Universidad Rey Juan Carlos

Escuela Técnica Superior de Ingeniería Informática

INGENIERÍA TÉCNICA EN INFORMÁTICA DE SISTEMAS

CURSO ACADÉMICO 2009/2010

Ampliación del optófono

Trabajo Fin de Carrera

— Autor —

Guillermo Rey Mora

—Tutor—

Antonio Sanz Montemayor

24 de junio de 2010

Agradecimientos

Mi agradecimiento va dirigido, en primer lugar, a Antonio Sanz por la ayuda y sobretodo la paciencia que ha tenido conmigo durante todo el tiempo de elaboraciónm de este proyecto.

Asimismo dar las gracias a José Manuel por su colaboración y disponibilidad así como por sus aportaciones, sugerencias y consejos que han sido de gran utilidad para mejorar la funcionalidad del proyecto.

Por último, agradecer a mi familia y mis amigos por su apoyo día a día.

Resumen

En este proyecto se ha desarrollado una aplicación capaz de generar sonidos estructurados a partir de imágenes. Estos sonidos pueden ser interpretados de tal manera que se pueda construir, a partir de ellos, una representación básica de la imagen de entrada. La aplicación está destinada a personas invidentes con el objetivo de ayudarles en la percepción y reconocimiento de imágenes. Como referencia principal, se han utilizado los avances realizados en la sonificación de imágenes por Peter B.L. Meijer.

Señalar que la aplicación realiza sonificaciones tanto con imágenes en escala de grises como en color. En escala de grises, se puede generar sonido a partir de imágenes en las que se haya detectado los bordes de los objetos, o si se trata de vídeo, en las que se haya detectado el movimiento que existe en una escena. Para las imágenes en color, se podrá realizar la búsqueda y posterior sonificación de hasta tres colores diferentes, asociando a cada uno de ellos, un timbre diferente.

Después de tratar y analizar las imágenes se crean los ficheros de audio que serán reproducidos posteriormente. A partir de las imágenes en escala de grises será generado un fichero de audio con formato WAVE, mientras que en las de color el fichero será de formato MIDI. El control de la aplicación se maneja desde una interfaz desde la cual se puede configurar los datos de entrada asociados a la imagen y al sonido.

Para concluir el proyecto se han realizado varias sesiones de experimentación con una persona invidente, en las que se ha intentado comprobar el nivel de funcionalidad de la aplicación. Los resultados obtenidos son satisfactorios ya que se ha conseguido interpretar imágenes con cierta complejidad y sobretodo se ha conseguido identificar ciertas características del entorno visualizado por una cámara fija o en movimiento.

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

El término con el que se denomina la transformación de una serie de datos en una señal acústica interpretable es conocido como sonificación. Se trata de una tecnología emergente con la que se quiere mejorar la comunicación y la interpretación de cierto tipo de datos. Un ejemplo de esta tecnología aparece en el dispositivo conocido como pulsioxímetro o saturómetro sonoro. Dicho dispositivo es utilizado en operaciones quirúgicas y su funcionamiento se basa en la emisión de frecuencias variables determinadas según los niveles de oxígeno en sangre de una persona. De esta manera el cirujano a través del sonido conoce el estado de la persona sin tener que desatender la operación. En este proyecto, los datos a transformar en audio provienen de una imagen digital, obtenida bien desde un fichero de imagen bien a partir de vídeo.

El mayor problema de este tipo de aplicaciones se encuentra en simplificar los datos de salida de forma que, a través de ellos, se pueda conseguir la recepción de información útil e interpretable. Para conseguir este objetivo, el proyecto se apoya en técnicas y métodos propios de la visión computacional, cuya finalidad es la extracción de información del mundo físico a partir de imágenes. A cada imagen se le pueden aplicar distintos tratamientos para conseguir una selección de características básicas con las que obtener la simplificación. La causa principal de esta simplificación parte de la idea de que la capacidad de percepción de información por el sentido auditivo es menor que por el visual. En concreto, según los estudios de Homer Jacobson, realizados entre 1950 y 1951, el ojo tiene la capacidad de percepción en unos $4{,}32 \times 10^6$ *bits*/segundo mientras que el oído percibe 44600 *bits*/segundo [3] [4].

El proyecto está dirigido a personas invidentes para que a través de los sonidos que escuchen puedan ser capaces de crear una representación mental de la imagen de entrada. Se trata de conseguir simular el término neurológico “sinestesia”, que se define como la

capacidad que tienen algunas personas para experimentar sensaciones propias de un sentido sensorial a través de otro.

1.1. Estudios similares

Uniendo los avances en visión computacional y la necesidad de investigar sobre aplicaciones que reduzcan las limitaciones con las que se tienen que enfrentar las personas invidentes, relacionadas con el entendimiento y percepción de imágenes, se han desarrollado multitud de trabajos. Algunos de ellos serán presentados y explicados en esta sección.

A comienzos del siglo XX Fournier d´Albe desarrolló un dispositivo denominado optófono capaz de generar sonido estructurado a partir de imágenes. Se basaba en proyectar una imagen en una pantalla compuesta por células de selenio. Estas células, según el nivel de luminosidad que captaban, producían una señal eléctrica (efecto fotoeléctrico) que se transmitía a un altavoz produciendo un sonido [10]. Este podía ser interpretado de tal manera que través de él intuían lo que se representaba en la imagen.

Actualmente hay que destacar, los avances realizados en la sonificación de imágenes por Peter Meijer [7] en su optófono, ya que mucho de los procesos que se realizan en este proyecto son similares al suyo.

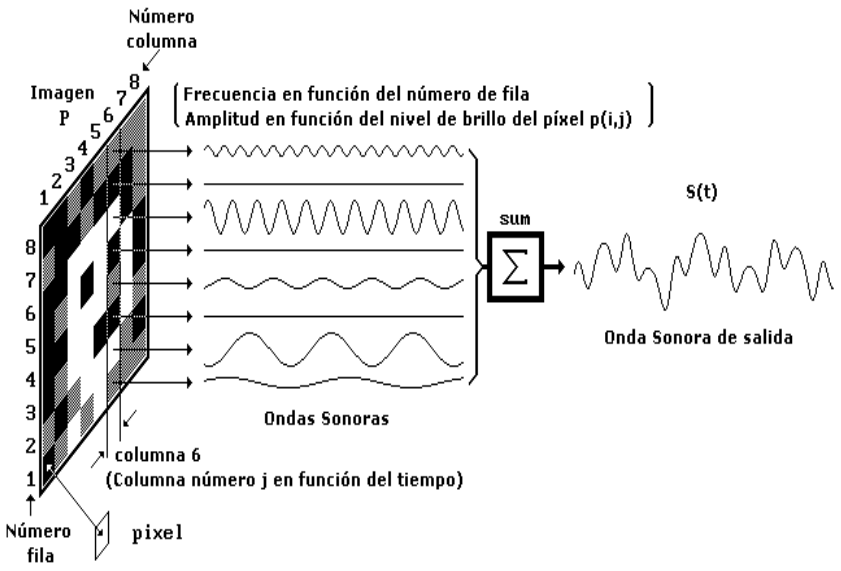


Figura 1.1: Gráfico resumen del proceso de sonificación utilizado por Peter Meijer

La figura 1.1 presenta un resumen del procedimiento de sonificación utilizado en su aplicación. La imagen es recorrida de izquierda a derecha columna por columna. De cada píxel, se analizará el brillo, para determinar la intensidad (volumen) del sonido y su posición vertical, para fijar a ese punto una frecuencia u otra. Para la asignación de la frecuencia se tiene en consideración que el máximo valor de esta, que provoca un sonido agudo, corresponderá al píxel superior y progresivamente según se disminuya la altura la frecuencia también lo hará generando sonidos más graves. Con estos valores y teniendo en cuenta la variable del tiempo, se consigue generar una onda sonora periódica finita.

La función para el cálculo de la onda sinusoidal es la siguiente:

$$S_y(t) = p\,(x, y) \cdot \sin\,(\omega_y \cdot t + \phi) \tag{1.1}$$

A continuación todas las ondas generadas, por cada píxel, son sumadas para conseguir una única onda de salida para cada columna. Esta suma se representa con la siguiente función:

$$S(t) = \sum_{x=1}^{M} p\,(x, y) \cdot \sin\,(\omega_y \cdot t + \phi) \tag{1.2}$$

La onda final que corresponda a una imagen será la concatenación de las distintas ondas de cada columna. Esta onda es reproducida posteriormente en estéreo, de modo que el sonido comienza por el canal izquierdo para ir progresivamente balanceándose hacia el derecho. De esta manera se genera un sistema de referencia espacial para determinar la ubicación de cada objeto de la imagen en su posición adecuada. Este programa es conocido como “The Voice” y actualmente puede ejecutarse en diferentes sistemas operativos, como Windows, Symbian o Android y puede ser descargado, en su versión gratuita, desde su web (*www.seeingwithsound.com*).
En la figura 1.2 se observa un ejemplo de utilización del sistema. Se trata de una cámara integrada en unas gafas oscuras, que realiza la función de recogida de imágenes. Estas serán procesadas, como se ha explicado anteriormente, por un pequeño equipo transportado en una mochila, para que posteriormente el sonido que se genere sea escuchado a través de unos cascos.

Por otra parte, en la Universidad de las Islas Baleares (UIB) han creado una aplicación de identificación de colores. Esta asocia a cada color un sonido distinto de modo que los colores se identifiquen fácilmente. Esta asociación se ha fijado después de realizar diferentes experimentaciones observando, que aunque cada persona relaciona de diferente manera los colores y los sonidos, generalmente se suelen asociar sonidos agudos a colores claros

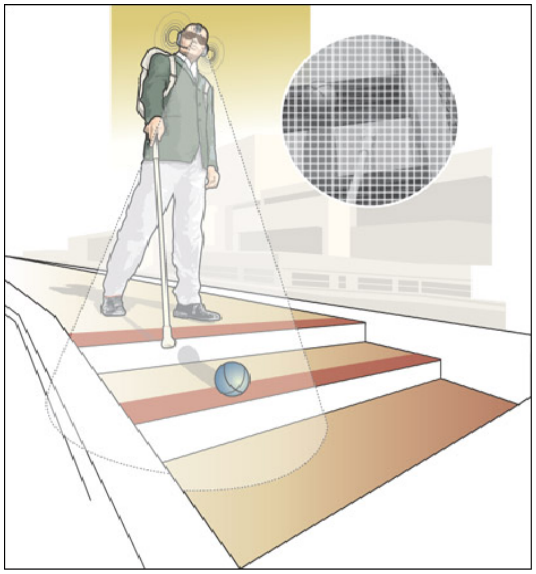
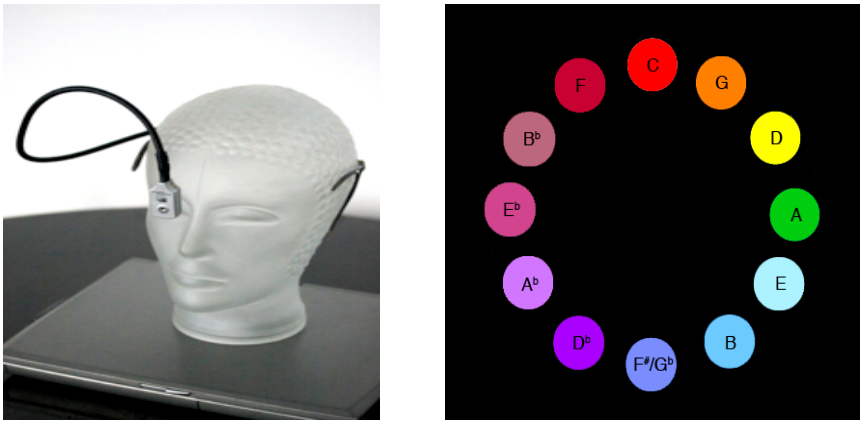


Figura 1.2: Ejemplo de la aplicación creado por Peter Meijer

y sonidos graves a colores oscuro [2]. El proyecto está destinado a en niños invidentes de corta edad.

Bajo el mismo enfoque, Adam Montando creó en 2003 el primer *eyeborg* [12]. Se trata de un dispositivo que, al igual que el proyecto anterior, asocia sonidos a colores para su identificación. Está destinado no sólo para personas invidentes si no también para individuos con impedimentos visuales como el daltonismo (imposibilidad para distinguir colores) o como la acromatopsia (visión en escala de grises). Se puede observar la fisonomía del dispositivo en el cuadro 1.1.

Como anécdota destacar que ya a finales del siglo XIX el compositor y pianista Aleksandr Skriabin realizaba asociaciones entre colores y sonidos porque, según él, tenía la capacidad sinestésica de oír colores (ver cuadro 1.1).



(a (b

Cuadro 1.1: a) Dispositivo *Eyeborg*; b) Asignación de notas musicales a colores

Con el mismo objetivo que las referencias anteriores, pero sin la utilización de la tecnología de la sonificación, se realizan trabajos paralelos. Uno de ellos trata de transformar señales visuales obtenidas de una cámara en impulsos eléctricos. Estos viajan hasta un material compuesto por microelectrodos que se coloca en la lengua de una persona. Los impulsos tienen que ser interpretados posteriormente, de manera similar a como ocurre con el sonido de los proyectos anteriores. De esta forma se puede conseguir proporcionar la suficiente información para crear una representación certera del entorno que se visualiza con la cámara [5].

En cuanto a los avances realizados para la adaptación y accesibilidad de las tecnologías para las personas invidentes destacar los diversos proyectos que se engloban en el término de tiflotecnologías. Desde la ONCE (Organización Nacional de Ciegos Españoles) y más en concreto desde su departamento de investigación conocido como CIDAT (Centro de Investigación, Desarrollo y Aplicación Tiflotécnica) se desarrollan multitud de proyectos entre los que se encuentran lectores de pantalla, tanto para ordenadores como para móviles, reconocedores de texto (OCR), conversores de voz a texto o las impresoras en Braille. Gracias a estas investigaciones, la accesibilidad a las tecnologías de la información y comunicación se ha visto mejorada considerablemente. No obstante, por lo que se refiere a percepción y reconocimiento de imágenes todavía no se han desarrollado sistemas cuya aplicación se haya generalizado.

1.2. Aspectos teóricos

1.2.1. Procesamiento de imágenes

El tipo de imágenes que se utilizan en la aplicación son las conocidas como rasterizadas, matriciales o mapa de *bits*. En realidad están definidas como una matriz de píxeles conocida como *raster*. Para acceder a cada uno de los píxeles hay que determinar su posición horizontal (eje x) y vertical (eje y). Cada píxel ofrece cierta información sobre una región elemental de la imagen. En imágenes en escala de gris esta información será el valor de brillo, mientras que en las de color será los datos propios del modelo de color utilizado. El modelo de representación de color utilizado en este caso es el conocido como RGB (*Red Green Blue*) [1] [11] el cual determina la composición del color de cada píxel por la intensidad de los colores primarios, rojo, verde y azul. El número de *bits* utilizado para representar cada píxel en el espacio RGB se llama profundidad de píxel, y en este caso, será igual a 8 *bits*. Los valores de cada color primario, por lo tanto, oscilarán entre 0 y 255.

Las imágenes serán tratadas con diferentes métodos de preprocesamiento y filtros cuyo objetivo será simplificar la información que posteriormente se utilice para la sonificación. Los conceptos teóricos se resumirán individualmente en los siguientes apartados.

1.2.1.1. Técnicas de procesamiento estáticas

Escalado

Técnica de preprocesamiento con la cual se modifica el tamaño de la imagen según un factor de escalado o a partir de unos valores exactos que indican la altura y la anchura a la que se quiere dimensionar. Para la optimización del tiempo necesario para procesar una imagen se realiza un escalado, en el cual se disminuye el tamaño de esta de manera considerable. La otra aplicación que tiene este método se consigue al disminuir la imagen y posteriormente devolverla a sus dimensiones originales (ver figura 1.3). Con esto se obtiene una pérdida de resolución que consigue la eliminación de pequeños detalles que posteriormente podrían dificultar la interpretación del sonido de salida.

Figura 1.3: Proceso de reescalado

Reducción rango dinámico de color

Operación mediante la cual se transforma una imagen en color a otra codificada en escala de grises o monocolor (ver figura 1.4). Para ello se aplica una media ponderada de los valores de color primarios de cada píxel obteniendo un único valor que hace referencia al nivel de luminosidad de este. La ecuación utilizada para calcular el nivel de luz de cada píxel es la que aparece más adelante, siendo los valores de ponderación (0.30, 0.59, 0.11) determinados según el grado de sensibilidad que tiene el ojo humano hacia cada color primario.

$$V_I = 0{,}30 \cdot V_R + 0{,}59 \cdot V_G + 0{,}11 \cdot V_B$$

(1.3)

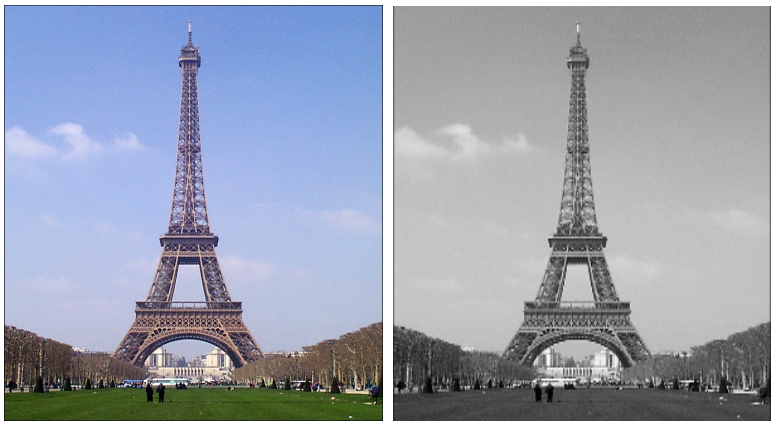


Figura 1.4: Transformación de imagen en color a escala de grises

Umbralización

La umbralización es un proceso que permite convertir una imagen en escala de grises a binario de forma que los píxeles sólo puedan ser representados con dos valores de brillo. Si un píxel no sobrepasa un valor umbral de luminosidad se representará en negro (0) y en caso contrario, el píxel tendrá el valor máximo de luz (255) y se representará de color blanco (ecuación 1.4). Se puede utilizar el mismo criterio pero de forma inversa (ecuación 1.5), o incluso empleando un umbral determinado por valores de brillo intermedios (ecuación 1.6). Con esta técnica se pretende diferenciar los distintos objetos representados en la imagen de lo que se considere el fondo [1]. Gracias a la umbralización se simplifica la cantidad de datos, provocando que las operaciones posteriores que se realicen con la imagen sólo se manejen valores lógicos (0,1), disminuyendo considerablemente su coste computacional.

$$g(x,y) = \begin{cases} 1 & si \quad f(x,y) > Umbral \\ 0 & si \quad f(x,y) \leq Umbral \end{cases} \tag{1.4}$$

$$g(x,y) = \begin{cases} 1 & si \quad f(x,y) < Umbral \\ 0 & si \quad f(x,y) \geq Umbral \end{cases} \tag{1.5}$$

$$g(x,y) = \begin{cases} 1 & si \quad Umbral1 < f(x,y) < Umbral2 \\ 0 & resto \quad de \quad casos \end{cases} \tag{1.6}$$

Filtro de paso alto

Figura 1.5: Imagen umbralizada

Cada uno de estos filtros, denominados lineales, basan su procedimiento en aplicar a la imagen, máscaras de convolución que se definen como una matriz que va desplazando su centro de píxel a píxel modificando los valores de cada uno de estos. El cálculo de los coeficientes que componen la máscara, por tanto, determinan el tipo de filtro u operación que se aplicará a la imagen. Para una mayor comprensión ver la ecuación 1.7 y su grafico 1.6 correspondiente.

$$y(i,j) = \frac{1}{D} \sum_{s=-a}^{a} \sum_{t=-b}^{b} x\,(s,t) \cdot f\,(i+s, j+t)\,(1.7)$$

Siendo f la función de transferencia, $x(s,t)$ el valor de la máscara y D el factor de división.

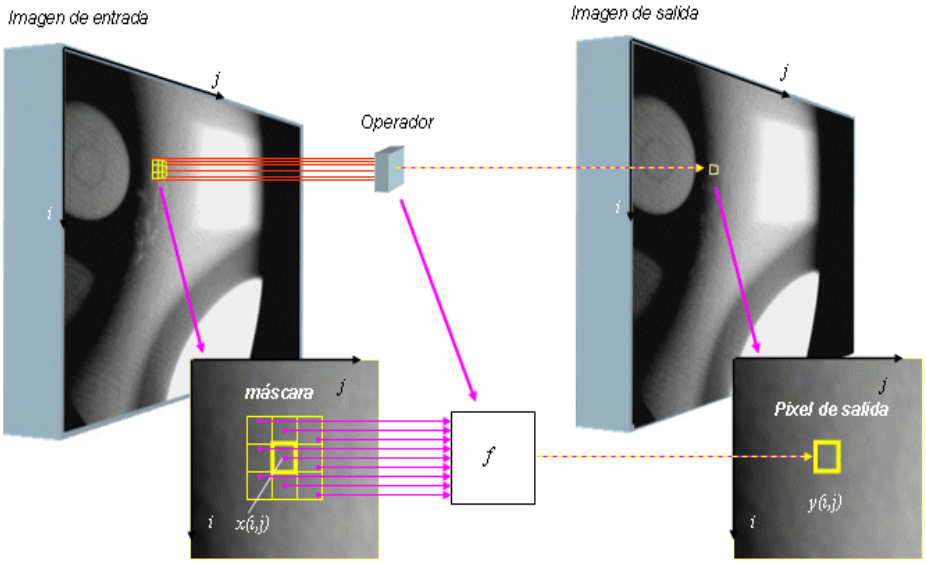


Figura 1.6: Gráfico de aplicación máscaras de convolución

El filtro de paso alto tiene como finalidad enfatizar diferencias y detectar bordes. En el presente trabajo se utiliza el filtro para buscar y representar las variaciones de intensidad lumínica o discontinuidades en los niveles de brillo que se corresponden a la frontera de los objetos [6]. De este modo se encuentran e identifican cada uno de los objetos presentes en la escena (ver figura 1.4). El filtro es aplicado tras haber realizado la umbralización consiguiéndose una reducción, aún más notable, en la cantidad de información de la imagen ya que solo serán representadas estas regiones fronterizas.

El método más común para la búsqueda de discontinuidades consiste en la correlación de la imagen con una máscara. Se calculará el producto de los elementos de la máscara por el valor de gris de los píxeles de la imagen encerrados por esta. Los valores de la máscara vienen determinados por aproximaciones discretas de la primera o segunda derivada de los niveles de grises de la imagen. La primera derivada en cualquier punto de la imagen vendrá dada por la magnitud del gradiente, mientras que la segunda derivada vendrá dada por la transformada de Laplace (ver figura 1.7).

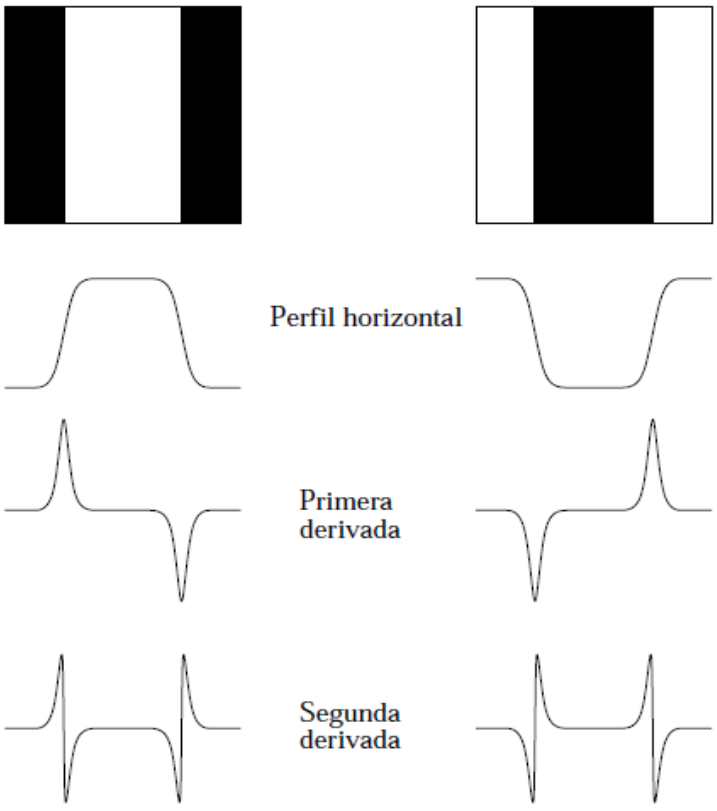
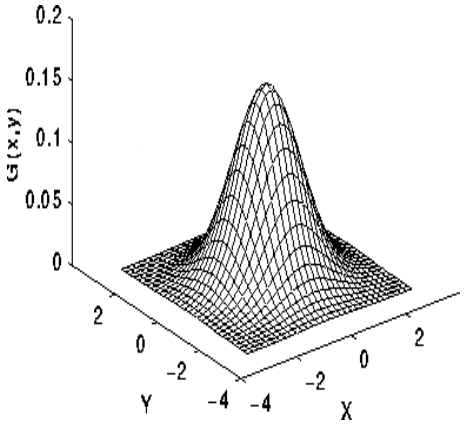


Figura 1.7: Gráfico de las derivadas aplicadas para detección de bordes

Filtro de paso bajo

Dentro de los filtros de paso bajo, cuya finalidad es reducir el ruido o eliminar pequeños detalles de una imagen, se ha utilizado para el desarrollo del proyecto un filtro de Gauss (ver cuadros 1.3 y 1.4). La idea principal del funcionamiento del filtro es reemplazar el valor de cada píxel por el promedio de los niveles de gris de los píxeles vecinos que se sitúen bajo la máscara de convolución. Los pesos de la matriz o máscara son calculados a partir del valor del píxel central por medio de la función de distribución gaussiana. Como se puede deducir en el cuadro 1.2 el peso de los píxeles vecinos disminuye a medida que se aleja del centro. En esa función de distribución σ será el valor de la desviación típica que determine el grado de suavizado que tendrá la imagen de salida. Este valor debe ser adecuado ya que si es demasiado grande provocará la obtención de una imagen desenfocada y si es reducido los cambios seran imperceptibles.

$$g(x,y)_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{x^2+y^2}{\sigma^2}\right)} \quad (1.8)$$



a)

$$\frac{1}{273}$$

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

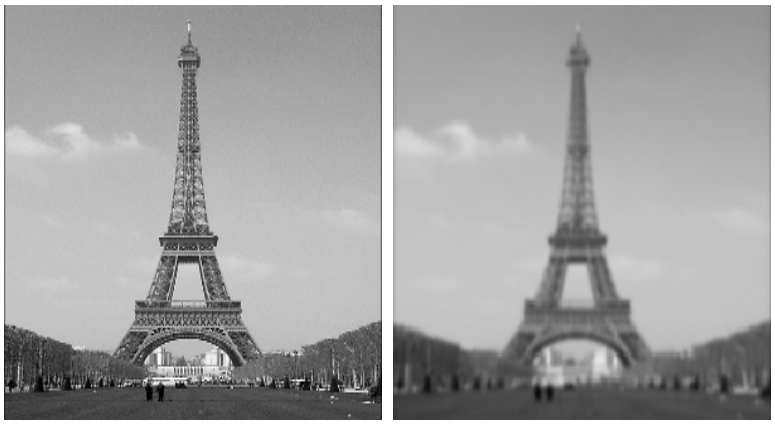
b)

Cuadro 1.2: a) Representación gráfica de la distribución de Gauss; (b Máscara de convolución

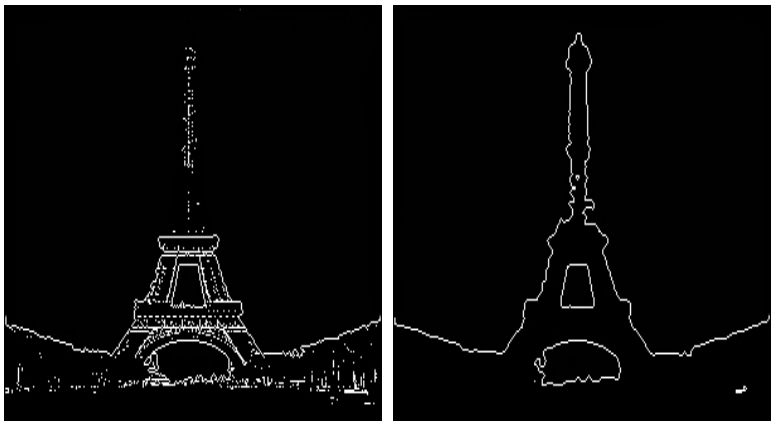
Reseñar que este tipo de filtros se utilizan como operación de preprocesamiento de la imagen, al igual que la conversión a escala de grises y el escalado.

1.2.1.2. Técnicas de procesamiento dinámicas

Aparte de los métodos de procesamiento de imágenes, en este proyecto también se necesita conocer las diferencias que se producen entre varios *frames* de vídeo. Con ello se podrá detectar el movimiento presente en la secuencia de vídeo, característica imprescindible para poder generar una idea sobre los cambios que se producen en un entorno.



Cuadro 1.3: Ejemplo aplicación filtro de Gauss



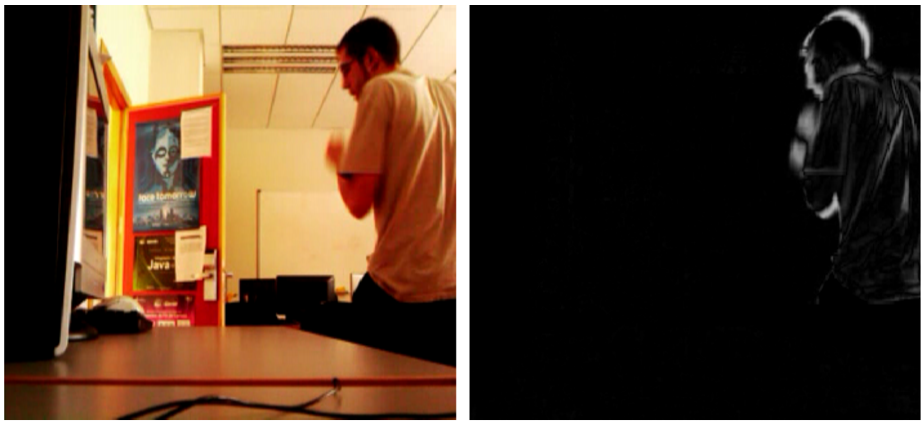
Cuadro 1.4: Imagen con detección de bordes aplicando o sin aplicar el filtro de Gauss

Imagen diferencia

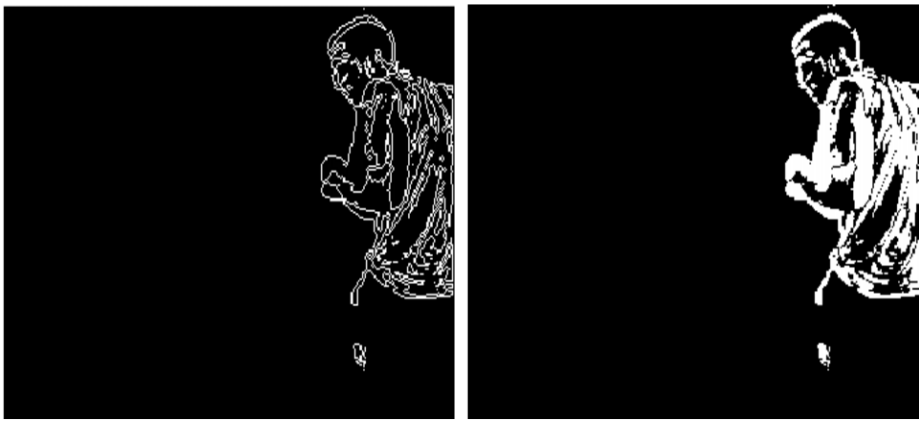
Este método genera una imagen cuyos valores de luminosidad serán la resta absoluta de los valores de dos *frames* de vídeo diferentes(ver figura 1.5). Los *frames* comparados suelen ser dos consecutivos en la secuencia de vídeo. De este modo la detección de movimiento no depende de cambios de iluminación ni de cambios de posición de la cámara. Con este sistema se representan las diferencias que se producen entre dos imágenes pero no se identifica el objeto que realiza el movimiento. Esto provoca que si un objeto que se encuentra en movimiento, en un momento dado, se queda estático, dejaría de ser representado en imagen. Por último apuntar que se trata de una operación sencilla y que por tanto, tiene un coste computacional bajo.

Tras calcular la imagen diferencia esta puede ser umbralizada o incluso se le puede aplicar un filtro de detección de bordes a continuación, para obtener un resultado menos complejo y en el que se diferencien claramente los cambios (ver figura 1.6).

$$g(x,y) = |f(x,y) - h(x,y)| \tag{1.9}$$



Cuadro 1.5: Ejemplo de imagen diferencia



Cuadro 1.6: Imagen diferencia con detección de bordes o umbralización

Sustracción de fondo

A través de la sustracción de fondo se obtiene una referencia de la parte fija de la imagen, de modo que sólo se represente las partes que se consideren móviles, aunque en ciertos instantes no produzcan movimiento alguno. En primer lugar, se debe obtener una imagen de referencia, en la que no deben de existir objetos en movimiento. Conforme se vayan obteniendo *frames* del vídeo, se comparará cada nueva imagen con la de referencia. En el caso de encontrarse regiones con diferencias significativas serán representadas, al ser consideradas parte móvil, en la imagen de salida.

La dificultad de aplicar este método al trabajo viene determinada porque la cámara no será fija. Dado que se pretende comprobar la utilidad de este proyecto en situaciones de movilidad, la mayoría de pruebas deben producirse en movimiento. Por lo tanto, a

la hora de determinar un fondo se deberá tener en cuenta los cambios de iluminación, cambios en el propio fondo, movimientos de la cámara, entre otros factores. Por lo explicado anteriormente se ha utilizado un método de sustracción de fondo en el que se actualizará constantemente los valores que determinan si un píxel forma parte o no del fondo. Para calcular esta condición se utiliza un método estadístico conocido como mezcla Gaussiana [9].

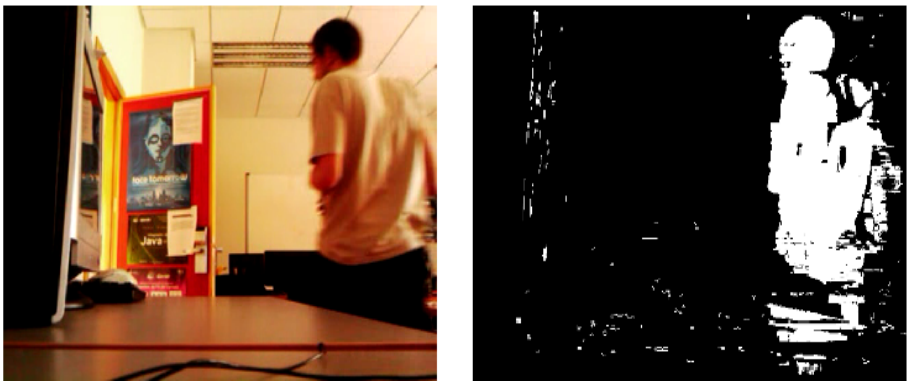


Figura 1.8: Imagen obtenida con sustracción de fondo

1.2.2. Sonido

El sonido se define como la propagación de una perturbación en un medio, en forma de onda, generada por el movimiento vibratorio de un cuerpo (ver figura 1.9). Estas perturbaciones llegan hasta nuestros oídos, convertiéndose en ondas mecánicas, que posteriormente son transportadas por el sistema nervioso hasta el cerebro donde se produce su interpretación.

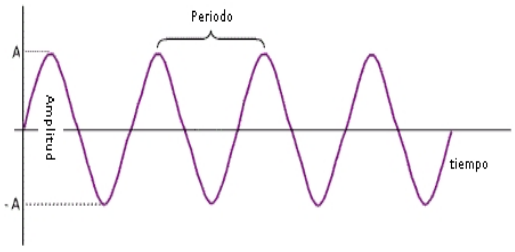


Figura 1.9: Forma y características básicas de una onda periódica

En la aplicación, existirá la necesidad de asociar las diferentes característica del sonido a la información que se obtenga de las imágenes para que se pueda generar sonido coherente. Estas características son:

- Tono: Es la característica por la cual se distingue entre sonido grave y agudo. El tono viene determinado por la frecuencia, que se define como el número de oscilaciones de la onda sonora por unidad de tiempo, y se mide en herzios (hz). Cuanto mayor sea la frecuencia más agudo será el sonido y en caso contrario más grave. El margen de frecuencias que tiene el oído humano se sitúa entre los 20 y los 20000 Hz.

- Intensidad: Propiedad del sonido que determina si lo que se escucha tiene más o menos volumen. Se define como la energía que atraviesa, por unidad de tiempo, una superficie dispuesta perpendicularmente a la dirección de propagación. La intensidad de una onda sonora es proporcional al cuadrado de su frecuencia y al cuadrado de su amplitud, y disminuye con la distancia al foco. Su unidad de medida es el decibelio (db) y su valor debe sobrepasar 0 decibelios para ser perceptible y no llegar a 140, donde se sitúa el umbral del dolor del oído humano.

- Timbre: Propiedad que nos permite distinguir sonidos provenientes de diferentes instrumentos aunque posean el mismo tono e intensidad. Esto se produce gracias a que cada instrumento emite una frecuencia compuesta por una serie de armónicos únicos, que son pequeñas variaciones en la onda acústica, que los diferencian (ver figura 1.10).

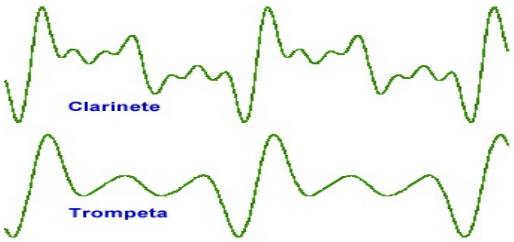


Figura 1.10: Ondas en las que se aprecian las variaciones armónicas

A continuación se explicarán los distintos formatos de sonido utilizados en el programa.

1.2.2.1. Formato WAVE (*Waveform Audio File Format*)

Formato de audio digital desarrollado y propiedad de Microsoft e IBM, variante del formato RIFF (*Resource Interchange File Format*). Sus datos suelen estar codificados en un formato sin compresión denominado como PCM (*Pulse-Code Modulation*). Este formato de datos contiene secuencias de bits correspondientes a los valores obtenidos en la conversión de la señal analógica del sonido, en forma de onda, a una señal digital. Con ello se consigue un sonido de alta calidad, pero el tamaño del fichero es considerablemente

mayor comparado con cualquier otro formato de audio con compresión.

Un fichero .wav organiza su contenido en pequeñas secuencias o *chunks* de datos, que tendrán cada uno de ellos un tipo de información específica. La división más importante es la que diferencia la cabecera del fichero de la región donde se almacenan los datos específicos del sonido. Esta última también estará dividida en dos subcampos, el "fmt" donde se almacena la información que describe el tipo de sonido y el campo de datos donde se almacena la codificación en secuencia de *bits* del sonido (ver figura 1.11).

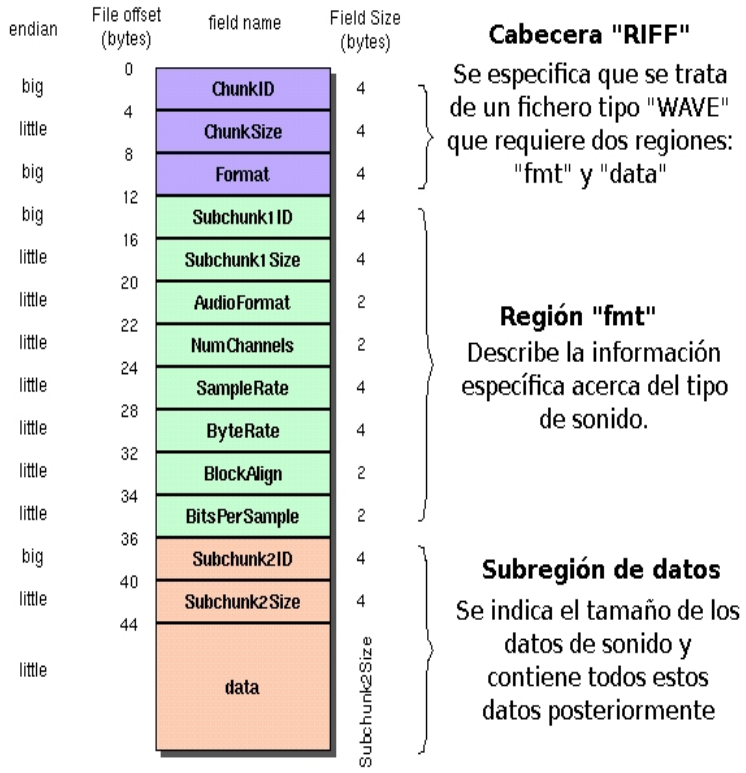


Figura 1.11: Forma canónica del formato WAVE

1.2.2.2. MIDI (*Musical Instrument Digital Interface*)

El MIDI es un protocolo de comunicación estándar, formado por un conjunto de comandos, que permite la comunicación entre diversos dispositivos como computadoras, sintetizadores o controladores. Estos comandos tienen como fin la generación de sonido a partir de su interpretación por parte de alguno de los dispositivos anteriores.

No podemos definir MIDI como un formato de sonido ya que, como se ha comentado anteriormente, no se guarda información de audio sino que guarda comandos en formato numérico relativos a la producción de sonido. Este formato numérico tiene como unidad

de medida el *byte*, existiendo dos tipos: de estado y de datos. Se diferencian por el valor del primer *bit*, siendo 1 ó 0 respectivamente. Los *bytes* de estado son aquellos que determinan el tipo de mensaje, por ejemplo notificar un cambio de nota en un canal o una asignación de instrumento nuevo en una pista. Los *bytes* de datos, por su parte, serán los que sucedan a los de estado para determinar los valores correspondientes al mensaje. Por ejemplo en la asignación de instrumentos el *byte* de estado tendrá que especificar los datos del instrumento con el que se generarán los sonidos.

Los archivos MIDI asimismo se organizan en bloques de datos, de forma similar a los ficheros WAVE (ver figura 1.12). El bloque de cabecera contiene información sobre parámetros constantes del sonido como el tipo de formato MIDI, número de pistas o la duración temporal del sonido. A su vez, cada pista estará formada con una pequeña cabecera, justo antes de la zona de almacenamiento de los distintos eventos. Los eventos MIDI serán todos aquellos *bytes* que contiene la descripción musical (inicio de una nota, cambio de instrumento, modificación de volumen o efectos) del sonido.

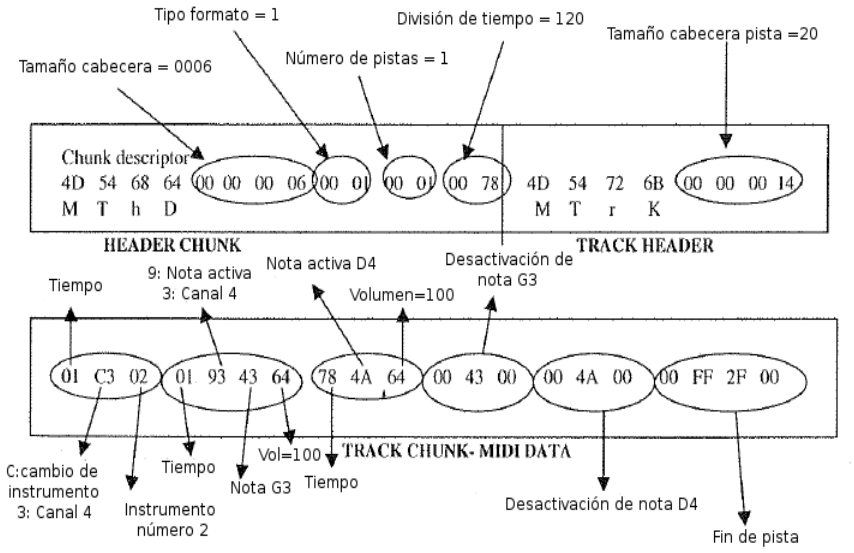


Figura 1.12: Formato MIDI

1.3. Herramientas utilizadas

1.3.1. OpenCV

Es una librería de visión artificial desarrollada por Intel, con una gran variedad de herramientas para el tratamiento de imágenes. Es compatible con la librería de procesamiento de imagen (IPL), también de Intel, que implementa operaciones sobre gráficos en bajo nivel. Está publicada bajo licencia BSD lo que permite su uso personal o comercial de forma gratuita. La librería esta escrita en lenguaje C y se puede usar tanto en Windows, Linux, MacOS o FreeBSD. Contiene más de 500 algoritmos con los que se puede realizar multitud de procesamientos de imagen. Alguna de las aplicaciones más importantes de OpenCV son: Sistemas de reconocimiento facial, detección de movimiento, identificación de objetos o visión estereoscópica.

Fue lanzada oficialmente en 1999 como una iniciativa de investigación de Intel para mejorar el rendimiento de aplicaciones con un uso intensivo de CPU relacionadas con la visión artificial. A partir del año 2000 se fueron publicando distintas versiones beta hasta que en 2006 se lanzó la primera versión 1.0 estable. Ya en 2009 se lanza su segunda versión estable que es la que se ha utilizado en este proyecto.

1.3.2. Gstreamer

Librería que permite la creación de aplicaciones multimedia capaces de reproducir audio o realizar tareas más complejas como mezclas de audio y vídeo. Se trata de una librería multiplataforma, de código abierto y escrita en lenguaje C utilizando la biblioteca GObject. Sus características principales son:

- Puede ser utilizado tanto en diferentes sistemas operativos (Linux, Windows, MacOS, Solaris, Symbian) como en diferentes procesadores o compiladores.
- Diseño orientado a objetos y basados en herencia.
- Multihilos y pipelines transparentes para el desarrollador.
- Traspaso de datos liviano lo que conlleva un alto rendimiento computacional.
- Carga dinámica de plugins.
- Soporta una gran variedad de formatos como por ejemplo MP3, Ogg/Vorbis, MPEG-1/2, AVI...

- Posibilidad de acceso a su API desde diversos lenguajes de programación como Python, Perl y Ruby.

1.3.3. QT

Librería multiplataforma para desarrollar interfaces gráficos y para crear aplicaciones como herramientas de consola o servidores. Qt esta escrita en lenguaje C++ aunque puede ser utilizada desde otros lenguajes (Java, Python, Ada, PHP...) a partir del uso de bindings. Está distribuida bajo licencia GNU LGPL, siendo codigo abierto y gratuito. El API de la biblioteca es muy amplio y gracias a sus funciones se puede entre otras cosas acceder a base de datos mediante SQL, acceder a ficheros XML, se pueden gestionar *threads* de ejecución, etc.

Gran número de aplicaciones utilizan esta librería, como pueden ser Google Earth, Skype, VirtualBox y sobretodo es conocida por ser utilizada por el entorno de escritorio KDE de Linux.

1.3.4. Eclipse

Eclipse es un entorno de desarrollo integrado (IDE, *Integrated Development Environment*). Está distribuido bajo términos de licencia pública de Eclipse (EPL), lo que le convierte en un software libre y gratuito. El proyecto fue creado en el año 2001 por IBM y primeramente era un entorno de desarrollo para aplicaciones Java (JDT). Posteriomente gracias a la posibilidad de añadir o crear distintos *plugins* se puede trabajar con diversos lenguajes de programación (C, C++, COBOL, Python, Perl, PHP...). De todas formas la mayor parte del código está escrito en Java, lo que le otorga la característica de ser un software multiplataforma. Actualmente es desarrollado por la llamada *fundación eclipse*, una organización independiente y sin ánimo de lucro.

Las características básicas que tiene este entorno de desarrollo son:

- Editor de texto con resaltado de sintaxis.
- Compilación en tiempo real.
- Integrado para realizar pruebas unitarias de JUnit.
- Refactorización de código y asistentes sencillos para creación de proyectos, clases, etc.

Capítulo 2

Objetivo

2.1. Finalidad del proyecto

El objetivo principal de este proyecto es la creación de un prototipo de software capaz de tratar imágenes de manera que se genere un sonido estructurado a partir de ellas. Ese sonido debe proporcionar suficiente información para que una persona, después de escucharlo, sea capaz de realizar una descripción de ciertas características de la imagen de entrada.

2.2. Objetivos parciales

Para la consecución del objetivo final del proyecto se han tenido que superar otra serie de objetivos menores en su desarrollo que a continuación se enumerarán en orden cronológico:

- Familiarización con el entorno de desarrollo Eclipse e integración de las distintas librerías a utilizar dentro de dicho entorno de desarrollo.
- Estudio del lenguaje de programación a utilizar, en este caso C++, además de estuidar las distintas funcionalidades de las librerías tanto de OpenCV como de Gstreamer.
- Analizar y comprender las diversas técnicas de procesamiento de imágenes. En primer lugar, haciendo uso de funciones para el tratamiento de imágenes para continuar después con el procesamiento de *frames* adquiridos de un vídeo o cámara.
- Conocer la estructura básica de los ficheros de audio para generarlos correctamente con los datos obtenidos de las imágenes.
- Conseguir la reproducción del fichero de sonido que previamente se ha generado.

- Generación de una interfaz con la librería QT que posibilite la realización de diversas pruebas, cambiando datos de configuración, de forma rápida y que permitiera la visualización tanto de la imagen original como la resultante de aplicar las distintas operaciones sobre ella.
- Realizar diferentes experimentaciones con imágenes y vídeos para poder llegar a conclusiones que favorezcan la optimización del funcionamiento de la aplicación.

2.3. Etapas

El proceso de funcionamiento de la aplicación, dividido etapas, es el siguiente:

- Adquisición de la imagen a través de un fichero o a través de la captura de *frames* de un vídeo o cámara.
- Etapa de preprocesamiento donde se aplican los tratamientos de imágenes con los que se eliminan detalles no necesarios o se determinan y realzan las regiones de interés.
- Fase de segmentación en la que se intentan aislar las regiones de interés del resto de la imagen.
- Reconocimiento y clasificación de las distintas características presentes en la imagen después de haberla modificado, o no, en las fases anteriores.
- Generación del fichero de audio a partir de los datos obtenidos en la fase de reconocimiento.
- Reproducción del sonido.

En el caso de utilización de vídeo este proceso se tendría que repetir de forma ininterrumpida. Hay que tener cuenta que, en ese caso, durante todo el proceso debe existir un *thread* de ejecución que vaya obteniendo los *frames* del vídeo.

Capítulo 3

Descripción Informática

En estas secciones se presentará todo lo relacionado con el desarrollo e implementación de la aplicación.

3.1. Requisitos

A continuación se especificarán los requisitos software que se necesitan cumplir para el correcto funcionamiento del proyecto.

3.1.1. Requisitos funcionales

Características requeridas del sistema que expresan las capacidades de acción del mismo, y definen su funcionalidad determinando qué debe hacer la aplicación para ser válida.

- La aplicación debe ser capaz de analizar y modificar las imágenes de entrada según los criterios introducidos en la interfaz, consiguiendo información útil para crear los ficheros de audio de salida.

- A la hora de obtener las imágenes de un fichero de vídeo o a través de la cámara, se tendrán que comparar algunas de estas para poder detectar los cambios que tienen lugar en la escena, y por tanto conocer el movimiento que se produce en la escena. Además debe existir un proceso paralelo al principal que vaya capturando *frames* a la velocidad indicada desde la interfaz.

- Para la sonificación de imágenes en escala de grises se generará un fichero de audio WAVE y para imágenes en color un fichero MIDI.

- Se deberá poder reproducir un archivo de audio, cuyas extensiones sean .wav o .mid.

3.1.2. Requisitos no funcionales

Características requeridas del sistema, del proceso de desarrollo, que no definen la funcionalidad de la aplicación aunque son necesarios para un funcionamiento correcto de esta.

- Se requiere que el sistema tenga acoplado una webcam, a través de la cual se obtenga la secuencia de vídeo en tiempo real.
- El sistema operativo sobre el que se ejecuta la aplicación debe ser UNIX.
- El equipo donde se ejecute la aplicación debe tener arquitectura Intel, para el correcto funcionamiento de OpenCV.

3.2. Diseño e implementación

Durante este apartado se explicará con detalle como se realiza la tarea de sonificación de una imagen digital a partir de diferentes características de esta.

3.2.1. Tratamiento de imágenes

Antes de analizar la imagen, transformando sus datos en un fichero WAVE o MIDI, se deben realizar diversos tratamientos de imagen de modo que se consiga una selección de información de acuerdo con una serie de datos que se obtienen desde la interfaz gráfica. A continuación se explica el proceso de implementación, así como las principales funciones de OpenCV que se han utilizado en él:

3.2.1.1. Captura

Primeramente hay que obtener la imagen de un fichero, con `cvLoadImage()`. En el caso de obtener las imágenes desde vídeo o cámara primero hay que crear un tipo de datos cvCapture con `cvCaptureFromFile()`o `cvCaptureFromCAM()`respectivamente. Posteriormente se irán capturando los *frames* a partir del tipo de dato anterior mediante la función `cvQueryFrame(cvCapture* capture)` .

Hay que señalar que cuando la aplicación maneja ficheros de vídeo o se obtienen imágenes de una cámara debe funcionar con varios procesos en paralelo. Uno se encargará del proceso principal mientras que el otro irá capturando los *frames* del vídeo controlando y ajustando la velocidad de reproducción. Para crear un proceso hijo del principal en C++ se consigue mediante la ejecución de `pthread create(thread, NULL,`

`getFrame, (void*) ar) != 0)` . Una vez creado el proceso se debe controlar las posibles condiciones de carrera, que pueden darse por compartir variables, mediante sincronización con *locks* usando `pthread mutex lock(ar.mutex1)` y `pthread mutex unlock(ar.mutex1)`.

3.2.1.2. Preprocesamiento

Fase se la que se intenta mejorar la calidad de la información que aporta la imagen según qué se quiera obtener posteriormente de ella.

Reducción rango dinámico de color

Esta tarea debe realizarse siempre que el modo elegido en la interfaz no sea igual al de búsqueda de color. Puede realizarse de varios modos. El primero de ellos consiste en señalar, en la función de carga de una imagen desde un fichero, que se inicialice en escala de grises. El otro es el que se utiliza en el caso de obtenerse las imágenes de un vídeo y se consigue ejecutando la función `cvCvtColor()`indicando en el último campo la conversión a desear, en este caso RGB2GRAY.

Suavizado

La función de suavizado se centra en única instrucción `cvSmooth()`a la que se indicará que el tipo de filtro sea Gaussiano, con el argumento `CV GAUSSIAN`, y se indicará también el coeficiente de desviación típica deseado. Puede ser utilizada tanto para imágenes en escala de grises como en color, aunque en esta aplicación solo es aplicado en el primero de los casos.

Escalado

Para implementar el escalado el primer paso es crear una nueva imagen vacía. Se le asignará los valores de dimensión a los que se pretende modificar la imagen original, pero con la misma profundidad y mismo número de canales de esta. Luego se copia la imagen original en la recién creada, indicando el tipo de escalado, de modo que tendremos una copia idéntica pero con dimensiones distintas. La instrucción que permite esta operación es `cvResize()`.

3.2.1.3. Segmentación

Fase en la que mediante distintos métodos se divide la información de la imagen en distintas áreas con significado o regiones de interés.

Umbralización

La umbralización de una imagen mediante un umbral de luz, es necesaria para diferenciar las diferentes regiones de interés, separando el fondo de los objetos. Para llevarla a cabo se ejecuta `cvThreshold()`. Hay que determinar el valor umbral y el tipo de umbralización que se va a realizar. Para este trabajo se utiliza el argumento `CV THRESH BINARY` .

Detección de bordes

Este método de segmentación es utilizado después de realizar el umbralizado. Se basa en la ejecución de la instrucción `cvFindContours()`con la que obtendremos en el tipo de dato `cvSeq**` todas las referencias a los contornos encontrados en la imagen. El último paso será dibujar los contornos referenciados con la variable anterior en una imagen con `cvDrawContours()`

Imagen diferencia

Conseguir la imagen diferencia entre dos *frames* gracias a OpenCV es sencillo, basta con aplicar la función `cvAbsDiff()`, teniendo en cuenta que las dos imágenes de entrada habrán sido codificadas en escala de grises previamente. Se comparará la última imagen obtenida con la última sonificada, no siendo *frames* consecutivos tal y como se explicaba que solían ser en la sección de introducción. Por ello si el tiempo de sonificación es amplio, en principio, aparecerán más diferencias y no habrá sensación de continuidad en la percepción. Esto produce la necesidad de que los tiempos fijados para el audio, en caso de ejecutar vídeo, sean cortos.

Sustracción de fondo

Para la sustracción la primera acción que hay que realizar será fijar un primer *frame* de referencia que será el primero que se obtenga del vídeo o de la cámara, mediante la función `cvCreateGaussianBGModel()`. La variable que devuelve la función es un tipo de datos `CvBGStatModel*`. Este tipo de dato guarda una referencia a lo que considera imagen de fondo `(CvBGStatModel* backgroundmodel→background)` y otra `(CvBGStatModel* backgroundmodel→foreground)` en la que se representa las partes móviles en cada momento. Esta última imagen será la que interese en la etapa de reconocimiento. Como el vídeo de entrada tendrá constantes cambios de iluminación y de posición, con cada nueva imagen a sonificar se vuelve a actualizar la imagen referencia de fondo con `cvUpdateBGStatModel()`. Al igual que con la imagen diferencia la actualización del fondo depende en gran medida de la duración que tenga el sonido. Se intentó realizar la actualización por cada nuevo *frame* que se obtenía en el proceso hijo, mientras el proceso

principal trataba la imagen, pero se ralentizaba la reproducción del vídeo.

3.2.1.4. Reconocimiento o clasificación

Este será el apartado en el cual se estudiará la imagen, según una serie de criterios prefijados. En las siguientes secciones, que explican la generación de los ficheros de sonido, se detallará como se realiza el estudio.

3.2.2. Generación fichero tipo WAVE

Primeramente, tanto en la creación del fichero WAVE como para el MIDI, la imagen de entrada se divide en filas y columnas (ver cuadro 3.1). En este caso la imagen debe estar codificada en escala de grises. Después de que se le haya aplicado los tratamientos explicados anteriormente a la imagen de entrada, se comienza con el estudio de la imagen columna por columna de izquierda a derecha. De cada cuadrante, que forma la columna, se obtiene el nivel de gris medio que contienen sus píxeles. Este valor será el que determine la intensidad o volumen de la onda correspondiente a esa región de la imagen. Por otra parte, según en que posición se sitúe el cuadrante respecto a la vertical se le asignará un valor de frecuencia distinto. El margen de frecuencias fijado en el fichero de configuración se sitúa en 2000Hz (sonido agudo) en la parte superior y 100Hz (sonido grave) en la parte inferior.

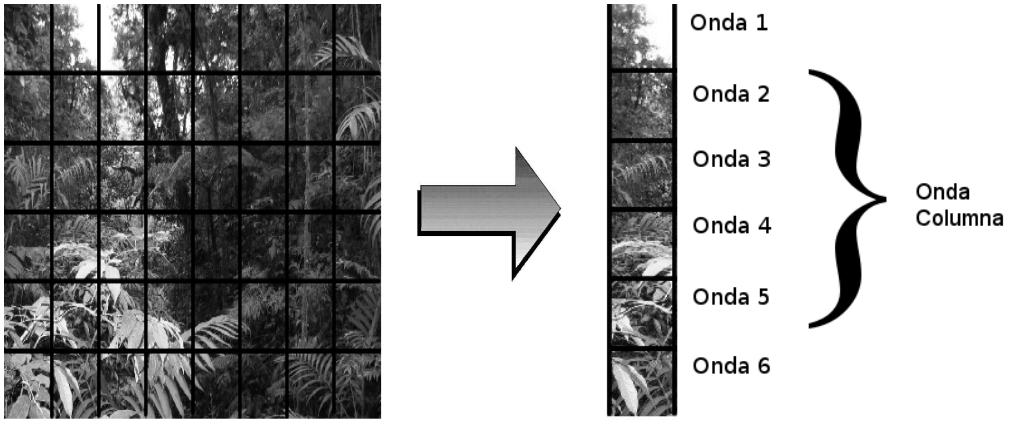
La última variable que falta para poder calcular la onda en cada cuadrante es el tiempo. Para obtener este valor primeramente hay que establecer el tiempo de reproducción para cada columna mediante la división de la duración total del sonido entre el número de columnas. El siguiente paso será calcular la diferencia de tiempo que existe entre *samples* de sonido, que serán cada uno de los datos que se escribirán en el fichero de audio posteriormente. Esto se consigue realizando la división de la cantidad de *samples* de sonido determinados a cada columna entre el tiempo asignado a cada columna. Señalar que el número de muestras o *samples* por segundo esta inicializado a 44100.

Con todo esto la función para calcular la onda es la siguiente [8]:

$$O(t) = A\left(I\left(y\right)\right) \cdot \sin\left(\omega\left(y\right) \cdot t\right) \tag{3.1}$$

Siendo O(t) el valor de la onda en cada instante, A el valor de la amplitud relacionado con el nivel de gris y ω el valor de la frecuencia que se asigna según la posición vertical.

La implementación se realiza de la siguiente manera. Por cada cuadrante se declara



Cuadro 3.1: Esquema del proceso de sonificación

un tipo de dato array de tamaño igual al número de *samples* o muestras por columna. En este array se irán almacenando los resultados de la función anterior en cada uno de los instantes de tiempo, determinando el valor de la onda en cada uno de ellos. Cuando esté procesada toda la columna y por tanto todos los arrays llenos se procede al cálculo de la onda de salida. Para ello se deberá sumar todos los arrays (ver figura 3.1), en uno del que se irán obteniendo posteriormente los datos que se escribirán de forma secuencial en el fichero de audio.

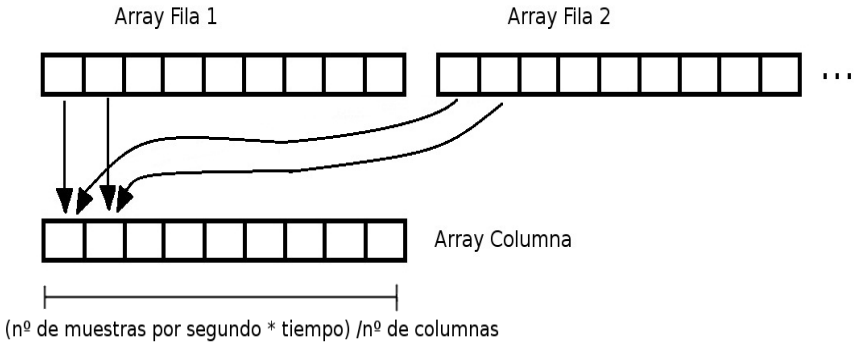


Figura 3.1: Suma de Arrays

Para determinar el comienzo y el fin de la sonificación de una imagen se escribirá en el fichero de audio una serie de *samples* que producirán el sonido de inicio y otro de fin con una frecuencia igual a 1200Hz.
La escritura de la cabecera y de los demás componentes del fichero de datos se realiza a partir de una librería de C++ encargada de ello. Está escrita entre 1996 y 2002 por Timothy J. Weber y su código es libre y puede ser descargado gratuitamente. Las funciones utilizadas de esta librería son las siguientes:

- `Wave.OpenWrite("file.wav").` Apertura del fichero.

- `Wave.SetupFormat(sampleRate, bitspersample, numChannels).` Escritura de la cabecera RIFF y la región "fmt" (Ver cuadros 3.2 y 3.3) con los valores de frecuencia de muestreo, número de bits por canal y el número de canales.

- `Wave.WriteSample(totalintensity[m][n]).` Escritura de cada uno de los valores de intensidad de la imagen (Ver cuadro 3.4).

- `Wave. WaveFile().` Cierre del fichero y liberación de memoria correspondiente al objeto Wave.

Un ejemplo de estructura de fichero WAVE aparece en la figura 3.2.

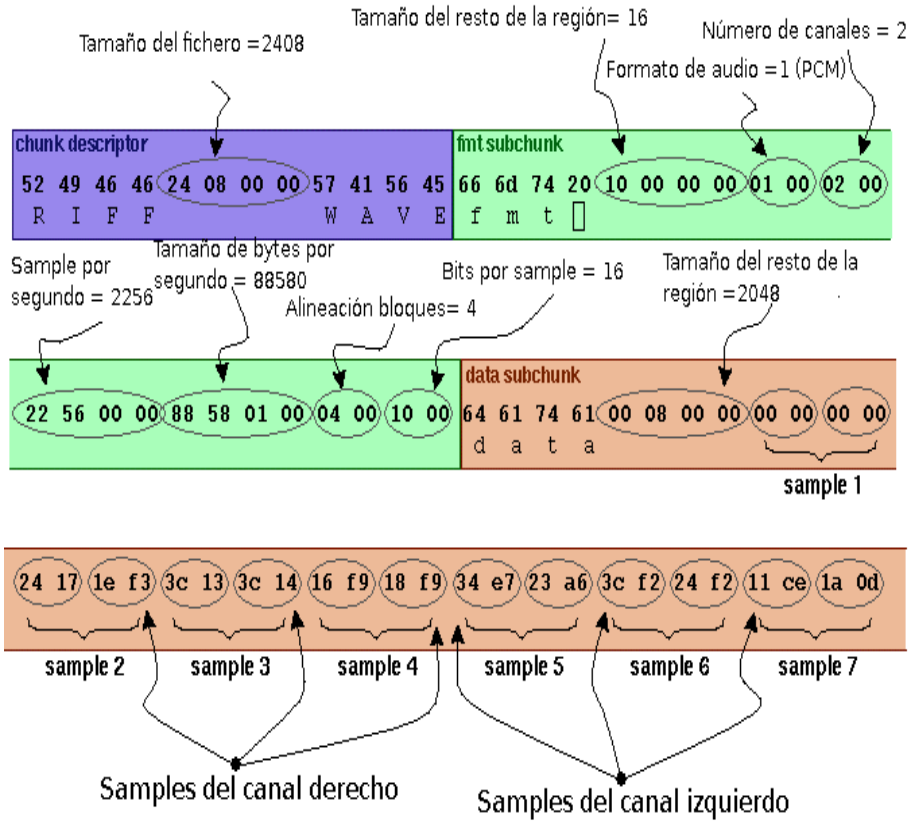


Figura 3.2: Organización de los *bytes* en el fichero WAVE

Offset	Tamaño	Nombre	Contenido
0	4 bytes	ChunkID	'RIFF' en codficación ASCII
4	4 bytes	Chunk Size	Tamaño del fichero - 8
8	4 bytes	Format	'WAVE'

Cuadro 3.2: WAVE: Cabecera RIFF

Offset	Tamaño	Nombre	Contenido
12	4 bytes	SubChunkID	'fmt' en codficación ASCII
16	4 bytes	SubChunk1Size	16 si el formato de codificación es PCM Número de *bytes* que quedan de la región fmt.
20	4 bytes	AudioFormat	0x0001 correspondiente al formato de codificación PCM
22	2 bytes	NumChannels	1:mono 2:estereo
24	4 bytes	SampleRate	Frecuencia de muestreo por ejemplo: 44100(hz)
28	4 bytes	ByteRate	Frecuancia de muestreo * valor de BlockAlign
32	4 bytes	BlockAlign	Tamaño de bloque que viene determinado por: Canales * bits/frecuencia de muestreo / 8
34	2 bytes	BitsPerSample	8 o 16

Cuadro 3.3: WAVE: Región del fichero que describe el formato del sonido

Offset	Tamaño	Nombre	Contenido
36	4 bytes	SubChunk2ID	'data'
40	4 bytes	SubChunk2Size	Tamaño de los datos de sonido
44	n bytes	Data	Secuencia de valores que componen el sonido.

Cuadro 3.4: WAVE: Región de datos del audio

3.2.3. Generación fichero tipo MIDI

El método de generación del fichero con comandos MIDI se asemeja bastante al anterior, aunque el tipo de datos que se utiliza es distinto. En este caso también se realiza un barrido de izquierda a derecha de las columnas de la imagen. Al contrario que en la generación del fichero de formato WAVE, se utilizará una imagen en color teniendo que analizar los valores correspondientes a cada color primario que la conforma (rojo, verde y azul). La intensidad se determina de forma proporcional a la semejanza del color buscado con el encontrado. Si es igual, la intensidad será máxima (1.0) y según se vaya distanciando los valores de color la intensidad disminuye. En concreto, cada unidad que se aleje el color buscado del encontrado es penalizado con -0.01 en el valor de la intensidad.

Por otra parte, al igual que en la generación del fichero WAVE, el tono del sonido dependerá de la posición del cuadrante en el eje vertical, pero en este caso no se determinará con un valor de frecuencia sino que se le asignará a cada fila una nota de la escala cromática diferente (ver figura 3.3). La nota central de la imagen será la referenciada con el número 69 en MIDI que equivale a un La con una frecuencia de 440Hz. Cada fila tiene un semitono de distancia con sus adyacentes. Cada fila que se sitúe inferiormente a la central será un semitono más grave que la anterior, y si es una fila superior será medio tono más alto, es decir más agudo.

MI4	FA4	FA#4/SOLb4	SOL4	SOL#4/LAb4	LA4	LA#4/SIb4	SI4	DO5	DO#5/REb5	RE5	RE#5/MIb5
64	65	66	67	68	69	70	71	72	73	74	75

Figura 3.3: Escala cromática

La novedad de este proceso es que se incluye la última característica básica del sonido, el timbre. La aplicación permite la opción de escoger la búsqueda de hasta tres colores de forma simultánea, de modo que a cada uno de ellos se les asigna un instrumento distinto para su identificación.

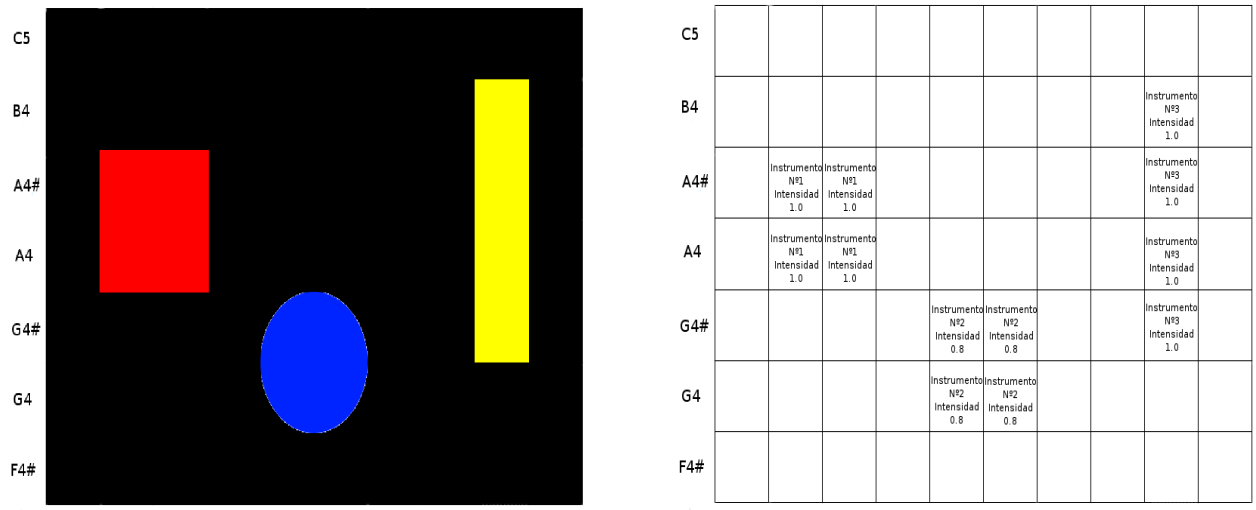
Para cada columna se escribirán en el fichero tantas notas como filas cuyo valor de color sean igual o semejante a alguno de los buscados, teniendo en cuenta el instrumento correspondiente asociado (ver cuadro 3.5). La variable del tiempo, en este caso, estará fijada por el resultado de multiplicar el número de columnas por el tiempo asignado a cada una de ellas. Este valor determina el punto de inicio y fin del sonido y será utilizado como argumento del evento de activación o desactivación de nota.

Esta información se va introduciendo en el fichero de audio en forma de comandos, que serán interpretados por el dispositivo controlador para reproducir sonido. Hay que enunciar que a cada fila se le asignará una pista MIDI independiente para evitar problemas a la hora de reproducir sonidos simultáneos. De la misma manera que el en fichero WAVE se introducen unos sonidos al inicio y fin como referencia, en este caso, se añade una nota de saxofón que haga la misma función. En la generación de MIDI se utiliza una librería de C++ específica para manejar y escribir este tipo de ficheros. Está realizada por J.D. Koftinoff Software y se distribuye con licencia GNU GENERAL PUBLIC LICENSE. Los comandos básicos utilizados de esta librería son:

- `MIDIFileWriteStreamFileName outstream(‘‘file.mid’’).` Creación del objeto que abre el fichero donde luego se escribirá toda la información. En el cuadro 3.6 puede verse la distribución que tendrá la cabecera propia de este fichero.

- `MIDIMultiTrack tracks.` Creación del objeto que organiza la escritura de los eventos en cada una de las pistas. La cabecera propia para cada pista puede verse en el cuadro 3.7.

- `MIDITimedBigMessage r.` Objeto que identifica el tipo de mensaje que se escribirá en el fichero. Se ha declarado uno por cada color primario, para poder controlar la escritura de los distintos eventos de cada instrumento.



Cuadro 3.5: Ejemplo de sonificación para MIDI

- `r.SetProgramChange(0,inst1).` Comando a través del cual se cambia el instrumento que se asignará a una pista determinada, indicando su número de referencia.
- `r.SetTime(timeallocated*240+SG).` Función que determina el valor del tiempo. Se debe utilizar tanto antes de activar una nota como para finalizarla.
- `r.SetNoteOn( 0, note, 64 ).` Activación de una nota en cierto canal.
- `r.SetNoteOff( 0, note, 0 ).` Terminación de una nota en un canal determinado.
- `r.SetControlChange(0, 7, 127*result).` Fijar la intensidad asociada a la pista.
- `tracks.GetTrack((j/pheight)+1)->PutEvent( r ).` Inclusión de un nuevo evento en cierta pista.
- `MIDIFileWriteMultiTrack writer( tracks, outstream ).` Creación del objeto que permitirá ejecutar la acción de escribir los datos al fichero.
- `writer.Write( numtracks, division ).` Función del objeto anterior utilizada para realizar la escritura en el fichero, al que hay que aportar el número de pistas que se han utilizado y la división temporal en la que esta fragmentado un segundo de sonido.

Un ejemplo de lo que contiene un fichero MIDI generado por la aplicación aparece en la figura 3.4.

Offset	Tamaño	Nombre	Contenido
0	4 bytes	ChunkID	"MThd"(0x4D546864)
4	4 bytes	Chunk Size	6 (0x00000006)
8	2 bytes	Format type	0 - 2
10	2 bytes	Number of tracks	1 - 65,535
12	2 bytes	time division	Número de *frames* por segundo

Cuadro 3.6: MIDI: Cabecera

Offset	Tamaño	Nombre	Contenido
0	4 bytes	ChunkID	"MThd"(0x4D546864)
4	4 bytes	Chunk Size	Este dato varía según el tamaño de los datos de la pista
8	n bytes	track event data	Sucesión de *bytes* con datos referentes a la pista

Cuadro 3.7: MIDI: Cabecera de pista

3.2.4. Reproducción ficheros WAVE y MIDI

Para explicar cómo se ha implementado la reproducción de los ficheros de audio primero hay que hacer una pequeña introducción de los componentes fundamentales de la librería que se utiliza para ello en el proyecto.

Gstreamer contiene como objeto fundamental los denominados elementos. Estos son funciones independientes que realizan, cada una de ellas, una tarea multimedia específica. Según su finalidad puede encontrarse elementos de entrada o salida de datos, decodificadores o codificadores y multiplexores o demultiplexores. Los diferentes elementos pueden estar unidos mediante enlaces denominados *pads*, para que la salida de una función comunique con la entrada de otra. El último componente básico se conoce como *bin*, o *pipeline*, y será donde se agrupen los diversos elementos, unidos mediante *pads*, de tal manera que su ejecución conjunta realice una tarea determinada, como puede ser la de reproducir un vídeo.

En la figura 3.5 se puede apreciar el conjunto de elementos necesarios para la reproducción de un fichero WAVE.

- Filesrc: Se encargará de obtener de la apertura y obtención de datos de un fichero de audio.
- Wavparse: Descodifica el contenido de un fichero .wav.
- Volume: Elemento regulador del volumen de salida.
- Alsasink: Elemento que hace posible la comunicación con el dispositivo de salida.

```
Header: Type=1 Tracks=14 Division=120
Start Track #0
Time:      0:  0    End Of Track
End Track   #0
Start Track #1
Time:      0:  0    End Of Track
End Track   #1
Start Track #2
Time:      1:105    Ch  2  NOTE ON     Note  69  Vel   64
Time:      1:105    Ch  2  PG CHANGE   PG    56
Time:      1:105    Ch  2  CTRL CHANGE Ctrl   7  Val   97
Time:      2:  1    Ch  2  NOTE OFF    Note  69  Vel    0
Time:      2:  1    End Of Track
End Track   #2
Start Track #3
Time:      1: 80    Ch  2  NOTE ON     Note  68  Vel   64
Time:      1: 80    Ch  2  PG CHANGE   PG    56
Time:      1: 80    Ch  2  CTRL CHANGE Ctrl   7  Val   19
Time:      1: 97    Ch  2  NOTE OFF    Note  68  Vel    0
Time:      1: 97    End Of Track
End Track   #3
Start Track #4
Time:      1: 64    Ch  2  NOTE ON     Note  67  Vel   64
Time:      1: 64    Ch  2  PG CHANGE   PG    56
Time:      1: 64    Ch  2  CTRL CHANGE Ctrl   7  Val   66
Time:      1: 80    Ch  2  NOTE OFF    Note  67  Vel    0
Time:      1: 80    End Of Track
End Track   #4
Start Track #5
Time:      1: 48    Ch  2  NOTE ON     Note  66  Vel   64
Time:      1: 48    Ch  2  PG CHANGE   PG    56
Time:      1: 48    Ch  2  CTRL CHANGE Ctrl   7  Val  101
Time:      1: 64    Ch  2  NOTE OFF    Note  66  Vel    0
Time:      1: 64    End Of Track
End Track   #5
Start Track #6
Time:      1: 25    Ch  2  NOTE ON     Note  65  Vel   64
Time:      1: 25    Ch  2  PG CHANGE   PG    56
Time:      1: 25    Ch  2  CTRL CHANGE Ctrl   7  Val   25
Time:      1: 40    Ch  2  NOTE OFF    Note  65  Vel    0
Time:      1: 40    End Of Track
End Track   #6
```

Figura 3.4: Ejemplo de fichero con instrucciones MIDI

Tal y como se puede observar la reproducción del fichero MIDI (Figura 3.6) es prácticamente exactA al WAVE con la única diferencia que el decodificador usado es wildmidi, propio para este tipo de fichero.

3.2.5. Interfaz de usuario

El último componente del programa es la interfaz de usuario, lugar desde donde se controlará la ejecución del programa. Para crear el aspecto visual de la interfaz se ha utilizado una herramienta de diseño gráfica de QT conocida como QT Designer, que puede también ser integrada dentro de Eclipse al ser instalada como un *plugin* más. Mientras se diseña la interfaz se genera de forma automática el código html correspondiente, además de las declaraciones de componentes en un fichero. En este se declaran todas aquellas funciones con las que se quiera enlazar la interfaz. Después para realizar esta conexión hay que utilizar la función `QObject::connect()` en la que hay que determinar que función corresponde a cada elemento de la interfaz.

El problema más relevante que se ha encontrado a la hora de acoplar OpenCV con QT, llegó en el momento de representar imágenes en la interfaz. Para poder llevarlo a cabo hay que realizar una conversión de imagen de `IplImage`, tipo de imagen de OpenCV, a

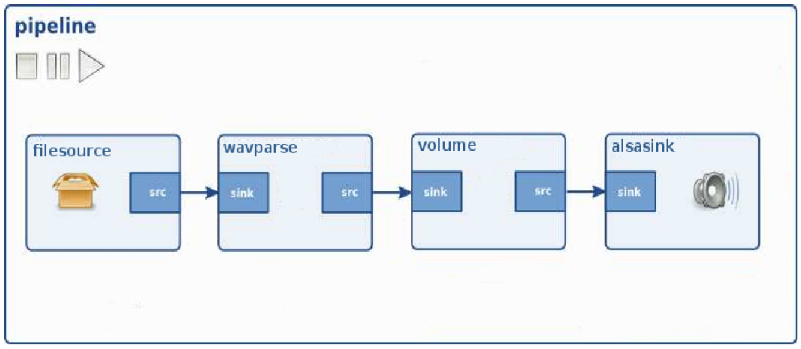


Figura 3.5: Gráfico de elementos de GStreamer para reproducción WAVE

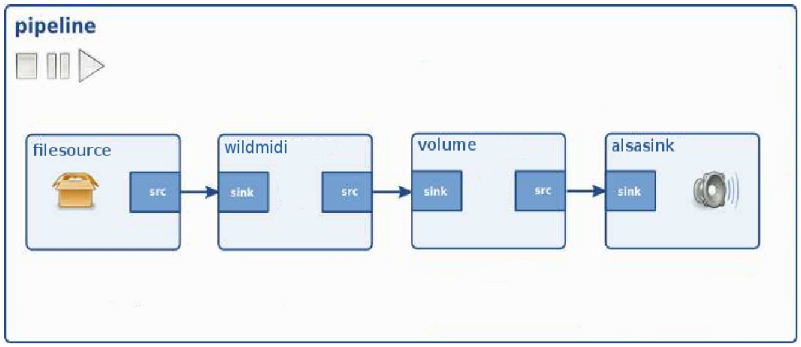


Figura 3.6: Gráfico de elementos de GStreamer para reproducción MIDI

`Qimage`, propio de Qt.

El primer paso será realizar una copia, línea por línea, de los datos de imagen del tipo `Iplimage`, contenidos en su campo `Imagedata`. Para ello hay que tener en cuenta la diferente representación de estos datos en un tipo y en otro. En el caso de estar codificada la imagen en escala de grises no hay mayor problema, pero si la imagen está en color existen diferencias. OpenCV representa los píxeles en RGB con 3 *bytes*, mientras que QT utiliza 4 *bytes*. Por lo tanto, se copiarán los 3 *bytes* de la imagen de tipo `IplImage` y luego se añadirá uno con valor igual a 0.

A continuación, se generá una variable tipo `Qimage`, a la que se pasará como parámetros los datos copiados anteriormente: altura, longitud e identificación de la profundidad necesaria para representar los píxeles `QImage::Format Indexed8` para escala de grises y para color `QImage::Format RGB32` .

Por último destacar que la inicialización de los distintos campos de la interfaz se hace a través de la lectura de un fichero de configuración donde están todos los valores necesarios.

La interfaz resultante se muestra en la figura 3.7.

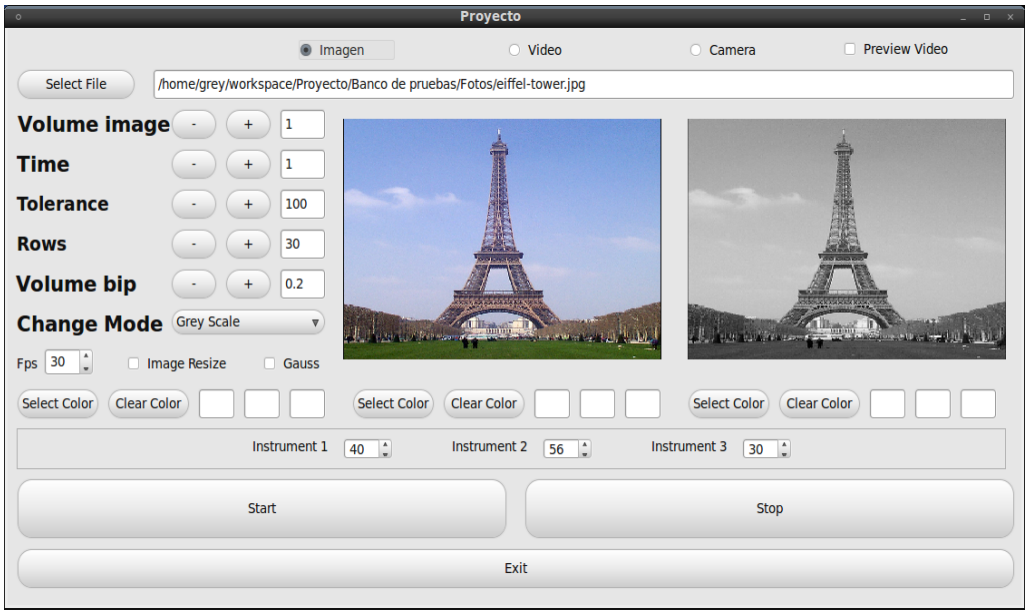


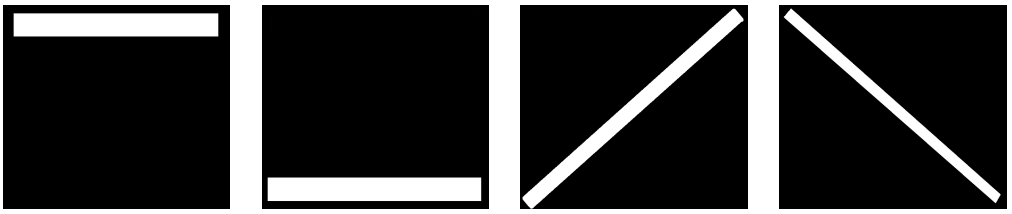
Figura 3.7: Interfaz gráfica de la aplicación

Capítulo 4

Resultados Experimentales

La fase de experimentación se ha tenido la oportunidad de realizarse con la colaboración de una persona invidente, alumno de la propia universidad. Este hecho aporta un gran valor a los resultados y a las pruebas, ya que se conoce de primera mano las dificultades, necesidades y sugerencias de las personas a las que va dirigido este trabajo. Reseñar que los resultados que posteriormente son presentados y analizados corresponden a dos sesiones de experimentación cuyos resultados han sido unificados.

Antes de realizar las pruebas experimentales se presentaron imágenes de ejemplo (ver cuadro 4.4) con las que explicar a esta persona el funcionamiento de la aplicación. Básicamente se trató de especificar que los tonos agudos correspondían a las partes superiores y los tonos graves a los píxeles inferiores a partir de las imágenes a) y b) del cuadro 4.4. Para que oyera la transición entre los tonos graves y agudos, y viceversa, se utilizaron las imágenes con diagonales c) y d) del anterior cuadro referenciado. También se intenta familiarizar al oyente con los sonidos de referencia que marcan el inicio y el fin del audio correspondiente a una imagen, para poder conseguir la identificación espacial de los objetos que se representen posteriormente.



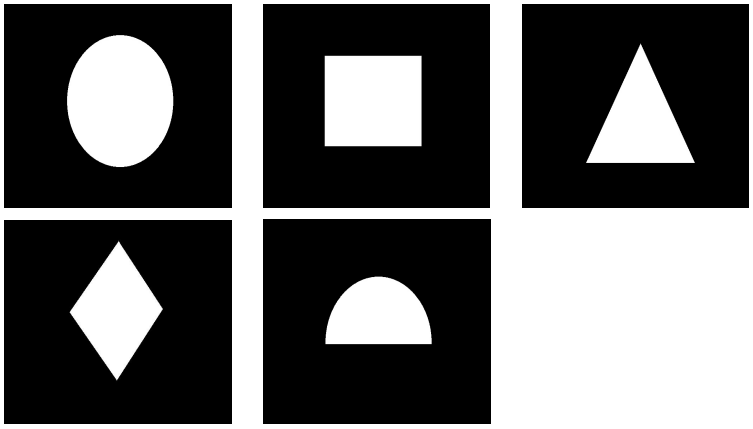
a) b) c) d)

Cuadro 4.1: a) Sonido Agudo; b) Sonido Grave; c) De grave a agudo; d) De agudo a grave

A continuación se desglosan las distintas experimentaciones separadas en secciones.

4.1. Formas geométricas

En primer lugar se presentaron una serie de figuras geométricas de forma individual (ver cuadro 4.2), para posteriormente sonificar imágenes en las que aparecían diversos objetos (ver cuadro 4.3). La diferenciación, en la primera sesión de experimentación, entre cuadrado, triángulo y círculo no supuso ningún esfuerzo y su identificación era exitosa. La confusión apareció al presentar, en la seguna sesión, las figuras de rombo y semicírculo. Estas dos figuras se confundían con el círculo y el triángulo respectivamente, por su parecido gráfico.



Cuadro 4.2: Formas geométricas presentadas individualmente

Después de hacer comparaciones entre las diferentes formas semejantes, para mejorar la asimilación de sus sonidos, se presentaron las imágenes con combinaciones de formas que se pueden observar en el cuadro 4.3. Los resultados obtenidos con cada una de esas imágenes también están recogidas en el cuadro:

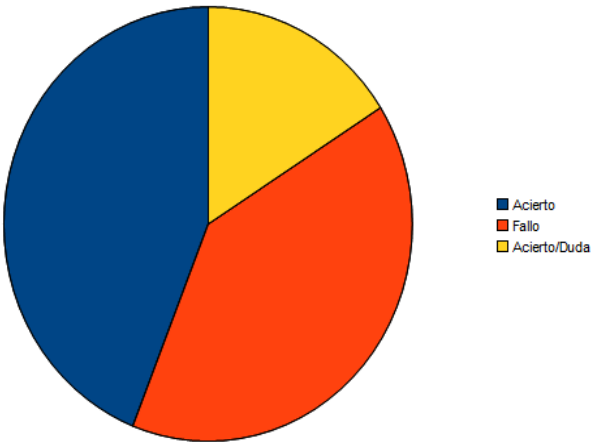


Figura 4.1: Diagrama por sectores de los porcentajes de acierto para la sonificación de formas geométricas

4.2. Números

El siguiente paso fue sonificar números representados de blanco en un fondo negro, de forma individual. Antes de recoger ningún dato se realizaba una sonificación de cada número para que tuviera una referencia de cada uno de ellos. Por este motivo la acción de identificar un número, o en otros casos figuras o letras, viene a ser una combinación entre la interpretación del sonido y la comparación de este sonido con otros ya memorizados. Demostrando esta teoría, cuando se presentaban objetos nuevos la dificultad que tenía para identificarlos era mucho mayor que si lo había escuchado con anterioridad.

Como se puede ver en la figura 4.2 hay una clara diferenciación entre los números que le son fácilmente identificables con los que no. Tanto 1,2,4,5, y 7 no presentan casi fallos, mientras que en el resto sí que se presentan bastantes errores. Los grupos de confusión se dividían en dos. El primero estaría compuesto por la dificultad de distinguir el número 3, ya que se suele reconocer como 5, y el segundo sería el compuesto por el 6,8 y 9. En este último grupo se hicieron comparaciones para que memorizase los sonidos para poder distinguirlos. Y aunque se siguieron produciendo fallos, estos se redujeron de forma considerable. En cuanto al tiempo, las imágenes solían ser sonificadas con una duración de dos segundos, aunque podía ser reducido este valor sin que disminuyese el porcentaje de aciertos. Esto no ocurriría posteriomente en las figuras con varios números, ya que si el tiempo disminuía la capacidad de diferenciación de números se veía también reducida.

En la figuras 4.4 y 4.5 se representan los números de varias cifras sonificados junto con los resultados asociados a cada uno de ellos. La duración del sonido generado variaba entre tres y cuatro segundos.

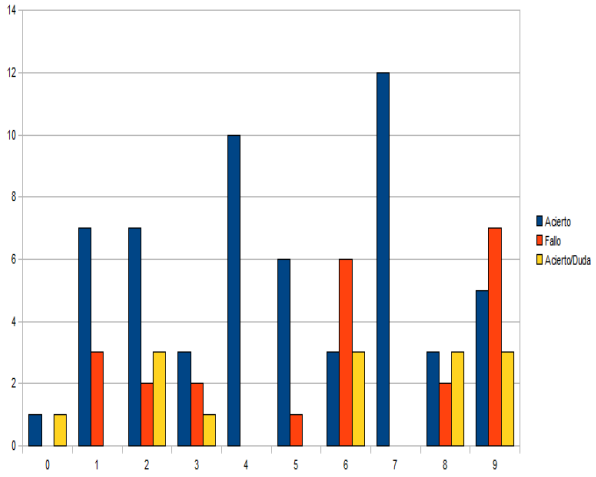


Figura 4.2: Gráfica de resultados obtenidos sobre números

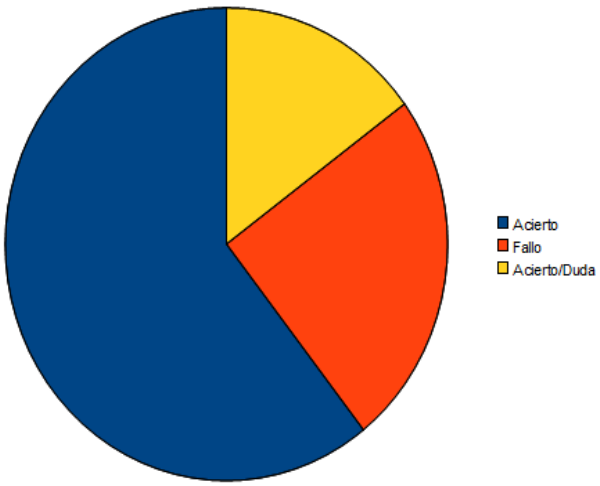


Figura 4.3: Diagrama por sectores de los porcentajes de acierto para la sonificación de números

4.3. Letras

Siguiendo el mismo procedimiento que a la hora de sonificar los números, se continuó con letras, primero de forma individual y después en secuencia de ellas. A partir de la gráfica de la figura 4.4 se pueden sacar una serie de conclusiones. Primeramente destacar que letras no presentadas con anterioridad como Y,Q,P y C no pudieron ser identificadas. Otras en cambio, por su complejidad menor o por su familiarización por pruebas anteriores, como A, L, T ó H le eran fácilmente identificables. Ya el resto de letras presentadas sí que solían ser acertadas con porcentajes mayores al cincuenta por ciento, pero entre ellas existían conjuntos de letras entre las que surgían dudas y errores. Algunos de estos errores se producían entre D-B, E-F Y M-N, justificados por el parecido gráfico entre ellas lo que produce que los sonidos sean muy semejantes. El gráfico con los porcentajes de aciertos se puede ver en la figura 4.5. El tiempo asignado al sonido en este caso aumentó a una media de dos segundos y medio, ya que con menos tiempo los errores aumentaban.

A continuación se presentaron palabras de tres o cuatro carácteres , para comprobar si era capaz identificar estas de un solo barrido. Salvo con la palabra HOLA ,que ya había escuchado en la primera sesión de experimentación, con las demás se tenía que sonificar varias veces la imagen para poder ser interpretada. El tiempo medio de los sonidos fue de unos tres segundos y medio, aunque en las repeticiones se llegó a fijar el sonido en cuatro segundos. Las imágenes utilizadas se pueden ver, junto a su resultado asociado, en las figuras 4.6 y 4.7.Por último destacar, que de alguna letra la persona que realizó la

experimentación desconocía su forma, lo que dificulta en gran medida su interpretación.

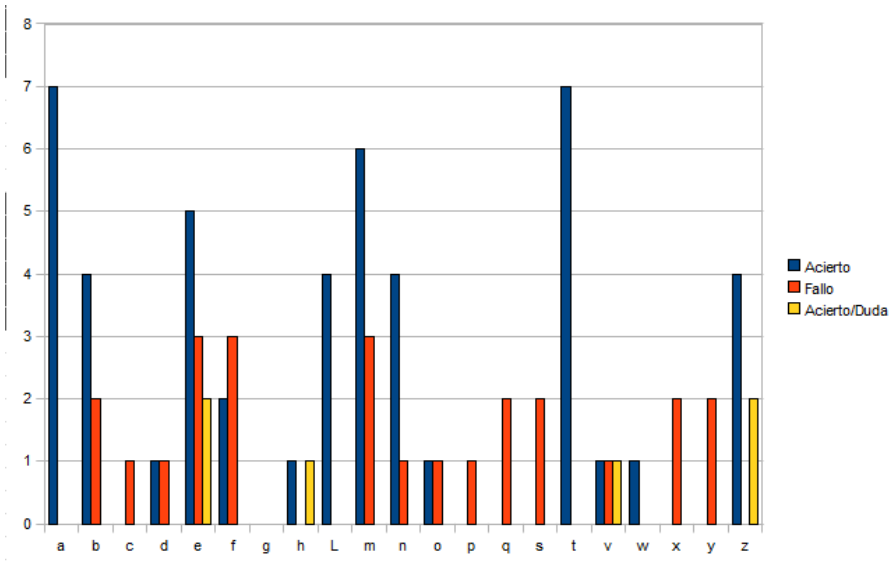


Figura 4.4: Gráfica de resultados obtenidos sobre letras

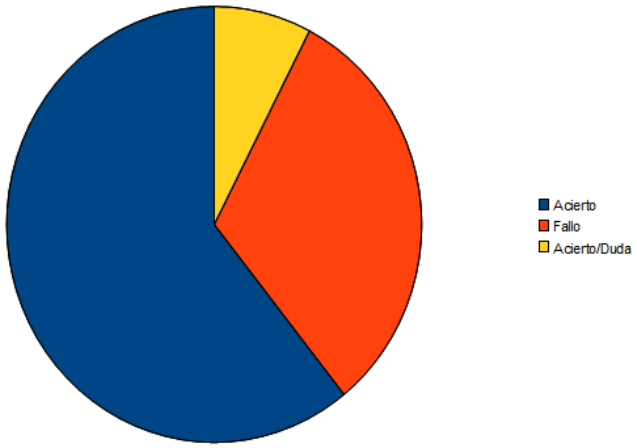


Figura 4.5: Diagrama por sectores de los porcentajes de acierto para la sonificación de letras

4.4. Colores

En la identificación de colores no se registraron las pruebas, pero fueron satisfactorias a la hora de distinguir los colores aunque en la especificación de la posición espacial en la que se encontraban los objetos surgían dudas. Esto viene provocado porque un mismo tono procedente de instrumentos diferentes da la sensación de ser distinto, lo que da lugar a la confusión. Las pruebas de búsqueda solo se realizaron con imágenes, no con vídeos.

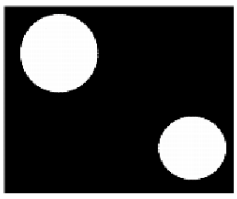
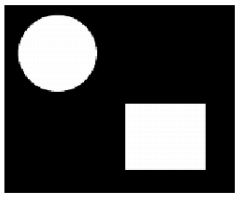
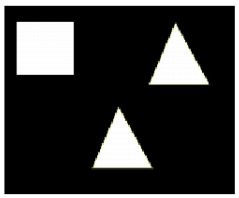
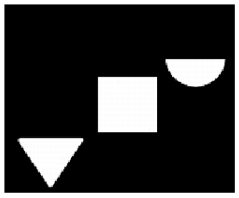
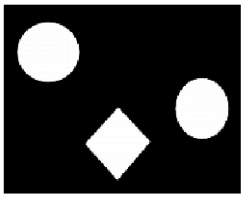
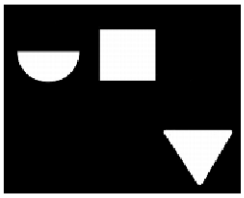
Algunas de la imágenes utilizadas fueron las que aparecen en el cuadro 4.8.

En cuanto a los intrumentos que se fijaron, como predeterminados, después de hacer las pruebas fueron: violín, trompeta, y guitarra eléctrica. Son sonidos muy diferentes entre sí y además son constantes en el tiempo, es decir, no sufrían un desvanecimiento de intensidad después de comenzar.

4.5. Pruebas dinámicas

Las pruebas de vídeo se han dividido en pruebas estáticas y dinámicas. Las primeras, con la cámara fija, se basaban en detectar el movimiento que se producía en el espacio donde se estaba grabando. Tanto con sustracción de fondo como a través de la imagen diferencia estos movimientos fueron identificables. Las imágenes asociadas a esta prueba son las que aparecen en la introducción en los cuadros 1.5, 1.6 y en la figura 1.8. El inconveniente descubierto es que al utilizar sonidos cortos es difícil hacer una representación exacta de dónde o cómo se produce el movimiento, pero también si se aumenta el tiempo de sonificación puede darse la situación de que se produzca un movimiento y no sea detectado.

Las pruebas dinámicas han consistido en moverse a través de pasillos del departamento. Para ello únicamente se ha utilizado la imagen diferencia ya que con la sustracción de fondo el sonido era indescifrable. Este problema en la sustracción de fondo viene producido porque no se realiza una rápida actualización de los datos del fondo, tal y como se explicó en la sección de descripción informática. Con esta prueba se averiguó dónde se situaban las puertas y el contorno de diversos objetos. Además se detectaron los movimientos que se producían delante de la cámara, como podía ser que alguna persona se cruzase (ver cuadro 4.9). Otra prueba dinámica, también en movimiento y con imagen diferencia, se realizó enfocando la cámara solo hacia el suelo de manera que si hubiese algún objeto fuese detectado. En principio al ser el suelo uniforme no generaba casi ruido, ya que las imágenes eran similares, pero si se encontraba algún otro objeto sería sonificado por ser una imagen con valores de brillo diferentes. Los errores surgían al detectar sombras que eran sonificadas por la aplicación y hacían entender al oyente que en ese punto existía un objeto.

	Reconocida a la segunda escucha ya que dudaba entre determinar si eran círculos completos o semicírculos.
	A la primera escucha fue acertada.
	Al igual que la número dos fue acertada sin ayuda de repeticiones.
	En este caso el error apareció a la hora de determinar la forma del triángulo superior, confundido por un semicírculo. La persona que realiza la experimentación apunta que en los tonos altos (agudos) le es más dificil la identificación de los detalles que dan lugar a la diferenciación entre las figuras.
	En esta figura tras varias repeticiones en las que existe alguna duda, la respuesta finalmente sería la correcta.
	Triángulo inicial confundido por semicírculo.
	Al tratarse de tres figuras muy similares no llega a distinguirlas.
	Confusión del rombo incial con un círculo.
	El semicírculo inicial no lo distingue.

Cuadro 4.3: Imágenes con varias figuras geométricas

	Después de varias repeticiones, con cierta duda acertó a decir el número correcto.
	Tras tres errores dió con el número. Extrañamente el número en el que se encontró dificultades fue el dos que era confundido con el siete, cosa que no había ocurrido anteriormente.
	Siguió apareciendo la confusión entre el tres y el cinco pero a la segunda repetición dió la respuesta correcta.

Cuadro 4.4: Conjunto de números de 2 cifras presentados

	En esta ocasión la cifra no reconocida fue el cero, que se confundió con el número ocho y seis posteriormente.
	Esta fue la cifra que no consiguió acertar, incluso después de varias repeticiones. Al ser los tres números que más dificultades tenía para distinguirlos, le fue imposible, al escucharlos de forma consecutiva, saber cuáles eran.
	Dudando entre decir que la segunda cifra era un tres o un nueve, finalmente acerto el número representado.

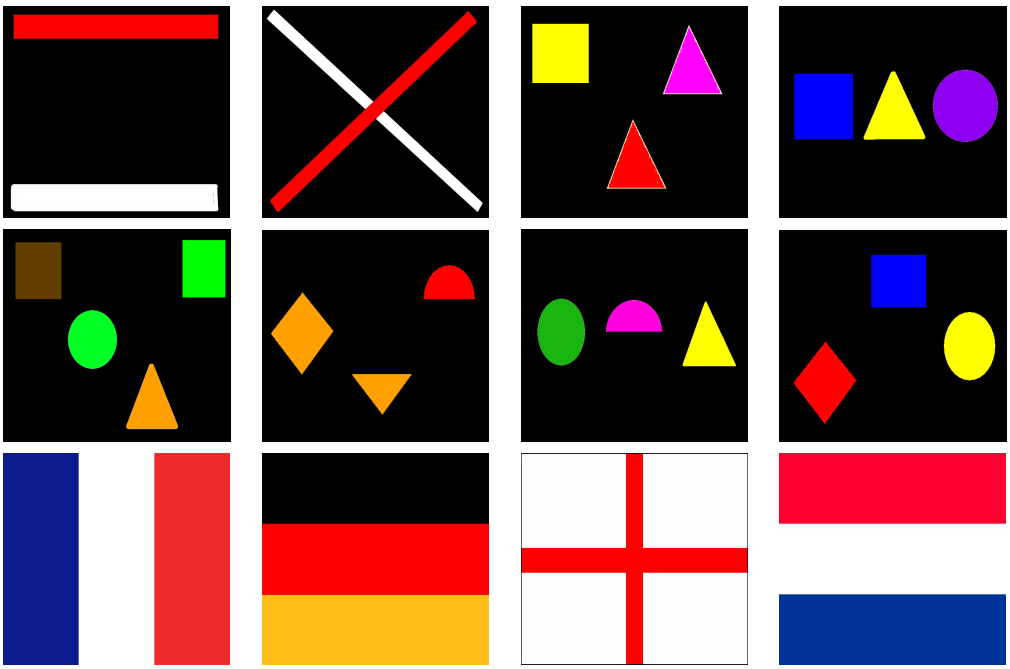
Cuadro 4.5: Conjunto de números de 3 cifras presentados

	Acertó de principio I y N y posteriormente, en el segundo intento, diferenció la F.
	Identificación de L y Z , pero la U, que no había sido sonoficada anteriormente, no puedo ser acertada.

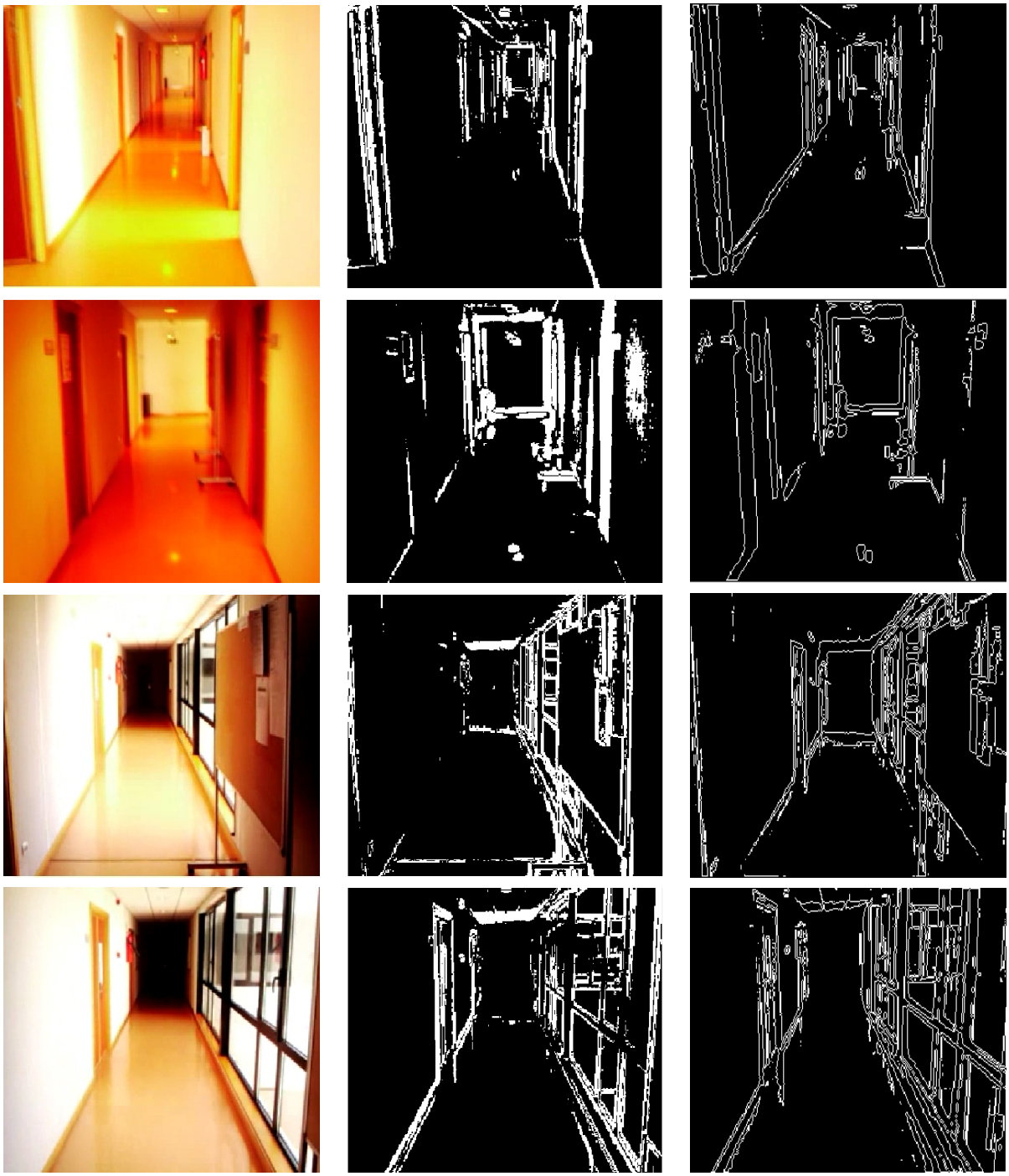
Cuadro 4.6: Conjunto de palabras de 3 cifras presentadas

	Acertada en la primera escucha.
	Acertó en declarar que la segunda y cuarta letra eran A. La confusión se produjo con C y S, dos carácteres que anteriormente no había sido capaz de acertar.
	No consiguió identificar ninguna letra correctamente.

Cuadro 4.7: Conjunto de palabras de 4 cifras presentadas



Cuadro 4.8: Imágenes de color



a) b) c)

Cuadro 4.9: a) Imagen original; b) Imagen diferencia binarizada; c) Imagen diferencia con detección de bordes

Capítulo 5

Conclusiones y trabajos futuros

5.1. Conclusiones

En este proyecto se ha desarrollado una aplicación que consigue sonificar imágenes digitales para intentar que a través del sonido una persona invidente pueda crear una representación mental de esta imagen. La idea de utilizar la sonificación para la ayuda a personas con discapacidades visuales es interesante, sobretodo, tras los avances realizados en esta materia por Peter Meijer, explicados en la introducción. De su trabajo parten muchas de las ideas desarrolladas en este proyecto, como es el proceso de sonificación de una imagen en escala de grises. En su desarrollo, se determina la intensidad del sonido a partir del brillo y el tono a partir de la posición vertical. La novedad conseguida respecto al programa de Meijer es la generación de sonidos con diferentes timbres a partir de imágenes en color. Se asociará a cada color buscado un instrumento diferente para su diferenciación.

Asimismo, con respecto a las imágenes en escala de grises, se han aplicado a estas una serie de tratamientos con los que se consigue ampliar la funcionalidad del proyecto. Se ha capacitado al programa con, por una parte, la posibilidad de detectar bordes de manera que únicamente se consiga sonificar el contorno de los distintos objetos, y por otra, la capacidad de detectar movimiento entre frames, aplicando técnicas de procesamiento dinámico. Por último destacar que se ha hecho hincapié en simplificar lo máximo posible el sonido de salida para facilitar su interpretación. Con ese fin se han puesto en práctica distintos métodos y filtros como escalado y filtro de Gauss.

Tras realizar la implementación, se llevó a cabo la experimentación. En ella se sacaron las conclusiones más notables así como un conjunto de ideas que pueden ser aplicadas en trabajos futuros. Se aprecia una interpretación del sonido satisfactoria, especialmente en imágenes estáticas, aún sin existir un proceso de entrenamiento sonoro específico. Al

usar como entrada del programa imágenes recogidas desde una cámara, se ha logrado detectar movimiento y se consigue crear una idea básica del entorno que es visualizado. Gracias a los avances obtenidos y a los resultados favorables se ha conseguido generar, en las personas invidentes que han probado el programa, una ilusión que se refleja en las ganas de seguir desarrollando la aplicación.

5.2. Trabajos futuros

Tras la finalización de la fase de experimentación han surgido multitud de ideas que podrían utilizarse en trabajos futuros. Las más destacables se explican a continuación:

- Conseguir mediante la captura de vídeo con varias cámaras (visión estéreo), un cálculo de distancias de modo que si la persona invidente se acercara a un objeto fuera avisada de ello. Esta función podría ser implementada a través de sonidos o incluso con la ayuda de un dispositivo físico con vibración, para que la sensación de aviso sea más clara.

- Acoplar el programa en un dispositivo móvil de tamaño reducido, para poder desplazarse con él de forma cómoda.

- Para mejorar la movilidad, en escenarios específicos, se podría implementar un método de filtrado de color de suelo de manera que cualquier objeto que sea diferente a él sea lo único que produzca sonido.

- En la reproducción del audio, crear un sonido estéreo que balancease del canal izquierdo al derecho de forma progresiva. De este modo se facilitaría la tarea de percepción espacial de las escenas sonificadas.

- Respecto a la búsqueda de colores, investigar el comportamiento del programa cambiando el espacio de color utilizados. Es decir, sustituir la codificación RGB por alguna otra, como HSV en la que el color está determinado por tres variables: tono, saturación y valor de color.

- En el apartado de color generar una base de datos de manera que un color determinado tuviese un instrumento asociado. De esta manera se evitaría la tarea de ir eligiendo los colores que se quieren detectar desde la interfaz.

Bibliografía

[1] J. F. Vélez A. B. Moreno. Á. Sánchez, J. L. Esteban. *Visión por computador.* Dykinson, 2003.

[2] Agencia EFE. Prueban un dispositivo que permitirá "oír" los colores a los niños ciegos. *20 minutos, edición digital*, 2010. http://www.20minutos.es/noticia/382332/0/bebes/ciegos/oir/.

[3] Homer Jacobson. The informational capacity of the human ear. *Science*, 112:143–144, 1950.

[4] Homer Jacobson. The informational capacity of the human eye. *Science*, 113:292–293, 1951.

[5] Isabel F. Lantigua. El soldado ciego que ve con la lengua. *El mundo digital*, 2010. www.madrimasd.org/informacionidi/noticias/noticia.asp?id=43145origen=notiweb.

[6] Marcos Martín. Técnicas clásicas de segmentación de imagen. http://poseidon.tel.uva.es/ carlos/ltif10001/segmenclasica.pdf, 2002.

[7] Peter B.L. Meijer. An experimental system for auditory image representations. In *IEEE Transactions on Biomedical Engineering, Vol. 39, No. 2*, pages 112–121, 1992.

[8] Gene Mosca. Paul A. Tipler. *Física para la ciencia y la tecnología.*, volume 1B. Editorial Reverté, 2005.

[9] Massimo Piccardi. Background subtraction techniques:a review. 2004.

[10] Marconi Transatlantic Wireless Telegraph. Makes light audible. *New York Times*, 1912.

[11] José Ramón Mejía Vilet. Procesamiento digital de imágenes. 2005.

[12] Greg Wade. Seeing things in a different light. *BBC*, 2005. http://www.bbc.co.uk/devon/news$_features$/2005/*eyeborg.shtml*.