Clase 1:
Las Librerías OpenCV

David Martín, Fernando García y José María Armingol

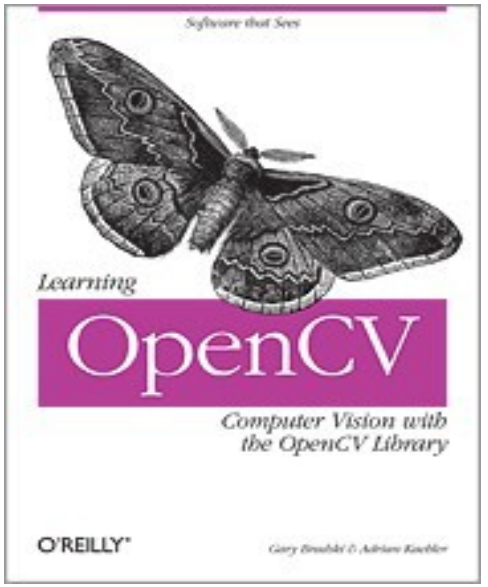
Curso 2015-2016

1.1:
Las librerías OpenCV

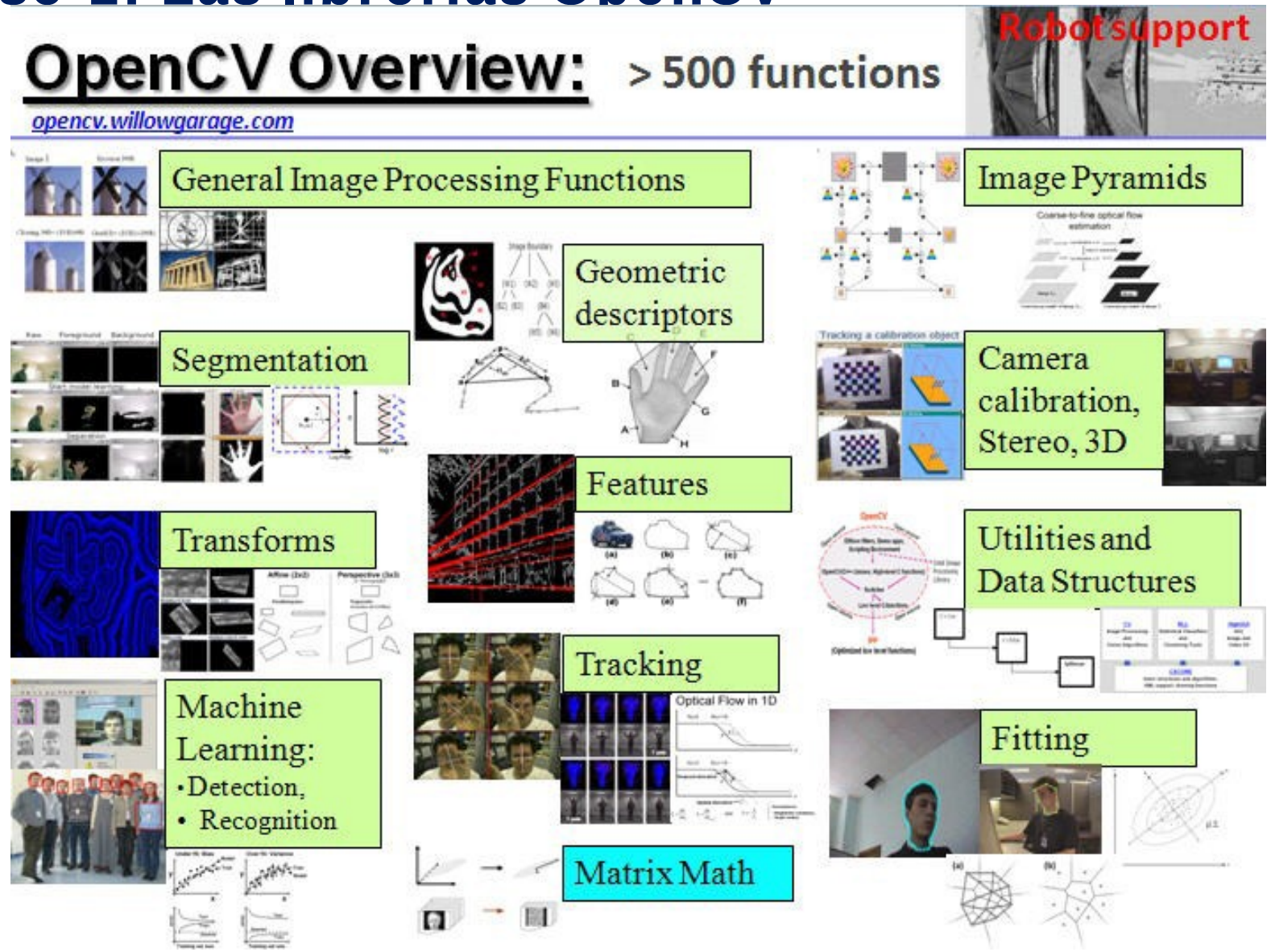
1.2:
OpenCV: Acceso a los píxeles, Histogramas

Clase 1: Las librerías OpenCV

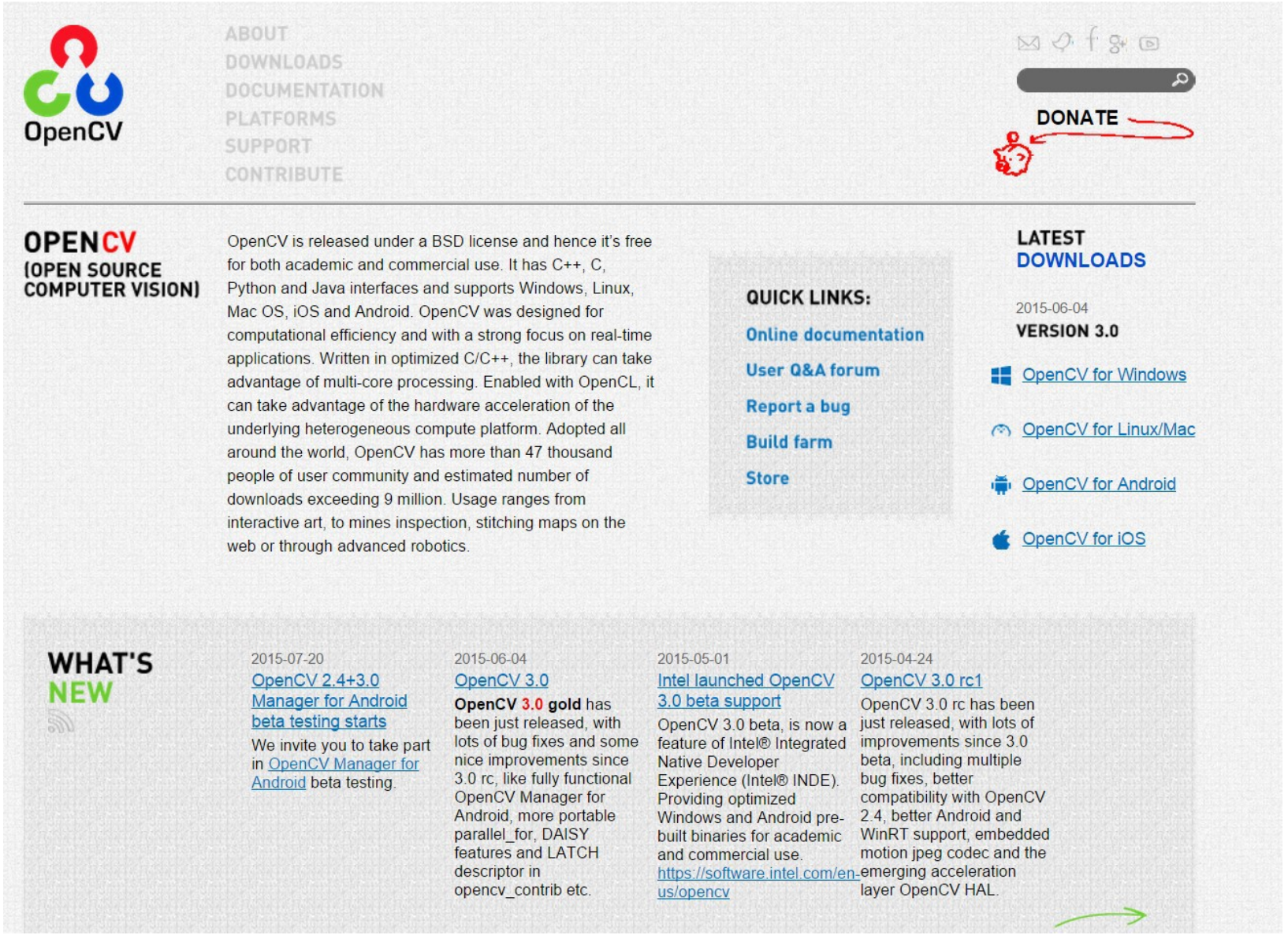
- ¿Qué son las librerías OpenCV?
  - Open Source Computer Vision
  - Librerías de visión por computador desarrolladas por Intel
  - 1999 versión alfa, 2015 versión 3.0
  - Licencia BSD . Pueden ser usadas para propósitos comerciales y de investigación.
  - Multiplataforma: Linux, MacOS X y Windows
  - 500 funciones (c, c++, python)
- sourceforge.net/projects/opencvlibrary
- Learning OpenCV. Computer Vision with the OpenCV Library. G. Bradski, A. Kaehler, O'Reilly Media, 2008



Clase 1: Las librerías OpenCV

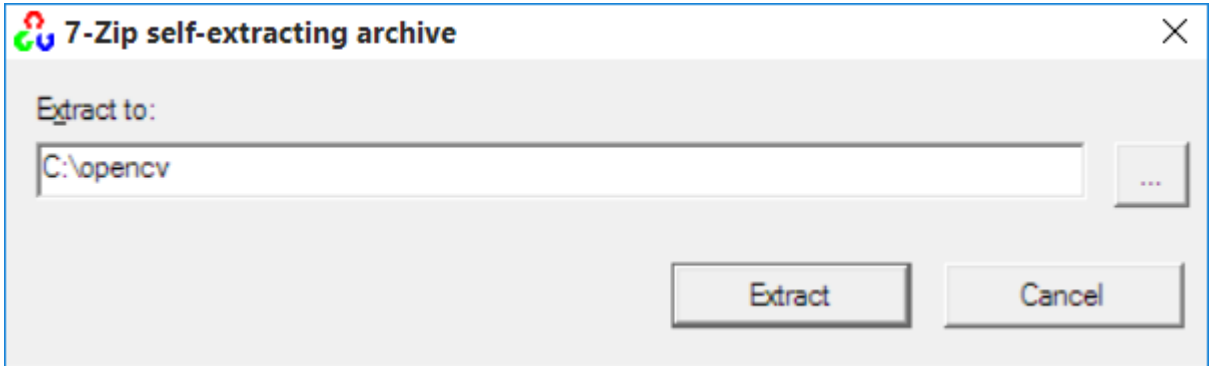


http://opencv.org



Clase 1: Las librerías OpenCV

- Instalación de las OpenCV
  - Descarga desde: sourceforge.net/projects/opencvlibrary
    - Versión 3.0.0 (Junio 2015)
  - Instalar en C:\opencv



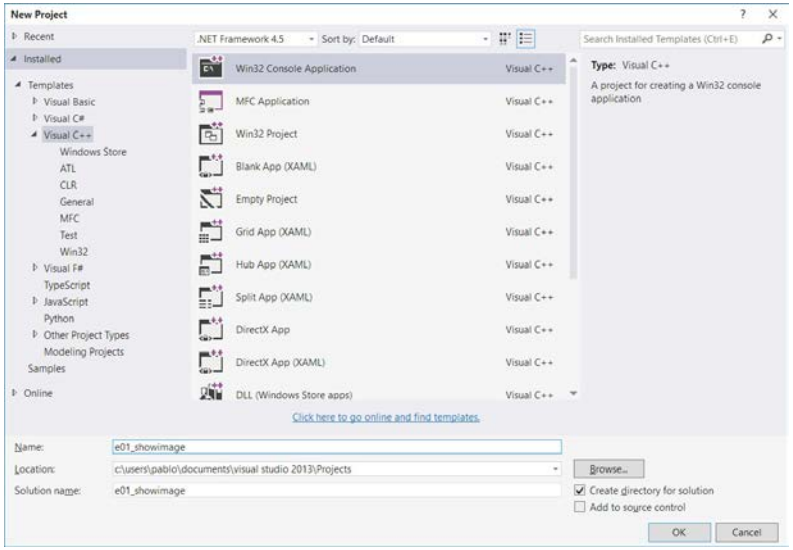
Clase 1: Las librerías OpenCV

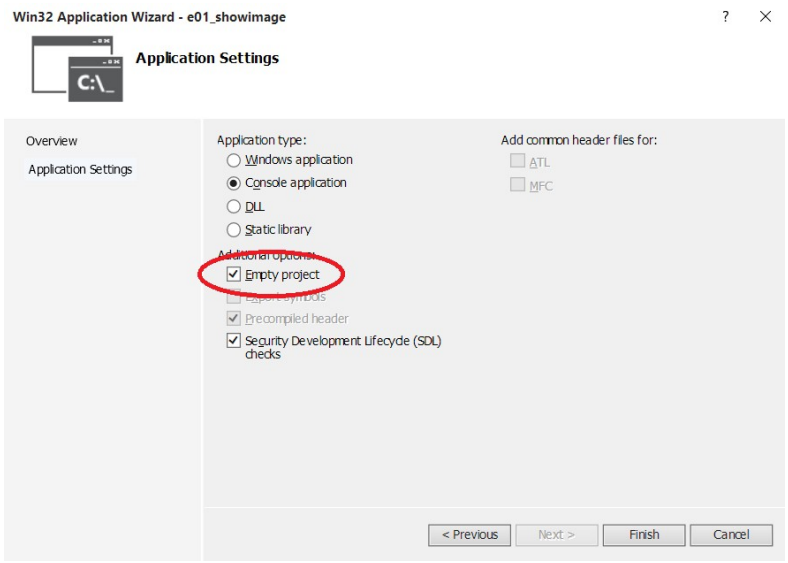
- En Microsoft Visual Studio (todas versiones)
  - Proyecto nuevo
  - Añadir en VisualC++ subdirectorios y la ruta de las librerías.
  - Seleccionar las bibliotecas del proyecto.

¿Cómo?

Clase 1: Las librerías OpenCV

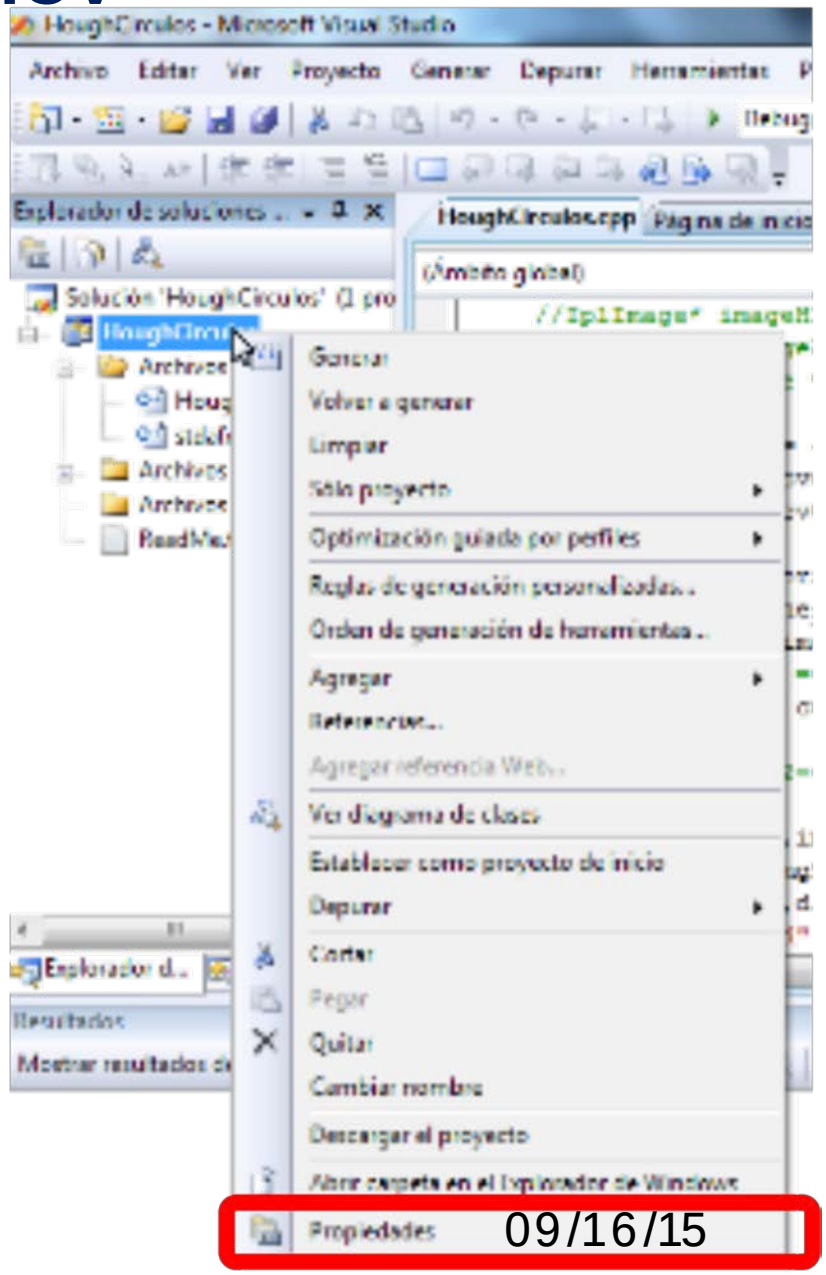
- Iniciar MS Visual C++ Archivo/Nuevo/Proyecto
- Seleccionar Tipo de proyecto:
  - Visual C++ -> Aplicación de consola Win32
- Nombre -> “L01_showimage”





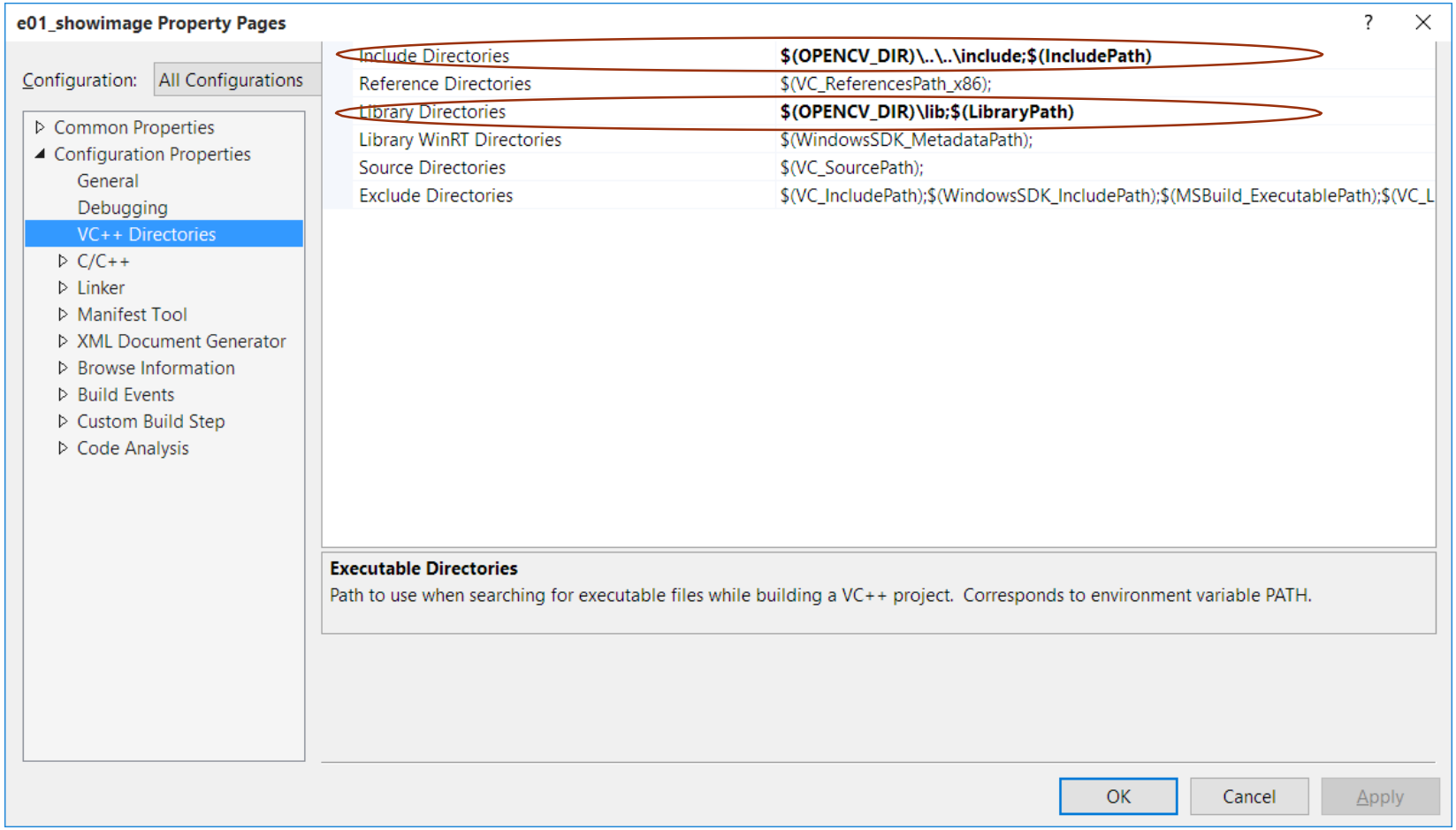
Clase 1: Las librerías OpenCV

- Incluir los subdirectorios y la ruta de las librerías
- Seleccionar el nombre del proyecto, pulsar botón derecho y pinchar en propiedades



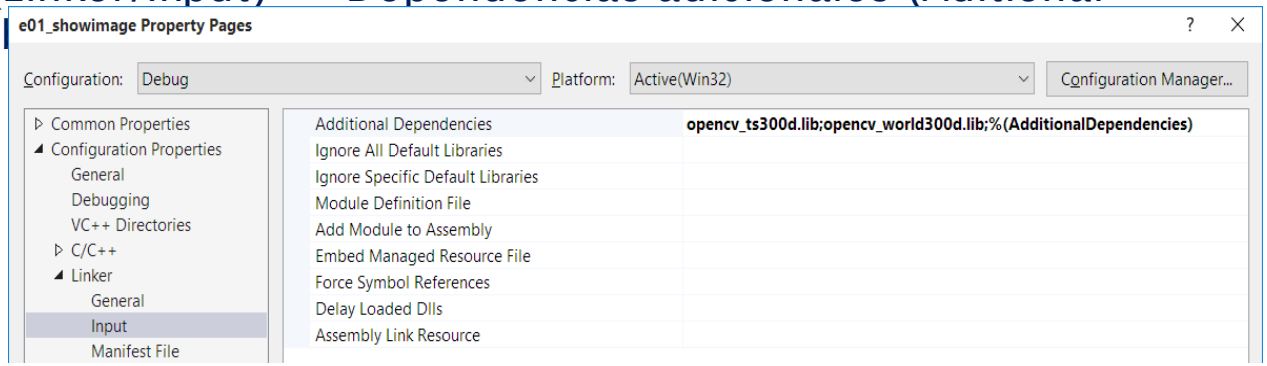
Clase 1: Las librerías OpenCV

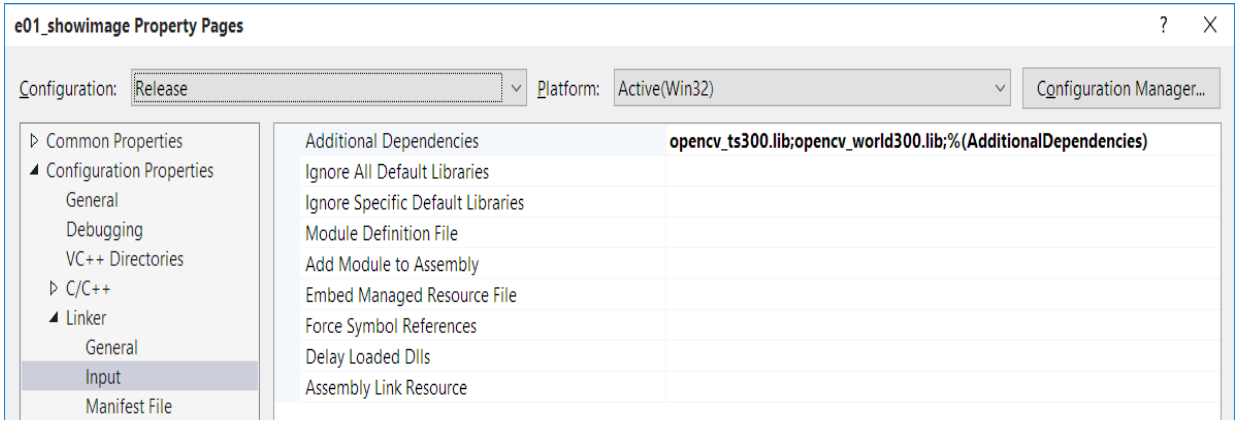
- Configuración: 'All configurations'
- Incluir la ruta:
  - $(OPENCV_DIR)\..\..\include
- Ruta de la librería:
  - $(OPENCV_DIR)\lib



Clase 1: Las librerías OpenCV

- Seleccionar las librerías de las opencv en el proyecto:
  - En Vinculador/entrada (Linker/Input) -> Dependencias adicionales (Aditional dependencies)
  - En modo 'debug':
    - opencv_ts300d.lib
    - opencv_world300d.li
  - En modo 'release':
    - opencv_ts300.lib
    - opencv_world300.lib



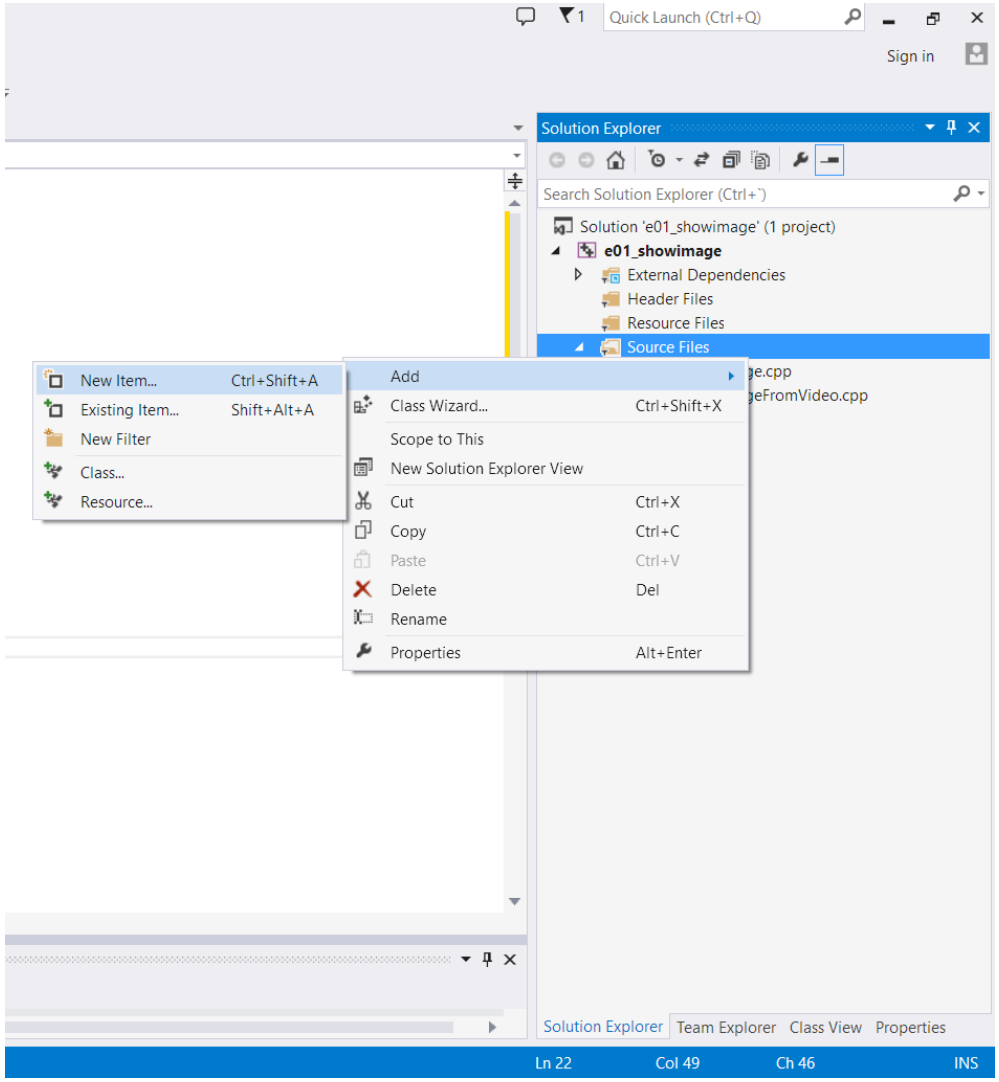


Clase 1: Las librerías OpenCV

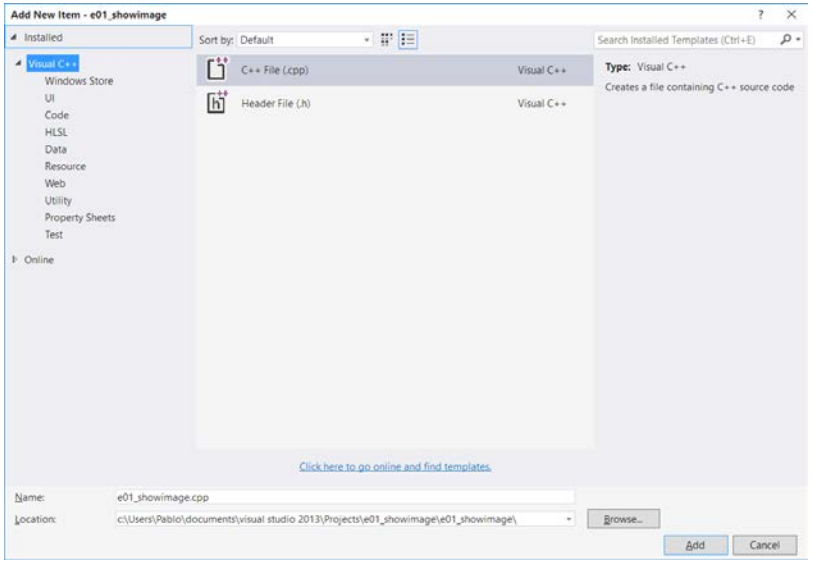
- Complete libraries:

opencv_ts300.lib
opencv_world300.lib
IlmImf.lib
ippicvmt.lib
libjasper.lib
libjpeg.lib
libpng.lib
libtiff.lib
libwebp.lib
opencv_calib3d300.lib
opencv_core300.lib
opencv_features2d300.lib
opencv_flann300.lib
opencv_hal300.lib
opencv_highgui300.lib
opencv_imgcodecs300.lib
opencv_imgproc300.lib
opencv_ml300.lib
opencv_objdetect300.lib
opencv_shape300.lib
opencv_photo300.lib
opencv_stitching300.lib
opencv_superres300.lib
opencv_videostab300.lib
opencv_video300.lib
zlib.lib

Clase 1: Las librerías OpenCV



- Se añaden los códigos fuente:



Clase 1: Las librerías OpenCV

- **Ejemplo 01:**
  - Mostrar una imagen de disco:
    - #include <opencv/cv.h>
    - 1. Nombre de la imagen en el disco
    - 2. Cargar la imagen y comprobar
    - 3. Mostrar la imagen
    - 4. Esperar a la pulsación de cualquier tecla
    - 5. Liberar la memoria

e01_showimage.cpp*

(Global Scope)

```
1  // e01_showimage.cpp: Load image from disk and show in window
2  //
3  #include "opencv\cv.hpp"
4  #include <iostream>
5
6  using namespace cv;
7  using namespace std;
8
9  int main(int argc, char* argv[])
10 {
11     // Object
12     Mat img;
13
14     // Load image from disk
15     img = imread("mandril.jpg");
16     if (!img.data){
17         cout << "error loading image" << endl;
18         return 1;
19     }
20
21     // Create window canvas to show image
22     namedWindow("original", CV_WINDOW_AUTOSIZE);
23
24     // Show image in the name of the window
25     imshow("original", img);
26
27     // Function for show the image in ms.
28     // 0 means wait until keyboard is pressed
29     waitKey(0);
30
31     // Free memory
32     destroyWindow("original");
33     // End of the program
34     return 0;
35 }
36
```

100 %

Clase 1: Las librerías OpenCV

- **Ejemplo 02: Mostrar una imagen perteneciente a un vídeo.**
  1. **Cargar el archivo del vídeo**
  2. **Comprobar que se ha cargado correctamente**
  3. **Extraer la primera imagen**
  4. **Comprobar que se ha cargado correctamente**
  5. **Mostrar la imagen**
  6. **Presionar una tecla**
  7. **Si ESCAPE, finalizamos el bucle**
  8. **Liberamos memoria**
  9. **Finalizamos el programa**

Clase 1: Las librerías OpenCV

- Ejemplo 02: Mostrar una imagen perteneciente a un vídeo.

e01_showimage - Microsoft Visual Studio

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANA

Local Windows Debugger | Debug

Server Explorer | Toolbox

e02_showimageFromVideo.cpp

(Global Scope)

```cpp
// e02_showimageFromVideo.cpp: Load image from video stream.

#include "opencv\cv.hpp"
#include <iostream>

#define ESCAPE 27

using namespace cv;
using namespace std;

int main(int argc, char* argv[])
{
    // Name of the video
    char* name = "honda-asimo.avi";

    // "Capture" image extraction
    VideoCapture capture;

    // Object instantiation
    Mat frame;

    // keyboard pressed
    char keypressed = 0;

    // Check the success for image reading
    bool success;

    // Load image from disk
    capture.open(name);
    // if not success, exit program
    if (!capture.isOpened()){
        cout << "error in VideoCapture: check path file" << endl;
        return 1;
    }

```

Server Explorer | Toolbox

e02_showimageFromVideo.cpp

(Global Scope)

```cpp
    }

    // Create window canvas to show image
    namedWindow("original", CV_WINDOW_AUTOSIZE);

    while (keypressed != ESCAPE){
        // read frame by frame in a loop
        success = capture.read(frame);

        // if no success exit program
        if (success == false){
            cout << "Cannot read the frame from file" << endl;
            return 1;
        }

        // Show image in window
        imshow("original", frame);

        // save the pres in keypressed
        keypressed = waitKey(0);
    }

    // Free memory
    destroyWindow("original");
    capture.release();
    // End of the program
    return 0;
}
```

91 %

Clase 1: Las librerías OpenCV

- **Ejemplo 03: Mostrar una imagen desde la cámara**
  1. **Abrir la cámara**
  2. **Comprobar que se ha cargado correctamente**
  3. **Obtenemos una imagen**
  4. **Comprobar la captura**
  5. **Mostrar la imagen**
  6. **Pulsar una tecla**
  7. **Si ESCAPE, finalizar el bucle**
  8. **Liberar memoria**
  9. **Finalizar programa**

Clase 1: Las librerías OpenCV

- **Ejemplo 03: Mostrar una imagen de una cámara**

e03_showimageFromCamera.cpp

(Global Scope)

```
1  // e02_showimageFromCamera.cpp: Load image from camera.
2
3  #include "opencv\cv.hpp"
4  #include <iostream>
5
6  #define ESCAPE 27
7
8  using namespace cv;
9  using namespace std;
10
11 int main(int argc, char* argv[])
12 {
13     // Start "Capture" in default device (0)
14     VideoCapture capture(0);
15     if (!capture.isOpened()){
16         cout << "error in VideoCapture: check device" << endl;
17         return 1;
18     }
19
20     // Object
21     Mat frame;
22
23     // keyboard pressed
24     char keypressed = 0;
25
26     // Check the success for image reading
27     bool success;
28
29     // Create window canvas to show image
30     namedWindow("original", CV_WINDOW_AUTOSIZE);
```

e03_showimageFromCamera.cpp

(Global Scope)

```
31
32     while (keypressed != ESCAPE){
33         // read frame by frame in a loop
34         success = capture.read(frame);
35         //capture >> frame;
36         // if no success exit program
37         if (success == false){
38             cout << "Cannot read the frame from file" << endl;
39             return 1;
40         }
41
42         // Show image in window
43         imshow("original", frame);
44
45         // save the pres in keypressed
46         keypressed = waitKey(0);
47     }
48
49     // Free memory
50     destroyWindow("original");
51     capture.release();
52     // End of the program
53     return 0;
54 }
```

1.1:
Las librerías OpenCV

1.2:
OpenCV: Acceso a los píxeles, Histogramas

1.2: OpenCV Acceso a los píxeles de una imagen y obtención de histogramas

Clase de problemas 2: OpenCV Acceso a los píxeles de una imagen

Estructuras en OpenCV:

OpenCV tiene varios tipos de estructuras de datos.

Los detalles de las estructuras se encuentran en el fichero: cxtypes.h

El tipo más sencillo es CvPoint, que es una estructura simple con dos variables enteras, X e Y, que son variables de tipo entero (int).

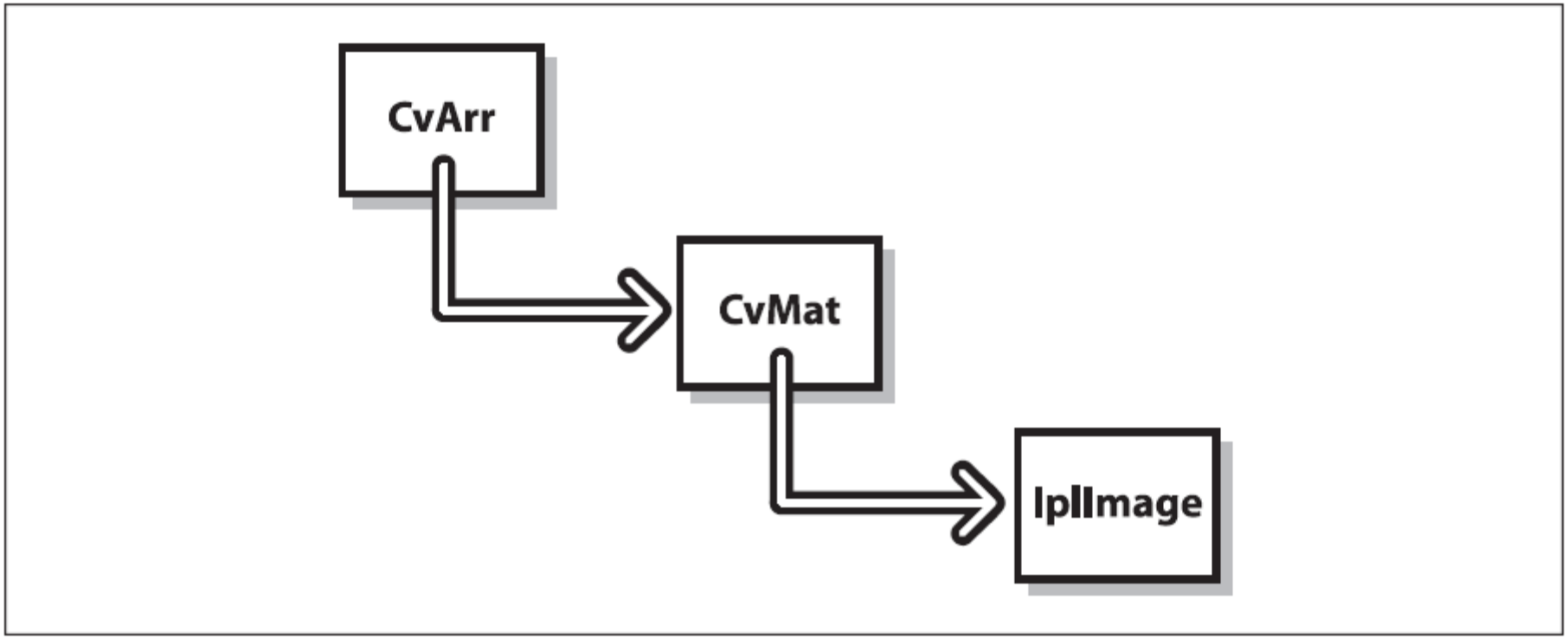
CvSize, CvRect, CvScalar, ….

Table 3-1. Structures for points, size, rectangles, and scalar tuples

Structure	Contains	Represents
CvPoint	int x, y	Point in image
CvPoint2D32f	float x, y	Points in $\Re^2$
CvPoint3D32f	float x, y, z	Points in $\Re^3$
CvSize	int width, height	Size of image
CvRect	int x, y, width, height	Portion of image
CvScalar	double val[4]	RGBA value

Clase de problemas 2: OpenCV Acceso a los píxeles de una imagen

Tipos matriz e imagen:



The IPLImage esta abandonado (OpenCV 1, en C). No es una clase en sí misma, sino una “estructura” muy compleja

OpenCV Acceso a los píxeles de una imagen

Estructura matriz e imagen:

```
typedef struct CvMat {
    int type;
    int step;
    int* refcount;    // for internal use only
    union {
        uchar*  ptr;
        short*  s;
        int*    i;
        float*  fl;
        double* db;
    } data;
    union {
        int rows;
        int height;
    };
    union {
        int cols;
        int width;
    };
} CvMat;
```

```
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;
```

OpenCV Acceso a los píxeles de una imagen

OpenCV nuevas clases (C++):

- OpenCV maneja la información a través de clases
- La definición de las clases está en el núcleo (core)
- 2D o array multidimensional denso (puede ser empleado para almacenar matrices, imagines, histogramas, descriptors de características, volúmenes…)
- El pixel tiene su propia definición de clase.

The OpenCV C++ reference manual is here:
*http://docs.opencv.org. Use **Quick Search** to find descriptions of the particular functions and classes*

Key OpenCV Classes

Point_	Template 2D point class
Point3_	Template 3D point class
Size_	Template size (width, height) class
Vec	Template short vector class
Matx	Template small matrix class
Scalar	4-element vector
Rect	Rectangle
Range	Integer value range
Mat	2D or multi-dimensional dense array (can be used to store matrices, images, histograms, feature descriptors, voxel volumes etc.)
SparseMat	Multi-dimensional sparse array
Ptr	Template smart pointer class

OpenCV Acceso a los píxeles de una imagen

Mat

OpenCV C++ n-dimensional dense array class

```
class CV_EXPORTS Mat
{
public:
    // ... a lot of methods ...
    ...

    /*! includes several bit-fields:
         - the magic signature
         - continuity flag
         - depth
         - number of channels
     */
    int flags;
    //! the array dimensionality, >= 2
    int dims;
    //! the number of rows and columns or (-1, -1) when the array has more than 2 dimensions
    int rows, cols;
    //! pointer to the data
    uchar* data;

    //! pointer to the reference counter;
    // when array points to user-allocated data, the pointer is NULL
    int* refcount;

    // other members
    ...
};
```

La clase Mat representa un array multidimensional denso y numérico puede ser “single-cannel” o “multi-channel”. Puede ser empelado para almacenar valores de vectores y matrices reales o complejos, incluso imágenes en escala de grises o color.

Clase de problemas 2: OpenCV Acceso a los píxeles de una imagen

Mat métodos:

Mat::Mat
Mat::~Mat
Mat::operator =
Mat::row
Mat::col
Mat::rowRange
Mat::colRange
Mat::diag
Mat::clone
Mat::copyTo
Mat::convertTo
Mat::assignTo
Mat::setTo
Mat::reshape
Mat::t
Mat::inv

Mat::mul
Mat::cross
Mat::dot
Mat::zeros
Mat::ones
Mat::eye
Mat::create
Mat::addref
Mat::release
Mat::resize
Mat::reserve
Mat::push_back
Mat::pop_back
Mat::locateROI
Mat::adjustROI
Mat::operator()

Mat::operator CvMat
Mat::operator IplImage
Mat::total
Mat::isContinuous
Mat::elemSize
Mat::elemSize1
Mat::type
Mat::depth
Mat::channels
Mat::step1
Mat::size
Mat::empty
Mat::ptr
Mat::at
Mat::begin
Mat::end

Clase de problemas 2: OpenCV Acceso a los píxeles de una imagen

Point

class **Point_**

```
template<typename _Tp> class CV_EXPORTS Point_
{
public:
    typedef _Tp value_type;

    // various constructors
    Point_();
    Point_(_Tp _x, _Tp _y);
    Point_(const Point_& pt);
    Point_(const CvPoint& pt);
    Point_(const CvPoint2D32f& pt);
    Point_(const Size_<_Tp>& sz);
    Point_(const Vec<_Tp, 2>& v);

    Point_& operator = (const Point_& pt);
    //! conversion to another data type
    template<typename _Tp2> operator Point_<_Tp2>() const;

    //! conversion to the old-style C structures
    operator CvPoint() const;
    operator CvPoint2D32f() const;
    operator Vec<_Tp, 2>() const;

    //! dot product
    _Tp dot(const Point_& pt) const;
    //! dot product computed in double-precision arithmetics
    double ddot(const Point_& pt) const;
    //! cross-product
    double cross(const Point_& pt) const;
    //! checks whether the point is inside the specified rectangle
    bool inside(const Rect_<_Tp>& r) const;

    _Tp x, y; //< the point coordinates
};
```

```
pt1 = pt2 + pt3;
pt1 = pt2 - pt3;
pt1 = pt2 * a;
pt1 = a * pt2;
pt1 += pt2;
pt1 -= pt2;
pt1 *= a;
double value = norm(pt); // L2 norm
pt1 == pt2;
pt1 != pt2;
```

For your convenience, the following type aliases are defined:

```
typedef Point_<int> Point2i;
typedef Point2i Point;
typedef Point_<float> Point2f;
typedef Point_<double> Point2d;
```

Example:

```
Point2f a(0.3f, 0.f), b(0.f, 0.4f);
Point pt = (a + b)*10.f;
cout << pt.x << ", " << pt.y << endl;
```

Clase de tipo 'Template' para puntos en 2D, especificado por sus coordenadas x e y. Una instancia de eta clase es intercambiable con estructuras de C (OpenCV 1 y 2) CvPoint y CvPoint2D32f . Hay también un operador "cast" para convertir coordenadas de puntos a un tupo específico. La conversión de coordenadas de coma flotante a entero se hace redondeando.

```
Point pt;
pt.x = 10;
pt.y = 8;

Point pt = Point(10, 8);
```

Clase de problemas 2: OpenCV Acceso a los píxeles de una imagen

Drawings

Circle:

```
void circle(
        InputOutputArray img,
        Point center,
        int radius,
        const Scalar& color,
        int thickness=1,
        intlineType=LINE_8,
        int shift=0 )
```

Scalar(blue, green, red);

-1 for filled circle

Opcional

```
Point center(img.rows / 2, img.cols / 2);
int radius = 25;
circle (img, center, radius, Scalar(0, 255, 0));
```

Clase de problemas 2: OpenCV Acceso a los píxeles de una imagen

Drawings

Line:

```
void line(
        InputOutputArray img,
        Point pt1,
        Point pt2,
        const Scalar& color,
        int thickness = 1,
        int lineType = LINE_8,
        int shift=0 )
```

Scalar(blue, green, red);

Opcional

```
Point start(img.rows / 2, img.cols / 2);
Point end (0, 0);
line (img, start, end, Scalar(0, 255, 0));
```

Clase de problemas 2: OpenCV Acceso a los píxeles de una imagen

Drawings

Rectangle:

```
void rectangle(
        InputOutputArray img,
        Point pt1,
        Point pt2,
        const Scalar& color,
        int thickness = 1,
        int lineType = LINE_8,
        int shift=0 )
```

Scalar(blue, green, red);

-1 for filled rectangle

Opcional

```
Point start(img.rows / 2, img.cols / 2);
Point end (img.rows , img.cols );
rectangle (img, start, end, Scalar(0, 255, 0));
```

Clase de problemas 2: OpenCV Acceso a los píxeles de una imagen

- Example 01: Drawings

L02_drawings.cpp

(Global Scope)

```cpp
// L02_drawings.cpp: Draw different shapes in the image
//
#include "opencv\cv.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main(int argc, char* argv[])
{
    // Object
    Mat img_original, img_modified;
    // Load image from disk
    img_original = imread("lena.jpg");

    // Copy images-> 2 ways to copy:
    //    img_original.clone()
    //    img_original.copyTo(img_modified)

    //img_modified = img_original.clone();
    img_original.copyTo(img_modified);

    if (!img_original.data){
        cout << "error loading image" << endl;
        return 1;
    }

    // Center of the image
    Point center(img_modified.rows / 2, img_modified.cols / 2);
    int radius = 25;

    // Draw a circle
    circle(img_modified, center, radius, Scalar(0, 255, 0));
```

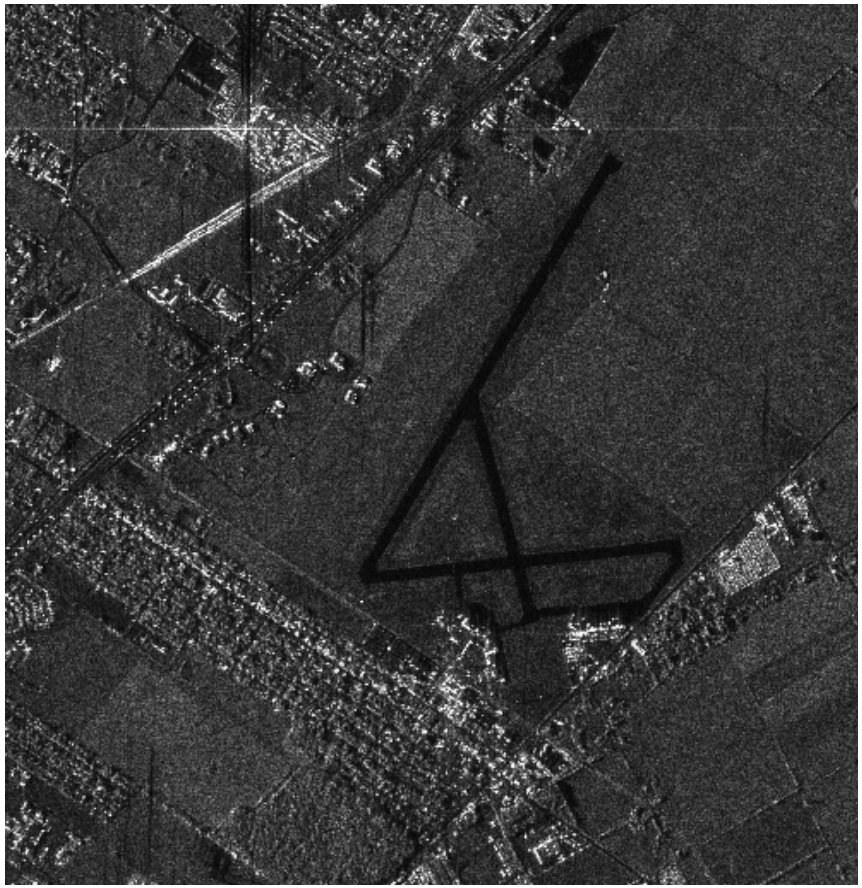
L02_drawings.cpp

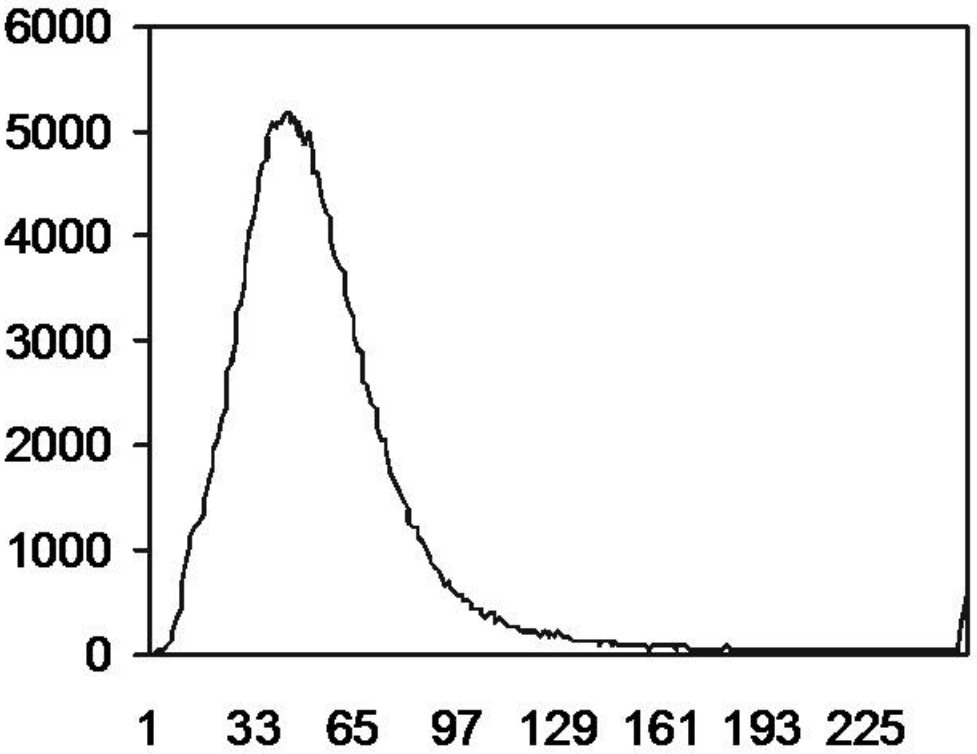
(Global Scope)

```cpp

    // Draw a line
    line(img_modified, center, Point(0,0), Scalar(255, 0, 0));

    // Draw a rectangle
    rectangle(img_modified, center, Point(img_modified.rows,
              img_modified.cols), Scalar(0, 0, 255));

    // Create window canvas to show image
    namedWindow("original", CV_WINDOW_AUTOSIZE);
    namedWindow("modified", CV_WINDOW_AUTOSIZE);

    // Show images
    imshow("original", img_original);
    imshow("modified", img_modified);

    // Function for show the image in ms.
    // 0 means wait until keyboard is pressed
    waitKey(0);

    // Free memory
    destroyWindow("original");
    destroyWindow("modified");
    // End of the program
    return 0;
}

```

Clase de problemas 2: OpenCV Histogramas

Histograma de la imagen





Clase de problemas 2: OpenCV Histogramas

```
void calcHist(
        const Mat* images,
        int nimages,
        const int* channels,
        InputArray mask,
        OutputArray hist,
        int dims,
        const int* histSize,
        const float** ranges,
        bool uniform=true,
        bool accumulate=false )
```

Opcional

- Example 02: Calculate histogram:
  - Load image (if bgr, must split first)
  - Configure histogram for each channel
  - Calculate histogram
  - Show each histogram channel
  - Free memory

Clase de problemas 2: OpenCV Histogramas

L02_histogram.cpp*

(Global Scope) — main(int, char

```cpp
// L02_histogram.cpp: Create histogram of one image
//
#include "opencv\cv.hpp"
#include <iostream>

using namespace cv;
using namespace std;

int main(int, char**)
{
    // Object
    Mat img_original;

    // Load image from disk
    img_original = imread("mandril.jpg", 0);
    if (!img_original.data){
        cout << "error loading image" << endl;
        return 1;
    }

    // Create window canvas to show image
    namedWindow("original", CV_WINDOW_AUTOSIZE);
    namedWindow("histogram", CV_WINDOW_AUTOSIZE);

    // Initialize parameters
    int histSize = 256;    // bin size
    float range[] = { 0, 255 };
    const float *ranges[] = { range };

    // Calculate histogram
    Mat hist;
    calcHist(&img_original, 1, 0, Mat(), hist, 1, &histSize, ranges, true, false);

```

Clase de problemas 2: OpenCV Histogramas

L02_histogram.cpp

(Global Scope) | main(int, char **)

```
33
34      // Show the calculated histogram in command window
35      double total;
36      total = img_original.rows * img_original.cols;
37      for (int h = 0; h < histSize; h++)
38      {
39          float binVal = hist.at<float>(h);
40          cout << " " << binVal;
41      }
42
43      // Plot the histogram
44      int hist_w = 512; int hist_h = 400;
45      int bin_w = cvRound((double)hist_w / histSize);
46
47      Mat histImage(hist_h, hist_w, CV_8UC1, Scalar(0, 0, 0));
48
49      // In order to fit in window sized 400x500 must be normalized:
50      // max value of same color pixel = 2740 --> 500
51      normalize(hist, hist, 0, histImage.rows, NORM_MINMAX, -1, Mat());
52
53      for (int i = 1; i < histSize; i++)
54      {
55          line(histImage, Point(bin_w*(i - 1), hist_h - cvRound(hist.at<float>(i - 1))),
56              Point(bin_w*(i), hist_h - cvRound(hist.at<float>(i))),
57              Scalar(255, 0, 0), 2, 8, 0);
58      }
59
60      imshow("original", img_original);
61      imshow("histogram", histImage);
62
63      waitKey(0);
64
65      destroyAllWindows();
66      return 0;
67  }
```