UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Interacción por color con OpenCV y OpenAL

Apellidos, nombre	Agustí i Melchor, Manuel (magusti@disca.upv.es)
Departamento	Departamento de Informática de Sistemas y Computadores (DISCA)
Centro	Escola Tècnica Superior d'Enginyeria Informàtica Universitat Politècnica de València

1 Resumen de las ideas clave

Este artículo aborda la construcción de un instrumento musical sencillo con un computador, sin usar los convencionales teclado y ratón como medios de interacción. Es un ejemplo de integración de medios, esto es, el usuario interactuará con el computador a través de objetos externos al propio sistema y el computador responderá emitiendo sonidos y, completándolo, con la imagen del teclado "virtual" del instrumento. El computador hará uso, para la entrada de datos, de técnicas de Visión por Computador (en adelante VxC) y se basará, para la salida, en la reproducción o generación de sonidos al estilo de cómo lo hace un instrumento real. Este instrumento nace de la inspiración de otros como el arpa láser[1] o el *theremín*[2], véase fig. 1, en los que no existe un contacto físico con el instrumento y la forma de interpretar se basa en modular las notas mediante la interacción del instrumentista con el instrumento.

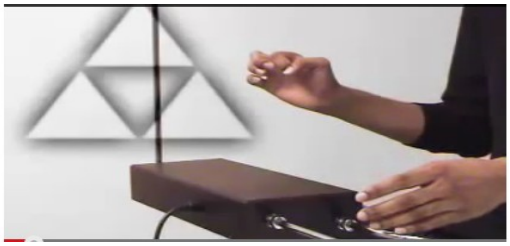


Figura 1: Arpa láser (izquierda) y Theremin (derecha). Fuentes: [3] y [4].

No es objetivo de este documento revisar el código fuente (que se puede obtener a vuelta de un correo electrónico dirigido al autor de este documento) hasta sus mínimos detalles, sino mostrar un ejemplo de uso de técnicas multimedia factibles en computadores actuales que permiten realizar el análisis de una escena visual o la generación de una respuesta de sonido con el uso de las API que se proponen.

2 Introducción

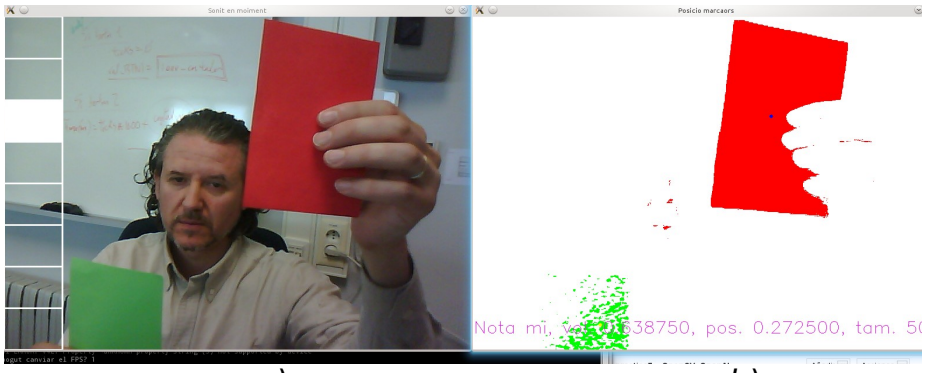
La idea que desarrolla el ejemplo consiste en ofrecer al usuario un interfaz al estilo de un instrumento musical, un teclado, con el que el usuario interactuará a través de la utilización de unas piezas de color predeterminado. Se utilizará OpenCV [1] para el procesado de imagen encaminado a la detección de esos objetos de color y OpenAL [2] para la generación de audio al estilo de síntesis por tabla de ondas. Ambas van a colaborar para obtener una extensión del área de trabajo del computador a su entorno físico inmediato.

La fig. 2 muestra un instante de la ejecución de la aplicación mostrando la ventana de interfaz de usuario (2a) y la de depuración (2b). Con esta aplicación

[1]Se puede leer más sobre este instrumento en la página web "Jean Michel Jarre Laser Harp sound by Elka Synthex". Disponible en: <http://www.polynominal.com/site/studio/gear/synth/elka_synthex/index.html#prettyphoto/14/>. La imagen se ha obtenido en la página del fabricante *Laser Harp* <http://www.harpelaser.com/>.

[2]Se puede leer sobre el funcionamiento del Theremin en "¿Cómo funciona el theremín?" en la URL <http://curiosidades.batanga.com/4922/como-funciona-el-theremin>. Las imágenes del *Theremin* se han obtenido de <http://en.wikipedia.org/wiki/L%C3%A9on_Theremin>.

se estudia un caso integración del audio y de la imagen como canales de comunicación del usuario con la aplicación. La entrada se realiza a partir de una imagen que es obtenida de una cámara; sobre la cual se determinará la acción que el usuario quiere realizar en función de la posición de dos objetos de color que exhiba, los llamaremos **marcadores**. Por su parte, la salida de información consistirá básicamente en hacer sonar una nota y variar el volumen de acuerdo a la posición de los marcadores.



a) *b)*

Figura 2: Interfaz de la aplicación: a) vista del instrumento y b) vista del proceso interno.

3 Objetivos

Una vez que el lector haya explorado los contenidos relacionados con la aplicación que aquí se diseña, será capaz de:

- Identificar cómo se procesa la secuencia de vídeo de una cámara digital a través del computador.
- Identificar cómo se generan sonido simples en un computador.
- Identificar cómo se puede ampliar el entorno de trabajo del computador al relacionarlo con el mundo exterior.

La estrategia para llevar a cabo el tema propuesto consiste en desarrollar la idea inicial en una serie de pasos más simples que puedan ser abordados y probados de forma independiente. Se considera que hay tres grandes aspectos a desarrollar:

- La aplicación se valdrá de la existencia de una cámara digital (p. ej., una cámara web con conexión USB) para seguir las acciones del usuario delante del computador.
- La aplicación ha de ser capaz de identificar dos marcadores, dos objetos de color (p. ej. uno rojo y uno verde), en la imagen RGB obtenida de una cámara, en tiempo de ejecución. Uno de los marcadores indicará la nota (el rojo) y el otro (verde) el volumen de la reproducción del sonido.
- La aplicación simulará el comportamiento de un instrumento virtual (un teclado) mediante la técnica de síntesis por tabla de ondas.
- Ha de ser posible interactuar al estilo tradicional (monitor y teclado) para facilitar las tareas de depuración y configuración de la aplicación.

De este manera, el usuario indicará a la aplicación qué nota deberá hacer sonar en función de la altura (dentro de la imagen) a la que muestre uno de los objetos (escogemos el de color rojo para esta tarea por fijar una decisión) y con

un volumen para la reproducción de los sonidos basado en la posición en horizontal del otro marcador, esto es, en función de la columna en que esté el objeto de color verde.

4 Desarrollo

Veamos ahora **cómo se ha diseñado la solución**, para después pasar a hablar de la implementación del algoritmo en la aplicación final. El algoritmo utilizado se puede ver como una secuencia de bloques que se pueden ir refinando y que permiten concentrar la atención en un aspecto y probar su corrección, sin olvidar su papel en la aplicación.

A partir de los objetivos fijados, la aplicación se puede describir como la respuesta sonora que la aplicación devuelve al usuario en función de la posición de los marcadores que este utiliza. El volumen estará en función de la proporción entre la columna que ocupa el marcador de volumen y el ancho de la imagen que proporcione la cámara. La nota a reproducir se obtendrá en función de la relación entre la coordenada horizontal del marcador relativo a las notas y el ancho de la imagen que se obtiene de la cámara. Por brevedad de esta exposición, esta solución fija las siguientes restricciones de partida para el desarrollo:

- Se ha escogido el color de los marcadores, de manera que los elementos del fondo que se utilice no deben contenerlos para que no se confundan. Y, puesto que son conocidos a priori, se podrá integrar esta información en el algoritmo.
- De partida, se considerará que se dispondrá de un total de ocho notas que permiten representar una escala de un instrumento. Estas se guardarán en ficheros de sonido en formato WAVE.

4.1 Diseño de la aplicación

Para implementar los objetivos propuestos se va a basar la solución en el diagrama de bloques que muestra la fig. 3. El computador ha de **analizar la escena** que tiene delante para tomar la decisión de cuál debe ser su repuesta.

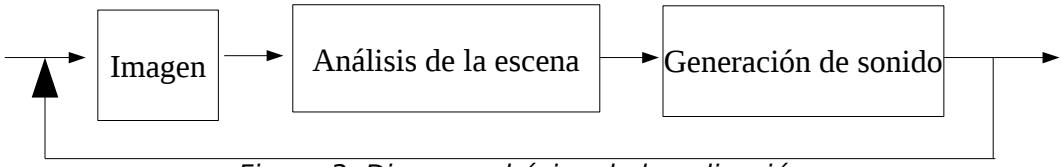


Figura 3: Diagrama básico de la aplicación.

Para ello llevará a cabo la **detección de los marcadores** utilizando el API de OpenCV. En nuestro caso los marcadores son objetos caracterizados por su color. La fig. 4 indica cómo este paso se puede dividir en dos acciones. En primer lugar la segmentación: se ha de analizar la imagen para detectar los puntos que están en color rojo (el marcador de la nota a hacer sonar) y los que están a color verde (por el marcador de volumen de la reproducción); el resto es considerado “fondo” de la imagen y no precisa ser analizado.

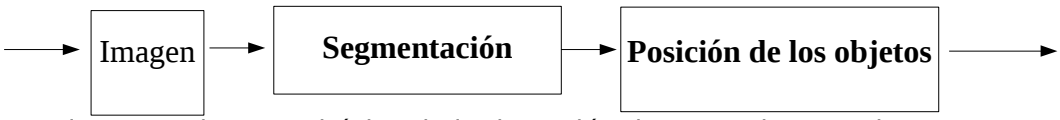


Figura 4: Diagrama básico de la detección de marcadores en la escena.

A partir de la determinación de la posición de los marcadores en función de las coordenadas de la imagen, se puede calcular la respuesta del sistema que, en este caso, recordemos está basada en el uso de sonidos.

La fig. 5 recoge la descomposición del bloque de **generación de sonido** en elementos más simples. A partir de las coordenadas de posición de los marcadores se eligen la nota y el volumen con el que seguir la aplicación. Determinados estos valores, se ejecutan las instrucciones de manipulación de audio con el API de OpenAL que hacen posible que se escuche el sonido escogido.

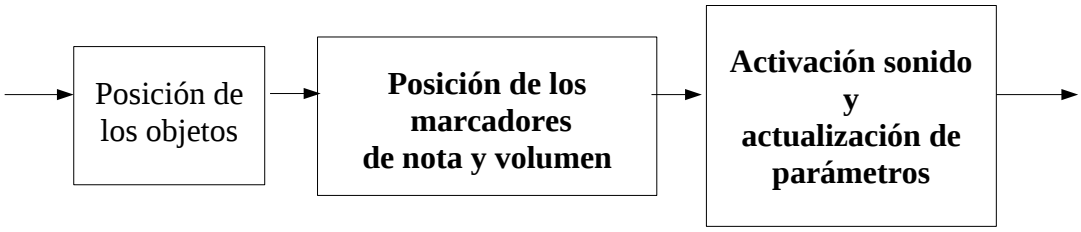


Figura 5: Diagrama básico de la respuesta basada en sonidos

Para cumplir con el tercer objetivo y dado que se pide que el ejemplo sea instructivo y permita la experimentación, la fig. 6 completa el diagrama básico inicial con los bloques que complementan las acciones auxiliares.

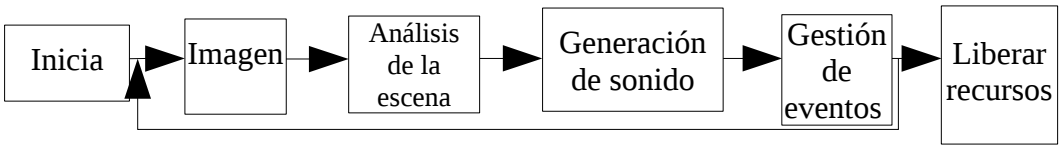


Figura 6: Revisión del diagrama básico de la aplicación.

Por un lado se añadirá un bloque de **gestión de eventos**, al final del bucle, que es el encargado de ofrecer los mecanismos de interacción tradicional y la información, en tiempo de ejecución, de la evolución de los parámetros que define el sistema. Esta información permitirá la evolución interna del sistema para estudiarlo, comprenderlo y depurar fallos. Para ello muestra la nota a reproducir, el volumen a emplear y el número de puntos a encontrar de un color para determinar la existencia de uno de los marcadores.

Por otro lado, la descripción de la solución se completa con dos bloques de operaciones básicas: **inicializaciones y liberación de recursos** que, antes y después del bucle principal respectivamente, se encargan de las tareas de gestión más básicas.

4.2 Implementación de la solución

El algoritmo inicial es el que muestra el listado 1. El programa es básicamente un bucle, una repetición de los pasos centrales de la fig. 3, controlado por una variable que evalúa si algún paso ha determinado que se debe terminar la aplicación.

Veamos algunos de los pasos con detalle. Si el lector tiene interés de ver la solución al completo no dude en dirigirse al autor de este documento. En aras

de la brevedad de este documento nos centraremos en las operaciones de "Análisis de la escena" y "Generación del sonido". Es necesario comentar que, sobre la versión inicial de Bonet y Cardona [5], se han introducido una serie de revisiones relativas a:

- Simplificar las expresiones que permiten determinar la nota que ha de sonar y las otras propiedades que se utilizan del audio en la aplicación.
- Optimizar la gestión de recursos de OpenAL, minimizando el número de lecturas de disco de las notas.
- Poner los cálculos de coordenadas de los marcadores en relación al tamaño de la imagen que proporciona la cámara.
- Añadir un parámetro de cuantificación de puntos de color detectados para robustecer el algoritmo frente a falsas detecciones de puntos de un color esperado debidas a ruido en la captura.

```
inicializarAudioVideo_y_ventanasAplicación();
mientras ( condicionDeAcabar == FALSO)
  cuadro ← obtenerImagen_y_ pintarInterfaz();
 (marcadorNota, marcadorVolumen)← segmentarImagen_y_detectarMarcadores( cuadro );
  generaNota( marcadorNota, marcadorVolumen );
  condicionDeAcabar ← gestiónDeEventos();
fmientras
liberarRecursos();
```

Listado 1: Pseudocódigo para el algoritmo del programa principal.

Obviaremos el cómo se han hecho las operaciones auxiliares, pero las describimos para que el lector tenga toda la información:

- "inicializarAudioVídeo_ y_ventanasAplicacion", inicializa el sistema de audio, cargar las notas de disco e inicializar el sistema de captura de vídeo.
- "liberar recursos", libera todos los demandados al sistema como los relacionados con el uso de ventanas, las estructuras para el tratamiento de imágenes y las fuentes y almacenes de audio.
- "gestión de eventos", encargada de recoger la interacción tradicional llevada a cabo con el teclado y el ratón, para permitir la depuración.

4.3 Captura de las imágenes

Este es el punto del programa principal que se ha denominado *obtenerImagen_y_ pintarInterfaz*. El listado 2 recoge las operaciones que se ha implementado (en negrita) y las que se utilizan del API de OpenCV, en este caso. La toma de imágenes debe ir acompañada de una comprobación acerca de si ha existido algún error en tiempo de ejecución o no se ha obtenido una imagen de la cámara, véase listado 2, en cuyo caso (al menos la solución más simple) debería detenerse el programa. Si se ha tomado una imagen, se ha de hacer una copia para poder mostrar resultados intermedios de la ejecución del algoritmo, manteniendo el contenido original para hacer visible el interfaz del instrumento.

```
(cuadre) ← obtenerImagen_y_ pintarInterfaz( imagen )
{
  cuadre = cvQueryFrame(capture);
   if( !cuadre )
   {
     areaMensatges(img, FINESTRA, "ERROR: no tinc imatge.\n" );
     cvWaitKey( 0 );
     acabar = TRUE;
   }
   else
   {
     cvShowImage( FINESTRA, cuadre );
     pintarInterfazInstumento( cuadre );
   }
}
```

Listado 2: Código para la toma de imágenes.

4.3.1 Análisis de la escena

Este es el punto del programa principal que se ha denominado *segmentarImagen_y_detectarMarcadores*. Tiene dos partes: la segmentación de la imagen y la determinación de la posición de los objetos. La **segmentación** se realiza de forma heurística: el color ha de ser un primario en el espacio de color RGB y se describe con una expresión que requiere de un valor mínimo de la componente de color que corresponda (roja o verde en nuestro caso) y de su relación con las otras dos.

La determinación de l**a posición de los marcadores** de volumen (verde) y de nota (rojo) se realiza por el cálculo del centro de gravedad del conjunto de puntos que lo componen. Este cálculo se puede hacer de diferentes formas, pero la más directa es obteniendo el valor estadístico medio de todas las coordenadas de puntos de los marcadores, color rojo y verde en nuestro caso.

El código que se muestra en el listado 3 realiza la asignación de valores iniciales a las variables que van a acumular la posiciones en columnas y filas de puntos de cada marcador, así como su número. Cada marcador es el resultado de evaluar un conjunto de variables que representan la situación de puntos de un determinado color. De esta manera, terminado el recuento, se pueden obtener las coordenadas del centro de gravedad de un objeto como el valor promedio de todos los puntos[3].

En el código se observará la identificación de los puntos por el color (escogido a priori) en base a los valores numéricos que toman las componentes RGB y que han sido determinados heurísticamente. Puesto que no es siempre el rojo un valor RGB con la componente de R al máximo valor y las otra dos a cero, se ha determinado que los objetos que se empleaban se representaban, más o menos bien, exigiendo que tuvieran un valor numérico mínimo de 80 (siendo 255 el máximo que puede tomar) para la componente de rojo (para el marcador de nota que es una tarjeta de este color) y que, a su vez, esta fuera dos veces mayor que la de verde y la de azul. Análogamente, se ha determinado el valor RGB para el marcador de volumen que es de color verde.

[3] Se puede ampliar este concepto que se conoce como los momentos espaciales de orden 1 en terminología de VxC, se puede consultar el documento de A. Sancho (2008). Tema 4. Características y representación de regiones y objetos. <http://www.escet.urjc.es/~visiona/tema4.pdf>.

```
(marcadorNota, marcadorVolumen) ← segmentarImagen_detectarMarcadores( imagen )
{
   cont_r = 0;    x_r_total = 0;    y_r_total = 0;    x_r_medio = 0;    y_r_medio = 0;
   cont_v = 0;    x_v_total = 0;    y_v_total = 0;    x_v_medio = 0;    y_v_medio = 0;

  // centro de gravedad de los píxeles verdes y rojos
  for(fila=0; fila < imagen->height; fila++)
   for(columna=0; columna < imagen->width; columna++)
   {
    color = cvGet2D( imagen, fila, columna );

    //marcador de volumen (color verde)
    if ( (color.val[G] > MINIMVERD) AND
        ((color.val[B] <= (color.val[G]/2)) AND (color.val[R] <= (color.val[G]/2))) )
    {
     y_v_total += fila;
     x_v_total += columna;
     cont_v++;
    }
    else  // marcador de nota (color rojo)
   if ( (color.val[R] > MINIMROIG) AND
       ((color.val[B] <= (color.val[R]/2)) AND (color.val[G] <= (color.val[R]/2))) )
   {
       y_r_total = y_r_total + fila;
       x_r_total = x_r_total + columna;
       cont_r++;
   }
   } // Fi de for(fila=0; ... for(j=columna; ...

  if (cont_v == 0)
    x_v_medio = y_v_medio = 0;
  else
  {
   x_v_medio = x_v_total / cont_v;
   y_v_medio = y_v_total / cont_v;
  }
   marcadorVolumen->x = x_v_medio/(float)imagen->width;
   marcadorVolumen->y  = x_v_medio/(float)imagen->height;
  if (cont_r == 0)

    marcadorNota ->x =  marcadorNota->y = 0;

  else
  {
   marcadorNota ->x = x_r_total / cont_r;
   marcadorNota->y = y_r_total / cont_r;
  }
```

Listado 3: Código para la etapa de análisis de la imagen.

Terminado el recuento, las coordenadas del marcador de volumen se convierten a valores en el rango entre 0 y 1, para facilitar el trabajo a la siguiente etapa de generación del audio. Obsérvese que la coordenada de fila no se utiliza, aunque se calcula, porque sólo se tiene en cuenta la posición en horizontal de este marcador para determinar la variación del volumen. De igual manera se procede a determinar la nota a reproducir a partir de la posición vertical del marcador de nota.

4.3.2 Generación del sonido

Esta etapa se puede subdividir en dos partes: la gestión de las fuentes de audio y la actualización de los parámetros relativos al sonido. El listado 4 muestra el cálculo para obtener la nota que ha de sonar. En función de la altura del centro de gravedad del marcador de nota y del número de notas, se determina la nota. A partir de esta información se actualiza el interfaz visual (**pintarTecla**) y el sonoro.

```
void generaNota(double coordY_marcadorNota,   double coordX_marcadorNota,
                double coordY_marcadorVol,    double coordX_marcadorVol)
{
 int numNota;

    numNota = (int)(perc_y * (double)NNOTES);
    pintarTecla( cuadre, numNota );

    // Gestión de las fuentes
    alGetSourcei(source, AL_SOURCE_STATE, &sourceState);
    if (sourceState == AL_PLAYING)
    {
      alSourcei (source, AL_GAIN, 0);
      alSourceStop(source);
    }

    // Activación de la fuente y actualización de parámetros
    alSourcei (source, AL_BUFFER, buffers[numNota]);
    sourcePos[0] = perc_v;
    listenerPos[0] = (1.0-perc_v);
    alListenerfv(AL_POSITION,listenerPos);
    alSourcefv(source,AL_POSITION,sourcePos);
    alSourcef(source,AL_GAIN, perc_x);
    alSourcePlay (source);

}
```

Listado 4: Código para la etapa de generación del sonido.

La parte de audio correspondiente a la gestión de las fuentes se encarga de determinar el estado de las mismas para, si está todavía reproduciéndose una nota, pararla. Después de lo cual, es posible asignar el sonido correspondiente a la nueva nota a tocar y actualizar el resto de parámetros en uso (como la posición del oyente y la fuente de sonido), lo que permite reiniciar el sonido.

4.4 Construcción y distribución del ejecutable

La solución aquí presentada utiliza una distribución de GNU/Linux Ubuntu 12.04, una versión 2.0 de OpenCV y una versión 1.0 de OpenAL (obtenida con OpenAL Soft 1.14). Para obtener el ejecutable es necesario compilar la aplicación contra las bibliotecas de funciones de OpenCV y OpenAL al estilo de la orden:

$ gcc programa.c -o programa `pkg-config --cflags opencv freealut openal` `pkg-config --libs opencv freealut openal`

Y para utilizar el ejecutable se habrá de acompañar este con un subdirectorio (a su mismo nivel) que contenga las notas en ficheros del formato escogido. Los nombres no se pueden cambiar sin modificar el código, pero el contenido de los

mismos se puede sobrescribir y la próxima ejecución utilizar un conjunto de sonidos diferente, si así se desea.

5 Conclusión

Tras la lectura y experimentación con los contenidos relativos a este documento, el lector será capaz de:

- Capturar vídeos, esto es tomar secuencias de imágenes a través de una cámara digital .
- Identificar objetos en la escena por su color en el espacio RGB.
- Dar una respuesta sonora basada en ficheros pregrabados como resultado de acción de la aplicación.

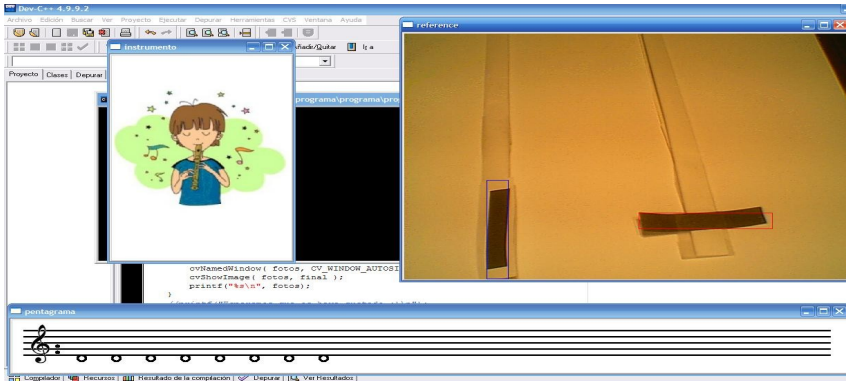


Figura 7: Propuesta final del trabajo de Bonet y Cardona [5].

Para animar al lector a utilizar el ejemplo expuesto, hay que decir que estas ideas fueron aplicadas a la práctica del aprendizaje de solfeo utilizando instrumentos. Para ello se desarrolló un interfaz, véase fig. 7, que muestra el instrumento en uso, al tiempo que una pequeña partitura donde van apareciendo las notas que se interpretan. Este registro de la interpretación realizada (la "partitura" así creada) se puede guardar para su posterior uso. Antes de cerrar este artículo quisiera agradecer a Elena y Jose Ignacio su interés en la realización de una primera implementación de la idea del autor y que ha servido de prueba de concepto para desarrollar esta versión.

6 Bibliografía

[1] Open Computer Vision Library. Disponible en: <http://sourceforge.net/projects/opencvlibrary/>. Consultada el 17 de marzo de 2015.
[2] OpenAL. Disponible en: <http://openal.org>. Consultada el 17 de marzo de 2015.
[3] J. M. Jarre. *Laserharp II*. Disponible en: <https://www.youtube.com/watch?v=9DlwQJX3qag>. Heineken Music Hall 20090526. Consultada el 17 de marzo de 2015.
[4] *The Legend of Zelda (Theme on Theremin)*. Disponible en: <https://www.youtube.com/watch?v=nJYho56INKU>. Consultada el 17 de marzo de 2015.
[5] E. Bonet y J. I. Cardona. (2010). Sonido en movimiento. Trabajo de la asignatura Integración de Medios Digitales.