Manejo de Bibliotecas Opencv

Alejandro Furfaro

Setiembre 2010

Agenda

1. Opencv.
2. Primer Ejemplo.
3. Aplicaciones y mas Funciones.
4. Segundo Ejemplo

¿Que es Opencv?

- **OpenCV** es una biblioteca open source para C/C++ para procesamiento de imágenes y visión computarizada, desarrollada inicialmente por Intel.
- Su primer versión estable fue liberada en 2006.
- En Octubre de 2009, se liberó el segundo release mayor: OpenCV v2
- http://opencv.willowgarage.com/wiki/

Generalidades

- Disponible en Linux, Mac, y Windows
- Tiene estructuras básicas de datos para operaciones con matrices y procesamiento de imágenes.
- Permite visualizar datos muy sencillamente y extraer información de imágenes y videos.
- Tiene funciones de captura y presentación de imágenes.

Opencv se compone de 4 Módulos

- **cv**
  Contiene las Funciones principales de la biblioteca
- **cvaux**
  Contiene las Funciones Auxiliares (experimental)
- **cxcore**
  Contiene las Estructuras de Datos y Funciones de soporte para Álgebra lineal
- **Highgui**
  Funciones para manejo de la GUI

Nombres de funciones y datos

- Convenciones para los nombres de las Funciones
  `cvActionTargetMod (...)`
  Action: Función core. Ej: set, create.
  Target: Elemento destino de la Acción. Ej: Contorno, polígono.
  Mod : Modificadores opcionales. Ej: Tipo de argumento.
- Matrix data types
  `CV_<bit_depth>(S|U|F)C<número de canales>`
  S: Entero Signado, **U**: Entero no Signado, **F**: Float
  Ej: `CV_8UC1` : matriz de un canal de 8 bits no signados,
  `CV_32FC2`: matriz de dos canales de 32 bits punto flotante.

Parámetros de imágenes y headers

- Tipos de datos de imágenes

  `IPL_DEPTH_<bit_depth>(S|U|F)`

  Ej: `IPL_DEPTH_8U` : imagen de 8 bits no signados.

  `IPL_DEPTH_32F`: imagen de 32 bits punto flotante.

  Headers

```
#include <cv.h>
#include <cvaux.h>
#include <highgui.h>
#include <cxcore.h> // innecesario, incluido en cv.h
```

1. Abrir eje1.c
2. Para compilar...
3. `gcc -oeje1 eje1.c -g -ggdb 'pkg-config --cflags --libs opencv' -Wall`

¿Que hicimos?

- Carga de una imagen

```
IplImage *image = cvLoadImage("Lena.bmp");
```

- Crear y Ubicar una ventana

```
cvNamedWindow ("ejemplo1", CV_WINDOW_AUTOSIZE);
cvMoveWindow ("ejemplo1", 100, 100);
 // desde borde superior izquierdo
```

- Mostrar la imagen en la ventana creada

```
cvShowImage("ejemplo1", image);
```

- Liberar recursos

```
cvReleaseImage(&image);
```

Prefijo cv

- Carga de una imagen

```
IplImage *image = cvLoadImage("Lena.bmp");
```

- Crear y Ubicar una ventana

```
cvNamedWindow ("ejemplo1", CV_WINDOW_AUTOSIZE);
cvMoveWindow ("ejemplo1", 100, 100);//desde
borde superior izquierdo
```

- Mostrar la imagen en la ventana creada

```
cvShowImage("ejemplo1", image);
```

- Liberar recursos

```
cvReleaseImage(&image);
```

Acción

- Carga de una imagen

```
IplImage *image = cvLoadImage("Lena.bmp");
```

- Crear y Ubicar una ventana

```
cvNamedWindow ("ejemplo1", CV_WINDOW_AUTOSIZE);
cvMoveWindow ("ejemplo1", 100, 100);//desde
borde superior izquierdo
```

- Mostrar la imagen en la ventana creada

```
cvShowImage("ejemplo1", image);
```

- Liberar recursos

```
cvReleaseImage(&image);
```

Destino

- Carga de una imagen

```
IplImage *image = cvLoadImage("Lena.bmp");
```

- Crear y Ubicar una ventana

```
cvNamedWindow("ejemplo1", CV_WINDOW_AUTOSIZE);
cvMoveWindow("ejemplo1", 100, 100);//desde
borde superior izquierdo
```

- Mostrar la imagen en la ventana creada

```
cvShowImage("ejemplo1", image);
```

- Liberar recursos

```
cvReleaseImage(&image);
```

IplImage: "La" Estructura

```
typedef struct _IplImage
{
  int nSize;
  int ID;
  int nChannels;
  int alphaChannel;
  int depth;
  char colorModel[4];
  char channelSeq[4];
  int dataOrder;
  int origin;
  int align;
  int width;
  int height;
  struct _IplROI *roi;
  struct _IplImage
*maskROI;
  void *imageId;
  struct _IplTileInfo
*tileInfo;
  int imageSize;
  char *imageData;
  int widthStep;
  int BorderMode[4];
  int BorderConst[4];
  char *imageDataOrigin;
} IplImage;
```

IplImage: "**La**" Estructura: Contenido

- **Nsize:** sizeof (IplImage)
- **ID:** Versión, siempre igual a 0
- **nchannels:** Número de canales. La mayoría de las funciones **OpenCV** soportan 1 a 4 canales.
- **alphaChannel:** Ignorado por **OpenCV**

IplImage: "**La**" Estructura: Contenido

- **depth:** Profundidad del canal en bits + el bit de signo opcional (`IPL_DEPTH_SIGN`).
  - **`IPL_DEPTH_8U`:** entero no signado de 8 bits.
  - **`IPL_DEPTH_8S`:** entero signado de 8 bits.
  - **`IPL_DEPTH_16U`:** entero no signado de 16 bits.
  - **`IPL_DEPTH_16S`:** entero signado de 16 bits.
  - **`IPL_DEPTH_32S`:** entero signado de 32 bits.
  - **`IPL_DEPTH_32F`:** Punto flotante simple precisión.
  - **`IPL_DEPTH_64F`:** Punto flotante doble precisión.

IplImage: "La" Estructura: Contenido

- **colorModel:** Ignorado por **OpenCV**. La función **CvtColor** de **OpenCV** requiere los espacios de color origen y destino como parámetros.
- **channelSeq:** Ignorado por **OpenCV**.
- **dataOrder:**
  - 0: `IPL_DATA_ORDER_PIXEL` - canales de color entrelazados.
  - 1: canales de color separados.
  - CreateImage solo crea imágenes con canales entrelazados. Por ejemplo, el layout común de colores de una imagen es: `b_00 g_00 r_00 b_10 g_10 r_10 ...`

IplImage: "**La**" Estructura: Contenido

- **origin:**
  - 0: origen extremo superior izquierdo.
  - 1: origen extremo inferior izquierdo, (estilo Windows bitmap).
- **align:** Alineación de las filas de la imagen(4 u 8). **OpenCV** ignora este campo usando en su lugar **widthStep**.
- **width:** Ancho de la Imagen en pixels.
- **height:** Alto de la Imagen en pixels.

IplImage: "La" Estructura: Contenido

- **roi:** Region Of Interest (ROI). Si no es `NULL`, se procesa solo esta región de la imagen.
- **maskROI:** Debe ser `NULL` en **OpenCV**.
- **imageId:** Debe ser `NULL` en **OpenCV**.
- **tileInfo:** Debe ser `NULL` en **OpenCV**.
- **imageSize:** Tamaño en bytes de la imagen. Para datos entrelazados, equivale a:
  `image->height * image->widthStep`

IplImage: "**La**" Estructura: Contenido

- **imageData:** Puntero a los datos alineados de la imagen.
- **widthStep:** Ta maño en bytes de una fila de la imagen alineada
- **BorderMode y BorderConst:** Modo de completamiento del borde, ignorado por **OpenCV**.
- **imageDataOrigin:** Puntero el origen de los datos de la imagen (no necesariamente alineados). Usado para desalojar la imagen.

IplImage

- La estructura **`IplImage`** se hereda de la Librería original de Intel.
- Formato nativo. **OpenCV** solo soporta un subset de formatos posibles de **`IplImage`**.
- Además de las restricciones anteriores, OpenCV maneja las ROIs de modo diferente. Las funciones de **OpenCV** requieren que los tamaños de las imágenes o los de las ROI de todas las imágenes fuente y destino coincidan exactamente.
- Por otra parte, la Biblioteca de Intel de Procesamiento de Imágenes procesa el área de intersección entre las imágenes origen y destino (o ROIs), permitiéndoles variar de forma independiente.

IplImage

- El tema es que cualquier imagen va a parar a una estructura de este tipo.
- **OpenCV** permite visualizar videos desde dos fuentes de información:
  - Cámara web conectada a la PC
  - Archivo avi.
- La imagen de video se compone de cuadros de n*m pixeles
- Cada cuadro se carga en una estructura **`IplImage`**

Funciones y procedimento

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

size: Tamaño en pixels del frame que va a contener la imagen:

```
typedef struct CvSize {
  int width;
  int height;
} CvSize;
cvSize (width,height); // es la función
inicializadora (Constructora)
```

depth: profundidad del pixel en bits: `IPL_DEPTH_8U`, `IPL_DEPTH_32F`.

channels:Número de canales por pixel. (1, 2, 3 o 4). Los canales están entrelazados. El layout de datos usual de una imagen color es `b0 g0 r0 b1 g1 r1`...

Ejemplos de creación de una imagen

Ejemplos:

```
// Crear una imagen con canal de 1 byte
IplImage* img1=cvCreateImage (cvSize(640,480),
IPL_DEPTH_8U,1);
// Crear una imagen con tres canal de float
IplImage* img2=cvCreateImage(cvSize(640,480),
IPL_DEPTH_32F,3);
```

Cerrar y Clonar

- Cerrar una imagen
  `cvReleaseImage(&img);`
- Clonar una imagen

```
IplImage*img1=cvCreateImage(cvSize
(640,480),IPL_DEPTH_8U,1);
IplImage* img2;
img2 = cvCloneImage (img1);
```

ROI Región Of Interest

- En la mayoría de las aplicaciones nos concentramos en cierta región de la pantalla, donde está la información que queremos procesar.
- Es como una submatriz de la matriz general.
- Setear u obtener la región de interés (ROI).
  - `void cvSetImageROI (IplImage* image, CvRect rect);`
  - `CvRect cvGetImageROI (const IplImage* image);`

Definiendo “cajas“ dentro de la Imagen

- **`CvRect`**: coordenadas de la esquina superior izquierda y el tamaño del rectángulo.

```
typedef struct CvRect {
  int x; //coordenada x de la esquina superior izquierda
  int y; //coordenada y de la esquina superior izquierda
  int width; //ancho del rectángulo
  int height; //alto del rectángulo
}
inline CvRect cvRect( int x, int y, int width, int height
); //inicialización
```

Estructuras asociadas

- Estructura **`CvScalar`**
- Es un contenedor de un arreglo de 1, 2, 3, o 4 doubles.
- Cada double pertenece al valor R G B y Alfa

```
typedef struct CvScalar {
  double val[4];
} CvScalar;
```

- En caso de imágenes monocromo contiene el valor en escala de gris en formato double.

Pixeles color y monocromáticos

- Inicializar **`val[0]`** con **`val0`**, **`val[1]`** con **`val1`**, etc.

```
inline CvScalar cvScalar(double val0, double val1=0,
double val2=0, double val3=0);
```

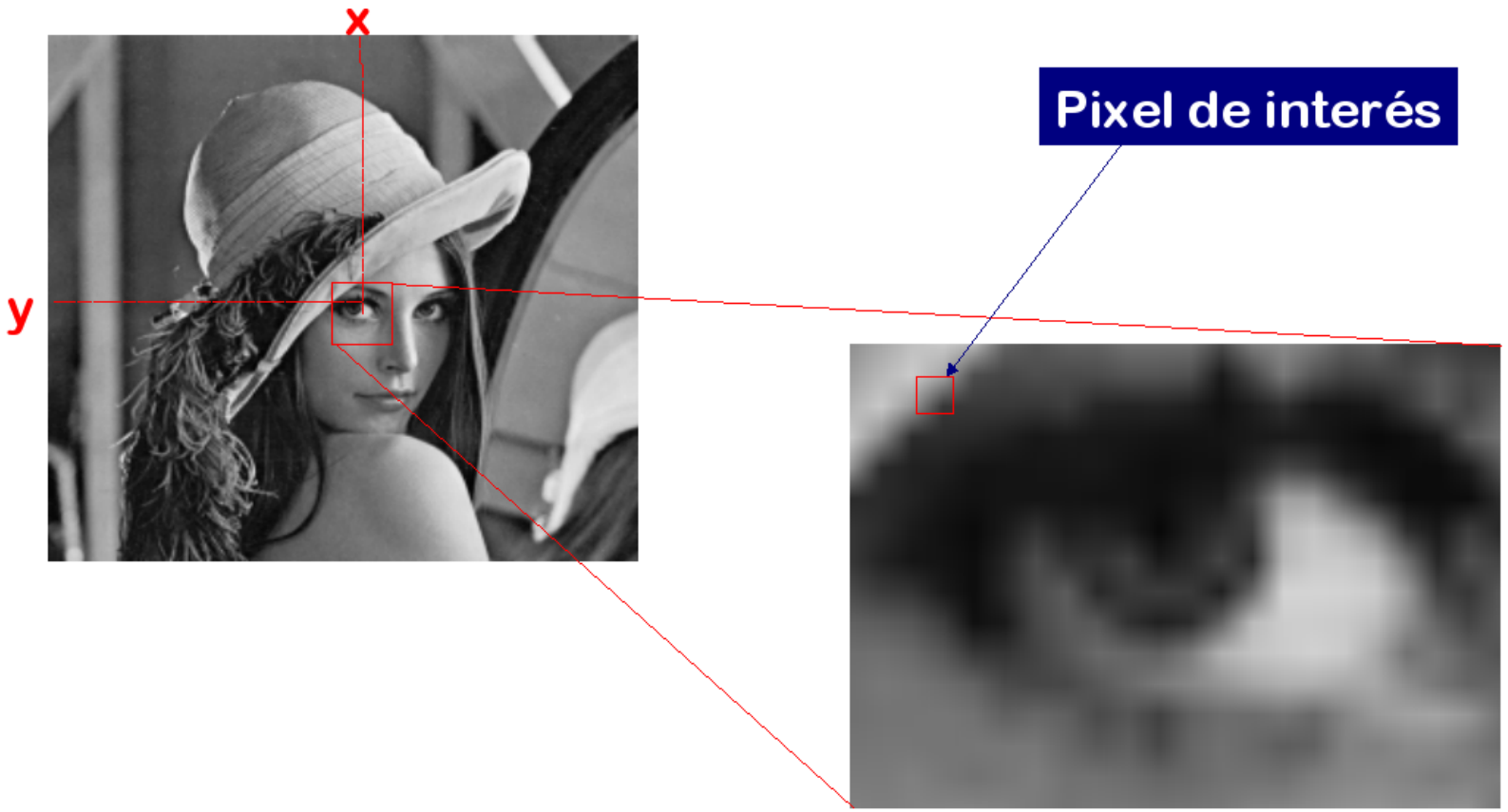
- Inicializar los cuatro elementos **`val[0]...val[3]`** con el valor **`val0123`**.

```
inline CvScalar cvScalarAll(double val0123);
```

- Inicializar **`val[0]`** con **`val0`**,y el resto (**`val[1]`**...**`val[3]`**) con ceros

```
inline CvScalar cvRealScalar(double val0);
```

Obteniendo el valor de un pixel



Obteniendo el valor de un pixel

```
CvScalar s = cvGet2D (img, row, col)
```

- Si la imagen está en escala de grises, **`s.val[0]`** es el valor del pixel.
- Si la imagen está en color, **`s.val[0]`**, **`s.val[1]`**, y **`s.val[2]`** son respectivamente R, G, y B.
- **`Img`** es un puntero a la **`IplImage`** obtenida al abrir o crear la imagen.
- **`row`** y **`col`** con **x** e **y** del slide anterior.

Fuente: archivo avi

1. Abrir aviexample.c
2. Para compilar...
3. `gcc -oavidemo aviexample.c -g -ggdb 'pkg-config --cflags --libs opencv' -Wall`

¿Que hicimos?

- Creamos una ventana llamada avidemo.
  `cvNamedWindow("avidemo", CV_WINDOW_AUTOSIZE);`
- Tomar un dspositivo de captura de Video.
  `CvCapture* capture = cvCreateFileCapture( argv[1] );`
- Crear un puntero a una estructura **`IplImage`** en donde se guardarán los frames.
  `IplImage* frame;`
- Luego entramos a un buclle infinito **`while(1)`**

¿Que hicimos?

- Se obtiene cada frame del avi mediante
  `frame = cvQueryFrame( capture );`
- Y lo mostramos (esto ya lo aprendimos)...
  `cvShowImage( "avidemo", frame );`
- Finaliza cuando el puntero al frame es NULL (encontró EOF).
  `if( !frame ) break;`
- Esperamos una tecla (Opencv tiene una función para esto también)
  `char c = cvWaitKey(33);`
- Liberamos recursos
  `cvReleaseCapture( &capture );`
  `cvDestroyWindow( "avidemo");`