UNIVERSIDAD POLITECNICA DE VALENCIA

Objetos gráficos complejos 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 dos objetos complejos realizados a partir de las funciones de dibujo nativas de OpenCV [1-3]. La complejidad reside en como se calculan las propiedades de los elementos gráficos: las coordenadas en pantalla. Con ello se demuestra que es posible utilizar estas opciones gráficas como elementos básicos de un interfaz mas elaborado. Aunque, recordémoslo, no es la tarea de OpenCV

2 Introducción

Este articulo muestra que se pueden crear contenidos visuales de cierta complejidad en cuanto a de dónde se adquiere la información y a cómo se representa en pantalla, que no es directamente realizable con las primitivas vectoriales que ofrece OpenCV, pero si a partir de ellas.

Se trata pues de familiarizarse con los parámetros de estas funciones, esto es las as propiedades de los gráficos que con ellas realizamos.

3 Objetivos

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

- Instanciar el histograma de una imagen, tanto en niveles de gris como en color, atendiendo a las características de cada una y con un formato establecido.
- Combinar la obtención de información del computador con una representación gráfica para su visualización.

4 Histogramas

Es un objeto que es básicamente un caso concreto de polilíneas. El histograma de una imagen nos muestra la distribución de colores de la misma. Si es una imagen en grises, los niveles de gris. En general, nos muestra la distribución de valores que se da en una serie de muestras.

En el caso de imágenes queremos obtener algo del estilo de las fig. 1 y 2, donde se muestra con las características propias de cada una los niveles de gris o de color que aparecen en una imagen. Para el primer caso se muestra una sola gráfica, para el segundo una por componente con un color para facilitar la interpretación de cual corresponde a cada componente del espacio de representacin RGB.

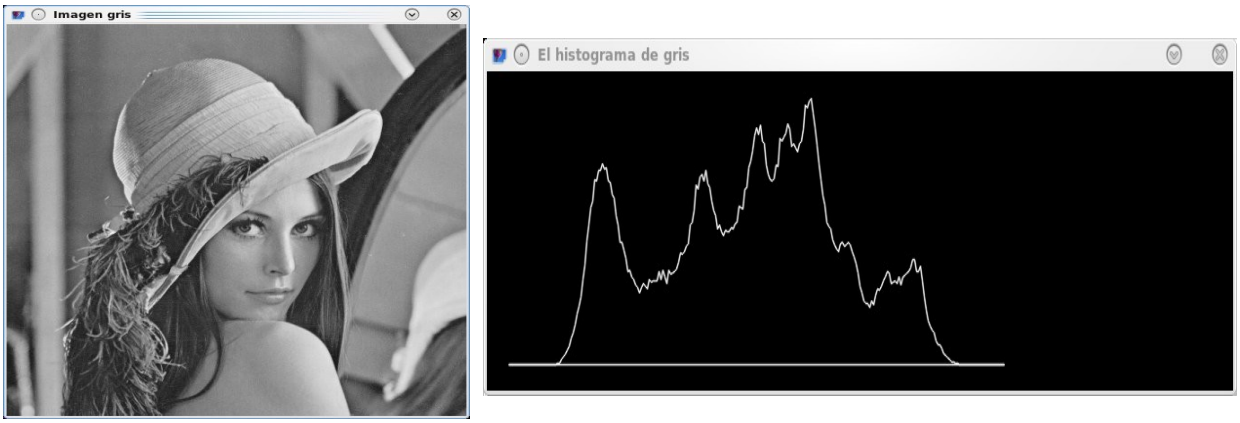

Figura 1: Una posible representación de un histograma de una imagen en niveles de gris en OpenCV.

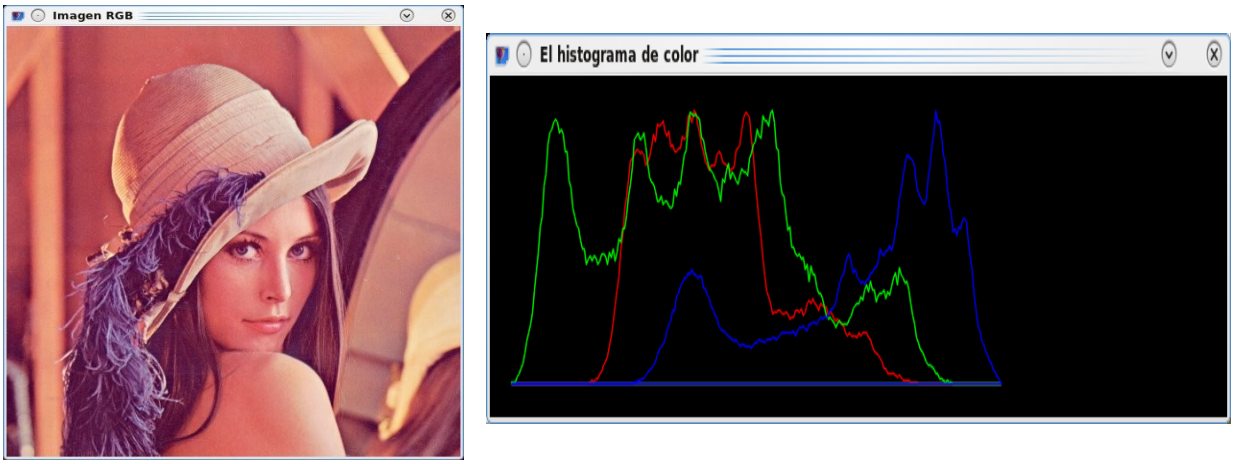

Figura 2: Una posible representación de un istograma de niveles de color de una imagen RGB (en este caso) en OpenCV.

Para facilitar el caso de estudio, proponemos ver una función mas simple y dejar que sea el lector el que la remodele a su gusto para esta tarea. La función del listado 1, dibuja sobre la imagen que recibe como parámetro el histograma predefinido. Observar que en un caso real debería ser calculado para la imagen en cuestión.

El histograma del ejemplo esta implementado como un vector con una serie de valores. Cada uno de ellos representaría cuantas veces aparece el nivel de gris i-ésimo en una imagen.

Observar que, básicamente, lo que nos hemos propuesto hacer es crear una correspondencia entre los valores numéricos del histograma (vector *histo*) y su representación gráfica mediante una polilínea (vector *puntsHisto*). Para ello la función define un histograma fijo y unas variables internas que parametrizar el dibujo de la polilínea. En un caso general deberían estar definidos en otra parte para adaptarse a la imagen o para permitir que el usuario los defina.

Nuestro ejemplo tiene fijos los valores a representar así como los que guían esta representación: los margenes del dibujo o la distancia de este respecto a los bordes de la imagen (ventana), la separación entre cada dos valores representados, etc.

```
void  pintaHistograma(IplImage* image)
{
 // Valores fijos
 int  histo[]={10, 0, 30, 0, 10, 0, 100, 0, 80, 0, 140, 0, 50, 0, 10};
 int margeBaix = 20, int margeEsq = 20;
 int base =  (int)image->height – margeBaix,
     separacioEntrePunts=10,
     nPunts=15;
 int contours = 1, is_closed = 1, thickness = 1, line_type = CV_AA, shift = 0;
 // Variables
 int  i=0,  nPuntsHisto;
 // Punts (coordenaes de pantalla a on pintar-lo: en fique dos m?s per vore si
 CvPoint  *puntsHisto;

 nPuntsHisto = nPunts + 2;
 puntsHisto = (CvPoint*)malloc( nPuntsHisto * sizeof(puntsHisto[0]));
 base = (int)image->height - margeBaix;

 puntsHisto[0] = cvPoint(margeEsq +(1*separacioEntrePunts), base+5);
 for (i=1; i<=nPunts; i++)
   {
    puntsHisto[i] = cvPoint(margeEsq +(i*separacioEntrePunts), base - histo[i-1]);

   }
 puntsHisto[nPunts+1] = cvPoint(margeEsq +(nPunts*separacioEntrePunts), base+5);
 cvFillConvexPoly(image, puntsHisto, nPunts+2, CV_RGB(255,128,64), line_type, shift);
}// Fi de " void  pintaHistograma( ..."
```

Listado 1. Ejemplos de primitivas gráficas en OpenCV.

Esto es, el código va recorriendo los valores del vector original para construir un nuevo vector, con parejas de coordenadas (columna y fila) que son las que pueden representar las primitivas gráficas.

Cada una de ellas esta construida a partir del valor a representar y los valores que presentan los margenes (base, margeEsq) y la separación entre puntos. Así, se obtienen resultados como los mostrados por las figuras de la tabla 1. El punto inicial (el 0) y el final (nPunts+1) se establecen siempre a valores prefijados.

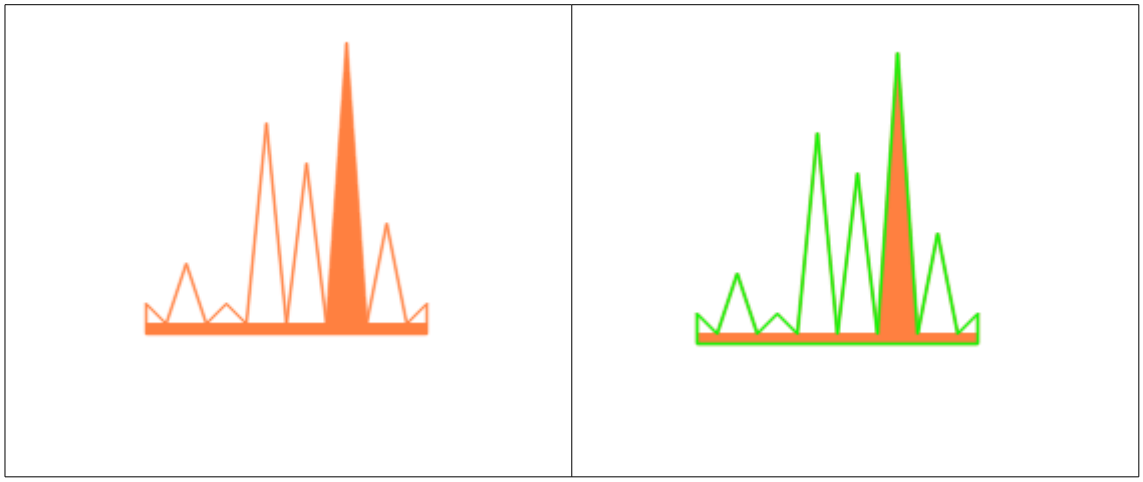

Tabla 1: Histograma dibujado por el codigo de este apartado a la izquierda. A la derecha, sugerencia final del listado 2 .

Como sugerencia final, en el listado 2 se muestra una modificación que se utilizaria para representar solo el borde del histograma y, en este caso, para resaltarlo.

```
...
  cvPolyLine( image, &puntsHisto, &nPuntsHisto, contours, is_closed,
              CV_RGB(0,255,0), CV_FILLED, line_type, shift);
 }// Fi de " void  pintaHistograma( ..."
```

Listado 2. Ejemplos de primitivas gráficas en OpenCV.

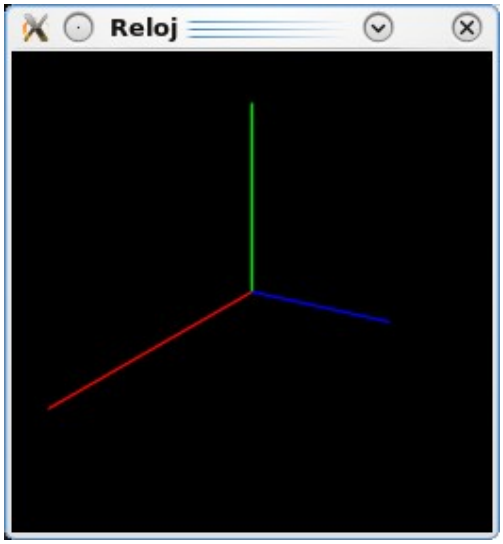

Figura 3: Captura de un instante de la ejecución: las 20:00:17.

5 Un ejemplo final: un reloj

Venga, hagamos un reloj analógico con tres varillas para las horas (en rojo), los minutos (en verde) y los segundos (en azul). La gracia de este ejemplo es ver cómo compaginar el sistema de coordenadas de la imagen, con el origen en la coordenada (0,0) y el de las varillas, en el centro de la imagen.

Hagamos uso de las matemáticas y dibujemos esas tres varillas como tres líneas de longitud y color diferente. Todas empiezan en un mismo punto y hay que ver dónde acaban. Un ejemplo de la salida de este código se puede ver en la fig. 3. El esquema de trabajo utilizado se basa en:

- Crear una imagen y una ventana.
- Bucle
  - Incializar la imagen a negro: `cvZero`.
  - Averiguar la hora y pintar las "varillas": `cvLine`.
  - Mostrarla en pantalla: `cvShowImage`.
  - Esperar un segundo: `cvWaitKey`.
- Liberar recursos y terminar.

El programa principal es el encargado de averiguar la información a mostrar, en este caso la hora del sistema, para lo que hace repetidas llamadas a time y localtime, como lo podría hacer a otras cuestiones.

La parte interesante es como maneja OpenCV esta información, que es la función *pintarReloj*. Esta, pinta las varillas, para lo que ha dibujar tres segmentos, para los que ha de decidir un valor inicial y uno final. Esto es proyectar cada segmento de acuerdo con un origen de coordenadas y un criterio para el ángulo. Esto se ve en :

- Centrar el dibujo. Lo que es lo mismo, averiguar las coordenadas del centro de la imagen para utilizarlas como centro de coordenadas. Si no, el (0,0), sería la esquina superior izquierda.
- Obtener una correspondencia de unidades de horas, minutos y segundos, por separado, Como las horas varían en un rango de 0 a 11 y los minutos y segundos de 0 a 59, se propone que cada uno muestre un paso de ángulo conforme a su rango dinámico.
  Además, como 0 es el ángulo del eje horizontal hay que moverlo hasta hacerlo vertical y que las 0 horas (por ejemplo) coincida con las 12 en el reloj.
- Obtener las coordenadas con *cvPoint* y pintar con *cvLine*, especificando los parámetros necesarios:longitud, color y formato de la línea para diferenciar horas de minutos y segundos

El listado 3 muestra el código completo que realiza esta tarea, En el supuesto de que el fichero en que se haya guardado el fuente se llame *opencv_reloj.c* se compilará, desde la línea de órdenes en GNU/Linux (por ejemplo), con la orden:

```
$gcc `pkg-config opencv --cflags --libs` opencv_reloj.c -o opencv_reloj
```

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

void pintarReloj( IplImage *img, int hora, int minuts, int segons );

int main(int argc,char* argv[]) {

 time_t tiempo;
 struct tm *tlocal;
 IplImage *imgOrg;

 cvInitSystem( argc, argv );
 printf("OpenCV version: %s\n", CV_VERSION);

 imgOrg = cvCreateImage(cvSize(255, 255),IPL_DEPTH_8U, 3);
 if (!imgOrg)
   {
    fprintf(stderr, "Problemas al crear la imagen.\n");
    return( 1 );
   }

 cvNamedWindow("Reloj", CV_WINDOW_AUTOSIZE );
 cvShowImage( "Reloj", imgOrg );
 cvMoveWindow( "Reloj", 0, 0 );

 pintarReloj( imgOrg, atoi(sHora), atoi(sMin), atoi(sSeg) );
 cvShowImage( "Reloj", imgOrg );

 do{
  tiempo = time( NULL );
  tlocal = localtime(&tiempo);
  printf("Hora %d:%d:%d\n", tlocal->tm_hour, tlocal->tm_min, tlocal->tm_sec);
  fflush( stdin );

  pintarReloj( imgOrg, tlocal->tm_hour, tlocal->tm_min, tlocal->tm_sec );
  cvShowImage( "Reloj", imgOrg );

 }while( (cvWaitKey(1000) & 255) != 27);

 return 0;
}


#define COLOR_HORES CV_RGB(255,0,0)
#define COLOR_MINUTS CV_RGB(0,255,0)
#define COLOR_SEGONS CV_RGB(0,0,255)

#define PI 3.14159265358979323846

float grausRadians( int graus )
{
 return( (graus * 2 * PI ) / 360);
```

```
}

int proyectarX(int llongitut, int graus)
{
  float aux;
  aux = llongitut * cos( grausRadians( graus ));

  return( round( aux ));
}

int proyectarY(int llongitut, int graus)
{
  float aux;
  aux = llongitut * sin( grausRadians( graus ));

  return( round( aux ));
}

#define LLONGH 125
#define LLONGM 100
#define LLONGS 75


void pintarReloj( IplImage *img, int hora, int minuts, int segons )
{
  int centroX, centroY,
      thickness = 1,
      line_type = CV_AA,
      shift = 0;
  int grausHora, grausMinuts, grausSegons;

  cvZero( img );
  centroX = img->width/2;
  centroY = img->height/2;

  grausHora = (hora - 3) * 30; //(360 / 12);
  grausMinuts = (minuts - 15) * 6; //(360 / 60);
  grausSegons = (segons - 15) * 6; //(360 / 60);

  cvLine(img,
         cvPoint(centroX,centroY),
            cvPoint(centroX+proyectarX(LLONGH, grausHora), centroY+proyectarY(LLONGH,
grausHora)),
         COLOR_HORES, thickness, line_type, shift );
  //  printf("Hora %d en %d,%d\n", hora, proyectarX(LLONGH, grausHora),
proyectarY(LLONGH, grausHora));

  cvLine(img,
         cvPoint(centroX,centroY),
            cvPoint(centroX+proyectarX(LLONGM, grausMinuts),
centroY+proyectarY(LLONGM, grausMinuts)),
         COLOR_MINUTS, thickness, line_type, shift );

  cvLine(img,
         cvPoint(centroX,centroY),
            cvPoint(centroX+proyectarX(LLONGS, grausSegons), centroY+proyectarY(LLONGS,
```

```
grausSegons)),
        COLOR_SEGONS, thickness, line_type, shift );

}
```

Listado 3. Un reloj analógico dibujado con las primitivas gráficas de OpenCV.

6 Concluyendo

A lo largo de este objeto de aprendizaje hemos visto:

- Un histograma de una imagen, con unas características y formato preestablecido.
- Una imagen dinámica, cuyos cambios vienen dador por el procesado de la información que se adquiere del "exterior" del programa.

Ahora toca poner de tu parte, por ejemplo:

- Ver el histograma de una sóla componente o todas con la ayuda de un control deslizante. Mostrar los valores del histograma asociados a la posición del ratón o con otro tipo de control.
- Añadiendo más detalles al reloj: unos círculos para cerrar la esfera, un fondo para esta, un texto como si fuera la marca del fabricante, las posiciones de las horas en la esfera, ..., o lo que esta viendo la cámara cada 10 (por ejemplo) segundos.

7 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.