OpenCV: introducción

FICH, UNL - Procesamiento Digital de Imágenes
Walter Bedrij

17 de marzo de 2016

Este documento presenta una reseña de la librería OpenCV para procesamiento de imágenes en C++. Se explica brevemente su instalación, compilación y las funciones fundamentales para cargar una imagen de archivo, obtener información de la misma, visualizarla y otras cuestiones de manejo básico,

La librería y su documentación está disponible en: `http://opencv.org/`. En nuestras clases usaremos la versión 2.x.

1. Instalación

`http://docs.opencv.org/doc/tutorials/introduction/linux_install/linux_install.html`
`http://docs.opencv.org/doc/tutorials/introduction/windows_install/windows_install.html`

Existe un complemento para ZinjaI que contiene la librería en la versión 2.4.10 compilada con MinGW, autocompletado y plantilla de projecto `http://zinjai.sourceforge.net/`

2. Compilación

`OpenCV` se divide en módulos según la funcionalidad que estos proveen, por ejemplo el módulo `highgui` permite realizar interfaces gráficas, leer y escribir imágenes a disco y capturar video desde una cámara. Para evitar la búsqueda de los módulos correspondientes a las funciones a utilizar, se incluirán todos ellos mediante

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

Utilizando el programa `pkg-config` se resolverán las dependencias de `OpenCV` y se enlazará contra *todos* los módulos (no existe penalidad)

```
g++ prog.cpp $(pkg-config --libs opencv) -o prog
```

3. Manejo básico de una imagen

- Crear una imagen vacía:

```
cv::Mat img(filas, columnas, tipo, valor);
cv::Mat img = cv::Mat::zeros(filas, columnas, tipo);
```

  - `valor`: Inicialización de los píxeles, es de tipo `cv::Scalar`.

- `tipo`: Define la forma de representación de la matriz. Se compone de la siguiente manera
  `CV_[n bits por elemento][signo][prefijo de tipo]C([n de canales])`
  Así, `CV_8UC(3)` indica que se almacenará la imagen utilizando `unsigned char` de 8 bits, y que cada píxel se define con 3 canales.

Ejemplos:

- `cv::Mat img(640, 480, CV_8UC(1))`: imagen en tonos de grises, inicialmente negra.
- `cv::Mat img(640, 480, CV_8UC(3), cv::Scalar(0,0,255))`: imagen color, inicialmente roja (BGR).
- `Mat C = (Mat_<double>(3,3) << 0, -1, 0, -1, 5, -1, 0, -1, 0)`: imagen de 3×3 utilizada como máscara para filtros

- Crear una imagen cargándola de un archivo:
  ```
  cv::Mat img = cv::imread("archivo.ext");
  cv::Mat img = cv::imread("archivo.ext", CV_LOAD_IMAGE_GRAYSCALE);
  ```
  (convierte a escala de grises)

- Campos de la estructura `Mat`: accesibles a través de las siguientes funciones, mediante la instrucción `img.funcion`:

  - `columns`: ancho, número de columnas.
  - `rows`: alto, número de filas.
  - `channels`: dimensión del pixel, número de canales.
  - `depth()`: especificación de tipo.
  - `at<tipo>(R,C)`: acceso a un píxel, el tipo debe corresponderse con el la matriz.
  - `ptr<tipo>(R)`: puntero a una fila, el tipo debe corresponderse con el la matriz.

  La matriz se accede desde el elemento (0,0) en la esquina superior izquierda, hasta el elemento (ancho-1,alto-1) en la esquina inferior derecha.

- Visualización de una imagen:
  Para mostrar las imágenes es necesario crear ventanas, estas son accedidas luego mediante el nombre:
  ```
  cv::namedWindow("nombre");
  cv::imshow("nombre", img);
  ```

- Grabación de una imagen:
  Para guardar en disco una imagen, simplemente se llama a la función:
  ```
  cv::imwrite("nombre.ext", img);
  ```
  Las imágenes a color se consideran en orden BGR.

  Se puede utilizar un parámetro adicional para especificar opciones específicas del formato de imagen, como ser la compresión.

- Video:
  La clase `cv::VideoCapture` proporciona una interfaz para capturar video desde una cámara o desde archivo.
  `cv::VideoCapture video(0);` (cámara por defecto)
  `cv::VideoCapture video("archivo.ext");` (lee desde un archivo de video)
  `cv::VideoCapture video("img%02d.jpg");` (lee una secuencia de imágenes: `img00.jpg`, `img01.jpg`, ...)

  Para obtener el siguiente fotograma, simplemente se utiliza
  `video >> frame;`
  donde `frame` es de tipo `cv::Mat`

  La cantidad de canales de la imagen dependerá del dispositivo de captura, pudiendo luego convertirse mediante la función `cv::cvtColor()`

3.1. Copia

Las imágenes son representadas mediante una cabecera, que contiene información como el tamaño y tipo de la imagen, y un puntero inteligente que apunta a la zona de memoria donde están almacenados los píxeles.

Al utilizar el operador de asignación o el constructor de copia, se copian la cabecera y el valor del puntero. Es decir, se realiza una *shallow copy*. Este comportamiento se asemeja al que se observa en los objetos de *java*, *python* o *smalltalk*.

Como consecuencia, si se pasa una `cv::Mat` como parámetro a una función, es prácticamente indistinto si el pasaje es por copia o referencia. Cualquier cambio en el parámetro de la función se verá reflejado en la variable usada en la llamada (excepto realocaciones).

Si se requiere trabajar con una copia verdadera (una *deep copy*), se deberán utilizar los comandos `clone()` o `copyTo()`

```
b = a.clone();
c.copyTo(d);
```

4. Programas de ejemplo

Muestra la imagen pasada como parámetro.

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

int main(int argc, char **argv){
  if(argc not_eq 2){
    std::cerr << "Debe suministrar una imagen\n";
    std::cerr << argv[0] << " imagen\n";
    return 1;
  }
  cv::Mat image = cv::imread(argv[1], CV_LOAD_IMAGE_UNCHANGED);
  cv::namedWindow("Imagen", CV_WINDOW_KEEPRATIO);
  cv::imshow("Imagen", image);
  cv::waitKey();
```

```
  return 0;
}
```

Detección de bordes de una imagen tomada de una cámara web.

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

int main(int argc, char **argv){
  cv::VideoCapture capture(0); //cámara por defecto
  if(not capture.isOpened()) //error
    return 1;
  cv::Mat frame, edge;

  do{
    capture>>frame;  //siguiente fotograma
    cv::cvtColor(frame, edge, CV_BGR2GRAY); //conversión a escala de grises
    cv::GaussianBlur(edge, edge, cv::Size(11,11), 2.5, 0.5); //reducción de ruido
    cv::Canny(edge, edge, 0, 30, 3); //detección de bordes
    cv::imshow("original", frame);
    cv::imshow("borde", edge);
  }while( cv::waitKey(30)==-1 ); //salir cuando se presione una tecla

  return 0;
}
```

5. Diferencias con CImg

Algunas diferencias que podrían afectar el cursado

- `OpenCV` no soporta el formato `gif`. Puede utilizarse el programa `imagemagick` o similar para convertir a un formato manejable (como ser `tiff`).
  ```
  convert archivo.gif archivo.tiff
  ```
- `OpenCV` maneja las representaciones de color `HSV` y `HLS`, pero no `HSI`.
- Una ventana puede mostrar solo una imagen a la vez.

6. Comentarios finales

Este documento es de libre distribución y reproducción total o parcial por cualquier medio. Comentarios y sugerencias a los contactos de `e-fich.unl.edu.ar` curso Procesamiento Digital de Imágenes.