a mobile robot. The problem is stated in classic Control Theory as finding the instantaneous control action $u(t)$ that makes a closed–loop error function $e(t)$ become zero for a given input function $r(t)$. Some authors, Petráš (2009), Monje et al. (2010) and Tenreiro-Machado (2010) for instance, have proposed different approaches to solve tracking problems when the input is a step function. However, a ramp is a more suitable input function to model a control system that keeps a fixed distance between the robot and the target: the slope can be dynamically adjusted to fit different path planning or navigation strategies. In the following, we will model the input using a unit–ramp.

The general expression for these transfer–functions is given by

$$G(s) = \frac{Y(s)}{U(s)} = K \frac{\prod_{k=0}^{m} (s + b_k)}{\prod_{k=0}^{n} (s + a_k)} \tag{4.13}$$

where $Y(s)$ and $U(s)$ are the output and input of the system, respectively. For a given controller $c(t)$, a closed feedback control loop can be described as shown on figure 4.1, where each system is represented by its related transfer–function, after applying the Laplace transform in order to obtain the representation in the frequency domain. We assume zero initial conditions for all of these functions. $C(s)$ is the transfer–function of the controller and $G(s)$ is the dynamic model of the system, e.g., the

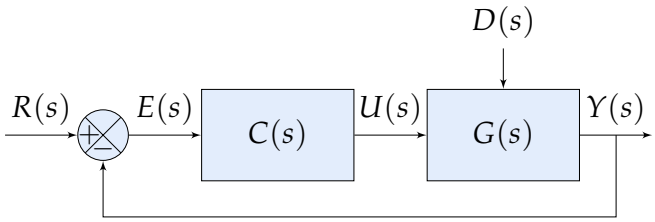


Figure 4.1 Feedback control loop.

Direct Current (DC) electric motor for our self–driving car, Verdino. Considering ideal conditions (no added disturbances, $D(S) = 0$), if the error function $E(s) = 0$ then the output $Y(s)$, which represents the distance or *position* with respect to the target, will follow the input $R(s)$. The Laplace transform of a unit–ramp, $r(t) = t$, is $R(s) = 1/s^2$, which we will use to model the input in this problem. Thereby, the dynamic model of an electric motor can be described as the second–order LTI transfer–function

$$G(s) = \frac{1}{(\tau_1 s + 1)(\tau_2 s + 1)} \tag{4.14}$$

describing the dynamics between the output and the input, with $\tau_1, \tau_2 \neq 0$. These parameters are usually obtained applying a parameter identification process to the real system. This class of transfer–functions are said to be of Type–0 (no roots at the origin).

Proportional–Integral–Derivative (PID) controllers of integer–order are commonly used to design suitable transfer–functions $C(s)$ that stabilize feedback control loops like equation 4.14. The fractional or non–integer generalization of the PID controller is expressed as $PI^{\lambda}D^{\mu}$, typically with $0 < \lambda, \mu < 1 \in \mathbb{R}$ (Podlubny, 1999b). The expression in the time domain for the control action $u(t)$ with respect to the error function

$e(t)$ is

$$u(t) = K_p e(t) + K_i D^{-\lambda} e(t) + K_d D^{u} e(t) \tag{4.15}$$

with a transfer–function of the type

$$C(s) = K_p + \frac{K_i}{s^{\lambda}} + K_d s^{\mu},\ \lambda, \mu > 0 \tag{4.16}$$

For systems that can exhibit delay, derivative actions are not recommended when designing the realization of a controller. Thus, to control a system like a DC motor, PI controllers are preferred. Whereas these controllers are designed to yield a low steady–state error, when the input is a ramp function, it is well known that a PI controller

$$C(s) = K_p + K_i/s \tag{4.17}$$

with an integer–order integral action —($\lambda = 1$ — is unable to yield a tracking error equal to zero.

Let $E(s)$ be the error function on a feedback control loop, consisting of an integer–order Proportional–Integral (PI) controller (equation 4.17), connected with a LTI system $G(s)$ as in figure 4.1. The transfer–function will be given by

$$E(s) = \frac{1}{1 + k_p G(s) + K_i \frac{G(s)}{s}} R(s) \tag{4.18}$$

When the input is a ramp function, the transfer–function $R(s)$ is

$$R(s) = \frac{r}{s^2} \tag{4.19}$$

where $r \in \mathbb{R}$ is the slope. Therefore, the steady–state error E_{ss}, applying

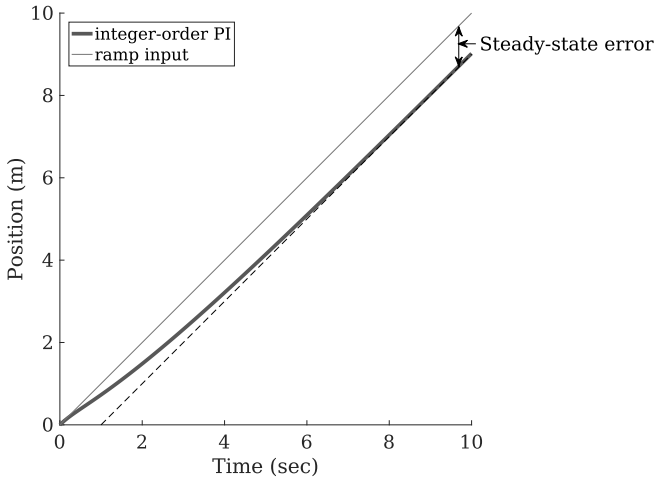


Figure 4.2 Finite steady–state error for a Type–1 system with a ramp input.

the Final–Value Theorem, can be obtained with

$$E_{ss} = \lim_{s\to 0} s\,E(s) = \tag{4.20}$$

$$= \lim_{s\to 0} s\,\frac{1}{1 + k_p G(s) + K_i \frac{G(s)}{s}}\,\frac{r}{s^2} \tag{4.21}$$

$$= \lim_{s\to 0} \frac{r}{s\big(1 + K_p G(s)\big) + K_i G(s)} \tag{4.22}$$

The steady–state gain for a Type–0 transfer–function $G(s)$ as in equation 4.13 can be obtained as

$$\lim_{s\to 0} G(s) = \lim_{s\to 0} K\,\frac{\prod_{k=0}^{m}(s + b_k)}{\prod_{k=0}^{n}(s + a_k)} = K\,\frac{\prod_{k=0}^{m} b_k}{\prod_{k=0}^{n} a_k} = K_g \tag{4.23}$$

where K_g is a constant value. Solving the limit in equation 4.22 applying

by equation 4.23 leads to the steady–state error

$$E_{ss} = \frac{r}{K_i K_g} = K_{ss} \tag{4.24}$$

where the constant value $|K_{ss}| > 0$.

Figure 4.2 shows the transient and steady–state values for a Type–I system with an integral–order PI controller and an unit–ramp function as the input. It can be seen that after the transient, the error in position becomes constant. A design method that yields a fractional–order controller suited for this problem is presented in the following contribution (Morell et al., 2013c).

4.3 Design of a fractional–order controller for the setpoint ramp tracking problem

This section includes the full text for the following article, one of the contributions of this thesis presented as a compendium of publications.

- Title: Design of a fractional–order controller for the setpoint ramp tracking problem
- Authors: Antonio Morell, Juan J. Trujillo, Margarita Rivero and Leopoldo Acosta
- Publication: Proceedings of the 2013 American Control Conference (ACC)
- Year: 2013
- ISSN: 0743–1619
- doi: 10.1109/ACC.2013.6580492

2013 American Control Conference (ACC)
Washington, DC, USA, June 17-19, 2013

Design of a Fractional – Order Controller for the Setpoint Ramp Tracking Problem

A. Morell[†], J.J. Trujillo[*], M. Rivero[§] and L. Acosta[†] (*IEEE Member*)

***Abstract*— It is well known that a classical PI controller cannot accurately follow a ramp setpoint. A real world situation which shows this behavior could be a system composed of a direct current motor whose input setpoint is an acceleration/deceleration ramp. In this paper, we propose the design of a fractional PI^{λ} controller, with a fractional–order parameter $\lambda > 1$, as a well suited approach to solve this problem. The design methodology is frequency–based, which allows finding the corresponding fractional controller parameters in order to guarantee stability, robustness and zero steady–state error.**

I. Introduction

Fractional calculus is the generalization of derivatives and integrals to an arbitrary order λ, which can be a real or even complex number. In recent decades, the growing interest in the applications of fractional models in many different fields of applied sciences and engineering is well known. Specifically, it has been applied to model the dynamics of processes with complex behaviors. For instance, [1]–[5] and their references are a representative selection of relevant publications. In particular, the fractional generalization of the PID control scheme $PI^{\lambda}D^{\mu}$ (with I^{λ} the fractional integral of order λ and D^{μ} the fractional derivative of order μ) has shown a better performance in both time and frequency domains than its classical counterpart on some applications. For instance, the reader can find different sources of information of important documents, tools and events in the field of fractional calculus [4]. For further references we also suggest [6]–[17].

Nowadays, even with all the efforts and achievements in this field, there still are theoretical problems and applications which need to be addressed. In this work the tracking problem of a ramp setpoint is faced. It is well known that an integer–order PI controller, the steady–state error of a given system is a constant value. This paper proposes a control loop design using fractional operators, in which the theoretical steady–state error is zero.

Although some authors have addressed speed tracking problems with step setpoints [14], [18], [19], our interest is related with ramp setpoints associated with speed control actions. In order to validate this proposition, we have applied a fractional PI^{λ} controller to improve the speed control for the DC motor installed on the VERDINO [20] prototype, an electric golf cart modified by our group at University of La Laguna. In adittion, our method simplifies the drawbacks of complex models, by the description of suitable tuning rules. Given some design specifications, such rules can be used to obtain the parameters of a fractional PI^{λ} controller, so that the system yielded by this method verifies these stability and robustness conditions.

This paper is organized as follows: Section II presents definitions and properties for some fractional operators and $PI^{\lambda}D^{\mu}$ controllers. Section III describes the real platform on which the experiments has been conducted, and justifies the PI^{λ} controller's choice. In Section IV, the proposed tuning rules needed to obtain the controller parameters are presented. Then, the method is applied to the real prototype in Section V, where stability and robustness are analyzed by means of frequency representation, and relevant results which show the better behavior of the proposed method are reported. Finally, the main conclussions are summarized in Section VI.

II. Fractional Control Systems

A. Fractional Operators

In this Section we describe some definitions and properties of the fractional operators which are applied in this paper. For further details please see [5], [8], [21].

Let $f(t)$ be a suitable real function. The definition of the fractional integral of $f(t)$ is

$$(I^{\lambda}_{a^+} f)(t) \triangleq \frac{1}{\Gamma(\lambda)} \int_a^t \frac{f(\tau)}{(t-\tau)^{1-\lambda}} d\tau, \tag{1}$$

where $a \in \mathbb{R}$, $t > a$, λ is the real positive integration order (with $n-1 < \lambda \leq n, n \in \mathbb{N}$), and $\Gamma(\lambda)$ is the Euler Gamma function.

The Laplace transform of a function is very important in many applications in engineering. For a suitable function $f(t)$, the Laplace transform of the integral is given by [5], [8]

$$L\{I^{\lambda} f(t)\} = \int_0^{\infty} \mathrm{e}^{-st} \left(\frac{1}{\Gamma(\lambda)} \int_a^t \frac{f(\tau)}{(1-\tau)^{1-\lambda}} d\tau \right) dt = \tag{2}$$

$$= \frac{1}{s^{\lambda}} F(s),$$

with $I^{\lambda}_{0^+} \equiv I^{\lambda}$ and zero initial conditions. See for example [5], [8].

The authors gratefully acknowledge the contribution of the Spanish Ministry of Economy and Competitiveness under Projects SAGENIA DPI2010–18349 and MTM2010–16499.

[†] *Department of Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores (ISAATC), Universidad de La Laguna, 38271 La Laguna, Tenerife, Spain* (e–mail: amorell@isaatc.ull.es , lacosta@ull.es)

[*] *Department of Análisis Matemático, Universidad de La Laguna, 38271 La Laguna, Tenerife, Spain.* (e–mail: jtrujill@ullmat.es)

[§] *Department of Matemática Fundamental, Universidad de La Laguna, 38271 La Laguna, Tenerife, Spain.* (e–mail: mrivero@ullmat.es)

978-1-4799-0176-0/$31.00 ©2013 AACC

The Riemann–Liouville definition for the fractional derivative of order μ, with $n-1<\mu\leq n$, for a function $f(t)$ is defined as

$$D^{\mu}_{a^+}f(t) \triangleq D^n(I^{1-\mu}_{a^+}f)(t), \tag{3}$$

while the Caputo fractional derivative is given by

$${}^cD^{\mu}_{a^+}f(t) \triangleq (I^{1-\mu}_{a^+}(D^n f))(t), \tag{4}$$

where $D\equiv\frac{d}{dt}$. The Grünwald–Letnikov's definition is

$$D^{\mu}_{a^+}f(t)=\lim_{h\to 0}\frac{1}{h^n}\sum_{j}^{(t-a)/h}(-1)^j\binom{\mu}{j}f(t-jh), \tag{5}$$

where

$$\binom{\mu}{j}=\frac{\Gamma(\mu+1)}{\Gamma(j+1)\Gamma(\mu-j+1)}. \tag{6}$$

It can be shown that the above definitions are equivalent for a wide class of functions [8]. As seen with the integral operator, we will use the nomenclature $D^{\mu}_{0^+}\equiv D^{\mu}$.

Fractional differentiation has a memory property, that is, the value of the fractional derivative function at time t depends on its past values, while the classical integer–order differentiation is a local operator.

Finally, the Laplace transform of D^{μ} is given by

$$L\{D^{\mu}f(t)\}=s^{\mu}F(s), \tag{7}$$

when $n-1<\mu\leq n$, and

$$f(0)=f'(0)=\ldots=f^{(n-1)}(0)=0. \tag{8}$$

B. $PI^{\lambda}D^{\mu}$ controllers

A non–integer linear time–invariant system can be represented as follows:

$$\sum_{k=0}^{n}a_kD^{\alpha_k}y(t)=\sum_{k=0}^{m}b_kD^{\beta_k}u(t)\ ,\ n\geq m\ , \tag{9}$$

where $\alpha_k,\beta_k\in\mathbb{R}$. They can be classified according to their orders of derivation, α_k and β_k: if they cannot be expressed as multiples of a certain λ value, we can say that the system is *non–commmensurate*. Likewise, if they can be represented as a term $k\lambda$, with $k=0,1,2,\cdots$, the system is said to be of *commensurate* order. In the latter case, the value λ can generally be a rational or irrational number, but in this paper we will consider it to be the former. Thus, (9) becomes:

$$\sum_{k=0}^{n}a_kD^{k\lambda}y(t)=\sum_{k=0}^{m}b_kD^{k\lambda}u(t)\ ,\ n\geq m\ . \tag{10}$$

Applying the Laplace transform (assuming zero initial conditions) to (10) we can obtain the continuous–time transfer function

$$G(s)=\frac{Y(s)}{U(s)}=\frac{\sum_{k=0}^{m}b_k(s^{\lambda})^k}{\sum_{k=0}^{n}a_k(s^{\lambda})^k}, \tag{11}$$

It should be noted that a complex variable function like

$$F(s)=\sum_{k=0}^{n}a_ks^{\lambda_k} \tag{12}$$

Fig. 1. VERDINO prototype, a fully electric two seat golf cart.

is multi–valued. Its domain can be seen as a Riemann surface [22], [23], with a finite number of sheets when $\forall k,\lambda_k\in\mathbb{Q}^+$. The q sheets of the Riemann surface, with $\lambda=1/q$, are determined by

$$s=|s|e^{j\phi},\quad (2k+1)\pi<\phi<(2k+3)\pi, \tag{13}$$

where $k=-1,0,\cdots,q-2$. Following the approach showed in [14], only the roots of the principal sheet are meaningful, since (starting from the second sheet), they are always related to monotonically decreasing functions. Therefore, they can be safely ignored.

In 1999 Podlubny [9] proposed a generalization of the classical PID controller known as $PI^{\lambda}D^{\mu}$, typically with $0<\lambda,\mu<1$, but in general $\lambda,\mu>0$. The integer cases correspond to the classical PID family of controllers, i.e., P, PD, PI and PID cases. In short, selecting a fractional order for each action allows controlling the tradeoffs between positive and negative effects of the classical derivative and integral actions, obtaining more flexible control solutions to new and already existing problems. The generalized expression in the time domain of the $PI^{\lambda}D^{\mu}$ controller is

$$u(t)=K_pe(t)+K_iD^{-\lambda}e(t)+K_dD^{u}e(t). \tag{14}$$

The transfer function of such controller is given by the Laplace transform with zero initial conditions:

$$G_c(s)=K_p+\frac{K_i}{s^{\lambda}}+K_ds^{\mu},\quad \lambda,\mu>0. \tag{15}$$

C. Stability considerations

Graphical or geometrical techniques of complex analysis based on the Cauchy's argument principle allow evaluating the closed–loop stability. The well–known Nyquist path can be used to determine the stability of a fractional–order system. Specifically, in closed–loop analysis, a system will be unstable if the path encloses the critical point $(-1,0)$, and stable otherwise. A similar procedure, described in [14], applies the mapping $\delta=s^{\lambda}$, to evaluate the stability properties by means of a contour defined inside a certain δ–plane.

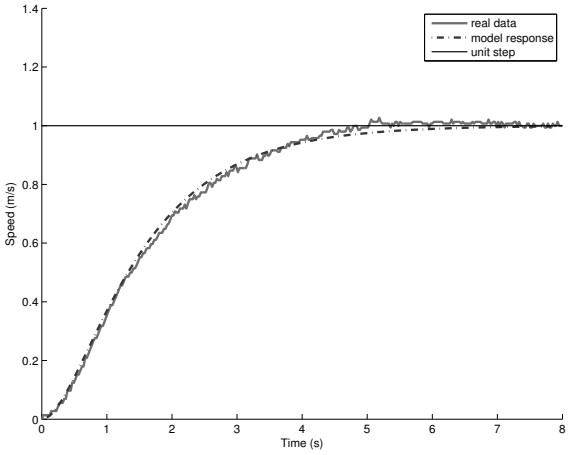


Fig. 2. Step response for real system and LTI model.

III. Speed Tracking Problem

A. Experimental model

Fig. 1 shows the test platform where the experiments have been conducted, VERDINO [20], a low–cost golf cart (a modified *EZ–GO TXT*–2). This fully electric two seat vehicle is a well suited experimental framework for developing and testing autonomous and self–driven systems. It is designed for passenger transportation and surveillance in non–structured environments. The vehicle has been modified by adding several sensors and actuators, which enable it to navigate by itself through urban areas, and assist the top–level software in many tasks, including trajectory tracking. In the following a controller which is able to accurately track a velocity ramp is presented.

Fig. 2 shows the step response of VERDINO's DC motor and the resulting model after a parameter identification process. Such model is described by the following transfer function:

$$G(s) = \frac{1.846}{s^2 + 3.053s + 1.846} = \frac{1}{(\tau_1 s + 1)(\tau_2 s + 1)}, \quad (16)$$

where $\tau_1 = 1.204$ and $\tau_2 = 0.45$ are the time constants.

This LTI model has been obtained after a parameter identification process, as an Autoregressive Model with Exogenous inputs (*ARX*). The system has been modeled using the DC motor voltage as the input, and taking the output from an optical encoder, which measures the rotational speed of the motor.

It is well known that a classical PID does not guarantee a zero tracking error against ramp setpoints for this kind of systems. In the next Section we propose a fractional PI^λ controller, with $\lambda > 1$, in order to solve this problem. When λ takes values greater than 1 it is also known that the stability region decreases with respect to the classical case. In this situation, the stability region for such case will be the complex plane region where the phase is lower than $\frac{\pi}{2}\lambda$ (Fig. 3). For a more detailed explanation, please refer to [15].

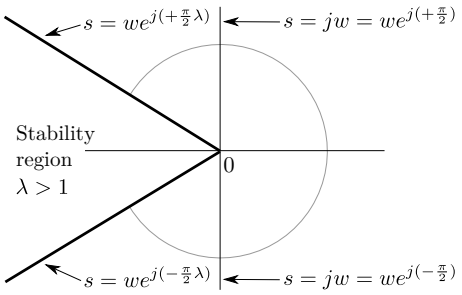


Fig. 3. Modified stability region on the fractional controller PI^λ ($\lambda > 1$).

B. Proposed PI^λ approach

Introducing a fractional–order integral action, the error becomes:

$$E(s) = \frac{1}{1 + \frac{G(s)}{s^\lambda}} \frac{r}{s^2} = \frac{s^{\lambda-2} r}{s^\lambda + G(s)}. \quad (17)$$

Then, the steady–state error is:

$$\lim_{s\to 0} sE(s) = \lim_{s\to 0} s \frac{s^{\lambda-2} r}{s^\lambda + G(s)} = \\ = \lim_{s\to 0} \frac{r}{s^\lambda + G(s)} \lim_{s\to 0} s^{\lambda-1} = K \lim_{s\to 0} s^{\lambda-1} \quad , \quad (18)$$

where K is a constant value. Thus, if $\lambda > 1$ the steady–state error will be zero.

$PI^\lambda D^\mu$ controllers are often applied with fractional orders $0 < \lambda, \mu < 1$. Integral actions with λ near to the value 2 will have, in general, stability and robustness problems, so the designed controller have to be tested in order to guarantee stability and a good enough phase margin.

The following Section will describe the proposed design method which yield the parameters of such fractional controller.

IV. Fractional–order PI^λ Controller Design

This Section describes the design procedure that yields the controller parameters. For simplicity, $s = w\, e^{j\frac{\pi}{2}\lambda}$ henceforth. To guarantee stability and robustness as well, we will use the hypotheses described in [14], [24]. Therefore, the parameters K_p, K_i, λ must verify the following:

1) Phase margin hypothesis:

$$\arg\left[C(s)G(s)\right]\Big|_{w=w_{cg}} = -\pi + \varphi_m, \quad (19)$$

where φ_m, the phase margin, is a design parameter, and $C(s)$, $G(s)$ are the controller and plant transfer functions, respectively. Thus, K_i can be evaluated as follows:

$$K_i = -\frac{w_{cg}^\lambda B}{\sin(\frac{\pi}{2}\lambda^2) + \cos(\frac{\pi}{2}\lambda^2) B}, \quad (20)$$

$$U = \frac{\left[\tau_1^2 w_{cg} + \tau_1 \cos(\frac{\pi}{2}\lambda)\right]\left[1 + \tau_2^2 w_{cg}^2 + 2\tau_2 w_{cg}\cos(\frac{\pi}{2}\lambda)\right] + \left[\tau_2^2 w_{cg} + \tau_2 \cos(\frac{\pi}{2}\lambda)\right]\left[1 + \tau_1^2 w_{cg}^2 + 2\tau_1 w_{cg}\cos(\frac{\pi}{2}\lambda)\right]}{\left[1 + \tau_1^2 w_{cg}^2 + 2\tau_1 w_{cg}\cos(\frac{\pi}{2}\lambda)\right]^{\frac{3}{2}}\left[1 + \tau_2^2 w_{cg}^2 + 2\tau_2 w_{cg}\cos(\frac{\pi}{2}\lambda)\right]^{\frac{3}{2}}} \quad (26)$$

$$K_p = \frac{1}{\left[1 + (\frac{K_i}{w_{cg}^{\lambda}})^2 + 2\frac{K_i}{w_{cg}^{\lambda}}\cos(\frac{\pi}{2}\lambda^2)\right]^{\frac{1}{2}} \prod_{i=1}^{2}\left[\left(1 + \tau_i^2 w_{cg}^2 + 2\tau_i w_{cg}\cos(\frac{\pi}{2}\lambda)\right)^{-\frac{1}{2}}\right]} \quad (27)$$

where

$$B = \tan\left[\left(\sum_{i=1}^{2}\tan^{-1}\frac{\tau_i w_{cg}\sin(\frac{\pi}{2}\lambda)}{1 + \tau_i w_{cg}\cos(\frac{\pi}{2}\lambda)}\right) - \pi + \varphi_m\right] \quad (21)$$

and τ_i are the time constants of the transfer function for the given second order system.

2) Robustness against gain variations:

$$\frac{\mathrm{d}\left(\arg\left[C(s)G(s)\right]\right)}{\mathrm{d}w}\Big|_{w=w_{cg}} = 0, \quad (22)$$

which guarantees robustness locally. This condition leads to the following cubic system:

$$F(\lambda) = A_3 K_i^3 + A_2 K_i^2 + A_1 K_i + A_0 = 0, \quad (23)$$

with coefficients

$$\begin{aligned} A_3 =& \lambda w_{cg}^{\lambda-1}\sin(\frac{\pi}{2}\lambda^2)\cos^2(\frac{\pi}{2}\lambda^2) \\ A_2 =& 2\lambda w_{cg}^{2\lambda-1}\sin(\frac{\pi}{2}\lambda^2)\cos(\frac{\pi}{2}\lambda^2) - U \\ A_1 =& \lambda w_{cg}^{3\lambda-1}\sin(\frac{\pi}{2}\lambda^2) - 2U w_{cg}^{\lambda}\cos(\frac{\pi}{2}\lambda^2) \\ A_0 =& -U w_{cg}^{2\lambda}, \end{aligned} \quad (24)$$

where U is given by (26).

3) Finally, a hypothesis for the gain crossover frequency:

$$|C(s)G(s)|\Big|_{w=w_{cg}} = 1, \quad (25)$$

which leads to (27), the equation that describes the proportional parameter K_p.

As will be described in the following Section, solving this system of equations yields the parameters K_p, K_i, λ that describe the PI^{λ} controller.

Remark (Robustness on fractional–order systems)**.** *The most common approach consists on controllers with $0 < \lambda < 1$. This leads to a wider stability region, compared with the classical case, $\lambda = 1$. Fractional orders $\lambda > 1$, like the proposed on this paper, clearly reduce the size of the stability region. Specifically, the region is limited by the semi–straight lines $s = w\,e^{\pm j(\pi/2)\lambda}$, with $w \in \mathbb{R}^+$ and Real part of s negative, as it was shown in Fig. 3. However, it allows an important feature such as being able to follow a setpoint ramp, as has been explained above.*

V. Experiments

A. Test setup

Some experiments have been conducted in order to validate the proposed PI^{λ} controller. VERDINO is meant to follow a pedestrian located in front of it. The robot should try to keep a desired distance between itself and the pedestrian, which moves along a straight line. The pedestrian cannot modify her direction, but can vary her speed. This implies that the car also has to adjust its own speed during the experiment, in order to successfully follow the behavior of the pedestrian.

The simulations have been carried out using the *Ninteger* MATLAB [25] toolbox developed by Duarte Valério [26]. The closed–loop implementation have been made using Simulink with the Real Time Workshop toolbox. The test setup does not include other obstacles, as this experiment focuses only on the tracking problem. The distance to the pedestrian and her speed are measured using a Laser Imaging Detection and Ranging (LIDAR) device, a Sick LMS111, with a maximum range of 20 meters. The LIDAR has an update frequency of $50\,Hz$, fast enough for tracking pedestrian movements. The distance to the pedestrian is used as the setpoint in the Simulink model, which includes the fractional controller. Then, the control action is sent to the control hardware of VERDINO by a serial protocol.

B. Design method

In order to apply the proposed design method, which consists on obtaining the controller parameters K_p, K_i, λ, the transfer function has to be expressed as shown on (16). The first step is to select suitable values for the phase margin φ_m and the crossover gain frequency w_{cg}. In this experiment, we have set $\varphi_m = 70°$ and $w_{cg} = 0.37\,rad/seg$. Applying the hypothesis (22) leads to the cubic system (23), with coefficients (24). In order to obtain the roots along the real axis of such system we propose applying the following graphical method:

- Plot $y = F(\lambda)$ and find the λ value that intersecs with the abscissa axis nearest to 1, where $F(\lambda)$ is the left hand side of (23).
- Since $y = F(\lambda)$ is a cubic curve, it can be shown that there will exist at least one intersection with the abscissa axis. In this example, $\lambda \approx 1.235$, as shown in Fig. 4.
- Once λ is found, apply (20) and (21) to obtain K_i. In this example, $K_i = 0.32$. When more than one real root exists, the chosen λ value should be the nearest to 1,

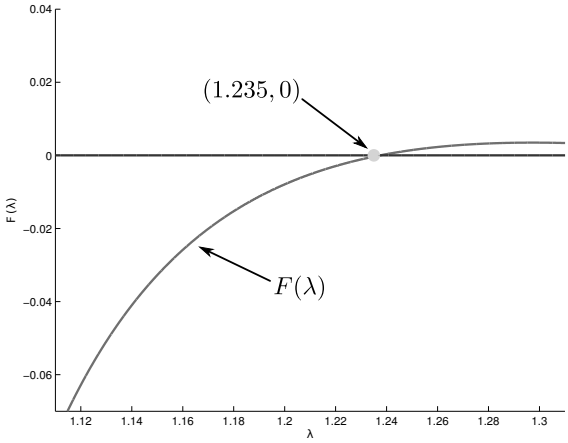


Fig. 4. Graphical method for finding λ.

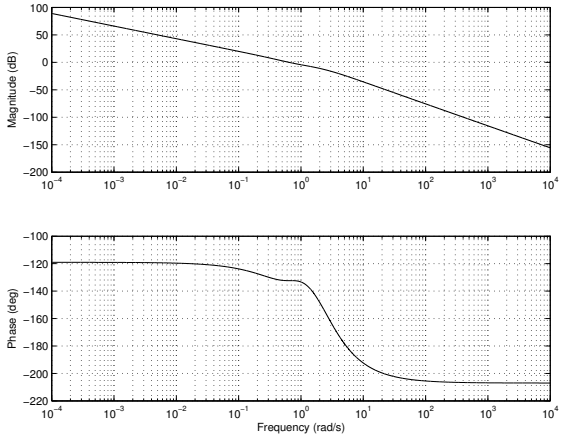


Fig. 5. Frequency response with controller parameters $K_p = 1.17$,$K_i = 0.32$,$\lambda = 1.235$.

since stability cannot be easily guaranteed if λ is closer to 2.

- Finally, the proportional gain can be easily obtained applying equation (27). $K_p = 1.17$ in this example.

C. Frequency and time analysis

The frequency response is shown in Fig. 5. As can be seen, both gain crossover frequency and phase margin specifications, $w_{cg} = 0.37\,rad/sec$ and $\varphi = 70°$, are satisfied.

Fig. 6 shows experimental data obtained from the described test setup. The pedestrian started to walk and tried to mantain a speed of $3.6\,m/s$ during 10 seconds. Then, it slowed down to approximately $2\,m/s$ for the last part of the experiment. She walked along a straight line during 15 meters. This experiment shows that the proposed PI^λ controller is able to significantly reduce the transient error, while the integer–order PI is not able to follow the setpoint closely, as expected.

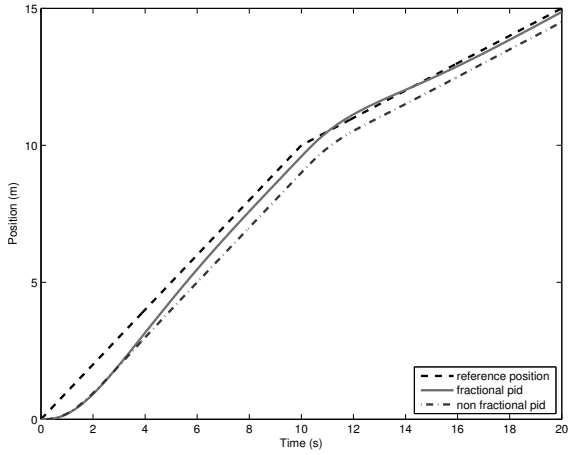


Fig. 6. Position tracking with speed changes.

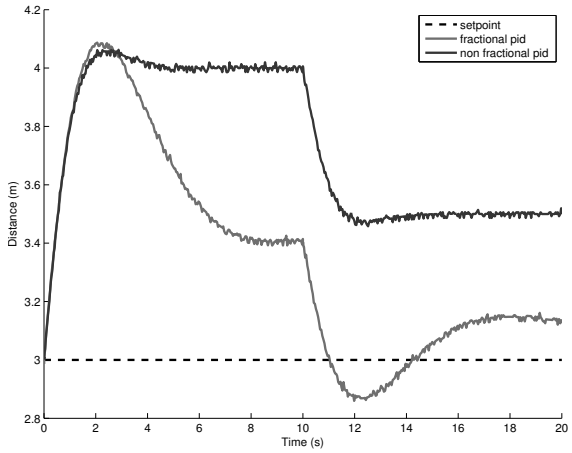


Fig. 7. Measured distance between VERDINO and the pedestrian. The setpoint was 3 meters.

Next, Fig. 7 shows the measured distance between VERDINO and the pedestrian during the experiment, which was set to 3 meters. It can be seen that with the proposed PI^λ controller, VERDINO follows the pedestrian closer than the integer–order PI, i.e., the distance is closer to 3 meters with the latter.

Finally, during the experiment we also measured the difference between the command effort with both approaches. Fig. 8 shows the relative difference Δ as follows:

$$\Delta = 100\frac{u_f - u_i}{u_i} \quad , \tag{28}$$

where u_f and u_i are the command effort for the fractional and integer order controllers, respectivelly. At most, the fractional–order controller needed about 10% more command than the integer version, which is reasonable, considering that the proposed controller is able to follow the speed variations of the pedestrian better.

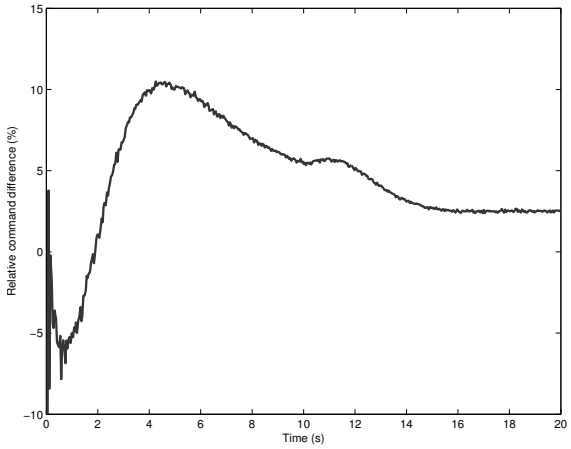


Fig. 8. Relative difference Δ on command effort for both approaches.

VI. Conclusions

In this paper we have proposed a fractional PI^{λ} controller, with a parameter $\lambda > 1$, in contrast with the most common approach, with fractional orders within the interval $(0, 1)$. The experiments conducted have shown that this controller is suitable for the speed ramp tracking problem, with an application in an electric car. In this paper we have considered the modification of the stability region when derivation and integration orders are not integer. Specifically, with $\lambda > 1$ we have found that the stability region is reduced.

The design method has been also described. Such method allows obtaining the controller parameters K_p, K_i, λ from the established stability and robustness specifications. The experiments has been conducted on a real system, with a test setup that allowed demostrating the better performance of the proposed PI^{λ} controller, compared with its integer order counterpart. Those experiments have also shown that the command effort is not greatly penalized, roughly by only a 10%, i.e., command values are similar to those obtained with a non fractional controller. However, results have shown a remarkable reduction of the tracking error with the proposed controller. Finally, it should be noted that the gain in reference tracking justifies, in some applications, the loss in robustness.

References

[1] R. Metzler and J. Klafter, "The random walk's guide to anomalous diffusion: a fractional dynamics approach," *Physics Reports*, vol. 339, pp. 1–77, 2000.

[2] R. Hilfer, *Applications of Fractional Calculus in Physics*. World Scientific Publishing Company, 2000.

[3] H. Liu, T. Oliphant, and L. Taylor, "General fractional derivative viscoelastic models applied to vibration elastography," *Ultrasonics, 2003 IEEE Symposium on*, vol. 1, Oct. 2003.

[4] J. A. Tenreiro-Machado, V. Kiryakova, and F. Mainardi, "Recent history of fractional calculus," *Communications in Nonlinear Science and Numerical Simulation*, vol. 16, pp. 1140–1153, 2011.

[5] A. A. Kilbas, H. M. Srivastava, and J. J. Trujillo, *Theory and Applications of Fractional Differential Equations*. Elsevier, Holland, 2006.

[6] A. Oustaloup, *La commande CRONE : Commande Robuste d'Ordre Non Entier*, ser. Traité des nouvelles technologies. Série automatique. Hermès, Paris, 1991.

[7] ——, *La Dérivation Non Entière. Théorie, Synthèse et Applications*. Hermès Science, Paris, 1995.

[8] I. Podlubny, *Fractional Differential Equations*. Mathematics in Science and Engineering, 1999, vol. 198.

[9] ——, "Fractional-order systems and $PI^{\lambda}D^{\mu}$–controllers," *Automatic Control, IEEE Transactions on*, vol. 44, pp. 208–214, Jan 1999.

[10] J. A. Tenreiro-Machado, "Special Issue : Fractional Order Systems," *Nonlinear Dynamics*, vol. 29, no. 1–4, pp. 1–385, Kluwer, 2002.

[11] J. Tenreiro-Machado and A. Luo, "Special Issue : Discontinuous and Fractional Dynamical Systems," *Computational and Nonlinear Dynamics, ASME Journal of*, vol. 3, no. 2, p. 125, April 2008.

[12] J. A. Tenreiro-Machado and R. Barbosa, "Special Issue : Fractional Differentiation and its Applications," *Vibration and Control, Journal of*, vol. 14, pp. 1253–1672, Sage Pub, September 2008.

[13] D. Baleanu, J. Tenreiro-Machado, and Z. B. Guvenc, *New Trends in Nanotechnology and Fractional Calculus Applications*. Springer, London, 2010.

[14] C. A. Monje, Y.-Q. Chen, B. M. Vinagre, D.-Y. Xue, and V. Feliu, *Fractional-Order Systems and Controls: Fundamentals and Applications.*, 1^{st} ed., ser. Advances in Industrial Control. Springer–Verlag, London, 2010.

[15] R. Caponetto, G. Dongola, L. Fortuna, and I. Petráš, *Fractional Order Systems: Modeling and Control Applications*. World Scientific Publishing Company, 2010.

[16] J. Sabatier, S. Poullain, P. Latteux, and A. Oustaloup, "Robust speed control of a low damped electromechnical system based on crone control: Application to a four mass experimental test bench," *Nonlinear Dynamics*, vol. 38, no. 1–4, pp. 383–400, 2004.

[17] A. L. Méhauté, J. Tenreiro-Machado, J. C. Trigeassou, and J. Sabatier, *Fractional Differentiation and its Applications*. Ubooks, Germany, 2005.

[18] I. Petráš, "Fractional–order feedback control of a DC motor," *Electrical Engineering, Journal of*, vol. 60, pp. 117–128, 2009.

[19] J. A. Tenreiro-Machado, "Effect of fractional orders in the velocity control of a servo system," *Computers and Mathematics with Applications*, vol. 59, pp. 1679–1686, 2010.

[20] E. González, L. Acosta, A. Hamilton, J. Felipe, M. Sigut, J. Toledo, and R. Arnay, "Towards a Multiagent Approach for the VERDINO Prototype," in *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2009, vol. 5518, pp. 21–24.

[21] S. G. Samko, A. A. Kilbas, and O. I. Marichev, *Fractional Integrals and Derivatives: Theory and Applications*. Gordon and Breach Science Publishers, 1993.

[22] S. Westerlund and L. Ekstam, "Capacitor theory," *Dielectrics and Electrical Insulation, IEEE Transactions on*, vol. 1, pp. 826–839, 1994.

[23] M. Cuadrado and R. Cabanes, *Temas de Variable Compleja*. Servicio de Publicaciones de la ETSIT UPM, Madrid, 1989.

[24] C. Monje, B. Vinagre, V. Feliu, and Y. Chen, "Tuning and auto–tuning of fractional order controllers for industry applications," *Control Engineering Practice*, vol. 16, no. 7, pp. 798–812, 2008.

[25] MATLAB, *version 7.14.0 (R2012a)*. Natick, Massachusetts: The MathWorks Inc., 2012.

[26] D. Valério, "Ninteger: A non–integer control toolbox for MATLAB," *Technical University of Lisbon*, Aug, 2005. [Online]. Available: http://web.ist.utl.pt/duarte.valerio/ninteger/ninteger.htm

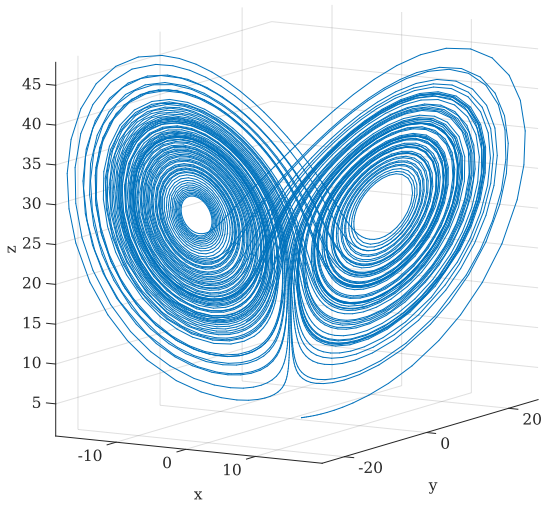
4.4 Fractional–order models and dynamical systems

In a practical sense, noisy sensors and non accurate observations can render very–long–range estimations nearly impossible to forecasts for nonlinear dynamical systems. Thanks to the attention drawn from these observations, chaos (along with fractals) has been identified as intrinsic features and behaviours universally present in nature. Chaos is said to be universal since several dissimilar dynamical systems have demonstrated to show the same nonlinear dynamics, from simple electronics circuits to involved models for weather forecasting. These concepts have been studied and developed by many scientists and practitioners of several fields, including robotics.

Nonlinear dynamics have fascinated the scientific community over the past century and recent years due to their close relation with natural phenomena and many physical systems. Lorenz (1963) properly described an effect that several scientists have acknowledge at some extent but that lacked the insight that he demonstrated. He found the first chaotic attractor in a three–dimensional autonomous system (shown in figure 4.3), describing the first mathematical and physical model which was considered the foundation of *Chaos theory*.

Chaos is considered by many scientists as one of the great revolutions of the past century, along with relativity and quantum mechanics. It can be defined as the study of unpredictable behaviour in nonlinear dynamical systems. Also called the dynamical systems theory or theory of nonlinear oscillations, it shows the sensitive dependence on initial conditions. This has become a major statement as it summarizes the unpredictable nature that many systems exhibit in the long run.

While chaotic behaviour and its consequences have been studied over



(a) Phase–space with $\sigma = 10$, $\beta = 8/3$, $\rho = 28$

$$\dot{x} = \sigma\,(y - x)$$
$$\dot{y} = \rho x - y + xz$$
$$\dot{z} = -\beta z + xy$$

(b) Lorenz system

Figure 4.3 Lorenz three–dimensional autonomous system (Lorenz, 1963), resembling a butterfly, where $x, y, z \in \mathbb{R}$ are pseudo–state variables and σ, ρ, β are positive parameters.

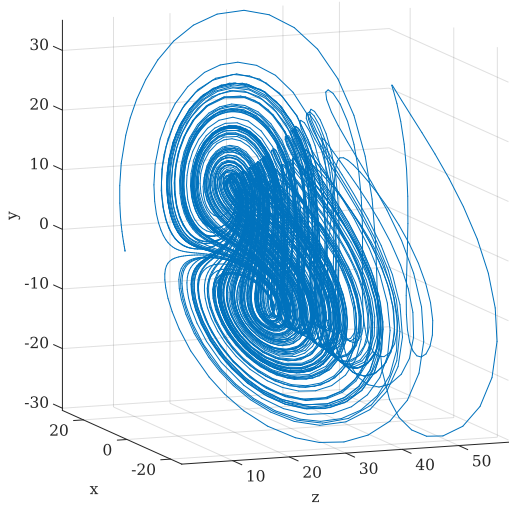
the past century, several applications and extensions remain to be properly established, along with relevant consequences and benefits yet to be discovered in our understanding of a wide variety of physical deterministic systems. However, unpredictable behaviour does not necessarily mean randomness: a deterministic system can exhibit chaotic behaviour which may appear random, mainly because a given solution may not show any periodic pattern. The main consequence is that, even being predictable for short time horizons, dynamical systems that exhibit chaos are unpredictable in the long run due to the known sensitive dependence on initial conditions: small variations and other unknowns may yield very different results on a deterministic process. The *Butterfly effect* has become the widespread name for this paradigmatic discovery.

Ravishankar and Ghosal (1999) probed the existence of chaotic be-

haviour on feedback–controlled two– and three–Degrees of Freedom (DOF) robots. They shown that for a certain range of parameters, the nonlinear dynamical system of differential equations demonstrate chaos. For the last three decades, many scientists have studied and described a wide range of applications of deterministic chaos in robotics. These applications probe the effects of applying the dynamics of natural behaviours that lead to chaos on bio–inspired mobile robots. The applications are two–fold: study observations of chaotic behaviour in robots (Goswami et al., 1998; Gritli et al., 2015; Nehmzow and Walker, 2005) and synthesize chaotic dynamics to study new approaches for solving different tasks (Kaygisiz et al., 2011; Luo et al., 2013; Nakamura and Sekiguchi, 2001). One of the relevant insights from these applications is that chaos is intrinsic to robot dynamics.

Control methods and algorithms based on chaos theory have started to be applied to mobile robots in the past few years. Path planning and other optimization algorithms have been found to benefit from the application of chaos. Nakamura and Sekiguchi (2001) developed the first mobile robot which navigates following a chaotic pattern, designing a controller that produces chaotic motion. Thanks to the sensitive dependence on initial conditions, the behaviour of the robot became unpredictable. This enabled the mobile robot to perform different tasks like surveillance and searching without any trajectory planning or workspace mapping. The coverage of the entire workspace is better and more efficient even when compared with other methods like those based on random walk. Not requiring a map is a huge advantage for chaotic mobile robots designed for specific tasks, since chaotic motion planning techniques ensure a rapid search of the workspace (Kaygisiz et al., 2011).

When considering a mobile robot as a chaotic system, the uncertainty models differ significantly from the stochastic ones. A relevant application for state estimation on dynamical systems like mobile robots is



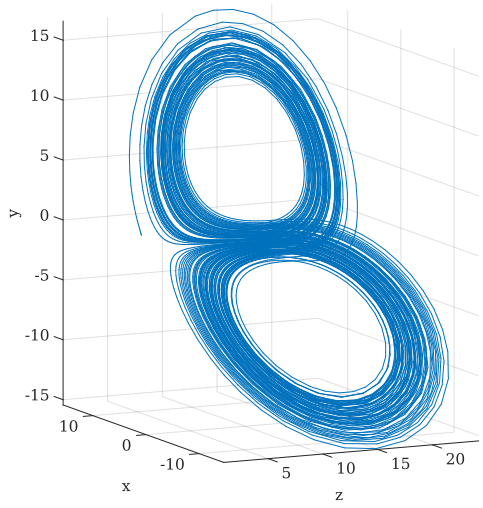
(a) Phase–space, with $a = 35$, $b = 3$, $c = 28$

$$\dot{x} = a\,(y - x)$$
$$\dot{y} = (c - a)\,x + cy - xz$$
$$\dot{z} = -bz + xy$$

(b) Chen system

Figure 4.4 Chen three–dimensional autonomous system (Chen and Ueta, 1999), where $x, y, z \in \mathbb{R}$ are pseudo–state variables and a, b, c are positive parameters.

the *Polynomial Chaos Theory* (Wiener, 1938). This framework is a non–sampling based technique that provides a probabilistic description for the effects of uncertainty with a low computational cost, compared with the well–known Monte Carlo (MC) simulations (Hover and Triantafyllou, 2006). It allows the separation of the stochastic elements of a dynamical system from its deterministic ones. Derived from the insight presented by Lorenz, it can be seen that due to noise and other disturbances generated by sensors and actuators, the localization of a mobile robot can only be predicted for short time horizons. Furthermore, the interaction with the environment is an additional source of constrains and perturbations, which also plays a relevant role on the dynamical behaviour of a mobile robot.



(a) Phase–space, with $a = 36$, $b = 3$, $c = 13$

$$\dot{x} = a\,(y - x)$$
$$\dot{y} = -xz + cy$$
$$\dot{z} = -bz + xy$$

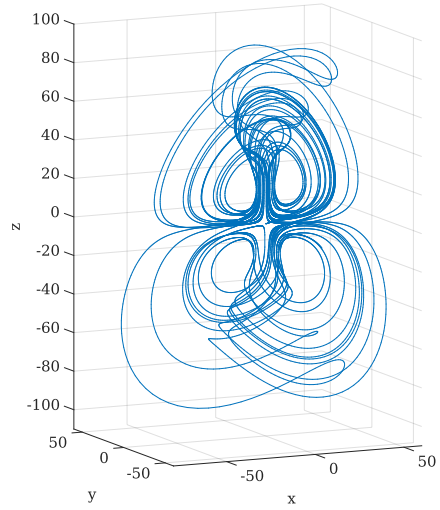
(b) Lü system

Figure 4.5 Lü three–dimensional autonomous system (Lü and Chen, 2002), where $x, y, z \in \mathbb{R}$ are pseudo–state variables and a, b, c are positive parameters.

Fractional–order hyperchaotic models

After Lorenz's discovery, more chaotic polynomial systems have been discovered and described. Chen and Ueta (1999) described the first Lorenz–like attractor, named the Chen system (figure 4.4), and a few years later, Lü and Chen (2002) coined the Lü system (figure 4.5), a new three–dimensional autonomous systems. While not being topologically equivalent, they are very similar to the first chaotic system described by Lorenz. In spite of their simplicity and similarity, Lü and Chen systems still motivate the development of new methods and tools to analyse chaotic systems. For instance, there still not exist an algebraically simple description for chaotic flow, remaining nowadays as an open topic (Leonov and Kuznetsov, 2015).

Many other nonlinear dynamical systems that exhibit chaotic behaviour



(a) Phase–space, with $a = 8$, $b = 43.75$, $c = 2$, $d = 10$, $g = 5$, $h = 0.2$, $h = 0.05$

$$\dot{x} = ax - yz$$
$$\dot{y} = xz - by$$
$$\dot{z} = cxy - dz + gxw$$
$$\dot{w} = kw - hy$$

(b) Dadras and Momeni (2010) system

Figure 4.6 Four–dimensional autonomous system (Dadras and Momeni, 2010), where $x, y, z, w \in \mathbb{R}$ are pseudo–state variables and a, b, c, d, g, h, k are positive parameters. This system exhibits *hyperchaos*.

under certain conditions have been described. One of the characteristics that define chaotic behaviours are the *strange attractors*. These relevant points have basins of attraction, where infinitesimally close points diverge exponentially over time around them. This divergence is measured by the Lyapunov Exponent (LE), which is positive for two points moving apart over time at an exponential rate. Regular chaotic systems have one positive LE. A positive LE is also an indicator that the system has a highly sensitive dependence on initial conditions. This number is the principal criteria of chaos, and it is related to the complexity of the dynamic behaviour of a given system.

Consider the system described by Dadras and Momeni (2010), which is shown on figure 4.6. Inspecting the phase space, the presence of more

complex dynamics is clear: the dynamics expand in more than one direction. Rossler (1979) introduced this new class of systems when reporting a system with more than one positive LE for the first time, exhibiting what was called *hyperchaos*. These new kind of attractors have more complicated dynamics than a chaotic system. The main consequence is that these class of systems are even more sensitive to perturbations. Hyperchaos is a relevant and interesting characteristic not only for theoretical research but also for its practical implications, being chaos control one of the more prominent in engineering. Due to their more complex nature, suitable modelling tools and control methods need to be considered. For these reasons, hyperchaotic systems are attracting the attention of an increasing number of scientists and engineers.

Several control techniques have been applied for controlling chaotic systems, like nonlinear control, adaptive control, sliding mode control and robust control, to name a few. However, these techniques introduce even more complexity on the analysis and synthesis of hyperchaotic systems. Thus, simple control methods are preferred when studying and implementing these systems. For instance, simple located control feedback methods can be applied to control full–pseudo–state–feedback representations of fractional–order dynamical systems. This is demonstrated in the following contribution, where a novel hyperchaotic fractional–order system is introduced (Morell et al., 2014).

4.5 Control of a novel fractional hyperchaotic system using a located control method

This section includes the full text for the following article, one of the contributions of this thesis presented as a compendium of publications.

- Title: Control of a novel fractional hyperchaotic system using a

located control method

- Authors: Antonio Morell,Abolhassan Razminia and Juan J. Trujillo
- Journal: Optimization: A Journal of Mathematical Programming and Operations Research
- Year: 2014
- ISSN: 0233–1934
- doi: 10.1080/02331934.2014.891034

Optimization, 2014
http://dx.doi.org/10.1080/02331934.2014.891034

Taylor & Francis
Taylor & Francis Group

Control of a novel fractional hyperchaotic system using a located control method

Antonio Morell[a], Abolhassan Razminia[b] and Juan J. Trujillo[c]*

[a] *Departamento de Ingeniería Informática, Universidad de La Laguna, La Laguna, Spain;* [b] *Dynamical Systems and Control Research Laboratory, Department of Electrical Engineering, School of Engineering, Persian Gulf University, Bushehr, Iran;* [c] *Departamento de Análisis Matemático, Universidad de La Laguna, La Laguna, Spain*

(*Received 26 June 2013; accepted 23 January 2014*)

Fractional order dynamics and chaotics systems have been recently combined, yielding interesting behaviours. In this paper, a novel integer order hyperchaotic system is considered. Then, a fractional order hyperchaotic representation of said system is proposed using a natural fractionalization. Two different linear control methodologies to deal with the complexity which introduce such systems are proposed. Those methods are able to modify the hyperchaotic behaviour of the system and force it to move towards a fixed point; i.e. steady state. These approaches give a general framework for taming such complex systems using simple linear controllers. The main tools for analysing the controlled system are Matignon stability criterion and RouthHurwitz test. Using a reliable numerical simulation, the designed system is simulated to verify the theoretical analysis.

Keywords: fractional calculus; non-linear fractional differential systems; hyperchaos; linear control methods; suppression of chaos; stabilization of bad behaviour in hyperchaotic system

AMS Subject Classifications: 34H10; 34H15; 26A33; 34A08

1. Introduction

Fractional calculus as an extension of ordinary or integer order calculus was introduced 300 years ago.[1,2] It has been found that the dynamical behaviour of many physical systems can be properly described by using fractional order system of differential equations. Moreover, such tool has been extensively applied in many fields which have seen an overwhelming growth in the last three decades. Some examples can be found in a wide variety of fields such as physics,[3,4] engineering,[5,6] mathematical biology,[7] non-linear control problems in Banach spaces,[8] finance,[9] optimal control,[10] impulsive evolution equations,[11] and life science.[12] In general, it can be stated that for almost all systems that contain internal damping, for instance, the traditional energy-based approach cannot be used to obtain a suitable description of the behaviour of non-conservative systems. However, it is well known that finding the physical meaning or geometrical interpretation of the fractional order operators is one of the main problems which remain open.

*Corresponding author. Email: jtrujill@ullmat.es

Moreover, chaos and its applications have been studied and developed by many scientists from various applied fields.[13,14] Many mathematical definitions of chaos exist but, roughly speaking, it may be described as a type of dynamic behaviour with the following characteristics: super-sensitivity to perturbation in initial conditions, noise-like behaviour, deterministic motion, trajectories of chaotic systems pass through any point an infinite number of times. As an important point it should be noted that chaotic systems represent a class of indeterminacy models differing from the stochastic models. On one hand, knowledge of the deterministic model's current system state allows predictions of future trajectories with an arbitrarily long period, whereas with the stochastic model precise forecasts cannot be made. On the other hand, and generally speaking, the forecast error for chaotic models grows exponentially, even for an arbitrarily short time. Consequently, a forecast can be made only on a limited number of cases, defined by the admissible forecast error.

A regular chaotic system has one positive Lyapunov exponent.[15] However, systems with more than one positive Lyapunov exponent are called hyperchaotic, which have more complicated dynamics than a chaotic system.[16] Hyperchaotic systems are very interesting and could have applications, for example, in communications or in mechanics problems. Such property can be seen from numerical simulations, Lyapunov exponents or spectral analysis.[16]

In chaos literature, three major kinds of problems can be found: suppression of chaos, chaotization and synchronization of chaotic systems. Stabilization problems of the unstable periodic orbiter, with the corresponding suppression of chaos,[17] arise in suppression of noise and vibrations of various constructions, elimination of harmonics in the communication systems, electronic devices, mechanics and so on.[13] Harmful vibrations can be either regular (quasiperiodic) or chaotic. Hyperchaotic vibration can also be found as a new and important behaviour that appears in practical engineering applications, which happens to be problems more complex than quasiperiodic and chaotic phenomena. The second class includes control problems of excitation or generation of chaotic oscillations,[18] also called chaotization or anticontrol of chaos problems. They arise when chaotic motion is the desired behaviour of the system. Pseudorandom-number generators and sources of chaotic signals in communication and radar systems are classical examples. Recent research suggests that chaotization of processes could produce an appreciable effect in chemical and biological technologies, as well as in handling of the loose materials. This problem is characterized by the fact that the trajectory of the system phase vector is not predetermined, i.e. it is either unknown or of no consequence for attaining the objective. Finally, another important class of control objectives corresponds to the problems of synchronization. Synchronization finds important applications, for instance, in vibration technology,[19] communications,[13] biology and ecology,[20] and so on.

According to the Rössler definition,[21] hyperchaotic systems exhibit more than one positive Lyapunov exponent (i.e. their dynamics can expand in more than one direction). Therefore, such systems are more sensitive to perturbations, external disturbances and parameter variations than basic chaotic ones, leading to some particular control difficulties. In contrast, hyperchaotic systems provide a great richness in UPOs. Thus, such systems are highly attractive for some applied fields such as non-linear circuits,[22] secure communication[23] and laser applications.[24] Obviously, many methods coming from the control theory framework are able to deal with hyperchaotic systems.[25–30] However, such methods often have rigorous analytical results that yield complex control laws, and in some cases the controller structure is more complex than the original hyperchaotic system.

Various rigorous control techniques have been applied for controlling chaos: non-linear control,[31,32] adaptive control,[33,34] sliding mode control [35] and robust control,[36] for example. However, such techniques have complex requirements and structures, which sometimes are more difficult to be accurately implemented on real systems. Moreover, their complexity introduces more non-linearities than those present in the original systems. Therefore, simpler control methods which can be easily implemented are more attractive and suitable for engineers. The main contribution of this paper is to develop a simple enough located feedback and a full-pseudo-state-feedback controller that can stabilize any unstable fixed point. Additionally, a new hyperchaotic incommensurate fractional order system is introduced, based on an integer order system with various dynamical regimes. As is known from the literature, controlling incommensurate fractional order systems is more difficult than the commensurate ones.

The rest of the paper is organized as follows. For the sake of self-containment of the paper, Section 2 briefly introduces the basic tools used in fractional calculus. In Section 3, a novel fractional hyperchaotic system is studied, and then two control techniques to stabilize their unstable behaviour are applied. Finally, Section 4 is devoted to the concluding remarks.

2. Preliminaries

2.1. *Fractional calculus background*

In this subsection some mathematical background is presented. For further details, please refer to [1,37].

Definition 2.1 The fractional integral operator of order $q > 0$, in the sense of Riemann–Liouville (R–L), for a Lebesgue integrable function is defined as follows

$$I_{a+}^{q} x(t) := \frac{1}{\Gamma(q)} \int_a^t (t-s)^{q-1} x(s) ds, \tag{1}$$

where $\Gamma(q) = \int_0^\infty e^{-z} z^{q-1} dz$ is the Gamma function.

Definition 2.2 The R–L fractional derivative of order q $(m-1 < q < m, m \in \mathbb{Z}^+)$, is defined as follows

$${}^{RL}D_{a+}^{q} x(t) := D^m I_{a+}^{(m-q)} x(t) = \frac{1}{\Gamma(m-q)} \frac{d^m}{dt^m} \int_a^t (t-s)^{m-q-1} x(s) ds, \tag{2}$$

Remark 1 For the R–L defitinion, fractional derivative and integral operators have the following properties (see [1,38])

$$(1) \quad \mathcal{L}\left\{ I_{0+}^{q} x(t) \right\} = s^{-q} x(s)$$

$$(2) \quad \lim_{q \to m} I_{0+}^{q} x(t) = I_{0+}^{m} x(t), \quad q > 0, \ m \in \mathbb{Z}^+$$

$$(3) \quad {}^{RL}D_{0+}^{q} c = \frac{c t^{q-1}}{\Gamma(1-q)}, \ c \in \mathbb{R}$$

As can be observed, the R–L differentiation of a constant is not zero. Moreover, its Laplace transform needs fractional derivatives of the function at initial time. These could give a good justification to introduce the so-called Caputo fractional derivative as follows:

Definition 2.3 The Caputo fractional derivative of order q ($m-1 < q < m, m \in \mathbb{Z}^+$) is defined as follows:

$$ {}^{C}D^{q}_{a+}x(t) := I^{(m-q)}_{a+} D^{m}x(t) = \frac{1}{\Gamma(m-q)} \int_a^t (t-s)^{m-q-1} x^{(m)}(s)ds, \tag{3} $$

Remark 2 For the fractional Caputo derivative operator of order q we have the following properties:

(1) ${}^{C}D^{q}_{0+}c = 0 \; q \in \mathbb{R}^+$
(2) ${}^{C}D^{q}_{0+} \; I^{q}_{0+}x(t) = {}^{RL}D^{q}_{0+} \; I^{q}_{0+}x(t) = x(t), \; 0 < q < 1$
(3) ${}^{RL}D^{q}_{0+} \; I^{q}_{0+}x(t) = x(t), \; q \in \mathbb{R}^+$

A fractional dynamical system with initial conditions could be given by

$$ \begin{cases} {}^{C}D^{q}_{0+}x(t) = f(x(t), t), \;\; m-1 < q < m \in \mathbb{Z}^+, \; t > 0 \\ \left[{}^{C}D^{k}_{0+}x(t)\right]_{t=0} = x^k_0, \;\; k = 0, 1, 2, ..., (m-1) \end{cases} \tag{4} $$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \times \mathbb{R} \to \mathbb{R}^n$, and $q = (q_1 q_2 ... q_n)^T$. If $q_1 = q_2 = ... = q_n$ are rational numbers, then (4) is called a commensurate fractional order dynamical system, and incommensurate otherwise. Moreover, the sum of the orders of all involved derivatives in (4), i.e. $\sum_{i=1}^{n} q_i$ is called the effective dimension of Equation (4), and usually n is called the inner dimension of the system.[39,40]

THEOREM 2.4 *Consider the following linear fractional differential system:*

$$ {}^{C}D^{q}_{0+}x(t) = Ax(t), \;\; x(0) = x_0, \tag{5} $$

with $x \in \mathbb{R}^n$, $A \in \mathbb{R}^{n\times n}$ and $q = (q_1 q_2 ... q_n)^T$, $0 < q_i \leq 1$. Also $q_i = \frac{n_i}{d_i}$, $gcd(n_i, d_i) = 1$. Let M be the lowest common multiple of the denominators d_i's. The zero solutions of the system (5) are globally asymtotically stable in the Lyapunov sense if all roots λ of the equation

$$ \triangle(\lambda) = det(diag(\lambda^{Mq_i}) - A) = 0 \tag{6} $$

satisfy $|arg(\lambda_i)| > \frac{\pi}{2M}$.

Proof. See, for example,[41].

2.2. *On numerical methods to solve fractional differential equations*

As was reported in [42], frequency approaches for simulating fractional chaotic systems are not reliable. Therefore, suitable alternatives for such simulation are time-domain techniques, although the numerical methods usually applied to solve ordinary differential equations (ODE) must be modified in order to be able to solve their counterpart fractional differential equations (FDE). See reference [43].

A modification of the Adams–Bashforth–Moulton algorithm was proposed by Diethelm et al. [44–46] to solve FDEs.

Consider $q \in (m-1, m]$, the following Cauchy type problem:

$$\begin{cases} {}^{C}D^{q}_{0+}x(t) = f(x(t), t), \ 0 \le t \le T \\ x^{k}(0) = x^{(k)}_{0}, \ k = 0, 1, ..., m-1 \end{cases} \tag{7}$$

is equivalent to the following Volterra integral equation:

$$x(k) = \sum_{k=0}^{m-1} x^{(k)}_{0} \frac{t^{k}}{k!} + i\frac{1}{\Gamma(q)} \int_{0}^{t} (t-s)^{q-1} f(s, x(s))ds \tag{8}$$

Consider the uniform grid $\{t_n = nh : n = 0, 1, ..., N\}$ for some integer natural number N and $h = \frac{T}{N}$. Let $x_h(t_n)$ be an approximation to $x(t_n)$. Also, assume that we have already estimated the approximations of $x_h(t_j), \ j = 1, 2, ..., n$, then $x_h(t_{n+1})$ can be estimated by the expression:

$$\begin{aligned} x_h(t_{n+1}) = &\sum_{k=0}^{m-1} x^{(k)}_{0} \frac{t^{k}_{n+1}}{k!} + \frac{h^{q}}{\Gamma(q+2)} f(t_{n+1}, x^{p}_{h}(t_{n+1})) \\ &+ \frac{h^{q}}{\Gamma(q+2)} \sum_{j=0}^{n} a_{j,n+1} f(t_j, x_n(t_j)) \end{aligned} \tag{9}$$

where

$$a_{j,n+1} = \begin{cases} n^{q+1} - (n-q)(n+1)^{q}, & \text{if } j = 0 \\ (n-j+2)^{q+1} + (n-j)^{q+1} - 2(n-j+1)^{q+1}, & \text{if } 1 \le j \le n \\ 1, & \text{if } j = n+1 \end{cases} \tag{10}$$

The values of the preliminary predictor $x^{p}_{h}(t_{n+1})$ are estimated by:

$$x^{p}_{h}(t_{n+1}) = \sum_{k=0}^{m-1} x^{(k)}_{0} \frac{t^{k}_{n+1}}{k!} + \frac{1}{\Gamma(q)} \sum_{j=0}^{n} b_{j,n+1} f(t_j, x_n(t_j)) \tag{11}$$

where

$$b_{j,n+1} = \frac{h^{q}}{q}((n-j+1)^{q} - (n-j)^{q}) \tag{12}$$

Finally, the error in this method is given by:

$$max_{j=0,1,...,N}|x(t_j) - x_n(t_j)| = O(h^{p}) \tag{13}$$

where $p = min(2, 1+q)$.

3. System description and control

Consider the novel hyperchaotic system described by Dadras and Momeni in [47]

$$\begin{aligned} \dot{x} &= ax - yz \\ \dot{y} &= xz - by \\ \dot{z} &= cxy - dz + gxw \\ \dot{w} &= kw - hy \end{aligned} \tag{14}$$

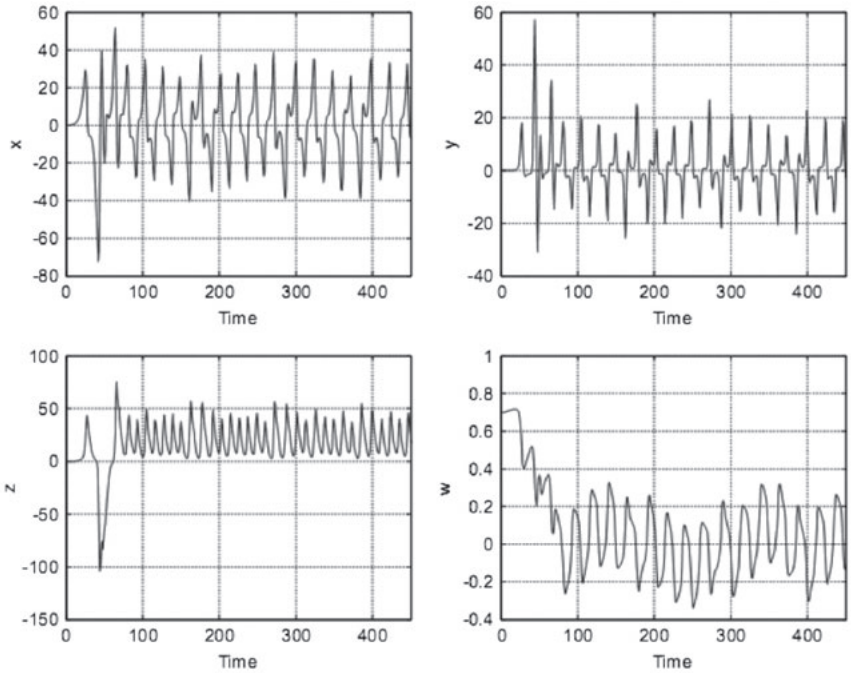


Figure 1. Numerical simulations of trajectories of system (14) with hyperchaotic natures.

Here $x, y, z, w \in \mathbb{R}$ are the pseudo-state variables, and a, b, c, d, g, h, k are positive constant parameters of the system. For instance, if $a = 8,\ b = 43.75,\ c = 2,\ d = 10,\ h = 0.2,\ k = 0.05$, one can find the following five points for the system (14):

$$\begin{aligned} &Q_1 : (0, 0, 0, 0) \\ &Q_2 : (-4, 4594, +1.9069, -18.7083, +7.6277) \\ &Q_3 : (+4.4594, +1.9069, +18.7083, +7.6277) \\ &Q_4 : (+4.4594, -1.9069, -18.7083, -7.6277) \\ &Q_5 : (-4.4594, -1.9069, +18.7083, -7.6277) \end{aligned} \tag{15}$$

For each equilibrium point we can compute the Jacobian matrix and then find the eigenvalues.

According to the results reported in [47], all of the above equilibrium points are unstable. Numerical simulation of this system are depicted in Figures 1 and 2. We point out that the system (14) can produce chaotic and quasiperiodic behaviours too.

Now consider the following fractional order dynamical system:

$$\begin{aligned} {}^{C}D_{0+}^{q_1} x &= ax - yz \\ {}^{C}D_{0+}^{q_2} y &= xz - by \\ {}^{C}D_{0+}^{q_3} z &= cxy - dz + gxw \\ {}^{C}D_{0+}^{q_4} w &= kw - hy \end{aligned} \tag{16}$$

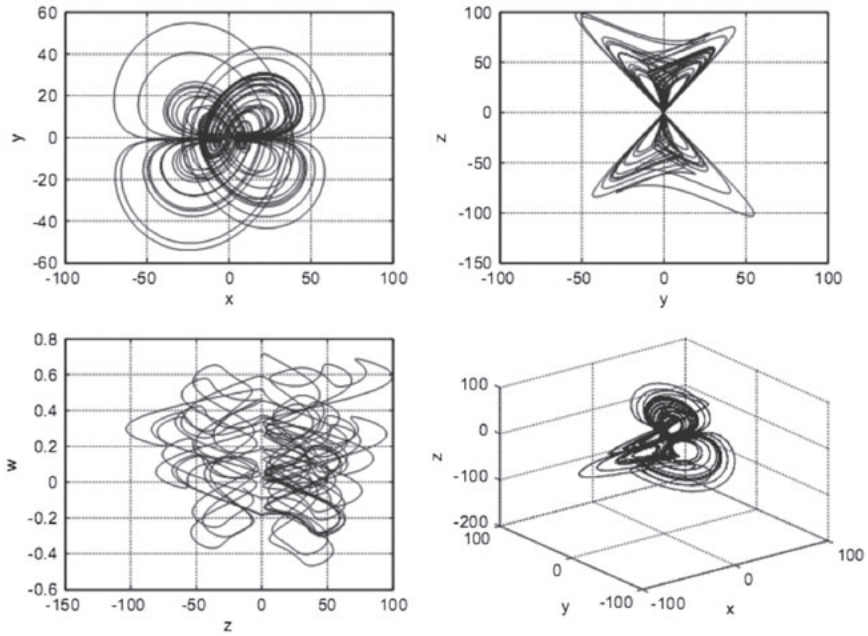


Figure 2. Phase portrait of system (14) with hyperchaotic natures.

Note that the equilibrium for this new fractional order hyperchaotic system are the same on its integer order counterpart; i.e. (14). Firstly, the Jacobian matrix for (16) can be obtained from the equilibrium points as follows

$$J_Q = \begin{pmatrix} a & -z^* & -y^* & 0 \\ z^* & -b & x^* & 0 \\ cy^* + gw^* & cx^* & -d & gx^* \\ 0 & -h & 0 & k \end{pmatrix} \tag{17}$$

For instance, let us focus on Q_2. A similar approach for other equilibrium can be used. So evaluating the Jacobian matrix in Q_2 we have

$$J_Q = \begin{pmatrix} 8 & 18.7083 & -1.9069 & 0 \\ -18.7083 & -43.75 & -4.4594 & 0 \\ 41.9524 & -8.9188 & -10 & -22.2971 \\ 0 & -0.2 & 0 & 0.05 \end{pmatrix} \tag{18}$$

The corresponding eigenvalues of Q_2 can be easily obtained as

$$\begin{aligned} \Lambda_2 &= (\lambda_1\ \lambda_2\ \lambda_3\ \lambda_4) \\ &= (-40.6033\ 0.0912\ -2.5940 + i13.4997\ -2.5940 - i13.4997) \end{aligned} \tag{19}$$

The equilibrium point Q_2 is a saddle-focus point; such equilibrium point is unstable, because one of the associated eigenvalues is a real positive number and consequently the necessary condition derived in [48] yields no meaningful result in this case. Therefore, it may exhibit chaos or hyperchaos for any order of fractional differentiation. Let us consider the commensurate case; i.e. $0 < q_1 = q_2 = q_3 = q_4 < 1$. If we take $q_i = 0.98,\ i = 1, 2, 3, 4,$

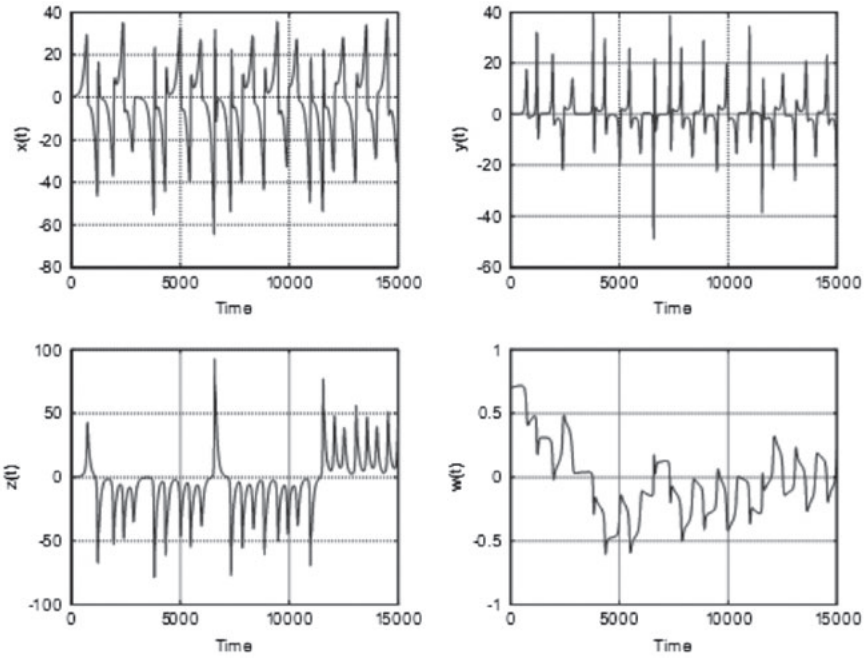


Figure 3. Numerical simulations of trajectories of system (16) with order $q = 0.98$ and hyperchaotic nature.

we can observe the hyperchaoticity behaviour in the above fractional order system in Figures 3 and 4. It should be noted that both hyperchaotic and chaotic behaviour can be observed, as can be seen in Figure 5. As a remark, the Lyapunov exponent criterion [49] can be used for identifying the chaotic or hyperchaotic behaviour of a fractional order non-linear dynamical system. It can be easily seen that for the hyperchaotic behaviour the Lyapunov exponents are $(+, +, 0, -)$ and for the chaotic one they are $(+, 0, -, -)$. It should also be noted that the following simulations were performed using the discussed method in Section 2.2.

It is known that if the integer order dynamical system is stable, then its fractional order counterpart is also stable, under some conditions. Thus, with the sufficient stability condition given in (16), it can be shown that the integer order (14) is stable. Therefore, the following theorem (3.1) can be stated:

THEOREM 3.1 *Consider the following controlled fractional order dynamical system:*

$$\begin{aligned} {}^{C}D_{0+}^{q_1}x &= ax - yz + u_1 \\ {}^{C}D_{0+}^{q_2}y &= xz - by + u_2 \\ {}^{C}D_{0+}^{q_3}z &= cxy - dz + gxw + u_3 \\ {}^{C}D_{0+}^{q_4}w &= kw - hy + u_4 \end{aligned} \tag{20}$$

If the control law is chosen as $u_1 = u_2 = u_3 = 0$ and $u_4 = -\mu(w - w^*)$, then a sufficient condition for asymptotical stability of Q_2 is $\mu > 0.15$.

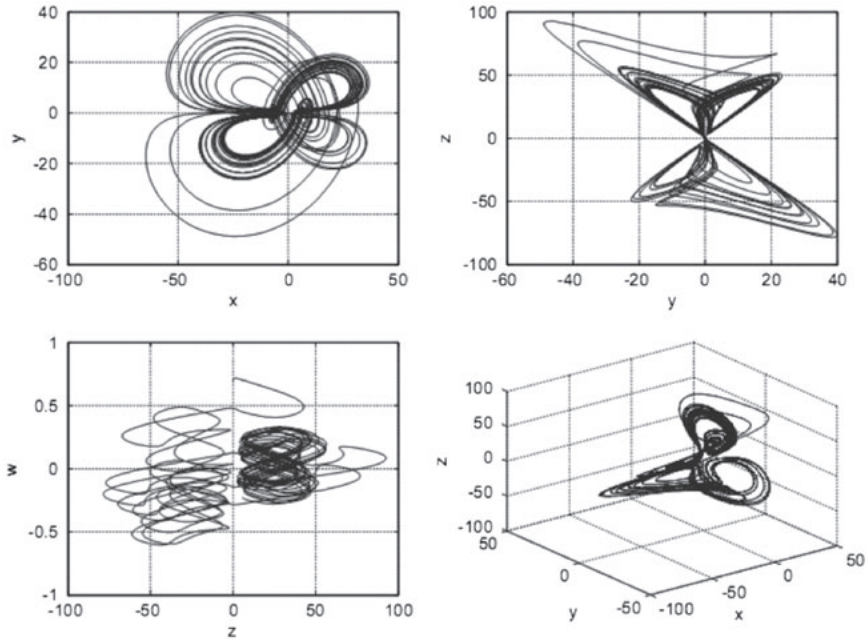


Figure 4. Phase portrait of system (16) with order $q = 0.98$ and hyperchaotic nature.

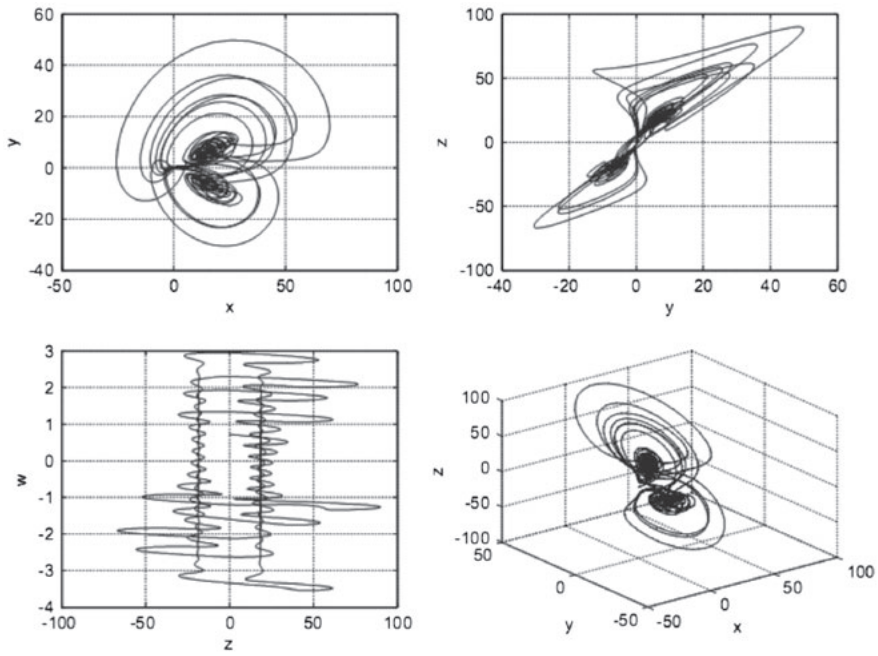


Figure 5. Phase portrait of system (16) with order $q = 0.85$ and chaotic nature.

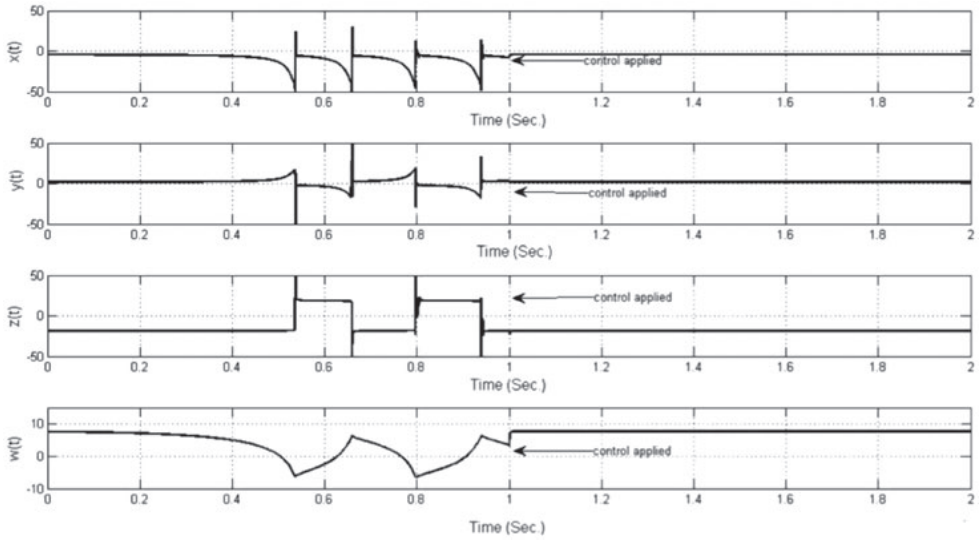


Figure 6. Numerical results for controlled system (20), with $(q_1, q_2, q_3, q_4) = (0.9, 0.9, 0.9, 0.9)$, applying the control law of Theorem 3.1.

Proof is straightforward applying Routh–Hurwitz criterion. See, for example,[31].

A numerical simulation with $\mu = 0.2$, and $q_i = 0.9$ is depicted in Figure 6. Using a similar approach we can develop several theorems to stabilize the fractional order hyperchaotic system (20). Note that the proposed Theorem 3.1 can be applied without any modification for the case of incommensurate orders. The reason is obvious: the Jacobian matrix is not changed and, when using the proposed control law, all eigenvalues of Q_2 are located in the left half plane. So for any incommensurate order set, the fractional order system (20) will be stable via the proposed control law. Simulations using the control law are repeated for an incommensurate order $(q_1, q_2, q_3, q_4) = (0.98, 0.94, 0.92, 0.90)$, see Figure 7. Similarly, all other equilibria can be controlled.

As another approach, now consider the controlled fractional order hyperchaotic system described in (20) where $q_i = \frac{n_i}{d_i}$ and $gcd(n_i, d_i) = 1,\ i = 1, 2, 3, 4$. If the control laws are chosen as:

$$\begin{aligned} u_1 &= -\mu_1(x - x^*), \qquad u_2 = -\mu_2(y - y^*), \\ u_3 &= -\mu_3(z - z^*), \qquad u_4 = -\mu_4(w - w^*) \end{aligned} \tag{21}$$

Then its Jacobian matrix evaluated in an equilibrium point is as follows:

$$J_Q = \begin{pmatrix} a - \mu_1 & -z^* & -y^* & 0 \\ z^* & -b - \mu_2 & x^* & 0 \\ cy^* + gw^* & cx^* & -d - \mu_3 & gx^* \\ 0 & -h & 0 & k - \mu_4 \end{pmatrix} \tag{22}$$

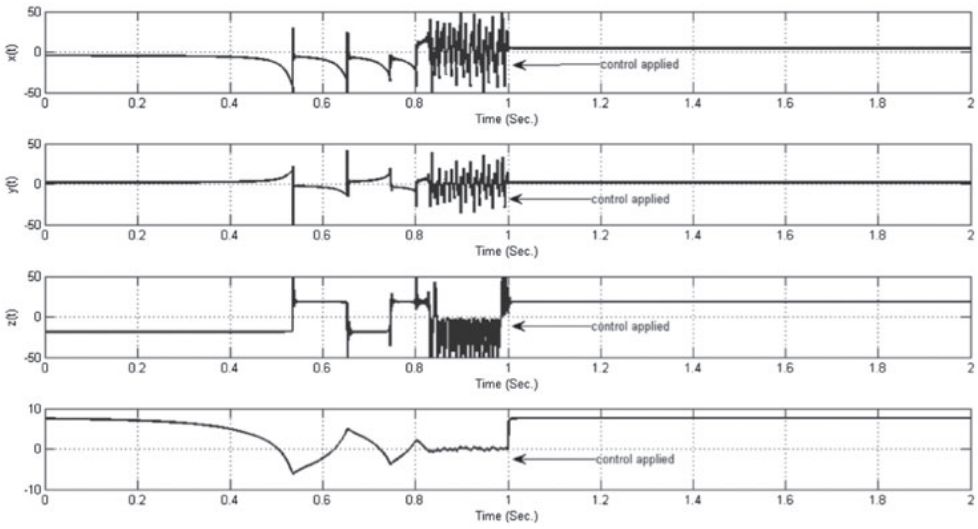


Figure 7. Numerical results for controlled system (20), with $(q_1, q_2, q_3, q_4) = (0.98, 0.94, 0.92, 0.9)$, applying the control law of Theorem 3.1.

According to Theorem 2.4, the characteristic equation is then:

$$\Delta(\lambda) = det \begin{pmatrix} \lambda^{m_1} - a + \mu_1 & z^* & y^* & 0 \\ -z^* & \lambda^{m_2} + b - \mu_2 & -x^* & 0 \\ -cy^* - gw^* & -cx^* & \lambda^{m_3} + d + \mu_3 & -gx^* \\ 0 & h & 0 & \lambda^{m_4} - k + \mu_4 \end{pmatrix} \tag{23}$$

where $m_i = M\frac{n_i}{d_i}$, $1 \leq i \leq 4$ and M is the lcm of $d_i's$. Manipulating the above determinant we can find some necessary and sufficient conditions for asymptotical stability of (20). For simplicity, here we consider the origin Q_1, as one of the saddle points. To stabilize Q_1, first we have the following characteristic equation:

$$\Delta(\lambda) = det \begin{pmatrix} \lambda^{m} - a + \mu_1 & 0 & 0 & 0 \\ 0 & \lambda^{m_2} + b - \mu_2 & 0 & 0 \\ 0 & 0 & \lambda^{m_3} + d + \mu_3 & 0 \\ 0 & h & 0 & \lambda^{m_4} - k + \mu_4 \end{pmatrix} \tag{24}$$

Since the above matrix is bottom-triangle its determinant can be easily obtained by multiplying its diagonal elements:

$$\Delta(\lambda) = (\lambda^{m_1} - a + \mu_1)(\lambda^{m_2} + b + \mu_2)(\lambda^{m_3} + d + \mu_3)(\lambda^{m_4} - k + \mu_4) \tag{25}$$

The characteristic equation has then the following four roots:

$$\begin{aligned} \lambda_1 &= (a - \mu_1)^{\frac{1}{m_1}}, & \lambda_2 &= (-b - \mu_2)^{\frac{1}{m_2}}, \\ \lambda_3 &= (-d - \mu_3)^{\frac{1}{m_3}}, & \lambda_4 &= (k - \mu_4)^{\frac{1}{m_4}} \end{aligned} \tag{26}$$

Thus, we can state the following results:

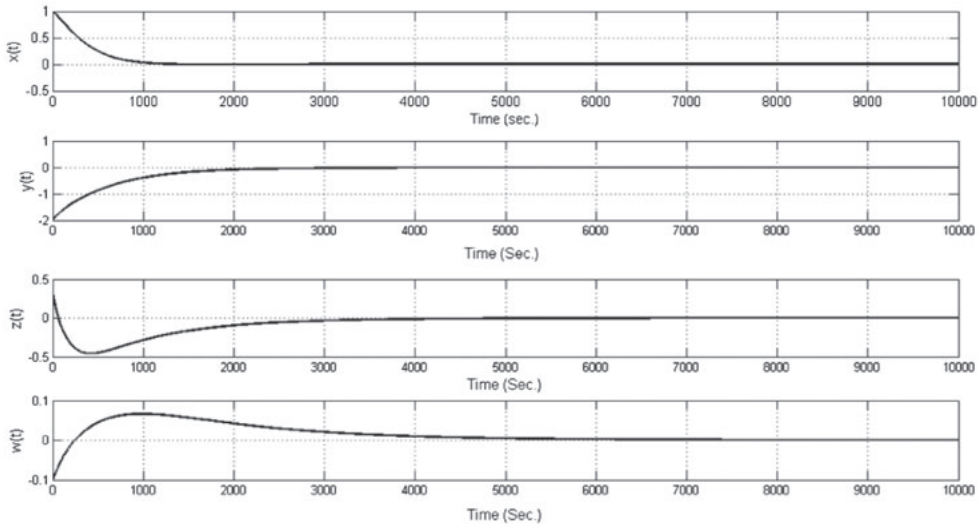


Figure 8. Numerical results for controlled system (20), with $(q_1, q_2, q_3, q_4) = (0.99, 0.97, 0.95, 0.93)$, when the control law of Theorem 3.2 is applied.

THEOREM 3.2 *Consider the controlled hyperchaotic fractional order system* (20). *The necessary and sufficient condition for asymptotical stability of* Q_1 *based on control laws* (21), *is that:*

$$min|arg(\lambda_i)| > \frac{\pi}{2M}, \quad 1 \leq i \leq 4 \tag{27}$$

COROLLARY 3.3 *If all* m_i, $i = 1, 2, 3, 4$ *are odd number, then the sufficient condition for asymptotical stability of* Q_1 *based on control laws* (21), *is* $a < \mu_1$, $-b < \mu_2$, $-d < \mu_3$, *and* $k < \mu_4$.

The argument for this corollary is obvious. In particular, for these inequalities, all eigenvalues are negative real numbers and thus their arguments are the same; i.e. $arg(\lambda_i) = \pi$ which is always greater than $\frac{\pi}{2M}$ for every $M \in \mathbb{N}$.

To see the effectiveness of the proposed control law, in the following we simulate the system (20) with orders: $(q_1, q_2, q_3, q_4) = (0.99, 0.97, 0.95, 0.93)$. For this system $M = 100$ and $m_1 = 99$, $m_2 = 97$, $m_3 = 95$, $m_4 = 93$. Thus we can use the Corollary 3.3. Then, a sufficient condition for the given controller (21), could be given as follows:

$$\mu_1 > 8, \ \mu_2 > -43.75, \ \mu_3 > -10, \ \mu_4 > 0.05 \tag{28}$$

All simulations were done using the algorithm presented in Section 2 and the corresponding results for $\mu_1 = 10$, $\mu_2 = -42$, $\mu_3 = -9$, $\mu_4 = 1$ and the initial conditions $(x_0, y_0, z_0, w_0) = (1, 2, 0.3, -0.1)$ are depicted in Figure 8.

4. Conclusions

In this work we have studied and discussed a new fractional order hyperchaotic system. Based on a reliable numerical simulation method, the fractional system was simulated and

several phenomena such as chaos and hyperchaos were studied. Moreover, using a simple located feedback control technique, a control law that is able to stabilize all equilibrium points of a fractional system has been proposed. The proposed control method can be applied to incommensurate order system without any modification. Also, based on the Matignon stability criterion, we developed a necessary and sufficient condition to stabilize the origin as one of saddle points for the fractional hyperchaotic system with incommensurate orders. All analytical results are validated by several illustrating numerical simulations.

Funding

This work was partially supported by Project SAGENIA DPI2010-18349 and MTM2010-16499 from MINECO of Spain and funds from the Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI) under programme Apoyo al Personal Investigador en Formación 2012, co-financed by FEDER funds (EU).

References

[1] Kilbas A, Srivastava H, Trujillo J. Theory and applications of fractional differential equations. Holland: Elsevier; 2006.
[2] Miller S, Ross B. An introduction to the fractional calculus and fractional differential equations. A Wiley-Interscience Publication. New York (NY): John Wiley and Sons; 1993.
[3] Hilfer R, editor. Applications of fractional calculus in physics. River Edge (NJ): World Scientific; 2000.
[4] Wang J, Zhou Y, Wei W. Fractional Schrödinger equations with potential and optimal controls. Nonlinear Anal. Real World Appl. 2012;13:2755–2766.
[5] Debnath L. Full state hybrid projective synchronization of a novel incommensurate fractional order hyperchaotic system using adaptive mechanism. IJMMS. 2003;54:3413–3442.
[6] Razminia A. Recent applications of fractional calculus to science and engineering. Int. J. Math. Math. Sci. 2003;54:3413–3442.
[7] Davis G, Kohandel M, Sivaloganathan S, Tenti G. The constitutive properties of the brain paraenchyma: part 2. Fractional derivative approach. Med. Eng. Phys. 2006;28:455–459.
[8] Wang J, Zhou Y. Analysis of nonlinear fractional control systems in Banach spaces. Nonlinear Anal. Theory Methods Appl. 2011;74:5929–5942.
[9] Scalas E, Gorenflo R, Mainardi F. Fractional calculus and continuous-time finance. Physica A. 2000;284:376–384.
[10] Wang J, Zhou Y, Wei W. Optimal feedback control for semilinear fractional evolution equations in Banach spaces. Syst. Control Lett. 2012;61:472–476.
[11] Wang J, Feckan M, Zhou Y. Relaxed controls for nonlinear fractional impulsive evolution equations. J. Optimiz. Theory Appl. 2013;156:13–32.
[12] Ahmad W, El-Khazali R. Fractional-order dynamical models of love. Chaos Solition Fract. 2007;33:1367–1375.
[13] Fradkov A, Evans R. Control of chaos: methods and applications in engineering. Annu. Rev. Control. 2005;29:33–56.
[14] Razminia A, Majd V, Baleanu D. Chaotic incommensurate fractional order Rössler system: active control and synchronization. Adv. Diff. Equ. 2011;15:1–12.
[15] Zhang H, Liu D, Wang Z. Controlling chaos, suppression, synchronization and chaotification. Berlin: Springer; 2009.
[16] Perruquetti W, Barbot J. Chaos in automatic control. London: Taylor and Francis group CRC Press; 2006.

[17] Pai N, Yau H-T. Suppression of chaotic behavior in horizontal platform systems based on an adaptive sliding mode control scheme. Comm. Nonlinear Sci. Numer. Simul. 2011;16:133–143.
[18] Ge Z, Yang C. Chaos synchronization and chaotization of complex chaotic systems in series form by optimal control. Chaos Solitons Frac. 2009;42:994–1002.
[19] Blekhman I. Synchronization of dynamic systems. Moscow: Nauka; 1971.
[20] Yu H, Zhong S, Agrawal R. Mathematics analysis and chaos in an ecological model with an impulsive control strategy. Comm. Nonlinear Sci. Numer. Simul. 2011;16:776–786.
[21] Rössler O. An equation for hyperchaos. Phys. Lett. A. 1979;71:155–157.
[22] Barbara C, Silvano C. Hyperchaotic behaviour of two bi-directionally Chua's circuits. Int. J. Circ. Theory Appl. 2002;30:625–637.
[23] Udaltsov V, Goedgebuer J, Larger L, Cuenot J, Levy P, Rhodes W. Communicating with hyperchaos: the dynamics of a DNLF emitter recovery of transmitted information. Opt. Spectrosc. 2003;95:114–118.
[24] Vicente R, Daudén J, Colet P, Toral R. Analysis and characterization of the hyperchaos generated by a semiconductor laser subject to a delayed feedback loop. IEEE J. Quantum Electron. 2005;41:541–548.
[25] Dou F, Sun J, Duan W, Lü K. Controlling hyperchaos in the new hyperchaotic system. Commun. Nonlinear Sci. Numer. Simul. 2009;14:552–559.
[26] Hsieh J, Hwang C, Wang A, Li W. Controlling hyperchaos of the Rössler system. Int. J. Control. 1999;72:882–886.
[27] Jang M, Chen C, Chen C. Sliding mode control of hyperchaos in Rössler systems. Chaos Solitons Frac. 2002;14:1465–1476.
[28] Mahmoud G, Mahmoud E. Synchronization and control of hyperchaotic complex Lorenz system. Math. Comput. Simul. 2010;80:2286–2296.
[29] Matouk A. Stability conditions, hyperchaos and control in a novel fractional order hyperchaotic system. Phys. Lett. A. 2009;373:2166–2173.
[30] Zhu C. Controlling hyperchaos in hyperchaotic Lorenz system using feedback controllers. Appl. Math. Comput. 2010;216:3126–3132.
[31] Hegazi A, Ahmed E, Matouk A. On chaos control and synchronization of the commensurate fractional order Liu system. Comm. Nonlinear Sci. Numer. Simul. 2013;18:1193–1202.
[32] Razminia A, Baleanu D. Fractional synchronization of chaotic systems with different orders. Proc. Rom. Acad. Series A. 2012;13:314–321.
[33] Razminia A. Full state hybrid projective synchronization of a novel incommensurate fractional order hyperchaotic system using adaptive mechanism. Indian J. Phys. 2013;87(2):161–167.
[34] Zheng Y, Nian Y, Wang D. Controlling fractional order chaotic systems based on Takagi-Sugeno Fuzzy model and adaptive adjustment mechanism. Phys. Lett. A. 2010;375:125–129.
[35] Yin C, Dadras S, Zhong S, Chen Y. Control of a novel class of fractional-order chaotic systems via adaptive sliding mode control approach. Appl. Math. Modell. 2013;37:2469–2483.
[36] Kuntanapreeda S. Robust synchronization of fractional-order unified chaotic systems via linear control. Comput. Math. Appl. 2012;63:183–190.
[37] Cafagna D. Fractional calculus: a mathematical tool from the past for the present engineer. IEEE Trans. Indust. Electronic. 2007;4:35–40.
[38] Li C, Deng W. Remarks on fractional derivatives. Appl. Math. Computat. 2007;187:777–784.
[39] Matignon D. Stability results for fractional differential equations with applications to control processing. Proc. Computat. Eng. Syst. Appl. Multi-conf. 1996;2:963–968.
[40] Wang Y, Li C. Does the fractional brusselator with efficient dimension less than 1 have a limit cycle? Phys. Lett. A. 2007;363:414–419.
[41] Deng W, Li C, Lu J. Stability analysis of linear fractional differential system with multiple time delays. Nonlinear Dyn. 2007;48:409–416.
[42] Tavazoei M, Haeri M. Limitations of frequency domain approximation for detecting chaos in fractional order systems. Nonlinear Anal. 2008;69:1299–1320.

[43] Baleanu D, Diethelm K, Scalas E, Trujillo J. Fractional calculus: models and numerical methods, series on omplexity. Vol. 33. Singapore: World Scientific Publication; 2012.
[44] Diethelm K. An algorithm for the numerical solution of differential equations of fractional order. Electron Trans. Numer. Anal. 1997;5:1–6.
[45] Diethelm K, Ford N. Analysis of fractional differential equations. J. Math. Anal. Appl. 2002;265:229–248.
[46] Diethelm K, Ford N, Freed A. A predictor-corrector approach for the numerical solution of fractional differential equations. Nonlinear Dyn. 2002;29:3–22.
[47] Dadras S, Momeni H. Four-scroll hyperchaos and four-scroll chaos evolved from a novel 4D nonlinear smooth autonomous system. Phys. Lett. A. 2010;374:1368–1373.
[48] Tavazoei M, Haeri M. Chaotic attractors in incommensurate fractional order systems. Physica D. 2008;237:2628–2637.
[49] Abarbanel D. Analysis of observed chaotic data. New York (NY): Springer-Verlag; 1996.

4.6 Results and discussion

This chapter explores two applications of fractional calculus for nonlinear model and control of complex dynamical systems. The first contribution demonstrates that a PI^{λ} controller, with a fractional–order $\lambda > 1$, yields a theoretical zero steady–state error for the position tracking problem. Then, it introduces a design method that yields the parameters of the controller that meet certain design rules. Results show that the controlled feedback loop is capable of following the input function, with changes in its slope. Even with a reduced stability region, this design method is relevant for solving the position tracking problem, which in some applications would justify the loss in robustness. Finally, more energetic commands are obtained when applying the fractional–order controller. This can be justified by the fact that the controller compensates the error with respect to the target tracked, compared with an integer–order controller.

When applied to model complex dynamics, fractional–order models have demonstrated yielding interesting results. In the second contribution, a novel hyperchaotic system is introduced, derived from an hyperchaotic integer one. Hyperchaotic systems also exhibit very interesting behaviours, being more difficult to control, as they are more sensitive to perturbations. While such systems have been successfully analysed and controllers have been developed, we are interested in those design methods that yield control rules simple and easy enough to implement, as they are relevant in some applications in engineering and other fields. Exploiting the fact that a fractional–order counterpart of an integer–order dynamical system is as stable as the original, a simple controlled version of such system can be defined by applying the Routh–Hurwitz criterion. Additionally, with the proposed control law, it can

be verified that the located method places all eigenvalues in left plane, guaranteeing stability, even for the more difficult incommensurate case. Simulations show the effectiveness of the proposed control law, managing to effectively control both the commensurate and incommensurate systems analysed.

"It's the questions we can't answer that teach us the most. They teach us how to think. If you give a man an answer, all he gains is a little fact. But give him a question and he'll look for his own answers."

— Patrick Rothfuss, The Wise Man's Fear

5

Conclusions

With this dissertation I have presented different approaches for explaining and solving a set of relevant topics for mobile robots related with dynamic state estimation. I started with the definition of probabilistic robotics as a common background and motivation to study the interaction between a robot and its environment and the nature of all processes involved. In this regard, we applied these techniques to investigate and bring new results from different topics related with localization and navigation in unstructured environments. This also has served to put in context other relevant contributions, that have motivated, for example, the application of techniques based on fractional calculus to solve a basic yet important problem like position tracking, or studying the control of a novel hyperchaotic system, a class of complex dynamical models that recently have started to be applied on mobile robotics.

The study of manipulators is relevant for mobile robots, since they are key components on exploration rovers, biped walkers or humanoid robots, to name a few. Considering this, I described a novel application

of a well known machine learning technique, Support Vector Regression, to an also well known set of kinematics problems. Firstly, I presented this application within a method to solve the forward kinematics in real–time on generalised parallel robots —Stewart platforms. Secondly, I extended and tailored the method in order to solve the inverse kinematics problem on serial robots, also in real–time. Then, the same method was demonstrated to be general enough to apply it on a more involved case, a parallel robot with joint offsets, demonstrating accuracy and also real–time performance.

I showed that a spatial decomposition method using Support Vector Machines to model the behaviour of a parallel or serial manipulator can achieve real–time performance, as well as good enough accuracy for a wide range of applications. The mean error of the implemented method is good enough for a wide variety of applications. However, the accuracy can be improved applying a simple correction step, based on the classic steps of prediction and correction from measurements as in filtering theory.

In general, this method outperformed in accuracy the polynomial version previously developed, except for one of the positions analysed. Compared to other methods for the case of a parallel robot with joint offsets, it did not achieved better accuracy. On highly populated regions, i.e., more training data for each Support Vector Regression (SVR) machine, the proposed method yield more accurate poses, with the trade–off of a higher training time. The technique used to generate the training set of poses has a great impact on the modelling capabilities. There is room for improvements on this step, using another class of random number generators, e.g., explore generators based on chaotic systems.

Furthermore, the method demonstrated to be able to yield all possible solutions for a given configuration or pose, in contrast to other numerical approximations. When obtaining the number of possible so-

lutions for a configuration, I decided to use a classifier based on the *k–means* method and a goodness–of–fit heuristic. While this implementation yielded good results, there is room for improving this step. Potentially, other fitting and non–Gaussian models can be explored.

The implementation presented can be greatly optimised, allowing to add more complexity in the off–line phase and to obtain faster on–line evaluations. As the learning process happens off–line, more complex models can be used, and the only limitation is the size in memory of the representation and the evaluation time of such models. At the same time, fine tuning the training parameters is needed to achieve good levels of accuracy, having special care for avoiding overfitting like in any model–fitting methodology. Related to the training parameters and kernels, I found that a kernel based on the Radial Basis Function (RBF) yielded the best accuracy/training time ratio, especially as compared to a Sigmoid kernel. Training all SVR machines with the same parameters demonstrated to be good enough to achieve the desired accuracy. However, tuning the training parameters depending on the region might improve accuracy on some of the worst cases.

Developing a self–driving car like Verdino motivated other lines of research, such as the application of fractional–order control to mobile robots. Exploring chaotic systems has also served to expand the scope and bring new techniques that have started to demonstrate applicability in mobile robots. While the application to control a fractional–order hyperchatic system is more a theoretic result, I consider that efforts in this direction might bring new results and applications to chaos theory on mobile robots. A new fractional–order hyperchaotic system was simulated and several phenomena, such as chaos and hyperchaos were studied.

The application of a controller with a fractional–order integral action, PI^{λ}, demonstrated to be suitable for solving the position tracking pro-

blem. The proposed fractional–order, $1 < \lambda < 2$, reduces the stability region of the system. However, it allows applying the proposed methodology which yields zero steady–state error. While using a fractional–order controller may improve the dynamical behaviour of the modelled system, it has an impact on the command effort, with a slight increase. More experiments will bring more evidences on this. The design method presented allows obtaining the set of parameters for the fractional–order controller that meet the given stability and robustness specification. Finally, it is important to consider that the implementation of the proposed PI^{λ} controller can be optimised, potentially with a lower level realization like a microcontroller or physical circuit.

Additionally, I worked on the definition of a sensorimotor system for a biped humanoid robot, also based on probabilistic techniques, to study the resemblance with a human being and its effects on robust walking patterns generation. We developed a self–driving electric car, Verdino, as the experimental platform on some of the contributions presented in this dissertation. This included the low level control and sensing, localization and navigation layers, applying techniques from probabilistic robotics. I consider that this fundamental background helps understanding what we do not yet know —and understand — about the behaviour of a coupled dynamic system like a mobile robot and its environment.

The realization of a versatile, mobile, general–purpose autonomous robot is one of the most desired goals of scientists and engineers since a very long time. In this dissertation I demonstrated some applications and developments that can help to move forward into this objective, combining some of the most important techniques with new tools, including machine learning, fractional–order control and chaos, under the common rule–set of probabilistic robotics.

Having that said, I consider that interesting research lines open being supported by these observations, as described in the following section.

5.1 Further research

- Support Vector Machines for regression can generalise the forward and inverse kinematics solutions in most cases, but other techniques can potentially yield more accurate results, since I applied a general approach.
- On–line Support Vector (SV) or non supervised learning methods can potentially yield interesting results and applications for real–time state estimation, where the model is updated and improved on runtime.
- The input space for the training algorithm of the Support Vector Machine (SVM) for each region has been randomly generated using a pseudo–random number generator. Using a chaotic generator to partition and populate the learning sets might improve the pose coverage for each region.
- Chaos dynamics have been demonstrated on certain types of robots. On complex biped robots, it is expected to find chaotic behaviours and bifurcation scenarios, also related with complex dynamics. Furthermore, chaos has been successfully applied to planning and navigation tasks, intended for exploration and evaluation of unstructured environments. Further research on the application of chaos dynamics and their control can be expected to yield more relevant results when applied to mobile robots.

References

B. D. O. Anderson and J. B. Moore. *Optimal filtering*. 1979.

I. Arasaratnam and S. Haykin. Cubature Kalman Filters. *IEEE Transactions on Automatic Control*, 54(6):1254–1269, June 2009. ISSN 0018–9286. doi: 10.1109/TAC.2009.2019800.

R. Arnay, N. Morales, A. Morell, J. Hernandez-Aceituno, D. Perea, J. T. Toledo, A. Hamilton, J. J. Sanchez-Medina, and L. Acosta. Safe and Reliable Path Planning for the Autonomous Vehicle Verdino. *IEEE Intelligent Transportation Systems Magazine*, 8(2):22–32, 2016. ISSN 1939–1390. doi: 10.1109/MITS.2015.2504393.

D. Baleanu, J. Tenreiro-Machado, and Z. B. Guvenc. *New Trends in Nanotechnology and Fractional Calculus Applications*. Springer, London, 2010.

M. Benosman and G. Le Vey. Control of Flexible Manipulators: A Survey. *Robotica*, 22(5):533–545, October 2004. ISSN 0263–5747. doi: 10.1017/S0263574703005642.

D. Blackwell. Conditional Expectation and Unbiased Sequential Estimation. *The Annals of Mathematical Statistics*, 18(1):105–110, 03 1947. doi: 10.1214/aoms/1177730497.

B. E. Boser, I. Guyon, and V. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

B. E. Boser, I. Guyon, and V. Vapnik. Pattern recognition system using support vectors, July 1997.

R. Caponetto, G. Dongola, L. Fortuna, and I. Petráš. *Fractional Order Systems: Modeling and Control Applications*. World Scientific Publishing Company, 2010.

M. A. Chace. Vector Analysis of Linkages. *Trans. of the ASME. Journal of Engineering for Industry*, 85(3):289–297, August 1963. doi: 10.1115/1.3669867.

G. Chen and T. Ueta. Yet another chaotic attractor. *International Journal of Bifurcation and Chaos*, 09(07):1465–1466, 1999. doi: 10.1142/S0218127499001024.

R. Clavel. DELTA, a fast robot with parallel geometry. In C. W. Burckhardt, editor, *Proceedings of the 18th International Symposium on Industrial Robots*, pages 91–100. Springer–Verlag, 1988.

S. Dadras and H. R. Momeni. Four–scroll hyperchaos and four–scroll chaos evolved from a novel 4d nonlinear smooth autonomous system. *Physics Letters A*, 374(11–12):1368–1373, 2010. ISSN 0375–9601. doi: 10.1016/j.physleta.2010.01.030.

M. M. Dalvand and B. Shirinzadeh. Forward Kinematics Analysis of Offset 6–RRCRR Parallel Manipulators. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 255(12):3011–3018, Dec 2011.

M. M. Dalvand and B. Shirinzadeh. Kinematics Analysis of 6–DOF Parallel Micro–Manipulators with Offset U–Joints: A Case Study. *International Journal of Intelligent Mechatronics and Robotics*, 2:28–40, 2012.

L. Debnath and D. Bhatta. *Integral Transforms and Their Applications, Third Edition*. Taylor & Francis, 2014. ISBN 9781482223576.

P. Del Moral. Nonlinear filtering: interacting particle resolution. *Markov processes and related fields*, 2(4):555–581, 1996.

F. Dellaert, W. Burgard, D. Fox, and S. Thrun. Using the CONDENSATION algorithm for robust, vision-based mobile robot localization. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 2, page 594, 1999. doi: 10.1109/CVPR.1999.784976.

J. Denavit and R. S. Hartenberg. A kinematic notation for lower-pair mechanisms based on matrices. *Trans. of the ASME. Journal of Applied Mechanics*, 22:215–221, 1955.

A. K. Dhingra, A. N. Almadi, and D. Kohli. A Gröbner–Sylvester Hybrid Method for Closed–Form Displacement Analysis of Mechanisms. *Trans. of the ASME. Journal of Mechanical Design*, 122(4):431–438, August 1999. doi: 10.1115/1.1290395.

A. Doucet. On Sequential Simulation–Based Methods for Bayesian Filtering. Technical report, Signal Processing Group, Dept. of Engineering, University of Cambridge, 1998.

A. Doucet, N. de Freitas, K. Murphy, and S. Russell. Rao–Blackwellised Particle Filtering for Dynamic Bayesian Networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 176–183, Stanford, CA, USA, 2000.

A. Doucet, N. De Freitas, and N. Gordon, editors. *Sequential Monte Carlo methods in practice*. Springer Verlag, 2001.

J. Duffy. *Analysis of Mechanisms and Robot Manipulators*. John Wiley & Sons, Inc., New York, NY, USA, 1980. ISBN 0470270020.

J. Duffy and C. Crane. A Displacement Analysis of the General Spatial 7–link, 7R Mechanism. *Mechanism and Machine Theory*, 15(3):153–169, 1980. ISSN 0094-0114X. doi: 10.1016/0094-114X(80)90001-4.

J. Duffy and J. Rooney. A Foundation for a Unified Theory of Analysis of Spatial Mechanisms. *Trans. of the ASME. Journal of Engineering for Industry*, 97(4):1159–1164, November 1975. doi: 10.1115/1.3438718.

G. A. Einicke and L. B. White. Robust extended Kalman filtering. *IEEE Transactions on Signal Processing*, 47(9):2596–2599, September 1999. ISSN 1053–587X. doi: 10.1109/78.782219.

J. Espelosín, L. Acosta, and D. Alonso. Path planning approach based on flock dynamics of moving particles. *Applied Soft Computing*, 13(4): 2159–2170, 2013. ISSN 1568–4946. doi: 10.1016/j.asoc.2012.12.015.

W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, January 1968. ISBN 0471257087.

W. Feng, J. Sun, L. Zhang, C. Cao, and Q. Yang. A support vector machine based naive bayes algorithm for spam filtering. In *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8, Dec 2016. doi: 10.1109/PCCC.2016.7820655.

A. Goswami, B. Thuilot, and B. Espiau. A Study of the Passive Gait of a Compass-Like Biped Robot. *The International Journal of Robotics Research*, 17(12):1282–1301, 1998. doi: 10.1177/027836499801701202.

V. E. Gough. Contribution to discussion of papers on research in Automobile Stability, Control and Tyre performance. In *Proceedings of the Auto Divisional Institute of Mechanical Engineers.*, pages 392–394, 1956–1957.

G. Grisetti, C. Stachniss, and W. Burgard. Improving Grid–based SLAM with Rao–Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, pages 2432–2437, April 2005. doi: 10.1109/ROBOT.2005.1570477.

G. Grisetti, C. Stachniss, and W. Burgard. Improved Techniques for Grid Mapping With Rao–Blackwellized Particle Filters. *Robotics, IEEE Transactions on*, 23(1):34–46, Feb. 2007. doi: 10.1109/TRO.2006.889486.

H. Gritli, S. Belghith, and N. Khraief. Ogy–based control of chaos in semi–passive dynamic walking of a torso–driven biped robot. *Nonlinear Dynamics*, 79(2):1363–1384, 2015. ISSN 1573-269X. doi: 10.1007/s11071-014-1747-9.

D. Hahnel, W. Burgard, D. Fox, and S. Thrun. An efficient fastSLAM algorithm for generating maps of large–scale cyclic environments from raw laser range measurements. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, volume 1, pages 206–211, 2003.

K. Hashimoto, T. Sawato, A. Hayashi, Y. Yoshimura, T. Asano, K. Hattori, Y. Sugahara, H. o. Lim, and A. Takanishi. Static and dynamic disturbance compensation control for a biped walking vehicle. In *2008 2nd IEEE RAS EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 457–462, Oct 2008. doi: 10.1109/BIOROB.2008.4762844.

J. Hernández-Aceituno, R. Arnay, J. Toledo, and L. Acosta. Using Kinect on an Autonomous Vehicle for Outdoors Obstacle Detection. *IEEE Sensors Journal*, 16(10):3603–3610, May 2016. ISSN 1530–437X. doi: 10.1109/JSEN.2016.2531122.

R. Hilfer. *Applications of Fractional Calculus in Physics*. World Scientific Publishing Company, 2000.

F. S. Hover and M. S. Triantafyllou. Application of polynomial chaos in stability and control. *Automatica*, 42(5):789–795, 2006. ISSN 0005-1098. doi: 10.1016/j.automatica.2006.01.010.

X. Huang and G. He. Forward kinematics of the general Stewart–Gough platform using Gröbner basis. In *2009 International Conference on Mechatronics and Automation*, pages 3557–3561, Aug 2009. doi: 10.1109/ICMA.2009.5246088.

M. Isard and A. Blake. CONDENSATION—Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*, 29 (1):5–28, 1998. ISSN 1573–1405. doi: 10.1023/A:1008078328650.

S. J. Julier and J. K. Uhlmann. New extension of the Kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068, pages 182–193, July 1997. doi: 10.1117/12.280797.

R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

B. H. Kaygisiz, M. Karahan, A. M. Erkmen, and I. Erkmen. *Robotic Approaches at the Crossroads of Chaos, Fractals and Percolation Theory*, pages

167–199. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-21922-1. doi: 10.1007/978-3-642-21922-1_6.

A. A. Kilbas, H. M. Srivastava, and J. J. Trujillo. *Theory and Applications of Fractional Differential Equations*. Elsevier, Holland, 2006.

T.-Y. Lee and J.-K. Shim. Algebraic Elimination–Based Real–Time Forward Kinematics of the 6–6 Stewart Platform with Planar Base and Platform. In *Robotics and Automation (ICRA), 2001 IEEE International Conference on*, volume 2, pages 1301–1306, 2001. doi: 10.1109/ROBOT.2001.932790.

G. Leonov and N. Kuznetsov. On differences and similarities in the analysis of lorenz, chen, and lu systems. *Applied Mathematics and Computation*, 256:334–343, 2015. ISSN 0096–3003. doi: 10.1016/j.amc.2014.12.132.

H. Liu, T. Oliphant, and L. Taylor. General fractional derivative viscoelastic models applied to vibration elastography. *Ultrasonics, 2003 IEEE Symposium on*, 1, October 2003.

J. S. Liu. Metropolized independent sampling with comparisons to rejection sampling and importance sampling. *Statistics and Computing*, 6(2):113–119, June 1996.

J. S. Liu and R. Chen. Sequential Monte Carlo Methods for Dynamic Systems. *Journal of the American Statistical Association*, 93:1032–1044, 1998.

K. Liu, J. M. Fitzgerald, and F. L. Lewis. Kinematic analysis of a Stewart platform manipulator. *IEEE Transactions on Industrial Electronics*, 40(2): 282–293, 1993. ISSN 0278-0046. doi: 10.1109/41.222651.

W. Liu, J. E. Fowler, and C. Zhao. Spatial logistic regression for support-vector classification of hyperspectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 14(3):439–443, March 2017. ISSN 1545-598X. doi: 10.1109/LGRS.2017.2648515.

E. N. Lorenz. Deterministic Nonperiodic Flow. *Journal of Atmospheric Sciences*, 20:130–148, Mar. 1963. doi: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2.

J. Lü and G. Chen. A New Chaotic Attractor Coined. *International Journal of Bifurcation and Chaos*, 12(03):659–661, 2002. doi: 10.1142/S0218127402004620.

Y. Luo, Q. Liu, X. Che, and Z. He. LMF Algorithm Based on Hyper–Chaos for the Solving of Forward Displacement in a Parallel Robot Mechanism. *International Journal of Advanced Robotic Systems*, 10(1):31, 2013. doi: 10.5772/54817.

S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart. A robust and modular multi–sensor fusion approach applied to mav navigation. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3923–3929, Nov 2013. doi: 10.1109/IROS.2013.6696917.

R. Martinez-Cantin and J. Castellanos. Unscented SLAM for large-scale outdoor environments. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 3427–3432, Aug. 2005.

J. McCarthy, M. L. Minsky, N. Rochester, and C. E. Shannon. *A Proposal for the Dartmouth Summer Research Project on Artificial Intelligence*. Dartmouth College, Hanover, New Hampshire, 1956.

A. L. Méhauté, J. Tenreiro-Machado, J. C. Trigeassou, and J. Sabatier. *Fractional Differentiation and its Applications*. Ubooks, Germany, 2005.

R. Metzler and J. Klafter. The random walk's guide to anomalous diffusion: a fractional dynamics approach. *Physics Reports*, 339:1–77, 2000. ISSN 0370-1573. doi: 10.1016/S0370-1573(00)00070-3.

Y. Minami Shiguematsu, A. Morell, K. Hashimoto, J. Toledo, L. Acosta, and A. Takanishi. Towards a sensorimotor system based on that of humans to study its effects on walking stabilization. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2015)*, Hamburg, Germany, October 2015. URL `https://iros2015wsperceptionandplanning.wordpress.com/program/`.

G. M. Mittag-Leffler. *Sur la nouvelle fonction E_α*, volume 137. C.R. Acad. Sci. Paris, 1903.

C. A. Monje, Y.-Q. Chen, B. M. Vinagre, D.-Y. Xue, and V. Feliu. *Fractional-Order Systems and Controls: Fundamentals and Applications.* Advances in Industrial Control. Springer–Verlag, London, 1^{st} edition, 2010.

N. Morales, R. Arnay, J. Toledo, A. Morell, and L. Acosta. Safe and reliable navigation in crowded unstructured pedestrian areas. *Engineering Applications of Artificial Intelligence*, 49:74–87, 2016a. ISSN 0952–1976. doi: 10.1016/j.engappai.2015.11.008.

N. Morales, A. Morell, J. Toledo, and L. Acosta. Fast Object Motion Estimation Based on Dynamic Stixels. *Sensors*, 16(8), 2016b. ISSN 1424–8220. doi: 10.3390/s16081182.

H. Moravec. Sensor fusion in certainty grids for mobile robots. *AI Magazine*, 9(2):61–74, Jul 1988. ISSN 0738-4602.

A. Morell, L. Acosta, and J. Toledo. An artificial intelligence approach to forward kinematics of stewart platforms. In *Control Automation*

(MED), 2012 20th Mediterranean Conference on, pages 433–438, July 2012. ISBN 978–1–4673–2531–8. doi: 10.1109/MED.2012.6265676.

A. Morell, M. Tarokh, and L. Acosta. Inverse kinematics solutions for serial robots using support vector regression. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4203–4208, May 2013a. doi: 10.1109/ICRA.2013.6631171.

A. Morell, M. Tarokh, and L. Acosta. Solving the forward kinematics problem in parallel robots using Support Vector Regression. *Engineering Applications of Artificial Intelligence*, 26(7):1698–1706, August 2013b. ISSN 0952–1976. doi: 10.1016/j.engappai.2013.03.011.

A. Morell, J. J. Trujillo, M. Rivero, and L. Acosta. Design of a fractional - order controller for the setpoint ramp tracking problem. In *American Control Conference (ACC), 2013*, pages 4245–4250, June 2013c. doi: 10.1109/ACC.2013.6580492.

A. Morell, A. Razminia, and J. J. Trujillo. Control of a novel fractional hyperchaotic system using a located control method. *Optimization: A Journal of Mathematical Programming and Operations Research*, 63(8, SI): 1219–1233, 2014. ISSN 0233–1934. doi: 10.1080/02331934.2014.891034.

K. Murphy. Bayesian Map Learning in Dynamic Environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 1015–1021, Denver, CO, USA, 1999. The M.I.T. Press.

F. Mustière, M. Bolić, and M. Bouchard. Rao–Blackwellised Particle Filters: Examples of Applications. In *Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on*, pages 1196–1200, may 2006.

Y. Nakamura and A. Sekiguchi. The chaotic mobile robot. *IEEE Transac-*

tions on Robotics and Automation, 17(6):898–904, Dec 2001. ISSN 1042-296X. doi: 10.1109/70.976022.

G. Narang, W. Kong, P. Xu, A. Narang, S. Singh, K. Hashimoto, M. Zecca, and A. Takanishi. Comparison of bipedal humanoid walking with human being using inertial measurement units and force-torque sensors. In *Proceedings of the 2013 IEEE/SICE International Symposium on System Integration*, pages 198–203, Dec 2013. doi: 10.1109/SII.2013.6776729.

U. Nehmzow and K. Walker. Quantitative description of robot–environment interaction using chaos theory. *Robotics and Autonomous Systems*, 53(3–4):177–193, 2005. ISSN 0921–8890. doi: https://doi.org/10.1016/j.robot.2005.09.009.

K. Ogata. *Modern Control Engineering*. Instrumentation and controls series. Prentice Hall, 5^{th} edition, 2010. ISBN 9780136156734.

K. B. Oldham and J. Spanier. *The Fractional Calculus: Theory and Applications of Differentiation and Integration to Arbitrary Order*, volume 111 of *Mathematics in Science and Engineering*. Academic Press, New York, 1974. ISBN 9780080956206.

A. Oustaloup. *La Dérivation Non Entière. Théorie, Synthèse et Applications*. Hermès Science, Paris, 1995.

L. Paz, P. Jensfelt, J. Tardos, and J. Neira. EKF SLAM updates in O(n) with divide and conquer SLAM. In *IEEE Int. Conf. Robotics and Automation*, pages 1657–1663, Apr. 2007.

D. Perea, J. Hernández-Aceituno, A. Morell, J. Toledo, A. Hamilton, and L. Acosta. MCL with sensor fusion based on a weighting mechanism versus a particle generation approach. In *Intelligent Transportation*

Systems (ITSC), 2013 16th International IEEE Conference on, pages 166–171, October 2013. ISBN 978-1-4799-2914-613. doi: 10.1109/ITSC.2013.6728228.

I. Petráš. Fractional–order feedback control of a DC motor. *Electrical Engineering, Journal of*, 60:117–128, 2009.

F. Pierrot, R. Clavel, and A. Fournier. DELTA: A simple and efficient parallel robot. *Robotica*, 8(22):105–109, 1990. doi: 10.1017/S0263574700007669.

I. Podlubny. *Fractional Differential Equations*, volume 198. Mathematics in Science and Engineering, 1999a.

I. Podlubny. Fractional-order systems and $PI^{\lambda}D^{\mu}$–controllers. *Automatic Control, IEEE Transactions on*, 44:208–214, Jan 1999b.

A. S. Ravishankar and A. Ghosal. Nonlinear Dynamics and Chaotic Motions in Feedback–Controlled Two–and Three–Degree–of–Freedom Robots. *The International Journal of Robotics Research*, 18(1):93–108, 1999. doi: 10.1177/027836499901800106.

F. Rosenblatt. The perceptron–a perceiving and recognizing automaton. Technical report, 85-460-1, Cornell Aeronautical Laboratory, 1957.

O. Rossler. An equation for hyperchaos. *Physics Letters A*, 71(2):155–157, 1979. ISSN 0375–9601. doi: 10.1016/0375-9601(79)90150-6.

B. Roth, J. Rastegar, and V. Scheinman. On the design of computer controlled manipulators. In *First CISM-IFTMM Symp. on Theory and Practice of Robots and Manipulators*, pages 93–113, 1973.

D. B. Rubin. Using the SIR algorithm to simulate posterior distributions. In M. H. Bernardo, K. M. Degroot, D. V. Lindley, and A. F. M. Smith,

editors, *Bayesian Statistics 3*, pages 395–402. Oxford University Press, 1988.

S. G. Samko, A. A. Kilbas, and O. I. Marichev. *Fractional Integrals and Derivatives: Theory and Applications*. Gordon and Breach Science Publishers, 1993.

K. Shojaie and A. Shahri. Iterated unscented SLAM algorithm for navigation of an autonomous mobile robot. In *IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pages 1582–1587, Sept. 2008.

B. Siciliano. The Tricept robot: Inverse kinematics, manipulability analysis and closed–loop direct kinematics algorithm. *Robotica*, 17:437–445, July 1999. ISSN 0263-5747. doi: 10.1017/S0263574799001678.

A. F. M. Smith and A. E. Gelfand. Bayesian Statistics without Tears: A Sampling–Resampling Perspective. *The American Statistician*, 46(2): 84–88, may 1992. ISSN 0003–1305. doi: 10.2307/2684170.

D. Stewart. A Platform with Six Degrees of Freedom. *Proc. of the Institute of Mechanical Engineering*, 180(part I(15)):171–186, 1965–1966.

P. Stone, R. Brooks, E. Brynjolfsson, R. Calo, O. Etzioni, G. Hager, J. Hirschberg, S. Kalyanakrishnan, E. Kamar, S. Kraus, K. Leyton-Brown, D. Parkes, W. Press, A. Saxenian, J. Shah, M. Tambe, and A. Teller. "Artificial Intelligence and Life in 2030." One Hundred Year Study on Artificial Intelligence. Technical report, 2015–2016 Study Panel, Stanford University, Stanford, CA, September 2016. URL `http://ai100.stanford.edu/2016-report`.

J. Tenreiro-Machado and A. Luo. Special Issue : Discontinuous and Fractional Dynamical Systems. *Computational and Nonlinear Dynamics, ASME Journal of*, 3(2):125, April 2008.

J. A. Tenreiro-Machado. Special Issue : Fractional Order Systems. *Nonlinear Dynamics*, 29(1–4):1–385, Kluwer, 2002.

J. A. Tenreiro-Machado. Effect of fractional orders in the velocity control of a servo system. *Computers and Mathematics with Applications*, 59: 1679–1686, 2010.

J. A. Tenreiro-Machado and R. Barbosa. Special Issue : Fractional Differentiation and its Applications. *Vibration and Control, Journal of*, 14: 1253–1672, Sage Pub, 2008.

J. A. Tenreiro-Machado, V. Kiryakova, and F. Mainardi. Recent history of fractional calculus. *Communications in Nonlinear Science and Numerical Simulation*, 16:1140–1153, 2011. ISSN 1007-5704. doi: 10.1016/j.cnsns.2010.05.027.

S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. Intelligent Robotics and Autonomous Agents. The M.I.T. Press, 2005.

S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney. Stanley: The robot that won the DARPA Grand Challenge. *Journal of Field Robotics*, 23(9):661–692, 2006. ISSN 1556–4967. doi: 10.1002/rob.20147.

L.-W. Tsai. *Robot Analysis and Design: The Mechanics of Serial and Parallel Manipulators*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1999. ISBN 0471325937.

J. J. Uicker, Jr., J. Denavit, and R. S. Hartenberg. An Iterative Method for the Displacement Analysis of Spatial Mechanisms. *Trans. of the ASME. Journal of Applied Mechanics*, 31(2):309–314, June 1964. doi: 10.1115/1.3629602.

V. Vapnik. *The nature of statistical learning theory*. Springer–Verlag New York, Inc., New York, NY, USA, 1995. ISBN 0-387-94559-8.

V. Vapnik, S. E. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In *Advances in Neural Information Processing Systems 9*, pages 281–287. The M.I.T. Press, 1996.

K. J. Waldron and K. H. Hunt. Series–parallel Dualities in Actively Coordinated Mechanisms. *Int. J. Rob. Res.*, 10(5):473–480, October 1991. ISSN 0278–3649. doi: 10.1177/027836499101000503.

Q. Wang, H. Chen, and Y. Shen. Decision tree support vector machine based on genetic algorithm for fault diagnosis. In *2008 IEEE International Conference on Automation and Logistics*, pages 2668–2672, September 2008. doi: 10.1109/ICAL.2008.4636624.

N. Wiener. The Homogeneous Chaos. *American Journal of Mathematics*, 60(4):897–936, October 1938. ISSN 1080–6377. doi: 10.2307/2371268.

N. Wiener. *The Extrapolation, Interpolation, and Smoothing of Stationary Time Series*. John Wiley & Sons, Inc., New York, N.Y., 1949.

A. T. Yang. Displacement Analysis of Spatial Five–Link Mechanisms Using (3x3) Matrices With Dual–Number Elements. *Trans. of the ASME. Journal of Engineering for Industry*, 91(1):152–156, February 1969. doi: 10.1115/1.3591499.

A. T. Yang and F. Freudenstein. Application of Dual–Number Quaternion Algebra to the Analysis of Spatial Mechanisms. *Trans. of the ASME. Journal of Applied Mechanics*, 31(2):300–308, June 1964. doi: 10.1115/1.3629601.

A. T. Yang and F. Freudenstein. Kinematic Analysis of Spatial Mechanisms by Means of Screw Coordinates. Part 2–Analysis of Spatial Mechanisms. *Trans. of the ASME. Journal of Engineering for Industry*, 93 (1):67–73, February 1971. doi: 10.1115/1.3427919.

Research achievements

Publications

12 publications: 5 Journal proceedings, 4 International Conference proceedings (peer–reviewed), 1 International Workshop proceedings (peer–reviewed), 2 other publications.

Journal proceedings

2013

- **A. Morell**, M. Tarokh, and L. Acosta. Solving the forward kinematics problem in parallel robots using support vector regression. *Engineering Applications of Artificial Intelligence*, 26(7):1698–1706, August 2013b. ISSN 0952–1976. doi: 10.1016/j.engappai.2013.03.011.

2014

- **A. Morell**, A. Razminia, and J. J. Trujillo. Control of a novel fractional hyperchaotic system using a located control method. *Optimization: A Journal of Mathematical Programming and Operations Research*, 63(8, SI):1219–1233, 2014. ISSN 0233–1934. doi: 10.1080/02331934.2014.891034.

2016

- R. Arnay, N. Morales, **A. Morell**, J. Hernandez-Aceituno, D. Perea, J. T. Toledo, A. Hamilton, J. J. Sanchez-Medina, and L. Acosta. Safe and Reliable Path Planning for the Autonomous Vehicle Verdino. *IEEE Intelligent Transportation Systems Magazine*, 8(2):22–32, 2016. ISSN 1939–1390. doi: 10.1109/MITS.2015.2504393.

- N. Morales, R. Arnay, J. Toledo, **A. Morell**, and L. Acosta. Safe and reliable navigation in crowded unstructured pedestrian areas. *Engineering Applications of Artificial Intelligence*, 49:74–87, 2016a. ISSN 0952–1976. doi: 10.1016/j.engappai.2015.11.008.

- N. Morales, **A. Morell**, J. Toledo, and L. Acosta. Fast Object Motion Estimation Based on Dynamic Stixels. *Sensors*, 16(8), 2016b. ISSN 1424–8220. doi: 10.3390/s16081182.

International Conference proceedings

2012

- **A. Morell**, L. Acosta, and J. Toledo. An artificial intelligence approach to forward kinematics of stewart platforms. In *Control Automation (MED), 2012 20th Mediterranean Conference on*, pages 433–438, July 2012. doi: 10.1109/MED.2012.6265676.

2013

- **A. Morell**, M. Tarokh, and L. Acosta. Inverse kinematics solutions for serial robots using support vector regression. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 4203–4208, May 2013a. doi: 10.1109/ICRA.2013.6631171.

- **A. Morell**, J. J. Trujillo, M. Rivero, and L. Acosta. Design of a fractional–order controller for the setpoint ramp tracking problem. In *American Control Conference (ACC), 2013*, pages 4245–4250, June 2013c. doi: 10. 1109/ACC.2013.6580492.

- D. Perea, J. Hernández-Aceituno, **A. Morell**, J. Toledo, A. Hamilton, and L. Acosta. MCL with sensor fusion based on a weighting mechanism versus a particle generation approach. In *Intelligent*

Transportation Systems (ITSC), 2013 16th International IEEE Conference on, pages 166–171, October 2013. ISBN 978-1-4799-2914-613. doi: 10.1109/ITSC.2013.6728228.

International Workshop proceedings

2015

- Y. Minami Shiguematsu, **A. Morell**, K. Hashimoto, J. Toledo, L. Acosta, and A. Takanishi. Towards a sensorimotor system based on that of humans to study its effects on walking stabilization. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, Hamburg, Germany, October 2015. URL https://iros2015wsperceptionandplanning.wordpress.com/program/.

Other proceedings

2015

- **A. Morell**, L. Acosta. *Actas del I Congreso de Jóvenes Investigadores de Canarias*. April 2015 (domestic conference / in Spanish).
- **A. Morell**. Estabilización dinámica de sistemas sensoriales en movimiento: Aplicación a un robot humanoide. *Actas del I Congreso de Estudiantes de Ingeniería Informática de la Universidad de La Laguna*. December 2015 (domestic conference / in Spanish).

Invited talks

2015

- Robotics Group of *Universidad de La Laguna* (GRULL). Technical demonstration: Autonomous Driving in Unstructured Areas with

Pedestrians. *18th IEEE International Conference on Intelligent Transportation Systems*. Las Palmas de Gran Canaria, Spain, September 2015.

Grants

2013

- Postgraduate scholarship grant for developing a Doctoral Thesis, partially funded by ESF (European Social Funds). Awarded by Resolution of the Agency of Research, Innovation and Information Society (ACIISI) of the Government of Canary Islands, June 6th, 2013. Official Bulleting of Canary Islands, No. 119/2013.
  - Total Amount: 30 126€
  - Host Institution: Universidad de La Laguna
  - Thesis title: Dynamic state estimation for mobile robots
  - Supervisors: Leopoldo Acosta Sánchez, Jonay Tomás Toledo Carrillo

2015

- Grant for short research internship stays. Awarded by Resolution of the Agency of Research, Innovation and Information Society (ACIISI) of the Government of Canary Islands, June 15th, 2015. Resolution No. 237 / 2015.
  - Total Amount: 7 842€
  - Host Faculty: Professor Atsuo Takanishi
  - Host Organization: Graduate School of Advanced Science and Engineering
  - Status: Visiting Research Fellow

- Research Theme: Improvements to Sensory Systems on the Move: Application to a Biped Humanoid Robot.
- Affiliation Period: May 4, 2015 – September 3, 2015 (4 months)

Part II

Further contributions

A

MCL sensor fusion weighting mechanism versus a generation approach

Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013), The Hague, The Netherlands, October 6-9, 2013

MoB6.4

MCL with sensor fusion based on a weighting mechanism versus a particle generation approach

Daniel Perea, Javier Hernández–Aceituno, Antonio Morell,
Jonay Toledo, Alberto Hamilton and Leopoldo Acosta (*Member, IEEE*)

***Abstract*— The combined action of several sensing systems, so that they are able to compensate the technical flaws of each other, is common in robotics. Monte Carlo Localization (MCL) is a popular technique used to estimate the pose of a mobile robot, which allows the fusion of heterogeneous sensor data. Several sensor fusion schemes have been proposed which include sensors like GPS to improve the performance of this algorithm. In this paper, an Adaptive MCL algorithm is used to combine data from wheel odometry, an inertial measurement unit, a global positioning system and laser scanning. A particle weighting model which integrates GPS measurements is proposed, and its performance is compared with a particle generation approach. Experiments were conducted on a real robotic car within an urban environment.**

I. Introduction

Localization is one of the most relevant problems in mobile robotics, specially in outdoor and urban areas. The information obtained from sensor devices might not be as accurate as expected, so it is of great importance to define algorithms that are robust to such problems. Specifically using a known map of static obstacles, dynamic obstacles are also very likely to appear and might add uncertainty to localization algorithms. Using a single source of sensing is not practical, thus many different forms of sensor fusion have been proposed. Multiple devices can mitigate the drawbacks a single sensor might have, especially by combining proprioceptive and exteroceptive measurements, such as odometry and global positioning systems (GPS).

A review of the literature shows that Laser Imaging Detection and Ranging (LIDAR) sensors are very popular because of their data update frequency and precision. They have been combined with wheel odometry, visual cameras [1], stereovision, GPS [2], three–dimensional geographic information systems (3D–GIS) [3], and combinations thereof [4]–[6]. A common localization algorithm which integrates the information provided by different sensors is the Monte Carlo Localization (MCL) [7] method. It is based on particle filters (PF), whose samples (or particles) are weighted according to their likelihood computed from each available device [8]. Fusion of wheel odometry and GPS using MCL has been studied before [9], including omnidirectional vision [10], LIDAR sensors [11], [12], and inertial sensors [13]. Several adaptive variations of MCL with a variable number on PF samples have been proposed, such as Bayesian Bootstrap Filtering [14], Self–Adaptive MCL (SAMCL) [15], or Merge–MCL [16]. Adaptive Monte Carlo Localization (AMCL) [17], [18] optimaly adapts the number of samples of the PF by means of Kullback–Leibler divergences (KLD) [19].

This work showcases how GPS sensor fusion methods are affected by *multipath interferences*, a phenomenon that leads to misplaced reports of GPS sensors. We propose a fusion of wheel odometry, an Inertial Measurement Unit (IMU), GPS and LIDAR using the AMCL algorithm. A particle weighting model which integrates GPS measurements is proposed, and its performance is compared with a sample generation approach. On the sample generation method, new particles are added to the PF when new absolute measurementes are obtained, whereas our proposal uses this information as a weighting function over the existing particles. Experiments were conducted on both approaches, and the results reflect the robustness and better performance of our implementation.

This paper is organized as follows. Section II briefly describes the AMCL algorithm which solves the localization problem. Section III presents and discusses the proposed GPS integration method. The particle generation approach is described in Sections IV, and the robustness of our proposal is discussed and compared with the former. The mobile robot platform and the experiments conducted are described in Section V. Finally, the most relevant conclusions are summarized in Section VI.

II. Localization with Particle Filters

The state–space of the localization problem on 2–D maps $\boldsymbol{x}_t$ is given by the position of the robot, as a pair of Cartesian coordinates (x, y), and its orientation angle (θ). The AMCL algorithm finds an estimation of the posterior $p(\boldsymbol{x}_t \mid z_t)$ for a mobile robot at time t, based on the observations (typically LIDAR scans) z_{t-1}.

A classical Monte Carlo method for solving the Bayesian filtering problem is the Bayesian importance sampling. One of the most popular particle filtering schemes is the *Sampling Importance Resampling* (SIR) algorithm [20], [21]. Typically, an AMCL algorithm merges wheel odometry and LIDAR information as they are available. A set of particles or samples represents the posterior about the trajectory of the robot, which are updated following the SIR algorithm, as follows:

Daniel Perea (corresponding author), Javier Hernández–Aceituno, Antonio Morell, Jonay Toledo, Alberto Hamilton and Leopoldo Acosta are with Departamento Ingeniería de Sistemas y Automática y Arquitectura y Tecnología de Computadores Universidad de La Laguna, 38203 La Laguna, Tenerife, Spain. E–mail address: dani at isaatc.ull.es, javier at isaatc.ull.es, amorell at isaatc.ull.es, jonay at isaatc.ull.es, alberto at isaatc.ull.es, lacosta at ull.es

978-1-4799-2914-613/$31.00 ©2013 IEEE

1) *Sampling:* A new generation of particles $\{\boldsymbol{x}_t^{(i)}\}$ is obtained from the previous generation $\{\boldsymbol{x}_{t-1}^{(i)}\}$ by sampling from a proposal distribution $\boldsymbol{\pi}$. A probabilistic odometry motion model $p(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}, u_{t-1})$ is used as such proposal distribution, where u_{t-1} is the odometry measurement at time $t-1$.
2) *Importance weighting:* The importance weight of a pose is a dimensionless value related to how likely is that the robot is located at $\boldsymbol{x}_t$. LIDARs measurements are used to compute the importance weight of each particle as follows:

$$w_t^{(i)} = p(z_t \mid \boldsymbol{x}_t^{(i)}) \tag{1}$$

All weights satisfy:

$$\sum_{i=1}^{N} w_t^{(i)} = 1 \quad , \tag{2}$$

on any time t, where N is the total number of samples on the PF.
3) *Resampling:* Particles with a low importance weight are more likely to be replaced by those with a higher weight. This step allows maintaining a discrete set of particles which approximate a continuous distribution, since a high number of particles on the PF is not computationally efficient. The optimal number of particles is typically given by Kullback–Leibler divergences.

III. GPS Integration

Robot localization in outdoor environments usually takes advantage of GPS devices. While providing useful information, GPS measurements might be misleading or even completely erroneous in some circumstances. The most relevant sources of error in urban scenarios are *multipath phenomena*.

Multipath is the propagation phenomenon that results in radio signals, e.g., satellite signals, reaching the receiving antenna by two or more paths. This happens mostly due to reflection and refraction phenomena, i.e., from water bodies and terrestrial objects such as mountains and buildings, resulting in a sudden "jump" in the GPS position estimate. Such source of error should not be ignored, given its negative impact on GPS resulting readings.

We consider GPS measurements given by 2–D Cartesian coordinates (x, y) and their respective covariances, as a *Universal Transverse Mercator* (UTM) projection from the *World Geodetic System* (WGS84) ellipsoid. In addition, an approximation of the orientation (*yaw*) of the robot is given by the Course–Over–Ground (COG) as the orientation (θ) of the vector between consecutive GPS positions, assuming only longitudinal movement of the robot. The covariance of this orientation is also available. When a multipath event is detected, which yields a high covariance on the current reported position, the orientation angle and its covariance are taken from an IMU device. In spite of the fact that IMU orientation reports are not very accurate, (compared with the COG value provided by a differential rover GPS), they are still useful when these reports are not available or are not valid. For example, in situations where the robot has stopped or is moving very slowly, the difference between two consecutive positions does not yield a valid orientation. A multipath event also causes the GPS device to report erroneous COG measurements.

The key idea of the GPS integration approach is to weight the existing particles considering both the pose estimation and the associated covariances reported by a GPS device. This new observation source z_t^{GPS} provides the following position and orientation parameters at a time t:

$$z_t^{\text{GPS}} = \left\langle \begin{bmatrix} \boldsymbol{\mu} \\ \mu_\theta \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma} & \mathbf{0} \\ \mathbf{0} & \sigma_\theta \end{bmatrix} \right\rangle \quad , \tag{3}$$

where $\boldsymbol{\mu} = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}$ and $\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$.

Assuming that position (x, y) and orientation (θ) are uncorrelated and follow a Gaussian Probability Density Function (PDF), the posterior given a GPS reading can be obtained as

$$p(\boldsymbol{x}_t \mid z_t^{\text{GPS}}) = f(x, y) \cdot f_{\text{WN}}(\theta) \quad , \tag{4}$$

where the PDF for the position is

$$f(x, y) = \frac{1}{2\pi \left|\boldsymbol{\Sigma}\right|^{1/2}} e^{-\frac{1}{2}\left(\begin{bmatrix} x-\mu_x \\ y-\mu_y \end{bmatrix}^T \boldsymbol{\Sigma}^{-1} \begin{bmatrix} x-\mu_x \\ y-\mu_y \end{bmatrix} \right)} \quad , \tag{5}$$

and the PDF corresponding to the orientation angle, which follows a wrapped normal distribution, is

$$f_{\text{WN}}(\theta) = \frac{1}{\sigma_\theta \sqrt{2\pi}} \sum_{k=-\infty}^{\infty} e^{-(\theta - \mu_\theta + 2\pi k)^2 / 2\sigma_\theta^2} \quad . \tag{6}$$

Instead of adding new particles to the PF, the existing set of particles are weighted according to (4) and LIDAR measurements z_t^{LIDAR}, which are conditionally independent of past measurements given knowledge of the state $\boldsymbol{x}_t$ [17]. Therefore, the new posterior is:

$$p(\boldsymbol{x}_t \mid z_t) = p(\boldsymbol{x}_t \mid z_t^{\text{LIDAR}}) \cdot p(\boldsymbol{x}_t \mid z_t^{\text{GPS}}) \, . \tag{7}$$

In the first iteration of the algorithm there is no initial hypothesis available. Although our method does not generate GPS–based particles, an initial particle set is needed. Thus, a particle set is created and distributed following the first GPS measurement. If a *kidnapped robot* event takes place, new particles could be added similarly to overcome this problem.

IV. GPS Particle Generation

When a GPS measurement is reported, it seems natural to add a new particle cluster to the PF [13]. A new set of samples is drawn from the Gaussian PDF centered at the GPS position. The m particles with the lowest weight in the filter are replaced with the new sample set. Unfortunately, adding new particles to the PF introduces some flaws. Assuming a correct initial robot localization, if the GPS output greatly differs from the current hypothesis, it may imply that a multipath interference has happened. Misplaced

978-1-4799-2914-613/$31.00 ©2013 IEEE

GPS reports should be discarded by the PF in order to not accept misleading hypotheses, incompatible with the current robot location. These misleading hypotheses yield incorrect output in the PF in situations where not enough significant landmarks are available in LIDAR scans. This behaviour can be seen in experiment 1 (Section V-A).

Following the proposed GPS weighting scheme, the aforementioned problems will not arise, and there will not be conflicting and ambiguous hypotheses which will eventually lead to a wrong robot localization.

During a multipath event, where a GPS measurement usually drifts from the actual robot position, the covariance values might not be properly delivered. In our implementation, the GPS weighting scheme behaves as follows:

- For multipath events where narrow covariances are incorrectly reported, particle weights will not be greatly affected, as the resulting gaussian model will have its mean far away from the position of the particles.
- When covariances are wide, the resulting GPS–centered gaussian model will affect more noticeably to the weighting of correctly localized particles, but we cannot just reject these reports. A high covariance is not only due to multipath interference phenomena, but also due to low number of visible satellites or bad constellation geometry. It is safer to treat them as any other GPS pose report.

It is worth noting that these two cases are not handled separately by the PF in our implementation. The GPS weighting is performed with every report received from the device with their respective covariance values. The PF handles them naturally in either case, in conjunction with the odometry model and the LIDAR weighting scheme.

Finally, if the mean of the reported GPS position is at the current localization hypothesis, there would not be a meaningful difference between the particle generation method and our implementation, because the current particle distribution would already include that GPS position.

V. Experiments

The experiments has been conducted on a test platform called VERDINO (Fig. 2), a fully electric two seat vehicle, based on an *EZ–GO TXT–2* golf cart. It is designed for passenger transportation and surveillance in non–structured environments. The vehicle has been modified by adding several sensors and actuators, which allows performing navigation tasks through urban areas.

Its sensorial system includes two differential GPS Javad Triumph–1 devices. The first one is a Rover GPS unit mounted on top of the vehicle, and the second one is a fixed Base station. With its position accurately defined, the Base is used for estimating the error introduced by each satellite, in order to send the corresponding corrections to the moving Rover unit. In addition, an IMU device aids during the estimation of the orientation of the vehicle, together with the Course–Over–Ground (COG) reporting capability of the GPS device. Finally, the robot includes two horizontal Sick LMS111 laser range finders, with a maximum range of 20

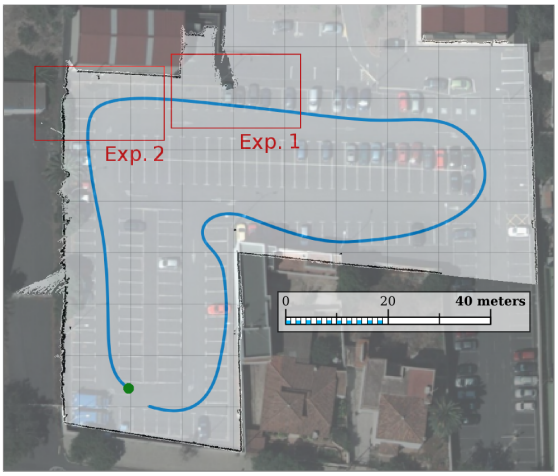


Fig. 1. Static localization map, with ground truth (blue) and experiment regions (red) identified. The stating position is marked as a green dot. The map covers an area of approximately $90\,m$ by $90\,m$.

Fig. 2. VERDINO prototype.

meters, and a wheel odometry system. It should be noted that our odometry sensor clearly suffers from a left drift during all experiments, noticeable in experiment 3. However, our method correctly handles these flawed reports.

The experiments were performed at the parking lot of the Computer Science Faculty of our campus, where VERDINO followed the path shown in Fig. 1, which we consider as our ground truth. This path was recorded under continuous and accurate GPS readings, with a reported position covariance under $0.02\,m$. It was inspected to guarantee that no multipath events occurred. The route was traced on the ground, and the vehicle was manually driven along it during the experiments. We assume that the error caused by reproducing the path this way is about $0.3\,m$ in the worst case scenario, way below a typical GPS measurement "jump" caused by a multipath event. The map used for Monte Carlo localization is a previously captured model of the static obstacles in the environment, which was georeferenced against the local vector topographic map from the Spatial Data Infrastructure of Canary Islands (IDECanarias).

978-1-4799-2914-613/$31.00 ©2013 IEEE

Two different experiments where multipath was present were conducted in order to compare the reliability of the localization. In order to verify the GPS contribution, a final experiment compares our approach with an AMCL implementation without GPS integration. The allowed maximum number of particles was 2000 for the GPS particle generation implementation, and 500 for the GPS weighting approach. The maximum localization error obtained with GPS particle generation was 3.72 meters away from the ground truth, whereas the error yielded by our approach was 0.22 meters. All experiments are described in detail in the following subsections.

A. Multipath with no static references

In the first experiment, the robot traverses a region of the map where no static references are within the range of the LIDAR system, and therefore the position estimation must rely exclusively on wheel odometry and GPS. The GPS sensor suffers from multipath interference and reports a misplaced absolute position. Fig. 3 compares the performance of both approaches under these circumstances. Fig. 3a shows the localization output of the PF with GPS particle generation, while Fig. 3b shows the localization output with GPS weighting only, as we propose. The dispersion of the particle set in the first case is higher, as the GPS particle generation continuously introduces new particles based on the sensor covariance, while the particle set is densely concentrated in the second case.

The experiment starts with the PF yielding a correct trajectory estimate on both cases. In Fig. 3a, when the multipath interference appears on the GPS, a new particle cluster is generated some meters away from the correct trajectory. LIDAR scans have no features to match against the map, while more incorrect GPS readings are received. As the PF performance decreases, more particles are generated. The PF evolves and switches its current hypothesis to the new cluster, producing a displaced trajectory. When multipath disappears, the PF switches again to the correct path. This incorrect behaviour is present in other implementations of GPS integration on MCL algorithms where new particles are added [13]. As can be seen on Fig. 3b, the output is correct in our implementation, in spite of suffering from multipath interference. By not adding new particles, only existing ones are considered and the GPS weighting is very low, given the distance to the displaced readings.

B. Valid LIDAR measurements with multipath

During the second experiment (see Fig. 4), the robot traverses an area where obstacles are visible, thus LIDAR scans can be matched with the map. Here, multipath interference also appears, but this time the output is different from Fig. 3. In Fig. 4a the newly created cluster does not affect the output of the PF, as the LIDAR scan matching has more influence than the GPS covariance weighting. Nonetheless, a great number of particles are always present on the wrong track during multipath, and each one is propagated with the odometry model and weighted according to each sensor reading. This produces a waste of processing power, and it is a source of potential errors in scan matching that may mislead the PF hypothesis. In Fig. 4b the output follows the trajectory closer, keeping a narrower particle set. The increased confidence in the hypothesis reduces the number of particles in the current set, freeing up CPU resources.

C. AMCL without GPS integration

A known, noticeable left drift in our odometry sensor is present in all of the experiments. When comparing our method with an implementation of AMCL without GPS integration, this flaw is corrected in both cases thanks to the observable LIDAR landmarks, as can be seen on the left half of Fig. 5. However, when these LIDAR references are not available, the output of the AMCL algorithm without GPS weighting drifts away. Due to the lack of information, the dispersion of the particle distribution grows over time. Meanwhile, thanks to GPS integration, the PF hypothesis of our proposal follows the ground truth closely.

VI. Conclusion

We have compared two different ways of combining GPS/IMU and LIDAR measurements within an AMCL based scheme which fuses data from multiple sensors. Experiments show that adding new particles to a PF on MCL algorithms are likely to create clusters which might trigger a correctly localized hypothesis to change to a wrong location. Our approach uses the provided global location information to weight the existing particles, combined with the corresponding likelihood obtained after the weighting process using LIDARs information.

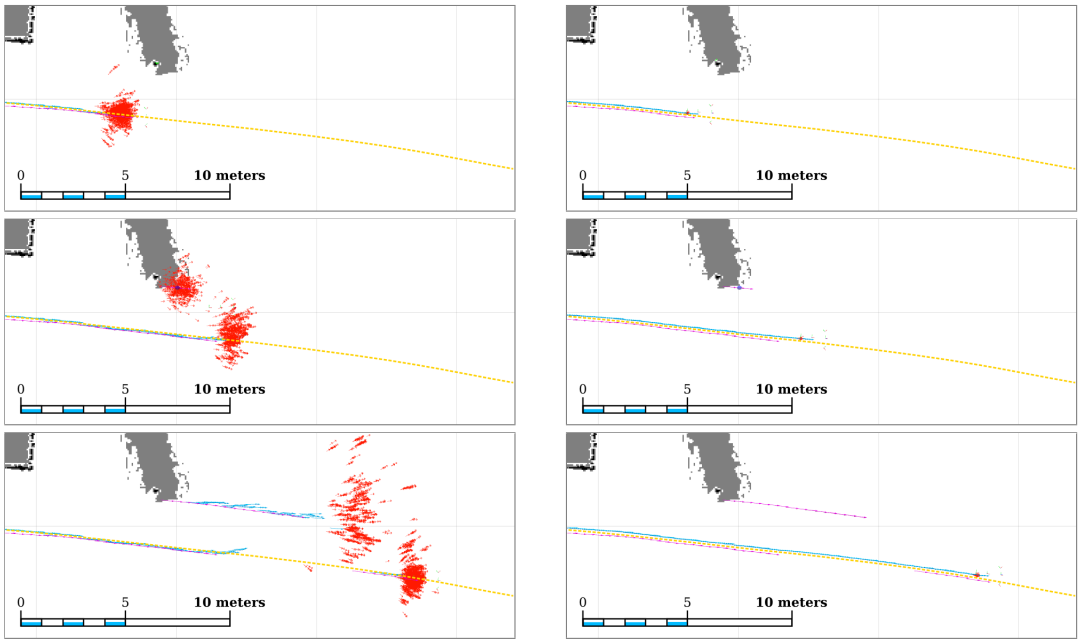
The maximum localization error yielded by the GPS particle generation method is below the assumed ground truth reproducibility margin. Our experiments show that, although more intuitive, adding new particles ignores the evolution over time of the calculated robot position. Therefore, its reliability is worsen whenever the GPS measurements are not accurate enough, e.g., when multipath interference occurs. Our approach combines all sensor readings successfully, avoiding large hypothesis changes.

Furthermore, by integrating GPS information with our method, the robustness of the hypothesis estimation increases against LIDAR measurements uncertainties and odometry unmodeled behaviours, such as drift. At the same time, the number of particles needed for a correct localization is lower, as the dispersion of the particle set is reduced.

Acknowledgment

The authors gratefully acknowledge the contribution of the Spanish Ministry of Economy and Competitiveness under Project SAGENIA DPI2010–18349. The research of both Daniel Perea and Antonio Morell is supported by a postgraduate grant from the *"Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI)"* and funds from the ESF. Javier Hernández–Aceituno's research is supported by the postgraduate grant *"Subprograma de Formación del Profesorado Universitario (FPU)"* from the Ministry of Economy and Competitiveness of Spain.

978-1-4799-2914-613/$31.00 ©2013 IEEE



(a) GPS particle generation (b) GPS weighting

Fig. 3. Experiment 1 – The dispersion is higher on (a), since the number of particles is also high (2000) and they are generated based on GPS measurements, while the robot traverses an area without enough spatial references, i.e., LIDAR readings do not report obstacles nearby. When applying the weighting approach (b), a maximum of 500 particles manage to keep the robot localized, even while the GPS suffers from a multipath event. Light blue arrows: the robot trajectory estimated by the PF; magenta arrows: GPS readings; blue ellipse: covariance of GPS reported positions; red arrows: the particle set distribution at any given time; yellow dotted line: ground truth.

REFERENCES

[1] P. Newman, D. Cole, and K. Ho, "Outdoor SLAM using Visual Appearance and Laser Ranging," in *IEEE Int. Conf. Robotics and Automation*, May 2006, pp. 1180–1187.

[2] L. Wei, C. Cappelle, and Y. Ruichek, "Unscented Information Filter Based Multi–Sensor Data Fusion using Stereo Camera, Laser Range Finder and GPS Receiver for Vehicle Localization," in *XIV Int. IEEE Conf. Intell. Transportation Systems*, Oct. 2011, pp. 1923–1928.

[3] J. Peng, M. El Najjar, C. Cappelle, D. Pomorski, F. Charpillet, and A. Deeb, "A Novel Geo–Localisation Method using GPS, 3D–GIS and Laser Scanner for Intelligent Vehicle Navigation in Urban Areas," in *Int. Conf. Advanced Robotics*, June 2009, pp. 1–6.

[4] C. Cappelle, M. El Badaoui El Najjar, F. Charpillet, and D. Pomorski, "Outdoor Obstacle Detection and Localisation with Monovision and 3D Geographical Database," in *IEEE Intell. Transportation Systems Conference*, Oct. 2007, pp. 1102–1107.

[5] L. Wei, C. Cappelle, and Y. Ruichek, "Localization of Intelligent Ground Vehicles in Outdoor Urban Environments using Stereovision and GPS Integration," in *XV Int. Conf. Advanced Robotics*, June 2011, pp. 192–197.

[6] C. Smaili, M. El Najjar, and C. Francois, "A Chained Form State Representation for Outdoor Vehicle Localisation," in *XIV Int. IEEE Conf. Intell. Transportation Systems*, Oct. 2011, pp. 1386–1391.

[7] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," in *XVI Nat. Conf. Artificial Intell., XI Innovative Applications of Artificial Intell. Conf.* American Association for Artificial Intelligence, 1999, pp. 343–349.

[8] D. Silver and A. Stentz, "Monte Carlo Localization and Registration to Prior Data for Outdoor Navigation," in *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, Sept. 2011, pp. 510–517.

[9] M. Moreira, H. Machado, C. Mendonca, and G. Pereira, "Mobile Robot Outdoor Localization using Planar Beacons and Visual Improved Odometry," in *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, Nov. 2007, pp. 2468–2473.

[10] E. Frontoni, A. Ascani, A. Mancini, and P. Zingaretti, "Robot Localization in Urban Environments using Omnidirectional Vision Sensors and Partial Heterogeneous apriori Knowledge," in *IEEE/ASME Int. Conf. Mechatronics and Embedded Systems and Applications*, July 2010, pp. 428–433.

[11] H. Chang, J. Choi, and M. Kim, "Experimental Research of Probabilistic Localization of Service Robots using Range Image Data and Indoor GPS System," in *IEEE Conf. Emerging Technologies and Factory Automation*, Sept. 2006, pp. 1021–1027.

[12] ——, "Probabilistic Localization of Service Robot by Sensor Fusion," in *Int. Joint Conf.*, Oct. 2006, pp. 3626–3631.

[13] M. Hentschel, O. Wulf, and B. Wagner, "A GPS and Laser–Based Localization for Urban and Non–Urban Outdoor Environments," in *IEEE/RSJ Int. Conf. Intell. Robots and Systems*, Sept. 2008, pp. 149–154.

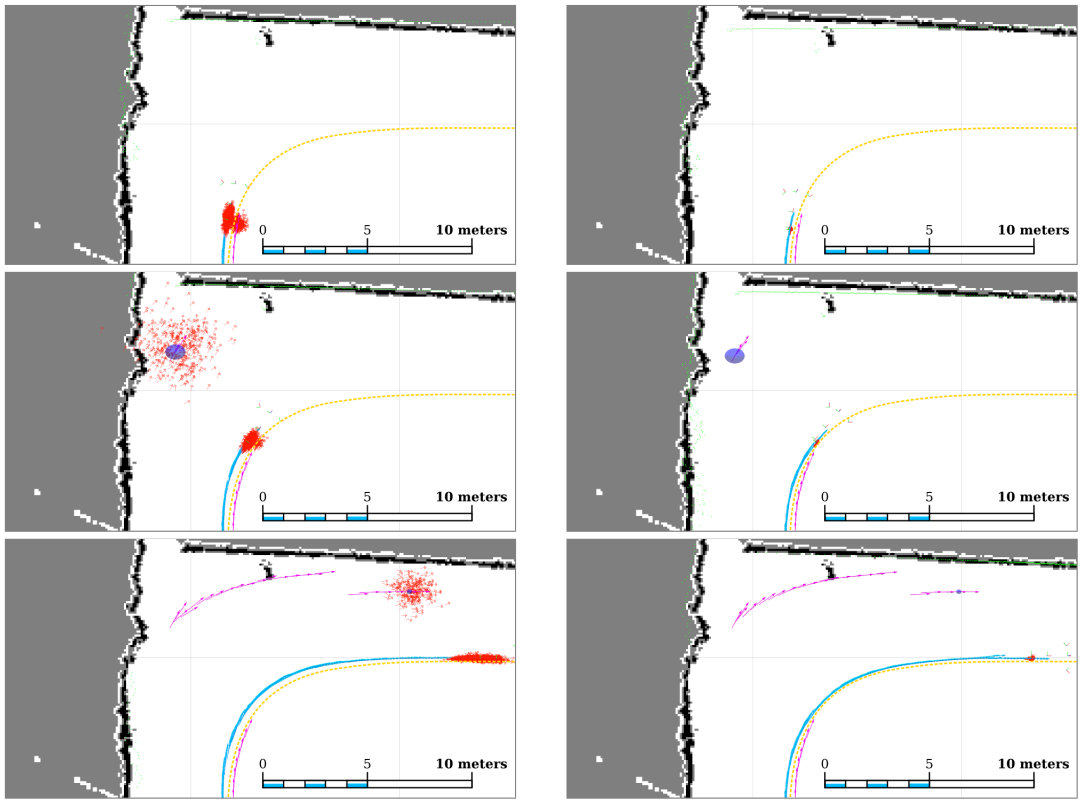
[14] T. Khalid, Z. Mourad, C. Jean-Bernard, and B. Mohammed, "Bayesian Bootstrap Filter for Integrated GPS and Dead Reckoning Positioning," in *IEEE Int. Symposium on Industrial Electronics*, June 2007, pp. 1520–1524.

[15] L. Zhang, R. Zapata, and P. Lepinay, "Self–Adaptive Monte Carlo for Single–Robot and Multi–Robot Localization," in *IEEE Int. Conf. Automation and Logistics*, Aug. 2009, pp. 1927–1933.

[16] T. Li, S. Sun, and J. Duan, "Monte Carlo Localization for Mobile Robot using Adaptive Particle Merging and Splitting Technique," in *IEEE Int. Conf. Information and Automation*, June 2010, pp. 1913–1918.

[17] D. Fox, "Adapting the Sample Size in Particle Filters Through KLD-Sampling," *International Journal of Robotics Research*, vol. 22, no. 12, pp. 985–1004, 2003.

978-1-4799-2914-613/$31.00 ©2013 IEEE



(a) GPS particle generation (b) GPS weighting

Fig. 4. Experiment 2 – Particles are once again widely spread on the particle generation approach (a) due to the higher number of particles at the GPS position estimate. Unlike experiment 1, the hypothesis does not jump after the multipath event, thanks to LIDAR readings matching a corner successfully. On (b), the particle set is narrower and also does not suffers from an undesired jump. Light blue arrows: the robot trajectory estimated by the PF; magenta arrows: GPS readings; blue ellipse: covariance of GPS reported positions; red arrows: the particle set distribution at any given time; green dots: LIDAR scans; yellow dotted line: ground truth.

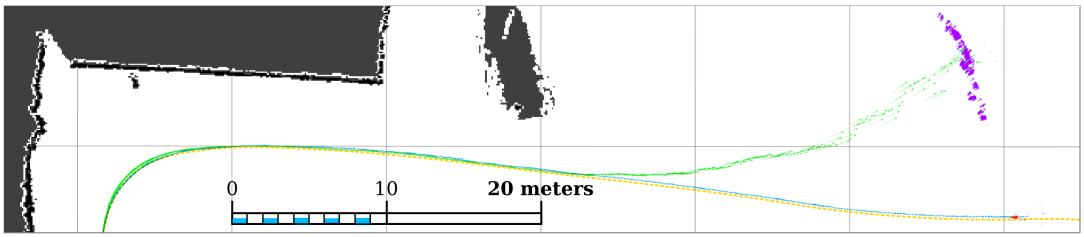


Fig. 5. Experiment 3 – Comparison between an implementation of AMCL that does not integrate GPS measurements and our proposal. Light blue arrows: robot trajectory computed by our approach; green arrows: robot trajectory calculated by AMCL without GPS readings; red arrows: particle set distribution generated by our approach; purple arrows: particle set distribution generated without GPS; yellow dotted line: ground truth.

[18] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, ser. Intelligent Robotics and Autonomous Agents. MIT Press, 2005.

[19] S. Kullback and R. A. Leibler, "On Information and Sufficiency," *Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[20] D. B. Rubin, "Using the SIR algorithm to simulate posterior distributions," in *Bayesian Statistics 3*, M. H. Bernardo, K. M. Degroot, D. V. Lindley, and A. F. M. Smith, Eds. Oxford University Press, 1988, pp. 395–402.

[21] A. F. M. Smith and A. E. Gelfand, "Bayesian Statistics without Tears: A Sampling–Resampling Perspective," *The American Statistician*, vol. 46, no. 2, pp. 84–88, may 1992.

978-1-4799-2914-613/$31.00 ©2013 IEEE

B

Towards sensorimotor system to study its effects on walking stabilization

Towards a Sensorimotor System Based on that of Humans to Study its Effects on Walking Stabilization

Y. Minami Shiguematsu[1], A. Morell[2], K. Hashimoto[3], J. Toledo[2], L. Acosta[2] and A. Takanishi[4]

minami.yukitoshi@ruri.waseda.jp amorell@isaatc.ull.es

[1] Graduate School of Science and Engineering, Waseda University, Tokyo, Japan
[2] Departament of Computer Science and Systems Engineering University of La Laguna, Santa Cruz de Tenerife, Spain
[3] Waseda Institute for Advanced Study, Waseda University, Tokyo, Japan
[4] Department of Modern Mechanical Engineering and Humanoid Robotics Institute (HRI), Waseda University, Tokyo, Japan

Background

There are many researches about bipedal walking stabilization using humanoid robots. For this, there are researches using inertial sensors mounted on the body of the robot and force/torque sensors on the feet, or others using vision sensors. However, humans use visual sensors (eyes), inertial sensors (vestibular system) and force/torque sensors (proprioceptive system) and combine the information from them to walk stably. Moreover, both visual and vestibular systems are located on the head, and humans are said to change the weight they put on each sensory input depending on the situation (Fig. 1) as each of the sensory systems has a different operating frequency (Table 1), e.g., relying more on vision when walking slowly and relying more on the vestibular system when walking fast.

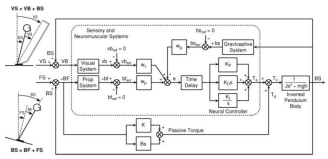


Fig. 1 "Independent channel" model of sensory integration in postural control. [Peterka, 2002]

Table 1 Characteristics of the sensory systems used in postural control. [Redfern, 2001]

Sensory system	Approximate frequency range	Examples of behavior or situation
Vision	< 0.1Hz (Very slow movements)	•Standing in a room without any movement in the visual field. •Standing near a bus that is moving slowly in the visual field.
Vestibular otoliths	< 0.5 Hz (Static gravity to moderate head tilt or linear movement)	•Standing with eyes closed on a soft or unstable floor. •Slow accelerations of a car.
Vestibular semicircular canals	0.5 - 1.0 Hz (Rotational motion of head)	•Head and eye movement control during walking or rapid head turns.
Somatosensation (Proprioception)	> 0.1 Hz (Joint position, muscle length and tension, and cutaneous sensation)	•Control of head position with respect to the torso. •Foot placement during gait; dynamic balance. •Stabilizing cues from touching a wall with the fingers.

Objective

To emulate the human sensory motor system to study its effects on bipedal walking stability.

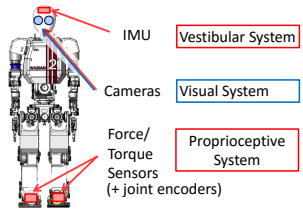


Fig. 2 Proposed sensor system

Proposed Method

•**Sensor fusion system** (Fig. 3)
- **Sensors**: Visual and inertial sensors mounted on the head of the robot (Fig.2) Force/torque sensors on the feet Joint encoders
- **Change weights** of each sensorial input depending on some parameters
- **Select** inputs
- Estimate the robot's Center of Mass states.

•**Head/Neck stabilization system** to compensate for walking perturbations such as vibrations to obtain better sensor data.

•**Walking stabilization system** using the information from the developed sensory system.

•**Loosely coupled sensor fusion system**

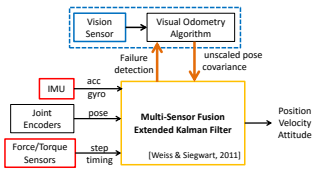


Fig. 3 Proposed sensor fusion system to be used

Sensor Fusion System

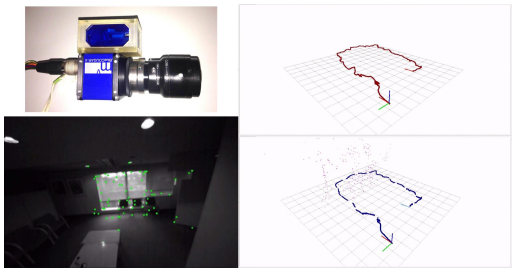


Fig. 4 Sensor setup (top-left), camera image (bottom-left), Multi-Sensor Fusion Extended Kalman Filter (top-right) and visual odometry algorithm only (bottom-right)

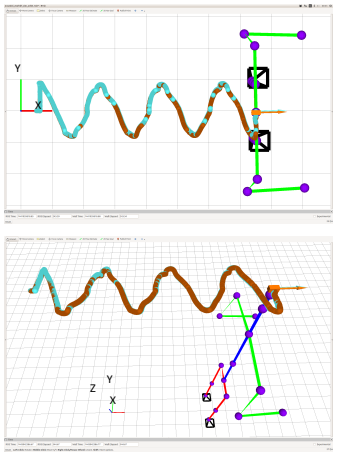


Table 2 RMS error between ground truth and reconstructed trajectory

	Noise σ = 0.001m	Noise σ = 0.005m
X-axis [m]	6.61×10^{-4}	2.14×10^{-3}
Y-axis [m]	1.23×10^{-4}	1.73×10^{-4}
Z-axis [m]	1.58×10^{-3}	1.06×10^{-3}
Roll [rad]	1.16×10^{-4}	1.53×10^{-4}
Pitch [rad]	3.41×10^{-4}	3.83×10^{-4}
Yaw [rad]	1.23×10^{-4}	1.73×10^{-4}

Fig. 5 Simulations of the sensor fusion system for a simple walking experiment input noise with σ = 0.001m. Ground-truth (red) and reconstructed trajectory (blue).

Future Work

Development of a robust online walking pattern generator using this new sensor fusion system, able to cope with different kinds of terrains, obstacles and external perturbations.

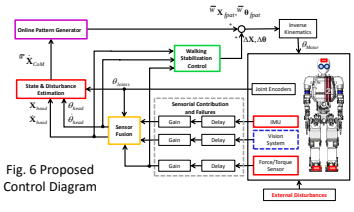


Fig. 6 Proposed Control Diagram

Graduate Program for Embodiment Informatics

GRULL Robotics Group University of La Laguna

C

Safe and reliable navigation in crowded unstructured pedestrian areas

Engineering Applications of Artificial Intelligence 49 (2016) 74–87

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

Safe and reliable navigation in crowded unstructured pedestrian areas

N. Morales*, R. Arnay, J. Toledo, A. Morell, L. Acosta

Departamento de Ingeniería Informática, Universidad de La Laguna, Spain

ARTICLE INFO

Article history:
Received 10 July 2015
Received in revised form
23 October 2015
Accepted 29 November 2015

Keywords:
Autonomous vehicles
Path planning
Unstructured environments
Obstacle avoidance

ABSTRACT

In this paper, the navigation system of the autonomous vehicle prototype Verdino is introduced. Two navigation levels are considered. In the first level, a trajectory is generated from the current position toward a goal that considers two different approaches. In the first, the minimum cost path is obtained using a classical approach (used for regular navigation). The second approach is a little more complex, relying on a set of precomputed primitives representing the motion model of the vehicle, which are used as part of an ARA* algorithm in order to find the best trajectory. This trajectory consists of both forward and backward motion segments for complex maneuvers. In the second level, a local planner is in charge of computing the commands sent to the vehicle in order to follow the trajectory. A set of tentative local trajectories is computed in the Frenét space and scored using several factors, described in this paper. Some results for the two navigation levels are shown at the end of this document. For the global planner, several examples of the maneuvers obtained are shown and certain related factors are quantified and compared. As for the local planner, a study on the influence of the defined weights on the vehicle's final behavior is presented. Also, from these tests several configurations have been chosen and ranked according to two different proposed behaviors. The navigation system shown has been tested both in simulated and in real conditions, and the attached video shows the vehicle's real-world performance.

1. Introduction

Generally speaking, the navigation system of autonomous vehicles is composed of two main levels: the global planner and the local planner. The purpose of the global planner is to obtain a feasible route from the vehicle's current position to a certain goal. In this level it is usual to generate the route by taking into account static information about the navigable areas of the environment. The second level calculates the commands required to control the vehicle in order to follow the global plan while dynamically adapting to changing environmental conditions. In this work two different global planners are used. The first one uses Dijkstra's algorithm to obtain the minimum cost path toward a desired goal. This planner does not take into account the non-holonomic restrictions of the vehicle but is very fast in computing the global plan. The second global planner uses "motion primitives", which are kinematically feasible movements, in order to construct the global plan. This planner produces more complex routes that integrate backward and forward maneuvers, taking into account the non-holonomic restrictions of the platform.

The second level of navigation is what is known as the local planner. In this work we used a local planner that computes a number of tentative trajectories which are evaluated in terms of different criteria. A winning tentative trajectory is selected and used to compute the commands needed to control the vehicle.

During the normal navigation of the vehicle, the non-primitive based global planner is used. The local planner generates tentative trajectories in order follow the global plan. However, there are certain situations in which the local planner cannot produce feasible paths. These situations arise in crowded environments or when the vehicle is not properly aligned with respect to the global plan. In order to solve these situations, a *recovery maneuver* is performed. These maneuvers, different from those generated by the local plan, are intended to re-align the vehicle to the global plan. If, after executing the recovery maneuvers, the vehicle is still unable to produce feasible local paths, the environment is considered to be complex and the primitive-based global plan is used. As noted, this planner constructs the global plan by connecting motion primitives and produces complex maneuvers which are used to guide the vehicle through a crowded area or to perform a parking maneuver.

* Corresponding author.
E-mail addresses: nestor@isaatc.ull.es (N. Morales), rafa@isaatc.ull.es (R. Arnay), jonay@isaatc.ull.es (J. Toledo), amorell@isaatc.ull.es (A. Morell), leo@isaatc.ull.es (L. Acosta).

http://dx.doi.org/10.1016/j.engappai.2015.11.008
0952-1976/

Fig. 1. Verdino prototype.

The method presented in this paper is the local planner of an autonomous robotic prototype called Verdino[1] (Acosta et al., 2012), as shown in Fig. 1. This platform is an electric driverless vehicle designed to transport passengers in a number of different environments, including pedestrian streets or tourist resorts.

The vehicle is a standard golf cart (a TXT-2 from EZ-GO[2]) that has been electronically and mechanically modified so it can be controlled by an on-board computer. It is equipped by default with six 6 V batteries, a speed controller, a 36*Vcc* electric motor, mechanical brakes and steering, and has a maximum speed of 23 Km/h.

In order to determine its location, the vehicle is equipped with an odometer system attached to each wheel that allows it to make relative position estimates. This information is combined in real time with that provided by an Inertial Measurement Unit (IMU) (an *Xsens Mti*[3]), which consists of 3 accelerometers, 3 gyroscopes and 3 magnetometers, in order to yield a three-dimensional orientation of the vehicle; and a centimetric DGPS (a *JAVAD GNSS Triumph-1*[4]) with a horizontal precision below 1 cm and a vertical precision of around 1.5 cm using Differential GPS (DGPS) at a 5 Hz frequency. There are also several Light-Detection And Rangings (LIDARs) units which are used for SLAM. All this information is combined using the method in Perea et al. (2013) in order to accurately localize the vehicle.

These LIDARs are also used to detect obstacles, a requirement of the method described in this paper. One of the advantages of these sensors is that they are very fast and precise. The vehicle is equipped with 5 LIDARs units. Two of them are located at the same plane in the front corners of the vehicle at a height of 50 cm. Each of these covers an angle of 270°. Another one is located at the front of the vehicle, at a height of 20 cm, tilted slightly upwards in order to detect small obstacles or non-traversable areas. Another sensor is at the top front of the vehicle and tilted downward to cover the blind areas of the other sensors. Finally, the last LIDAR unit is situated at the rear of the vehicle and is used for backward maneuvers. The sensor models are the *SICK LMS 100* and *SICK LMS 111*,[5] which offer a maximum detection distance of 20 m, a precision of 10–35 mm, and a maximum angular resolution of 0.25° at a maximum rate of 50 Hz.

Section 2 presents a review of the most relevant methods discussed in literature. In Section 3, the method is introduced, divided into the *Costmap Generation* (Section 3.1), *Global Planner* (Section 3.2) and *Local Planner* (Section 3.3) subsystems. Section 4 shows some of the results and the conclusions are discussed in Section 5.

[1] http://verdino.webs.ull.es
[2] http://www.ezgo.com
[3] http://www.xsens.com
[4] http://www.javad.com
[5] http://www.sick.com

2. Previous work

Autonomous vehicles have been the subject of study for a long time. Proof of this is the extensive literature available in the field, especially since the initial DARPA Challenge (Buehler et al., 2007, 2009). Since then, research efforts related to this problem have increased significantly.

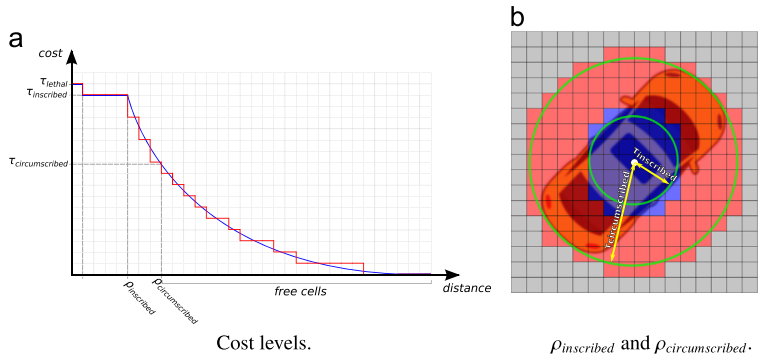
In many of these works, several approaches have been proposed to allow vehicles to follow a certain path from their position toward to a given target. Early works only considered a local planning level, leading to situations in which the robot were trapped at local minima, as they were unable to perform multi-stage maneuvers. To reduce this tendency to reach local minima, new algorithms started to use a combination of global and local information (Brock and Khatib, 1999; Philippsen and Siegwart, 2003).

Since then, researchers have invested considerable effort in improving global planners in order to ensure that the paths generated can be easily followed by the vehicle for which they were intended (Song and Amato, 2001; Likhachev et al., 2005; Knepper and Kelly, 2006). However, this is a very expensive task, especially over large distances. Some of these methods are based on a discrete optimization scheme (Thrun et al., 2006; Montemerlo et al., 2008; Werling et al., 2010; Ferguson et al., 2008). Sampling based approaches are suitable for planning problems in a high dimensional space. The algorithm builds a collision-free path from the initial configuration to the goal. The configuration that defines the position and orientation of the vehicle is sampled. Of all the approaches of this kind, Rapidly-exploring Random Trees (RRT) and its variants are widely used in non-holonomic motion planning applications. RRTs are incrementally built in such a way that the estimated distance from a specific point to the tree is quickly reduced. However, real-time implementations require efficient heuristics for the sampling configuration. Some examples of this type of method are Van Nieuwstadt and Murray (1998), LaValle (2001), Kuwata et al. (2009) and Likhachev and Ferguson (2009).

Based on this idea, two different global planning approaches are considered in this paper, depending on the situation. The first approach is used for regular navigation. At this level, vehicle restrictions are not considered, leaving to the local planner the task of dealing with the limitations given by motion constraints. This approach, based on Dijkstra, is used for long planning.

The other, required for maneuvers like parking or obstacle avoidance in complex circumstances, uses vehicle motion primitives. This planner relies on an improvement to the method in Likhachev and Ferguson (2009), which employs a multi-resolution lattice search space to reduce the complexity of the global search so as to generate bounded suboptimal solutions. Our method is able to improve results by modifying the way in which weights are considered based on the characteristics of Verdino platform.

At a different level, it is also possible to find many methods intended to act as intermediary between the global planner and the vehicle itself. These methods compute a finite set of trajectories based on a parametric model, usually polynomial functions of a certain order. The problem that arises is that – even though the space of possible solutions is reduced – having efficient planning can introduce suboptimality, causing overshoots or stationary offsets in curves. Some other methods, part of the discrete optimization approaches, are based on transforming the configuration space through the Frenét space. In Werling et al. (2010), long-term objectives, such as maintaining speed, following and stopping, are realized. This is done through optimal control strategies within the Frenét frame of the street. In Thrun et al. (2006),



Cost levels. $\rho_{inscribed}$ and $\rho_{circumscribed}$.

Fig. 2. Costmap computation concepts. (a) Cost levels. (b) $\rho_{inscribed}$ and $\rho_{circumscribed}$.

lateral offset is defined as the perpendicular to an established base trajectory. This allows the vehicle to drive along the road parallel to this trajectory. In order to select the optimal path, the cost function takes into account passing over obstacles and the distance with respect to the center of the current road. In Chu et al. (2012), a set of candidate paths is also generated whose endpoints are at fixed positions at different offsets with respect to the base frame. This base frame is not set in the center of the road, however, since it could be dangerous when computing the costs in certain scenarios. Instead, the authors use a security cost for each candidate path. Path security is computed by blurring the binary data of the obstacles. They also take into account certain criteria, such as the smoothness cost or the path consistency between iterations.

3. Method

The proposed system consists of several subsystems that operate separately as stand-alone processes depending one into another. These subsystems are *Costmap Generation*, *Global Planner* and *Local Planner*. The first is in charge of computing the costmap that will be used by the other two methods in order to compute the trajectories, considering the safety of the different possibilities (taking into account the obstacles in the environment as well as the estimates of the expected changes in the near future); the second is used to compute a trajectory that allows the vehicle to travel between the current position and the goal in the unstructured map. The third provides the system with the mechanisms needed to follow said trajectory by computing the commands required by the low-level controller to move the prototype, as well as a set of recovery behaviors that allow the vehicle to return to normal navigation when unexpected situations arise.

3.1. Costmap generation

The costmap holds information about occupied/free areas in the map by way of an occupancy grid. It uses sensor data and information from the static map to store and update information about obstacles in the world, which are marked on the map (or cleared, if they are no longer there). Costmap computation is supported on a layered costmap (Lu et al., 2014), which is used to integrate the different information sources into a single-monolitic costmap. At each layer, information about occupied/free areas in the vehicle's surroundings is maintained in the form of an occupancy grid using the different observation sources as input. Using this information, both dynamic and static obstacles are marked on the map. The layers used are described in Section 3.1.1. At each layer, the costmap is represented as follows:

Each cell in the map can have 255 different cost values:

- A value of 255 means that there is no information about this specific cell on the map.
- 254 means that a sensor has marked this specific cell as occupied. This is considered as a lethal cell, meaning the vehicle should never enter it.
- The rest of the cells are considered as free, but with different cost levels depending on an inflation method relative to the size of the vehicle and its distance to the obstacle.

The cost value of free cells decreases with the distance to the nearest occupied cell, in keeping with the expression:

$$\mathcal{C}(i,j) = \exp(-1.0 \cdot \alpha \cdot (\| c_{ij} - \overrightarrow{o} \| - \rho_{inscribed})) \cdot 253 \quad (1)$$

In this expression, α is a scaling factor that increases or decreases the decay rate of the obstacle's cost. $\| c_{ij} - \overrightarrow{o} \|$ is the distance between cell $c_{ij} \in \mathcal{C}$ (where $\mathcal{C}$ is the set of cells in the costmap) and the obstacle. Finally, $\rho_{inscribed}$ is the inscribed radius, which is the inner circle of the limits of the vehicle. This radius is depicted in Fig. 2.

Even though all of them are free cells, four different distance thresholds are defined in order to assign different danger levels in the map, as depicted in Fig. 2:

- τ_{lethal}: There is an obstacle in this cell posing a collision hazard. It is represented by cost level 254.
- $\tau_{inscribed}$: Cell distance to the nearest obstacle is below $\rho_{inscribed}$. If the center of the vehicle is in this cell, it is also a collision hazard, so areas below this distance threshold should be avoided. The cost level is always 253.
- $\tau_{circumscribed}$: If the vehicle's center is in this cell, it will very likely collide with an obstacle, depending on its orientation. A cell with a distance to an obstacle below this threshold should be avoided, though the possibility remains of not colliding with an obstacle despite being inside this cell.
- The remaining cells are assumed to be safe (except for those with an *unknown* cost and whose occupancy status is unknown and are thus regarded as lethal).

In the approach presented, only those paths passing through cells with a cost below $\tau_{circumscribed}$ are considered. This cost is obtained using Eq. (1) and other cost factors that will be explained later. Paths passing through the cells over this threshold will be truncated at the last safe point.

To compute the costmap and the costs associated with each cell, the `ROS` plugin `costmap_2d`[6] was used, which implements some of the functionalities described in this section.

[6] http://wiki.ros.org/costmap_2d

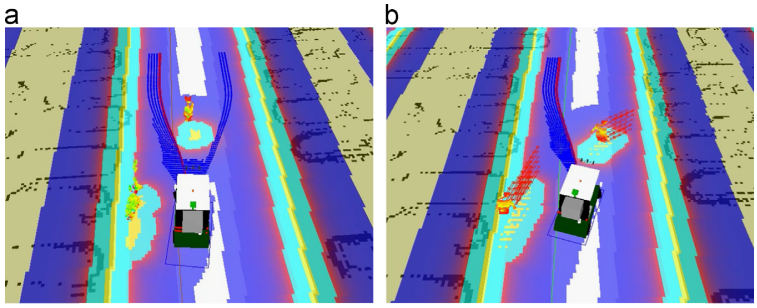


Fig. 3. Comparison of the vehicle's behavior with (b) and without (a) the obstacle motion estimate. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

3.1.1. Layered costmap

Four different layers are being considered:

- A first layer represents the obstacles in a static map previously captured using the method described in Grisetti et al. (2007). This map, computed using LIDARs as the input, represents the static obstacles in the whole area in which the vehicle will move. This map is modified by hand in order to define those areas the vehicle should not enter, even if it is safe to do so, like for example bike lanes or parking lots. This layer is the only one used by the non-primitive based global planner (see Section 3.2.2), since non-static obstacles are not being considered for non-primitive based trajectory generation (non-static meaning those obstacles that are not already included in the map). These are supposed to be avoided at the local planning level.
- A second layer, also based on a static map, is included. For optimization reasons, in this and following layers, the costmap is not computed for the whole map at each iteration. Instead, only the cells in a 40×40 m area centered around the vehicle's current position are updated. The goal is not to update the whole map, since these layers are only used for local planning or local maneuvering. Static obstacles are also included for local planning, since it is undesirable for the vehicle to pass along restricted areas while avoiding obstacles. This allows the vehicle to know which areas are forbidden at the local planning level.
- A third layer is used to represent the dynamic obstacles detected by the various sensors (including the LIDARs, Velodyne and stereo cameras). Using this input, the ground is detected and removed, extracting just the vertical obstacles with which the vehicle could collide. The parameters in this layer are chosen so that obstacle inflation is higher than that computed for the second layer. This gives more priority to the obstacles being detected in real time over those on the static map.
- The last layer provides an estimate of the future motion of dynamic obstacles. To do this, input point clouds are segmented using a voxel grid in order to reduce dimensionality. The world surrounding the vehicle is divided into a discrete number of voxels of equal size. For each voxel, an occupancy probability is assigned, based on the number of points from the input point cloud in its neighborhood. Using this probability, valid voxels (with a higher occupancy probability) are distinguished from the noisy ones (with a smaller probability).

During this process, a set of particles is assigned dynamically to each voxel. These particles have a dual function: denoting hypotheses (as happens with classical particle filters), and being used as the building blocks of the world model. At each frame, the set of particles obtained in the previous frame (each of these with a certain pose (x, y, z) and speed (vx, vy, vz)) is evolved using their motion model and assigned to the corresponding new voxel, based on the time elapsed between frames and the ego-motion. Then, the particles are re-weighted and resampled. Particles that survived to the resampling process are used to construct the objects that model the environment. This is done by joining all the voxels that share a similar orientation and speed based on their related particles. This method, beyond the scope of this paper, provides the proposed planning scheme with an estimate of the future motion of the obstacles.

Using this motion information, fake obstacles are generated from the original ones based on their initial position and speed. Obstacles are propagated along their estimated motion vector based on a user-defined evolution time t_{evol}. These fake obstacles are included in the costmap and inflated accordingly. The inflation parameters for this layer are chosen so the vehicle can move closer to fake obstacles than to the currently detected obstacles (in the previous layer).

All these layers are combined into a single costmap, as described in Lu et al. (2014).

Fig. 3 shows a comparison of the vehicle's behavior using the costmap with and without the obstacle motion estimate. The vehicle is represented together with the set of tentative paths (in blue) and the winning path (in red). In Section 3.3, the way in which these paths are computed is explained.

Fig. 3 a shows a person being detected by the on board sensors. This person is moving from the right to the left side of the road. However, since no motion information is being used, the planner is avoiding this obstacle from the left side. The problem is that it is crossing the obstacle's trajectory, leading to a future collision (requiring the car to stop). In Fig. 3b another example is shown. In this case, the obstacle is moving from left to right and forward in the direction of the vehicle (there is another obstacle at the side of the prototype, but it is not affecting it). In this image, the red arrows indicate the motion detected. Since this motion is included in the costmap using the false obstacles, the vehicle attempts to avoid the obstacle by driving down the side on which it is not crossing its trajectory.

3.2. Global planner

There are two global planners in use in the Verdino prototype: the primitive-based planner and the non-primitive global planner. These planners are intended to obtain a feasible path going from the vehicle's current position to a specified goal.

Although both methods are included in this section, their aim in the system is completely different. The *Non primitive-based global planner* (Section 3.2.2) is used for regular navigation, while the *Primitive-based global planner* (Section 3.2.1) is used to recover the vehicle from situations in which there is a persistent obstacle

in the way, or if the vehicle is performing some complex maneuver, like parking.

3.2.1. Primitive-based global planner

The primitive-based global planner constructs a path from the vehicle's position to a desired goal. The path is generated by combining "motion primitives", which are short, kinematically feasible motions. These motion primitives are generated using a kinematic model of the vehicle (Espelosín et al., 2013) in order to comply with the vehicle's curvature restrictions.

These primitives are computed by considering a set of predefined orientations. For each orientation, the model is evolved until it reaches one of the predefined orientations, at different speeds. This process is done both forward and backward. This process yields a set of small trajectories that fulfill vehicle restrictions, which will be used as the building blocks for the planner.

Having calculated these trajectories, an ARA* algorithm (Likhachev et al., 2003) is used to search for a feasible path. At each node expansion, a new x y and θ position is explored until the best path is found or the exploration time finishes (in which case the best path found so far is used). During this search, the cost of backward primitives is set higher than the cost of forward ones to prevent the vehicle from using backward paths as much as possible without lowering performance. The original search algorithm described in Likhachev and Ferguson (2009) has also been improved by adding a new cost that penalizes the concatenation of forward and backward primitives. This is done with the intention of planning more natural paths, even if they are long, rather than maneuvering excessively in a short space to achieve a correct orientation and then going straight to the goal. For an illustration of the differences in the solutions obtained before and after the addition of these modifications, see Fig. 11.

Once a path is constructed, those points are identified where the direction of motion changes between going forward to backward and vice versa. Using these points, the global plan is subdivided into paths with a unique direction of motion. The local planner is then fed with these sub-plans sequentially in order to make the vehicle follow them properly, stopping at the final points of each sub-path and changing direction until the goal is reached.

3.2.2. Non primitive-based global planner

The non-primitive based global planner computes the minimum cost path from the vehicle's position to the goal using Dijkstra's algorithm. The assumption here is that the vehicle is holonomic (i.e. it can perform turn-in-place movements).[7] Obviously, the Verdino prototype, being an Ackermann vehicle, is unable to perform such movements; however, given the speed of the search algorithm to obtain the global plan, this planner is being used as a rough estimate of the route that the vehicle is going to follow. The static obstacles in the costmap are then over-inflated in order to make the planner construct smooth paths that can be followed by an Ackermann vehicle. This effect is especially important in sharp turns, see Fig. 6.

Since in this case the routes generated are not constructed considering the non-holonomic restrictions of our platform, the initial angle between the vehicle's orientation and the orientation of the global plan will often be larger than the maximum angle required by the local planner to generate feasible paths. This is why the non primitive-based planner is used in combination with a local planner state machine that takes into account this circumstance and reorients the vehicle properly before using the *Frenét*-based local planner, see Section 3.3.5.

[7] http://wiki.ros.org/navfn

3.3. Local planner

Once the global path is defined, a method is needed for computing the steering and speed commands so as to control the vehicle along this path. This method should also be able to avoid the obstacles present in the road, and do so safely and efficiently. The approach used to solve this problem is based on the method described in Chu et al. (2012), in combination with some ideas proposed in Thrun et al. (2006), taking into account the characteristics of the *Verdino* prototype.

The basic idea behind local path generation is to define a set of *feasible paths* and choose the best option in terms of their cost. The winning path defines the steering and speed commands that the vehicle will use. Having options among local paths is useful for overcoming the presence of unforeseen obstacles in the road.

The current Euclidean coordinate system is transformed into a new system based on the *Frenét* space. To compute this space, the global path is considered as the base frame of a curvilinear coordinate system. The feasible local paths are defined in terms of this base frame as follows:

- The nearest point (where the distance is computed perpendicular to the global path) to the main trajectory will be the origin of the curvilinear coordinate system.
- The horizontal axis will be represented by the distance over the global path, along its direction.
- The vertical axis is represented by the vector that is perpendicular to the origin and points left of the path direction.

In this schema, trajectories can be easily computed in the curvilinear space (that is, maneuvering information is generated). These are then transformed to the original Euclidean space, in which the obstacles' information is added by assigning costs to each path.

Based on this idea, the method can be divided into five stages, as shown in Fig. 4.

1. *Generation of the costmap*: Using the information generated by the sensors or by the methods described in previous chapters, the system constructs a costmap in which costs are related to the distance to obstacles.
2. *Base frame construction*: Based on the global path constructed in the previous section (see Section 3.3.1), the base frame of the curvilinear coordinate system is generated.
3. *Candidate path generation*: Candidate paths are generated onto the curvilinear space. Then, they are transformed to the Euclidean space.
4. *Selection of the winning path*: Costs for all the paths are assigned, and the one with the lowest value is selected.
5. *Computation of vehicle commands*: Vehicle speed and steering angles are computed based on the characteristics of the winning path.

3.3.1. Base frame construction

In this stage, the base frame of the curvilinear coordinate system is defined that enables the algorithm to compute the trajectories in this space as if the global plan were a rectilinear trajectory. At this point, the potential presence of obstacles or the restrictions associated with the vehicle's motion model are not considered. This stage is limited to the generation of trajectories. The geometric relationship between the path in Euclidean and curvilinear coordinates is shown in Fig. 5.

The origin of the base frame coordinate system is the point closest to the vehicle's position in the global plan.

The base frame's arc length (s, in the right image) is obtained as the distance from each point along the global plan (shown as a

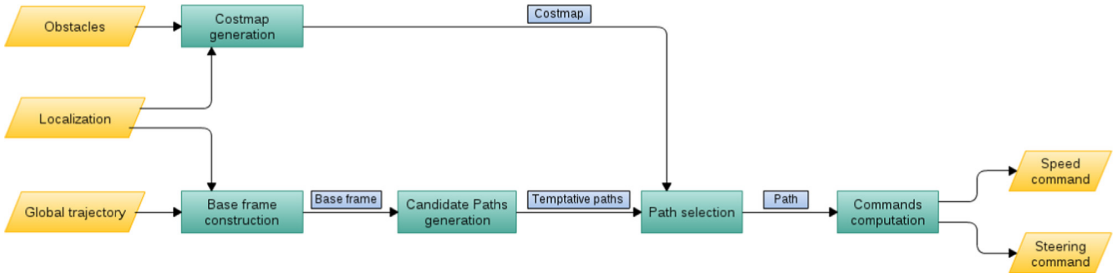
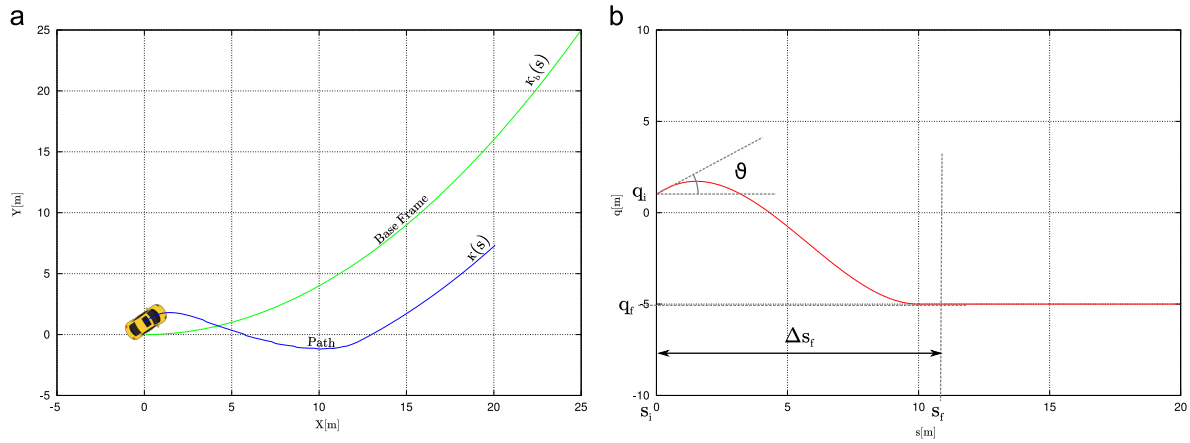


Fig. 4. Pipeline of the method described on this section.



Euclidean space. Curvilinear space.

Fig. 5. Conversion of a trajectory between the Cartesian and Frenét spaces. (a) Euclidean space. (b) Curvilinear space. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

green line) to the origin. This distance is represented in the *x-axis* of the curvilinear system. The *y-axis*, q, represents the perpendicular lateral distance with respect to the path. Values on the left side are positive and those on the right are negative.

To compute the transformation between the Euclidean and curvilinear coordinate systems, the path curvature κ is needed. This value is computed as follows (Chu et al., 2012; Werling et al., 2010; Barfoot and Clark, 2004):

$$\kappa = \frac{S}{Q} \cdot \left(\kappa_b \cdot \frac{(1 - q \cdot \kappa_b) \cdot (\partial^2 q / \partial s^2) + \kappa_b \cdot (\partial q / \partial s)^2}{Q^2} \right), \tag{2}$$

where

$$\begin{cases} S = sign(1 - q \cdot \kappa_b) \\ Q = \sqrt{\left(\frac{\partial q}{\partial s}\right)^2 + (1 - q \cdot \kappa_b)^2} \end{cases} \tag{3}$$

A generated path will be rejected if $q > 1/\kappa_b$. In this case, the path curvature and direction are opposite those of the base frame. The path violates the non-holonomic condition of the vehicle's movement, so the vehicle enters a recovery state, as described in Section 3.3.5.

Only paths with a lateral offset q equal to or smaller than the curvature radius of the base frame $1/\kappa_b$ are accepted. If $q = 1/\kappa_b$, that means that the path passes through the center of curvature of the base frame. Also, the maximum curvature a path can have and remain feasible for the vehicle is limited by the maximum steering angle. If this restriction is violated, the corresponding path is rejected. The curvature is directly related to the movement of the vehicle, which can be described using several models. A simplified version (Barfoot and Clark, 2004), which ignores the related physical effects (like inertia or mass), is:

$$\begin{cases} \dot{x} = |\vec{v}| \cdot \cos(\theta) \\ \dot{y} = |\vec{v}| \cdot \sin(\theta) \\ \dot{\theta} = |\vec{v}| \cdot \kappa \end{cases} \tag{4}$$

Since these physical effects do not affect the geometric shape of the path, they can be ignored in the path generation step. At this point, other physical restrictions, like the maximum curvature that the vehicle is able to follow or the maximum speed, are not considered. In Eq. (4), $[\dot{x}\ \dot{y}\ \dot{\theta}]^T$ are the estimated position and orientation of the vehicle and $|\vec{v}|$ is the speed. This simplified model assumes that the vehicle only has two degrees of freedom, represented by the speed $\vec{v}$ and κ. Since the only concern at this point is the geometric generation of the path, the vehicle's speed $\vec{v}$ can be removed from the model. This is done by expressing the movement of the vehicle in terms of the distance traveled. Thus, the following relationship is established (Barfoot and Clark, 2004):

$$|\vec{v}| = S \cdot Q \cdot \frac{\partial s}{\partial t} \tag{5}$$

If the speed of the vehicle is substituted into the model described in Eq. (4), the differential equation of the movement can

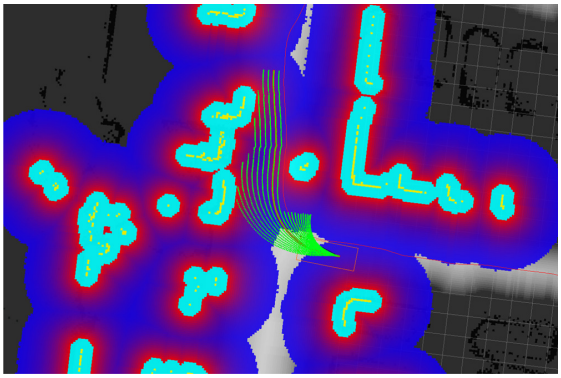


Fig. 6. Paths truncation example. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

be represented in terms of the base frame's arc length s:

$$\begin{cases} \frac{\partial x}{\partial s} = Q \cdot \cos(\theta) \\ \frac{\partial y}{\partial s} = Q \cdot \sin(\theta) \\ \frac{\partial \theta}{\partial s} = Q \cdot \kappa \end{cases} \quad (6)$$

3.3.2. Candidate paths

As we have seen, the path is generated in the curvilinear space without considering the obstacles in the environment. These will be taken into account later, once the tentative trajectories are transformed to the Euclidean space.

Maneuvering path generation. The curvature of the generated paths is defined by the lateral offset q with respect to the base frame. The first- and second-order derivatives of q are needed to compute κ (see Eqs. (2) and (3)), so a function dependent on the lateral offset is needed to compute a smooth lateral change.

q can be defined by a sequence of a cubic polynomial and a set of constants (Barfoot and Clark, 2004):

$$q(s) = \begin{cases} q_i & \text{if } s \le s_i \\ a \cdot \Delta s^3 + b \cdot \Delta s^2 + c \cdot \Delta s + q_i & \text{if } s_i < s < s_f \\ q_f & \text{if } s_f \le s \end{cases} \quad (7)$$

$$\frac{\partial q}{\partial s}(s) = \begin{cases} 3 \cdot a \cdot \Delta s^2 + 2 \cdot b \cdot \Delta s + c & \text{if } s_i \le s < s_f \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

$$\frac{\partial^2 q}{\partial s^2}(s) = \begin{cases} 6 \cdot a \cdot \Delta s + 2 \cdot b & \text{if } s_i \le s < s_f \\ 0 & \text{otherwise} \end{cases}, \quad (9)$$

where $\Delta s = s - s_i$.

In Fig. 5b, the components involved in this process are depicted.

- The initial length s_i is zero, due to the process carried out at the beginning of each iteration (Section 3.3.1). The lateral offset q_i is also known, which is the lateral offset with respect to the global path's origin.
- The angle θ defines the difference between the vehicle's heading angle and the tangent angle of the base frame at the current position. If $\theta > 40^\circ$, the vehicle enters a recovery state, described in Section 3.3.5. Once the vehicle is correctly headed toward the path, this recovery state ends.
- s_f is a parameter that controls the longitudinal distance needed to reach offset q_f. This distance should be dependent on the speed; however, since the prototype's top speed is limited, Δs_f can be regarded as the distance needed to go from q_i to the greatest q_f at the top speed.
- The different q_f are computed separately for each path based on the parameters defined by the user. s_f is also a free parameter.

The parameters s_i, q_i, and s_f are shared by all candidate paths. Modifying the value of q_f yields different tentative trajectories, so the only difference between them is the lateral offset at the end of the paths. This lateral offset gives flexibility in how obstacles are avoided. To do this, the width of the road is divided into as many segments as the number of desired paths. q_f will be the perpendicular distance between the base frame and the corresponding road width division. Using this technique, the desired number of paths is created. The set of paths should cover the entire width of the road and ensure that the vehicle is able to avoid the obstacles, if possible.

In the tests conducted we considered a maximum width of 4 m (a little wider than the roads in our testing area) and a horizon of 10 m in the direction of the global path. This horizon yields a prediction time of 2 s at the vehicle's maximum speed. A total of 21 paths were evaluated, with a distance between them (Δq_f) of 20 cm.

Candidate path generation. Once the paths are computed in the curvilinear coordinate system, they are transformed to the Euclidean space where their associated costs are evaluated.

With the paths in Euclidean coordinates, the maximum distance they can reach individually (if obstacles are considered) can be calculated. To do so, the cells c_{ij} in the costmap associated with the points of the trajectory are checked. If this cost exceeds the value associated with the threshold $\tau_{circumscribed}$, the path is truncated at this point, as shown in Fig. 6, where the generated paths are shown using a colored costmap representation where blue means a low value, while red is used for the higher costs. Yellow and cyan correspond to cells in lethal and inscribed cells, respectively. The points iterated before reaching this cost are kept, although it is very unlikely for this path to be the winner, since the occlusion cost will be maximum and the length of the path shortened (see Section 3.3.3). As shown, when a path collides with an obstacle, it is not completely removed. The reason is that there are certain situations in which the maximum length cannot be reached with any path. However, it is still desirable to approach the maximum reachable point slowly in the hope that the obstacles that are blocking the way will disappear over the next few iterations. This is a typical situation in crowded areas with many pedestrians: the path is blocked, but when pedestrians see an approaching vehicle they move out of the way. However, if the vehicle reaches a point in which it cannot move for a long time, the recovery behavior is triggered. The problem with this strategy is that one of the colliding paths could win even if there is a path that can go through a clear area. In order to avoid this, a schema based on a weighted cost function has been implemented which permits a smart selection of the winning path, as explained in the next section.

3.3.3. Winning path

The winning path is selected by using a linear combination $J[i]$ of weighted cost functions involving the following parameters: occlusion, length, distance to the global path, curvature and consistency of the path. $J[i]$ is evaluated as follows:

$$J[i] = \omega_o C_o[i] + \omega_l C_l[i] + \omega_d C_d[i] + \omega_\kappa C_\kappa[i] + \omega_c C_c[i] \quad (10)$$

Here, i is the path index, and C_o, C_l, C_d, C_κ and C_c are the occlusion, length, distance to the global path, curvature and consistency costs, respectively. Their relative factors ω_k, $k \in \{o, l, d, \kappa, c\}$ are the associated weights used to adjust the influence of each of the costs to the final cost value. All these costs are normalized to

1.0, and

$$\sum_{i \in \{o,l,d,\kappa,c\}} w_i = 1.0, \tag{11}$$

making it easy to determine the proportional influence of each weight. Based on this, the value of $J[i]$ will always be inside the interval [0 ...1]. This normalization is done internally regardless of the weights introduced for the parameters. So if the sum of the parameters is different from one, the parameters are scaled so their sum becomes 1.0.

The following costs will be computed for each candidate path independently from the Euclidean space.

Occlusion. The occlusion cost is related to the safety of the path. This cost estimates the goodness of a path, with the best paths being those passing far enough from the obstacles. This is done by iterating along the path, simulating the car's footprint at each position. The occlusion cost corresponding to trajectory point i will be the maximum cost of each of the cells $c_{ij} \in \mathcal{C}$ under the car's footprint at that position. As a result, the occlusion cost will be

$$C_o = \frac{\max\{c_i\}}{255}, \quad i = 1 \ldots L \tag{12}$$

In this expression, L is the length of the current path being evaluated and $\max\{c_i\}$ is the maximum value of all the costs associated with a point on the path. As noted in Section 3.1, the maximum value for each cost is 255, meaning C_o is divided by this value to normalize it to 1.

Length. This cost represents the length of the current path. By iterating along the points in the path, the distance between them is accumulated and the real distance traveled in Euclidean coordinates is known. The longer a path is the better, as it is assumed that it will traverse an obstacle-free zone. Thus, long paths should produce low cost values. This is done through the expression:

$$C_l = 1 - \frac{\sum_{i=1}^{L} \| p_i - p_{i-1} \|}{q_{f_{max}} + s_f} \tag{13}$$

Here, p_i is a certain point inside the evaluated path and $q_{f_{max}}$ is the maximum value that q_f can have for a certain path. The lengths are normalized to a value that a path will never reach. This cost is subtracted from 1.0 in order to make it comparable to the remaining costs (as noted, lower values are preferred with respect to the higher ones).

Distance to the global path. This cost represent the lateral offset of the vehicle with respect to the global path. Tuning the weight associated with this cost will change the behavior of the vehicle when returning to the global path after avoiding an obstacle. It is computed as follows:

$$C_d = \frac{\sum_{i=1}^{L} \| p_i - nearest(p_i, g) \|}{L \cdot q_{f_{max}}}, \tag{14}$$

where $nearest(p, g)$ is the nearest point in the global path g to point p. This cost is normalized with respect to the maximum expected offset, $q_{f_{max}}$.

Curvature. This cost is used to assign priority to smoother paths. Let $p(x_i, y_i), i = 1 \cdots L$, be a point along the path. Then,

$$C_\kappa = \max\left\{ \frac{x_i' \cdot y_i'' - x_i'' \cdot y_i'}{(x_i' + y_i')^{3/2}} \right\}, \quad i = 1 \ldots L \tag{15}$$

Consistency. This cost avoids continuous changes in winning paths between iterations. Once the vehicle starts a maneuver, it is preferable to keep the same behavior during the following iterations. This is done through the following expression:

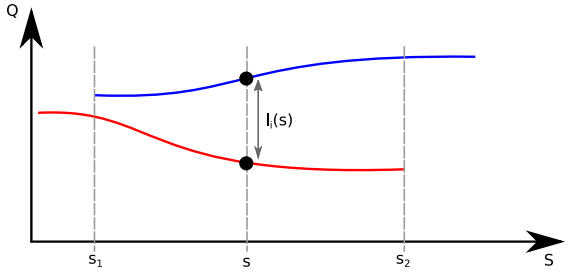


Fig. 7. Representation of how the consistency cost is computed.

$$C_c = \frac{1}{s_2 - s_1} \int_{s_1}^{s_2} l_i \, dS \tag{16}$$

For a better understanding of this equation, consider Fig. 7, where the lateral cost $l_i(s)$ is the distance between the current path and the previous winning path at the same longitudinal position s, and s_1 and s_2 are the first and last positions, respectively, over S for which there are common points in both trajectories.

Selection of the winning path. Once all costs are computed, the expression described in Eq. (10) is applied. In those paths for which it is impossible to advance due to the presence of a nearby obstacle or because the car is incorrectly aligned with the global path (meaning that no valid paths can be generated in this situation), the cost will be negative (invalid path).

Of all the remaining paths, the one with the smallest cost (winning path W) is selected. If for any reason there are no valid paths, a recovery maneuver (see Section 3.3.5) is initiated.

3.3.4. Computing the vehicle commands

The last step required for the vehicle to follow the trajectory output by the global planner is to compute the steering angle and speed commands that will be sent to the vehicle's built-in controller.

- For the *steering angle command*, the ith point in the winning trajectory at which the distance exceeds the user-defined value $d_{steering}$ is used to compute the difference between the current and desired orientations of the vehicle. The steering angle is obtained using a *PID* controller tuned to minimize this difference.
- The *speed command* is computed based on the vehicle's distance to the target and the maximum acceleration allowed. First, the required speed increment is computed:

$$\Delta S = \begin{cases} S_{max} & \text{if } W_l > (d_{safe} + d_{ramp}) \\ (W_l - d_{safe}) * S_{max} / d_{ramp} - S_0 & \text{otherwise} \end{cases} \tag{17}$$

Here, W_l is the length of the winning path. d_{safe} is a user-defined safety distance: if the winning path length is below this distance, the car stops, since that means that the winning path is unable to avoid a nearby obstacle. d_{ramp} is the user-defined distance required to reach the maximum speed S_{max} when the car is completely stopped. S_0 is the current speed. Computing ΔS in this way ensures that the vehicle's response is smooth, avoiding sudden speed increments. Knowing this value, it is possible to compute the speed command S:

$$S = \begin{cases} 0 & \text{if } W_l < d_{safe} \quad \text{or} \quad (S_0 + \Delta S) < 0 \\ \min(S_0 + \Delta S, S_{max}) & \text{if otherwise} \end{cases} \tag{18}$$

In the event of a backward maneuver, $S = -S$.

3.3.5. Recovery behavior

There are two scenarios in which the vehicle executes a recovery maneuver. The first occurs when the vehicle is not

correctly aligned with respect to the global plan and the initial angle is too large to produce feasible local paths that comply with the curvature restrictions of the vehicle. The second case happens when the vehicle is correctly aligned but there are no feasible local paths to follow due to the presence of a nearby obstacle.

The vehicle's recovery behavior involves two main phases. In the first phase, a set of feasible paths is generated using a different schema than that used in Section 3.3. In the second phase, these paths are weighted in order to choose the best option. The vehicle will then try to follow the winning path. The recovery paths are chosen from among four options: two forward paths and two backward paths, moving the steering wheel to the maximum allowed angle to both the left and right sides. The recovery maneuver consists of a sequence of one or more of these paths.

In the second case, the vehicle only goes backward if possible for a short distance in order to gain enough space for the generation of feasible local paths that can avoid the obstacle.

As shown in Fig. 8a, when the angle θ between the vehicle's heading and the orientation of the global plan at the closest point to the vehicle surpasses a certain threshold $\theta_{normalToRecovery}$, the vehicle starts a recovery maneuver. In Fig. 8b and c, the vehicle follows the recovery paths generated in order to align itself with the global plan. When θ is below the threshold $\theta_{recoveryToNormal}$, the vehicle is considered to be properly oriented and valid Frénet local paths can be generated, see Fig. 8d.

If the vehicle is still unable to continue after every recovery maneuver has been executed, a new global plan is generated using the *Primitive-based global planner* (see Section 3.2.1).

3.3.6. Local planner state machine

The local planner uses two different state machines depending on the global planner being used. When using the *primitive-based global planner*, the local planner switches to a state machine that uses four states: *Forward*, *Backward*, *Stopping* and *Stopped*. In this case, the primitives used by the global planner restrict the initial θ to valid angles. The local planner generates Frénet paths both forward and backward depending on the state. The transitions in this state machine can be seen in the pseudo-code below:

```
switch state do
  case FORWARD
    If v_d = =0 then
      state ← STOPPING
    else
      generatePathsForward()
    end if
  case BACKWARD
      if v_d = =0 then
      state←STOPPING
    else
    generatePathsBackward()
  end if
case STOPPED
      if v_d > 0 then
      generatePathsForward()
    else if v_d < 0 then
      generatePathsBackward()
  end if
  If v > v_t1 then
      state←FORWARD
    else if v < −v_t1 then
    state←BACKWARD
    end if
case STOPPING
    reduceVelocity()
  if v < v_t2 then
      state←STOPPED
end if
```

Where

- v is the vehicle's current velocity.
- v_d is the target velocity.

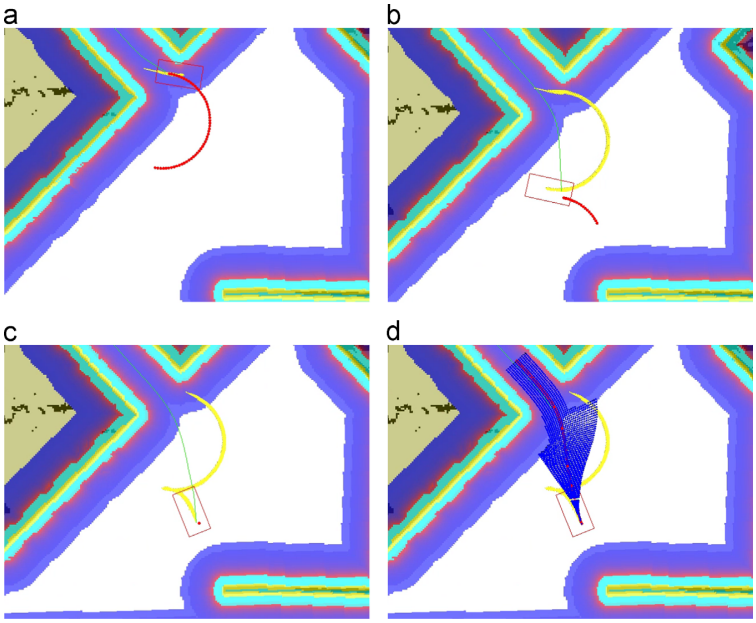


Fig. 8. Recovery maneuver.

- v_{t1} is a velocity threshold used to change from a stopped state to one of the movement (forward or backward) states.
- $v_{t2} \approx 0$ is a velocity threshold which decides the moment in which the state should be changed from *stopping* to *stopped*. It is used to guarantee that the vehicle reaches a sufficiently low speed between changes in the direction of motion.

When the vehicle uses the *non primitive-based global planner*, the local planner state machine enables the vehicle to perform recovery maneuvers. This state machine has the following states: *Forward*, *Stopping*, *Stopped*, *Recovery* and *Avoid*. These states are implemented to change between the normal navigation of the vehicle and the maneuvers needed to recover the global plan orientation or to avoid a permanent obstacle.

For example, as commented in the previous section, if the vehicle is going forward and the angle between its heading and the global plan exceeds a certain threshold, it will perform a recovery maneuver to orient itself properly. In order to do this, the local planner state machine transits from a *Forward* state to a *Stopping* state. When the vehicle's velocity is ≈ 0, it transits to the *Stopped* state, which guarantees that the vehicle is stopped. Once in this state, the vehicle can now initiate the recovery maneuver. The state machine is intended to guarantee that the vehicle stops between changes in the direction of motion and to maintain these transitions gradual enough to be comfortable for the passengers.

Local planner state machine transition pseudo-code for the non primitive-based global planner:

```
switch state do
  case FORWARD
      if θ_ini < θ_normalaIToRec or v_d = = 0 then
    state ← STOPPING
  else
      generatePathsForward()
    else if
case STOPPED
      if θ_ini < θ_normalToRecovery then
    if v_d > 0 then
      generatePathsForward()
    else if v_d < 0 then
      generateAvoidManeuver()
end if
if  v > v_t1 then
      state ← FORWARD
  else if v < −v_t1 then
      state ← AVOID
end if
else if |v_d| > 0 then
      generateRecoveryManeuver()
if |v| > v_t1 then
      state ← RECOVERY
    end if
  end if
case STOPPING
      reduceVelocity()
  if(then v < v_t2)
      state ← STOPPED
  end if
case RECOVERY
    if θ_ini < θ_recoveryToNormal or v_d = = 0 then
      state ← STOPPING
else
    generateRecoveryManeuver()
end if
case AVOID
    if timer < = 0 or v_d = = 0 then
      state ← STOPPING
    else
      generateAvoidManeuver()
  end if
```

Where

- $\theta_{recoveryToNormal}$ is the angular threshold at which the vehicle is considered to be properly reoriented with respect to the global path.
- $\theta_{normalToRecovery}$ is the angular threshold at which the vehicle is considered to be incorrectly oriented with respect to the global path.
- v is the vehicle's current velocity.
- v_d is the target velocity.
- v_{t1} is a velocity threshold used to change from a stopped state to one of the movement (forward or backward) states.
- $v_{t2} \approx 0$ is a velocity threshold for changing from a stopping state to a stopped state. It is used to guarantee that the vehicle reaches a sufficiently low speed between changes in the direction of motion.

The attached video shows an example of the vehicle operating in real conditions. This video demonstrates the good performance and reliability of the vehicle. The local planner and recovery behaviors are able to run at 10 Hz on an *i7-3770K* processor with 16 Gb of *DDR-3 RAM*, *SSD* storage and an NVIDIA GeForce GT 640.

4. Results

In this section, full performance graphs are presented showing the results obtained for the tests performed.

4.1. Experimental Setup

In order to study the behavior of our method, several trajectories were followed while recording a set of measured variables. These variables were *distance to path*, which measures the distance from the center of the vehicle to the closest point in the global path; *footprint cost*, which measures the maximum cost of the cells below the vehicle's footprint at each iteration; *speed*, assuming that faster trajectories are preferred; and the *curvature* of the trajectory followed.

We used a simulator to maintain the exact same conditions for every test. In each test, the vehicle started at the exact same position and traveled toward the exact same goal. Obstacles were always in the same locations, with the only changing values being the input parameters under evaluation. The results were validated by running tests under real conditions using the Verdino platform.

As seen in Section 3.3.3, there are five different parameters that influence the overall cost, which will determine the winning path chosen. Each parameter has an associated weight. A base configuration of $\omega_d = 0.17$, $\omega_o = 0.2$, $\omega_c = 0.02$, $\omega_l = 0.7$, $\omega_k = 0.01$ provided good empirical results. Using this base configuration as a starting point, different weight configurations were obtained by incrementally varying each weight.

Although the sum of these weights is greater than 1.0 ($\omega_d + \omega_o + \omega_c + \omega_l + \omega_k = 1.1$), the normalization described in Eq. (11) is done internally at the software level. In fact, the weight values shown in this section were kept at their non-normalized values to make it easier to see the parameter differences between the various configurations tested, as well as with respect to the base configuration.

Fig. 9 shows how the variation in each individual weight influences the measured variables. To perform these tests, the

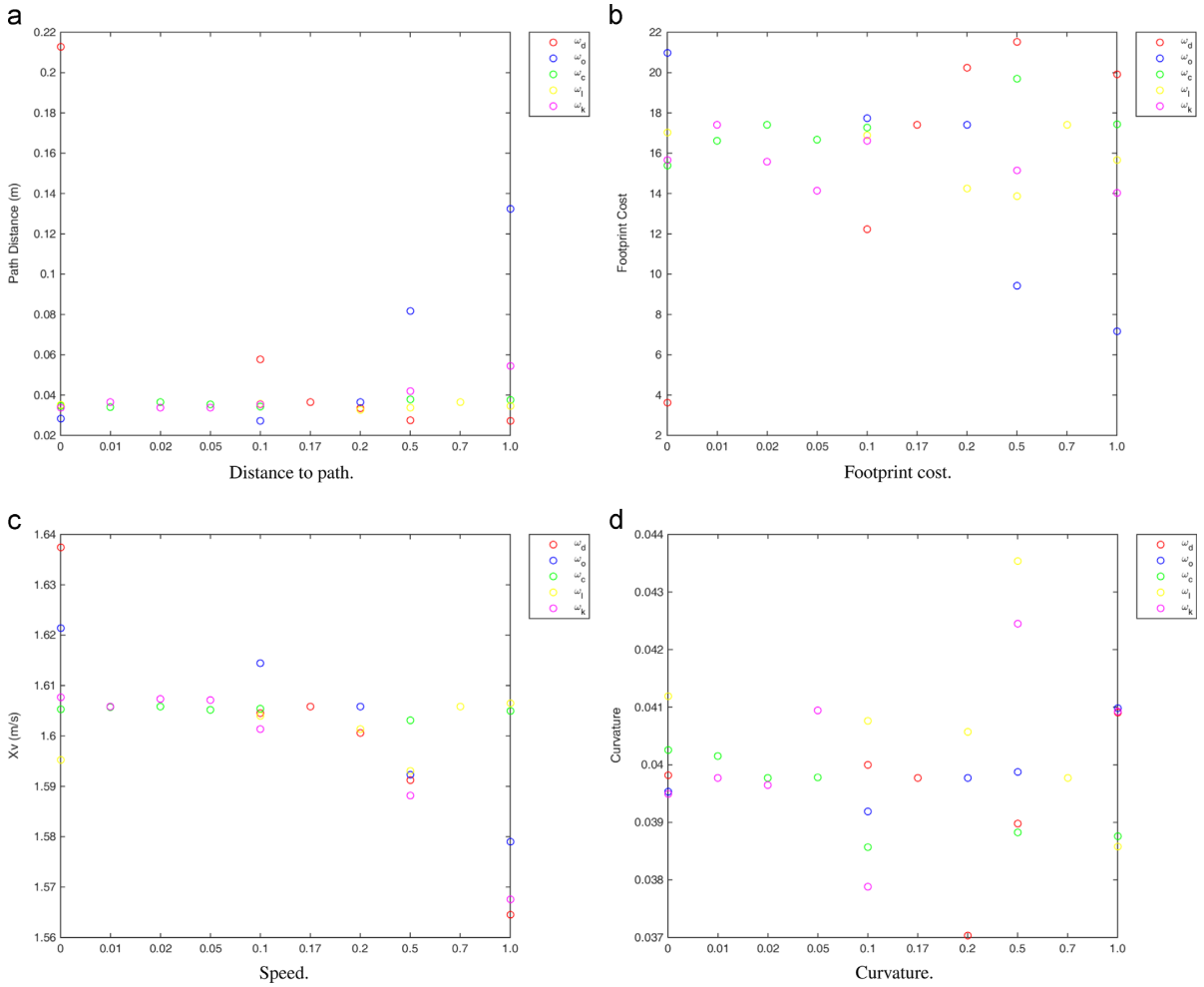


Fig. 9. Results obtained from parameters. (a) Distance to path. (b) Footprint cost. (c) Speed. (d) Curvature.

remaining weights are not modified with respect to the base configuration. Fig. 10 shows a ranking with different cost weight configurations. In Fig. 10a, the evaluation was carried out by taking into account the path distance and the footprint cost. In Fig. 10b, the evaluation takes into account the footprint cost and path distance again, but also the velocity and the curvature of the path followed. These last two measured variables are weighted by 0.5 in order to lessen their influence. Each ranking is built by aggregating the normalized value of the measured variables for each configuration.

Also, in order to compare the response of the maneuvers achieved by the method presented with the original planner, some parking and direction change operations were conducted. Again, the same initial position and goal (and the desired orientations in both cases) were commanded to the vehicle in simulated conditions, with more than ten different maneuvers being tested. Of these, three were selected as the most significant, with the rest yielding comparable results.

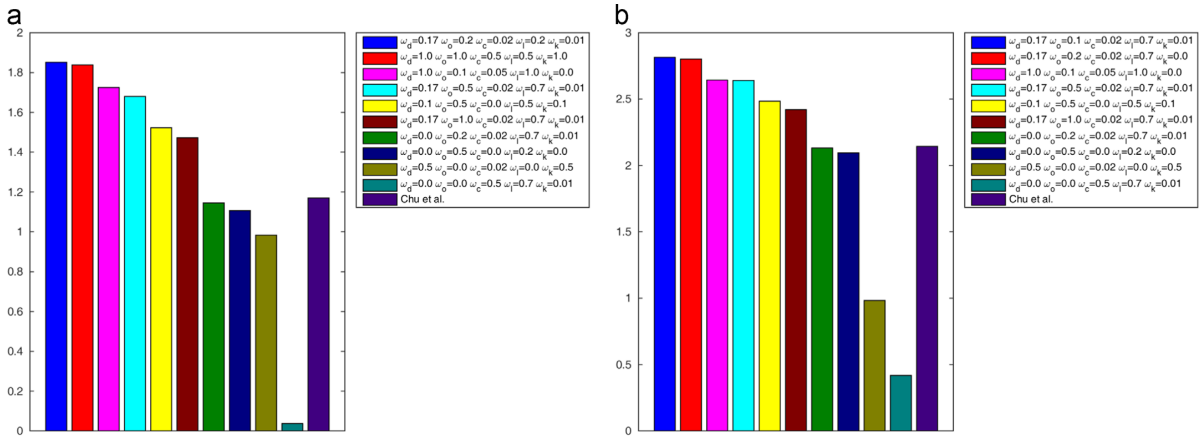
These exact same maneuvers were computed and followed using both our modified method and the original one. We recorded the measurements shown in Fig. 11, which are the positions followed by the vehicle, distance traveled, turns required and the cost of passing through the different positions indicated by the global planner.

4.2. Cost weight determination

As expected, the variables measured are predominantly influenced by the weight of the most relevant cost. However, the influence of the other weights is not as evident. In this sense, see Fig. 9a, where varying the path distance cost makes the path distance proportionately variable to change.

As Fig. 9b shows, the relationship between this cost weight and the footprint cost is inversely proportional. This behavior may be produced due to the global plan traversing relatively high cost areas, mainly in closed curves. When the path distance cost weight is high, the vehicle is forced to stay on the global path, meaning the cost can rise. If the footprint cost weight is high (the vehicle's priority is to avoid obstacles as much as possible), the distance to the global path grows.

Additionally, if the path length cost weight is incremented, the selection of long paths is favored. Since the velocity command is computed proportionately to the path length, higher speeds are obtained when this cost weight is raised, see Fig. 9c. The



Only distance to path and footprint cost are considered. All measured variables are considered.

Fig. 10. Results obtained from rankings. (a) Only distance to path and footprint cost are considered. (b) All measured variables are considered. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

navigation speed also depends on the distance to the obstacles, which is why the footprint cost weight also influences this variable. Moreover, if the curvature cost weight is high, the cost of the winning path in closed curves also increases, forcing the vehicle to slow down.

The curvature of the trajectory is influenced by a combination of the different cost weights, and no particular cost influences it predominantly (see Fig. 9d).

The rankings in Fig. 10a and b show which weight configurations produce the best results based on different criteria. In Fig. 10a the ranking takes into account the footprint cost and the path distance. In this sense, the configurations in the first places of the ranking mainly favor following the global path and avoiding obstacles. In the ranking shown in Fig. 10b the speed achieved by the vehicle and the curvature of the trajectory are also taken into account.

The ranking of the best configuration is shown in purple for the three weights used in the method presented in Chu et al. (2012). For the ranking in Fig. 10a, the best configuration was $\omega_o = 0.2$, $\omega_c = 0.02$ and $\omega_k = 0.01$; and for the ranking in Fig. 10b, this configuration was $\omega_o = 0.1$, $\omega_c = 0.02$ and $\omega_k = 0.01$. As depicted in the charts, the use of the additional cost weights proposed in this work (ω_d and ω_l) yields better results in terms of vehicle behavior.

4.3. Global planner results

In order to measure the performance gain obtained with the modified primitive-based global planner, we compared it with the original in terms of footprint cost, number of changes in the direction of motion and distance traveled. Different maneuvers were tested. The left side of Fig. 11 shows both global paths superimposed on top of a cost map. The red rectangle represents the vehicle's footprint and the red arrow, the goal. On the right side of the same figure is a comparison of the footprint cost of following each global plan.

As the figure shows, the modified primitive-based global planner generates paths with fewer changes in the direction of motion than the original planner. This results in a more comfortable driving experience. These advantages come with no added drawbacks in terms of footprint cost or distance traveled, which are comparable to those obtained with the original primitive-based global planner.

5. Conclusions

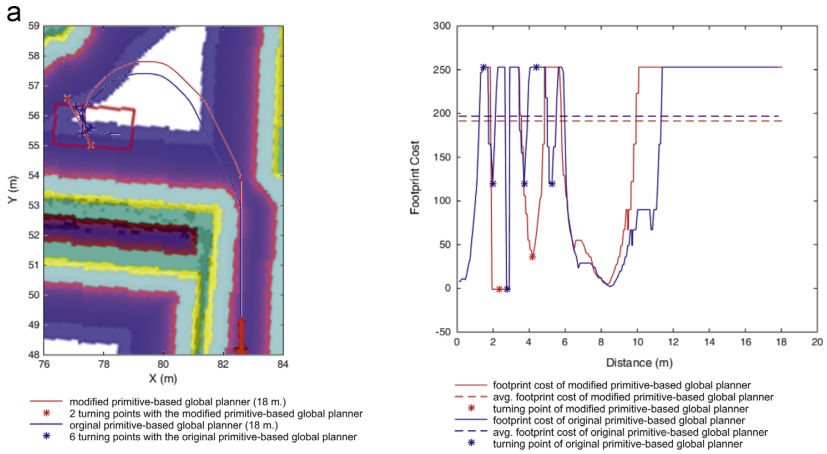
In this paper, a completely operational navigation system is presented that renders our Verdino platform fully autonomous. It is able to navigate in unstructured areas with many dynamic obstacles, such as pedestrian areas. It relies on a multi-modular system consisting of a multilayered costmap, a local planner and two different global planners (switched depending on the situation).

Based on the layered costmap described in Lu et al. (2014), a configurable costmap is implemented. In this costmap, each layer is assigned to an obstacle type, with static obstacles on a costmap, and dynamic obstacles and an estimate of the future motion of obstacles supported by a simplified schema based on voxels and a particle filter. Dynamic obstacles are mainly detected using the on-board LIDARs, as well as the Velodyne.

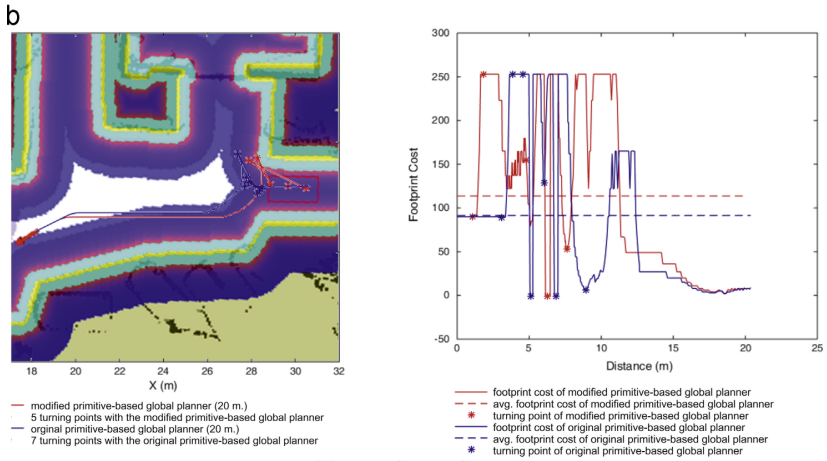
Based on a combination of the ideas described in Chu et al. (2012) and Thrun et al. (2006), adapted to the specificities of our prototype, we developed a local planner for the Verdino autonomous vehicle. This planner is expanded with the addition of new weights, used to evaluate the tentative paths generated, which allow it to select the most reliable trajectory. The results presented show weight configurations that optimize the navigation of the vehicle in terms of safely avoiding obstacles and of its ability to follow the desired global plan. As was shown, the addition of new weights increases the performance of the resulting configurations with respect to the *state of the art* approach described in Chu et al. (2012).

Two different global planners are used in the presented system. One is a non-primitive-based global planner that does not take into account the vehicle's non-holonomic restrictions and is very fast to compute. It is mainly used when the vehicle is navigating in clear areas. The other is a primitive-based global planner that takes into account the vehicle's non-holonomic restrictions by combining motion primitives, which are short, kinematically feasible motions. Some improvements were made to the standard primitive-based global planner in order to obtain simplified paths that avoid the generation of multiple changes in the direction of motion in small areas.

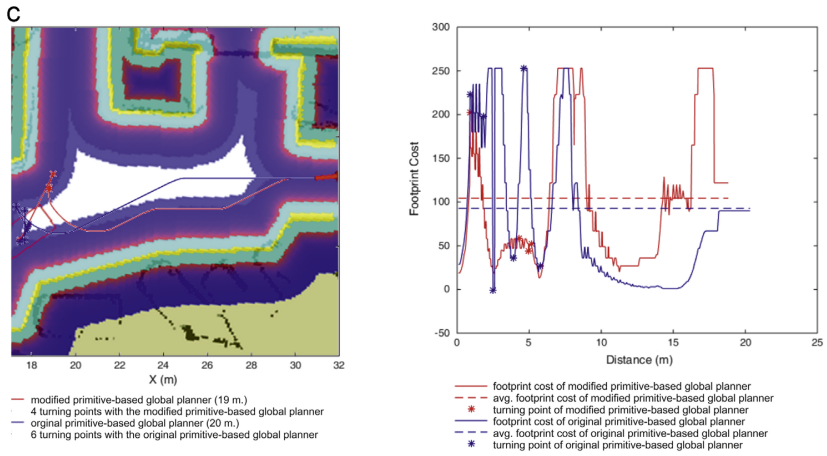
The local planner presented is capable of working in two modes. In the first mode, which relies on the non primitive-based global plan, the vehicle executes recovery maneuvers when there is a static obstacle blocking the road, or when it is not correctly oriented with respect to the global plan. In the second mode, it



Maneuvering test 1.



Maneuvering test 2.

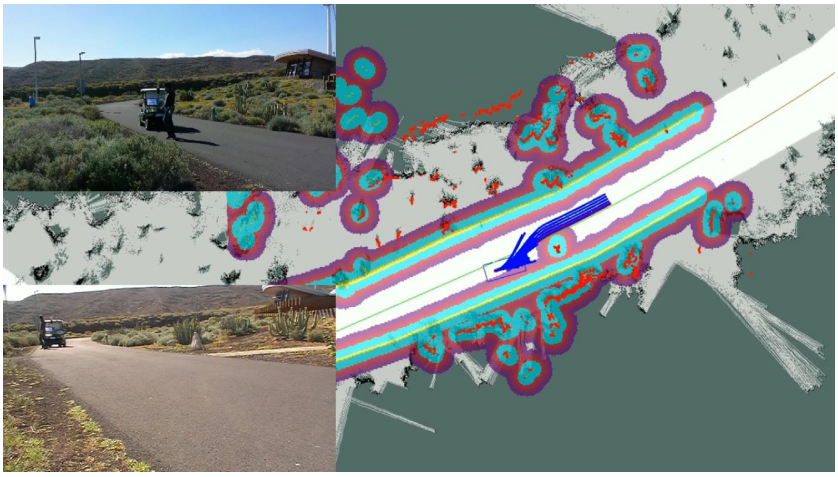


Maneuvering test 3.

Fig. 11. Maneuvering tests and their associated results. (a) Maneuvering test 1. (b) Maneuvering test 2. (c) Maneuvering test 3.

follows the modified primitive-based global plan to perform maneuvers requiring greater precision, like navigating in a crowded area or parking, for example.

The system was tested in real conditions in a wide variety of scenarios and situations, where it demonstrated that it is able to perform in very different conditions while exhibiting both safe and



Video S1. Some examples of the vehicle operating in real conditionsA video clip is available online. Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.engappai.2015.11.008

efficient behavior. This finding is evident both in the results described in this paper, and in the attached video.

Acknowledgments

This work was funded in part by the STIRPE DPI2013-46897-C2-1-R project and by the Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI), with additional funding provided by the FEDER funds (EU).

Appendix A. Supplementary material

The following are the supplementary data to this article:
Video S1

References

Acosta, L., Toledo, J., Arnay, R., Espelosín, J., Morales, N., Perea, D., Moreno, L., 2012. Verdino, prototipo eléctrico de vehículo autoguiado. In: XXXIII Jornadas de Automatica, pp. 729–736.

Barfoot, T., Clark, C., 2004. Motion planning for formations of mobile robots. Robot. Auton. Syst. 46, 65–78.

Brock, O., Khatib, O., 1999. High-speed navigation using the global dynamic window approach. In: Proceedings of the 1999 IEEE International Conference on Robotics and Automation, IEEE, Detroit, MI, USA, 1999, pp. 341–346.

Buehler, M., Iagnemma, K., Singh, S., 2007. The 2005 DARPA Grand Challenge: the Great Robot Race, 1st edition. Springer Publishing Company Incorporated, Heidelberg, Germany.

Buehler, M., Iagnemma, K., Singh, S., 2009. The DARPA Urban Challenge: Autonomous Vehicles in City Traffic. Springer Tracts in Advanced Robotics. Springer, Heidelberg, Germany.

Chu, K., Lee, M., Sunwoo, M., 2012. Local path planning for off-road autonomous driving with avoidance of static obstacles. IEEE Trans. Intell. Transp. Syst. 13, 1599–1616.

Espelosín, J., Acosta, L., Alonso, D., 2013. Path planning approach based on flock dynamics of moving particles. Appl. Soft Comput. 13, 2159–2170.

Ferguson, D., Howard, T.M., Likhachev, M., 2008. Motion planning in urban environments. J. Field Robot. 25, 939–960.

Grisetti, G., Stachniss, C., Burgard, W., 2007. Improved techniques for grid mapping with Rao–Blackwellized particle filters. IEEE Trans. Robot. 23, 34–46.

Knepper, R.A., Kelly, A., 2006. High performance state lattice planning using heuristic look-up tables. In: IROS, pp. 3375–3380.

Kuwata, Y., Karaman, S., Teo, J., Frazzoli, E., How, J., Fiore, G., 2009. Real-time motion planning with applications to autonomous urban driving. IEEE Trans. Control Syst. Technol. 17, 1105–1118.

LaValle, S.M., 2001. Randomized kinodynamic planning. Int. J. Robot. Res. 20, 378–400.

Likhachev, M., Ferguson, D., 2009. Planning long dynamically feasible maneuvers for autonomous vehicles. Int. J. Robot. Res. 28, 933–945.

Likhachev, M., Gordon, G.J., Thrun, S., 2003. ARA*: anytime A* with provable bounds on sub-optimality. In: Advances in Neural Information Processing Systems.

Likhachev, M., Ferguson, D.I., Gordon, G.J., Stentz, A., Thrun, S., 2005. Anytime dynamic A*: an anytime, replanning algorithm. In: The International Conference on Automated Planning and Scheduling, pp. 262–271.

Lu, D.V., Hershberger, D., Smart, W.D., 2014. Layered costmaps for context-sensitive navigation. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), IEEE, Chicago, IL, USA, pp. 709–715.

Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Marcil, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A., Thrun, S., 2008. The Stanford entry in the urban challenge Junior. J. Field Robot. 25, 569–597.

Perea, D., Hernandez-Aceituno, J., Morell, A., Toledo, J., Hamilton, A., Acosta, L., 2013. MCL with sensor fusion based on a weighting mechanism versus a particle generation approach. In: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), IEEE, The Hague, Netherlands, pp. 166–171.

Philippsen, R., Siegwart, R., 2003. Smooth and efficient obstacle avoidance for a tour guide robot, in: Proceedings of The IEEE International Conference on Robotics and Automation (ICRA), IEEE, Taipei, Taiwan.

Song, G., Amato, N.M., 2001. Randomized motion planning for car-like robots with C-PRM. In: Proceedings of the 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, Maui, HI, USA, 2001, pp. 37–42.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., Mahoney, P., 2006. Stanley: the robot that won the DARPA Grand Challenge. J. Field Robot. 23, 661–692.

Van Nieuwstadt, M.J., Murray, R.M., 1998. Real-time trajectory generation for differentially flat systems. Int. J. Robust Nonlinear Control 8, 995–1020.

Werling, M., Ziegler, J., Kammel, S., Thrun, S., 2010. Optimal trajectory generation for dynamic street scenarios in a Frenét Frame. In: 2010 IEEE International Conference on Robotics and Automation, IEEE, Anchorage, AK, USA, pp. 987–993.

D

Safe and reliable path planning for the autonomous vehicle verdino

Safe and Reliable Path Planning for the Autonomous Vehicle Verdino

Rafael Arnay, Néstor Morales, Antonio Morell, Javier Hernández-Aceituno, Daniel Perea Ström, Jonay Toledo, Alberto Hamilton, Javier Sánchez-Medina, and Leopoldo Acosta

Universidad de La Laguna, Canary Islands, Spain Email: rafa@isaatc.ull.es

***Abstract*—This paper introduces a local planner which computes a set of commands, allowing an autonomous vehicle to follow a given trajectory. To do so, the platform relies on a localization system, a map and a cost map which represents the obstacles in the environment. The presented method computes a set of tentative trajectories, using a schema based on a Frenét frame obtained from the global planner. These trajectories are then scored using a linear combination of weighted cost functions. In the presented approach, new weights are introduced in order to satisfy the specificities of our autonomous platform, Verdino. A study on the influence of the defined weights in the final behavior of the vehicle is introduced. From these tests, several configurations have been chosen and ranked according to two different proposed behaviors. The method has been tested both in simulation and in real conditions.**

Digital Object Identifier 10.1109/MITS.2015.2504393
Date of publication: 20 April 2016

1939-1390/16©2016IEEE

I. Introduction

Generally speaking, the navigation system of autonomous vehicles is composed of two main levels: the global planner and the local planner. The first level consists of the generation of a feasible route from the current position of the vehicle to a desired goal. The second level computes the necessary commands to control the vehicle in order to follow the global plan, while dynamically adapting to the changing environment conditions.

The problem we want to solve is safely following a predefined route while avoiding dynamic obstacles. This problem is not trivial, since several factors have to be taken into account. For starters, the safety of pedestrians is crucial: the vehicle has to navigate close to the desired route while keeping a safe distance to the obstacles. Secondly, the navigation has to be comfortable from the passengers point of view: when following the global path and avoiding obstacles, the performed maneuvers have to prevent both abrupt changes of linear and angular speed and high curvature trajectories.

IMAGE COURTESY OF HTTP://VERDINO.WEBS.ULL.ES

The method presented in this paper is the local planner of an autonomous robotic prototype called Verdino[1] [1], shown in Fig. 1. This electric vehicle was designed for people transportation in pedestrian environments.

When a new destination is selected, the global planner builds a feasible path from the current vehicle position to the desired goal. The vehicle then follows this path by using the method presented in this paper. In this method, the euclidean space surrounding the vehicle is transformed to the Frenét space, using the computed global trajectory as basis. Then, a set of tentative paths is computed, considering the following restrictions: all paths should start at the position and orientation of the vehicle, and should end parallel to the global trajectory, at a parameterized lateral distance from it. This way paths are computed using just geometrical information, making the problem simpler by not needing to define a kinematic model of the vehicle to generate the trajectories in the euclidean space. Once trajectories are computed, they are transformed back to the euclidean space, in which they are scored based on different variables like, for example, their curvature or their distance to obstacles. Impossible paths (those which can not be followed by the vehicle) are removed. Using these scores, a winner path is selected and used for the computation of the next speed and steering commands.

This vehicle used as a testing platform is a standard golf car, which has been electronically and mechanically modified so it can be controlled by an on-board computer. It is equipped by default with six 6 V batteries, a speed controller, a 36 Vcc electrical motor, mechanical brakes and steering, and has a maximum speed between 19 and 23 Km/h.

In order to localize itself, the vehicle is equipped with an odometry system attached to each wheel, which allows making relative position estimations. This information is combined with the information provided by an Inertial Measurement Unit (IMU) and a centimetric DGPS. Several Light-Detection And Ranging (LIDAR) sensors are used both for SLAM (Simultaneous Localization and Mapping) and obstacle detection. All this information is combined using the method in [2], so the vehicle is properly localized. The vehicle is controlled by an on board computer.

II. Previous Work

In the literature, it is possible to find several planning methods that have been applied in the generation and selection of local paths. Most of these methods are based on a discrete optimization scheme [3], [4], [5] and [6]. From all the approaches of this kind, Rapidly-exploring Random Trees (RRT) and its variants are widely used in non-holonomic motion planning applications. However, real time implementations require efficient heuristics for the sampling configuration. Some examples of this kind of methods are [7], [8] and [9].

[1]http://verdino.webs.ull.es

FIG 1 Verdino prototype.

Some other methods, like the method introduced in this paper, are based in the transformation of the configuration space through the Frenét space. Some examples of this technique are the methods in [5], [3] and [10]. In [5], long term objectives are pursued, like speed keeping, merging, following, stopping. This is done through optimal control strategies within the Frenét frame of the street. In [3], lateral offset is defined as the perpendicular direction to an established base trajectory. This allows the vehicle to drive along the road, parallel to this trajectory. In [10], a set of candidate paths are also generated, with endpoints in fixed positions at different offsets with respect to the base frame, but they do not set this base frame in the center of the road, using a security cost for each candidate path instead. The safety of the path is computed by blurring the binary data of the obstacles.

III. Method

In order for the vehicle to navigate, a global plan has to be defined. This global plan is a rough estimate of the path that the vehicle has to follow to go from its current position to a desired goal.

The global path is generated using the NavFn global planner[2]. This planner implements Dijkstra's algorithm to find the best path through a cost map, which represents the goodness of the navigable areas taking into account static obstacles.

The vehicle can follow the global plan using the local planner. Obstacles are represented in a *costmap*, which is an occupancy grid which the local planner needs in order to select the best trajectories and avoid obstacles.

A. Generation of the Costmap

The costmap maintains information about occupied/free areas in the map, as an occupancy grid. It uses sensor data and information from the static map to store and update information about obstacles in the world, which are marked in the map (or cleared, if they are no longer there).

[2]http://wiki.ros.org/navfn

Each cell in the map can have 255 different cost values:

- A value of 255 means that there is no information about this specific cell in the map.
- 254 means that a sensor has marked this specific cell as occupied. This is considered as a lethal cell, so the vehicle should never enter there.
- The rest of cells are considered as free, but with different cost levels depending on an inflation method relative to the size of the vehicle and its distance to the obstacle.

The cost value of free cells decreases with the distance to the nearest occupied one, following the expression:

$$C(i,j) = \exp(-1.0 \cdot \alpha \cdot (\| c_{ij} - \vec{o} \| - \rho_{\text{inscribed}})) \cdot 253 \quad (1)$$

In this expression, α is a scaling factor which increases or decreases the decay rate of the cost of the obstacle. $\| c_{ij} - \vec{o} \|$ is the distance between cell $c_{ij} \in C$ (where C is the set of cells in the costmap) and the obstacle. Finally, $\rho_{\text{inscribed}}$ is the inscribed radius, which is the inner circle of the limits of the car. For a better explanation of the way in which the costmap is computed, please refer to [11]. An implementation of this method is available at http://wiki.ros.org/costmap_2d, as part of the Robotic Operating System (ROS) framework used for the development and testing of our approach. In the tests described in section IV a value of $\alpha = 3.0$ has been used.

B. Local Planner

Once the global path is defined, a method to compute the steering and speed commands is needed, in order to control the vehicle along this path. This method should also be able to avoid the obstacles present in the road. This has to be done in a safe and efficient way. The method developed to solve this problem is based on the solution described in [10], in combination with some ideas proposed in [3], taking into account the characteristics of the *Verdino* prototype.

The basic idea of the local path generation is to define a set of *feasible paths* and choose the best option in terms of their cost. The winner path defines the steering and speed commands that the vehicle will use. Having options among local paths is useful to overcome unforeseen obstacles in the road.

The current euclidean coordinate system is transformed into a new system based on the Frenét space. This space is computed as follows: the global path is considered as the base frame of a curvilinear coordinate system. The feasible local paths are defined in terms of this base frame in the following way:

- The nearest point of the main trajectory to the vehicle (where the distance is computed perpendicular to the global path), will be the origin of the curvilinear coordinate system.

- The horizontal axis will be represented by the distance over the global path, along its direction.
- The vertical axis is represented by the vector which is perpendicular to the origin point and points left of the path direction.

In this schema, trajectories can be easily computed in the curvilinear space (that is, maneuvering information is generated). These are then transformed to the original euclidean space, in which the obstacles information is added by assigning costs to each path.

As seen, the local path generation can be divided into two stages: the *candidate paths generation* and *the winner path selection*.

1) Candidate Paths Generation

In this stage, the base frame of the curvilinear coordinate system is defined, so that the algorithm will be able to compute the trajectories in this space as if the global plan were a rectilinear trajectory. The geometric relationship of the path in euclidean and curvilinear coordinates is shown at Fig. 2.

The coordinate origin of the base frame is defined as the closest point to the vehicle in the global path. The arc length of the base frame (s on the right image) is obtained as the distance of each point along the global plan (represented as a green line) to the coordinate origin. This distance is represented on the *x-axis* of the curvilinear system. On the *y-axis*, q represents the perpendicular lateral distance with respect to the path. The left side is represented by positive values and the right side by negative values.

For the computation of the transformation between the euclidean and the curvilinear coordinate system, path curvature κ is needed. This value is computed as follows [10], [5], [12]:

$$\kappa = \frac{S}{Q}\cdot\left(\kappa_b \cdot \frac{(1-q\cdot\kappa_b)\cdot(\partial^2 q/\partial s^2)+\kappa_b\cdot(\partial q/\partial s)^2}{Q^2}\right), \quad (2)$$

$$\text{where } \begin{cases} S = \text{sign}\,(1-q\cdot\kappa_b) \\ Q = \sqrt{\left(\frac{\partial q}{\partial s}\right)^2 + (1-q\cdot\kappa_b)^2} \end{cases} \quad (3)$$

There, κ_b is the curvature of the segment of the base trajectory used for the computation of the Frenét space. The candidate paths generation is performed in the curvilinear space, without considering the obstacles in the environment. These will be taken into account later, once the tentative trajectories are transformed to the euclidean space.

Maneuvering Paths Generation: The curvature of the generated paths is defined by the lateral offset q with respect to the base frame. First and second order derivatives of q are needed to compute κ (see equations 2 and 3), so a function dependent on the lateral offset is needed to compute a smooth lateral change.

q can be defined by a sequence of a cubic polynomial and a set of constants [10]:

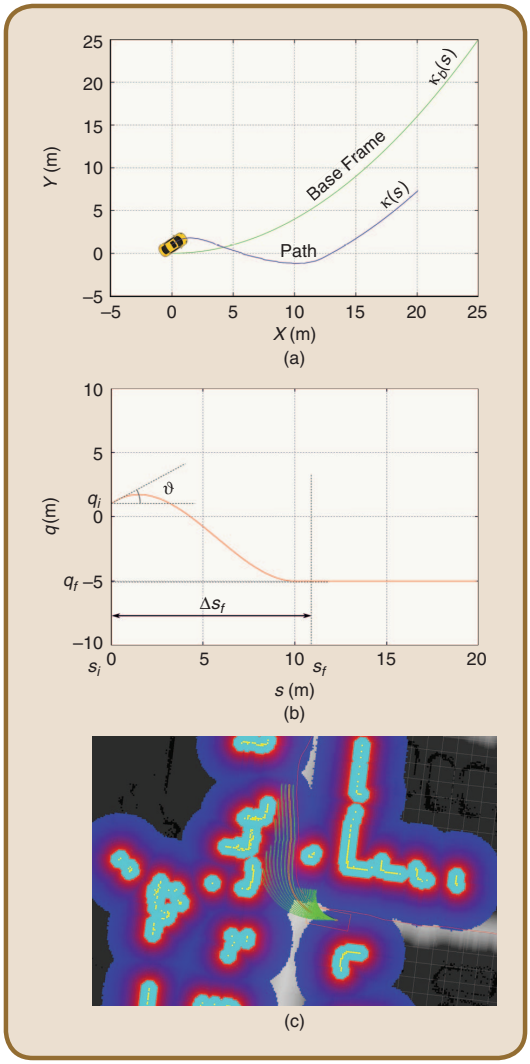


FIG 2 Conversion of a trajectory between the Cartesian and Frenét spaces, and paths truncation example.

$$q(s) = \begin{cases} a\cdot\Delta s^3 + b\cdot\Delta s^2 + c\cdot\Delta s + q_i & \text{if } s_i \le s < s_f \\ q_f & \text{if } s_f \le s \end{cases}$$
$$\frac{\partial q}{\partial s}(s) = \begin{cases} 3\cdot a\cdot\Delta s^2 + 2\cdot b\cdot\Delta s + c & \text{if } s_i \le s < s_f \\ 0 & \text{if } s_f \le s \end{cases}, \quad (4)$$
$$\frac{\partial^2 q}{\partial s^2}(s) = \begin{cases} 6\cdot a\cdot\Delta s + 2\cdot b & \text{if } s_i \le s < s_f \\ 0 & \text{if } s_f \le s \end{cases}$$

where $\Delta s = s - s_i$.

In Fig. 2b, the components involved in this process are depicted.

- The initial length s_i is zero, since the global planner is being pruned as the vehicle advances. Lateral offset q_i with respect to the global path's origin is also known.
- Angle θ defines the difference between the vehicle heading angle and the tangent angle of the base frame at the current position. If $\theta > 40°$, the vehicle enters in a recovery state, described in section III-B3. Once the vehicle is headed properly towards the path, this recovery state ends.
- s_f is a parameter that controls the longitudinal distance needed to reach offset q_f. This distance should be dependent on the speed. However, as the top speed of the prototype is not too high, Δs_f can be considered as the distance needed to go from q_i to the biggest q_f at top speed.
- The different q_f are computed separately for each path attending to the parameters defined by the user. In our implementation, the method receives as input the maximal width covered between the outer left and the outer right generated path $w_{\max}$, and the total number of paths to be generated (n_{paths}). So the value of q_f is computed as $w_{\max}/(n_{\text{paths}} - 1)$. s_f is also a free parameter provided directly by the user as the maximal longitudinal length that is desired to be covered by the paths.

Once the paths are generated in the curvilinear coordinate system, they are transformed to the euclidean space. In this new space, it is possible to evaluate their corresponding costs.

To define the maximum length of each candidate path, the cost of the cells corresponding to the points in the trajectory are inspected. If this cost is over threshold $\tau_{\text{circumscribed}}$, (which is the cost of cells at a distance equal to the radius of the minimum circumference that contains the outer limits of the vehicle footprint), the path is truncated at this point, as shown in Fig. 2c, where the generated paths are shown together with a colored costmap representation. In this figure, blue represents a low cost value, while the red color is used for the higher costs. Yellow and cyan correspond to lethal cells and inscribed cells, respectively.

2) Winner Path Selection

At each iteration, the winner path is selected through the use of a linear combination $J[i]$ of weighted cost functions, related to the following parameters: occlusion, length, distance to the global path, curvature and consistency of the path. $J[i]$ is evaluated as follows:

$$J[i] = \omega_o C_o[i] + \omega_l C_l[i] + \omega_d C_d[i] + \omega_\kappa C_\kappa[i] + \omega_c C_c[i] \quad (5)$$

Here, i is the path index, and C_o, C_l, C_d, C_κ and C_c are the costs of occlusion, length, distance to the global path, curvature and consistency, respectively. Their relative factors ω_k, $k \in \{o, l, d, \kappa, c\}$ are the associated weights that allow to adjust the influence of each of the costs to the final cost value.

The following costs will be computed for each candidate path independently in the euclidean space.

a) Occlusion

The occlusion cost is related to the safety of the path. This cost estimates the goodness of a path, such that the best paths are those which pass far enough from the obstacles. To do so, the footprint of the vehicle is simulated in each point of the path. The occlusion cost corresponds to the normalized maximum cost along the path:

$$C_o = \frac{\max\{c_i\}}{255}, \qquad i = 1 \ldots L \quad (6)$$

In this expression, L is the length of the current path being evaluated. $\max\{c_i\}$ is the maximum value of all the costs, associated to a point in the path.

b) Length

This cost is related to the length of the current path. The longer the path is, the lower its associated cost is. In general, a long path implies the existence of an area which is free of obstacles and can be safely traversed.

$$C_l = 1 - \frac{\sum_{i=1}^{L} \| p_i - p_{i-1} \|}{q_{f_{\max}} + s_f} \quad (7)$$

Here, p_i is a certain point inside the evaluated path. $q_{f_{\max}}$ is the maximum value that q_f can have for a certain path. Lengths are normalized to a value that a path will never reach.

c) Distance to the Global Path

This cost represents the lateral offset of the vehicle with respect to the global path. Tuning the associated weight of this cost will change the behavior of the vehicle when returning to the global path, after an occasional obstacle is avoided. It is computed as follows:

$$C_d = \frac{\sum_{i=1}^{L} \| p_i - \text{nearest}(p_i, g) \|}{L \cdot q_{f_{\max}}}, \quad (8)$$

where nearest(p, g) is the nearest point in the global path g to point p. This cost is normalized with respect to the maximum expected offset, $q_{f_{\max}}$.

d) Curvature

This cost allows to give priority to smoother paths. Let $p(x_i, y_i), i = 1 \cdots L$, be a point in the path. Then,

$$C_\kappa = \max\left\{ \frac{x_i' \cdot y_i'' - x_i'' \cdot y_i'}{(x_i' + y_i')^{3/2}} \right\}, \qquad i = 1 \ldots L \quad (9)$$

e) Consistency

This cost avoids continuous changes in winner paths between iterations. Once the vehicle starts a maneuver, it is

preferable to keep the same behavior during the following iterations. This is done through the following expression:

$$C_c = \frac{1}{s_2 - s_1} \int_{s_1}^{s_2} l_i \, ds \tag{10}$$

Lateral cost $l_i(s)$ is the distance between the current path and the previous winner path, at the same longitudinal position s; s_1 and s_2 are the first and last positions over s for which there are common points in both trajectories. At the beginning of the trip, the oscillation cost is 0, so it does not affect to the final choice of the path.

Once all costs are computed, the expression described in equation 5 is applied. In those paths for which it is impossible to advance, due to the presence of a nearby obstacle, the cost is forced to a negative value in order to indicate the rest of the system that this path is invalid.

The path with the lowest cost is selected (winner path W). If for some reason there is no valid path, the vehicle stops until the road is free of obstacles. If this situation does not change for a while, the recovery behavior process starts.

3) Recovery behavior

There are two scenarios in which the vehicle executes a recovery maneuver. The first one occurs when the vehicle is not correctly aligned with respect to the global plan, and the initial angle is too large to produce feasible local paths that comply with the curvature restrictions of the vehicle. The second case happens when the vehicle is correctly aligned but there are no feasible local paths to follow, due to the presence of a nearby obstacle.

In the first case, the recovery maneuver is intended to align the vehicle to the global path again. Same as for the local planner, the recovery behavior of the vehicle is composed of

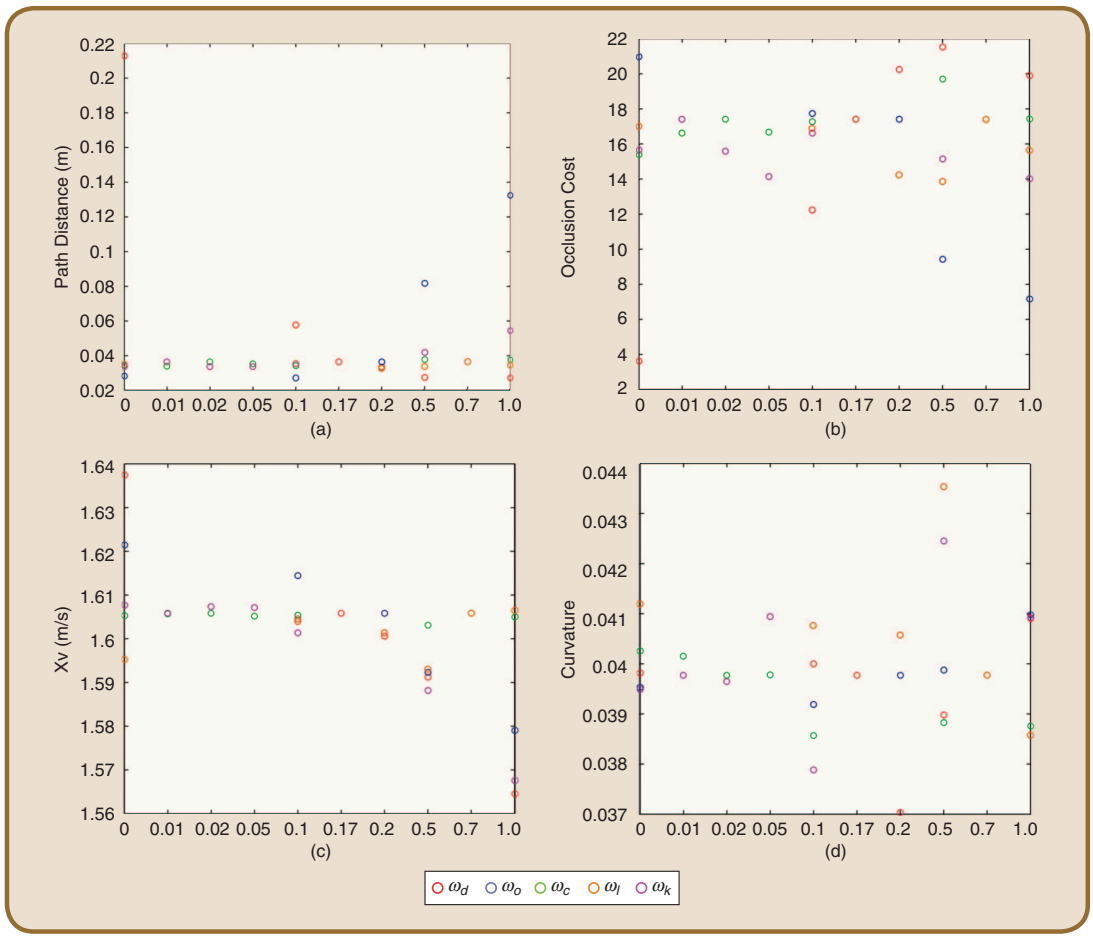


FIG 3 Results obtained from parameters.

two main phases. In the first phase, a set of feasible paths is generated. In the second phase, these paths are weighed in order to choose the best option. The vehicle will then try to follow the winner path. The recovery paths are chosen among four options: two forward paths and two backwards paths, setting the steering wheel to the maximum allowed angle at both left and right sides. The recovery maneuver is composed of a sequence of one or more of these paths.

In the second case, if possible, the vehicle just moves backwards for a short distance, in order to obtain enough space for the generation of feasible local paths.

IV. Results

In this section, the behavior of the local planner is described, using different cost weights as input.

A. Experimental Setup

In order to study the behavior of our method, several trajectories were followed, while recording a set of measured variables. These variables were *distance to path*, which measures the distance from the center of the vehicle to the closest point in the global path; *occlusion cost*, which measures the maximum cost of the cells below the vehicle footprint at each iteration; the *speed*, assuming that faster trajectories are preferred; and the *curvature* of the followed trajectory.

Since keeping the exact same conditions for all the tests is desirable, a simulator was used. In each test, the vehicle started at the exact same position and traveled towards the exact same goal. Obstacles were always in the same locations, and the only changing values were the input parameters under evaluation. The obtained results were then validated with some tests under real conditions, using the Verdino platform.

As seen in section III-B2, there are five different parameters that influence the overall cost, which will determine the chosen winner path. Each parameter has an associated weight. A base configuration of $\omega_d = 0.17, \omega_o = 0.2,\ \omega_c = 0.02, \omega_l = 0.7, \omega_k = 0.01$, which

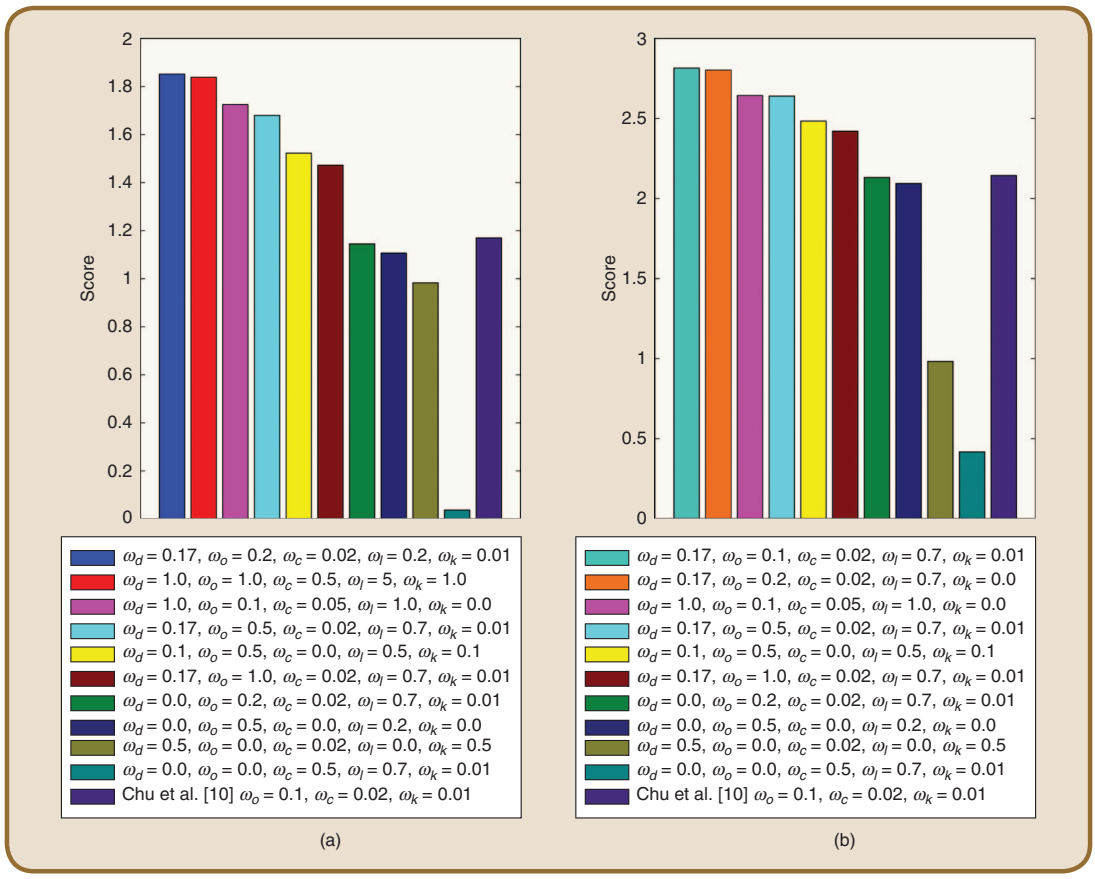


FIG 4 Results obtained from rankings.

provided good empirical results, has been used. Using the base configuration as a starting point, different weight configurations have been obtained by incrementally varying each weight.

> The recovery paths are chosen among four options: two forward paths and two backwards paths, setting the steering wheel to the maximum allowed angle at both left and right sides.

Fig. 3 shows how the variation of each individual weight influences the measured variables. In order to establish the relative importance of each weight with respect to the base configuration, for each performed test one of them varies in the [0, 1] range, while the rest keep their default values. Fig. 4 shows a ranking with different cost weights configurations. In Fig. 4a, the evaluation has been done by taking into account the path distance and the occlusion cost. In Fig. 4b, the evaluation takes into account the occlusion cost and path distance again, but also the velocity and the curvature of the followed path. Verdino is intended to be used in pedestrian areas. If the speed and curvature costs are taken into account when computing the overall cost of the trajectories, paths with high speed and predominantly straight are more likely to be selected. This means that the vehicle will be more aggressive and less capable of maneuvering. In crowded environments, it is usually better to take a longer, slower path that skirts obstacles (people) by a large margin, than a fast, straight path that traverses near obstacles. For this reason, in order to reach a compromise between speed and maneuverability, the

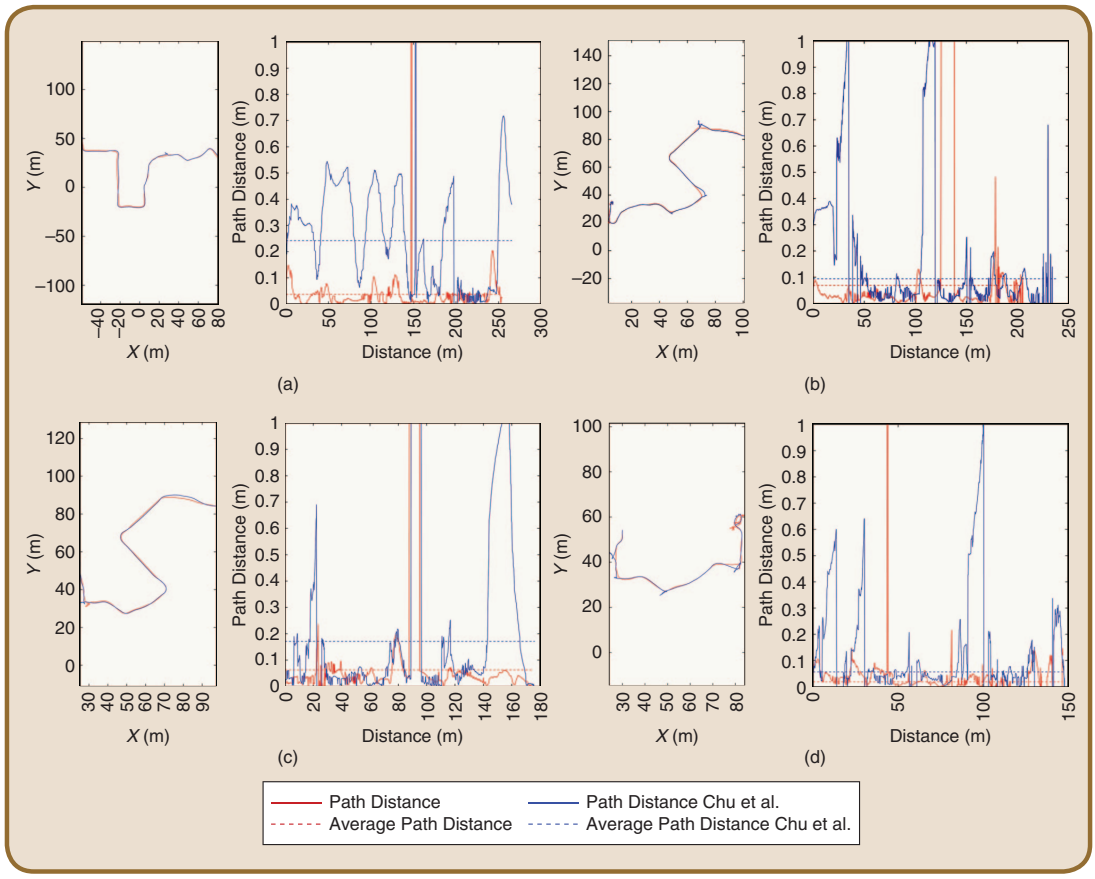


FIG 5 Traversed routes comparison of the proposed method (red) and the method proposed in Chu et al. [10] (blue).

The curvature of the trajectory is influenced by a combination of the different cost weights, and no particular cost influences it predominantly.

measured speed and the curvature are weighted by 0.5 to lessen their influence.

In order to aggregate variables with different units, all measured variables are normalized between 0 and 1 using the maximum and minimum values for each range. Also, prior to aggregating results, some variables are inverted. For example, the occlusion cost and the path distance are to be minimized, whereas speed and path length are to be maximized.

B. Cost weights determination

As expected, the measured variables are observed to be more greatly affected when varying their corresponding cost weight, whereas the influence of the other weights is not as evident. For instance, as Fig. 3a shows, varying the path distance cost causes the path distance variable to change proportionally. As Fig. 3b shows, the relationship between this cost weight and the occlusion cost is inversely proportional. This behavior may be produced due to the global plan traversing relatively high cost areas, mainly in closed curves. This occurs because the global planner does not take the vehicle dynamics into account when computing the global plan. When the cost weight of the path distance is high, the vehicle is forced to stick to the global path as much as possible and thus the cost can raise. If the occlusion cost weight is high, the vehicle's priority is to avoid obstacles as much as possible, so the distance to the global path grows.

Additionally, if the path length cost weight increases, the selection of long paths is favored. As the velocity command is computed as inversely proportional to the cost, in general, higher speeds are obtained when paths are longer (Fig. 3c). The navigation speed also depends on the distance to the obstacles, which is why the occlusion cost weight also influences this measurement. Moreover, if the curvature cost weight is high, the cost of the winner path in closed curves also increases. As speed is inversely proportional to the cost of the winner path, this causes the vehicle to slow down.

The curvature of the trajectory is influenced by a combination of the different cost weights, and no particular cost influences it predominantly (see Fig. 3d).

The rankings of Fig. 4a and Fig. 4b show which weight configurations produce the best results, based on different criteria. In Fig. 4a the ranking takes into account the occlusion cost and the path distance. In this sense, the configurations in the first places of the ranking favor mainly following the global path and avoiding obstacles. In the ranking shown in Fig. 4b the speed achieved by the vehicle and the curvature of the trajectory are also taken into account.

For the ranking in Fig. 4a, the best configuration was $\omega_d = 0.17, \omega_o = 0.2, \omega_c = 0.02, \omega_l = 0.2,$ and $\omega_k = 0.01$ (shown in blue color); and for the ranking in Fig. 4b, this configuration was $\omega_d = 0.17, \omega_o = 0.1, \omega_c = 0.02, \omega_l = 0.7,$ and $\omega_k = 0.01$ (shown in light blue color).

The method presented in [10] uses three weights: occlusion, curvature and consistency. For the sake of comparison, the path distance and path length costs weights have been set to zero in order to replicate the configuration used in [10]. This configuration is shown in violet color in Figs. 4a and 4b. As can be seen, there are several configurations with five costs that outperform the configuration with three costs proposed in [10]. As depicted in the charts, the use of the additional cost weights proposed in this work (ω_d and ω_l) produces better results in terms of vehicle behavior.

The charts of Fig. 5 compare the proposed method using the winner five-costs configuration of Fig. 4b (light blue color), to the method proposed in [10] that uses a three costs configuration, shown in violet color in Fig. 4b. On the left side of each figure, the traversed trajectory of the proposed method (red) and the method proposed in [10] (blue) are shown. On the right side of each figure, it is depicted the distance to the global path along the traversed trajectory of the presented method (red) and the method proposed on [10] (blue). Dashed lines show averaged path distance results.

As can be seen, the addition of the path distance costs makes the vehicle navigate closer to the global plan. Controlling the distance to the global plan is of capital importance in complex scenarios, like the ones Verdino is intended for. In these scenarios, mainly pedestrian areas, there are sharp turns and narrow navigable zones. Without a path distance cost, the vehicle may not follow the global plan properly. For example, when approaching a curve in which the vehicle passes from a wider to a narrower area, if the car predominantly selects paths with low occlusion costs, it will turn late. In sharp turns this can force the vehicle to initiate a recovery behavior to reorient itself, before it can continue following the global plan again. In the left side of the examples shown in Fig. 5, there are several situations where the method proposed in [10] has to

initiate a recovery behavior. In these situations, there is an abrupt change in the distance to the path, as shown in the right side of the examples. This occurs noticeably less frequently with the proposed method, which sticks properly to the global path.

Taking into account the obtained results, the winner configuration of Fig. 4b was chosen for the Verdino prototype. This method works at 10 *Hz*, on a *i7-3770K* processor, 16 Gb of *RAM DDR-3* memory, *SSD* storage and a NVIDIA GeForce GT 640. These times have been obtained under real conditions.

V. Conclusions

This paper presents a system which follows a global plan, by generating different tentative paths and choosing the most suitable one.

Different configurations have been tested and ranked, in order to select the parameters that will ultimately influence the vehicle behavior. These parameters are the length of the generated path, its distance to the global path, its proximity to obstacles, its curvature and its consistency. Using the obtained results, a winner configuration has been selected to be used in the real prototype Verdino.

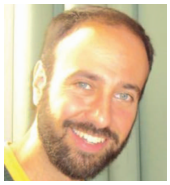
The results are compared to a similar method that uses three cost weights to select the local paths. As shown in the results section, the inclusion of two additional weights improve the navigation behavior of the prototype. This is the main contribution of the presented method with respect to the method presented in [10]. The inclusion of the path distance cost makes the vehicle follow the global plan more accurately than in the previous work presented in [10]. Additionally, the path length cost allows us to control the influence that the length of the winner path has in the speed of the vehicle. Another advantage is the use of a recovery behavior to overcome unforeseen situations, complementing the local planner and ensuring that the vehicle never gets stuck.

However, there is still room for improvement in the presented approach. One of the main drawbacks of the method is that if the angle between the vehicle and the path is too big, it is not possible to generate the paths, or they could not be followed by the vehicle due to physical restrictions. However, the use of the recovery maneuvers minimizes the impact of this limitation. Also, the obstacles are being considered as static, and no information about their previous motion is in use, which could lead to a more intelligent behavior of the vehicle. Although the iteration frequency of the method is high enough to reduce the effects of this lack of information, further research must be done in order to detect obstacles trajectories and to be able to include them in the costmap.

Acknowledgment

This work was supported by the project STIRPE DPI2013-46897-C2-1-R and the funds from the Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI), cofinanced by FEDER funds (EU).

About the Authors



Dr. Rafael Arnay was born in 1982. He received the M.Sc. and Ph.D. degrees (with honors) in Computer Science from the University of La Laguna (Spain), in 2007 and 2014, respectively. He joined the Department of Computer Science and Systems, University of La Laguna in 2007, where he is currently a Postdoctoral Researcher. His current research interests include unsupervised learning, object segmentation/recognition, feature extraction and deep learning.

Dr. Néstor Morales is a researcher at Universidad de La Laguna. He received his Ph.D. in Computer Sciences on 2014, and his Computer Sciences Engineer degree on 2007. His PhD was related to different techniques for computer vision based obstacle detection for autonomous vehicles both using monocular and stereo vision, and the ways used for their avoidance. He has published several papers in several national and international conferences and journals. His research interests are computer vision, obstacle detection and tracking, and mobile robots planning.

Antonio Morell was born in 1981 in Santa Cruz de Tenerife, Spain. He received his B.Sc. degree in Computer Science (Hardware Engineering) in 2006 and his B.Sc. degree in Automation and Industrial Electronics Engineering in 2010 from the University of La Laguna (ULL), Spain. After receiving his M.Sc. degree in Research, Development and Innovation in Science and Engineering in 2011 also from the ULL, he worked as a researcher in Project SAGENIA, funded by the Spanish Ministry of Economy and Competitiveness until 2013. Since then, he is a Research Assistant supported with a grant from the Canarian Agency for Research, Innovation and Information Society (ACIISI), pursuing a Ph.D. in Engineering Physics. His research interests include sensor fusion, mobile robots, autonomous navigation, biped locomotion and robot modeling and control.

Javier Hernández-Aceituno was born in 1986 in Santa Cruz de Tenerife, Spain. He graduated in Computer

Engineering in 2010 at the Universidad de La Laguna (ULL), Spain. He is currently conducting his research as a Ph.D. student in ULL. His current research includes mapping, navigation, and obstacle detection algorithms for autonomous vehicles.

Daniel Perea Ström was born in Stockholm, Sweden. He received the B.Eng. degree in computer science and systems engineering in 2007, and the M. Eng. degree in computer science engineering from the University of La Laguna, in 2008. Since 2007 he is with the Robotics Group of the University of La Laguna, conducting research in the field of mobile robotics. Since 2009 he is developer of the self driving car Verdino in charge of the localization and mapping systems. His research interests are in the areas of robot navigation, simultaneous localization and mapping, exploration, and machine learning approaches.

Dr. Jonay Toledo is an Assistant Professor at University of La Laguna (ULL). He received his Master in Computer Science in 2001, Master in Electronics in 2002 and Ph.D. in Automatic Control in 2008. His current research interests include mobile robots, autonomous vehicles, automatic control and embedded systems. He is part of the development team of Verdino, an autonomous electric car designed to drive autonomously in pedestrian areas. He has published several conference and journal papers in robotics, automatic control, artificial intelligent and autonomous cars.

Dr. Alberto Hamilton was born in 1968. He received the M. Sc. in 1991 in Physics from University Complutense of Madrid and Ph.D. in Computer Science in 1997 from University of La Laguna (Spain). He joined the Department of Computer Science and Systems, University of La Laguna in 1991, where he is currently senior lecturer. His current research interests include mobile robotics, intelligent agents and pervasive computing.

Dr. Javier Sánchez-Medina earned his Engineering Master Degree at the Telecommunications Faculty on 2002, and his PhD at the Computer Science Department on 2008. He is interested in the application of Evolutionary and

Parallel Computing techniques to Intelligent Transportation Systems. He has more than 20 international conference and 10 international journal papers. He is also very active as a volunteer of the IEEE ITS Society, where he has been serving in a number of different positions. Currently he is EiC of the ITS Podcast, the ITS Newsletter Vice-president of the IEEE ITSS's Spanish chapter and General Chair for IEEE ITSC2015.

Dr. Leopoldo Acosta is a Full Professor at the Universidad de La Laguna, Spain. He has been involved in several competitive nationally-funded research projects related to Artificial Intelligence and Robotics, four of them as project leader. He has directed six Ph.D.

References

[1] L. Acosta, J. Toledo, R. Arnay, J. Espelosín, N. Morales, D. Perea, and L. Moreno, "Verdino, prototipo eléctrico de vehículo autoguiado," in *XXXIII Jornadas de Automática*, Sept. 2012, pp. 729–736.

[2] D. Perea, J. Hernandez-Aceituno, A. Morell, J. Toledo, A. Hamilton, and L. Acosta, "MCL with sensor fusion based on a weighting mechanism versus a particle generation approach," in *Proc. 16th Int. IEEE Conf. Intelligent Transportation Systems (ITSC 2013)*, Oct. 2013, pp. 166–171.

[3] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *J. Field Robot.*, vol. 23, no. 9, pp. 661–692, Sep. 2006.

[4] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, A. Petrovskaya, M. Pflueger, G. Stanek, D. Stavens, A. Vogt, and S. Thrun, "Junior: The Stanford entry in the Urban Challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, Sep. 2008.

[5] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a Frenét Frame," in *Proc. 2010 IEEE Int. Conf. Robotics and Automation*, May 2010, pp. 987–993.

[6] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *J Field Robot.*, vol. 25, no. 11-12, pp. 939–960, Nov. 2008.

[7] M. J. Van Nieuwstadt and R. M. Murray, "Real-time trajectory generation for differentially flat systems," *Int. J. Robust Nonlinear Contr.*, vol. 8, no. 11, pp. 995–1020, Sep. 1998.

[8] S. M. LaValle, "Randomized kinodynamic planning," *Int. J. Robot. Res.*, vol. 20, no. 5, pp. 378–400, May 2001.

[9] Y. Kuwata, S. Karaman, J. Teo, E. Frazzoli, J. How, and G. Fiore, "Real-time motion planning with applications to autonomous urban driving," *IEEE Trans. Contr. Syst. Technol.*, vol. 17, no. 5, pp. 1105–1118, Sep. 2009.

[10] K. Chu, M. Lee, and M. Sunwoo, "Local path planning for off-road autonomous driving with avoidance of static obstacles," *IEEE Trans. Intell. Transport. Syst.*, vol. 13, no. 4, pp. 1599–1616, Dec. 2012.

[11] D. V. Lu, D. Hershberger, and W. D. Smart, "Layered costmaps for context-sensitive navigation," in *Proc. Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ Int. Conf.*, 2014, pp. 709–715.

[12] T. Barfoot and C. Clark, "Motion planning for formations of mobile robots," *Robot. Autonomous Syst.*, vol. 46, no. 2, pp. 65–78, Feb. 2004.

ITS

E

Fast object motion estimation based on dynamic stixels

sensors

MDPI

Article

Fast Object Motion Estimation Based on Dynamic Stixels

Néstor Morales *, Antonio Morell, Jonay Toledo and Leopoldo Acosta

Departamento de Ingeniería Informática, Universidad de La Laguna, Avda. Astrofísico Francisco Sánchez, s/n, San Cristóbal de La Laguna 38271, Spain; amorell@isaatc.ull.es (A.M.); jonay@isaatc.ull.es (J.T.); leo@isaatc.ull.es (L.A.)

* Correspondence: nestor@isaatc.ull.es; Tel.: +34-922-318-286

Academic Editor: Felipe Jimenez

Received: 22 April 2016; Accepted: 22 July 2016; Published: 28 July 2016

Abstract: The stixel world is a simplification of the world in which obstacles are represented as vertical instances, called stixels, standing on a surface assumed to be planar. In this paper, previous approaches for stixel tracking are extended using a two-level scheme. In the first level, stixels are tracked by matching them between frames using a bipartite graph in which edges represent a matching cost function. Then, stixels are clustered into sets representing objects in the environment. These objects are matched based on the number of stixels paired inside them. Furthermore, a faster, but less accurate approach is proposed in which only the second level is used. Several configurations of our method are compared to an existing state-of-the-art approach to show how our methodology outperforms it in several areas, including an improvement in the quality of the depth reconstruction.

Keywords: stixels; object tracking; object clustering; 3D reconstruction; autonomous vehicles

1. Introduction

Considerable work has been carried out to improve the efficiency and performance of obstacle-detection methods applied to Advanced Driver Assistance Systems (ADAS). Many solutions are based on dense environment reconstruction using disparity maps. Although these methods are useful for a detailed understanding of the environment, the reconstruction is dense and relies heavily on computer resources. Minimizing the area of the image to be processed allows for a simpler and lighter reconstruction based on certain assumptions.

Given a 3D reconstruction of the world, typically from a stereo input, though it can be 3D LIDAR data, a depth camera or similar, the objective is to simplify the scene's complexity by removing those parts of the environment with no information. The main objects in the scene are kept, but they are simplified. The model only focuses on the dominant objects in the scene, without a pixel-wise depth map, meaning the model can be estimated much faster than with traditional tracking methods. To this end, Badino et al. [1] proposed a representation of the world based on a set of rectangular sticks called stixels (from stick and pixel). Each stixel is defined by its 3D position relative to the camera and stands vertically on the ground, having a certain height, as shown in Figure 1. This compact, but flexible representation of the world can be used as the common basis for scene understanding tasks. The stixels can be generated without calculating a depth map by using techniques, such as V-disparity—or column-wise disparity—[2], which also offers substantial computational advantages. This fact is also the main reason why the original implementations from [1,3] are not generally used.

Figure 1. Stixels superimposed on their original image indicating detected obstacles. Colors encode the distance to the camera.

The main advantages of using such an approach are:

- Compact: significant reduction in data volume.
- Complete: information of interest is preserved.
- Stable: small changes in underlying data do not cause rapid changes in the representation.
- Robust: outliers have little or no impact on the resulting representation.

This obstacle detection and tracking method has been developed as part of the obstacle detection subsystem of our autonomous vehicle (Verdino) [4] (shown in Figure 2). Verdino is an electric vehicle designed to transport people in different environments, including pedestrian streets or tourist resorts, without needing a driver. Therefore, its behavior must be mainly reactive, with safety as its top priority. It has been modified to be able to drive autonomously at a maximum speed of 6 m/s, operated by an onboard computer. To this end, the original steering, brakes and accelerator have been modified and various sensors mounted on it [5,6], including a stereo camera. A crucial task for safe navigation is environment reconstruction, obstacle detection and motion prediction, so that Verdino can safely avoid obstacles.

Figure 2. The algorithm is intended for the Verdino prototype, designed to travel in pedestrian environments.

Tracking capabilities can be estimated by how stixels move between frames [7]. Stixels are valid for representing the area around a vehicle, and they provide enough detail for motion detection at a lower computational cost than optical flow where maximum object speed is limited. The contribution of this work can be summarized as:

- Good reconstruction quality in terms of computed depth. Free space computation without disparity maps has some drawbacks involving low depth accuracy. Object reconstruction and the detection scheme improve the correction of stixel depths and remove false obstacles.
- Better detection results and faster tracking than other methods, as in [7].
- Better robustness after changes between images (for example, when faced with a low frame rate).
- Stixel obstacle detection in crowded pedestrian areas provides reliability and speed at the same time.

In the next section, we discuss previous research on stixels. Section 3 describes the method pipeline. Section 4 presents a set of tests. Finally, conclusions are drawn in Section 5.

2. Previous Work

The problem of obstacle tracking has been well studied for its application in ADAS. In [8], a review of techniques applied to on-road systems, including vehicle detection, tracking and behavior understanding, is presented, making a special emphasis on vision-based algorithms. Many of these approaches use monocular vision for this task. An example is the work in [9], where lines painted on the road are detected by a single monocular camera, and an automatic steering control, speed assistance for the driver and localization of the vehicle are presented. In [10], the authors go one step further, trying to predict pedestrian behavior based on the Gaussian process, dynamical models and probabilistic hierarchical trajectory matching.

Stereo vision is also used to detect obstacles [11] using 3D information. Based on how much information they use, two subcategories can be found. First, there is a set of methods falling inside the category of 2.5D solutions. In this category, the complete information provided by 3D points is not used. Some of these methods use the 3D point as a feature, as in [12], in which dense variational optical flow estimation is combined with Kalman filtering for temporal smoothness and robustness. In [13], obstacles are represented as a rigid 3D point set, being tracked in terms of feature displacements and depth measurements. A very popular choice is the use of occupancy grids, like in [14,15]. About 3D solutions, they are usually based on complex grid maps that use complete 3D information. There are many ways of doing such a representation, i.e., with octree connected cubes [16] or voxel grids [17], used not only for stereo vision data [18]. This category includes sensor fusion approaches, like that in [19], where an obstacle tracking system for urban scenarios is made by a combination of odometry, LIDAR and computer vision, or in [20], where visible and FIR cameras are used to detect pedestrians.

Object tracking can be divided into online systems (for which tracking is done on a frame-by-frame basis), or offline systems (which take longer sequences into account), like in [21,22]. In the online systems, targets are usually followed using classic tracking approaches, like the Extended Kalman Filters (EKFs) [23], particle filters [24] or mean-shift tracking [25]. In [26], a simultaneously detection and trajectory estimation over a hypothesis test model extended with stereo depth and visual odometry is presented. Some solutions try to model the social behavior of the pedestrians in order to improve the obtained tracks, as happens in [27–29]. Other approaches use an intermediate solution between online and offline systems, like the Multi-Hypothesis Tracking (MHT) [30] or the Joint Probabilistic Data Association Filters (JPDAFs) [31].

Methods based on stixels [1,2,32,33] simplify the world defining only the 3D position relative to the camera and the height of the obstacle. Depending on how stixels are computed, two main trends emerge. In [3,34–36], free space is based on disparity maps, which use a probabilistic scheme to reduce the number of parameters. The number of objects captured along every column is assumed

to be small. Flying objects are penalized, and elevated objects have higher depths than lower ones. The work in [35] improves on [34] by using three different stereo confidences. In [3], a free space scheme that is able to reduce computational costs with a Kalman filter for tracking and clustering stixels is presented. Finally, in [36] the probabilities of a collision in a roundabout are computed.

The other research line is based on free space computation without disparity maps. In [2,7,33,37], a very high frame rate is achieved using a Sum of Absolute Differences (SAD) cube, with a cost associated with each row, column and disparity combination. This cube is used to compute the v-disparity, yielding a ground plane model. Stixels are computed using the points at the boundary with the ground (obtained with Dynamic Programming (DP)), including the height limitations of expected obstacles and left-to-right occlusion constraints.

3. Method

The method described in Figure 3 consists of the following steps:

1. Free space is computed from a stereo pair in order to estimate the ground plane.
2. Stixels are obtained and placed on the ground based on their depth and position.
3. At the first level, the stixels are tracked as per [7]. The set of stixels in the current frame is compared and matched to the previous one.
4. Stixels are clustered based on their projected position in 3D.
5. Using these clusters and the tracked stixels, tracking is performed at the stixel level. Obstacles in the scene and their velocities are calculated, and their positions in previous frames are recorded to estimate their future motion.
6. In the second level, tracking is performed only at the object level. Each obstacle is compared to obstacles detected in previous frames, meaning that stixel-level tracking is no longer needed.

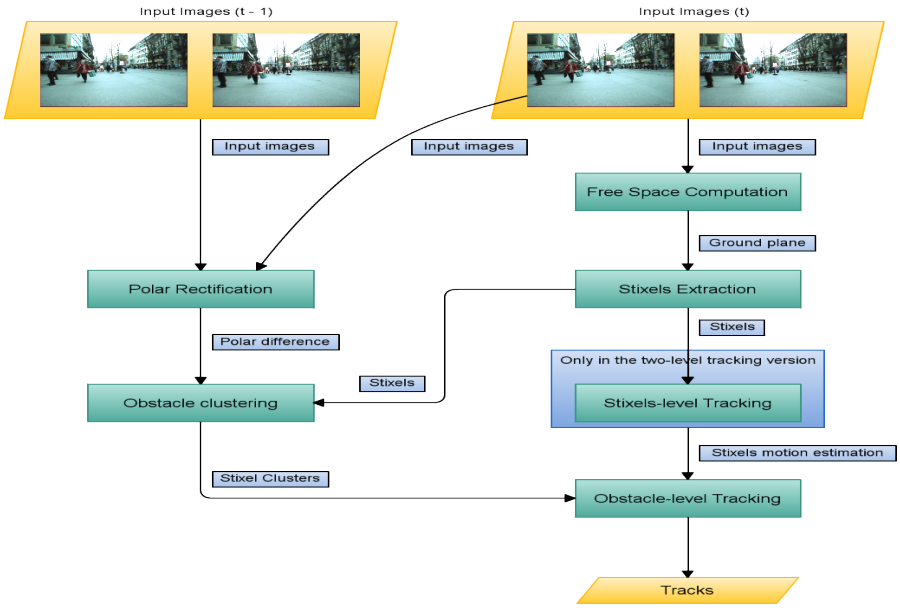


Figure 3. Graphical description of the method described in this paper.

Open-loop tracking is used at both the obstacle and stixel levels in order to reduce the calculation time. To determine the next position, only elements in the current frame are considered and linked to the following frame. The steps are detailed in the sections that follow. In Section 4, the advantages

and drawbacks of using either approach are detailed, and two-level-based tracking is described in the attached video (method pipeline). This pipeline is also valid for non-stixel based object tracking (see Section 3.3.2) if the first step of the algorithm is ignored.

3.1. Computing Stixels

Our stixel extraction method is similar to the one in [2], with the following assumptions:

- The algorithm's input is a calibrated stereo image pair.
- A Lambertian surface is assumed.
- The ground is planar, at least locally.
- Objects are mainly vertical with a limited height.
- The stereo rig has negligible roll with respect to the ground plane.

3.1.1. Computing the Free Space

The ground plane is estimated using data collected in the v-disparity domain [2]. Instead of computing and projecting a dense stereo depth map (much more computationally expensive), a function $f(u,v) = D$ is obtained in which (u,v) is the pixel position and D is the disparity of this position. For each row, the disparity with the lowest cost is extracted, and the ground level is obtained by robustly fitting a line on the v-disparity image. For optimization purposes, only one row of each N (where N is the number of rows) is computed, and the ground plane is interpolated.

3.1.2. Stixel Extraction

Stixel detection divides the image into multiple row bands b_i. Inside each band b_i and for each column u_i, the pixel with the largest horizontal gradient is selected [33]. This reduces the computational cost while increasing accuracy and provides us with a set of possible locations in which the bottom coordinate of each stixel could be located. In the presence of a horizontal stripe that could confuse the algorithm (like, for example, in the presence of cobbles), errors will be bounded by the band height.

Having a set of potential row bands that could be used as the bottom coordinate of each stixel, the next step is to localize the optimal one. The likelihood of the presence of a stixel q at row band b is calculated based on the cost of the presence of a vertical object at that location; the probability of the supporting ground being present; and a smooth term to force the left-right occlusion restrictions, by promoting ground-object boundaries with few jumps. The ways in which these costs are computed are beyond the topics covered in this paper, but more information can be found in [2]. The minimum size of the stixel is set to 10 pixels. The results after this stage are shown in Figure 1, where stixels (in colored depth scale) are superimposed on the left image. More information is provided in Sections 3.1 and 3.2 and in [2].

3.2. Tracking

Two different tracking approaches have been explored. The first one is based on two tracking levels. The first level tracks stixels independently (stixels in the current frame are matched with another, or none, in the previous frame by minimizing the cost function associated with matching two stixels). In [7], this is done using DP. In our implementation, a bipartite matching graph is used. In the second level, stixels are clustered into objects, which are matched based on the inner stixels previously tracked.

In the other approach, only the second level is performed. The tracking does not consider stixels included in objects. Stixels are only used in the clustering and reconstruction process. Section 3.2.1 applies only to the two-level approach, while Section 3.3 is common to both approaches.

For this stage, some assumptions were made:

- All stixels are assumed to be properly estimated.
- The maximum object speed is limited, so the search range between stixels is constrained. As there is just one stixel per column, matching is limited to a search in the u direction.
- Since two consecutive frames are relatively close, the same stixel at time t and $t-1$ should look similar, including its height. Section 4.3.2 shows that this restriction can be reduced depending on the tracking approach.

3.2.1. Stixel-Level Tracking

The tracking objective is to match each stixel at column $q_i\{t\}$ with the corresponding stixel in the previous frame $(t-1)$. This process can be thought of as a pair matching problem. A bipartite graph, in which the nodes are the stixels in frames t and $t-1$ and the edges are associated with a certain motion cost c_m, is used to match the stixels. This is represented by Equation (1).

$$c_m(u_i\{t\}, u_j\{t-1\}) = \begin{cases} f_{cost}(u_i\{t\}, u_j\{t-1\}) & \text{matching} \\ \infty & \text{other} \end{cases} \quad (1)$$

Here, a match is applicable if and only if the following restrictions are satisfied:

- $|X(u_i\{t\}) - X(u_j\{t-1\})| < \tau_{max\_disp}$, where parameter τ_{max_disp} indicates the maximum stixel displacement between frames; and $X(u)$ is the position in 3D coordinates in the longitudinal axis X, which grows from left to right in 3D Cartesian coordinates. Axis Y is the vertical axis, which grows downwards, and the Z axis starts from the local coordinate system of the robot towards its front.
- $u_i\{t\}$ is not the first frame in which stixel $u_i\{t\}$ appears.
- Stixels $u_i\{t\}$ and $u_j\{t-1\}$ are not occluded.
- $f_{cost}(u_i\{t\}, u_j\{t-1\}) < \tau_{max\_cost}$.

If a match is not found, the cost is infinite, and thus, the link is not included in the graph. The cost function is described in Equation (2).

$$f_{cost}(u_i\{t\}, u_j\{t-1\}) = c_{SAD} + c_{hist} + c_{height} \quad (2)$$

with:

$$\begin{aligned} \alpha_{SAD} &\cdot f_{SAD}(u_i\{t\}, u_j\{t-1\}) \\ \alpha_{hist} &\cdot f_{hist}(u_i\{t\}, u_j\{t-1\}) \\ \alpha_{height} &\cdot f_{height}(u_i\{t\}, u_j\{t-1\}) \end{aligned} \quad (3)$$

Here, $(\alpha_{SAD} + \alpha_{hist} + \alpha_{height} = 1)$ are the weights of each cost function, which are described next.

3.2.2. Sum of Absolute Differences

In the bibliography, stixel matching is based on SAD applied pixel-wise over the RGB color scheme between frames $u_i\{t\}$ and $u_j\{t-1\}$. In [7], stixels are resized to measure 30 px. It is also used in the Results Section in order to compare the approaches.

3.2.3. Histogram Matching

Our method relies on histograms to match stixels. Stixel size varies due to object position changes between frames or due to noise in the stixel height detection. To normalize this effect, a histogram is computed for each stixel, and a Hellinger distance between frames is calculated [38].

$$f_{hist}(u_i\{t\}, u_j\{t-1\}) = 2 \times \sqrt{1 - \sum_{i=1}^{d} \sqrt{H(u_i\{t\})[i] \times H(u_j\{t-1\})[i]}} \quad (4)$$

$H(u)[i]$ is the i-th bin in the histogram computed for stixel u, and d is the number of bins in the histogram. In our implementation, $d = 64$, and the histograms are normalized.

Using this method to match stixels could lead to a bad score in certain circumstances, like in the extreme case in which both stixels have a constant, but almost similar brightness. In the unlikely circumstance that this happens, neighbor stixels will be properly matched. This will allow, in the next step, to correct these situations and match the stixels at the object level properly. This fact will be made clearer in Section 3.3.

3.2.4. Height Difference

This metric is used to complement others, since by itself, it is not discriminative enough for a proper match, but it can help in the case of very similar scores in two or more possible matches. f_{height} is computed as in Equation (5).

$$f_{height}(u_i\{t\}, u_j\{t-1\}) = 1 - |h(u_i\{t\}) - h(u_j\{t-1\})| \tag{5}$$

$h(u)$ is the height in real-world coordinates of the stixel in column u.

Section 4 shows the results for different α_{SAD}, α_{hist} and α_{height}. f_{cost} is used to weight links between bipartite graph nodes. Figure 4 shows a representation of this graph. Nodes (stixels) at the current time are represented as u_{p_i} and previous stixels as u_{q_i}. Match costs are assigned to edges as $\omega_{i,j}$. The minimization problem is shown in Equation (6).

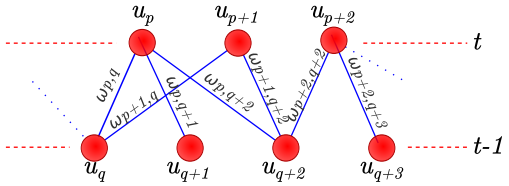


Figure 4. Bipartite matching graph representation for matching stixels between frames.

$$\hat{\mathcal{M}} = \arg\min_{\mathcal{M}} \sum_{(i,j)\in\mathcal{M}} \omega_{i,j}, \quad \exists!(i,\cdot) \wedge \exists!(\cdot,j) \tag{6}$$

A $O(n \cdot m \cdot log(n))$ Edmond's maximum weighted matching algorithm [39] is used instead of DP [7]. This achieves better times and ensures that each match is performed one-to-one. In [7], a stixel can be matched with more than one stixel in the next frame. This complicates trajectory tracking, since multiple paths can be obtained for the same stixel. In our implementation, a matching set that maximizes the whole matching cost was chosen, ensuring that each stixel is matched with just one stixel.

3.3. Obstacle-Level Tracking

In this section, we describe the obstacle-level tracking. The first step is clustering, which joins every stixel with a similar depth into the same obstacle. The aggregation step fuses obstacles obtained from clustering with similar characteristics. In obstacle filtering, false obstacles are removed using obstacle motion and two-camera information. After these steps, obstacles are tracked between two consecutive frames. The algorithm steps are detailed in the following sections.

3.3.1. Clustering

The first step is clustering, the goal of which is to join stixels with similar depths into fewer obstacles. Each obstacle consists of a set of similar stixels, from left to right. The Algorithm 1 is used for this step.

Algorithm 1 Clustering algorithm.

```
1:  function CLUSTERING(Q{t})
2:      O ← ∅
3:      o ← ∅
4:      for each stixel q_i ∈ Q, from left to right do
5:          if |depth(q_i) − depth(q_{i−1})| > τ_{depth_dist} then
6:              if width(o) > τ_{min_width} then
7:                  o ← ∅
8:              end if
9:          end if
10:         o ← o ∪ q_i
11:     end for
12: end function
```

$\mathcal{Q}\{t\}$ are the stixels in current frame t. From left to right, stixels are accumulated until the depth difference between stixels is greater than τ_{depth_dist}. When the right border of an obstacle is reached, it is added to $\mathcal{O}$, and the clustering process starts for new obstacles. If an obstacle is not wide enough, it will be rejected. Stixels generated due to noise, as shown in Figure 1, are removed. $\mathcal{O}$ also includes parameters, such as object depth, which is computed as the minimum depth between all of the clustered stixels. Figure 5a shows the results after clustering.

Obstacle Aggregation

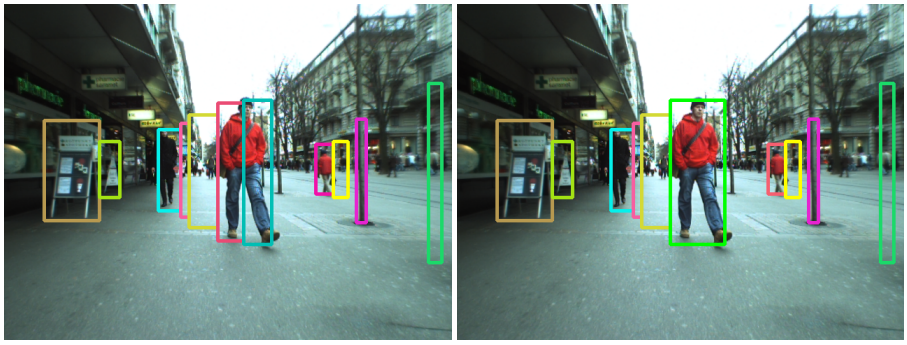
Sometimes, stixels are located at a depth different from their real position, as shown in Figure 5a where the legs of a person in the foreground are separated enough to show the ground between them. This confuses the detection process, which regards the obstacle's base as the central part of this person and not his feet. The process described in Algorithm 2 reduces this effect.

Algorithm 2 Aggregation algorithm.

```
1:  function AGGREGATION(O)
2:      O' ← ∅
3:      o' ← ∅
4:      for each object o_i ∈ O, from left to right do
5:          if |X(o_i) − X(o_{i−1})| > τ_{lateral_aggregation_dist} or |Z(o_i) − Z(o_{i−1})| > τ_{depth_dist} then
6:              O ← O ∪ o'
7:              o' ← ∅
8:          end if
9:          o' ← o' ∪ o
10:     end for
11: end function
```

All previously-detected obstacles are tested, again from left to right. If the lateral distance (in world coordinates) is less than $\tau_{lateral\_aggregation\_dist}$, the depth difference is checked again. If it is less than τ_{depth_dist}, the two obstacles are joined. Figure 5 shows this process. In the left image, the person in first plane is divided into two different obstacles. After aggregation, this is assigned to a single obstacle. The final obstacle depth between the two obstacles is regarded as minimal.

Sensors **2016**, *16*, 1182 9 of 23



(a) **(b)**

Figure 5. Obstacles detected before and after aggregation. The algorithm joins the stixels that belong to the same obstacles. (**a**) Obstacles before aggregation; (**b**) Obstacles after aggregation.

Obstacle Filtering

Figure 5b shows some false obstacles, such as those between the two pedestrians on the left side of the image (in pink and yellow). There is another next to the man in the background (yellow) and the last one on the right side of the image (green). Signs and poles are not considered false obstacles, since they are elements to be avoided.

In order to distinguish real obstacles from false ones, the images captured are recorded so that motion can be detected. Motion can originate both from obstacles (i.e., a person walking) and camera movement. This allows detecting occluded or changing areas so that new obstacle borders can be detected.

The search for correspondences between the two images relies on polar rectification, as in [40]. The first step defines the common region between images, so the epipoles and the homography H must be calculated using the fundamental matrix F [41]. Epipolar geometry is described by Equation (7).

$$m_{L,t-1}^{T} \times F \times m_{L,t} = 0 \tag{7}$$

where $m_{L,t-1}$ and $m_{L,t}$ are homogeneous representations of corresponding image points in the left image of frames t and $t-1$. Correct correspondences must be obtained in order to yield F, so they are computed in the following order [17]: $I_{L,t} \rightarrow I_{R,t} \rightarrow I_{R,t-1} \rightarrow I_{L,t-1} \rightarrow I_{L,t}$, where $I_{\{L,R\},t}$ is the left (L) or right (R) image in frame t. From an initial set of features in $I_{L,t}$, valid matches in $I_{R,t}$ are obtained. The cycle is complete when $I_{L,t}$ is reached, keeping only valid matches. Figure 6 shows the results of the matching process, where each matching cycle is represented by the same random color. A match is valid if the following holds:

- The points obtained should be the same for the entire cycle.
- Features in $I_{L,t}$ must be in the same row as $I_{R,t}$. The same applies to $I_{L,t-1}$ and $I_{R,t-1}$.
- The distances between features in frames t and $t-1$ should be similar.

In order to detect changed pixels and to remove false stixels, frame t is aligned to $t-k$ [40] to obtain a pixel-wise absolute difference. A stixel is considered valid if it is consistent in the left and right images, in the current and previous frame. Figure 7 shows this difference thresholded, binarized and projected back to current image coordinates. Small noise differences are rejected. For each obstacle, its Region Of Interest (ROI) is determined, and its top half is rejected, meaning the algorithm only looks for obstacle motion close to ground, since obstacles usually exhibit more motion in their lower half (legs or wheel movements). In static obstacles, the motion due to camera movement is more or less uniform throughout the entire object. Changes due to perspective are also small over planar ground.

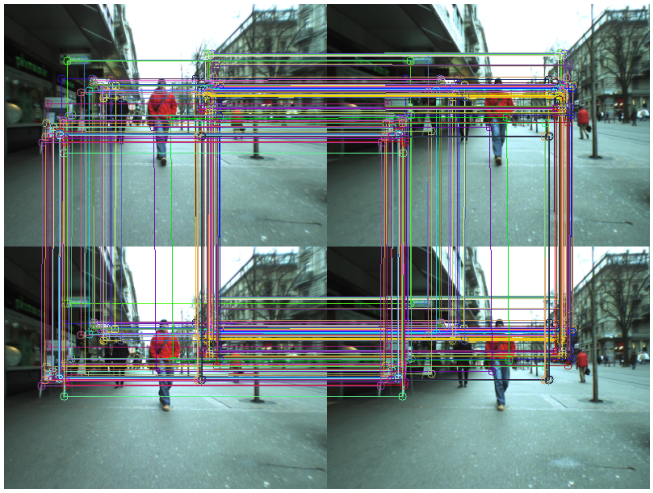


Figure 6. Common points between frames t and $t-1$ in the matching cycle.

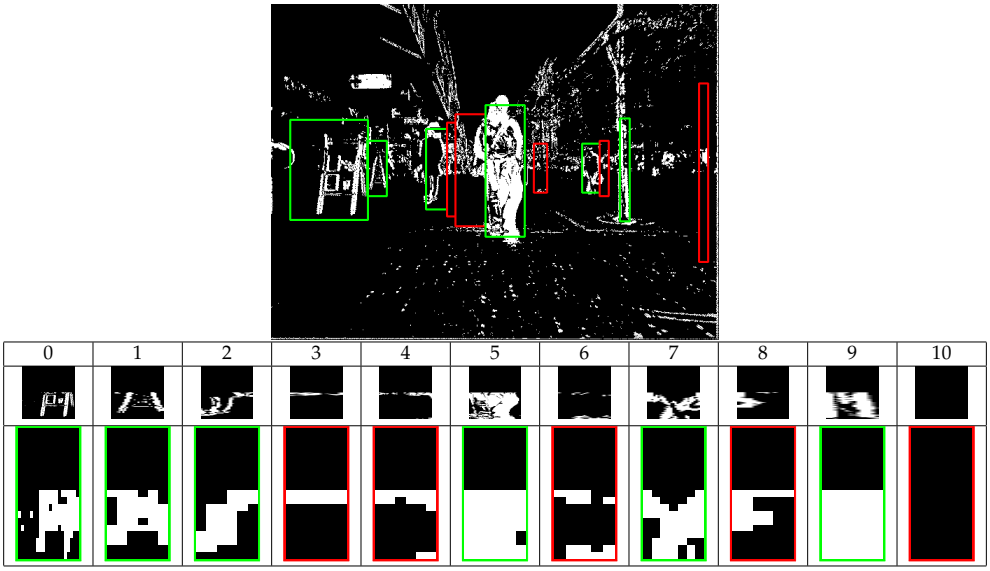


Figure 7. Object filtering phase. Binarized motion image (**top**) ($k = 0.2$ s); Occupancy maps on the ground (**bottom**).

The points obtained after the thresholding process are located in their corresponding position in 3D coordinates. The ground is divided into an occupancy grid of 10×10 cm cells. When a point falls in the cell, it is marked as occupied. Figure 7 shows examples of motion, ROI and an occupancy grid. Real obstacles, like 5 or 7, exhibit higher densities compared to 4. Even the motion of 2 (a man in a black suit where the colors complicate detection) is properly detected. To improve detection, a frame is not compared to the one immediately preceding it, but to that corresponding to $t-k$ (in seconds, where t is the current time), which makes differences due to motion more noticeable. In our tests, $k = 0.2$ s, which, despite being a conservative value, makes the differences appreciable. Obstacles are rejected as per Equation (8).

$$false(o) = \begin{cases} true & \text{if } \frac{count(G_o,true)}{count(G_o,true)+count(G_o,false)} > \tau_{occ} \\ false & otherwise \end{cases} \quad (8)$$

$count(G_o, j)$ counts occupied cells in the occupancy grid G_o. τ_{occ} is the threshold parameter. The width of each obstacle in real-world coordinates is also checked. Figure 7 shows rejected (red) and accepted (green) obstacles.

3.3.2. Tracking

The first tracking method is based on Section 3.2.1, where the initial stixel level matching is used to maximize matches between obstacles. The second one matches directly using template matching techniques since the differences between frames are small. The results from applying both methods are shown in Section 4.3. The first method exhibits better recall along frames; however, the second is faster, with lower, but acceptable, recall.

Two-Level Tracking Approach

The tracking problem is regarded as a pair matching process repeated over time. The correspondence matrix $C_{|\mathcal{O}\{t\}|\times|\mathcal{O}\{t-1\}|}$ stores the number of correspondences between stixels in current and previous frames. The tracking process is described in Algorithm 3.

Algorithm 3 Two-level tracking algorithm.

```
function TRACKING(O{t}, O{t−1})
    C_{|O{t}|×|O{t−1}|} ← 0
    for each object o{t} ∈ O{t} do
        for each stixel q{t} ∈ o do
            Find correspondence q{t−1} for q{t}
            Find the object o{t−1} ∈ O{t−1} associated to q{t−1}
            if o{t−1} found and ‖o{t} − o{t−1}‖ < τ_max_obst_dist then
                C(o{t}, o{t−1}) ← C(o{t}, o{t−1}) + 1
            end if
        end for
    end for
end function
```

Two objects are associated between frames if there is at least one stixel correspondence and they are sufficiently close, assuming that the motion between frames is small (if the frame rate is high). Matched pairs $\hat{\mathcal{C}}$ are obtained by solving the maximization problem in Equation (9) using a correspondence matrix. Each track is stored in an internal structure that associates tracks with obstacles, allowing for the inclusion of new obstacles. The results are shown in Figure 8, Section 4 and in the method pipeline video.

$$\hat{\mathcal{C}} = \arg\max_{\mathcal{C}} \sum_{(i,j)\in\mathcal{M}} C(i,j), \quad \exists!(i,\cdot) \wedge \exists!(\cdot,j) \quad (9)$$

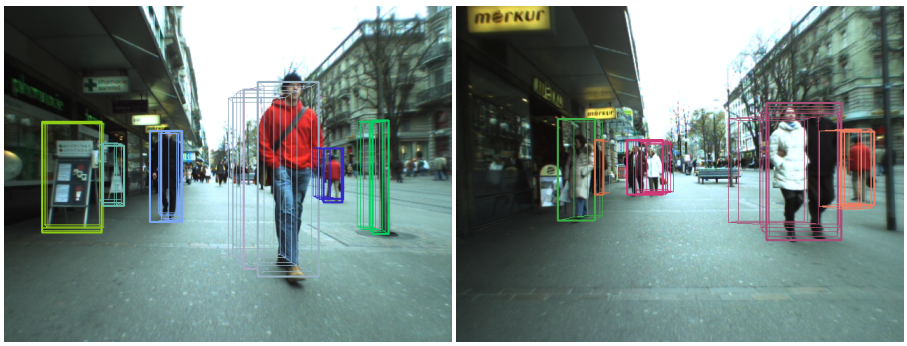
(a) **(b)**

Figure 8. Two-level based tracking algorithm results. In the image, the stixels detected in the current and previous frames are superimposed. Both frames were extracted from Bahnhof sequence. (**a**) Frame 30; (**b**) Frame 320.

Object Tracking Approach

A cost matrix (Equation (10)) is not generated using stixels associated with obstacles, since this information is not available. The histogram difference described in Section 3.2.1 is used, but for each pair of obstacles and not at the stixel level. The tracking problem thus becomes the same as in the two-level tracking case, in which Equation (9) is maximized. Figure 9 and Section 4 show some tracking results.

$$C(o\{t\}, o\{t-1\}) = 1 - \left(2 \times \sqrt{1 - \sum_{i=1}^{d} \sqrt{H(o\{t\})[i] \times H(o\{t-1\})[i]}}\right) \quad (10)$$



(a) **(b)**

Figure 9. Object-based tracking results. In the image, the stixels detected in the current and previous frames are superimposed. Both frames were extracted from Bahnhof sequence. (**a**) Frame 15; (**b**) Frame 126.

3.3.3. Integration with the Navigation Subsystem

This work is intended to provide the input for the navigation subsystem of our autonomous vehicle, Verdino. The navigation scheme is an adaptation of [42] using [6] as the localization system.

It computes a set of tentative trajectories based on the Frenét space [43,44] (which is shaped according to a global plan, which connects the current position to a given target [45,46]). These trajectories are projected back to Euclidean space. Tentative paths are weighted, using factors such as length, curvature and safety. A layered costmap [47] is used to connect the navigation subsystem and obstacle detection using an occupancy grid. Information on obstacles (stixels and their motion) is stored or updated by marking them on the map. The costmap consists of two different layers.

The first layer represents the stixels in the current frame, projected and transformed to map coordinates. The technique of growing the obstacles allows planning the vehicle's movements as if it were a point, without occupying space, which simplifies the planning. Every obstacle detected by the vision module is grown to vehicle size, so that the vehicle will not crash into obstacles even when the vehicle's planning does not consider size (Layer 1). The world map is a grid in which obstacles are represented using values from 0 to 255, where 0 represents a free area and 255 an obstacle. The cost of each cell $c(x, y)$ in the map is calculated using Equation (11). This cost is used by the autonomous vehicle to calculate a safe path that avoids the obstacles detected by this stixel method.

$$cost(c) = 253 \times e^{\beta \times (\rho - \|nearest(c) - c\|)} \quad (11)$$

β is a scaling factor that defines the cost function's slope; $nearest(c)$ is the nearest obstacle cell; c is the current position; and ρ is the circumscribed radius of the vehicle.

The second layer represents the obstacle's motion, transformed and referenced to the map. A Kalman filter is applied to past trajectories to predict future ones. Obstacle growth is also carried out in this layer, but the vehicle is allowed to approach the possible future positions of obstacles more than it is allowed to approach them in the current position (Layer 1). Figure 10 shows the navigation subsystem integration. Tentative trajectories are long in free areas and short when close to obstacles. The attached video stixel world-based navigation shows a full navigation sequence.

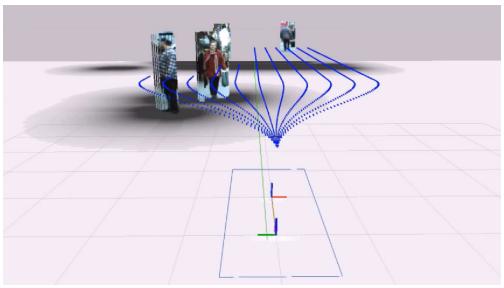


Figure 10. Navigation subsystem integration with stixel detection. A gray-scale costmap layer is included where black represents an obstacle and white is free space. The gray scale is generated using Equation (11). The possible routes that the vehicle can take are shown in blue.

4. Results

Four factors were considered when evaluating the method:

1. The quality of the clustering process.
2. Stixel depth accuracy compared to object-level tracking.
3. How well tracks are recalled under various conditions.
4. Computational time.

The results obtained in this paper are compared to [7,37] using the Bahnhof sequence [26] (7400 obstacle annotations, height $\geq$ 40 px, 999 stereo pairs, 640 $\times$ 480 pixels, 15 fps).

4.1. Clustering

This test is applied to the clustering method described in Section 3.3.1. Detections are compared to actual obstacles in each frame. The method is tested with and without filtering, as described in Section 3.3.1. Figure 11 shows its results.

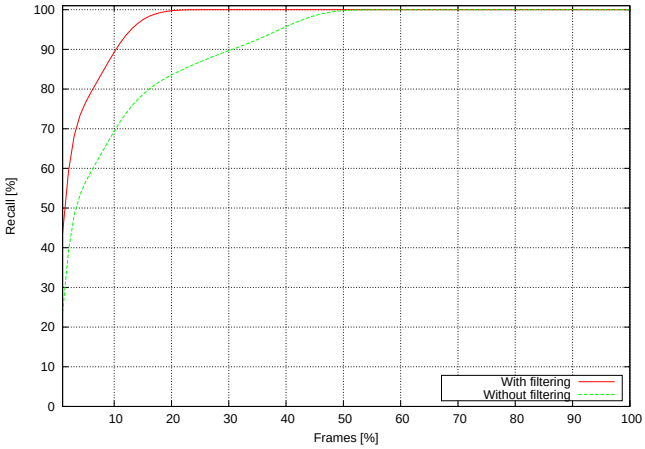
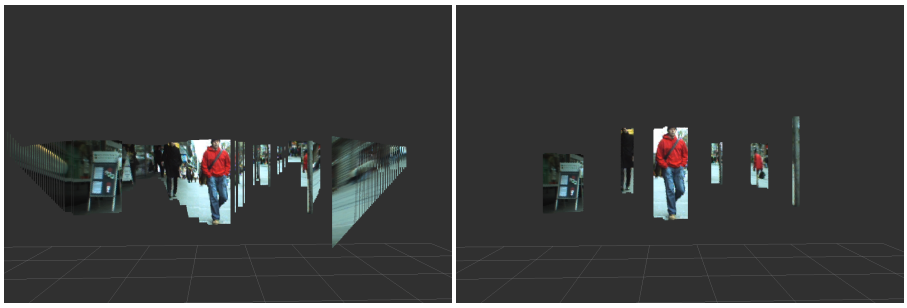


Figure 11. Obstacle detection rate as a function of the number of frames analyzed for a sequence.



(a) **(b)**

Figure 12. Stixel comparison between [2] and this paper in the same frame. (**a**) Stixels detected by [2]; (**b**) Stixels detected by our method.

Figure 11 shows, ordered by recall, the whole sequence processed frame by frame. The total sequence frame percentage is on the x axis and the recall on the y axis. The results of the stixel detection method are compared to the annotations included in the dataset in order to calculate recall. The graph indicates that analyzing only one frame yields a recall rate of 50% for the filtered option and 30% for the non-filtered option for all of the obstacles included in the whole sequence, which are presented in the current frame. If only 10% of the sequence is analyzed, the recall grows to 90% in obstacle detection (70% error in the non-filtered option). This means that by analyzing just 10% of the frames in the sequence, 90% of the obstacles present in those frames can be detected. Analyzing 20% of the frames in the sequence yields a 100% recall rate (55% of frames in the non-filtered version).

Figure 12a, shows the original stixels (projected in 3D) with considerable noise (especially between obstacles) and free areas detected as obstacles. In Figure 12b, only obstacle stixels are represented, with the depths restored after the clustering process.

Sensors **2016**, *16*, 1182 15 of 23

4.2. Stixel Accuracy

Stixel depth after clustering is compared to the disparity map shown in Figure 13 and used as the ground truth. The error in the pixels is calculated as an average of disparity differences between the stixel depth and the disparity map using Equation (12).

$$error = \frac{\sum_{q_i \in Q} \sum_{v \in V} \| disp_{GT}(v, q_i) - disp_Q(q_i) \|}{N \times d_{max}} \times 100 \quad (12)$$

$V = \{b(q_i), \ldots, t(q_i)\}$, $b(q_i)$ is the bottom row of stixel q_i; $t(q_i)$ is the top row of stixel q_i; $disp_{GT}(i, j)$ is the ground truth disparity at a certain row i and column j; $disp_Q(q_i)$ is the disparity computed for the stixel q_i; N is the total number of pixels being compared; and d_{max} is the maximum disparity allowed.

Figure 13 shows the error for each frame in a sequence in ascending order. The stixel error (red) grows faster than the clustered obstacle error (green). Approximately 95% of the frames with clustered obstacles have a disparity error below 10%. However, just 60% of the frames exhibit a disparity error below this value with the original stixel computation. Figure 14 shows the ground truth disparity map, the original stixels [37] and the clustered obstacles in a color scale where red represents lower disparities (further) and blue higher disparities.

4.3. Tracking

In this section, tracking evaluation tests are shown in terms of the recall measured using two different criteria: tracking capabilities after a few frames (track length) and performance when the time between frames is increased. Table 1 shows a selection of the most representative configurations.

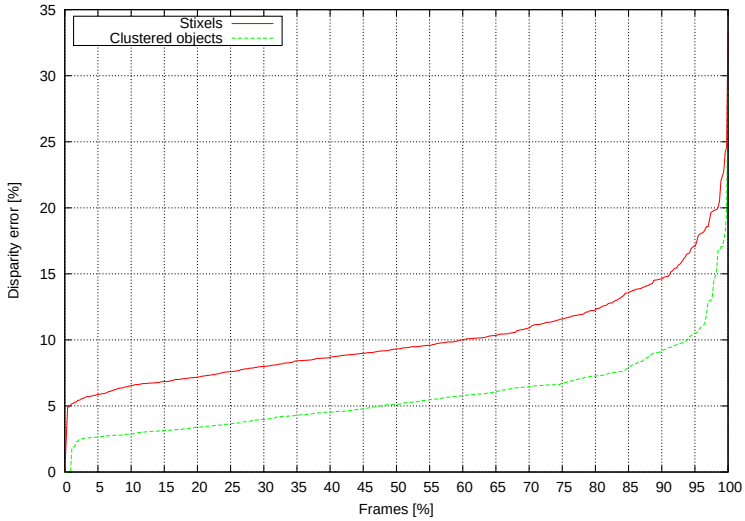


Figure 13. Disparity difference between stixel and object clustering.

There are two configurations based on [7]: the first one just uses the SAD cost, and the second one is the final configuration described in [7]. Configurations 3 to 6 apply the method presented in Section 3.3.2 (two-level tracking approach), where Configuration 1 and 2 parameters are used, plus an evaluation of α_{hist} factor. Configuration 7 presents object-based tracking results (Section 3.3.2, object tracking approach).

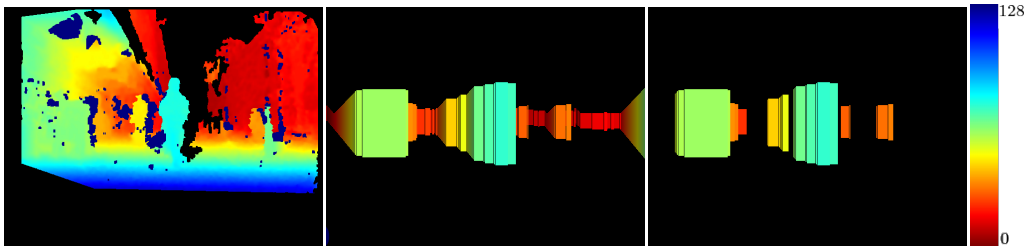


Figure 14. Comparison between the ground truth, stixels and reconstructed objects. The color code represents distance to the camera. (**a**) Ground truth; (**b**) Obtained stixels; (**c**) Reconstructed objects.

Table 1. Parameter configurations results.

	α_{SAD}	α_{hist}	α_{height}	
Configuration 1	1	-	0	Gunyel et al. [7]
Configuration 2	0.5	-	0.5	Gunyel et al. [7]
Configuration 3	1	0	0	Two-level tracking
Configuration 4	0.5	0	0.5	Two-level tracking
Configuration 5	0	1	0	Two-level tracking
Configuration 6	0	0.5	0.5	Two-level tracking
Configuration 7	-	-	-	Object tracking

4.3.1. Sequence Performance

Tracking capabilities are evaluated as per [7], using annotated obstacle bounding boxes as the ground truth. Each configuration evaluated predicts bounding box positions up to Δ frames in the future. For each frame, recall is evaluated using the intersection over the union metric. Figure 15 shows the recall vs. Δ frames evaluation starting from every frame in the video sequence.

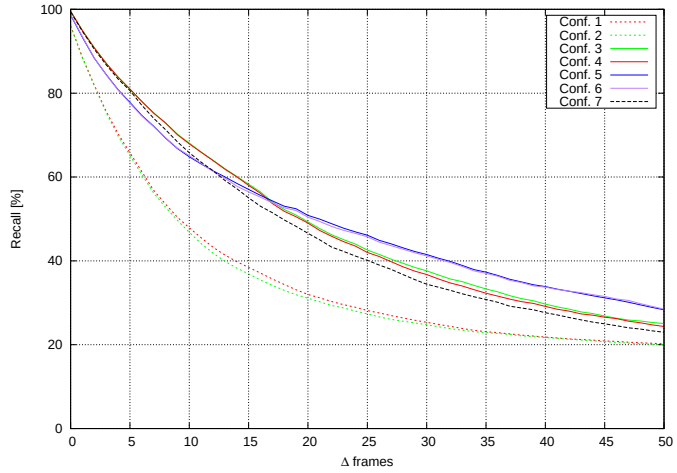


Figure 15. Recall obtained with different configurations.

Configurations 1 and 2 [7] fall quite fast, with a recall below 70% after just five frames. Furthermore, the α_{height} contribution is not clear. Two-level tracking methods yield better results,

especially when $\alpha_{hist} \neq 0$. The second tracking level filters much of the noise, making tracking more reliable. Figure 16 shows qualitative results for Configurations 1, 5 and 7. The trajectories obtained for Configuration 5 are the longest and smoothest, and the effect of avoiding multiple matches for the same stixel are also evident. In Configuration 1, the trajectories for many stixels start from the same single stixel. Configurations 5 and 6 use α_{hist} and Configurations 3 and 4 α_{SAD}. Histograms are normalized just before matching, while the sum of absolute differences is done pixel by pixel. This results in longer tracks in Configurations 5 and 6, since matching is more robust to illumination changes.

(a) **(b)** **(c)**

Figure 16. Stixel level tracking with Configurations 1, 5 and 7. (**a**) Configuration 1; (**b**) Configuration 5; (**c**) Configuration 7.

Object-based tracking (Configuration 7) shows good results for the first few frames, but it falls faster than two-level-based methods, since two-level tracking is more tolerant to clustering errors. If in one frame, a relatively large portion of the background is considered an obstacle, the histogram will change, and the matching score could be small. Figure 16 shows comparable quality tracks in Configurations 5 and 7, but 5 achieves longer tracks.

4.3.2. Performance at Different Frame Rates

In this section, we analyze recall as a function of Δ frames. The tests from the previous section are repeated, but now, the time step between frames is increased k frames each time, with $k = 0.06, \ldots, 1.2$ s (from 1 up to 20 fps, in a 15-fps video). Figure 17 shows recall versus time step increment. Four different profiles are detected involving Configurations 1 to 7. This tests also confirms that α_{height} is negligible. The most tolerant configuration is 7, since the object-level based tracking is able to handle slightly larger changes than stixel-level tracking.

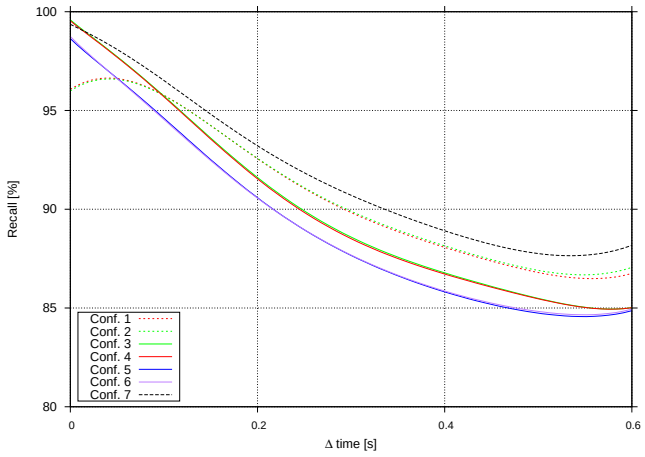


Figure 17. Recall of different configurations vs. frame rates.

Figure 18 compares recall, Δ frames and Δ time for Configurations 1, 3, 5 and 7. When Δ frames ≈ 0, the pattern shown in Figure 17 is repeated. However, when Δ frames starts to increase, Configurations 3 and 5 do not fall as fast as Configuration 1, which confirms the conclusions drawn from previous tests. Configuration 7 achieves a higher recall than the other configurations.

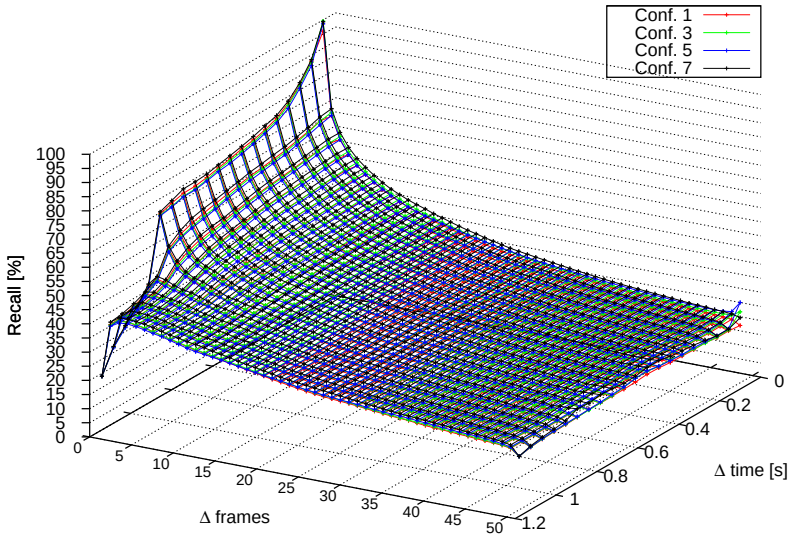


Figure 18. Tracking capabilities at different frame increments for Configurations 1, 5 and 7.

4.3.3. Performance with Other Sequences

In order to assess the performance of our algorithm in situations other than those found in the Bahnhof sequence, other sequences were processed, yielding the results described in this section. The sequences studied were Sunnyday, Jelmoli and Loewenplatz. The last one is quite interesting, since it was not obtained in a pedestrian area, the main focus of our application, meaning faster changes between frames. It will also allow us to ascertain how our algorithm behaves in an environment for which it was not originally designed.

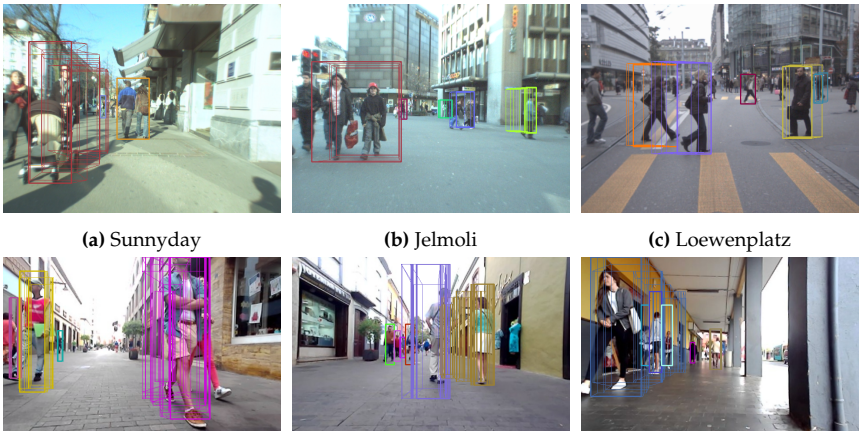
The algorithm was also tested in our own sequences, called Herradores, Carrera and Trinidad, which were taken in the areas in which the vehicle is expected to operate. Since there is no ground truth available for those sequences, only some examples of the output obtained are shown in this section.

In Figure 19, we can see that the algorithm is able to detect the pedestrians and follow them along their paths. Sequences Herradores and Carrera are quite challenging since the horizontal lines in the cobblestone can confuse the algorithm, but it was able to handle this with no apparent problems.

Figure 20 shows a comparison of the output obtained for those sequences for which a ground truth was available for Configurations 1, 3, 5 and 7. Configurations 2, 4 and 6 are not shown, both for clarity reasons and because, as previously shown, the differences between them and Configurations 1, 3 and 5 (respectively) are negligible.

Again, our method offers a clear improvement over the one presented in [7]. Furthermore, in Configurations 3 and 5, there is a noticeable improvement associated with our use of histogram comparisons for measurement. Note as well that in pedestrian environments, the behavior is similar. However, the use of the algorithm in vehicles exhibits worse behavior, since, as shown by the Loewenplatz sequence, the performance is significantly reduced. The main reason for this is the large changes in the images due to an increase in the vehicle's speed. This is confirmed by the results obtained for Configuration 7, the results of which are not as degraded as they were for the other configurations, for that sequence.

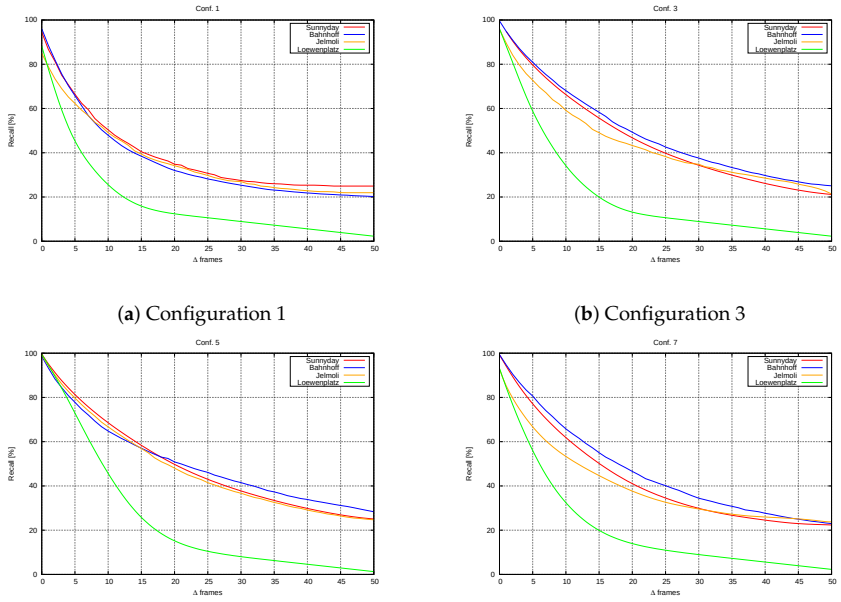
Sensors **2016**, *16*, 1182



(a) Sunnyday **(b)** Jelmoli **(c)** Loewenplatz

(d) Herradores **(e)** Carrera **(f)** Trinidad

Figure 19. Other sequences processed. The top row shows the output for three very well-known sequences. The bottom row shows the results for the sequences obtained by our vehicle, Verdino, in the environment in which it will operate. (**a**) Sequence Sunnyday from ETHZ dataset; (**b**) Sequence Jelmoli from ETHZ dataset; (**c**) Sequence Loewenplatz from ETHZ dataset; (**d**) Sequence Herradores obtained from prototype Verdino; (**e**) Sequence Carrera obtained from prototype Verdino; (**f**) Sequence Trinidad obtained from prototype Verdino.



(**a**) Configuration 1 (**b**) Configuration 3

(**c**) Configuration 5 (**d**) Configuration 7

Figure 20. Recall obtained for the sequences tested. Note that our algorithm outperforms that in [7] for every sequence, especially Configuration 5. (**a**) Recall obtained with Configuration 1; (**b**) Recall obtained with Configuration 3; (**c**) Recall obtained with Configuration 5; (**d**) Recall obtained with Configuration 7.

4.4. Computation Time

Figure 21 shows that the fastest configuration is 7. This is to be expected, since only object comparisons are involved and few obstacles are compared in each frame, vs. the 640×640 comparisons for the worst case in the stixel-level tracking. Figure 21 also shows that graph-based methods are slightly faster.

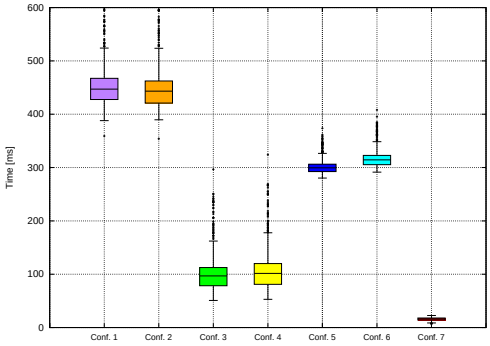


Figure 21. Times obtained for each configuration.

The algorithm was tested using Verdino's onboard computer, an i7-3770K processor with 16 Gb of RAM DDR-3 memory, SSD storage and an NVIDIA GeForce GT 640. Every method was implemented modularly using an Indigo ROS [48] Ubuntu-based distribution. The navigation method is able to work in real time for an autonomous vehicle, the implementation of the method is available at [49].

5. Conclusions

In this paper, we present an innovative object tracking method based on the stixel world [1] and applied to driver assistance. Our work expands and improves upon that presented in [7]. The use of a two-level tracking system offers robust stixel tracking, and the obstacle-based approach provides robustness, even at low frame rates. Once the output of the method is connected to the navigation subsystem through a layered costmap, it is ready to be used in our platform, Verdino, or in an autonomous car.

A simple, but effective clustering method based on stixels is introduced that yields a good detection rate. Moreover, we have shown how these clustered objects can be used as the basis for reconstructing the initial disparities, offering noticeable improvements and reducing the disparity error by almost one-half.

The results obtained by several configurations were evaluated. Two of them correspond to [7]; four of them present different parameter configurations for the two-level based approach; and a last configuration is based on the obstacle tracking approach.

The performance obtained along the sequence was measured in terms of recall, with the two-level-based method exhibiting better results than the others. The most important factor in the algorithm is α_{hist}, followed by α_{SAD}. The contribution from α_{height} is negligible. The obstacle-based approach does not seem to be a good choice when the frame rate is high, but it offers a good solution at lower frame rates, since it is more tolerant to large changes between images. It is also the fastest, making it a good choice to save computational resources.

The method works in real time using both variants, and it is fully integrated into Verdino, providing a fast, vision-based reconstruction of the environment. The method is ready to be used for navigation purposes in obstacle avoidance tasks. Videos demonstrating the effectiveness of the method in dense environments are also included.

Supplementary Materials: The following are available online at http://www.mdpi.com/1424-8220/16/8/1182/s1: Video S1: MethodPipeline.mp4. Video S2: stixelsNavigation.mp4.

Acknowledgments: This work was supported by Project STIRPE (Hacia un Sistema de Transporte Inteligente en Urbanizaciones y Recintos Peatonales) DPI2013-46897-C2-1-R and the funds from the Canary Islands Agency for Research, Innovation and the Information Society (ACIISI - Agencia Canaria de Investigación, Innovación y Sociedad de la Información), co-financed by FEDER funds (EU). The authors are grateful for the financial grant given to the Universidad de La Laguna by the Ministry of Economy, Industry, Commerce and Knowledge of the Government of the Canary Islands (Spain), 85% co-funded by European Social Funds.

Author Contributions: N. Morales, J. Toledo and L. Acosta conceived the method; N. Morales and J. Toledo implemented the method; N. Morales and L. Acosta designed the experiments; N. Morales, J. Toledo and A. Morell performed the experiments; N. Morales and A. Morell analyzed the data, and wrote the article.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ADAS	Advanced Driver Assistance Systems
LIDAR	Light-Detection And Ranging
ROI	Region Of Interest

References

1. Badino, H.; Franke, U.; Pfeiffer, D. The stixel world —A compact medium level representation of the 3D-world. In *Pattern Recognition*; Springer: Berlin/Heidelberg, Germany, 2009.
2. Benenson, R.; Mathias, M.; Timofte, R.; van Gool, L. Pedestrian detection at 100 frames per second. In Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA,16–21 June 2012; pp. 2903–2910.
3. Pfeiffer, D.; Franke, U. Efficient representation of traffic scenes by means of dynamic stixels. In Proceedings of the 2010 IEEE Intelligent Vehicles Symposium, San Diego, CA, USA, 21–24 June 2010; pp. 217–224.
4. VERDINO Research Project. Available online: http://verdino.webs.ull.es (accessed on 26 July 2016).
5. Morales, N.; Toledo, J.T.; Acosta, L.; Arnay, R. Real-time adaptive obstacle detection based on an image database. *Comput. Vis. Image Underst.* **2011**, *115*, 1273–1287.
6. Perea, D.; Hernandez-Aceituno, J.; Morell, A.; Toledo, J.; Hamilton, A.; Acosta, L. MCL with sensor fusion based on a weighting mechanism versus a particle generation approach. In Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), The Hague, The Netherlands, 6–9 October 2013; pp. 166–171.
7. Gunyel, B.; Benenson, R.; Timofte, R.; van Gool, L. Stixels motion estimation without optical flow computation. *Comput.Vis.* **2012**, *7577*, 528–539.
8. Sivaraman, S.; Trivedi, M.M. Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis. *IEEE Trans. Intell. Transp. Syst.* **2013**, *14*, 1773–1795.
9. Olivares-Mendez, M.A.; Sanchez-Lopez, J.L.; Jimenez, F.; Campoy, P.; Sajadi-Alamdari, S.A.; Voos, H. Vision-based steering control, speed assistance and localization for inner-city vehicles. *Sensors* **2016**, *16*, doi:10.3390/s16030362.
10. Keller, C.; Gavrila, D. Will the pedestrian cross? A study on pedestrian path prediction. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 494–506.
11. Flohr, F.; Dumitru-Guzu, M.; Kooij, J.; Gavrila, D. A probabilistic framework for joint pedestrian head and body orientation estimation. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 1872–1882.
12. Rabe, C.; Müller, T.; Wedel, A.; Franke, U. Dense, robust, and accurate motion field estimation from stereo image sequences in real-time. In *Computer Vision—ECCV 2010*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 582–595.
13. Barth, A.; Franke, U. Estimating the driving state of oncoming vehicles from a moving platform using stereo vision. *IEEE Trans. Intell. Transp. Syst.* **2009**, *10*, 560–571.
14. Danescu, R.; Pantilie, C.; Oniga, F.; Nedevschi, S. Particle grid tracking system stereovision based obstacle perception in driving environments. *IEEE Intell. Transp. Syst. Mag.* **2012**, *4*, 6–20.

15. Schauwecker, K.; Zell, A. Robust and efficient volumetric occupancy mapping with an application to stereo vision. In Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 31 May–7 June 2014; pp. 6102–6107.
16. Wurm, K.M.; Hornung, A.; Bennewitz, M.; Stachniss, C.; Burgard, W. OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems. In Proceedings of the ICRA 2010 Workshop on Best Practice in 3D Perception and Modeling for Mobile Manipulation, Anchorage, AK, USA, 7 May 2010.
17. Broggi, A.; Cattani, S.; Patander, M.; Sabbatelli, M.; Zani, P. A full-3D voxel-based dynamic obstacle detection for urban scenario using stereo vision. In Proceedings of the 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), The Hague, The Netherlands, 6–9 October 2013; pp. 71–76.
18. Yu, Y.; Li, J.; Guan, H.; Wang, C. Automated extraction of urban road facilities using mobile laser scanning data. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 2167–2181.
19. Fotiadis, E.P.; Garzón, M.; Barrientos, A. Human detection from a mobile robot using fusion of laser and vision information. *Sensors* **2013**, *13*, 11603–11635.
20. González, A.; Fang, Z.; Socarras, Y.; Serrat, J.; Vázquez, D.; Xu, J.; López, A.M. Pedestrian detection at day/night time with visible and FIR cameras: A comparison. *Sensors* **2016**, *16*, doi:10.3390/s16060820.
21. Zhang, L.; Li, Y.; Nevatia, R. Global data association for multi-object tracking using network flows. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Anchorage, AK, USA, 23–28 June 2008; pp. 1–8.
22. Henschel, R.; Leal-Taixé, L.; Rosenhahn, B. Efficient multiple people tracking using minimum cost arborescences. In *Pattern Recognition*; Springer International Publishing: Cham, Switzerland, 2014; pp. 265–276.
23. Li, W.; Song, D. Featureless motion vector-based simultaneous localization, planar surface extraction, and moving obstacle tracking. In *Algorithmic Foundations of Robotics XI*; Springer International Publishing: Cham, Switzerland, 2015; pp. 245–261.
24. Vatavu, A.; Danescu, R.; Nedevschi, S. Stereovision-based multiple object tracking in traffic scenarios using free-form obstacle delimiters and particle filters. *IEEE Trans. Intell. Transp. Syst.* **2015**, *16*, 498–511.
25. Kassir, M.M.; Palhang, M. A region based CAMShift tracking with a moving camera. In Proceedings of the 2014 Second RSI/ISM International Conference on Robotics and Mechatronics (ICRoM), Tehran , Iran, 15–17 October 2014; pp. 451–455.
26. Ess, A.; Leibe, B.; Schindler, K.; van Gool, L. Robust multiperson tracking from a mobile platform. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 1831–1846.
27. Leal-Taixé, L.; Pons-Moll, G.; Rosenhahn, B. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. In Proceedings of the2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, Spain, 6–13 November 2011; pp. 120–127.
28. Leal-Taixé, L.; Fenzi, M.; Kuznetsova, A.; Rosenhahn, B.; Savarese, S. Learning an image-based motion context for multiple people tracking. In Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Columbus, OH, USA, 23–28 June 2014; pp. 3542–3549.
29. Pellegrini, S.; Ess, A.; Schindler, K.; van Gool, L. You'll never walk alone: Modeling social behavior for multi-target tracking. In Proceedings of the 2009 IEEE 12th International Conference on Computer Vision 2009, Kyoto, Japan, 29 September–2 October 2009 ; pp. 261–268.
30. Davey, S.J.; Vu, H.X.; Arulampalam, S.; Fletcher, F.; Lim, C.C. Histogram probabilistic multi-hypothesis tracker with colour attributes. *IET Radar Sonar Navig.* **2015**, *9*, 999–1008 .
31. Bar-Shalom, Y.; Daum, F.; Huang, J. The probabilistic data association filter. *IEEE Control Syst.* **2009**, *29*, 82–100.
32. Scharwächter, T.; Enzweiler, M.; Franke, U.; Roth, S. Stixmantics: A medium-level model for real-time semantic scene understanding. In *Computer Vision—ECCV 2014*; Springer International Publishing: Cham, Switzerland, 2014; pp. 533–548.
33. Benenson, R.; Mathias, M.; Timofte, R.; van Gool, L. Fast stixel computation for fast pedestrian detection. In *Computer Vision—ECCV 2014*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 11–20.
34. Pfeiffer, D.; Franke, U.; Daimler, A.G. *Towards a Global Optimal Multi-Layer Stixel Representation of Dense 3D Data*; BMVA Press: Dundee, UK, 2011; pp. 1–12.

35. Pfeiffer, D.; Gehrig, S.; Schneider, N. Exploiting the power of stereo confidences. In Proceedings of the 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, 23–28 June 2013; pp. 297–304.
36. Muffert, M.; Milbich, T.; Pfeiffer, D.; Franke, U. May I enter the roundabout? A time-to-contact computation based on stereo-vision. In Proceedings of the 2012 IEEE Intelligent Vehicles Symposium, Alcala de Henares, Spain, 3–7 June 2012; pp. 565–570.
37. Benenson, R.; Timofte, R.; van Gool, L. Stixels estimation without depth map computation. In Proceedings of the 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops), Barcelona, Spain, 6–13 November 2011; pp. 2010–2017.
38. Chung, J.; Kannappan, P.; Ng, C.; Sahoo, P. Measures of distance between probability distributions. *J. Math. Anal. Appl.* **1989**, *138*, 280–292.
39. Edmonds, J. Paths, trees, and flowers. *Can. J. Math.* **1965**, *17*, 449–467.
40. Pollefeys, M.; Koch, R.; van Gool, L. A simple and efficient rectification method for general motion. In Proceedings of the Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20 –27 September 1999; pp. 496–501.
41. Luong, Q.T.; Faugeras, O.D. The fundamental matrix: Theory, algorithms, and stability analysis. *Int. J. Comput. Vis.* **1996**, *17*, 43–75.
42. Chu, K.; Lee, M.; Sunwoo, M. Local path planning for off-road autonomous driving with avoidance of static obstacles. *IEEE Trans. Intell. Transp. Syst.* **2012**, *13*, 1599–1616.
43. Arnay, R.; Morales, N.; Morell, A.; Hernandez-Aceituno, J.; Perea, D.; Toledo, J.T.; Hamilton, A.; Sanchez-Medina, J.J.; Acosta, L. Safe and reliable path planning for the autonomous vehicle verdino. *IEEE Intell. Transp. Syst. Mag.* **2016**, *8*, 22–32.
44. Morales, N.; Arnay, R.; Toledo, J.; Morell, A.; Acosta, L. Safe and reliable navigation in crowded unstructured pedestrian areas. *Eng. Appl. Art. Intell.* **2016**, *49*, 74–87.
45. Morales, N.; Toledo, J.; Acosta, L. Generating automatic road network definition files for unstructured areas using a multiclass support vector machine. *Inf. Sci.* **2016**, *329*, 105–124.
46. Morales, N.; Toledo, J.; Acosta, L. Path planning using a multiclass support vector machine. *Appl. Soft Comput.* **2016**, *43*, 498–509.
47. Lu, D.V.; Hershberger, D.; Smart, W.D. Layered costmaps for context-sensitive navigation. In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, 14–18 September 2014; pp. 709–715.
48. ROS.org. Available online: http://www.ros.org (accessed on 26 July 2016).
49. Stixel_world at GitHub. Available online: https://github.com/nestormh/stixel_world (accessed on 26 July 2016).

Este trabajo ha sido realizado gracias a los fondos de la Agencia Canaria de Investigación, Innovación y Sociedad de la Información (ACIISI), confinanciados por los fondos FEDER (EU).

Gobierno de Canarias
Agencia Canaria
de Investigación, Innovación
y Sociedad de la Información

Unión Europea
Fondo Social Europeo