UNIVERSIDAD POLITECNICA DE VALENCIA

Secuencias de vídeo desde fichero y cámara con OpenCV

Apellidos, nombre	Agustí Melchor, Manuel[1] (magusti@disca.upv.es)
Departamento	[1]Dpto. De Ing. De Sistemas y Computadores
Centro	Universidad Politécnica de Valencia

UNIVERSIDAD POLITECNICA DE VALENCIA

1 Resumen

En este artículo vamos a presentar las funciones de adquisición de imágenes y vídeo de OpenCV [1-3]. También veremos como manipular estas secuencias de imágenes obtenidas en directo desde una cámara (una “cámara web” por la alta disponibilidad de estas en los equipos presentes, externas o internas) o a partir de ficheros de vídeo.

2 Introducción

Este artículo muestra cómo adquirir, generar y reproducir secuencias de vídeo utilizando las funciones de OpenCV. Los orígenes de datos en este caso pueden ser obtenidos de disco o de un dispositivo hardware de adquisición de imágenes. Siempre serán imágenes en formato de mapas de bits, en los formatos y CODEC conocidos por OpenCV

3 Objetivos

Una vez que el alumno se lea con detenimiento este documento, será capaz de:

- Tomar fotografías estáticas a partir de una cámara.
- Leer secuencias de imágenes de una cámara o de un fichero de vídeo de disco.
- Generar un fichero de vídeo como resultado de un proceso, el que sea, de nuestra aplicación.

4 Desarrollo

El vídeo es una secuencia de imágenes, así que las operaciones sobre una imagen estática serán posibles ahora si las secuenciamos a la adecuada velocidad. Empezaremos viendo cómo obtener las imágenes de un fichero de disco, para después obtenerlas de la cámara. Finalmente, veremos cómo guardar lo que hagamos sobre una secuencia de imágenes en un fichero en disco que pueda ser utilizado por un reproductor estándar de vídeo.

4.1 Cargar y visualizar un vídeo

Por ejemplo, se pueden abrir ficheros sin comprimir en AVI o MPEG, permitiendo parar y continuar la reproducción de la secuencia (tecla 'r') o reiniciar la misma (tecla 'i'). Cuando llega al final se para y espera ESC, pudiendo ser reiniciada. El fichero de vídeo a utilizar se recibe como primer parámetro de la línea de órdenes.

```
// gcc `pkg-config opencv --cflags –libs` opencv_player.c -o opencv_player
#include <stdio.h>
#include <cv.h>
#include <cxcore.h>
#include <highgui.h>

#define fOriginal "OpenCV player"

int main(int argc, char* argv[])
{
 int tecla, i, j;
 IplImage *imgOrg;
 CvSize imgSize; // size of visualisation image
 CvCapture* videoOrg;
 char nomVideoOrg[1024];
 double fps, tiempoEntreCuadros;
 int salir = 0, // Controla la condición de terminación de la aplicación
    parar; // La reproducc. del vídeo.
 long int nCuadro = 0;

 if (argc < 2)
 {
  fprintf(stderr, "Falta un parámetro: el nombre del archivo de vídeo a reproducir\n" );
  return( 1 );
 }

 videoOrg = cvCreateFileCapture( argv[1] );
 if (videoOrg == NULL)
 {
  fprintf(stderr, "Problemas abriendo el fichero de vídeo >%s<\n", nomVideoOrg );
  return( 2 );
 }

 printf("Mostrando el contenido de la fuente de vídeo: >%s<\n \
ESC, q, Q – Salir.\n”, nomVideoOrg );

 imgOrg = cvQueryFrame( videoOrg );
 if (imgOrg == NULL)
 {
  fprintf(stderr, "Problemas obteniendo imígenes en %s\n", nomVideoOrg );
  return( 3 );
 }
 nCuadro++;
 cvShowImage( fOriginal, imgOrg );
 cvNamedWindow( fOriginal, CV_WINDOW_AUTOSIZE );

 imgSize.width = imgOrg->width;
 imgSize.height = imgOrg->height;
fps = cvGetCaptureProperty( videoOrg, CV_CAP_PROP_FPS );
printf(“Vídeo de %dx%d y %f cuadros por segundo\n”, imgOrg->width, imgOrg->height, fps);

 while ( !salir )
 {
   cvShowImage( fOriginal, imgOrg );
   nCuadro++;
   fprintf(stdout, "%ld\r", nCuadro );
```

```
    fflush( stdout );

  tecla = cvWaitKey( tiempoEntreCuadros ) & 255;
  switch ( tecla )
  {
   case ESC: case 'q':  case 'Q':    // Si 'ESC', q ó Q, ¡acabar!
    salir = TRUE;
    break;
  } // Fin de "switch ( tecla )"

 cvReleaseCapture( &videoOrg );
 cvDestroyAllWindows( );
 return 0;
}// Fin de "main"
```

Listado 1. Ejemplos de reproducción de ficheros de vídeo en OpenCV.

El esquema del proceso es el siguiente:

- Inicializar la fuente de vídeo: *CvCapture*, *cvCreateFileCaptur.*
- Bucle
  - Cargar un cuadro de la secuencia de vídeo: *cvQueryFrame.*
  - Mostrar la imagen: cvShowImage.
- Liberar recursos y terminar.
- Observar que no se libera la imagen obtenida con *cvQueryFrame* por su naturaleza interna a OpenCV.

Observar que tras abrir la fuente de datos cvCreateFileCapture es necesario extraer el primer cuadro, con la función cvQueryFrame, para obtener las propiedades de la secuencia de imágenes: así, por ejemplo el número de cuadro por segundos que se obtiene como el resto de propiedades con la función cvGetCaptureProperty; y, en este caso, especificando la propiedad que se quiere averiguar CV_CAP_PROP_FPS.

Observar también que no se libera la imagen obtenida a partir de cvQueryFrame por su naturaleza interna a OpenCV, sí las restantes que se creen, como es habitual.

4.2 Captura y procesado de imágenes estáticas a través de la cámara

Veamos ahora cómo manipular imágenes en mapas de bits, obtenidas a través de una cámara con OpenCV, utilizando una cámara web como ejemplo más popular y usual de este tipo de periféricos. El esquema del proceso del ejemplo actual es:

```
#include <stdio.h>
#include <cv.h>
#include <cxcore.h>
#include <highgui.h>

#define AND &&
#define OR ||
#define fOriginal "Original"
#define fResultat "Resultat"

int main(int argc, char* argv[])
{
 int key, i, j;
 IplImage *imgOrg,
         *imgDst;
 CvCapture* videoOrg;
 int isColor, fps, frameW, frameH;
 long int nCuadres = 0;
 int  modo = 0;

  printf("Procesando la secuencia de imágenes recibida desde la cámara \n [ESC, q , Q -
Salir].\n" );
  videoOrg = cvCreateCameraCapture( (argc > 1? atoi(argv[1]) : CV_CAP_ANY)  );
  if (videoOrg == NULL)
  {
    fprintf(stderr, "Problemas abriendo la conexión con la cámara %d.\n",
          (argc > 0? atoi(argv[1]) : CV_CAP_ANY) );
   return( 2 );
  }

  cvNamedWindow( fOriginal, CV_WINDOW_AUTOSIZE );
  imgOrg = cvQueryFrame( videoOrg );
  if (imgOrg == NULL)
  {
    fprintf(stderr, "Problemas obteniendo imágenes de la cámara %d<\n",
          (argc > 0? atoi(argv[1]) : CV_CAP_ANY));
   return( 3 );
  }
  cvShowImage( fOriginal, imgOrg );
  cvMoveWindow( fOriginal, 0, 0 );

  imgDst = cvCreateImage( cvSize(imgOrg->width, imgOrg->height),
                                  imgOrg->depth, imgOrg->nChannels );
  cvNamedWindow( fResultat, CV_WINDOW_AUTOSIZE );
  cvShowImage( fResultat, imgDst );
  cvMoveWindow( fResultat, imgOrg->width + 10, 0);

  while( bContinuar )
  {
   imgOrg = cvQueryFrame( videoOrg );
   if (imgOrg == NULL )
   {
    bContinuar = 0;
   }
   else
   {
```

```
    cvShowImage( fOriginal, imgOrg );
    cvFlip( imgOrg, imgDst, modo );
    cvShowImage( fResultat, imgDst );

    key = cvWaitKey( 25 ) & 255;
     bContinuar = !(!imgOrg OR
       key == ESC OR key == 'q' OR key == 'Q' );
    }
  }// de   while( bContinuar )

 cvReleaseCapture( &videoOrg );
 cvReleaseImage( &imgDst );
 cvDestroyAllWindows( );
 return 0;
}// Fin de "main"
```

Listado 2. Adquisición de imágenes a partir de una *cámara web*.

- Inicializar la fuente de vídeo: *CvCapture*, *cvCreateCameraCapture.*
- Bucle
  - Cargar un cuadro de la secuencia de vídeo: *cvQueryFrame.*
  - Procesar y mostrar la imagen.
- Liberar recursos y terminar: cvReleaseCapture.

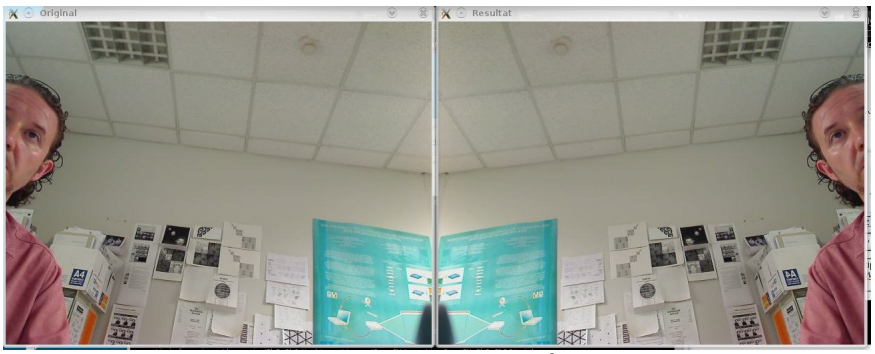

Figura 1. Ejemplo de salida del código de captura de imágenes desde la cámara.

4.3 Procesar y guardar un vídeo

En el anterior apartado se procede a partir de la carga de una imagen estática desde fichero o de la cámara, entonces se dispone de la información y se puede aplicar un procesado a la imagen en cuestión. Si se trata de una secuencia de imágenes (un vídeo), el procesamiento se aplica sobre cada uno de sus cuadros de forma repetida.

```
#include <stdio.h>
#include <cv.h>
#include <cxcore.h>
#include <highgui.h>

#define sFinestra "Grabando vídeo"

int main( int argc, char** argv )
{
 CvCapture* capture = NULL;
 IplImage* frame;
 char c;

 IplImage *bgr_frame;
 double fps;
 CvSize size;
 CvVideoWriter *writer;
 char nomVideoOrg[1024], nomVideoDest[1024];
 double tempsEntreCuadres;

  sprintf( nomVideoOrg, "%s", (argc > 1? argv[1] : "laser.mpg")  );
  sprintf( nomVideoDest, "%s", (argc > 2? argv[2] : "prova.mpg") );
  printf("Guardando la secuencia de imágenes de %s en el archivo %s\n [ESC, q , Q - Salir].\n",
        nomVideoOrg, nomVideoDest);

 cvNamedWindow( sFinestra, CV_WINDOW_AUTOSIZE )
  if( strcmp( nomVideoOrg, "-") == 0)
  {
   capture = cvCreateCameraCapture(0);
  } else {
   capture = cvCreateFileCapture( nomVideoOrg );
 }

 frame = cvQueryFrame(capture);//Init the video read
 cvShowImage( sFinestra, frame );
 if ((strcmp( nomVideoOrg, "-") == 0) OR (fps == 0))
   fps = 30;
 else
   fps = cvGetCaptureProperty ( capture, CV_CAP_PROP_FPS );

 size = cvSize( (int)cvGetCaptureProperty( capture, CV_CAP_PROP_FRAME_WIDTH),
              (int)cvGetCaptureProperty( capture, CV_CAP_PROP_FRAME_HEIGHT)
             );
 printf("Creant el writer de %dx%d\n",
        (int)cvGetCaptureProperty( capture, CV_CAP_PROP_FRAME_WIDTH),
        (int)cvGetCaptureProperty( capture, CV_CAP_PROP_FRAME_HEIGHT)
       );
 writer = cvCreateVideoWriter( "provaMPEG.mpg", CV_FOURCC('P','I','M','1'), fps, size, 1 );
 if  (writer == NULL )
 {
   printf("No se puede abrir provaMPEG.mpg \n");
   exit( .1 );
 }

 tempsEntreCuadres = 1000.0 /(double)fps;
 while(1) {
```

```
 frame = cvQueryFrame( capture );
 if( !frame ) break;
 cvShowImage( sFinestra, frame );

 printf("Grabant un cuadre ...\n");
 cvWriteFrame( writer, frame );
 printf("Grabat.\n");
 c = cvWaitKey( tempsEntreCuadres );
 if( c == 27 ) break;
}

cvReleaseVideoWriter( &writer );
cvReleaseImage( &logpolar_frame );
cvReleaseCapture( &capture );
cvDestroyWindow( sFinestra );

return(0);
}
```

Listado 3. Generando un fichero de vídeo MPEG.

Para ello deberá observase que en la secuencia de captura de imágenes desde la cámara o desde la lectura de un fichero, se ha de esperar a disponer de una imagen para entonces aplicar el proceso correspondiente. Este puede ser desde simplemente mostrarla hasta modificarla en función de lo que haya sucedido en relación con cuadros anteriores y mostrar un resultado distinto del contenido visual de entrada.

El esquema del proceso del ejemplo actual guarda en un AVI (con el CODEC MPEG) el resultado de reproducir un fichero de vídeo que recibe como parámetro. La secuencia de llamadas de funciones de OpenCV es:

- Inicializar la fuente de vídeo: *CvCapture*, *cvCreateFileCapture* / *cvCreateCameraCapture* y el destino de vídeo *CvVideoWriter*, *cvCreateVideoWriter*.
- Bucle
  - Cargar un cuadro de la secuencia de vídeo: *cvQueryFrame.*
  - Mostrar la imagen: *cvWriteFrame*.
- Liberar recursos y terminar: *cvReleaseVideoWriter*.

5 Concluyendo

A lo largo de este objeto de aprendizaje hemos visto ejemplos de uso operaciones sobre adquisición de imágenes tanto con origen en disco, como en directo a partir de una cámara web. El lector ha podido experimentar con.

- Tomar fotografías estáticas a partir de una cámara.
- Leer secuencias de imágenes de una cámara o de un fichero de vídeo de disco.

- Generar un fichero de vídeo como resultado de un proceso, el que sea, de nuestra aplicación.

Ahora toca ponerse a experimentar, por ejemplo:

- Tomar fotografías de la cámara o a partir de la reproducción de un vídeo al pulsa una tecla.

- Prueba a generar versiones de un mismo fichero de vídeo a otro cambiando el CODEC y comprobarás que se obtienen calidades diferentes y tamaños de ficheros diferentes. También puedes generar versiones de un mismo fichero con un mismo CODEC pero cambiando el número de cuadros por segundo, ¿se te ocurre qué se puede esperar de la calidad y ocupación del fichero resultante? Empieza con el MPEG, utilizando un número de cuadros por segundo de 30 y el CODEC `CV_FOURCC('D','I','V','X')`.

6 Bibliografía

[1] “Open Computer Vision Library”. Disponible en http://sourceforge.net/projects/opencvlibrary/

[2] Gary Bradski y Adrian Kaehler, “Learning OpenCV: Computer Vision with the OpenCV”, O'Reilly Press, Octubre 2008.

[3] “OpenCVWiki” , Disponible en http://opencv.w.illowgarage.com/wiki/Welcome.