Visión Artificial

Práctica 1: La biblioteca OpenCV

Laboratorio de Sistemas Inteligentes

Universidad Carlos III de Madrid

Índice

- La librería OpenCV
- Instalación
- Ejecución del primer programa
- Capturar imágenes desde un vídeo o desde una cámara

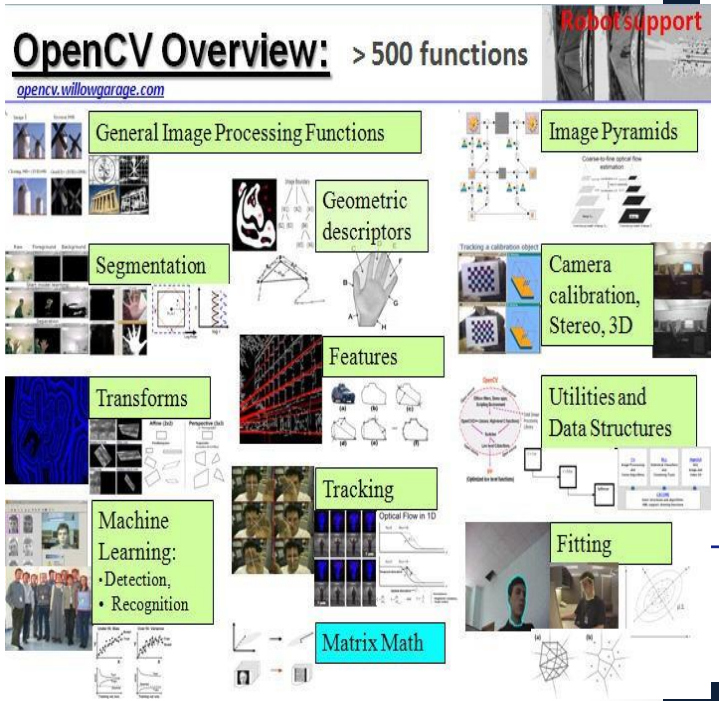
La biblioteca OpenCV

- ¿Qué es la biblioteca OpenCV?
  - Open Source Computer Vision
  - Librerías de visión por computador desarrolladas por Intel
  - 1999 versión alfa, 2022 versión 4.6.0
  - Apache 2 License.
  - Multiplataforma: Linux, MacOS X, Windows, Android/iOS
  - Funciones en C, C++, Python, Java

opencv.org

La biblioteca OpenCV

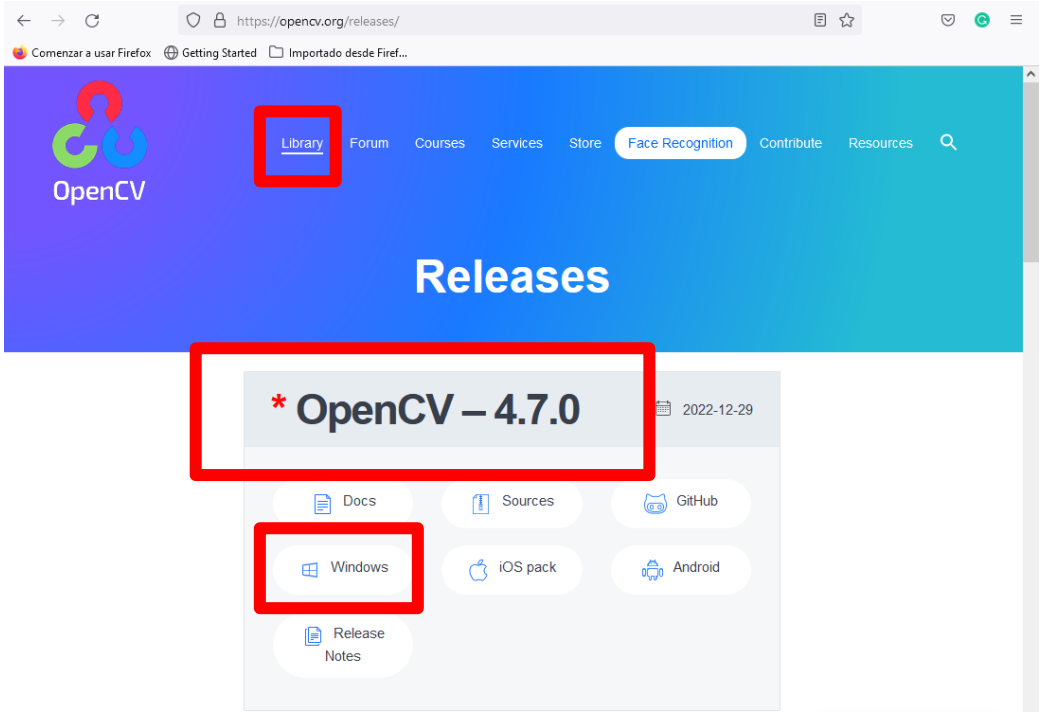
- Adquisición imágenes/video
- Procesamiento 2D
- Extracción características
- Machine Learning
- Reconocimiento - Clasificación
- Calibración 3D
- Localización – Reconstrucción 3D
- Aceleración GPU



Índice

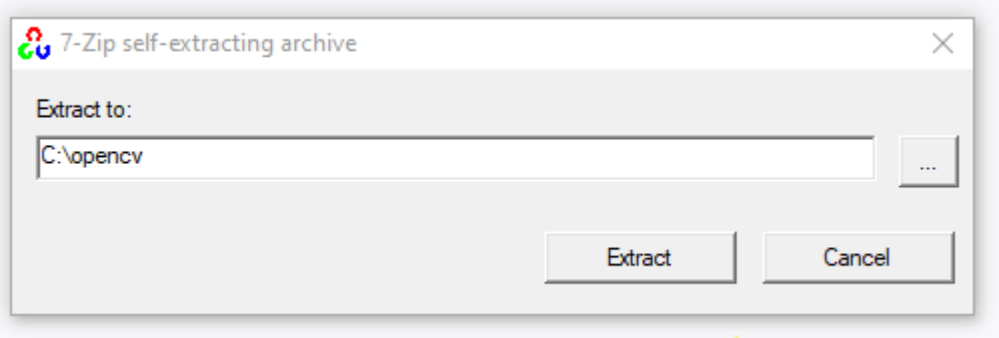
- La librería OpenCV
- Instalación
- Ejecución del primer programa
- Capturar imágenes desde un vídeo o desde una cámara

Instalación: opencv.org



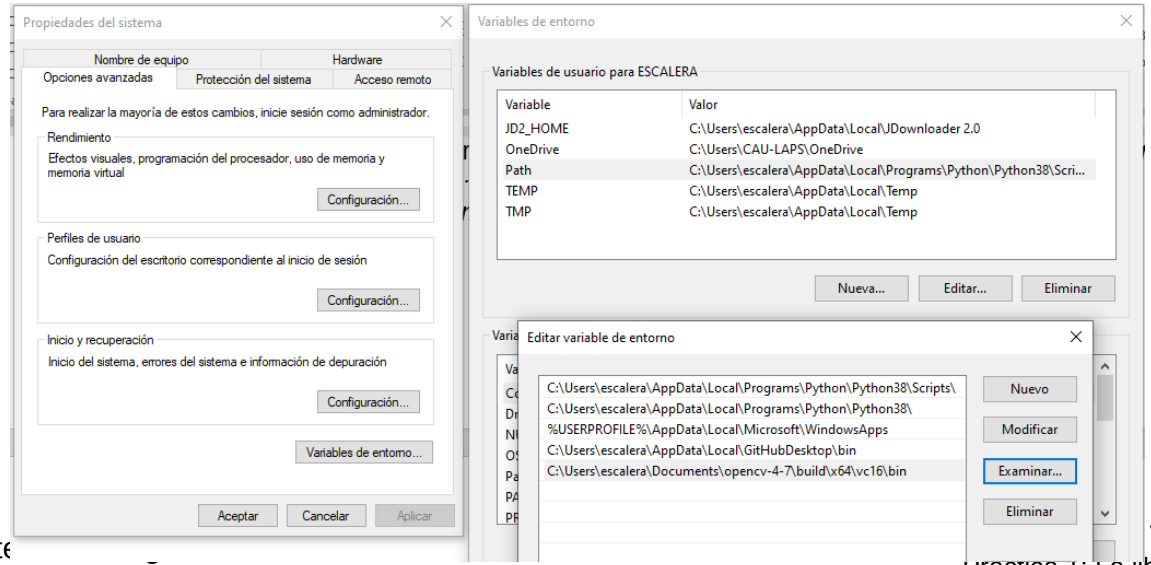
Instalación: opencv.org

- Al extraer, elegir el directorio donde queremos que se almacenen los ficheros
  - Por defecto es el directorio raíz



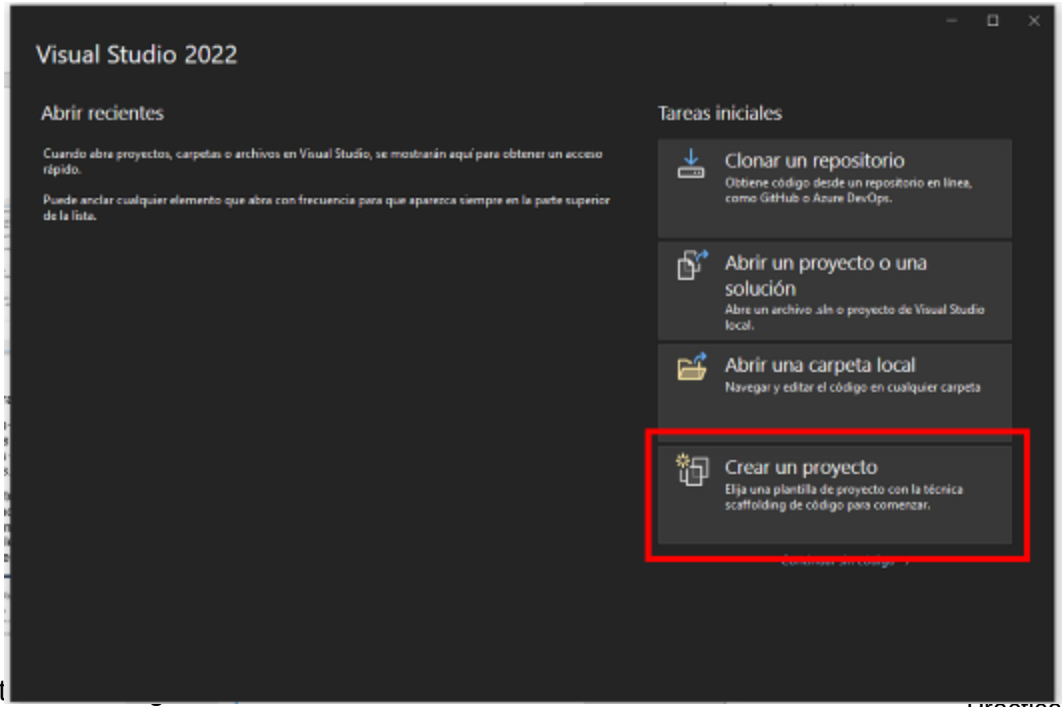
Instalación: Windows

- En “buscar”, “variables de entorno”
  - “Variables de usuario”-> PATH -> “Editar”
    - C:\Users\NombreUsusario\Documents\opencv\build\x64\vc16\bin

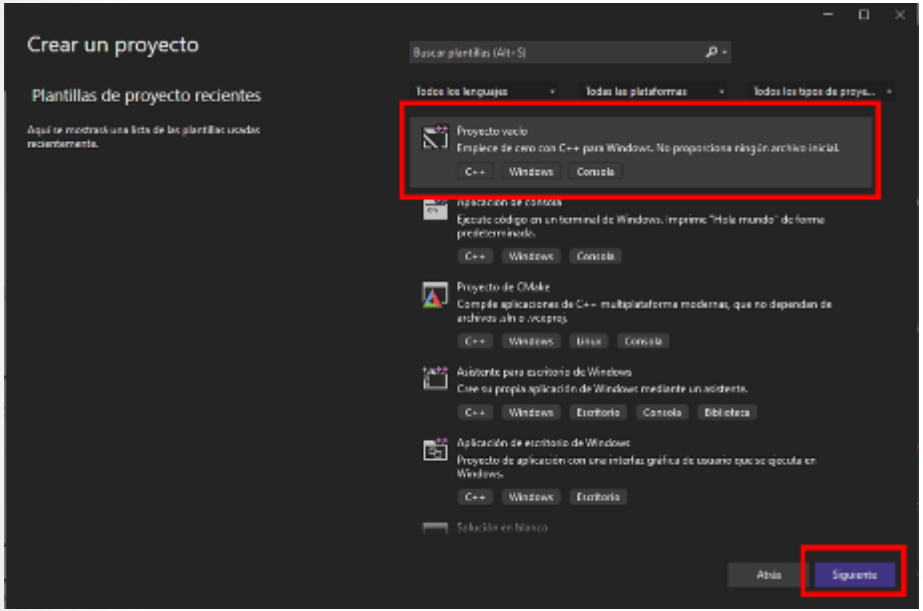


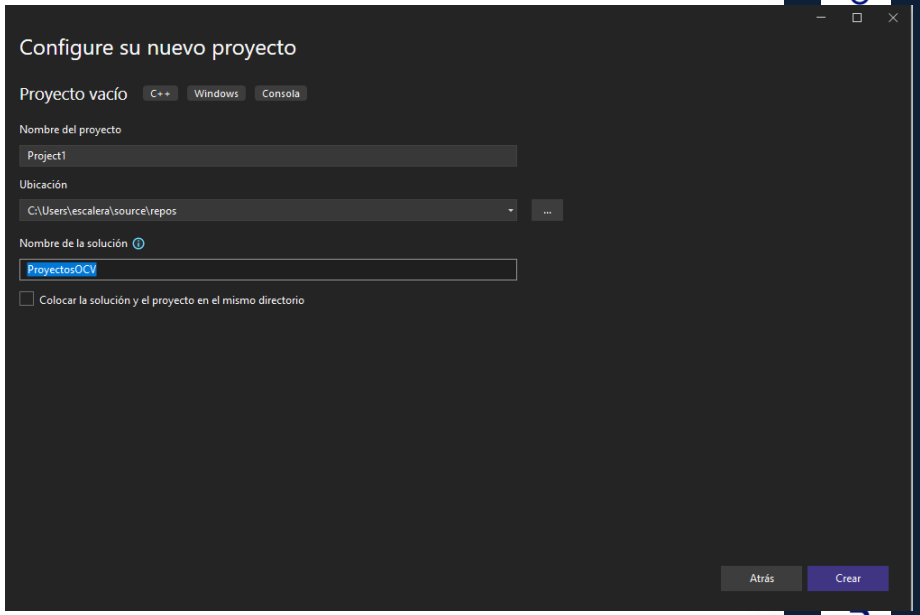
Instalación: MVS

- Microsoft Visual Studio

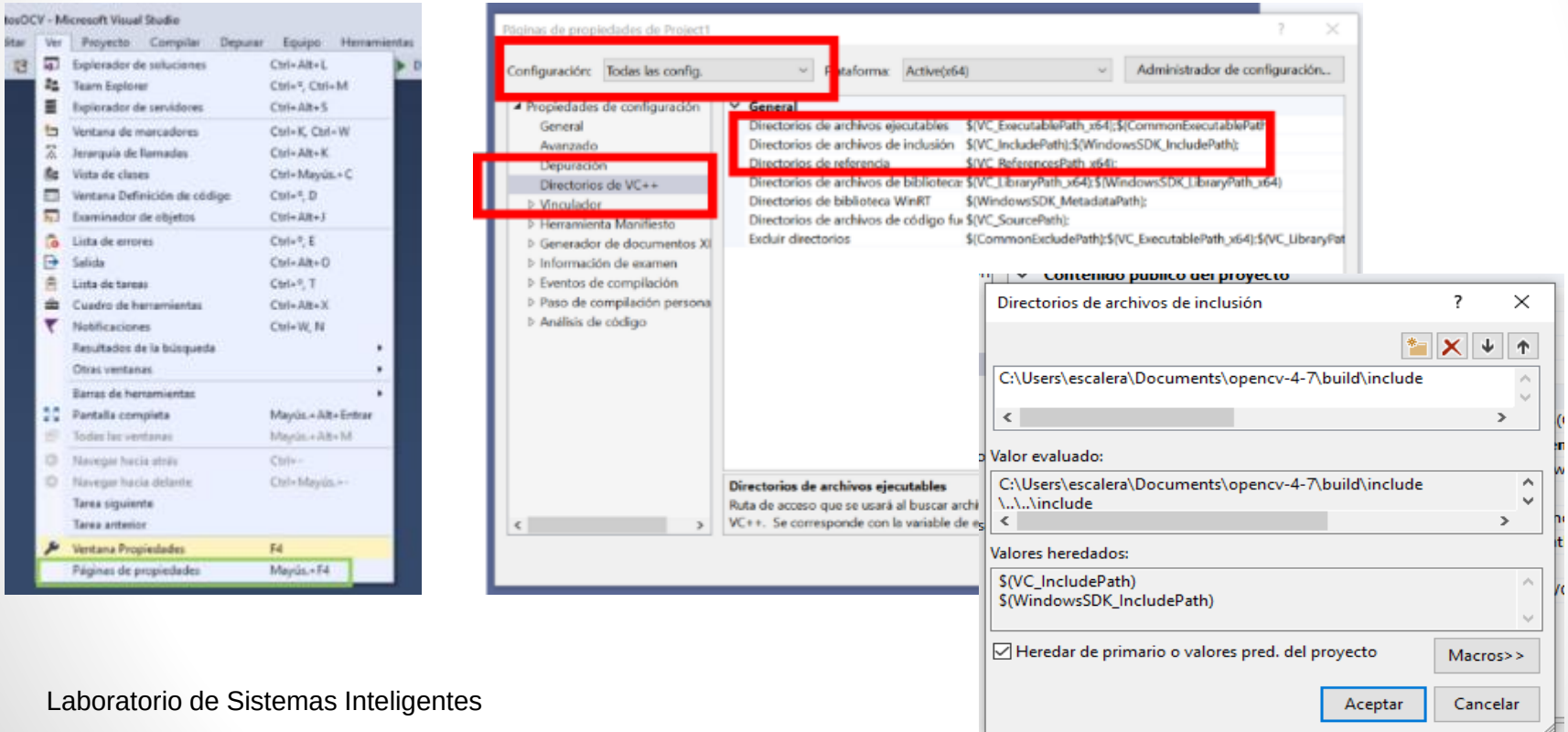


Instalación: MVS

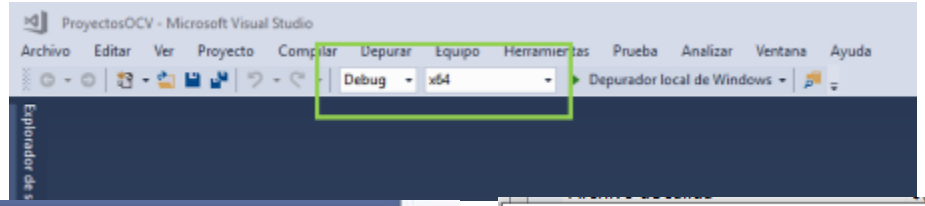


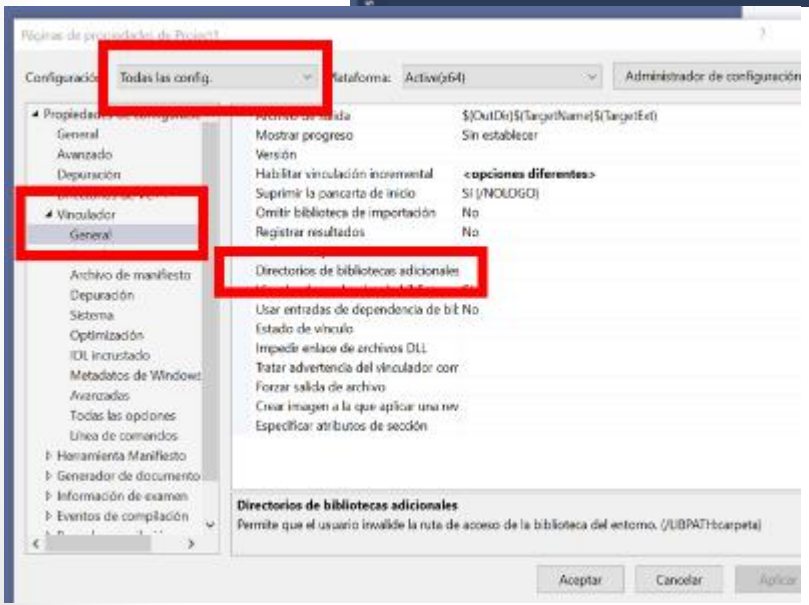


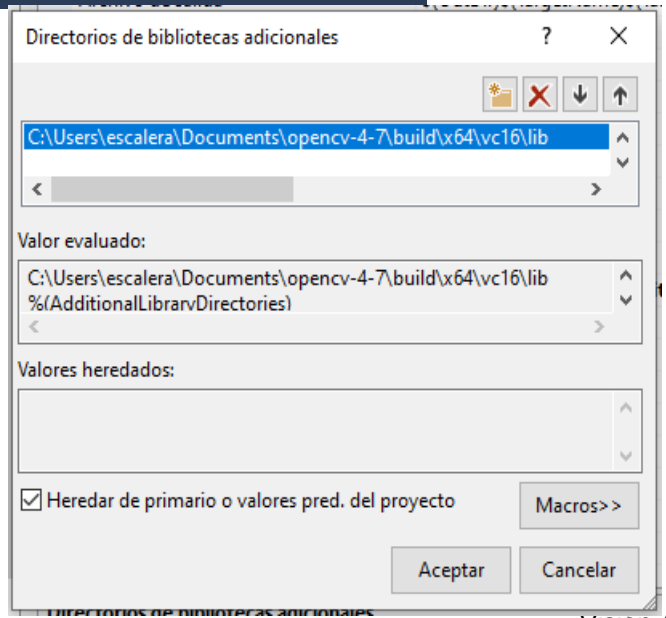
Instalación: MVS



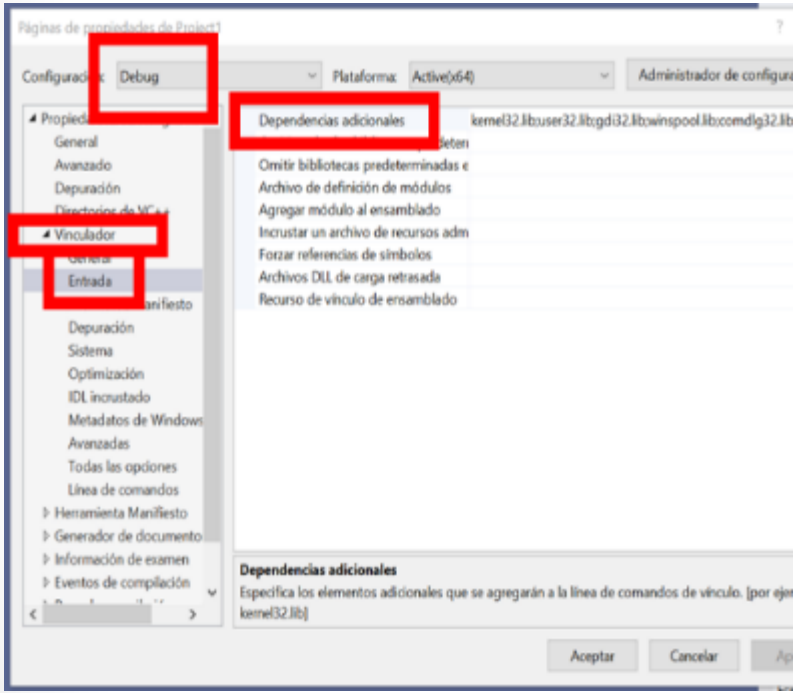
Instalación: MVS

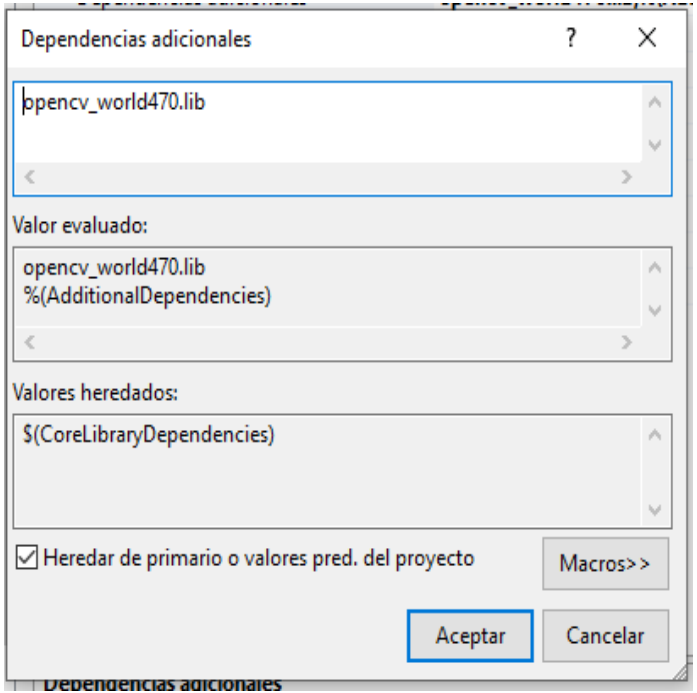






Instalación: MVS



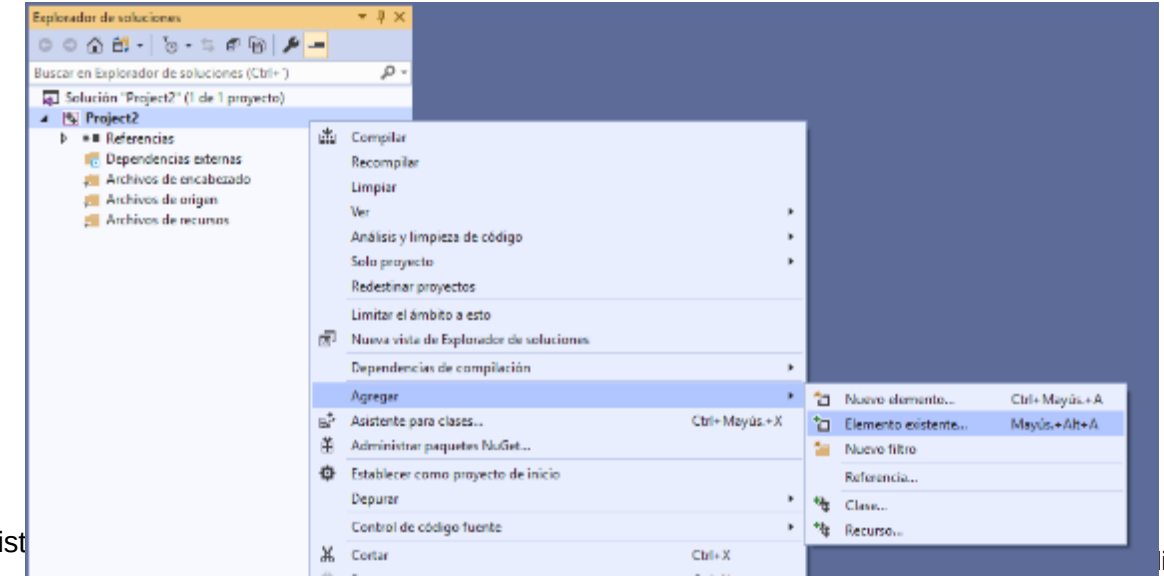


Índice

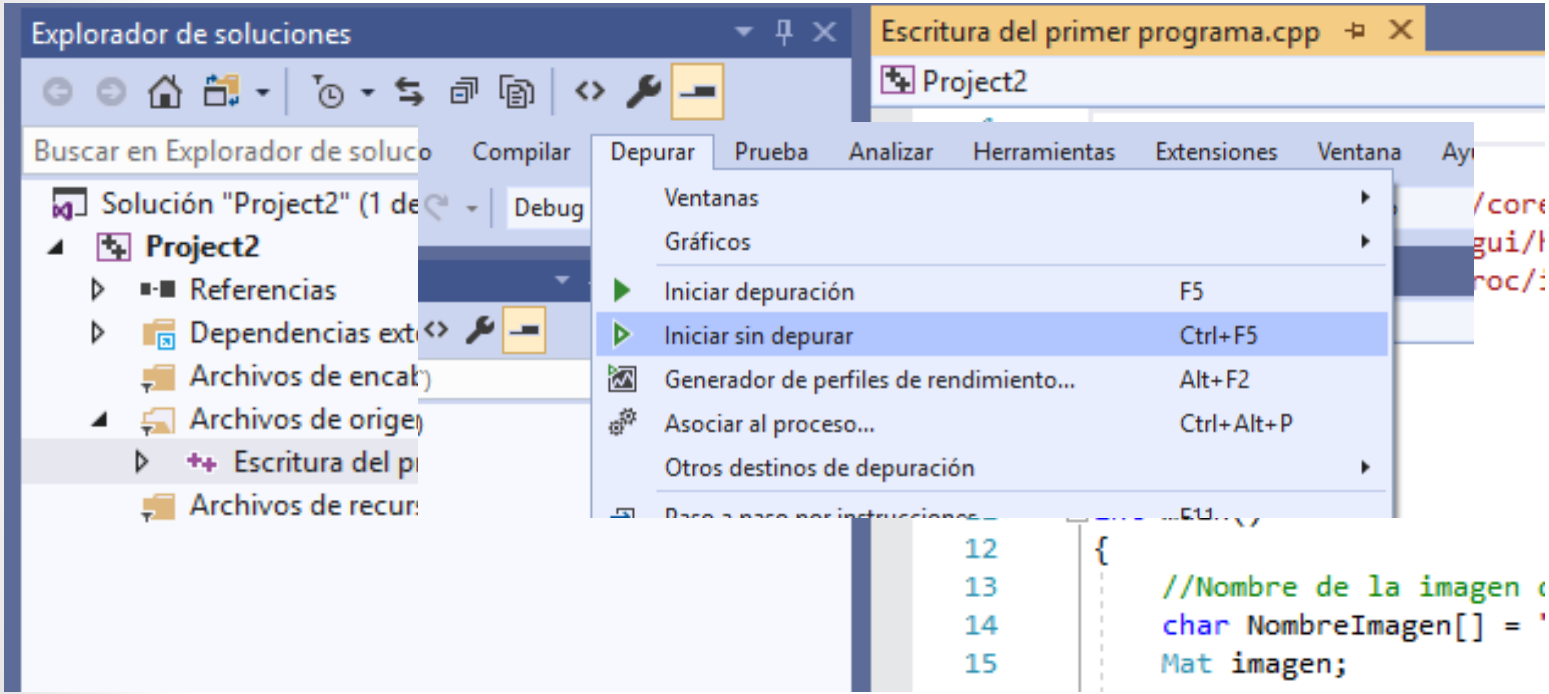
- La librería OpenCV
- Instalación
- Ejecución del primer programa
- Capturar imágenes desde un vídeo o desde una cámara

Ejecución del primer programa

- En el explorador de soluciones, si situamos el ratón en el proyecto y apretamos el botón derecho, se nos desplegará otro menú
- En él, tendremos la opción de “agregar” un “elemento existente”, que serán cada uno de los ejemplos que vamos a usar en el curso



Ejecución del primer programa

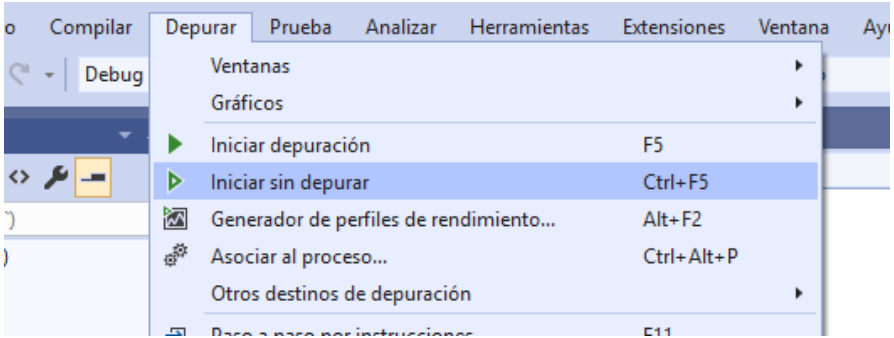


Ejecución del primer programa

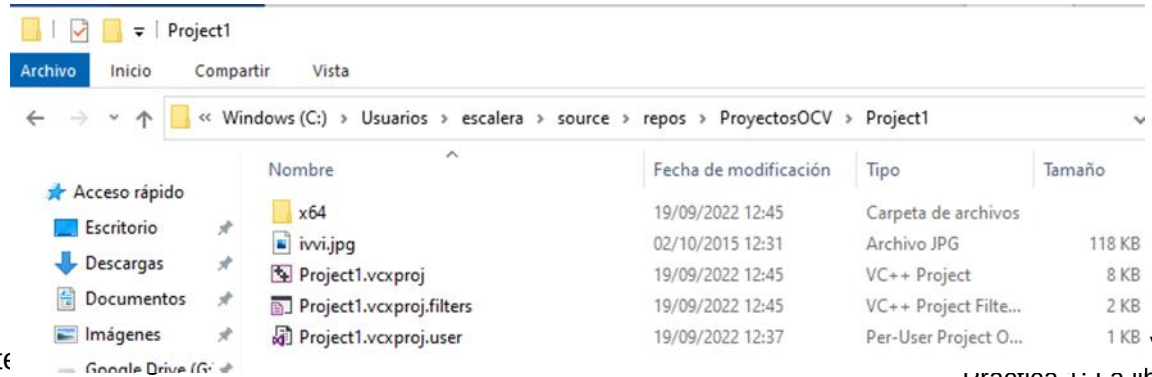
1. Nombre de la imagen en el disco
2. Cargar la imagen y comprobar que se cargado correctamente
3. Mostrar la imagen
4. Esperar a la pulsación de cualquier tecla

```cpp
//Se carga una imagen de disco y se muestra por pantalla
#include <opencv2/highgui/highgui.hpp>
#include <iostream>

using namespace std;
using namespace cv;

#define IMAGEN "ivvi.jpg"

int main()
{
    // Variables
    //Nombre de la imagen que se va a cargar
    char NombreImagen[] = IMAGEN;
    Mat imagen;

    //Se carga la imagen desde disco y se comprueba que lo ha hecho correctamente
    imagen = imread(NombreImagen);
    if( !imagen.data ) {
        cout<< "Error al cargar la imagen: " << NombreImagen <<endl;
        exit(1);
    }

    //Mostrar la imagen
    // Se crea un lienzo donde mostrar imagenes
    // Se asocia la imagen con el lienzo

    namedWindow("Original", WINDOW_AUTOSIZE);
    imshow("Original", imagen );

    // Se muestra la imagen un tiemp omedido en ms.
    // 0 significa que se espera hasta pulsar un tecla
    waitKey(0);

    return 0;
}
```

Ejecución del primer programa



- ¿Dónde está la imagen?



Ejecución del primer programa

Mat

class 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
    ...
};
```

Ejecución del primer programa

imread

Loads an image from a file.

C++: Mat **imread**(const string& **filename**, int **flags**=1)

Python: cv2.**imread**(filename[, flags]) → retval

C: IplImage* **cvLoadImage**(const char* **filename**, int **iscolor**=CV_LOAD_IMAGE_COLOR)

C: CvMat* **cvLoadImageM**(const char* **filename**, int **iscolor**=CV_LOAD_IMAGE_COLOR)

Python: cv.**LoadImage**(filename, iscolor=CV_LOAD_IMAGE_COLOR) → None

Python: cv.**LoadImageM**(filename, iscolor=CV_LOAD_IMAGE_COLOR) → None

Ejecución del primer programa

Parameters

filename – Name of file to be loaded.

flags – Flags specifying the color type of a loaded image:

- CV_LOAD_IMAGE_ANYDEPTH - If set, return 16-bit/32-bit image when the input has the corresponding depth, otherwise convert it to 8-bit.
- CV_LOAD_IMAGE_COLOR - If set, always convert image to the color one
- CV_LOAD_IMAGE_GRAYSCALE - If set, always convert image to the grayscale one
- **>0 Return a 3-channel color image.**

Note: In the current implementation the alpha channel, if any, is stripped from the output image. Use negative value if you need the alpha channel.

- **=0** Return a grayscale image.
- **<0** Return the loaded image as is (with alpha channel).

Ejecución del primer programa

- Formatos soportados
  - Windows bitmaps - *.bmp, *.dib
  - JPEG files - *.jpeg, *.jpg, *.jpe
  - JPEG 2000 files - *.jp2
  - Portable Network Graphics - *.png
  - Portable image format - *.pbm, *.pgm, *.ppm
  - Sun rasters - *.sr, *.ras
  - TIFF files - *.tiff, *.tif

Ejecución del primer programa

namedWindow

Creates a window.

C++: void **namedWindow**(const string& **winname**, int **flags**=WINDOW_AUTOSIZE)

Python: cv2.**namedWindow**(winname[, flags]) → None

C: int **cvNamedWindow**(const char* **name**, int **flags**=CV_WINDOW_AUTOSIZE)

Python: cv.**NamedWindow**(name, flags=CV_WINDOW_AUTOSIZE) → None

Parameters

name – Name of the window in the window caption that may be used as a window identifier.

flags – Flags of the window. Currently the only supported flag is `CV_WINDOW_AUTOSIZE`. If this is set, the window size is automatically adjusted to fit the displayed image (see `imshow()`), and you cannot change the window size manually.

Ejecución del primer programa

imshow

Displays an image in the specified window.

C++: void **imshow**(const string& **winname**, InputArray **mat**)

Python: cv2.**imshow**(winname, mat) → None

C: void **cvShowImage**(const char* **name**, const CvArr* **image**)

Python: cv.**ShowImage**(name, image) → None

Parameters

winname – Name of the window.

image – Image to be shown.

The function **imshow** displays an image in the specified window. If the window was created with the CV_WINDOW_AUTOSIZE flag, the image is shown with its original size. Otherwise, the image is scaled to fit the window. The function may scale the image, depending on its depth:

Ejecución del primer programa

waitKey

Waits for a pressed key.

C++: int **waitKey**(int **delay**=0)

Python: cv2.**waitKey**([delay]) → retval

C: int **cvWaitKey**(int **delay**=0)

Python: cv.**WaitKey**(delay=0) → int

delay – Delay in milliseconds. 0 is the special value that means "forever".

Índice

- La librería OpenCV
- Instalación
- Ejecución del primer programa
- Capturar imágenes desde un vídeo o desde una cámara

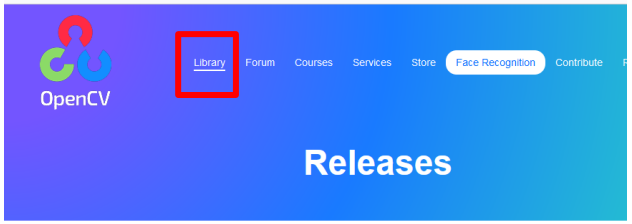
Capturar imágenes desde un vídeo

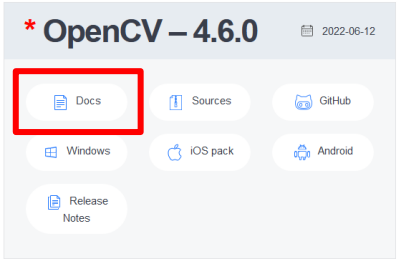
1. Cargar el archivo del vídeo
2. Comprobar que se ha cargado completamente
3. Extraer una imagen
4. Comprobar que no se ha llegado al final del vídeo
5. Mostrar la imagen
6. Espera a presionar , o no, una tecla
7. Si no es ESC ir a 3

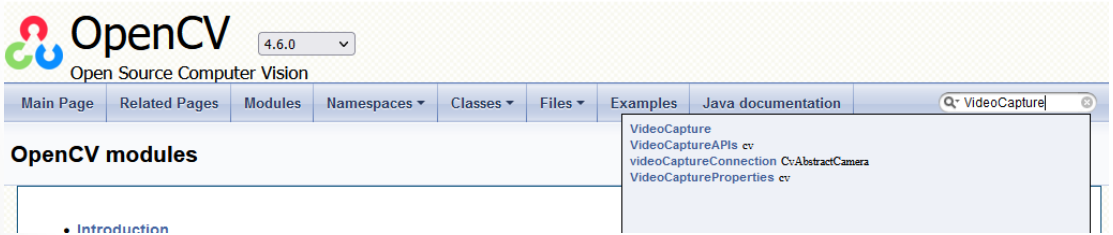
```cpp
//Se cargan imagenes desde un video y se muestran por pantalla
#include <opencv2/highgui/highgui.hpp>
#include <iostream>

using namespace cv;
using namespace std;

#define TECLA_ESCAPE 27
#define VIDEO "icab_demo_EPS.mp4"

int main(int argc, char* argv[])
{
    // // Variables
    Mat frame;
    VideoCapture capture(VIDEO);
    // tecla presionada
    char pressedKey = 0;
    // Se crea un lienzo donde mostrar imagenes
    namedWindow("video", WINDOW_AUTOSIZE);

    // Carga el video
    //capture.open(VIDEO);
    if (!capture.isOpened()){
        cout << "Error al cargar el video!" << endl;
        return 1;
    }
    else{
        while (pressedKey != TECLA_ESCAPE){
            // Se lee el vidoe imagen a imagen
            //capture >> frame;
            capture.read(frame);
            // Se comprueba que no se ha llegaod al final
            if (frame.empty()) {
                cout << "Se ha llegado al final del video" << endl;
                return 1;
            }
            // Se muestra la imagen
            imshow("video", frame);
            // Se espera 20 ms
            pressedKey = waitKey(20);
        }
    }
}
```

Capturar imágenes desde un vídeo

1. Ir a opencv.org / library /docs
2. Buscar VideoCapture
3. Fijarse en la función get()







Capturar imágenes desde una cámara

- #define CAMARA 0
- #define VIDEO CAMARA

```
#define TECLA_ESCAPE 27
#define CAMARA_0 0


int main()
{
    // Variables
    Mat frame;
    VideoCapture capture(CAMARA_0);
    // tecla presionada
    char pressedKey = 0;
    // Se crea un lienzo donde mostrar imagenes
    namedWindow("video", WINDOW_AUTOSIZE);

    // comprobar que la camara se ha conectado
    if (!capture.isOpened()){
        cout << "Error al conectarse a la camara!" << endl;
        return 1;
    }
```