UNIVERSIDAD POLITECNICA DE VALENCIA

Interacción con las ventanas de 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 para la entrada y salida de datos nativa en OpenCV [1-3], esto es, desde cómo se muestran imágenes en ventanas, pasando por como se puede cargar una imagen desde un fichero, hasta cómo se pueden recoger datos del usuario mediante el teclado y el ratón.

2 Introducción

Para cualquier aplicación la entrada y salida de datos es importante. En el caso de aplicaciones de procesado de imágenes por computador lo es, como en cualquier otra aplicación, pero adquiere un nivel de complejidad extra al involucrar a las imágenes y a la interacción del usuario con estas.

Ejemplos de estas actividades son: mostrar las imágenes, guardarlas o leerlas utilizando diferentes formatos en disco, así como gestionar la visualización y refresco de las imágenes en pantalla.

Revisaremos qué ofrece OpenCV nativamente a respecto de estas cuestiones, aunque dejamos fuera un tema interesante por cuestiones de brevedad: la creación y manipulación de ficheros de texto estructurados: YAML y XML.

3 Objetivos

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

- Mostrar imágenes en ventanas y modificar sus propiedades.
- Guardar y cargar una imagen en (desde) un fichero.
- Recoger datos del usuario durante la ejecución del programa.

4 Desarrollo

Las funciones que vamos a explorar están a mitad camino entre puramente de interfaz visual del programa (asumiremos que estamos en un entorno gráfico) y las nucleares de OpenCV que hacen referencia a la manipulación de imágenes en disco.

4.1 Mostrar imágenes

En primer lugar crearemos una imagen para concentrarnos en las funciones relativas a la creación y manipulación de ventanas. El listado 1 muestra un pequeño código que tiene los pilares fundamentales.

El esquema de trabajo utilizado en el ejemplo se basa en:

- Inicializar las imágenes. Se crea una imagen con los píxeles a negro.
- Crear la ventana, con la función *cvNamedWindow*. A esta se le pasan dos parámetros, el primero permite identificar a la ventana, siempre que se la referencie debe utilizarse esta cadena de caracteres; el segundo dice si la ventana ha de cambiar de tamaño para mostrar al completo al imagen que

se le asocie. Si se el pasa la constante negada se observará el efecto. Se puede ver, en un caso y en el otro, que el ratón cambia de forma al acercarlo al borde de la ventana y permite redimensionarla, o no.

Es importante hacer notar que esta función sólo es necesario llamarla una vez: se crea la ventana y ya está. Para modificar sus propiedades no es necesario volverla a crear.

- Mostrar la imagen. La función *cvShowImage* hace que el contenido de una determinada imagen se muestre en pantalla en una determinada ventana. Esta se puede mover, ocultar, minimizar, etc. y no hay que preocuparse de esas tareas.

  No es una asociación en tanto que si se modifica la imagen no se actualiza en pantalla … A menos que se haga otro *cvShowImage*, claro. Se puede llamar a esta función cuantas veces se considere oportuno.

  Observar que la ventana se sitúa en pantalla en las coordenadas (0,0) pro que la función *cvMoveWindow* modifica la posición de la misma. Se puede mover tantas veces se quiera. Pruébalo.

- Esperar un tiempo máximo una entrada de teclado. Con *cvWaitKey* daremos tiempo a probar la aplicación y comprobar que lo se acaba de decir es cierto.

- Liberar recursos y terminar. Finalmente, es importante liberar recursos. Se puede cerrar una ventana una por una (con *cvDestroyWindow*, pasándole la cadena de caracteres que la identifica) ó todas las que hubiera creadas por la aplicación actual con *cvDestroyAllWindows*.

```
#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 );
  cvMoveWindow( "Hola, OpenCV", 0, 0 );
  cvWaitKey(0);

  cvDestroyAllWindows();
  cvReleaseImage( &img );

  return( 0 );
}
```

Listado 1. Un primer programa en OpenCV,

4.2 Cargar y guardar imágenes del sistema de archivos

Hagamos algo igual de interesante con las imágenes pero a partir de alguna ya existente y no esa indiferente imagen negra que acabamos de crear en el apartado anterior.

Se puede pasar una ruta hasta un fichero de tipo imagen en mapa de bits y dejar que se cargue en la variable que utiliza el ejemplo anterior cambiando las líneas que creaban e inicializaban la imagen en aquel por las del listado 2.

```
 img = cvLoadImage( argv[1], CV_LOAD_IMAGE_UNCHANGED);
if (!img)
{
  return( 1 );
}
cvSaveImage("imagen.png",img );
```

Listado 2. Operaciones de cargar y guardar en disco una imagen en OpenCV,

La función *cvLoadImage* permite llevar a memoria el contenido del fichero, que se puede cargar tal cual o forzar a que convierta a una imagen en grises con el segundo parámetro con valor *CV_LOAD_IMAGE_GRAYSCALE*. Personalmente, prefiero convertirla después y verla tal cual, por eso se utiliza otro valor en el trozo propuesto.

No hay que crear la imagen, reservar memoria, antes de leerla de disco. Es la propia función que lee el fichero de disco la que, a partir de la información de la cabecera, hace el trabajo.

Es importante señalar que a veces se da mal la ruta al fichero (en la mayor parte de los casos) o hay algún problema. Es la tarea de la sentencia condicional que sigue a la carga de datos. De no comprobar si se ha cargado correctamente el código sigue y fracasa en cualquier acceso a los contenidos de la imagen con un error difícil de seguir.

Este trozo de código también genera un fichero en el directorio de trabajo con la función *cvSaveImage*. Se pueden utilizar los formatos reconocidos por OpenCV, consulta la documentación para ver los que soporta la versión que tienes instalada. No tiene nada que ver el formato de la variable imagen con el del fichero de partida o el formato escogido para guardar (en este caso PNG), lo de los formatos es cosas de las funciones, no hay que preocuparse por ello, también es cierto que no hay mucho control sobre variar por ejemplo el grado de compresión de la imagen en disco al guardarla ...

4.3 Entrada y salida básica

Para recoger datos del usuario a través del *display*, entendiendo este como el teclado (periférico estándar de entrada) y el terminal (la salida estándar de un proceso): ¿cómo se pueden leer caracteres del teclado y cómo se pueden escribir mensajes?

Empezando por lo segundo con *printf* (o cualquiera de la familia) que permite escribir en los descriptores de salida asociados al proceso. Por ejemplo, el listado 3, es una recomendación, dice antes de terminar bruscamente (si hubiera algún error leyendo el fichero) que ha habido problemas con tal fichero,

para que el usuario pueda reaccionar. En caso de que la lectura se haya efectuado correctamente se muestran alguna de las propiedades de la imagen con *fprintf*.

```
 img = cvLoadImage( argv[1], CV_LOAD_IMAGE_UNCHANGED);
if (!img)
{
 printf("Problemas al cargar la imagen %s\n", argv[1]);
 return( 1 );
}
 fprintf(stderr, "Se ha cargado una imagen de %dx%d, de %d planos.\n",
       img->width, img->height, img->nChannels );
```

Listado 3. Salida básica de texto en OpenCV,

Por lo que respecta a la entrada de caracteres por teclado, si las ventanas de OpenCV tienen el foco del ratón no se puede acceder a los eventos que genera el entorno gráfico con las pulsaciones del teclado y las conversiones debidas al mapa de teclado, etc.

En este caso recurriremos a la función *cvWaitKey*, a la que le indicamos con el único parámetro que recibe cuántos milisegundos esperará esta entrada de teclado (si es 0 esperará el tiempo que haga falta).

Se puede sustituir la espera sin fin del ejemplo anterior con la del listado 4 donde se puede ver cómo se detecta la pulsación del la tecla *ESCape* para terminar el bucle y no se hace más que mostrar un mensaje, en otro caso.

```
// Asumiendo que se han declarado antes
 int tecla, // Tecla leída
      salir; // Controla la condición de terminación de la aplicación

...

 salir = 0;
 while ( !salir )
 {
 tecla = cvWaitKey(25) & 255;  // Espera una tecla 25 milisegundos

 switch ( tecla )
 {
  case 27:
  case 'q':
  case 'Q':    // Si 'ESC', q ó Q, ¡acabar!
    salir = 1;
    break;

 default: ;     fprintf(stdout, "Se ha pulsado la tecla %c (código %d)\n", tecla, tecla);
 } // Fin de "switch ( tecla )"
}// Fin de  "for(;;)"
```

Listado 4. Entrada básica de texto en OpenCV,

4.4 Entrada y salida gráfica

De una forma más visual los datos que proporciona el usuario se pueden obtener y mostrar mediante el uso de una (o varias) barra de desplazamiento asociada a la ventana donde se muestra una imagen.

El uso de una barra para asignar un valor a todos los puntos de la imagen se muestra en el listado 5 y una captura de la ejecución de la aplicación se muestra en la fig. 1. El ejemplo que desarrollamos en este apartado muestra una típica interfaz realizada con OpenCV en la que el parámetro que se varía está gobernado por un control de desplazamiento. La aplicación termina al pulsar la tecla *escape* o *q*.

Figura 1: Captura de pantalla en el momento de escoger el valor 223.

El esquema de trabajo utilizado en el ejemplo se basa en el uso de las funciones *cvWaitKey, cvCreateTrackbar, cvSet, cvAddS* y *cvZero* El esquema seguido es:

- Incializar: declarar la variable de tipo imagen, inicializarla y mostrar la magen
- Crear una barra de desplazamiento con `cvCreateTrackbar()`
  - Establece el rango entre [0, valorMax], el valor actual y la función encargada de actualizar el proceso a cada movimiento de la barra de desplazamiento.
- Bucle. Donde se actualiza el valor con que se inicializa la imagen al valor escogido en el control de desplazamiento.
- Liberar recursos y terminar

Es importante observar la función cambioDeValor que es la que responde al evento de modificar la posición del control, en tanto que recibe un único parámetro (que le hemos llamado posBarraDesplz) que es el valor actual y, como se muestra con el mensaje, también está accesible en la variable que se le haya asociado en la creación del control (valor).

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

#define fORG "Imagen original"
#define fDST  "Imagen umbralizada"
#define AMPLE 255
#define ALT 255
#define FALSE 0
#define TRUE 1

int valor = 0;
int colorAnterior = 0;
int  estaPintado = 0;

// Declaraciones de prototipos locales
void cambioDeValor( int pos );
int procesarImg( IplImage *imgOrg, IplImage *imgDst, int valor );

// Programa principal
int main(int argc, char* argv[])
{
 IplImage *imgOrg,*imgDst;
 int tecla, // Tecla leída
      salir; // Controla la condición de terminación de la aplicación

 if (argc > 1)  valor = atoi( argv[1] );

 imgOrg = cvCreateImage(cvSize(AMPLE, ALT),IPL_DEPTH_8U, 1);
 if (!imgOrg)
 {
  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",
           imgOrg->width, imgOrg->height, imgOrg->nChannels );
 cvNamedWindow( fORG, CV_WINDOW_AUTOSIZE );
 cvShowImage( fORG, imgOrg );
 cvMoveWindow( fORG, 0, 0 );

 cvCreateTrackbar( "Valor", fORG, &valor, MAXPIXEL, cambioDeValor );

 salir = FALSE;
 while ( !salir )
 {
  tecla = cvWaitKey(25) & 255;  // Espera una tecla los milisegundos que haga falta
  switch ( tecla )
  {
   case ESC:  case 'q': case 'Q':    // Si 'ESC', q ó Q, ¡acabar!
     salir = TRUE;
     break;
  default: ;
  } // Fin de "switch ( tecla )"
  procesarImg( imgOrg, imgOrg, valor );
  cvShowImage( fORG, imgOrg ); //fDST, imgDst );
```

```
}// Fin de  "for(;;)"
 printf( "Preparados para salir\n" );
 cvDestroyAllWindows( );
 cvReleaseImage( &imgOrg );
 return 0;
}

void cambioDeValor( int posBarraDesplz )
{
  printf("El nuevo valor de inicialización es %d (%d) \n", valor, posBarraDesplz);
}// Fin de "void cambioDeValor(...

// Inicializa una imagen a un valor
// Entrada: imgOrg - la imagen de partida (se obvia)
//              valor - el valor de color/gris a usar
// Salida: imgDst - la imagen resultado (en niveles de gris o color) resultante.
int procesarImg( IplImage *imgOrg, IplImage *imgDst, int valor )
{
  if (colorAnterior != valor)
   {
    colorAnterior = valor;
    cvZero( imgDst );
    cvAddS( imgDst, cvScalarAll( valor ), imgDst, NULL );
   }
} // Fin de "int procesarImg( IplImage *imgOrg, int umbral, IplImage *imgDst )"
```

Listado 5. Uso de barras de desplazamiento en OpenCV,

4.5 Recoger datos por parte del usuario: eventos del ratón

Desarrollaremos ahora la gestión de los eventos del ratón: posición del cursor y pulsación de los botones. Para ello estableceremos un manejador para las acciones de ratón con la función `cvSetMouseCallback()`.

Con ello añadiremos al ejemplo anterior la funcionalidad de mostrar el color (nivel de gris en este caso) del píxel bajo el cursor donde se haga *click* con el botón secundario. Por otra parte pintará en blanco sobre la imagen mientras se mantenga el botón principal pulsado.

Se habrá de añadir. previamente a la función principal, el prototipo de la función, se puede llamar como se quiera, pero debe tener los parámetros y tipos mostrado en el listado 6.

```
void my_mouse_callback(int event, int x, int y, int flags, void* param  );
```

Listado 6. Gestión de los eventos del ratón en OpenCV,

En la función principal, después de asignar la barra de desplazamiento, asignamos esta funcionalidad con la línea del listado 7. Es como la típica línea de creación de hilos en POSIX: el tercer parámetro puedes utilizarlo o no, en este caso hemos aprovechado para pasarle la imagen que se quiere modificar. En caso de no querer pasar nada se le asignará NULL.

```
cvSetMouseCallback( fORG, my_mouse_callback, (void*)imgOrg  );
```

Listado 7. Gestión de los eventos del ratón en OpenCV,

La función está centrada alrededor de la diferenciación del evento de ratón que ha generado el evento. Cada evento tiene asociado unas instrucciones que desarrollan el estado propuesto para cada acción del botón. Observar como finalmente todo consiste en mirar qué hay en una posición de la imagen (*cvGet2D*) ó modificarlo (*cvSet2D*). La posición (fila y columna de la imagen se obtiene directamente de la posición del ratón ya que están alineados, ambos "sistemas de referencia" tienen el (0,0) en la esquina superior izquierda. La implementación completa de la función se muestra en el listado 8.

Aunque queda alguna cosa más que explorar con el parámetro *flag*, el ejemplo muestra los usos más habituales: detectar cuándo se mueve el ratón, donde está en un momento dado ó qué botón del ratón ha modificado su estado y si ha sido pulsado o soltado.

```
// Procesado de eventos del ratón
void my_mouse_callback( int event, int x, int y, int flags, void* param )
{
  IplImage* image = (IplImage*) param;
  CvScalar  elColor;

  switch( event ) {
  case CV_EVENT_MOUSEMOVE:
    if( estaPintado == CIERTO )
    {
      cvSet2D(image, y, x, cvScalarAll( MAXPIXEL ) );
    }
    cvShowImage( fORG, image );
    break;

  case CV_EVENT_LBUTTONDOWN:
    estaPintado = CIERTO;
    cvSet2D(image, y, x, cvScalarAll( MAXPIXEL ) );
    break;

  case CV_EVENT_LBUTTONUP:
    estaPintado = FALSO;
    break;

  case CV_EVENT_RBUTTONUP:
    elColor = cvGet2D(image, y, x);
    if (image->nChannels == 1)
      printf(" En coordenadas %d,%d color %d\n", y, x, (int)elColor.val[0]);
    else
      printf(" En coordenadas %d,%d color %d, %d, %d\n", y, x,
           (int)elColor.val[0], (int)elColor.val[1], (int)elColor.val[2]);
    break;
  } // fi del switch
}// fi de "void my_mouse_callback( ..."
```

Listado 8. Gestión de los eventos del ratón en OpenCV,

5 Concluyendo

A lo largo de este objeto de aprendizaje hemos visto la mayor parte de los mecanismos de interacción con el usuario que ofrece OpenCV de forma nativa. El lector ha podido experimentar con operaciones de:

- Mostrar imágenes en ventanas y modificar sus propiedades.
- Guardar y cargar una imagen en (desde) un fichero.
- Recoger datos del usuario durante la ejecución del programa

Ahora toca ponerse a experimentar, ¿qué tal hacer una aplicación para mostrar todas esas fotos que tienes en tu disco? Piensa que no sólo puedes leer los ficheros, también puedes modificarlas antes de mostrarlas, mientras lo haces ... o casi (si no te pones manos a la obra) lo que se te ocurra.

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.