Ejercicios con OpenCV

Dr. Alfonso Alba Cadena
Facultad de Ciencias, UASLP
fac@fc.uaslp.mx

Septiembre 2011

1 Sobre OpenCV

OpenCV (Open Source Computer Vision) es una librería de funciones escritas en C/C++ para realizar tareas de procesamiento de imágenes y visión computacional que van desde lo básico (acceso a pixeles individuales, despliegue en pantalla, dibujo de formas geométricas) hasta lo avanzado (filtrado, detección de bordes, transformaciones geométricas).

OpenCV puede utilizarse bajo diversas plataformas (incluyendo Windows, MacOS, Linux, Android e iOS) y con diversos compiladores. En particular, en Windows soporta los compiladores Microsoft Visual C++ y MinGW (e.g., bajo Code::Blocks).

Este documento contempla la versión 2.2 de OpenCV.

2 Estructura IplImage

La estructura `IplImage` contiene la información de una imagen y sus características (para mayores detalles consultar la guía de referencia de OpenCV). Los principales campos de esta estructura son:

- `int width` - contiene el ancho de la imagen en pixeles.
- `int height` - contiene la altura de la imagen en pixeles.
- `int depth` - indica el formato de los pixeles, de acuerdo a las siguientes constantes
  - `IPL_DEPTH_8U` - Entero de 8 bits sin signo (e.g., escala de grises de 0 a 255).
  - `IPL_DEPTH_8S` - Entero de 8 bits con signo.
  - `IPL_DEPTH_16U` - Entero de 16 bits sin signo.
  - `IPL_DEPTH_16S` - Entero de 16 bits con signo.
  - `IPL_DEPTH_32S` - Entero de 32 bits con signo.

  - `IPL_DEPTH_32F` - Punto flotante de 32 bits (la escala de grises toma el rango de 0 a 1).
  - `IPL_DEPTH_64F` - Punto flotante de 64 bits (la escala de grises toma el rango de 0 a 1).

- `int nChannels` - número de canales (e.g., 1 para imágenes en escala de grises, 3 para imágenes RGB).
- `char *imageData` - apuntador al arreglo que contiene la imagen. El programador debe realizar la conversión al tipo de apuntador adecuado, de acuerdo al campo `depth`.
- `int widthStep` - número de bytes que ocupa cada renglón de la imagen en el arreglo `imageData`. Este valor puede ser mayor que el que requiere el ancho de la imagen.

La imagen contenida en una estructura `IplImage` está organizada como un arreglo bidimensional en orden lexicográfico; sin embargo, por razones de eficiencia, el número de columnas en este arreglo no es necesariamente igual al número de pixeles en cada renglón de la imagen. El campo `widthStep` de la estructura `IplImage` indica exactamente cuántos bytes ocupa cada renglón de la imagen dentro del arreglo. En otras palabras, el apuntador dado por `imageData + y * widthStep` precisamente apunta al inicio del y-ésimo renglón de la imagen. Sin embargo, es importante recordar que este apuntador está definido como apuntador a `char` y en muchos casos será necesario realizar un typecast al tipo de apuntador adecuado para el formato de la imagen.

Considere el siguiente ejemplo:

```
#include <opencv2/opencv.hpp>

int main() {
    IplImage *img;  // img es un apuntador a una estructura IplImage

    // Aqui creamos la estructura y reservamos espacio para una imagen
    // de 320x240 pixeles en formato de punto flotante (32 bits) y
    // en escala de grises (un solo canal)
    img = cvCreateImage(cvSize(320, 240), IPL_DEPTH_32F, 1);

    // Ahora asignamos al pixel (160,120) el color blanco
    float *row = (float *)(img->imageData + 120 * img->widthStep);
    row[160] = 1;

    cvShowImage("imagen", img); // Despliega la imagen
    cvWaitKey(-1);              // Espera una tecla
    cvReleaseImage(&img);       // Libera la imagen
    return 0;
}
```

3 Funciones Básicas

```
IplImage *cvCreateImage(CvSize size, int depth, int channels);
```

Crea una imagen con un tamaño, formato y número de canales dado, y devuelve un apuntador a la estructura `IplImage` correspondiente.

```
void cvReleaseImage(IplImage **image);
```

Libera la memoria utilizada por una imagen.

```
void cvSet(CvArr *arr, CvScalar value, const CvArr *mask = NULL);
```

Llena una matriz o una imagen con un mismo valor. El tercer parámetro es opcional y define una máscara que determina que pixeles serán modificados. Si el pixel correspondiente en la máscara es distinto de cero, entonces el pixel en la imagen de entrada tomará el valor `value`.

El valor asignado a los pixeles debe proporcionarse como una estructura de tipo `CvScalar` la cual contiene un arreglo `val` de cuatro elementos de tipo `double`. La forma mas fácil de crear un elemento `CvScalar` es mediante la función `cvScalar` que toma como parámetros entre 1 y 4 valores de tipo `double` y devuelve el `CvScalar` correspondiente.

Ejemplo de uso:

```
// Crea una imagen en formato RGB de tipo flotante y la llena con color rojo
IplImage *img = cvCreateImage(cvSize(320,240), IPL_DEPTH_32F, 3);
cvSet(img, cvScalar(1,0,0));
cvShowImage("imagen", img);
```

```
void cvSetZero(CvArr *arr);
```

Llena una matriz o una imagen con ceros.

```
void cvSetReal2D(CvArr *arr, int idx0, int idx1, double value);
```

Asigna el valor `value` al elemento de una matriz o una imagen localizado en el renglón `idx0` y columna `idx1`. Es importante notar que si `*arr` es una imagen (i.e., de tipo `IplImage *`), los índices `idx0` e `idx1` representan, respectivamente, las coordenadas Y y X del pixel a modificar.

```
double cvGetReal2D(CvArr *arr, int idx0, int idx1);
```

Devuelve el valor asignado al elemento ubicado en el renglón `idx0` y columna `idx1` de la matriz o imagen dada por `arr`.

```
void cvCopy(const CvArr *src, CvArr *dst, const CvArr *mask = NULL);
```

Copia el contenido de la matriz o imagen `src` en la matriz o imagen `dst`. Ambas imagenes deben tener el mismo tamaño, formato y número de canales. Dada la matriz o imagen `mask`, esta indicará cuáles pixeles serán copiados.

4 Funciones básicas de interface

```
void cvShowImage(const char *name, const CvArr *image);
```

Despliega la imagen `image` en una ventana cuyo nombre está dado por la cadena `name`. Si ya existe una ventana con el mismo nombre, la imagen de esa ventana será reemplazada.

Nota: En versiones de OpenCV anteriores a la 2.2 era necesario crear primero la ventana mediante la función `cvNamedWindow`.

```
void cvDestroyWindow(const char *name);
```

Destruye y cierra la ventana con nombre `name`.

```
void cvDestroyAllWindows();
```

Destruye y cierra todas las ventanas creadas con `cvShowImage` y `cvNamedWindow`.

```
int cvWaitKey(int delay = 0);
```

Esta función detiene la ejecución durante `delay` milisegundos, o indefinidamente (`delay <= 0`) hasta que se origine un evento del teclado. El resultado devuelto es el código de la tecla presionada, o bien -1 si no se presionó ninguna tecla antes de reanudar la ejecución.

5 Ejemplo: cvSetReal2D vs apuntadores

El siguiente ejemplo muestra la diferencia entre utilizar las funciones cvGetReal2D y cvSetReal2D y utilizar el apuntador `imageData` para acceder a los pixeles de una imagen.

```
#include <opencv2/opencv.hpp>
#include <ctime>
#include <iostream>

using namespace std;

#define TIME    10
#define TIMECLK (TIME * CLOCKS_PER_SEC)

int main() {
    IplImage *img;  // img es un apuntador a una estructura IplImage
    clock_t ti;
    int iter, x, y;
    double fps;
```

```
    float *row;

    img = cvCreateImage(cvSize(640, 480), IPL_DEPTH_32F, 1);

    // Llena una imagen continuamente con ruido utilizando cvSetReal2D
    iter = 0;
    ti = clock();
    while ((cvWaitKey(1) < 0) && ((clock() - ti) < TIMECLK)) {
        for (y = 0; y < img->height; y++) {
            for (x = 0; x < img->width; x++) {
                cvSetReal2D(img, y, x, (double)rand() / RAND_MAX);
            }
        }
        cvShowImage("imagen", img); // Despliega la imagen
        iter++;
    }
    fps = (double)iter * CLOCKS_PER_SEC / (clock() - ti);
    cout << "Frames per second (cvSetReal2D) : " << fps << endl;


    // Llena una imagen continuamente con ruido utilizando apuntadores
    iter = 0;
    ti = clock();
    while ((cvWaitKey(1) < 0) && ((clock() - ti) < TIMECLK)) {
        for (y = 0; y < img->height; y++) {
            row = (float *)(img->imageData + y * img->widthStep);
            for (x = 0; x < img->width; x++) {
                *row++ = (double)rand() / RAND_MAX;
            }
        }
        cvShowImage("imagen", img); // Despliega la imagen
        iter++;
    }
    fps = (double)iter * CLOCKS_PER_SEC / (clock() - ti);
    cout << "Frames per second (pointers) : " << fps << endl;

    cvReleaseImage(&img);
    return 0;
}
```