Trabajo de Fin de Grado

Instrumento musical controlado mediante visión por computador

Computer vision controlled musical instrument

Fernando González López - Peñalver

7 de julio de 2015

D. **José Luis Sánchez de la Rosa**, con N.I.F. 42.080.654-S profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutor

D. **Francisco José Fumero Batista**, con N.I.F. 45.731.321-F profesor Titular de Universidad adscrito al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como cotutor

C E R T I F I C A (N)

Que la presente memoria titulada:
"Instrumento musical controlado mediante visión por computador"

ha sido realizada bajo su dirección por D. **Fernando González López - Peñalver**, con N.I.F. 78.855.165-X.
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firman la presente en La Laguna a 7 de Julio de 2015

Agradecimientos

Muchísimas gracias a mis padres y a mi hermana que siempre me apoyaron.

Gracias a Fran y José por animarme (y ayudarme) a presentar este proyecto.

Y gracias a José Luis por toda la ayuda e ideas aportadas.

Licencia

Resumen

En el presente documento se realiza una exposición de los elementos más importantes del Trabajo de Fin de Grado.

En este trabajo se trata la creación de un prototipo de instrumento musical electrónico cuya interfaz con el usuario ofrezca elementos que hagan uso de técnicas de visión por computador. En concreto se describe la implementación de un secuenciador musical con múltiples pistas. Las pistas, y las secuencias que generan son controladas mediante un tablero (físico) sobre el que se colocan una serie de piezas. Debido a la flexibilidad que ofrece el sistema, el concepto de tablero puede ser cualquier elemento que se capture con una cámara de vídeo en tiempo real. Con ejemplos que van desde una hoja de papel con una serie de casillas dibujadas, hasta el uso de las ventanas de un edificio a modo de pasos en la secuencia.

Los objetivos principales son la creación de un instrumento musical electrónico y el estudio de nuevas formas de interacción entre un músico y sus instrumentos musicales. Una de las partes más importantes en un instrumento musical es la interfaz que ofrece al usuario y el como ésta puede restringir o dar libertad en función de sus características.

Por esa razón el proyecto se divide en dos grandes partes: la síntesis de sonido y creación del secuenciador musical, y la creación de las herramientas necesarias para implementar la visión por computador. El documento recoge toda la información necesaria para comprender ambos campos de estudio y el trabajo llevado a cabo en cada uno de ellos.

Por último se expone el resultado obtenido, haciendo una descripción de su implementación, funcionalidades y restricciones. Se trata de una aplicación de escritorio que ofrece las herramientas para configuración, control y ejecución del instrumento (y sus elementos de visión por computador) a través de una interfaz gráfica.

Palabras clave: Secuenciador musical, síntesis de sonido, interfaz, visión por computador

Abstract

This document gathers the exposition of the most important elements in the Final Degree Project.

This paper is about the prototyping of an electronic instrument which offers some user interface elements that make use of computer vision techniques. Specifically the implementation of a multitrack music sequencer is described. This tracks, and the generated sequences are controlled by a (physical) board on which a number of pieces are placed. Due to the flexibility offered by the system, the concept of board can be anything that can be captured with a video camera in real time. With examples ranging from a piece of paper with a series of cells drawn, up to using the windows of a building as the steps of the sequence.

The main objectives are the creation of an electronic musical instrument and the study of new forms of interaction between a musician and his musical instruments. One of the most important parts of a musical instrument is the interface provided to the user and how it may restrict or not according to their characteristics.

For that reason the project is divided into two main parts: the synthesis of sound and music sequencer creation, and creating the necessary tool for implementing computer vision. The document contains all the information necessary to understand both field of study and the work carried out in each of them.

Finally the result is exposed, with a description of its implementation, functionality and restrictions. It's a desktop application that provides the tools for configuration, control and execution of the instrument (and its computer vision elements) through a graphical interface.

Keywords: Music sequencer, sound synthesis, interface, computer vision

Índice general

1. Introducción **10**
- 1.1. Introducción . . . 10
- 1.2. Estado del arte . . . 10
  - 1.2.1. Uso de la visión por computador en la música . . . 11
- 1.3. Importancia de la interacción en la composición e interpretación musical . . . 11
- 1.4. Propuesta propia . . . 12
- 1.5. Tecnología utilizada . . . 13
- 1.6. Fases del proyecto . . . 13

2. Objetivos **14**
- 2.1. Objetivos generales . . . 14
  - 2.1.1. Creación de un instrumento musical electrónico completo . . . 14
  - 2.1.2. Exploración de nuevas formas de control de instrumentos musicales electrónicos . . . 15
- 2.2. Objetivos específicos . . . 15
  - 2.2.1. Conocer mejor el funcionamiento e implementación de la síntesis de sonido . . . 15
  - 2.2.2. Creación de bibliotecas de apoyo a proyectos similares . . . 16

3. Sonido **17**
- 3.1. Síntesis de sonido . . . 17
  - 3.1.1. Introducción . . . 17
  - 3.1.2. Conceptos . . . 17
  - 3.1.3. Tipos de síntesis implementados: . . . 18
    - 3.1.3.1. Síntesis aditiva . . . 18
    - 3.1.3.2. Síntesis sustractiva . . . 19
  - 3.1.4. Otros tipos de síntesis . . . 19
    - 3.1.4.1. Síntesis granular . . . 19
    - 3.1.4.2. Síntesis por modulación de frecuencia . . . 20
- 3.2. CSound . . . 20
  - 3.2.1. Breve introducción a CSound . . . 20
    - 3.2.1.1. Conceptos básicos . . . 21
  - 3.2.2. API CSound . . . 22
    - 3.2.2.1. CSound6.NET . . . 22
- 3.3. Secuenciador musical . . . 22
  - 3.3.1. ¿Qué es un secuenciador musical? . . . 23
  - 3.3.2. Funcionamiento a alto nivel . . . 23

4. Visión por computador **24**
4.1. Introducción . . . 24
4.2. OpenCV . . . 25
4.2.1. EmguCV . . . 25
4.3. Técnicas utilizadas . . . 25
4.3.1. Filtrado por color discreto (Implementación de OpenCV) . . . 26
4.3.2. Filtrado por color probabilístico . . . 27
4.3.2.1. Biblioteca de selección de muestras . . . 28
4.3.3. Reconocimiento de las piezas . . . 28
4.3.4. Corrección de perspectiva . . . 29

5. Instrumento musical controlado mediante Visión por Computador **31**
5.1. El instrumento . . . 31
5.1.1. Funcionamiento . . . 31
5.1.1.1. Duración variable de cada paso . . . 32
5.1.2. Configuración . . . 32
5.2. Detalles de la implementación . . . 33
5.2.1. Breve aproximación a los UDOs implementados . . . 33
5.2.1.1. Sustractivo . . . 33
5.2.1.2. Aditivo . . . 35
5.2.1.3. Sampler multipista . . . 36
5.2.1.4. El secuenciador . . . 37
5.2.2. Diagrama de clases . . . 38
5.2.3. El flujo de los fotogramas . . . 41
5.2.3.1. Análisis de los pasos del tablero . . . 41
5.2.4. Control manual sobre los parámetros del dispositivo de captura . . . 42
5.3. Estado actual del instrumento . . . 43
5.3.1. Funcionalidades . . . 43
5.3.2. Restricciones . . . 46

6. Conclusiones y líneas futuras **47**

7. Summary and conclusions **49**

8. Presupuesto **50**

Índice de figuras

1.1. KORG MS-20 12

2.1. Secuenciador matricial 15

3.1. Estructura síntesis aditiva 18
3.2. Estructura síntesis sustractiva 19
3.3. Síntesis FM 20
3.4. CSound opcode 21
3.5. Secuenciador analógico 23

4.1. Filtrado de color discreto 26
4.2. HSV 27
4.3. Distribución normal RGB 27
4.4. Filtrado de color probabilístico 28
4.5. Procesado de imagen filtrada 29
4.6. Corrección de perspectiva 30
4.7. Corrección de perspectiva sobre el tablero 30

5.1. Diagrama de funcionamiento de sintetizador sustractivo 33
5.2. Overdrive / Clipping 35
5.3. ADSR 35
5.4. Diagrama de clases 39
5.5. Paso irregular ROI 42
5.6. Máscara de paso 42
5.7. Filtrado de paso irregular 42
5.8. Control parámetros cámara 43
5.9. Captura de pantalla 1 44
5.10. Captura de pantalla 2 45

Capítulo 1

Introducción

1.1. Introducción

Vivimos un momento en el que el acceso a la tecnología (y el avance de la misma) por parte de las personas es sin duda el mayor de toda la historia. Gracias al acceso a tecnologías avanzadas y a precios asequibles, el mundo de la música está en pleno auge. Existen gran cantidad de instrumentos, herramientas y acceso a información, lo que facilita y acerca la composición de música. Aún existen (y así seguirá siendo) gran variedad de instrumentos «tradicionales» (mucho más asequibles que años atrás) como guitarras, pianos, baterías y sus variantes eléctricas. Por otro lado la creación de instrumentos y herramientas cien por cien electrónicas, como los sintetizadores musicales y los efectos de pre y postprocesado del sonido han supuesto un gran avance desde mediados del siglo XX.

No ha sido hasta la última década, cuando el uso de los homólogos virtuales (de instrumentos y herramientas electrónicos), en forma de programas de ordenador, se ha incrementado notablemente. Esto se debe especialmente al fácil acceso a los mismos, a la gran cantidad de alternativas tanto gratuitas como de pago y el desarrollo de herramientas para su creación.

Las aplicaciones informáticas conocidas como Estaciones de trabajo de Audio Digital (EAD, o DAW[1] del inglés), son sistemas dedicados a la grabación, edición y producción de audio. Éstas han permitido, tanto a bandas como a artistas en solitario, producir sus obras personalmente, dado que se han reducido muchísimo los costes y los conocimientos técnicos requeridos. Esto sin duda repercute favorablemente en la existencia de, cada día, más y más artistas musicales que publican sus obras haciendo uso de herramientas online, lo que propicia su difusión a nivel global.

1.2. Estado del arte

Actualmente, en la búsqueda de nuevas formas de interacción persona-computador, el campo de la visión por computador cobra especial importancia. Mediante el uso de diferentes técnicas se pueden conseguir nuevas interfaces, como: seguimiento de ojos para control del puntero, identificación facial, identificación de códigos y patrones concretos, gestos, poses, ...

[1] Digital Audio Workstation

1.2.1. Uso de la visión por computador en la música

La aplicación de la visión por computador en el campo de la interacción con la música ha tenido una evolución dispar.

- Existe un instrumento musical comercial que hace un uso extensivo de la visión por computador para su control. Se trata de *Reactable*[2], un proyecto impulsado por el *Grupo de Tecnología Musical de la Universidad Pompeu Fabra de Barcelona*. Es un tablero retroproyectado, sobre la que se colocan una serie de piezas. Estas piezas tienen un patrón dibujado en la parte inferior. Dicho patrón es procesado por un sistema de visión por computador, que reconoce de que módulo se trata y su posición dentro del tablero. El funcionamiento de *Reactable* se basa en el uso de módulos que se conectan mediante el uso de gestos de la mano y por su proximidad en el tablero. *Reactable* ha tenido un éxito razonable y es utilizado por músicos en todo el mundo.
- Por otro lado existen infinidad de proyectos realizados «Ad hoc[3]». Un ejemplo es:
  - Un secuenciador musical en el que son las personas (con sus propios cuerpos) las que controlan la música[4]. Este proyecto, que (entre otros) ha servido de inspiración en el desarrollo, fue realizado por el estudio «Espada y SantaCruz»[5] en el año 2012.

  Ninguno de estos proyectos está excesivamente documentado y de muchos se desconocen las técnicas o tecnología utilizadas.

Por otro lado, en el ámbito académico existen algunas publicaciones que exploren en profundidad nuevos medias de interacción con la música. A continuación se exponen una serie de artículos que pueden resultar de interés:

- En primer lugar destacar el artículo «Electronic Music Interfaces» (Disponible en la bibliografía[CV1]), que realiza un recorrido por los distintos tipos de interfaces relacionadas con la música y las implementaciones de las mismas. Su contenido no se basa únicamente en la interacción mediante la visión por computador, pero ésta se describe en varias de sus secciones. Se realiza un análisis a interfaces percusivas, para instrumentos de cuerda (pulsada y frotada), para instrumentos de viento e incluso para la voz.
- El artículo «The Concept of a Visual Interface for Conducting Humans and Synthesizers» (también en la bibliografía [CV2]) muestra la creación de una batuta con la que controlar el tempo y la expresión de un instrumento. Para ello hace uso de técnicas, de visión por computador, para el reconocimiento de gestos hechos con dicha batuta en tiempo real.

1.3. Importancia de la interacción en la composición e interpretación musical

A la hora de crear un sintetizador de música es crucial tener muy en cuenta la interfaz que se va a ofrecer al usuario para su interacción con el mismo.

[2] http://www.reactable.com
[3] Que es apropiado, adecuado o especialmente dispuesto para un determinado fin.
[4] http://www.miguelespada.es/?p=1210
[5] http://www.espadaysantacruz.com/

De nada sirve tener una herramienta con innumerables características y funciones, si el uso de las mismas espera que el usuario recuerde complicadas combinaciones de botones, o que el acceso a éstas requiera de numerosos pasos (como puede ser un submenú, dentro de un menú, dentro de otro menú principal). Por otro lado, tampoco es conveniente ofrecer demasiado detalle sobre dichas funcionalidades, ya que se puede acabar con una interfaz demasiado confusa y de difícil comprensión.

Existen, en la historia de los sintetizadores de música, varios casos de sistemas que cayeron en estos errores; por ejemplo, el sintetizador semi-modular KORG MS-20 (Figura 1.1). Considerado aún hoy en día un buen sintetizador (tecnológicamente), muchas personas prefieren usar otros similares debido a lo confusa que es su matriz de programación. Con dicha matriz decidieron apartarse de los estándares de la época y ofrecer gran cantidad de funcionalidades, que generalmente no estaba de mano del usuario controlar. Algunos de estos controles, aunque funcionaban correctamente, no suelen utilizarse debido a su escasa utilidad o relevancia en el sonido generado.

Figura 1.1: KORG MS-20

Como con cualquier herramienta, el usuario espera poder utilizarla sin demasiadas molestias ni una configuración excesiva. Además, en procesos creativos como son la composición e interpretación musical, añadir niveles de complejidad innecesarios distraerá el foco de atención, y la concentración es crucial en el desarrollo de esas tareas.

1.4. Propuesta propia

Es importante destacar que la elaboración de este proyecto ha sido una propuesta personal, y surge a raíz de otro similar desarrollado durante el verano de 2013. En dicho proyecto participamos yo y otros tres miembros del equipo creativo del «Estudio veintiocho y medio[6]». Se trataba de una instalación interactiva en la que los usuarios controlaban una serie de secuencias musicales colocándose en unas marcas sobre el suelo (detectándolos mediante visión por computador) y haciendo uso de otros controles más tradicionales, como potenciómetros y botones.

Durante unas prácticas de la asignatura de Sistemas de Interacción Persona Computador, se comentó el trabajo con el profesor (Francisco J. Fumero Batista) y se mostró bastante interesado. En la siguiente clase práctica se mostró un método de filtrado de color que pareció muy interesante para implementar en dicho proyecto (el método se explica más adelante). Tras varias reuniones con Francisco, éste propuso presentar un proyecto similar como Trabajo de Fin de Grado. En principio se trataba de un secuenciador MIDI[7], es decir un

[6] http://veintiochoymedio.com/

[7] «[...] es un estándar tecnológico que describe un protocolo, una interface digital y conectores que permiten que varios instrumentos musicales electrónicos, computadoras y otros dispositivos relacionados se conecten y comuniquen entre si.» Wikipedia

dispositivo para secuenciar las notas de otros instrumentos mediante el envido de señales de control, incapaz de producir sonido por sí mismo.

Más adelante, tras el proceso de asignación de tutor para el proyecto, el tutor asignado (José Luis Sánchez de la Rosa) animó a implementar también la síntesis de audio, para así conseguir un producto más completo e interesante. Dado el interés personal en la síntesis de audio y la aparente sencillez de las herramientas necesarias, se procedió a incluirlo en el Trabajo de Fin de Grado. De esta forma se consigue un resultado mucho más completo y que no depende de otros instrumentos para la producción de sonido o secuenciación de los pasos.

1.5. Tecnología utilizada

Para el desarrollo de toda la parte de síntesis de sonido y programación de las secuencias se ha empleado CSound; se describe qué es y su funcionamiento básico en la sección 3.2 en la página 20.

Para el desarrollo de las técnicas de visión por computador se ha empleado la biblioteca OpenCV, haciendo uso del *wrapper* para .NET, EmguCV; se describen en la sección 4.2 en la página 25.

Para conseguir aunar la síntesis de audio en CSound y las técnicas de visión por computador en una interfaz de usuario agradable y fácil de implementar se ha optado por utilizar C# en el entorno de Windows Forms[8]. La experiencia previa con el lenguaje y sus peculiaridades fueron los factores decisivos a la hora elegirlo. Las facilidades que aporta el entorno de desarrollo .NET, como los contenedores de datos, el sistema de consultas LINQ y la extensa documentación que ofrece han sido cruciales a lo largo de todo el desarrollo.

1.6. Fases del proyecto

Dada su naturaleza, este proyecto fue dividido en dos fases principales de desarrollo. Una dedicada al sonido y otra a la visión por computador.

La primera fase consistió en desarrollar todo lo relacionado con la síntesis de sonido y la secuenciación musical. Durante las primeras semanas hubo que aprender a utilizar el lenguaje de programación CSound y cómo utilizarlo para obtener los resultados deseados. Durante esta fase se desecharon distintas implementaciones de la lógica del secuenciador y de las técnicas de síntesis empleadas. Esto ocurría cada vez que conseguía comprender correctamente algún aspecto nuevo de las herramientas utilizadas. Esta primera fase comenzó, aproximadamente, el día 22 de Febrero de 2015 y finalizó alrededor de un mes después.

La segunda fase consistió en desarrollar todas las herramientas y métodos necesarios del campo de la visión por computador. En concreto se desarrollaron las técnicas de filtrado por color y corrección de perspectiva descritas en la sección 4.3. También se desarrollaron las bibliotecas de apoyo necesarias, como las descritas en las secciones 4.3.2.1 y 5.2.4. Esta fase requirió aproximadamente un mes también.

Una vez se habían desarrollado, por separado, las dos ramas principales del proyecto, se pasó a crear el prototipo de instrumento musical, secuenciado, controlado mediante visión por computador. Se creó la interfaz gráfica con todo lo necesario para que el usuario configure y utilice el instrumento correctamente, y que será considerado como el resultado final de este proyecto. Durante esta fase se definieron nuevas funcionalidades a medida que se iba diseñando la interacción con el usuario.

[8] https://msdn.microsoft.com/es-es/library/dd30h2yb(v=vs.110).aspx

Capítulo 2

Objetivos

2.1. Objetivos generales

2.1.1. Creación de un instrumento musical electrónico completo

El objetivo principal, del Trabajo de Fin de Grado, es la creación de la primera versión funcional (o prototipo) de un instrumento musical apto tanto para la composición como para la interpretación de melodías y ritmos secuenciales. Además la peculiaridad de dicho instrumento musical es su interfaz principal con el usuario, que hace uso de técnicas de visión por computador para el control de la música creada. Se trata de un producto autocontenido, es decir, capaz de funcionar por sí mismo sin requerir de otros medios de control, o síntesis de audio, externos.

Es un instrumento musical secuenciado, esto significa que no se controla utilizando un teclado sino mediante el ajuste de una serie de melodías y ritmos que son reproducidos, de modo secuencial.

El usuario controla las características de esta secuencia, ajustando (entre otros) la longitud de cada uno de sus pasos, la nota que que hacen sonar y su duración, o cuáles son reproducidos y cuáles no. Para ello el usuario dispone de un tablero dividido en una serie de pasos. Sobre esos pasos el usuario coloca unas piezas que indican que pasos deben sonar, su duración (en función del tamaño de la pieza) y la pista a la que están asociadas (el color de la pieza). Dicho tablero es recogido en vídeo, en tiempo real, para analizar la posición de las piezas sobre el tablero.

Este instrumento ofrece las funcionalidades necesarias para su correcta utilización y funciona de forma confiable y repetible. Es importante que se trate de un producto fiable, pues la música, en muchos niveles, se basa en la precisión de la ejecución.

Se desea ofrecer al usuario una nueva forma de interactuar con un instrumento musical y ofrecer así nuevas posibilidades; evitando siempre el terminar con un sistema demasiado complejo o difícil de utilizar.

El usuario dispondrá en todo momento de las herramientas necesarias para definir un tablero, los pasos y las piezas de distintos colores y tamaños.

El funcionamiento detallado del instrumento se describe más adelante, en la sección 5.1 en la página 31.

2.1.2. Exploración de nuevas formas de control de instrumentos musicales electrónicos

Desde casi el comienzo de los instrumentos electrónicos, se han empleado los mismos métodos para el control de los mismos: El teclado para crear melodías y armonías, los controles rotatorios o deslizantes para establecer valores, botones para accionar eventos, ... todo ello montado sobre un panel con (casi) ninguna flexibilidad.

Con la creación de los sintetizadores de audio virtuales (creados mediante elementos software) se consiguió ganar en flexibilidad y posibilidades de configuración, ofreciendo más libertad para el usuario (o desarrollador) en la creación de sistemas más complejos. Aún así, se han seguido utilizando las versiones virtuales de los controladores clásicos (teclado, sliders, botones, ...), con lo que la interacción no ha variado demasiado.

Con este proyecto se espera ofrecer al usuario la libertad para crear su panel e incluso los elementos que se emplearán para su control. Se espera así ayudar a mejorar el proceso creativo posibilitando el uso de elementos y técnicas más allá de los nombrados anteriormente.

Existen ejemplos de dispositivos de interacción musical, que aún siendo realmente simples, han conseguido facilitar y acercar la composición musical a más gente. En concreto cabría mencionar los secuenciadores matriciales (Figura 2.1), que se popularizaron a partir del año 2000. Son dispositivos, físicos o virtuales, compuesto por una matriz de pulsadores, en los que se establece qué nota debe sonar (en el eje Y) y en que momento de la secuencia hacerlo (en el eje X).

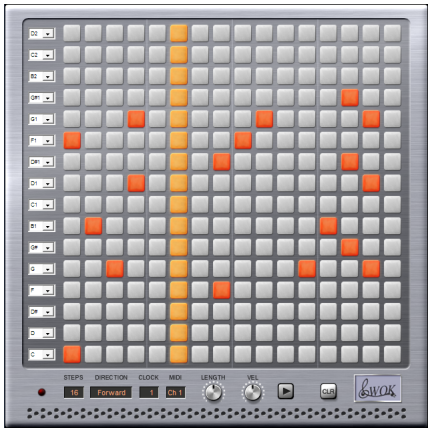

Figura 2.1: *Secuenciador matricial*

Productos como éste y otros similares han abierto nuevas posibilidades a la hora de la creación musical, lo que ha repercutido en el auge actual de la música electrónica en general, acercándola cada vez a más personas.

2.2. Objetivos específicos

2.2.1. Conocer mejor el funcionamiento e implementación de la síntesis de sonido

Debido a inquietudes personales, se puede decir que el proceso de síntesis sustractiva (sección 3.1.3.2) es razonablemente bien entendido por el autor de este proyecto, pero sólo había sido estudiado a nivel de hardware (sintetizadores físicos). Durante los últimos años se han ido adquiriendo suficientes conocimientos que permiten idear y construir pequeños circuitos electrónicos analógicos capaces de implementar las distintas funciones que

intervienen en esta técnica de síntesis de sonido (osciladores, filtros, amplificadores, generadores de envolvente, ...).

Como se ha comentado anteriormente, este proyecto inicialmente no iba a incluir la síntesis de sonido. Pero después de que se presentase la posibilidad de modelar sintetizadores musicales mucho más complejos, y empleando otras técnicas de síntesis, pareció una gran idea incluirla en el proyecto. La primera razón fue el interés personal en aprender la síntesis de sonido mediante software. Y la segunda es que de este modo se trataría de un instrumento musical completo e independiente, lo que hace al proyecto mucho más atractivo.

Gracias a este proyecto se ha profundizado en la síntesis sustractiva y se ha aprendido sobre otros muchos métodos de síntesis sonora (aunque no se implementasen todos).

2.2.2. Creación de bibliotecas de apoyo a proyectos similares

Como se comentó anteriormente, no existe casi ninguna documentación en el ámbito de la visión por computador en la música. Por ello, parece crucial ofrecer una serie de bibliotecas (desarrolladas en el proyecto) para facilitar la creación de proyectos similares (ya sean propios o de cualquier otra persona). Por esta razón, todo el código desarrollado (y su documentación) se publicará con una licencia abierta.

Las bibliotecas abarcan varios aspectos del proyecto como pueden ser:

- Ejecución y control de CSound en tiempo real desde otra aplicación. Facilita la carga de los ficheros necesarios (orquesta y partitura), la configuración de los parámetros del intérprete CSound y la creación, ejecución y control del hilo en el que ejecutarlo.
- Una herramienta para el ajuste de los parámetros de cámaras web; brillo, contraste, saturación, ... (Ver sección 5.2.4 en la página 42).
- Biblioteca con el método de filtrado (4.3.2) que expande los propuestos por OpenCV para ajustarlos mejor a las necesidades del proyecto. La biblioteca acepta cualquier espacio de color de los disponibles en OpenCV y métodos para realizar el filtrado síncrona o asíncronamente.
- Biblioteca de apoyo a la anterior, que facilita la obtención de muestras de color para utilizar en el filtrado (Ver sección 4.3.2.1 en la página 28).
- Biblioteca con una serie de clases que ayuden a unir ambos aspectos (los musicales con los elementos físicos detectados mediante visión por computador). Es en sí la biblioteca que contiene toda la lógica generada y explicada en la sección 5.2.2 en la página 38.

Capítulo 3

Sonido

3.1. Síntesis de sonido

3.1.1. Introducción

La síntesis de sonido es el proceso mediante el cual se obtienen sonidos a partir de medios no acústicos. Ya sea mediante variaciones en el voltaje de señales eléctricas (como en los sintetizadores analógicos) o haciendo uso de software especializado para la síntesis digital. Un sintetizador puede utilizarse para imitar otros instrumentos o en la creación de nuevos timbres.

Existen varios métodos empleados en la generación de una señal sonora. Entre las técnicas más populares se encuentran la síntesis sustractiva, la síntesis aditiva, síntesis por modulación de frecuencia o la síntesis basada en muestras.

3.1.2. Conceptos

Se recopilan primero los conceptos relacionados con la síntesis de sonido, para ayudar a la comprensión del resto del capítulo.

- **Tono**: es la sensación auditiva (o atributo psicológico) atribuida al sonido, que los distingue entre agudos y graves en función de su frecuencia. Un tono puro es el generado por una onda sinusoidal perfecta. En éste sólo se genera la frecuencia fundamental (y ningún armónico). Los tonos fundamentales no existen más allá de la teoría y todos los tonos que percibimos están compuestos por la superposición varias frecuencias de onda con distintas amplitudes y fases.

- **Timbre**: también conocido como color tonal o calidad tonal, es la cualidad que posee un sonido o tono que permite diferenciarlo de otro incluso cuando éste tenga la misma frecuencia y amplitud. El sonido emitido por un instrumento no es una vibración simple, sino una mezcla de señales cuyas frecuencias constituyen valores múltiplos de una fundamental, denominados armónicos. Por ejemplo, una nota La 440 Hz producida por una guitarra es distinta a la misma nota producida por una flauta. Ambas notas poseen la misma frecuencia fundamental, pero varían ampliamente en su estructura armónica. Esa variación armónica es lo que se percibe como el timbre.

- **LFO** del inglés, «Low Frequency Oscillator» u Oscilador de Baja Frecuencia: Se trata de un oscilador que genera señales a frecuencias (generalmente) por debajo de los 20 hercios. Estas frecuencias se encuentran

por debajo del límite de audición del oído humano. Por esta razón su función no es la de producir sonidos. Las señales que generan son utilizadas en la modulación de otros parámetros del instrumento. Suelen generar distintos tipos de ondas (seno, cuadrada, diente de sierra, ...). Algunas implementaciones permiten establecer la frecuencia de oscilación como fracción (1/1, 1/2, 1/3; ...) de una frecuencia base suministrada. Esto permite sincronizar el LFO con otros módulos.

3.1.3. Tipos de síntesis implementados:

En la elaboración de este proyecto se han implementado dos técnicas para la síntesis de sonido. A continuación se hará una breve explicación de las técnicas y se aondará más en detalles de la implementación en la sección 5.2.1 en la página 33.

3.1.3.1. Síntesis aditiva

Es una técnica con la que se crean timbres mediante la adición de ondas sinusoidales con distintas características (como se muestra en la Figura 3.1). Se basan en la teoría de Fourier, según la cual se puede representar cualquier función de onda mediante la suma de ondas simples (sinusoidales).

Los timbres estarán formados por una cantidad variable de armónicos que varían el tono en el tiempo con respecto a la frecuencia fundamental. Para dotar de mayor riqueza al sonido también se introducen variaciones con el tiempo en la amplitud y la fase de las ondas sinusoidales.

El timbre obtenido en este tipo de síntesis procede directamente de la adicción de numerosos osciladores sinusoidales o mediante el uso de tablas precomputadas con las funciones de ondas resultantes. El uso de estas tablas permite generar timbres realmente complejos y para los cuales se requerirían cientos de osciladores individualmente configurados y afinados, lo cual, requiere de la configuración de gran cantidad de parámetros. En la implementación del proyecto se utilizó una técnica en la que se generan tablas que almacenan el ratio de amplitudes y fases entre armónicos; se explica detalladamente en el apartado 5.2.1.2 en la página 35.

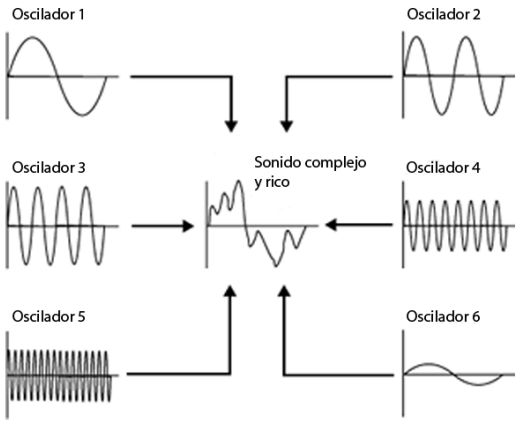

Figura 3.1: *Síntesis aditiva*

3.1.3.2. Síntesis sustractiva

Se trata de una técnica en la que determinadas partes de una señal rica en armónicos son atenuadas mediante el uso de filtros, con lo que se consigue alterar el timbre del sonido.

La base de esta técnica se encuentra en el uso osciladores capaces de generar señales ricas en armónicos. Por lo general se combinan varios de estos osciladores, con distintas formas de ondas, para dotar de mayor riqueza al sonido.

También es común el uso de generadores de ruido de diferentes clases (ruido blanco, rosa, marrón, ...). El objetivo es enriquecer el timbre, rellenando partes del espectro del sonido no cubiertas por las señales de los osciladores.

La última pieza clave en este tipo de síntesis son los filtros, que permiten filtrar las señales de audio para atenuar ciertas frecuencias, resaltando así otras. Por lo general se utilizan filtros paso bajo (atenúa frecuencias altas), paso alto (atenúa frecuencias bajas) y pasa banda (atenúa todo el espectro salvo la banda establecida). En la Figura 3.2 se muestran los elementos básicos y cómo afectan a las ondas y armónicos generados.

A todo esto se pueden sumar distintas formas de modular ciertos parámetros de los filtros y osciladores y la generación de envolventes, lo que permite la creación de timbres realmente complejos.

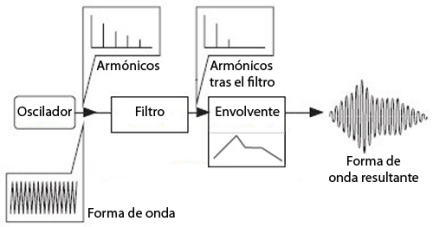

Figura 3.2: *Síntesis sustractiva*

3.1.4. Otros tipos de síntesis

Durante el desarrollo del proyecto se estudiaron otros tipos de síntesis de sonido que sería interesante explicar brevemente, debido a sus peculiaridades y/o funcionalidades.

3.1.4.1. Síntesis granular

En esta técnica una fuente de sonido, o forma de onda, es descompuesta en una serie de fragmentos (granos), generalmente de muy corta duración. Esos fragmentos son reestructurados, re-ordenados y reproducidos siguiendo una serie de patrones y métodos.

Existen dos atributos principales en este proceso:

- El primero es la duración de cada grano de sonido. Si la duración es muy corta (normalmente por debajo de los 0'02 segundos), se podrán distinguir menos características del sonido original. Cuanto mayor sea la duración, mejor se podrá distinguir el sonido o la forma de onda original.
- El segundo es la frecuencia a la que se generan cada uno de los granos. Si se producen a menos de 20 hercios (límite inferior del espectro audible por el humano) se percibirá como una serie de pulsaciones rítmicas. Al incrementar la frecuencia, los granos serán más dificiles de distinguir, percibiéndose un tono con cierto carácter de zumbido. La nota fundamental percibida será exactamente la de la frecuencia de generación de los granos.

3.1.4.2. Síntesis por modulación de frecuencia

Se basa en la utilización de una forma de onda simple (cuadrada, sinusoidal, triangular, ...) que es manipulada mediante la modulación de su frecuencia con una frecuencia de modulación que se encuentre también dentro del espectro audible.

En otras palabras, esta técnica distorsiona la frecuencia portadora de un oscilador modulándola con otra señal. Se puede observar el resultado, producto de la interacción entre la onda moduladora y la portadora, en la Figura 3.3.

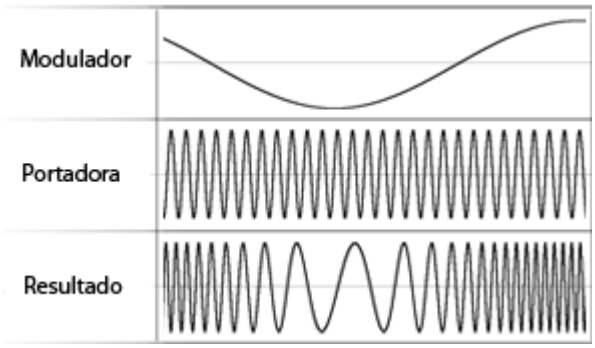

Figura 3.3: *Síntesis FM*

Este tipo de síntesis puede generar timbres inarmónicos [1] con interesantes características musicales.

Para generar sonidos armónicos, la señal de modulación tiene que tener relación armónica con la señal portadora.

A medida que aumenta la frecuencia de modulación, el sonido generado se vuelve progresivamente más complejo. Mediante el uso de frecuencias de modulación que no sean múltiplos de la fundamental (portadora), se pueden llegar a generar incluso sonidos percutidos.

3.2. CSound

CSound es un lenguaje de programación para la creación de sonido y música. Tiene las características del paradigma de «programación estructurada». Se llama CSound porque está escrito en C, al contrario que algunos de sus predecesores. Se encuentra bajo la licencia LGPL, por lo que se considera software libre.

3.2.1. Breve introducción a CSound

Originalmente el lenguaje se desarrolló durante 1985 en el *Instituto de Tecnología de Massachusetts* (MIT) por *Barry Vercoe* y se basaba en un sistema anterior llamado «Music 11». Desde entonces el lenguaje ha ido creciendo sin parar y actualmente se encuentra en su versión seis (6) con más de 1700 unidades generadoras (u opcodes).

Se trata de un lenguaje de programación no visual (como sí lo son otros similares) que hace uso de opcodes que son tratados como módulos que deben conectarse entre sí o a los que se les debe proporcionar cierta información para que produzcan un resultado deseado.

[1] Grado en el que los armónicos producidos ocurren en múltiplos impares de la frecuencia fundamental.

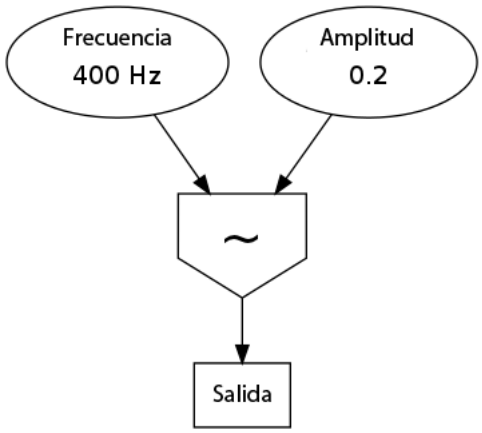

Figura 3.4: *Opcode CSound*

En la Figura 3.4 se puede observar como a un módulo oscilador se le proporcionan los valores de frecuencia y amplitud para obtener una forma de onda con dichas características en la salida. Conectando entre sí los distintos módulos y con ciertas estructuras de control, se consiguen desarrollar sistemas musicales.

La instrucción CSound equivalente a la Figura 3.4 sería una similar a «*salida* OSCIL *frecuencia, amplitud*», siendo **OSCIL** el opcode que implementa la funcionalidad de un oscilador.

Posee una extensa biblioteca estándar en la que se incluyen opcodes para la generación de señales, enrutamiento de señales de audio o de control, modificadores de señales, control de instrumentos, operaciones matemáticas y sobre tablas, distintos tipos de conversores de unidades y análisis espectral entre muchos otros.

Se trata de un lenguaje interpretado, con lo que necesita del intérprete de CSound para poder funcionar. Este intérprete primero compila y analiza el programa antes de comenzar la ejecución del mismo.

3.2.1.1. Conceptos básicos

A continuación se exponen los conceptos básicos para comprender las explicaciones dadas sobre la implementación realizada en CSound.

- **Instrumento**: Es un bloque de código que contiene la lógica deseada para producir un sonido (o cualquier otro tipo de señal de control). Un instrumento recibe parámetros desde la partitura y éstos definen cuando tocar el instrumento, durante cuanto tiempo y cualquier otra serie de parámetros que requiera (tono, envolvente, ...).
- **Orquesta**: Es la sección de un programa CSound en la que se definen los instrumentos. Además también es donde se establece la configuración de CSound a utilizar, como el dispositivo de salida de audio, dispositivos MIDI y otros parámetros necesarios (frecuencia de muestreo de audio, mono o estéreo, ...).
- **Partitura**: Es la sección que contiene las instrucciones de cuando y como encender y apagar qué instrumentos de una determinada orquesta. Una partitura está compuesta de líneas de texto, en la que cada una controla a un instrumento en un tiempo dado. Por ejemplo «i 1 0 2» encenderá el instrumento uno durante dos segundos desde el segundo cero de la partitura.

- **Opcode**: Pequeñas máquinas que hacen un trabajo y que son programadas mediante sus conexiones. Por lo general un opcode tendrá parámetros de entrada y de salida. Están escritos en el lenguaje de programación C. CSound permite a los desarrolladores crear opcodes en C para extender su funcionalidad. Para ello se ofrecen las bibliotecas y documentación necesarias.
- **User Defined Opcode (UDO)**: Es una característica del lenguaje, que permite al programador definir sus propios opcodes. Con ésta, un usuario puede crear sus propios opcodes desde dentro del propio CSound (estando limitado al uso de los opcodes y la semántica definidos por el propio CSound). Esta característica es de gran utilidad de cara a la modularidad y reutilización del código generado. Una vez se ha definido un UDO, puede ser invocado y repetido todas las veces que sea necesario siendo necesaria una única línea de código cada vez.
- **Tabla:** Es un lugar donde almacenar datos de forma ordenada. Cada tabla tiene un tamaño y son indexadas empezando a contar por cero (0). Son utilizadas extensivamente por el lenguaje, usándose para almacenar desde las notas a interpretar (tabla de poca longitud) hasta una colección de todas las muestras de un fichero de audio digital (con potencialmente, cientos de miles o millones de elementos, si se tiene en cuenta que la frecuencia de muestreo, de audio, estándar es de 44100 muestras por segundo).

3.2.2. API CSound

CSound ofrece una interfaz de programación de aplicaciones (API) que permite el control de una instancia de CSound mediante una serie de diferentes funciones, lo que hace posible su integración CSound en cualquier tipo de aplicación. Está escrita en C, pero existen adaptaciones o *wrappers* para muchos lenguajes de programación, como C++, Python, Java, C#, ...

La API ofrece métodos para configurar e instanciar CSound y su posterior control en tiempo real. Se ofrece la funcionalidad para crear una serie de canales software que permiten la comunicación entre los instrumentos de dicha instancia y el exterior (otra aplicación). Estos canales permiten la transmisión tanto de señales de control como de audio. Además ofrece acceso directo, tanto lectura como escritura, a las tablas y variables globales de la instancia en funcionamiento.

3.2.2.1. CSound6.NET

Se trata de un *wrapper* para la API de CSound (desde la versión 6), haciéndola disponible a programas del entrono .NET escritos C#.

Ofrece acceso directo a todas las características de la API original y además extiende la funcionalidad de la misma ofreciendo clases y métodos de apoyo, que nos abstraen del uso de punteros no manejados y otros aspectos de más bajo nivel.

Hace uso de las funcionalidades del lenguaje de programación C#, como los eventos, delegados, excepciones, programación asíncrona, genéricos y el bloque «using».

Para más información consultar el sitio del proyecto: https://csound6net.codeplex.com

3.3. Secuenciador musical

Al tratarse de un instrumento musical que funciona de modo secuencial, se describen a continuación una serie de conceptos relacionados con los secuenciadores musicales.

3.3.1. ¿Qué es un secuenciador musical?

Un secuenciador musical es un dispositivo físico (como en la Figura 3.5), o componente software, en el que se pueden grabar, editar y reproducir música mediante el manejo individual de las notas o señales de control. Es un generador de señales, que produce una serie ordenada de valores ajustables y en secuencia cíclica. Dichos valores son generalmente utilizados para establecer la frecuencia fundamental de un sonido a generar, consiguiendo así crear una secuencia de notas que se repite una y otra vez. En otras ocasiones, dichos valores son utilizados para modular ciertos parámetros de algún módulo dentro de un sintetizador (como por ejemplo la frecuencia de corte de un filtro).

Figura 3.5: *Secuenciador analógico*

3.3.2. Funcionamiento a alto nivel

Un secuenciador musical ofrece al usuario una secuencia de pasos y para cada uno de éstos una serie de controles (generalmente controles de ajuste rotatorio) que ajustan los parámetros asociados con cada paso. Al ponerse en marcha, el secuenciador generará los valores establecidos para el primer paso y continuará haciendo lo mismo con el resto de los pasos, a una velocidad establecida por el usuario (tempo). De esta forma, todos los pasos tienen la misma duración, y así ocurre generalmente. Una vez se alcance el último paso se volverá al primero, creando así una secuencia cíclica. Por lo general los secuenciadores (físicos o virtuales) ofrecen al usuario las siguientes prestaciones:

- Ajuste de la longitud de la secuencia, teniendo en cuenta la limitación inherente a un dispositivo físico al que no se le pueden añadir más pasos al panel de control. En el caso de un secuenciador software la longitud puede ser arbitrariamente larga.
- Múltiples pistas en las que el avance de los pasos es simultaneo, es decir, todas las pistas reproducen el mismo paso en cada momento. Con esto se podrían generar melodías en varios instrumentos al mismo tiempo (uno por cada pista), o generar una melodía y usar el resto de pistas para modular alguno de los parámetros del instrumento.
- Ajuste del tempo de reproducción de la secuencia.
- Distintos modos de reproducción: Hacia arriba (del paso 1 al último), hacia abajo (del último paso al primero) o seleccionando aleatoriamente un paso cada vez.

Capítulo 4

Visión por computador

4.1. Introducción

Es el campo en el que se incluyen los métodos para adquirir, procesar, analizar y entender imágenes, y más en general, datos de alta dimensionalidad del mundo real, para producir información útil.

La visión por computador es empleada en el control de procesos, en la navegación, en la detección de eventos, en la organización de información, en el modelado de objetos y en la interacción e interactividad. Para todas estas aplicaciones, la visión por computador se basa en una serie de tareas básicas:

- **Reconocimiento**: Que se trata de uno de los problemas clásicos de la visión por computador. Es la capacidad de reconocer objetos específicos o alguna de sus características. Existen diferentes variantes dentro de esta tarea, como pueden ser la identificación, detección y reconocimiento de objetos, la detección de ciertas poses u orientaciones de objetos o personas, el reconocimiento de caracteres óptico (OCR[1]) o el reconocimiento facial, entre otros.
- **Análisis de movimiento**: Con tareas encargadas de estimar el movimiento relativo de objetos o características en una secuencia temporal de imágenes. Entre estas tareas se incluyen el seguimiento (o «tracking» en inglés), que analiza el movimiento de un pequeño conjunto de características de interés en una secuencia de imágenes. Por ejemplo el carril ocupado por un vehículo en una carretera a medida que avanza por ésta.
- **Reconstrucción de una escena**: Que intenta realizar una reconstrucción 3D de una escena a partir de una serie de imágenes (o un vídeo) de la escena original.
- **Restauración de imágenes**: Con lo que se intenta reducir el ruido de imágenes. Esto se realiza mediante filtros y otras técnicas. Es de empleada, de algún modo, en todos los sistemas electrónicos de fotografía, pues ayudan a mejorar el rendimiento de los mismos. También se enmarcan dentro de estas tareas los procesos de corrección automáticos de color o perspectiva.

[1] Optical Character Recognition

4.2. OpenCV

OpenCV es una biblioteca que ofrece un conjunto de funciones orientadas principalmente a la visión por computador en tiempo real. Es distribuida bajo la licencia BSD, por lo que se posee libertad tanto para usos académicos como comerciales.

Posee interfaces para los lenguajes de programación C, C++, Python y Java y soporta Windows, Linux, Mac OS, iOS y Android.

Está escrita en C/C++ y puede hacer uso de las capacidades multi-núcleo de las CPUs modernas. Además una parte de su funcionalidad (enfocada a trabajo con matrices de gran tamaño) es capaz de hacer uso de la capacidad de proceso de las tarjetas gráficas, mediante el uso de OpenCL[2]. La arquitectura de las tarjetas gráficas (y sobre todo las más modernas) está optimizada para los cálculos en coma flotante y el procesado en paralelo.

La primera versión de OpenCV vio la luz en Junio del año 2000, y hasta la actualidad sigue evolucionando y creciendo. Con una extensa comunidad «online», y una excelente documentación, no es de extrañar que tenga registradas más de nueve millones de descargas.

Para más información acudir al sitio web del proyecto: http://opencv.org.

4.2.1. EmguCV

EmguCV es un *wrarpper* multi-plataforma de la biblioteca de procesado de imágenes OpenCV. Permite el acceso a las funciones y estructuras de datos que ofrece OpenCV desde el entorno .NET, siendo compatible con los lenguajes C#, Visual Basic, Visual C++, IronPython, ... Re-implementa gran parte de las clases y otras estructuras de datos, para adaptarlo a las funcionalidades ofrecidas por el entorno .NET. Aún así cabe destacar que ofrece acceso directo a OpenCV haciendo uso de la *invocación de plataforma*[3], permitiendo hacer uso de todo el potencial que ofrece.

Al igual que OpenCV, es capaz de funcionar en diversos sistemas operativos, a los que añade el soporte para Windows Phone.

Se encuentra licenciado bajo LGPL, pero permite la creación de productos comerciales fuera de las licencias libres mediante la compra de una licencia de uso comercial.

Para más información acudir al sitio web del proyecto: http://www.emgu.com.

4.3. Técnicas utilizadas

Debido a las características del proyecto, las tareas a realizar, con respecto a la visión por computador, son en su mayoría de la rama del reconocimiento. Debemos reconocer las piezas empleadas por el usuario.

Para el correcto funcionamiento del sistema necesita ser capaz de reconocer las piezas sobre un tablero ya configurado. La estrategia que se ha optado por utilizar es la de hacer un filtrado por color de la imagen recibida. De esta forma, al disponer de piezas de distintos colores, podemos aislarlas filtrando una imagen, permitiendo pasar únicamente los píxeles del color deseado.

Las complicaciones con este método aparecen generalmente, debido al hecho de que un objeto, de color aparentemente uniforme, realmente está compuesto por píxeles con pequeñas variaciones tonales; es decir, no podemos utilizar un valor absoluto para realizar el filtrado y esperar obtener el resultado correcto. Es la

[2] https://es.wikipedia.org/wiki/OpenCL

[3] «Los servicios de invocación de plataforma (PInvoke) permiten llamar desde código administrado a funciones no administradas e implementadas en una DLL.». Más información: https://msdn.microsoft.com/es-es/library/aa288468(v=vs.71).aspx

consecuencia de las pequeñas variaciones de luz, el propio material del objeto e incluso el ruido generado por el dispositivo de captura.

A continuación se exponen dos aproximaciones para tratar de reducir el impacto de este problema: una es la opción ofrecida por OpenCV y la otra se basa en los conocimientos adquiridos durante las prácticas de la asignatura mencionada anteriormente.

4.3.1. Filtrado por color discreto (Implementación de OpenCV)

El mecanismo que propone OpenCV para realizar un filtrado por color pasa por el uso de una de sus funciones para trabajar sobre arrays. Esto es así puesto que en OpenCV las imágenes son tratadas como arrays bidimensionales, en los que cada elemento es del tamaño del espacio de color empleado. Por ejemplo un elemento/píxel de una imagen en escala de grises está representado por un único valor (la luminosidad) y en una imagen RGB[4] cada píxel tiene tres valores, uno por color.

En concreto, se trata de la función «inRange» la que se encarga de comprobar si cada uno de los píxeles, de una imagen, se encuentra dentro de un rango de valores establecido. De esta forma para filtrar las piezas rojas de la Figura 4.1, se captura el color de uno de sus píxeles y se establece el rango de valores a utilizar. Es decir se deben escoger otras dos combinaciones de valores RGB (o el espacio de color utilizado) entre las que se encuentre el rango de colores que tienen las piezas.

Este proceso supone un problema de cara a la usabilidad, pues establecer correctamente dichos valores puede resultar complicado y monótono.

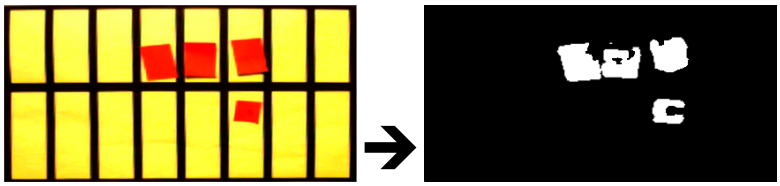

Figura 4.1: *Filtrado de color discreto*

La función devuelve una imagen binaria en la que un píxel es establecido a uno (blanco en la imagen) si se encuentra dentro del rango especificado y a cero (negro en la imagen) en caso contrario. Debido a esta característica, las imágenes obtenidas pueden no mostrar la forma ni el tamaño correctos de las piezas que se quieren filtrar, tal y como se observa en la parte derecha de la Figura 4.1. Una posible solución, a este último problema, es establecer un rango de colores más amplio para que se incluyan los valores que se han podido quedar cerca de los límites establecidos, pero generalmente eso repercute en un incremento del ruido filtrado, pues se pueden empezar a filtrar partes de la imagen no deseadas. Ese ruido procede de tonalidades parecidas lógicamente (a nivel algorítmico), pero no tanto para el ojo humano.

Cabe destacar que se puede mejorar la eficacia de este método si se trabaja con el espacio de color HSV[5] debido a que divide la imagen en tonalidad, saturación y brillo. Al trabajar con tonalidades resulta más sencillo establecer los rangos de colores, pues se centrará sobre todo en ajustar el valor asociado a la tonalidad.

En este caso hay que tener en cuenta el modo en el que se codifica dicho valor de tonalidad (H). Como puede observarse en la Figura 4.2, el tono es establecido como el grado de un ángulo cuyos valores posibles van de 0^{o} a 360^{o} (que es la base del cono mostrado). Es importante hacer notar que las tonalidades rojas se

[4] Red, Green, Blue / Rojo, Verde, Azul
[5] Hue, Saturation, Value

encuentran tanto próximas a los 0^{o} como a los 360^{o} (como muestra la flecha circular). Si se emplea la función «inRange» habrá que implementar correctamente el modo de establecer el rango de valores a utilizar, pues pueden desecharse píxeles que realmente son necesarios.

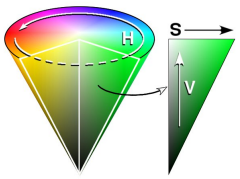

Figura 4.2: *HSV*

4.3.2. Filtrado por color probabilístico

Este método se explicó en una clase práctica y como se comentó anteriormente, pareció lo suficientemente interesante y potente como para ser implementado. Además los resultados generados son mucho más convenientes que los obtenidos con el método anterior, por lo que éste es el filtrado de color que se utiliza finalmente en el proyecto.

Se trata de construir una distribución normal multivariada (de tamaño tres), en la que cada una de sus dimensiones correspondan a uno de los canales de color empleados (en nuestro caso RGB), como se muestra en la Figura 4.3. Para ello se debe obtener una muestra (o colección de muestras), que no son más que pequeñas secciones de la imagen que contengan solamente píxeles del rango de color que deseamos filtrar. Con esos datos se realiza un ajuste de los mismos a la susodicha distribución normal. De esta forma obtenemos una herramienta estadística que nos permite realizar predicciones (inferencia) sobre la probabilidad que tiene un píxel de ser clasificado como parte de la muestra. Dicho de otra forma, nos permite comprobar, mediante el uso de valores continuos (probabilidad, de cero a uno), lo similar que es cada píxel de una imagen a los proporcionados como muestra.

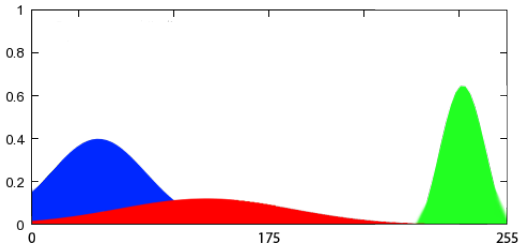

Figura 4.3: *Distribución normal RGB*

Con este método obtenemos la media para cada uno de los canales de color, lo que resulta en el color medio de las muestras utilizadas. Además se obtiene una matriz de covarianzas que relaciona entre sí los distintos canales de color. Gracias a esta información somos capaces de generar una imagen, en este caso, en escala de grises, en la que el valor de cada píxel es el devuelto por la función de densidad de nuestra distribución. Es decir, un valor entre cero (0) y uno (1), en función de su probabilidad.

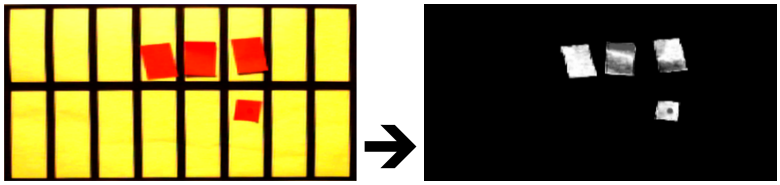

Figura 4.4: *Filtrado de color probabilístico*

Puede observarse en la Figura 4.4, que la imagen obtenida (a partir de la misma que con el método discreto) muestra de un modo más claro las piezas sobre el tablero, sobre todo su tamaño y formas originales. Este método nos permite una configuración más rápida por parte del usuario (usando 4.3.2.1) y nos proporciona unos resultados más fiables de cara al mismo.

En este caso, y aunque la biblioteca soporta el uso de cualquier espacio de color, se recomienda el uso de RGB o uno en el que el la importancia (subjetiva) de cada canal de color sea la misma. Para el uso que se desea hacer, en este proyecto, del filtrado de color, utilizar un espacio de color como HSV puede repercutir negativamente, ya que la información más relevante se encontrará en el canal que represente la tonalidad y no tanto en los que representen la saturación o el brillo (ya que justamente queremos contrarrestar las variaciones en esos dos últimos valores). Una posible solución sería el asignar pesos a las dimensiones de la distribución normal, pero eso una vez más introduce un paso (y posible problema) en el proceso de filtrado por color.

Para todos los cálculos estadísticos relacionados, se ha hecho uso de la biblioteca «Accord Statistics», que forma parte del framework Accord.NET. Ofrece una gran variedad de clases y herramientas para la realización de cálculos estadísticos. Además utiliza algoritmos matemáticos optimizados para reducir el consumo de recursos (CPU sobre todo). Está publicada bajo la licencia LGPL, con lo que también es considerada software libre.

Para más información consultar su sitio web: http://accord-framework.net

4.3.2.1. Biblioteca de selección de muestras

Como apoyo al método probabilístico de filtrado por color, debía implementarse la herramienta necesaria para la recolección de las muestras de una imagen en tiempo de ejecución. Para ello se creó una biblioteca con la que se pueden extraer partes de una imagen mostrada en una interfaz gráfica de Windows Forms.

Esta herramienta permite al usuario crear rectángulos sobre la imagen. Una vez completado ese proceso, la herramienta devuelve un listado de muestras con lo contenido dentro de cada uno de los rectángulos. Esos datos son aglutinados y reorganizados para realizar el ajuste a nuestra distribución normal.

Se ha demostrado la utilidad de esta herramienta durante el uso práctico del sistema, ya que permite establecer rápidamente el color asignado a cada una de las pistas del instrumento. Durante el proceso de calibrado es conveniente colocar una pieza a cada extremo del tablero y obtener las muestras de cada una de ellas. De esta forma nos aseguramos de incluir en nuestro ajuste las posibles variaciones de luminosidad que pueden darse desde un extremo a otro del tablero.

4.3.3. Reconocimiento de las piezas

Una vez hemos filtrado la imagen utilizando el color deseado, necesitamos ser capaces de distinguir el número de elementos detectados, su tamaño y posición dentro de la imagen. Para ello la imagen obtenida (en escala de grises) es procesada de nuevo con el fin de extraer la información:

- En primer lugar la imagen es desenfocada ligeramente con el fin de suavizar y homogeneizar la distribución de los píxeles.
- Seguidamente se establecen a uno (1) todos los valores superiores a cero (0) en la imagen anterior (desenfocada). Obtenemos de este modo una imagen binaria, en la que el color blanco representa el objeto detectado. Gracias al método de selección de muestras y a la precisión del filtrado de color probabilístico, la existencia de ruido (píxeles blancos que no deberían estarlo) es casi nula.

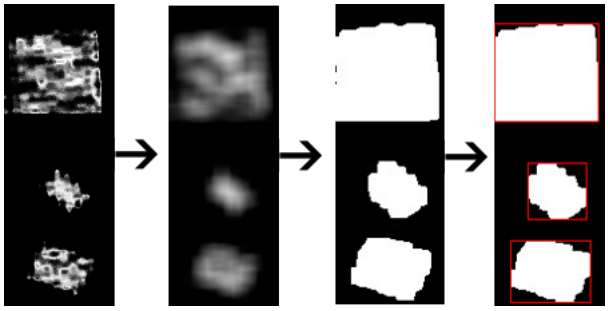

Figura 4.5: *Procesado de imagen filtrada*

- A continuación se realiza una búsqueda de contornos en la imagen, utilizando las herramientas proporcionadas por OpenCV.
- Como no importa la forma en sí de la figura, sino su tamaño y posición, a cada contorno se lo encierra en un rectángulo («bounding box»). Este rectángulo se construye tomando los valores máximos y mínimos para X e Y de cada objeto detectado. De esta forma obtenemos lecturas mucho más consistentes sobre el área y la posición de las piezas, al deshacernos de las variaciones que suelen crearse al borde de las mismas (mezcla de colores y cambios de luminosidad) de un fotograma a otro. Cabe destacar que, con esta técnica, una pieza colocada diagonalmente (como la pieza central en la Figura 4.5) generará un área mayor. Por esa razón las piezas tendrán que ser lo suficientemente diferentes en tamaño para evitar lecturas erróneas.
- Para calcular la posición de las piezas se utiliza el centro de masa del rectángulo generado, el cual es mucho más eficiente de calcular que el de un polígono formado por cualquiera de los contornos detectados (pues el primero tiene cuatro vértices y es un polígono regular y el otro decenas de vértices y generalmente irregular).

El siguiente paso es analizar los resultados obtenidos para asociar las piezas con el secuenciador de música. Este proceso se explica, más adelante, en la sección 5.1.1.

4.3.4. Corrección de perspectiva

Dada la naturaleza del proyecto, por lo general, sólo necesitaremos una sección de la imagen obtenida por la cámara web; aquella en la que se encuentra el tablero. Además, para dotar de mayor flexibilidad al sistema, se debe otorgar cierto grado de libertad a la hora de colocar la cámara con respecto al tablero. Sería un gran inconveniente tener colocar la cámara siempre en la misma posición, ángulo y distancia del tablero; algo que

seguramente requeriría de la construcción de una estructura física y por ende haría del sistema algo más rígido y poco transportable.

Afortunadamente OpenCV dispone de las herramientas necesarias para realizar una corrección de perspectiva, a los fotogramas capturados, en tiempo real.

Primero se establece un área mediante un polígono de cuatro vértices (a lo contenido en ese polígono se le realizará la transformación) y proporcionar otro polígono de cuatro vértices sobre el que proyectar el primero; como puede observarse en la Figura 4.6.

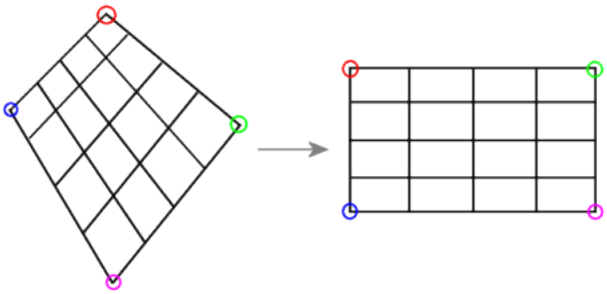

Figura 4.6: *Corrección de perspectiva*

En nuestro caso, se realiza una proyección en un rectángulo de las mismas dimensiones del fotograma. Con esto se consigue que el tablero sea lo único que se muestre en los fotogramas. Al ocupar el tablero la totalidad del fotograma, una vez aplicada esta técnica (y hasta cierto grado) dará igual la posición y rotación del tablero frente a la cámara.

Del mismo modo conseguimos que el tamaño detectado de las piezas sea independiente también de la posición de la cámara, de no ser así, si la cámara está más lejos, el tamaño de las piezas es menor (y viceversa). En la Figura 4.7 observamos el proceso de corrección de perspectiva y el resultado obtenido.

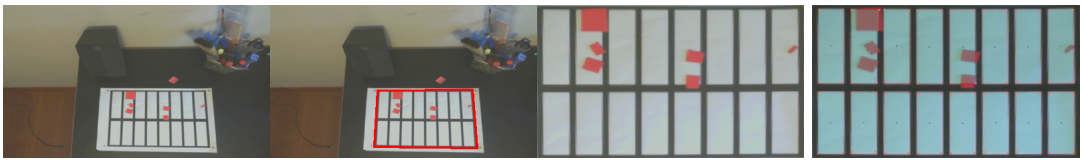

Figura 4.7: *Corrección de perspectiva sobre el tablero*

Además dado que los pasos del tableros son definidos por el usuario, haciendo uso de la interfaz gráfica ofrecida, es importante que no deba hacerlo cada vez que se cambie la posición de la cámara o la posición y/o rotación del tablero. De este modo se pueden almacenar y cargar las configuraciones de tablero y conseguir que estas coincidan con el tablero de la imagen, como se en la parte derecha de la Figura 4.7, se muestran los pasos con un ligero tinte azulado.

Capítulo 5

Instrumento musical controlado mediante Visión por Computador

5.1. El instrumento

El producto final es una aplicación de escritorio desde la que lanzar, controlar y configurar el instrumento. Dicho programa se encarga de manejar toda la interacción (tanto por visión por computador como mediante la interfaz gráfica ofrecida) y de controlar la ejecución de la instancia CSound encargada de la generación del sonido y la secuenciación de los pasos.

5.1.1. Funcionamiento

Se trata de un instrumento con varias pistas. Cada una de estas pistas está asociada a un sintetizador de sonido implementado en CSound. De esta manera cada una puede producir un timbre distinto.

Se distingue entre pistas melódicas y rítmicas; las primeras se utilizan para crear melodías (secuencias de notas) y las segundas en la creación de ritmos (haciendo uso del «Sampler multipista» en la página 36).

Además a cada pista se le asigna un color, que se corresponde con el de las piezas que se utilizarán para controlarla. Esta selección se hace con la herramienta explicada en la sección 4.3.2.1 en la página 28.

Las pistas poseen una serie de pasos ordenados, que conforman la secuencia. Esos pasos se asocian con los dibujados por el usuario sobre la imagen mostrada del tablero a utilizar. Todas las pistas comparten los mismos pasos sobre el tablero; la forma en la que definimos a qué pista corresponde es mediante el color de la pieza. Se puede reducir el número de pasos asociados a cada pista, pero una pista nunca puede tener más pasos de los dibujados.

El tablero, a parte de servir de guía a la hora de definir(dibujar) los pasos, es también una referencia visual para el usuario en el momento de colocar las piezas; para saber en qué lugar deben ir.

Llegados a este punto, se debe recordar que una buena parte de este proyecto es la búsqueda de nuevas interfaces de control, y por ello el concepto de tablero es muy flexible. Podrían utilizarse los patrones geométricos que poseen algunas ventanas, colocando la cámara en el interior y utilizando piezas adhesivas desde el otro lado de la ventana. Una idea más experimental sería colocar la cámara orientada a un aparcamiento y utilizar las plazas del mismo a modo pasos del tablero y los coches de colores como piezas sobre el mismo. Tampoco existen restricciones a la hora de dibujar los pasos de la secuencia. Pueden variar en forma, tamaño y posición, permitiéndose incluso la superposición total o parcial de los mismos (si por cualquier razón así lo quisiera

el usuario). Cabe recordar que estos usos alternativos constituyen uno de los aspectos más atractivos del proyecto. El propósito no es (solamente) crear un tablero fijo sobre el que poner piezas de colores, sino ofrecer una herramienta para la exploración de nuevas formas de interacción con la música.

Al tratarse de un secuenciador musical, desde el momento en el que el usuario lo pone en marcha, éste comienza a recorrer la secuencia de pasos uno a uno. Sobre cada uno de esos pasos el usuario coloca piezas de distintos colores (que definen a qué pista pertenecen) y tamaños.

En una pista melódica, si no existe una pieza de su color en el paso actual, no se reproducirá ningún sonido. Por el contrario, si se detecta una pieza de dicho color, se reproducirá la nota establecida (haciendo uso de la interfaz gráfica) para ese paso con una duración asociada al tamaño (área) de la pieza. Si se detecta más de una pieza para un color en un paso, se utiliza la que posea un área mayor. La justificación de esa decisión es excluir el (posible) ruido que se introduzca en la imagen y pueda ser detectado como pequeñas piezas.

En el caso de que se trate de una pista rítmica, el proceso varía ligeramente. En esta ocasión el tamaño de la pieza no define la duración del paso, sino qué sección rítmica debe sonar (bombo, caja, ...). Este tipo de pistas permiten la presencia de varias piezas de un mismo color en un paso (ignorando duplicados), con lo que en un mismo paso pueden sonar simultáneamente varias partes del instrumento rítmico. Para estas pistas la duración de sus pasos es siempre la misma, cosa que facilita la composición, al evitar que se puedan crear ritmos demasiado complejos y poco agradables. Lo que sí se ofrece es la capacidad de establecer la duración de esos pasos mediante figuras musicales (corchea, negra o blanca). De esta forma se pueden producir ritmos el doble de rápidos (corchea) o lentos (blanca) que el tempo global del instrumento (negra).

5.1.1.1. Duración variable de cada paso

Una característica notable del secuenciador implementado es la capacidad que tiene para definir la duración de cada uno de sus pasos individualmente (únicamente en las pistas melódicas). Dicha característica es poco común en los secuenciadores, donde la duración de cada paso es siempre la misma, lo que reduce la complejidad de las secuencias que pueden generarse y resta potencial al instrumento.

Al permitir este ajuste, las pistas comienzan a desfasarse entre sí, puesto que unas se reproducirán a más velocidad que otras. Con esto se consiguen muy buenos resultados, musicalmente hablando. Debido al desfase, se puede tardar bastantes ciclos en reproducir exactamente la misma secuencia de notas (en el conjunto de todas las pistas). Esto dota al sonido de más riqueza, pues introduce una serie de variaciones de baja frecuencia (lentas) entre las secuencias reproducidas.

5.1.2. Configuración

Para poder utilizar correctamente el instrumento el usuario debe configurar una serie de parámetros. Éstos deben establecerse en el orden en el que son expuestos a continuación (En la sección 5.3.1 detallan los elementos de la interfaz gráfica empleados en esta configuración):

- Se debe seleccionar y activar el dispositivo de captura a utilizar. Esto es útil sobre todo en ordenadores portátiles, pues generalmente no deseamos utilizar la cámara integrada (por su poca flexibilidad en el posicionamiento) sino cualquier otra que se encuentre conectada al sistema.
- Establecer, sobre la imagen mostrada en tiempo real, los cuatro puntos que contengan el área a utilizar como tablero. El propósito es realizar la corrección de perspectiva y sólo mostrar la zona de interés seleccionada.
- Dibujar cada uno de los pasos del tablero sobre la imagen mostrada. Existe la posibilidad de guardar y cargar tableros. Los pasos son almacenados en ficheros XML junto al resto de información necesaria.

- Se ofrecen al usuario tres (3) posibles figuras musicales a utilizar en cada paso; corchea, negra y blanca. Éstas establecen la duración de los pasos. El usuario debe especificar un rango de valores que represente el área asociada a cada uno de los tipos de piezas. Todos estos valores se almacenan entre ejecuciones, por lo que no se espera tener que cambiarlos si se utiliza la misma configuración de tablero y piezas.
    - En el caso de las pistas rítmicas, se ofrecen tres (3) posibilidades, cada una asignada a una sección del instrumento rítmico; bombo, caja y hi-hat[1]. Se asignarán los tamaños de piezas a cada una de ellas. Esto permite el uso de piezas diferentes, a las de las pistas melódicas, en las pistas rítmicas.
- Establecer el color de las piezas de cada una de las pistas. El proceso es bastante rápido y también se almacenan los valores entre ejecuciones, con lo que si las condiciones lumínicas son las mismas no es necesario volver a realizar el proceso.
- Configurar el resto de parámetros del instrumento como el tempo (Bpm[2]), las notas de las secuencias de cada pista, el largo de cada pista, ...

5.2. Detalles de la implementación

5.2.1. Breve aproximación a los UDOs implementados

A continuación se explican individualmente los instrumentos creados en CSound. Se trata tanto de los instrumentos encargados de producir sonido, como de la lógica del propio secuenciador.

5.2.1.1. Sustractivo

La implementación de esta técnica de síntesis dio como resultado el instrumento reflejado en la Figura 5.1.

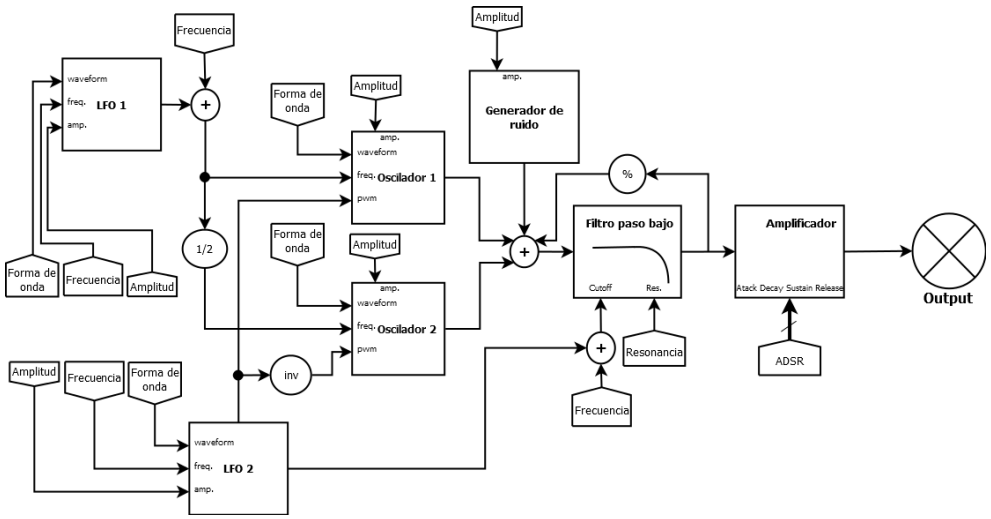

Figura 5.1: *Diagrama de funcionamiento de sintetizador sustractivo*

[1] Charleston, Charles, o Contratiempos
[2] Del inglés Beats Per Minute (Pulsaciones Por Minuto)

Como puede observarse está compuesto por:

- Dos osciladores principales, encargados de generar las señales de audio a la frecuencia deseada. Ambos osciladores son iguales (usando el mismo opcode). Permiten establecer la forma de onda deseada (seno, triangular, cuadrada, pulso modulado, diente de sierra y diente de sierra invertido), la frecuencia de oscilación, la amplitud («volumen») de la señal y el ancho de pulso para las formas de onda que lo utilicen. Destacar que la frecuencia de ambos osciladores es establecida por el mismo parámetro, y que el oscilador número dos (2) reproducirá siempre notas a la mitad de la frecuencia que el oscilador número uno (1).

    - La generación de ondas sonoras mediante modulación de ancho de pulsos (PWM[3] del inglés) modifica el ciclo de trabajo de dicha señal, variando así la cantidad de energía de éstas. A nivel sonoro, la variación del ciclo de trabajo, produce un efecto de barrido similar al de un filtro, con cierto grado de distorsión. Con esta técnica se pueden producir sonidos aún más ricos armónicamente.

- Dos LFO. Cada uno de ellos posee parámetros para ajustar su forma de onda, frecuencia y amplitud. Ofrece las mismas formas de onda que los osciladores principales. Cada uno de estos módulos permite establecer frecuencias desde los 0.0001 hasta los 20 hercios. El primero de ellos se encarga de modular la frecuencia de entrada a los osciladores principales, produciendo un efecto «vibrato». El segundo LFO establece los valores de ciclo de trabajo en los osciladores principales. Obsérvese que la señal enviada al oscilador número dos (2) es invertida respecto a la del número uno (1). Además también se encarga de modular la frecuencia de corte en el filtro paso bajo.

- Haciendo uso de un módulo de mezclado de audio, se combinan las señales generadas por ambos osciladores y la del generador de ruido. En este caso se trata de un generador de ruido blanco, con amplitud variable. Se ha optado por utilizar un generador de ruido blanco, pues es una señal aleatoria que contiene todas las frecuencias y todas ellas tienen la misma potencia. Con esto se consigue enriquecer el espectro armónico de los osciladores antes del filtrado.

- A continuación viene la fase de filtrado. Se trata de un filtro paso bajo, es decir que atenúa frecuencias superiores a las de la frecuencia de corte (en inglés, «cutoff») establecida. Se trata de un filtro modelado a partir del famoso filtro en escalera inventado por Bob Moog a mediados de la década de 1960. Este tipo de filtros añaden un poco de distorsión a la señal, lo que la dota de un sonido peculiar y reconocible. Es un filtro resonante, lo que significa que no sólo filtrará, sino que es capaz de potenciar ciertas frecuencias. Cabe destacar la existencia de un ciclo de retroalimentación ajustable, que conecta la salida del filtro con su entrada. Con esto se consigue un efecto de «Overdrive» en las frecuencias de resonancia. Este se produce cuando una señal es amplificada por encima de los valores permitidos, lo que resulta en un recorte de parte de la misma. Esos cambios bruscos en la frecuencia de onda generan armónicos, lo cual enriquece el sonido. En la parte superior de la Figura 5.2 se muestra la forma de onda y en la parte inferior el espectro de armónicos generado por la señal. Se puede observar como al recortarla se generan armónicos.

[3] Pulse-Width Modulation

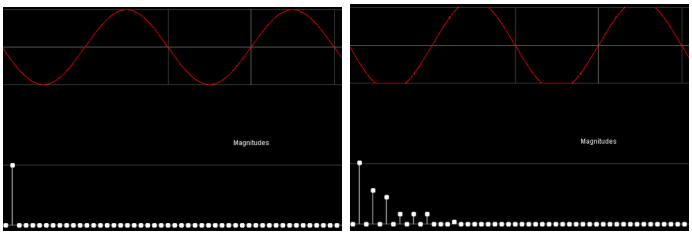

Figura 5.2: *Overdrive / Clipping*

- Por último se suministra la señal de audio filtrada a un amplificador, encargado de aplicar una envolvente[4] mediante los ajustes ADSR, «Attack, Decay, Sustain & Release» (o Ataque, Decaimiento, Sostenimiento y Relajación). Las envolventes se utilizan generalmente para establecer el volumen final de la salida de un amplificador. Con esto se consiguen crear variaciones en el volumen del sonido con cada nota producida. Como puede verse en la Figura 5.3, la envolvente (línea de puntos) cambia la amplitud de la onda (en azul) en función de los cuatro (4) parámetros que ofrece. Este módulo es esencial en cualquier sintetizador de música. La salida de este módulo es lo que se considera la salida de audio del instrumento.

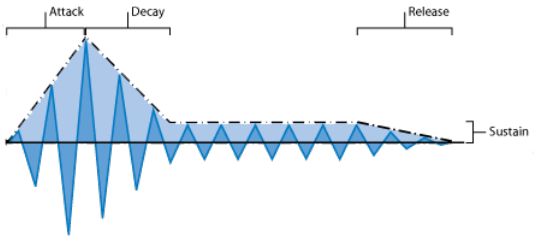

Figura 5.3: *ADSR*

5.2.1.2. Aditivo

En la implementación de este método de síntesis se ha optado por emplear la técnica reflejada en el documento [AU1] de la bibliografía. En éste se muestra como construir un UDO que funciona de forma recursiva, llamándose a sí mismo para cada uno de los parciales (o armónicos) que forman parte del timbre generado.

En esta técnica se emplean tablas denominadas RAP que es un acrónimo de las palabras inglesas para Ratio, Amplitud y Fase («Ratio, Amplitude and Phase»). Éstas son estructuras de datos personalizadas que describen los parciales a ser generados. Cada uno está descrito por el ratio de éste respecto a la frecuencia fundamental, su amplitud y la fase de la onda. Estas tablas son generadas en el momento en el que se ejecuta la instancia CSound.

Cada una de las entradas en, una tabla RAP, almacena los datos de un armónico. Con cada llamada recursiva se extrae y procesa una fila de la tabla. Se aplican además una serie de modulaciones a la frecuencia y amplitud de cada parcial (a discreción del usuario) . Con esto se introduce cierta variación, a los datos obtenidos, en el momento de ser procesados.

El proceso de filtrado se realiza individualmente, para cada parcial y antes de generar cualquier onda sonora (consiguiendo una aproximación más «pura» a la síntesis aditiva, evitando elementos sustractivos). En

[4]Es una señal empleada para definir la evolución temporal en amplitud de cualquier sonido.

este proceso se atenúan ciertos parciales y se resaltan otros. Todo este proceso es configurable por el usuario, mediante los parámetros proporcionados.

La generación de sonido se realiza mediante la creación de un oscilador sinusoidal para cada parcial, después de establecer sus valores en los pasos anteriores. La llamada recursiva se realiza con los mismos parámetros suministrados por el usuario, salvo por uno que se encarga de controlar el nivel de recursión actual (y por ende el parcial que está siendo generado).

El UDO creado ofrece la siguiente interfaz (o parámetros de configuración):

- Frecuencia base: La frecuencia de la nota fundamental a generar.
- Modulación en frecuencia: Admite señales que varíen continuamente el tono calculado para cada parcial.
- Frecuencias mínima y máxima a generar. Con esto se consigue limitar la banda de armónicos a generar.
- Tabla RAP a utilizar: Las tablas generadas representan distintos tipos de formas de onda (armónicamente ricas) y son preconstruídas al iniciarse la ejecución del código.
- Curvas de modulación de amplitud y filtrado, que permiten modular continuamente esos parámetros. No son más que envolventes (más sencillas que un ADSR) que se aplican tanto a la amplitud del sonido como a la frecuencia de corte del filtrado.
- Ancho del filtrado, que se traduce en la pendiente de atenuación del proceso de filtrado. Esa pendiente es el ritmo al que se rechazan frecuencias indeseadas (pasadas la frecuencia de corte o «cutoff»).
- Índice de recursión, que debe ser establecido por el usuario siempre a cero (0), y se encarga de controlar la ejecución recursiva del instrumento.

A partir del UDO generado, se pueden crear instrumentos que controlen, y abstraigan al usuario, de ciertas partes del mismo. De este modo podemos, por ejemplo, crear un instrumento que genere sonidos que recuerden a instrumentos de cuerda (como violines), utilizando una tabla RAP de una forma de onda «diente de sierra» con 32 armónicos y definiendo una serie de curvas que generen el característico «vibrato» en frecuencia y que modele la envolvente a aplicar a la salida. Esos valores son suministrados al opcode (UDO), dejando al usuario la, más sencilla, tarea de definir la duración del sonido, la amplitud y la frecuencia de la nota fundamental. Mediante esta técnica podemos definir distintos instrumentos que simplifiquen la utilización de la síntesis aditiva creada.

Las ventajas del uso de estas tablas, combinan la potencia que ofrecen las tablas de ondas pregeneradas (que contienen los valores de la forma de onda a generar, como muestras de audio) y el uso de decenas de osciladores (uno por cada parcial). Gracias a las tablas RAP se pueden configurar los parámetros para controlar, individual y consistentemente, cada uno de los osciladores empleados en la síntesis aditiva.

La construcción de las tablas RAP se lleva a cabo utilizando «instrumentos generadores», que es el nombre que se da en CSound a los instrumentos encargados de generar o manipular datos y no de producir o modificar sonido. Se proveen instrumentos generadores para crear tablas RAP de formas de onda de diente de sierra, cuadrada, triangular y arbitrarias. Estos instrumentos permiten establecer el número de parciales que se generarán. Estos son los instrumentos que se llaman al inicio de la ejecución para crear las tablas deseadas.

5.2.1.3. Sampler multipista

Para la implementación del instrumento rítmico, se decidió utilizar muestras de audio pregeneradas con los sonidos deseados. Se podría haber modelado el sonido de cada una de esas partes con un instrumento

sustractivo. No se trata de una tarea simple y además con el uso de esas muestras de audio se puede disponer de distintos tipos de sonidos para cada una de las partes. En concreto se ha implementado utilizando sonidos de una batería obtenidos de una librería de sonidos.

Dado el funcionamiento de CSound, los ficheros son leídos y cargados en tablas al comienzo de la ejecución, con lo que se leerán automáticamente desde la memoria principal (acceso más rápido).

El funcionamiento interno de este instrumento es bastante sencillo. Éste recibe los parámetros típicos (cuando empezar a sonar y durante cuanto tiempo), y parámetros para definir qué muestra de audio utilizar (en concreto, qué tabla) y a qué amplitud (volumen) y frecuencia se reproducirá la señal generada. El cambio de frecuencia permite la afinación de los componentes rítmicos.

5.2.1.4. El secuenciador

Se trata de la parte más crítica del instrumento. Es el encargado de avanzar, paso a paso, por los valores establecidos en las distintas pistas, generando señales para activar cada instrumento en el momento justo. Puesto que los pasos de la secuencia deben ser reproducidos a intervalos bien definidos (para que no se produzcan desviaciones en el tempo y para que todos los pasos duren exactamente lo que deben durar), la lógica del secuenciador fue desarrollada también en CSound. El lenguaje ofrece facilidades para la creación de eventos dependientes del tiempo; como pueden ser las herramientas para la generación de señales de reloj precisas. Una implementación en C# (lenguaje utilizado en la visión por computador y el programa final) haciendo uso de los temporizadores ofrecidos por el entorno .NET habría limitado seriamente el tempo máximo a utilizar. Todos los temporizadores disponibles poseen intervalos mínimos demasiado grandes para generar eventos a tempos por encima de los 160 pulsaciones por minuto. Además ninguna de las implementaciones garantiza una precisión de milisegundos, con lo que producirían desviaciones en el tempo.

Todas las pistas obedecen al tempo general establecido por el usuario, aunque debido a la duración variable de cada paso, la reproducción suele desincronizarse.

El secuenciador dispone de una señal de reloj con la que se define el tempo. La frecuencia de este reloj es el doble de la establecida por el usuario. Esto es así puesto que al definir el tempo como pulsos por minutos se está definiendo el valor de duración de una negra (siendo así 60 BPM, o 60 pulsos por minutos, una negra cada segundo) y nosotros queremos que el secuenciador avance a velocidad de corchea (pues es la duración mínima que se podrá seleccionar y el resto de ellas se puede calcular sumando corcheas).

Cada una de las pistas melódicas del secuenciador dispone de tres (3) tablas de la misma longitud (la longitud máxima de la pista) y de un índice (común a las tres tablas) para el acceso a sus elementos y saber el paso actual de la secuencia. Cada posición en estas tablas contiene información sobre un paso de la secuencia. En la primera tabla se establece la duración de cada uno de los pasos, mediante el uso de los valores 1, 2 ó 4, que representan corchea, negra o blanca respectivamente. Se utiliza un acumulador para contar el número de pasos del reloj que se reciben antes de producir un avance. Es decir, si se ha leído una duración de negra (2), el secuenciador mantendrá encendido el instrumento asociado durante dos ciclos de reloj. La segunda de las tablas almacena las notas para cada uno de los pasos y la última tabla es la encargada de establecer si un paso está activado o no (1 si está activado, 0 en caso contrario).

```
Duration_Track1 = { 1, 1, 2, 2, 1, 1, 1, 4 }
Notes_Track1 = { 6.05, 6.08, 6.10, 6.08, 7.03, 7.01, 7.03, 7.01 }
Enabled_Track1 = { 0, 0, 0, 1, 1, 1, 0, 0 }
```

Destacar la notación empleada para seleccionar las notas (una de las muchas ofrecidas por CSound), y que consiste en el uso de números con dos decimales. La parte entera indica la escala y la parte decimal indica el

semitono dentro de esa escala (hasta un máximo de 12 semitonos que hay en una escala).

Para el instrumento rítmico también se han empleado tres (3) tablas, pero con una estructura diferente, pues todos los pasos rítmicos son de la misma duración y no requieren notas. Por esa razón, cada una de estas tablas contiene valores binarios (y está asociada a cada una de las partes de la pista rítmica), indicando si cada uno de sus pasos está activo o no. De esta forma se pueden reproducir varias partes del instrumento rítmico a la vez (estableciendo el mismo paso a 1 en las tablas deseadas) simplemente ejecutando, a la vez, distintas instancias del instrumento del punto anterior.

```
Kick = { 1, 1, 0, 1, 1, 1, 0, 1 }
Snare = { 0, 0, 1, 0, 0, 1, 0, 1 }
Hihat = { 1, 0, 1, 1, 0, 0, 0, 0 }
```

Una importante característica que ofrece el secuenciador es la de generar señales MIDI a la vez que las llamadas a los instrumentos CSound. Se utiliza un canal MIDI por cada pista. De esta forma se puede desactivar la síntesis de sonido (totalmente o de algunas pistas) y utilizar el sistema para controlar otros instrumentos electrónicos. Gracias a ello se extiende la utilidad del proyecto, pues deja de estar restringido al uso de los sintetizadores implementados, dotando al usuario de, aún, más libertad.

Un aspecto curioso derivado del funcionamiento de CSound, es que el instrumento «Secuenciador» debe ser ejecutado como cualquier otro (desde la partitura). Para ello se debe especificar la duración de la instrucción dada en segundos. En este caso se ha utilizado un valor muy elevado (65536), por lo que si el sistema se ejecuta durante más de 18'2 horas, el secuenciador se detendrá; dejándose de producir sonidos. Este valor puede aumentarse o se pueden encadenar llamadas (de larga duración) al instrumento.

5.2.2. Diagrama de clases

Se expone a continuación un diagrama de clases simplificado (para mostrar las clases, atributos y métodos más relevantes) que muestra las estructuras de datos implementadas y como se relacionan. También se hará una descripción de las clases y de sus aspectos más relevantes.

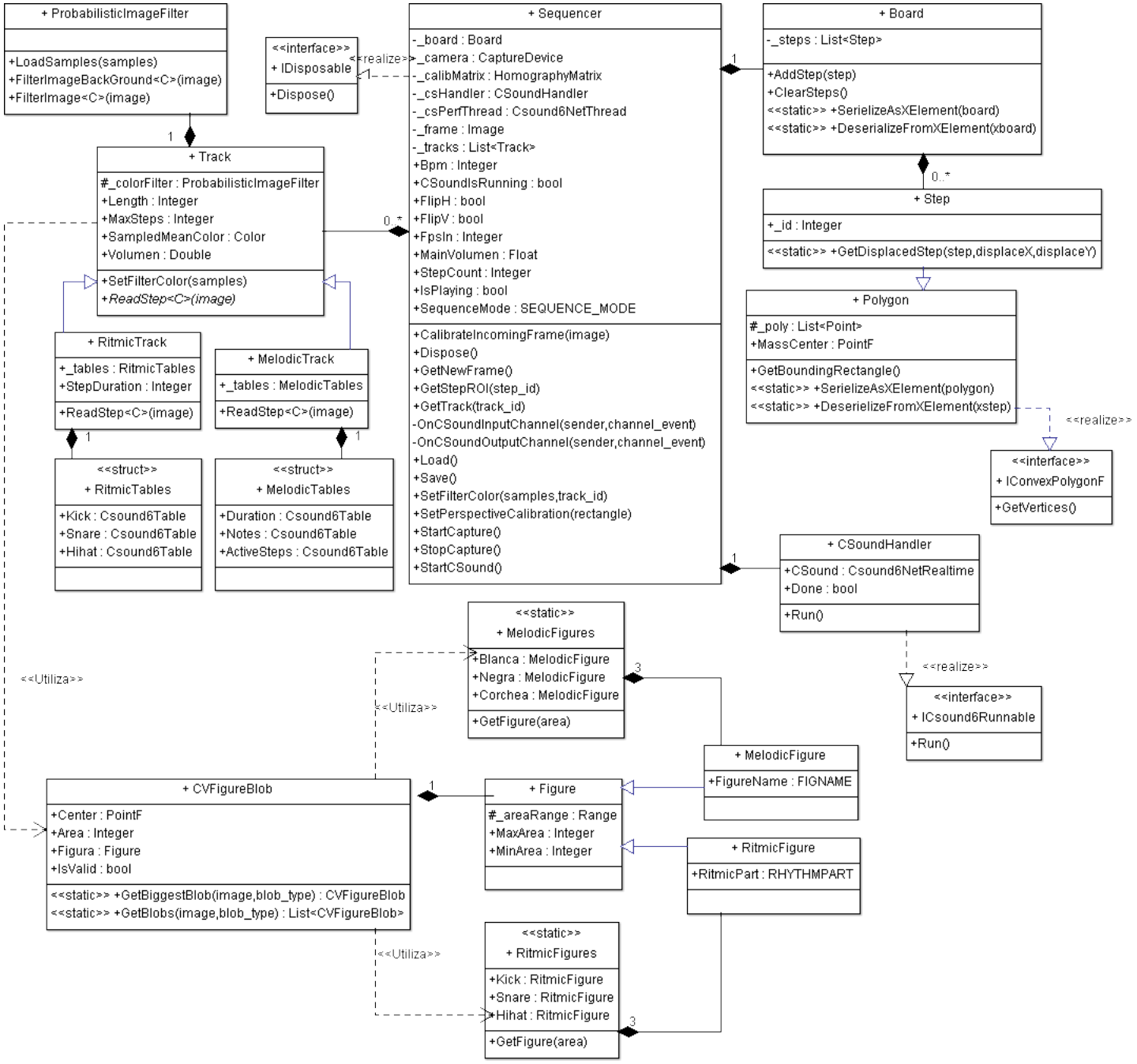

Figura 5.4: *Diagrama de clases*

Se puede observar como el elemento principal del diagrama es la clase «Sequencer». Ésta contiene toda la

lógica necesaria para aunar la síntesis de sonido (instanciando CSound en un hilo propio) y el control mediante visión por computador (haciendo uso de un tablero y sus piezas).

A continuación se presenta un listado describiendo los detalles del sistema.

- La clase **Board** representa el tablero a utilizar. Posee los métodos necesarios para serializar[5] y deserializar[6] sus atributos en formato XML. Está compuesto por una colección de objetos de la clase **Step**, que representa cada uno de los pasos creados por el usuario. La clase *Step* es hija de la clase **Polygon**. Ofrece un método de clase para desplazar todos los puntos de un polígono, ya que estos son relativos al tamaño de la imagen sobre la que se dibujan. La clase *Polygon* es implementada para ofrecer mejor integración con la biblioteca EmguCV a través de la interfaz **IConvexPolygonF**. Dicha interfaz nos facilita definir el área representada por un paso en el proceso de análisis, explicado en la sección 5.2.3.1 en la página siguiente. La clase *Polygon* ofrece, además, métodos y atributos de apoyo, como el centro de masa del polígono, o el rectángulo mínimo que lo contenga.

- La clase **Figure** es la encargada de almacenar el área asociada con una figura musical (negra, corchea, ...) o parte de una pista rítmica (bombo, caja, ...). Dado que el sistema permite establecer áreas diferente a las piezas de las pistas melódicas y las de las rítmicas, se han creado dos clases hijas que poseen un atributo que representa el nombre que se le ha dado a la figura o parte rítmica. Finalmente se crearon instancias globales de las figuras y partes rítmicas utilizadas en el prototipo. Esas instancias se encuentran encapsuladas dentro de las clases estáticas **MelodicFigures** y **RitmicFigures**. Éstas almacenan instancias de las figuras (o partes rítmicas) con unos valores que definan su área. Ofrecen además un método que permite asignar una de las figuras a un área dada, que es empleado en el proceso de análisis de los piezas en los pasos.

- Por otro lado se encuentra la clase **Track** que representa las pistas del secuenciador. Tienen atributos con los que configurar el largo de pista a utilizar, el número máximo de pasos (restringido por las tablas en CSound) y un volumen individual. Además cada pista posee una instancia propia del filtro de color implementado (y explicado en la sección 4.3.2 en la página 27). Una vez configurado el filtro (obtenidas las muestras) se tiene acceso al color generado por la media de las muestras. Esta clase es padre de las clases **RitmicTrack** y **MelodicTrack**. Estas se implementan, debido a que es necesario diferenciar ambos tipos de pista, pues su funcionamiento es diferente (por ejemplo, sólo las pistas melódicas permiten definir las notas de los pasos). Cada una de estas clases hija, hace uso de un *struct* (diferente) que aglutina los objetos con los que se obtiene acceso directo a las tablas de la instancia CSound en ejecución.

- A continuación la clase **CVFigureBlob** tiene el papel de unir la visión de computador con la síntesis de sonido. Esta clase almacena la información necesaria de una pieza, obtenida mediante el análisis con visión por computador, que son el área y la posición del centro de masa del objeto. Además posee un objeto *Figure* con una de las instancias globales de las figuras (explicadas en el punto anterior). Tiene dos métodos de clase con los que obtener la mayor pieza leída en una imagen o una colección de todas ellas. Los resultados son objetos de la clase *CVFigureBlob* con todos sus atributos definidos.

- Por último se hará una descripción de la clase **Sequencer**. Posee un objeto de la clase *Board* y una colección de objetos *Track*. La clase *Sequencer* se encarga de enlazar ambas partes restringiendo el número de pasos de las pistas a los disponibles en el tablero. Dado que este secuenciador hace uso de la visión por computador, esta clase posee un objeto **CaptureDevice,** que no es más que una clase ofrecida por

[5] «Proceso de codificación de un objeto en un medio de almacenamiento con el fin de transmitirlo [...] como una serie de bytes o en un formato humanamente legible como XML o JSON.» Wikipedia.

[6] Proceso contrario a la serielización, donde se construyen objetos a partir de una codificación del mismo.

EmguCV para manejar el acceso a los dispositivos de captura de vídeo compatibles. Se almacena en todo momento el último fotograma obtenido. Si se ha configurado la corrección de perspectiva el fotograma almacenado tendrá esa corrección. Además se almacena la matriz de desplazamiento que necesita el proceso de corrección de perspectiva y que se emplea con la llegada de cada nuevo fotograma. Destacar la utilización de un objeto de la clase **CSoundHandler** que se encarga de configurar y lanzar el hilo de ejecución (al implementar la interfaz **ICSound6Runnable**). El secuenciador almacena la referencia al hilo de ejecución para su manejo.Respecto a los métodos ofrecidos, cabe destacar a «GetStepROI(step_id)», que se encarga de obtener una sección de la imagen (ROI[7]) que represente solamente el contenido de un paso del tablero (que es la imagen con la que comienza en proceso explicado en la sección 5.2.3.1). Ofrece también los manejadores de eventos encargados de los canales empleados en la comunicación con CSound. Existe un manejador para cada dirección (entrada y salida), ya que ambos procesos son a petición de la instancia CSound.

5.2.3. El flujo de los fotogramas

Una parte importante del funcionamiento interno del sistema es el camino que siguen los fotogramas capturados y las operaciones que se van realizando sobre ellos. Se presentan a continuación a modo de lista ordenada:

1. Se obtiene el fotograma desde el dispositivo de captura. Aquí se espeja la imagen horizontal y/o verticalmente (a elección del usuario). Si no se ha realizado el ajuste de perspectiva, se muestra el fotograma en la interfaz gráfica. No se continuará hasta que se defina la corrección de perspectiva.

2. Se realiza la corrección de perspectiva al fotograma original. Este nuevo fotograma corregido es almacenado en el secuenciador como el fotograma actual, hasta que llegue uno nuevo. Se muestra el fotograma en la interfaz gráfica.

3. Una vez comenzada la ejecución de la instancia CSound, esta comenzará a enviar mensajes por los canales especificados. Uno de esos mensajes indica el paso actual en la reproducción de la secuencia. Cada vez que se recibe un cambio de paso en una de las pistas, el sistema selecciona el polígono (previamente definido por el usuario) asociado con éste. Seguidamente se realiza el proceso de filtrado de color (con el color asociado a la pista) sólo de la porción de la imagen contenida en el polígono seleccionado. De esta manera evitamos realizar el filtrado de color para el fotograma completo para cada uno de los colores de los canales; reduciendo así drásticamente el uso de CPU. Además este proceso se realiza de forma asíncrona, mediante el uso de los eventos C#, con lo que varios filtrados de color (de distintas pistas) pueden realizarse a la vez, lo que facilita que cada pista pueda ser procesada independientemente de las otras.

5.2.3.1. Análisis de los pasos del tablero

Para proporcionar al usuario la mayor libertad posible a la hora de establecer el tablero, no se han introducido restricciones a la forma, tamaño o posición que puedan tener los pasos que lo componen.

Las imágenes en OpenCV son tratadas como matrices bidimensionales. Por esa razón aislar una región, no rectangular, del fotograma principal requiere de una serie de pasos y ajustes.

[7] del inglés «Region Of Interest», es la nomenclatura empleada en OpenCv para las secciones de imágenes. Generalmente no son más que referencias a secciones de una imagen; es decir, no son una copia de una parte de una imagen, sino una parte de las mismas. Si la imagen original cambia dentro de la sección de interés, esta reflejará dichos cambios.

1. Al seleccionar un fragmento de la imagen, éste tiene que ser rectangular, pues a nivel de código son el mismo tipo de dato; una imagen de un tamaño (ancho x alto) dado. En ésta, y cuanto más compleja sea la forma del paso, se incluirán zonas del tablero que no se desea que formen parte del filtrado de color. Si esas zonas forman parte del proceso de filtrado, se podrían detectar piezas fuera del paso actual. Se puede ver como en la parte derecha de la Figura 5.5 se detectaría una pieza perteneciente al siguiente paso (la imagen pertenece a un paso de un tablero circular).

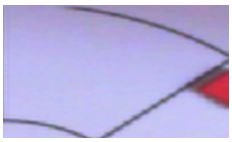

Figura 5.5: *Paso irregular ROI*

2. Para solucionarlo, se crea una máscara a partir del polígono que define al paso.

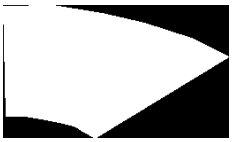

Figura 5.6: *Máscara de paso*

3. Finalmente se copia a una nueva imagen el resultado de aplicar la máscara a la muestra original. La máscara dejará intactos los píxeles que coincidan con sus píxeles con valor uno (1), blancos en la imagen, y pondrá a cero (0) el resto. De esta forma se obtiene una imagen rectangular en la que sólo se muestra el paso sobre un fondo negro.

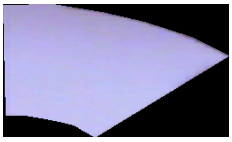

Figura 5.7: *Filtrado de paso irregular*

5.2.4. Control manual sobre los parámetros del dispositivo de captura

Debido a la importancia que tienen para el sistema las características de las imágenes obtenidas durante su uso, hubo que implementar una herramienta que permitiese el control fino de los parámetros asociados al dispositivo de captura utilizado.

Se trata de un control para Windows Forms en el que se muestran los parámetros que ofrece el dispositivo para su ajuste (brillo, contraste, gamma, saturación, ...). Gracias al uso de estos controles se pueden ajustar las propiedades de la imagen con el fin de facilitar el proceso de filtrado de color. En algunas condiciones es conveniente ajustar la cámara con valores poco habituales, como puede ser un contraste muy bajo y un brillo

elevado; lo que produce una foto con un tinte grisáceo, pero con una drástica reducción del ruido. El uso de este control facilita la configuración y utilización del sistema.

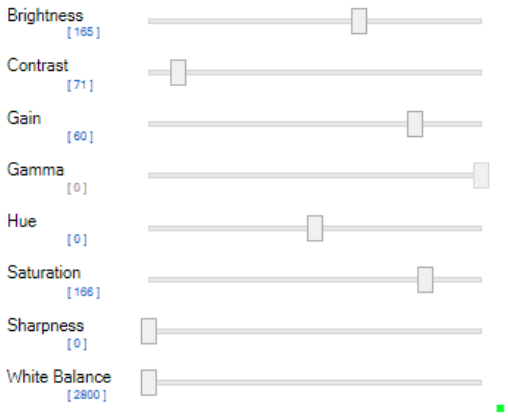

Figura 5.8: *Control parámetros cámara*

Para acceder a los parámetros de configuración de los dispositivos de captura se hace uso de la biblioteca DirectShow.NET, que se trata de un *wrapper* de la funcionalidad de DirectShow para entornos .NET. DirectShow es una API ofrecida por los sistemas Windows y construye una arquitectura para la retransmisión de contenidos multimedia a través del propio sistema operativo. Gracias a DirectShow se pueden ajustar cada uno de los parámetros ofrecidos por el control, en función del dispositivo. Esto permite el uso tanto de dispositivos que utilizan rangos de valores entre cero (0) y uno (1), como otros con rangos de valores totalmente arbitrarios y nada estandarizados. El control detecta automáticamente los rangos de valores utilizados y los parámetros que se pueden ajustar (puesto que algunos dispositivos no ofrecen todos los ajustes). Obsérvese, en la Figura 5.8, como el parámetro «Gamma» se encuentra desactivado (por no permitir el dispositivo su ajuste) y como los valores actuales de cada parámetro (en azul, bajo el nombre) se encuentra en rangos de valores distintos entre sí.

Por último cabría destacar que muchos modelos de webcams (sobre todo los más modernos) almacenan los ajustes hechos mediante este método. De esta forma la configuración se mantiene entre ejecuciones del sistema, lo que repercute favorablemente en la usabilidad del mismo.

5.3. Estado actual del instrumento

5.3.1. Funcionalidades

Se presenta a continuación un listado con las funcionalidades implementadas en el prototipo realizado. Se muestran también una serie de capturas de pantalla de la interfaz gráfica ofrecida, resaltando las partes asociadas a cada funcionalidad.

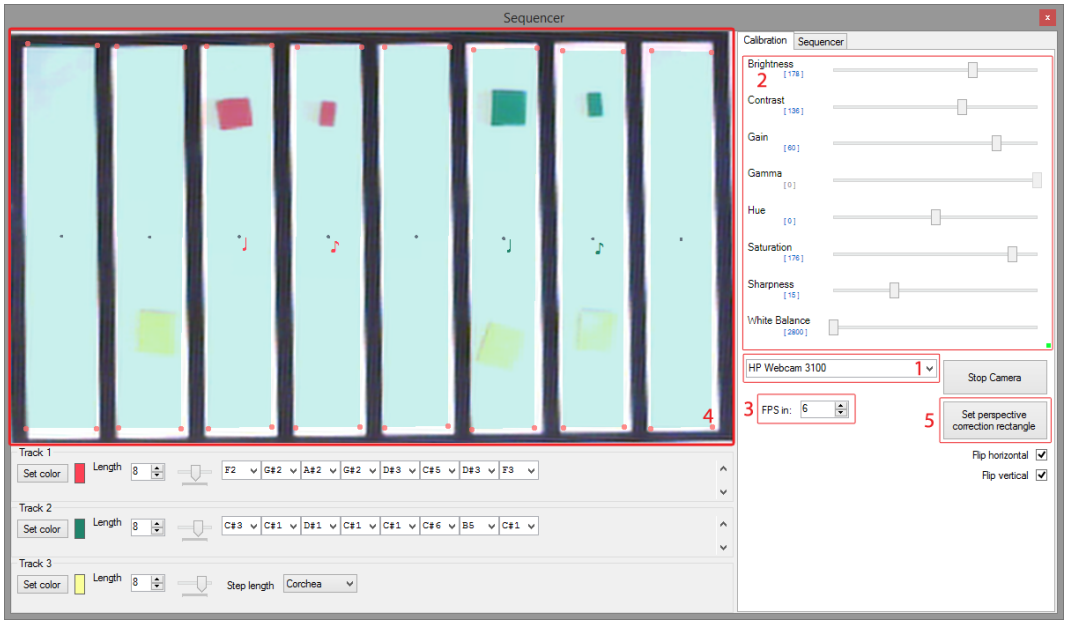

Figura 5.9: *Captura de pantalla 1*

- Selección de dispositivo de captura: El usuario puede seleccionar que dispositivo de entrada de vídeo a utilizar [1 en la Figura 5.9].
- Control sobre los parámetros del dispositivo de captura seleccionado [2 en la Figura 5.9]. (Utilizando le control descrito en la página 42).
- Capacidad de controlar la cantidad de fotogramas por segundo que se leerán del dispositivo de captura de vídeo [3 en la Figura 5.9].
- Monitor principal, dentro de la interfaz gráfica [4 en la Figura 5.9], que permite la visualización en tiempo real de las imágenes captadas, los pasos dibujados y los valores asociados por cada pista a cada paso. Sobre éste se realizan las operaciones de selección de muestras, para el filtrado de color, el dibujado de pasos y la configuración de la corrección de perspectiva. Cabe destacar la posibilidad de espejar la imagen mostrada tanto vertical como horizontalmente, lo que provee de más libertad al usuario a la hora de colocar el dispositivo de captura de vídeo.
- Definir y aislar en la imagen el área que contenga al tablero [5 en la Figura 5.9].(Realizando el proceso de corrección de perspectiva en la página 29).

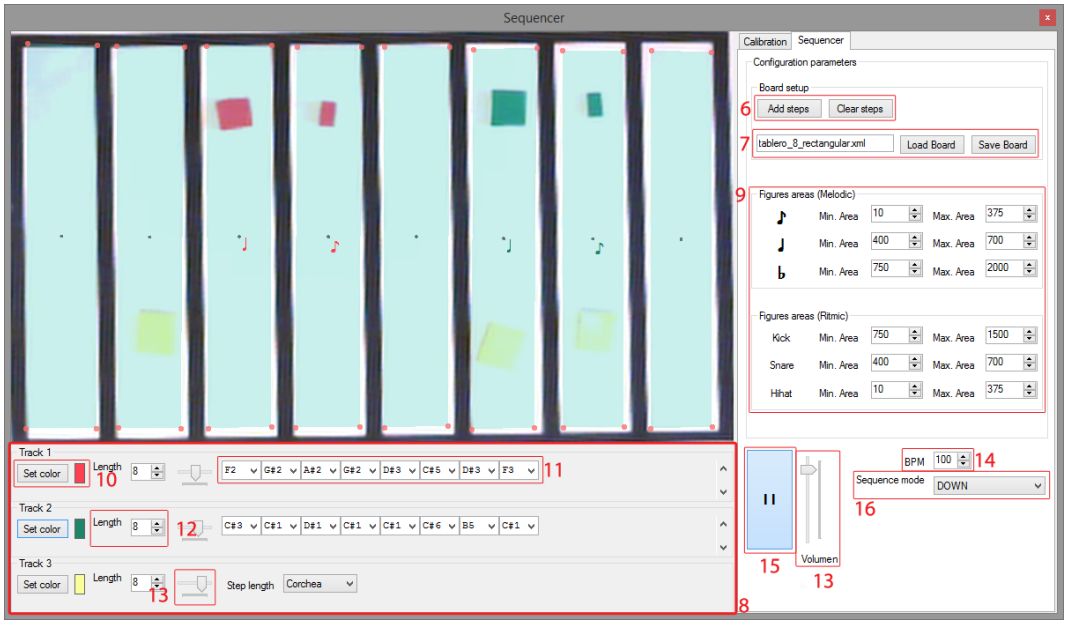

Figura 5.10: *Captura de pantalla 2*

- Dibujado y edición de los pasos, del secuenciador, sobre la imagen mostrada del tablero [6 en la Figura 5.10]. Los pasos no tienen restricción de tamaño, forma o posición.
- Capacidad para el almacenado y carga de los pasos, con el fin de reutilizar los distintos tableros creados [7 en la Figura 5.10].
- Se ofrecen tres (3) pistas, cada una con un instrumento diferente asociado [8 en la Figura 5.10]. En concreto dos (2) pistas melódicas y una (1) rítmica. En este prototipo la primera pista produce sonidos parecidos a un instrumento de viento metal mediante la utilización de la síntesis sustractiva. La segunda pista hace uso de la síntesis aditiva y genera un timbre similar a un violín. Y la tercera pista (la rítmica) reproduce los sonidos de una batería.
- Controles para definir los rangos de áreas asignados a las figuras musicales (para las pistas melódicas) o partes rítmicas (para las pistas rítmicas) [9 en la Figura 5.10].
- Cada pista posee una opción para recoger las muestras a utilizar en el filtrado de color. Una vez recogidas y procesadas, se muestra el color medio generado, para una mejor identificación, por parte del usuario, de piezas y su pista [10 en la Figura 5.10].
- Capacidad de establecer, en cualquier momento, cada una de las notas de la secuencia, en pistas melódicas [11 en la Figura 5.10].
- Posibilidad de cambiar, también en cualquier momento, la longitud de cualquiera de las pistas. Las pistas pueden tener diferentes longitudes, unas respecto de las otras. Una pista nunca podrá tener más pasos de los dibujados [12 en la Figura 5.10].

- Control de volumen: Cada pista tiene un control de volumen individual. Además existe un control de volumen general, que afecta a todas las pistas por igual [13 en la Figura 5.10].
- Se puede establecer el tempo de reproducción, estableciendo un valor a modo de pulsaciones por minuto (Bpm) [14 en la Figura 5.10].
- Controles para poner en marcha, y detener, el secuenciador de música (Play / Pause) [15 en la Figura 5.10].
- Elección del modo de la secuencia, hacia arriba o hacia abajo [16 en la Figura 5.10].

5.3.2. Restricciones

Debido a la naturaleza del proceso de visión por computador, a que se trata de un prototipo, a dependencias de la tecnología utilizada o por decisión propia, el instrumento desarrollado posee las siguientes restricciones.

- Necesita del entorno .NET (versión 4.5) para su ejecución. Además, el uso de bibliotecas nativas, lo obliga a ser ejecutarlo en entornos Windows. Aún así, no se utilizan funciones o características avanzadas con lo que, en principio, su ejecución debería ser posible utilizando Wine[8]. (No se ha probado).
- El número máximo de pasos por pista está predefino a nivel de código (con un valor de 16) dentro de la lógica del secuenciador (5.2.1.4 en la página 37). Esto es así, debido a que el tamaño de las tablas debe ser definido en el momento de su creación y no puede aumentarse en tiempo de ejecución.
- El número de pistas está preestablecido por el sistema. La razón es que en CSound se deben definir las tablas para cada una de esas pistas desde un principio, sin poder hacerlo en tiempo de ejecución. Al tratarse de un instrumento que pretende ser sencillo e intuitivo, esta restricción parece correcta de cara a la usabilidad. De esta forma evitamos el uso de excesivas capas, cada una con su filtro de color y sus piezas correspondientes; lo que crearía confusión sobre el tablero.
- Se ha establecido un valor máximo para el tempo. Se trata de 450 BPM; ya que con valores superiores la secuencia es tan rápida que se torna ininteligible.
- El máximo de fotogramas leídos por segundo es de 24. Por encima de ese valor muchos dispositivos se vuelven inestables, pudiendo generar fotogramas incompletos o con errores.
- El sistema es relativamente sensible a los cambios de luminosidad. Para obtener resultados óptimos se recomienda el uso de fuentes de luz estáticas y que no varíen en intensidad. Esto limita su uso bajo la luz del sol, requiriendo que se tomen nuevas muestras cada cierto tiempo debido a los cambios de tonalidad que se producen en su luz a lo largo del día. Afortunadamente el proceso de muestreo es sencillo.
- El área establecida como tablero deberá permanecer inmóvil una vez se ha realizado la corrección de perspectiva y el dibujado de los pasos. De no ser así, los pasos dibujados no coincidirán con los del tablero enfocado por la cámara. Siempre se puede volver a configurar la corrección de perspectiva para reencuadrar el tablero en el fotograma.

[8] Acrónimo recursivo en inglés para «Wine Is Not an Emulator». Es una reimplementación de la API de Win16 y Win32 para sistemas operativos basados en Unix.

Capítulo 6

Conclusiones y líneas futuras

La creación de nuevas interfaces para el control de instrumentos musicales es un campo poco explorado en la actualidad. Esto es así pues los músicos están muy acostumbrados a las interfaces clásicas (como teclados y pulsadores) y el gran cambio que puede suponer comenzar a utilizar nuevas opciones. Para contribuir en este campo, e intentar facilitar ese cambio de paradigma en cuanto al control de los instrumentos musicales, se decidió llevar adelante este proyecto.

Debido a las razones expuestas en este documento, también ha sido parte integral del mismo el aprender e implementar diferentes técnicas de síntesis de sonido. En cuanto a esta materia se han adquirido conocimientos que abarcan muchos y distintos aspectos, como pueden ser la utilización de las herramientas necesarias (aprender a programar utilizando CSound) y la exploración en profundidad de diferentes técnicas de síntesis de sonido. Además en el caso de la síntesis aditiva, se aprendieron conceptos y técnicas muy interesantes (en concreto el uso de las tablas RAP y el uso de UDOs recursivos). También se ha estudiado (y expuesto) a fondo el funcionamiento de un secuenciador musical, desde los detalles de más bajo nivel (como la programación del reloj y sus eventos) a los de más alto nivel como la implementación de los modos de secuencia (arriba/abajo).

En el campo de la visión por computador, se han explorado técnicas con las que conseguir la funcionalidad deseada. Se ha obtenido un conocimiento bastante profundo sobre el filtrado por color de imágenes, llegando incluso al desarrollo de una biblioteca con la que implementar una técnica diferente a las ofrecidas (por OpenCV). Gracias a la implementación de esta técnica y a las diferencias que guarda con otras similares, se han comprendido las limitaciones a las que se enfrenta la visión por computador. Pequeñas variaciones de luminosidad o tonalidad (casi imperceptibles por el ojo humano) han de tenerse en cuenta, pues pueden arrojar resultados poco precisos. Estas limitaciones pudieron ser superadas (en cierta medida) gracias a la técnica de muestreo de color implementada (en conjunción al filtrado de color).

Además se han creado una serie de clases y estructuras de datos con las que facilitar el desarrollo de proyectos similares (que traten de aunar la producción de sonido con la visión por computador). Estas clases implementan varias técnicas de apoyo, como pueden ser el análisis mediante visión por computador de secciones individuales o el manejo de instancias CSound durante su ejecución.

Este proyecto también a servido para ampliar conocimientos a nivel de de desarrollo, aprendiendo a utilizar nuevas características de los lenguajes de programación empleados (como la ejecución asíncrona de C#).

Por último es importante destacar que existe un repositorio con todo el proceso de desarrollo del proyecto. Éste se encuentra alojado en «Github» y contiene todo el código asociado con el prototipo desarrollado (pero no todo el código generado durante el proyecto, como pueden ser el proceso de la implementación en CSound, y todas las versiones desechadas del secuenciador o los instrumentos). Se puede acceder al mismo mediante el

siguiente enlace: https://github.com/aladaris/TFG-CV

Al tratarse de un proyecto personal, se pretende seguir el desarrollo del mismo. Durante la realización de este Trabajo de Fin de Grado fueron surgiendo diferentes formas de extenderlo, pero debido a las limitaciones de tiempo del mismo, o a que estaban fuera del alcance definido, no pudieron ser implementadas. Se expone a continuación una serie de características a implementar en el futuro:

- **Automatización del reconocimiento del tablero y corrección de perspectiva:** Con el fin de facilitar aún más la configuración. Se trata de ser capaces de detectar el tablero y realizar sobre el mismo la corrección de perspectiva. Una vez hecho eso, se procederá detectar los pasos dibujados sobre el mismo. Esta característica está enfocada al uso con tableros y pasos bien definidos, no para su uso en las versiones más flexibles del concepto de tablero.
- **Interfaz de control para los parámetros de los sintetizadores de sonido:** Ofreciendo una interfaz gráfica que de acceso a la configuración de los parámetros internos de los instrumentos implementados en CSound (como puede ser el tipo de onda a utilizar, o frecuencia de corte del filtro en un sintetizador sustractivo).
- **Número de pistas variable:** Permitiendo al usuario añadir y quitar pistas a su antojo. Como se ha comentado anteriormente no conviene tener un gran número de pistas, pues eso implica un mayor número de piezas sobre el tablero. Si se satura demasiado el tablero, se perderá en usabilidad, pues es posible generar confusión en el usuario. Por esta razón se establecerá un máximo al número pistas.
- **Elección del instrumento asociado a cada pista:** De esta forma se permitirá al usuario establecer qué instrumento sonará en una pista determinada, dotándolo así de mayor libertad y potencial.
- **Implementación de otros métodos de síntesis de sonido:** En concreto los descritos en el punto 3.1.4 en la página 19. Con esto se desea conocer más a fondo la síntesis de sonido y ofrecer un mayor repertorio de timbres a utilizar.
- **Crear un sistema portátil:** Haciendo uso de sistemas como Raspberry Pi[1] se podría conseguir construir un instrumento mucho más completo aún. Con esto se espera conseguir un producto más similar al concepto de un instrumento musical, siendo algo que se puede transportar y que es capaz de funcionar por sí mismo. Con la posibilidad de ejecutar la próxima versión de Windows (la número diez) en la Raspberry Pi, en un principio esta idea es factible, pues los problemas de compatibilidad con la tecnología utilizada se reducen al mínimo.

[1] https://www.raspberrypi.org/

Chapter 7

Summary and conclusions

Creating new interfaces for controlling musical instruments remains an unexplored field at present time. This is so because the musicians are very accustomed to the classic interfaces (like keyboards) and the great change that may involve start using new options. To contribute in this field, and try to facilitate this paradigm shift in the control of musical instruments, it was decided to carry out this project.

For the reasons set previously on this document, it has also been an integral part learning and implementing various techniques of sound synthesis. Regarding this matter a good amount of knowledge has been acquired, such as the use of the tools (learn programming and using CSound) and in-depth exploration of different sound synthesis techniques. Also in the case of additive synthesis, interesting concepts and techniques were learned. In particular the use of RAP tables and using a recursive UDO. It has also been studied (and exposed) in-depth the working of a musical sequencer, from the lower-level details (such as clock programming and events) tho the highest level of implementation like the sequence modes (up / down).

In the field of computer vision, some techniques has been explored to achieve the desired functionality. A deep enough knowledge has been acquired in the color filtering field, with the development of a library with which implement a different technique than those offered by OpenCV. Thanks to the implementation of this technique and the differences it has with others, the limitations that computer vision faces on this regard has been understood. Small variations in brightness and hue (almost imperceptible to the human eye) must be taken into account, as they can lead to inaccurate results. These limitations could be overcome (to some extent) due the color sampling technique implemented (in conjunction with the color filtering).

A series of classes and data structures that facilitate the development of similar projects (which try to combine the sound synthesis with computer vision) has been created. These classes implement several technical support, such as analysis by computer vision of sections or handling individual instances of CSound.

This project also served to increase knowledge at a development perspective, using new features of the programming languages used (such as asynchronous programming in C#).

Finally it is important to mention that there is a repository with the entire process of project development. This is hosted on «Github» and contains all the code associated with the prototype (but not all the code generated during the project, such as all the discarted versions of the sequencer or the instruments). You can access it through the following link: https://github.com/aladaris/TFG-CV

Capítulo 8

Presupuesto

Debido a las características de este proyecto, el único gasto asociado al mismo es el de las horas de desarrollo. Se detalla el precio de la hora de desarrollo y el número de horas empleadas en cada tarea. Las horas de desarrollo en CSound se consideran más económicas pues no se tenían conocimientos previos sobre el mismo.

Concepto	**Coste por hora**	**Horas**	**Coste total**
Programación de la síntesis de sonido y el secuenciador musical	10 €	100	1000 €
Programación de biblioteca de filtrado por color y herramientas asociadas	16 €	100	1600 €
Creación del prototipo funcional	16 €	110	1760 €
	TOTAL		4360 €

Cuadro 8.1: Presupuesto

Bibliografía

[AU1] Jacob Joaquin. *F-Table Based Additive Synthesizer.* CSOUND JOURNAL: Home | Issue 13

[AU2] Juan Bermúdez Costa. *Nueva generación de instrumentos musicales electrónicos.* Barcelona. Marcombo, S.A, 1977. ISBN: 84-267-0213-9

[AU3] Barry Vercoe et Al. 2005. *The Canonical Csound Reference Manual.* http://www.csounds.com/manual/html/index.html

[AU4] Sound On Sound. *SYNTH SECRETS Part 14: An Introduction To Additive Synthesis.* SoundOnSound.com June 2000, http://www.soundonsound.com/sos/jun00/articles/synthsec.htm

[AU5] FLOSS Manuals. *CSound.* http://booki.flossmanuals.net/csound/introduction/

[CV1] Joseph Paradiso. *Electronic Music Interfaces.* http://web.media.mit.edu/~joep/SpectrumWeb/SpectrumX.html

[CV2] Reinhold Behringer, Rockwell Scientific Company (RSC) . *The Concept of a Visual Interface for Conducting Humans and Synthesizers.* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.545&rep=rep1&type=pdf