UNIVERSIDAD POLITECNICA DE VALENCIA

Introducción al empleo de técnicas de Visión por Computador mediante 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 una introducción al tema de procesado de secuencias de imágenes utilizando el API de OpenCV. Aunque es un material de apoyo, está pensando de forma que la realización práctica de lo presentado en este documento conduzca a la obtención de una serie de resultados experimentales: tanto código como imágenes de resultados.

Se va a presentar un enfoque que permita trabajar en cualquier sistema operativo desarrollando sistemas de características multimedia que hagan uso de la imagen, en este caso, como un medio característico del que extraer y sobre el que mostrar información.

2 Introducción

El presente desarrollo está encaminado a ofrecer una perspectiva final de proyecto de integración. Para ello, se va a participar en la implementación de un prototipo de aplicación que utiliza las imágenes como medio principal de entrada y de salida de información. Estas aplicaciones utilizan técnicas de Visión por Computador (VxC) para análisis y procesado de imágenes en mapa de bits.

Los objetivos de este objeto de aprendizaje son:

- Dar a conocer al alumno, de un modo participativo, el uso de OpenCV para llevar a cabo operaciones básicas de VxC.
- Utilizar una plataforma de experimentación abierta y multiplataforma que permita al lector proseguir su autoaprendizaje de forma independiente.

OpenCV es una biblioteca de código abierto para tareas de visión por computador originalmente desarrollada por Intel y publicada bajo licencia BSD, que permite que sea usada libremente para propósitos comerciales y de investigación. La biblioteca es multiplataforma, puede ser usada en Mac OS X, MS/Windows y GNU/Linux. El logotipo original se puede ver en la Figura 1.

En la web, se puede encontrar para descargar la última versión en el sitio web de *Sourceforge* [1] y la explicación teórica de sus conceptos en el libro "oficial" de OpenCV: "Learning OpenCV: Computer Vision with the OpenCV Library" [2].

Figura 1. Logotipos de OpenCV: el inicial de Intel (izquierda) y el actual del OpenCVWiki (derecha), tomados de los sitios web de ambos.

UNIVERSIDAD POLITECNICA DE VALENCIA

3 Objetivos

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

- Determinar qué pasos son necesarios para escribir una aplicación que utilice OpenCV.
- Manipular (instanciar y modificar) una imagen en un computador en términos de OpenCV.
- Generar un ejecutable a partir del código creado y las bibliotecas de programación necesarias, sin instalar ningún entorno pesado, desde la línea de órdenes.
- Reconocer los errores de este tipo de aplicaciones, atendiendo a los resultados devueltos por las operaciones utilizadas.

4 Desarrollo

En la actualidad los computadores tienen potencia de cálculo suficiente como manipular tipos de datos complejos, en particular, al disponer de periféricos que permitan mostrar y adquirir información en formato visual, las capacidades de comunicación entre las personas y las máquinas han alcanzado un punto interesante al abrir un camino “natural” de interacción entre ambos.

Antes de iniciar una tarea más compleja es necesario conocer las **bases**, así que es momento de ponernos manos a la obra. Vamos a ver las operaciones más básicas: cómo es una imagen para un computador y cómo se puede manipular una imagen desde un interfaz de programación (API[1]) multiplataforma de amplia difusión como es OpenCV. Se desarrollarán pequeños programas para la tarea que se aborde en cada punto, En estos se trata de familiarizarse con las funciones y el modo de trabajo de OpenCV. En cada sistema se instalará de una forma, pero siempre se puede descargar del sitio web en [1].

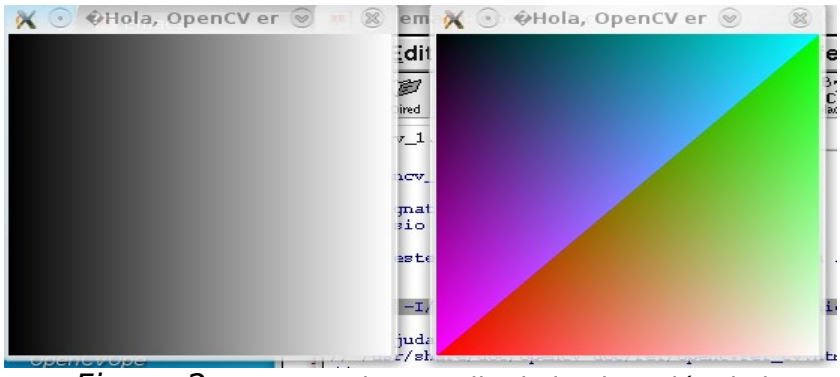

Figura 2.- Captura de pantalla de la ejecución de las aplicaciones propuestas en este artículo.

En este texto vamos presentar un esquema de trabajo: cómo se estructura y compila un programa que utiliza el API de OpenCV, qué parámetros pueden recibir y qué resultados devuelven. Es aconsejable tener la ayuda siempre a

[1] Colaboradores de Wikipedia. *Interfaz de programación de aplicaciones* [en línea]. Wikipedia, La enciclopedia libre, 2010 [fecha de consulta: 29 de junio del 2010]. Disponible en<http://es.wikipedia.org/wiki/Interfaz_de_programaci%C3%B3n_de_aplicaciones>

mano, puedes comprobar que está junto a la instalación realizada[2], otras referencias útiles están disponibles en [2], [3] y [4]. El ejemplo que vamos a desarrollar va a obtener una salida final en pantalla del estilo de la Figura 2.

4.1 Un primer programa

Este ejemplo es casi lo más básico que se puede hacer, se pueden utilizar menos líneas de código, pero este primer ejemplo nos permite la creación, manipulación, visualización de una imagen y liberación de recursos. ¡Qué más se le puede pedir al primer ejemplo! Es el que aparece en el Listado 1.

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

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

  img = cvCreateImage(cvSize(255, 255),IPL_DEPTH_8U, 1);
  cvZero( img );

  cvNamedWindow( "Hola, OpenCV", CV_WINDOW_AUTOSIZE );
  cvShowImage( "Hola, OpenCV", img );

  cvWaitKey(0);

  cvDestroyAllWindows();
  cvReleaseImage( &img );

  return( 0 );
}
```

Listado 1. Un primer programa en OpenCV,

Para ejecutarlo se compila rápidamente desde la línea de órdenes con

```
$ gcc `pkg-config opencv --cflags --libs` ejemploOpenCV_1.c -o ejemploOpenCV_1
```

Atención a las comillas, que son las de ejecución: es como poner una tilde abierta a un espacio en blanco. Se ejecuta con la orden

```
$ ./ejemploOpenCV_1
```

La ejecución termina al pulsar cualquier tecla estando activa la ventana de OpenCV, observe que no basta con cerrar la ventanas de resultados. ¿Simple, verdad? ¡Pero es un primer paso, no esta mal!

Como se habrá observado, todo gira en torno a la representación en mapa de bits de las imágenes en la estructura *IplImage*, que no se muestra en su totalidad por brevedad:

```
typedef struct _IplImage {
    int  nSize;         /* sizeof(IplImage) */
```

[2]En GNU/Linux es habitual que se encuentre en /usr/share/doc/opencv-doc/ref/opencvref_cv.htm.

```
    int  nChannels;     /* Most of OpenCV functions support 1,2,3 or 4 channels */
    int  depth;         /* pixel depth in bits */
    int  width;         /* image width in pixels */
    int  height;        /* image height in pixels */
    char *imageData;  /* pointer to aligned image data */
    int  widthStep;   /* size of aligned image row in bytes */
    ...
  } IplImage;
```

Las variables de tipo *IplImage* son inicializadas como estructuras en memoria con la función *cvCreateImage* y que se asignan valores a los píxeles de la imagen con la función *cvZero*.

La implementación utilizada en el ejemplo se basa en las definiciones de variables del tipo de datos *IplImage* y las funciones .*cvCreateImage, cvNamedWindow, cvShowImage, cvSize, cvWaitKey, cvDestroyAllWindows* y *cvReleaseImag*. El esquema de trabajo utilizado en el ejemplo se basa en:

- Declarar las variables: usando el tipo *IplImage*.
- Inicializar la imagen con `cvCreateImage`.
- Asignar a cada punto de la imagen un valor cero (color negro) con *cvZero*.
- Mostrar la imagen en pantalla con *cvNamedWindow*, *cvShowImage* y *cvSize*, hasta que se pulse tecla (*cvWaitKey*).
- Liberar recursos y terminar, con *cvDestroyAllWindows* y *cvReleaseImag*.

Ahora debes reflexionar sobre el código propuesto, por que las funciones señalas serán habituales en la mayoría de casos. Ahora toca convertirte en un elemento crítico: revisa el código, asegúrate que sabes qué hace cada función.

Por ejemplo:

- ¿Qué propiedades de una imagen se definen al crearla con *cvCreateImage*?
- ¿Te has dado cuenta que la función de creación de ventanas (*cvNamedWindows*) no muestra ninguna imagen de por si? ¿Que recibe varios parámetros y los valores que pueden tomar? ¿Has caído en la cuenta de que no hará falta llamarla si quieres actualizar los datos mostrados? Pero sí que habrás de utilizar *cvShowImage*.
- *cvWaitKey* ¿espera un tiempo dado en segundos o milisegundos? ¿Te has dado cuenta que si no está activa alguna ventana creada por OpenCV no se obtiene respuesta?

Esta ejemplo adolece de la falta de control de errores, lo cual es fuente de errores en tiempo de ejecución que no suelen tener una respuesta rápida debido a la dificultad de averiguar en qué punto del programa se puede dar un mal funcionamiento. Por este motivo, el siguiente ejemplo aborda la identificación de las situaciones que típicamente es conveniente confirmar que se llevan a cabo correctamente. De paso se aprovechará para observar otras propiedades y funciones de OpenCV.

4.2 Accediendo a las componentes de una imagen

En este apartado se describirá cómo se crea una imagen de niveles de gris, cómo se accede a los píxeles, cómo se averiguan las propiedades de una imagen, en este caso nos preocupamos sólo de las que afectan a la resolución de la misma.

La implementación utilizada en el ejemplo se basa en las definiciones de variables del tipo de datos *IplImage* y las funciones *.cvCreateImage, cvNamedWindow, cvShowImage, cvSize, cvWaitKey, cvDestroyAllWindows, cvReleaseImag, cvGet2D y cvSet2D*. El esquema de trabajo utilizado en el ejemplo se basa en:

- Declarar las variables.
- Inicializar la imagen con `cvCreateImage`.
- Bucle de recorrido de la imagen.
- Asignar un nivel de gris a cada punto de la imagen con `cvSet2D`.
- Liberar recursos y terminar.

El código es el del Listado 2. Las imágenes son inicializadas como estructuras en memoria con la función cvCreateImage() y que se asignan valores a los píxeles de la imagen con la función cvSet2D accediendo a los elementos de la imagen por su número de columna (x) y su número de fila (y). Atención al orden en que se deben escribir las coordenadas: primero la fila, luego la columna. El tipo de cada componente es char (8 bits, sin signo) por lo que pueden tomar valores en el rango [0,255].

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

#define fGRIS "Hola, OpenCV en gris!"
#define ANCHO 255
#define ALTO 255

// Declaraciones de prototipos locales
int  incializarImgGris( IplImage *imgOrg );

// Programa principal
int main(int argc, char* argv[])
{
    IplImage *imgOrgGris;

     cvInitSystem( argc, argv );
     printf("OpenCV version: %s\n", CV_VERSION);
    printf("Creando una imagen y mostrándola en pantalla [Cualquier tecla – Salir].\n");

    imgOrgGris = cvCreateImage(cvSize(ANCHO, ALTO),IPL_DEPTH_8U, 1);
    if (!imgOrgGris)
    {
      fprintf(stderr, "Problemas al crear la imagen en grises\n");
      return( 1 );
    }
```

```
    fprintf(stderr, "Se ha creado una imagen de %dx%d, de %d planos.\n",
          imgOrgGris->width, imgOrgGris->height, imgOrgGris->nChannels );

    incializarImgGris( imgOrgGris );

    cvNamedWindow( fGRIS, CV_WINDOW_AUTOSIZE );
    cvShowImage( fGRIS, imgOrgGris );
    cvWaitKey(0); // Espera una tecla los milisegundos que haga falta

    cvDestroyAllWindows( );
    cvReleaseImage( &imgOrgGris );
    cvReleaseImage( &imgOrgRGB );

    return 0;
   }

//
// Entrada: imgOrg - la imagen de partida.
// Salida: imgOrg - la imagen resultados que se devuelve modificando la original.
//
int  incializarImgGris( IplImage *imgOrg )
{
     int x, // indice de las columnas
         y; // indice de las filas
     CvScalar colorDst;

     if ( imgOrg->nChannels != 1)
      return( 1 );
     else
     {

     for ( x = 0; x < imgOrg->width; x++ )
      for ( y = 0; y < imgOrg->height; y++ )
      {
       colorDst.val[0] = x;
       cvSet2D(imgOrg, y, x, colorDst);
      } // Fin de "   for ( y = 0; y < imgOrg->height; y++ )"
     }// Fin de if-else ( imgOrg->nChannels != 1)
} // Fin de "int  incializarImgGris( IplImage *imgOrg )"
```

Listado 2. Accediendo a las componentes de una imagen de niveles de gris.

Esto ya es un poco más divertido, ¿eh? Pero también se ha hecho más complejo. Deberás comprobar que el esqueleto del programa principal no ha variado básicamente respecto del ejemplo inicial.

El cambio está básicamente en la función *inicializarImgGris*, que asigna valores a una imagen de un sólo plano de color (imagen de grises o de niveles de gris) con un gradiente de valores de gris. Esto es los valores de cada columna serán todos iguales y variarán linealmente desde la inicial (la de índice más bajo, 0) hasta la final (la de índice más alto, que no casualmente hemos hecho 255).

¿Has analizado el código? Deberías, por que el doble bucle de recorrido de las imágenes es necesario en muchas ocasiones y la función *cvSet2D* es una de las posibles maneras de modificar el contenido de un píxel de la imagen. Lo más interesante es ver cómo se manipula el color, en general, de un elemento de las imágenes a través del tipo *CvScalar*, que es un vector, del que ahora

manipulamos el primer elemento, el 0. Por que esta dinámica va a ser extensible a todas las imágenes, sólo a que un número mayor de planos de color, más componentes de este *CvSCalar* se verán involucrados.

Venga, vamos a complicarlo un poco más que hay situaciones en las que el color en las imágenes es interesante, ¿o no?

4.3 Accediendo a las componentes de una imagen en color

De forma análoga, es posible escribir sobre las componentes de color (por ejemplo una que tiene los tres planos RGB) de un píxel de la imagen asignándole la memoria correspondiente. Así que en el programa principal habrá que modificar la función *cvCreateImage*, para que le asigne tres plano a la variable,

La función de inicialización es similar a la anterior, está en el Listado 3: asignan valores a cada componente, utilizando la misma función e interpretando el contenido de color conforme indique la propiedad de número de componentes de esa imagen.

```
// Índices para los planos de una imagen de color
#define R 2
#define G 1
#define B 0
#define L 255

int  incializarImgRGB( IplImage *imgOrg )
{
     int x, // indice de las columnas
         y; // indice de las filas
     CvScalar colorDst;

     if ( imgOrg->nChannels != 3)
       return( 1 );
     else
     {

     for ( x = 0; x < imgOrg->width; x++ )
      for ( y = 0; y < imgOrg->height; y++ )
      {
       colorDst.val[R] = (double)y;
       colorDst.val[G] = (double)x;
       colorDst.val[B] = (double)((y+x) % (int)L);

       cvSet2D(imgOrg, y, x, colorDst);
      } // Fin de "   for ( y = 0; y < imgOrg->height; y++ )"
     }// Fin de if-else ( imgOrg->nChannels != 3)
} // Fin de "int  incializarImgRGB( IplImage *imgOrg )"
```

Listado 3. Accediendo a las componentes de una imagen en color.

El lector curioso habrá notado que se han creado unas macros para referencias las componentes de color más fácilmente. ¿Se ha percatado de en qué orden están en memoria?

Pero ¿ha notado cómo se ha conseguido crear los gradientes en cada componente como ya habrá observado en la anterior Figura 2? Ahora que no la

tiene a vista, ¿cree que puede avanzar que se obtendrá al ejecutar esta inicialización.

5 Concluyendo

A lo largo de este objeto de aprendizaje hemos visto qué estructura tienen las aplicaciones que utilizan el API de OpenCV. Dicha estructura queda reflejada en el gráfico siguiente.

- Determinar qué pasos son necesarios para escribir una aplicación que utilice OpenCV.
- Manipular (instanciar y modificar) una imagen en un computador en términos de OpenCV.
- Generar un ejecutable a partir del código creado y las bibliotecas de programación necesarias, sin instalar ningún entorno pesado, desde la línea de órdenes.
- Reconocer los errores de este tipo de aplicaciones, atendiendo a los resultados devueltos por las operaciones utilizadas.

¿Has leído hasta aquí y no has probado lo que se ha expuesto? Si no has llevas ejemplos a la práctica no tendrás un banco de pruebas donde probar tus propias ideas.

Ya se que los conceptos son muy básicos, pero te recuerdo que nuestros objetivos en este caso han sido cumplidos. Las nociones más básicas han sido presentadas, ahora te toca a ti atreverte a pensar en las imágenes como en un tipo de datos más que puedes procesar.

¿No te ha gustado el entorno de desarrollo? ¿Te parece que no tiene mucho provecho crear imágenes? Pues adelante, si te has haciendo preguntas es que tienes capacidad de aprender,e anota tus ideas y atrévete a llevar alguna hasta el final

Quien sabe, entre todos conseguiremos que los computadores puedan ver algo de lo que sucede a su alrededor y ofrezcan nuevas formas de interacción mucho más naturales para nosotros.

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.

[4] Vadim Pisarevsky, “Introduction to OpenCV”, Intel Corporation, Software and Solutions Group.