

Nota: Algunas de las imágenes que aparecen en esta presentación provienen del libro:  
Visión por Computador: fundamentos y métodos.  
Arturo de la Escalera Hueso. Prentice Hall.

# Sistemas de Percepción

## Visión por Computador

Arturo de la Escalera  
José María Armingol  
Fernando García  
David Martín  
Abdulla Al-Kaff



# Las Librerías OpenCV

# 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](http://sourceforge.net/projects/opencvlibrary)
- Learning OpenCV. Computer Vision with the OpenCV Library. G. Bradski, A. Kaehler, O'Reilly Media, 2008

- <http://opencv.org>



**OpenCV**  
(OPEN SOURCE COMPUTER VISION)

OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 9 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

**WHAT'S NEW**

Date	News Item
2015-07-20	<a href="#">OpenCV 2.4+3.0 Manager for Android beta testing starts</a> We invite you to take part in <a href="#">OpenCV Manager for Android</a> beta testing.
2015-06-04	<a href="#">OpenCV 3.0</a> <b>OpenCV 3.0 gold</b> has been just released, with lots of bug fixes and some nice improvements since 3.0 rc, like fully functional OpenCV Manager for Android, more portable parallel_for, DAISY features and LATCH descriptor in opencv_contrib etc.
2015-05-01	<a href="#">Intel launched OpenCV 3.0 beta support</a> OpenCV 3.0 beta, is now a feature of Intel® Integrated Native Developer Experience (Intel® INDE). Providing optimized Windows and Android pre-built binaries for academic and commercial use. <a href="https://software.intel.com/en-us/opencv">https://software.intel.com/en-us/opencv</a>
2015-04-24	<a href="#">OpenCV 3.0 rc1</a> OpenCV 3.0 rc has been just released, with lots of improvements since 3.0 beta, including multiple bug fixes, better compatibility with OpenCV 2.4, better Android and WinRT support, embedded motion jpeg codec and the emerging acceleration layer OpenCV HAL.

**QUICK LINKS:**

- [Online documentation](#)
- [User Q&A forum](#)
- [Report a bug](#)
- [Build farm](#)
- [Store](#)

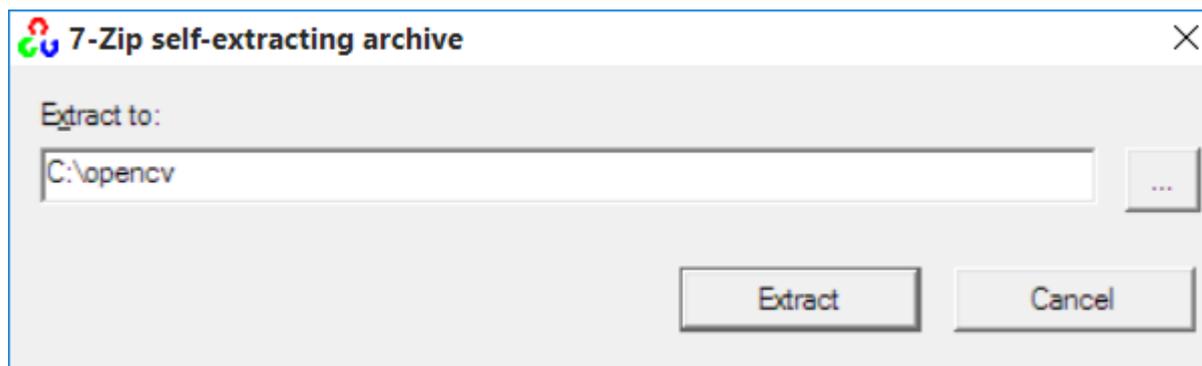
**LATEST DOWNLOADS**

2015-06-04  
**VERSION 3.0**

- [OpenCV for Windows](#)
- [OpenCV for Linux/Mac](#)
- [OpenCV for Android](#)
- [OpenCV for iOS](#)

# Las librerías OpenCV

- Instalación de las OpenCV
  - Descarga desde:  
[sourceforge.net/projects/opencvlibrary](http://sourceforge.net/projects/opencvlibrary)
- Instalar en
  - C:\opencv



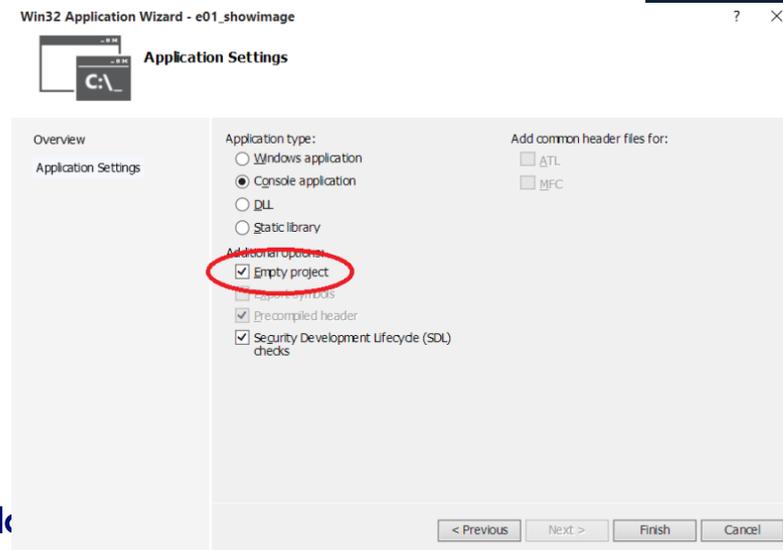
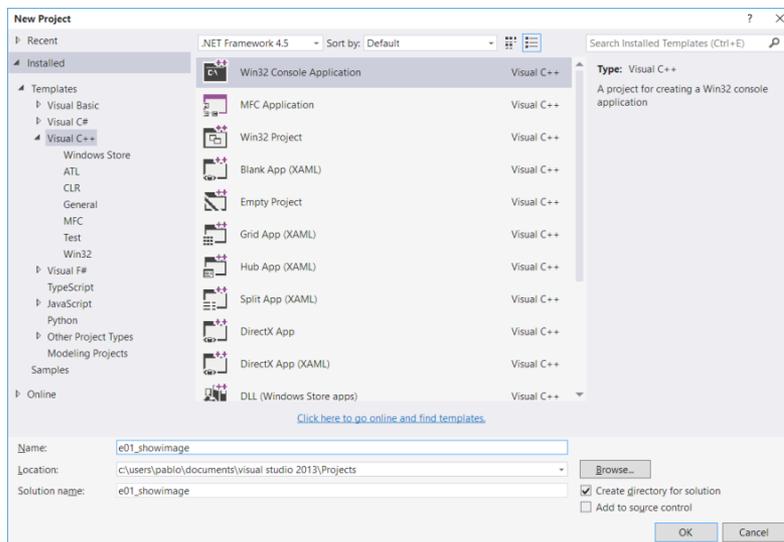
# Las librerías OpenCV

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

¿Cómo?

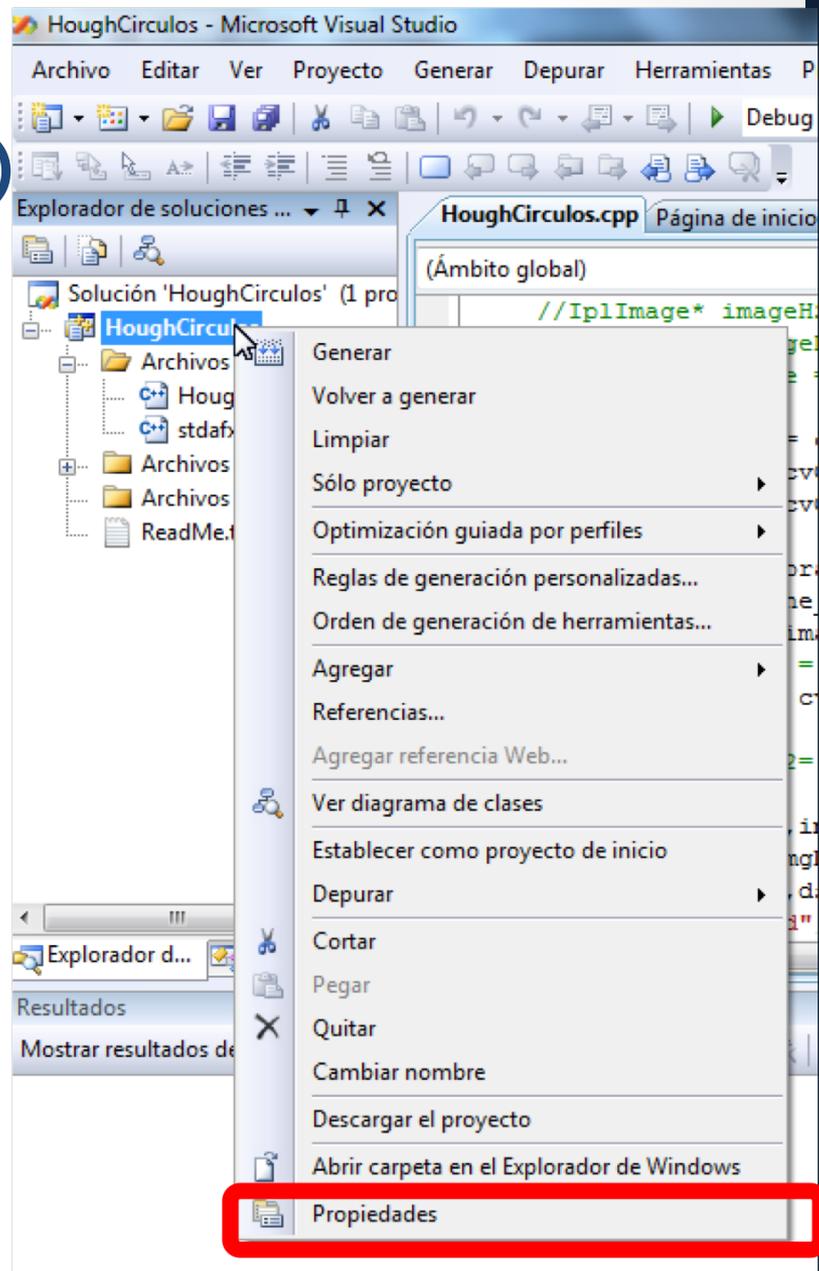
# Las librerías OpenCV

- Iniciar MS Visual C++  
Archivo/Nuevo/Proyecto
- Seleccionar Tipo de proyecto:
  - Visual C++ -> Aplicación de consola Win32
- Nombre -> “L01\_showimage”



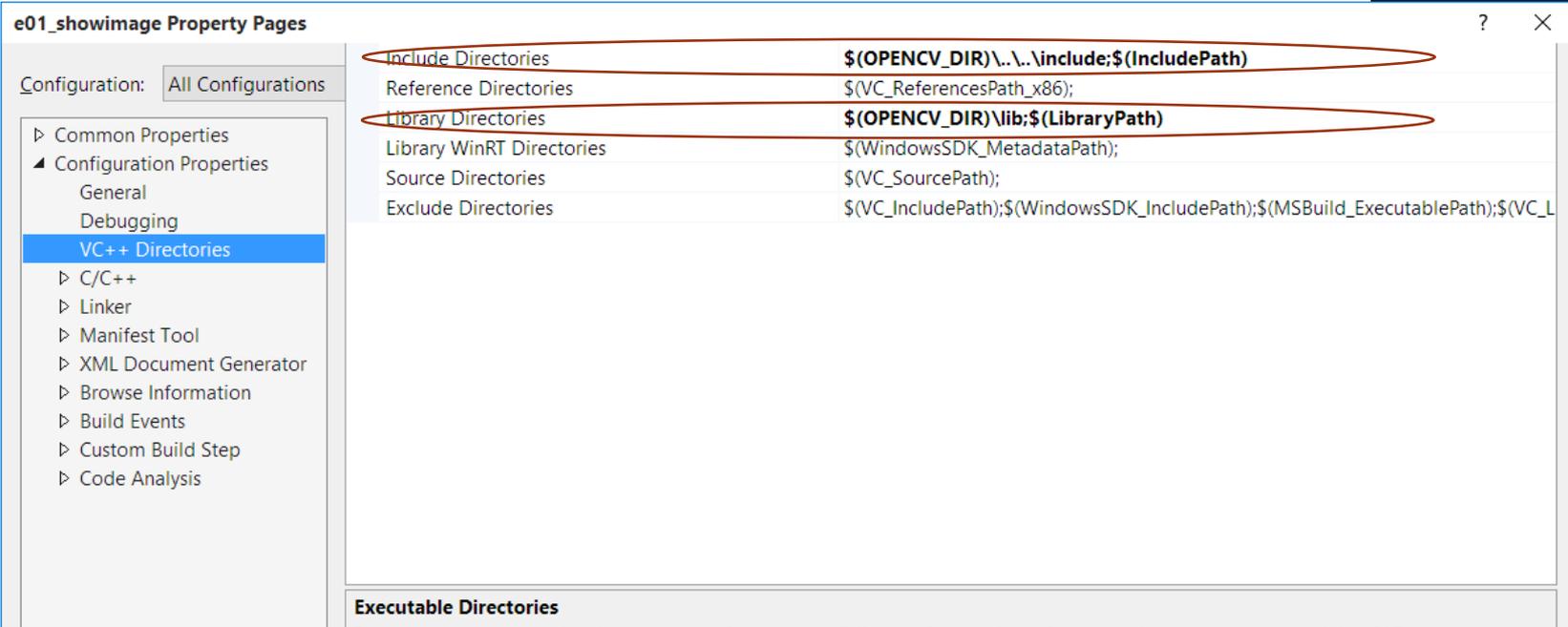
# Las librerías

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



# Las librerías OpenCV

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



**e01\_showimage Property Pages**

Configuration: All Configurations

- ▶ Common Properties
- ▲ Configuration Properties
  - General
  - Debugging
  - VC++ Directories**
  - ▶ C/C++
  - ▶ Linker
  - ▶ Manifest Tool
  - ▶ XML Document Generator
  - ▶ Browse Information
  - ▶ Build Events
  - ▶ Custom Build Step
  - ▶ Code Analysis

Include Directories	<code>\$(OPENCV_DIR)\..\..\include;\$(IncludePath)</code>
Reference Directories	<code>\$(VC_ReferencesPath_x86);</code>
Library Directories	<code>\$(OPENCV_DIR)\lib;\$(LibraryPath)</code>
Library WinRT Directories	<code>\$(WindowsSDK_MetadataPath);</code>
Source Directories	<code>\$(VC_SourcePath);</code>
Exclude Directories	<code>\$(VC_IncludePath);\$(WindowsSDK_IncludePath);\$(MSBuild_ExecutablePath);\$(VC_L</code>

**Executable Directories**

Path to use when searching for executable files while building a VC++ project. Corresponds to environment variable PATH.

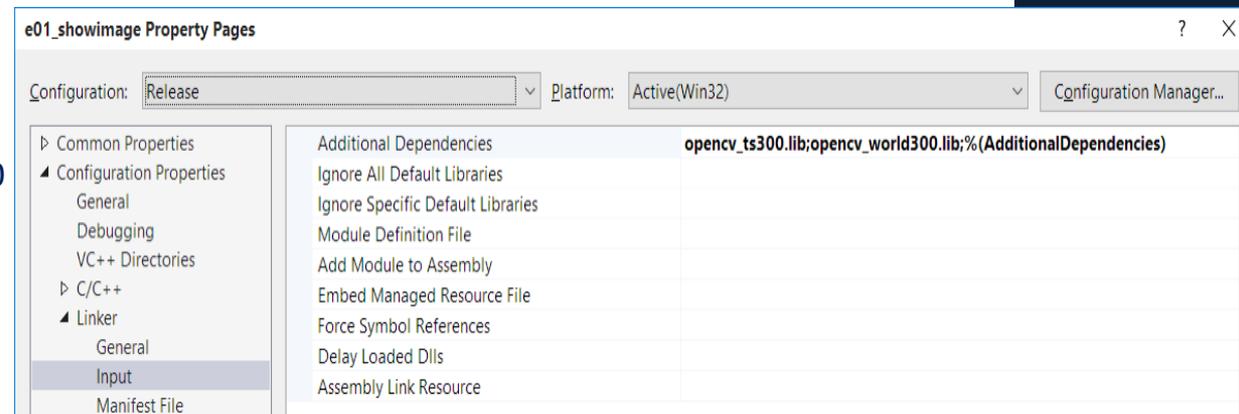
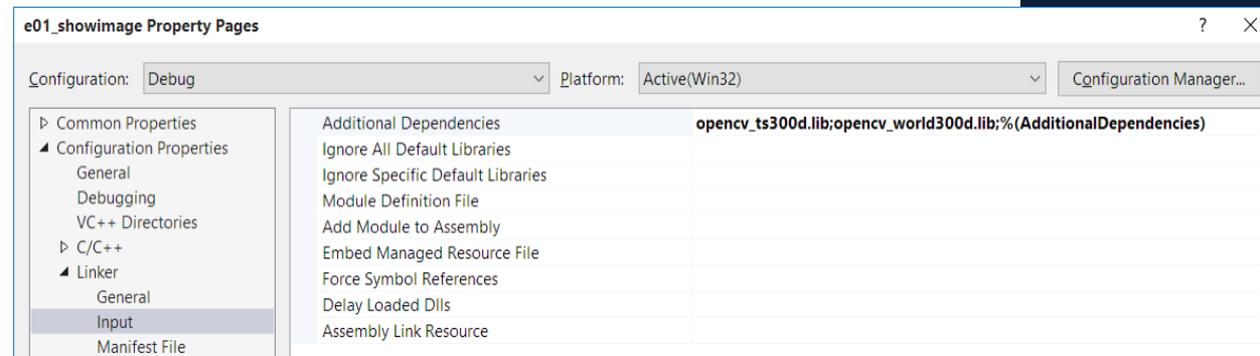
# Las librerías OpenCV

- Seleccionar las librerías de las opencv en el proyecto:
- En Vinculador/entrada (Linker/Input) -> Dependencias adicionales (Additional dependencies)

- En modo 'debug':
- opencv\_ts300d.lib
- opencv\_world300d.lib

- En modo 'release':
- opencv\_ts300.lib
- opencv\_world300.lib

- NOTA:
- 300 indica que es la versión 3.0
- Dependiendo de la que sea cambiará



# Las librerías OpenCV

- Complete libraries:

opencv\_ts300.lib

opencv\_world300.lib

l1m1mf.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

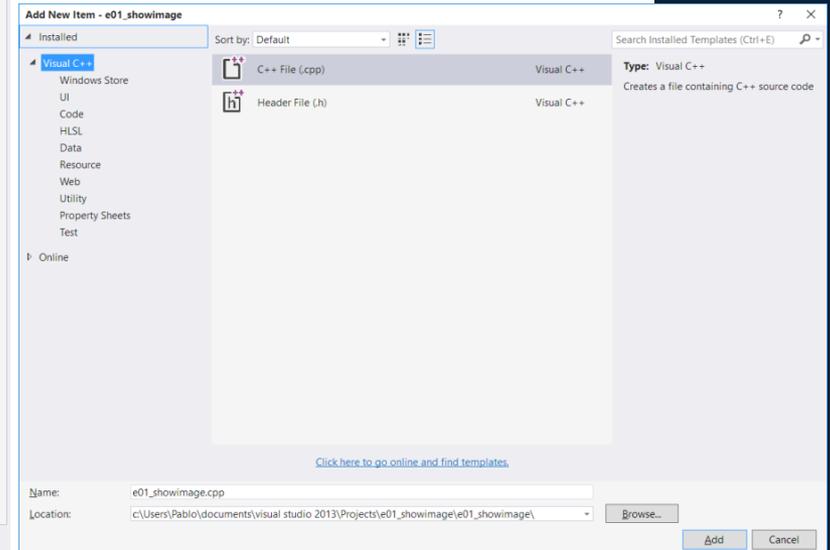
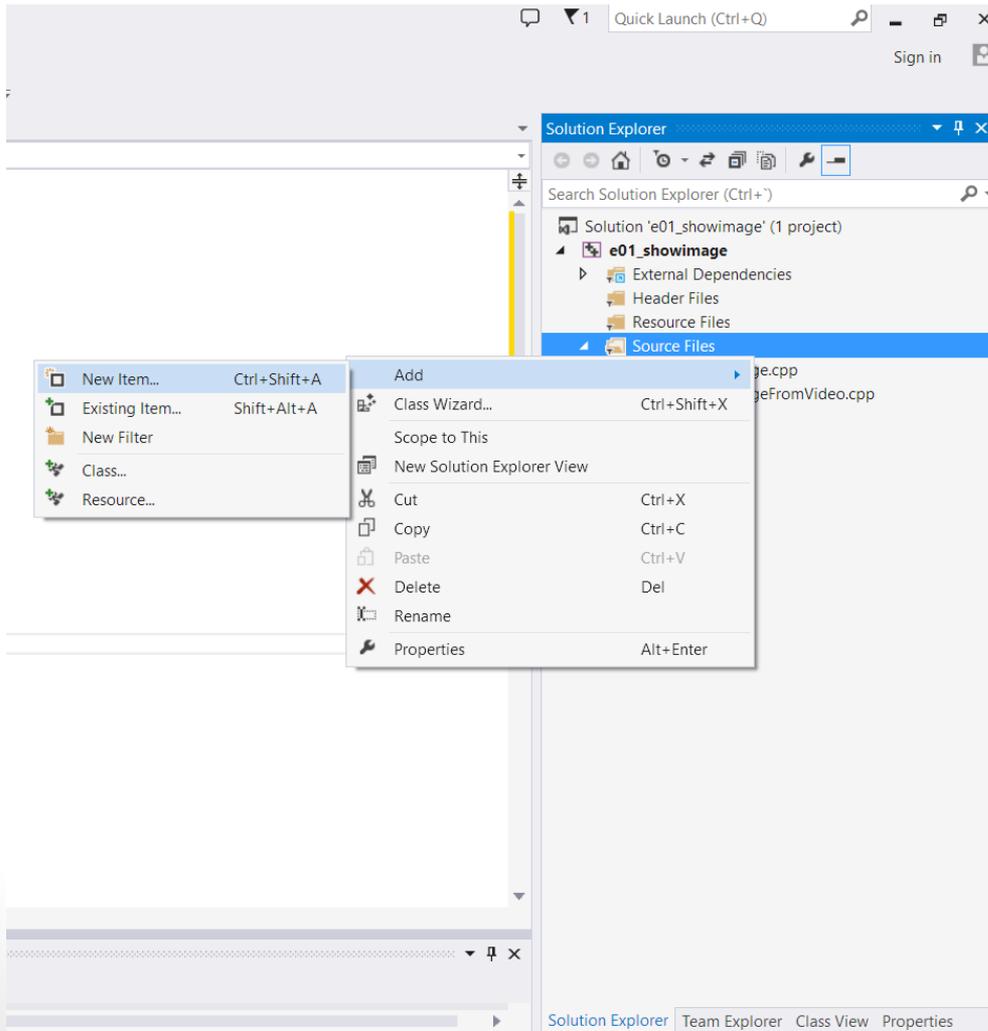
opencv\_videostab300.lib

opencv\_video300.lib

zlib.lib

# Las librerías OpenCV

- Se añaden los códigos fuente:



# 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* X
(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 instantiation
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
```

# 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

# Las librerías OpenCV

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

```
e02_showimageFromVideo.cpp
(Global Scope)
1 // e02_showimageFromVideo.cpp: Load image from video stream.
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     // Name of the video
14     char* name = "honda-asimo.avi";
15
16     // "Capture" image extraction
17     VideoCapture capture;
18
19     // Object instantiation
20     Mat frame;
21
22     // keyboard pressed
23     char keypressed = 0;
24
25     // Check the success for image reading
26     bool success;
27
28     // Load image from disk
29     capture.open(name);
30     // if not success, exit program
31     if (!capture.isOpened()){
32         cout << "error in VideoCapture: check path file" << endl;
33         return 1;
34     }
35
```

```
e02_showimageFromVideo.cpp
(Global Scope)
34 }
35
36 // Create window canvas to show image
37 namedWindow("original", CV_WINDOW_AUTOSIZE);
38
39 while (keypressed != ESCAPE){
40     // read frame by frame in a loop
41     success = capture.read(frame);
42
43     // if no success exit program
44     if (success == false){
45         cout << "Cannot read the frame from file" << endl;
46         return 1;
47     }
48
49     // Show image in window
50     imshow("original", frame);
51
52     // save the pres in keypressed
53     keypressed = waitKey(0);
54 }
55
56 // Free memory
57 destroyWindow("original");
58 capture.release();
59 // End of the program
60 return 0;
61 }
```

91 %

# 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

# Las librerías OpenCV

- Ejemplo 03: Mostrar una imagen de una cámara

```
e03_showimageFromCamera.cpp [X]
(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
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 instantiation
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 [X]
(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 }
```