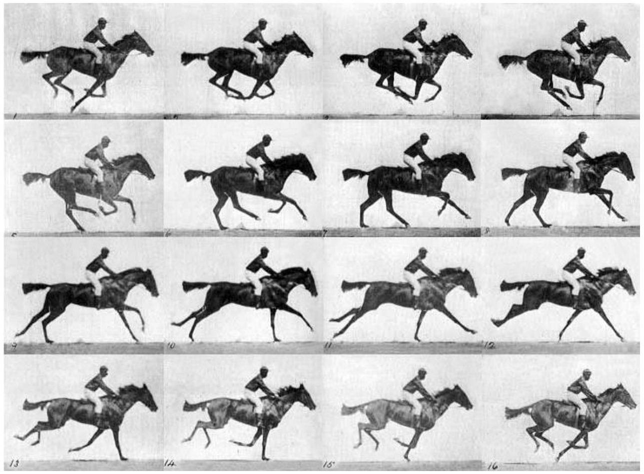
COMPUTER VISION AND IMAGE PROCESSING

LAB SESSION 3
WORKING WITH VIDEO STREAMS

ALESSIO TONIONI - alessio.tonioni2@unibo.it

Video streams



The Horse in Motion. Eadweard Muybridge, 1878

- **Video:** Temporal sequence of images (*frames*).
- **Frame Rate:** number of frame in 1sec of video. Usually one of 24/30/60.
- **Video Elaboration (*naive*):** Each frame is elaborated separately with image processing algorithms.
- **Video Elaboration (optimized):** Only keyframe of the original video are elaborated. The missing ones are reconstructed by interpolation.

Video streams – OpenCV

C++

`cv::VideoCapture`: object that represents a video stream (from either a file or a device such as a webcam).

From File:

```
cv::VideoCapture
stream(std::string filename);
```

From Camera:

```
Cv::VideoCapture stream(int
camID);
```

C

`CvCapture`: struct that represents a video stream (from either a file or a device such as a webcam).

From File:

```
CvCapture *stream  =
cvCaptureFromFile(char
*filename);
```

From Camera:

```
CvCapture *stream  =
cvCaptureFromCAM (int camID);
```

`camID` usually «counts» the number of cameras plugged to the computer (starting from 0)

Video streams– OpenCV (cont.)

C++

Read a single frame:

```
cv::Mat frame;
cv::VideoCapture stream;
stream.read(frame);
```

If no frame is available (video ended) the function returns `false`.

C

Read a single frame:

```
IplImage * frame =
cvQueryFrame(CvCapture
*stream);
```

If no frame is available (video ended) the function returns `NULL`.

Video streams– OpenCV (cont.)

C++

When done close the stream:

```
cv::VideoCapture stream
(filename);
…
stream.release();
```

C

When done close the stream:

```
CvCapture *stream  =
cvCaptureFromFile(filename);
…
cvReleaseCapture(&stream);
```

Video Stream Labs

- **Goal**: implement single frame elaboration and two frame difference to detect changes in video sequences.

- You can either use one of the video in the '*videos*' folder inside *ElabImage* or use your webcam.

- To choose between C or C++ comment or uncomment the first define on top of '*lab3.cpp*'.

C++

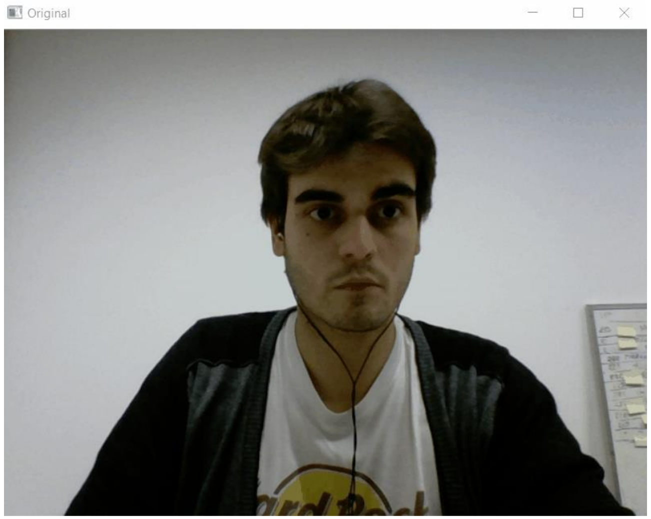
```
#define OPENCV_CPP_INTERFACE
```

C

```
//#define OPENCV_CPP_INTERFACE
```

Exercise 0

- Open '**Lab_3.cpp**'
- Add in function ***main*** the code necessary to open a video stream either from a video file or from your webcam
- Write a **while** loop to iterate over all the frame in the video and display it one at time (i.e. reproduce the video).



Exercise 1

- Add inside the while loop in function ***main*** the code necessary to apply the **convolution** introduced in lab_2 to each frame of the video stream.
- Display in two separate window the original frame and the convolved one.
- You can either use your function developed in the last laboratory or the OpenCV implementation.
- Try out different convolutional kernel.
- OpenCV documentation – link.

Exercise 1- OpenCV Convolution

C++

```
void filter2D(cv::Mat& src,
cv::Mat& dst, int ddepth,
InputArray kernel)
```

C

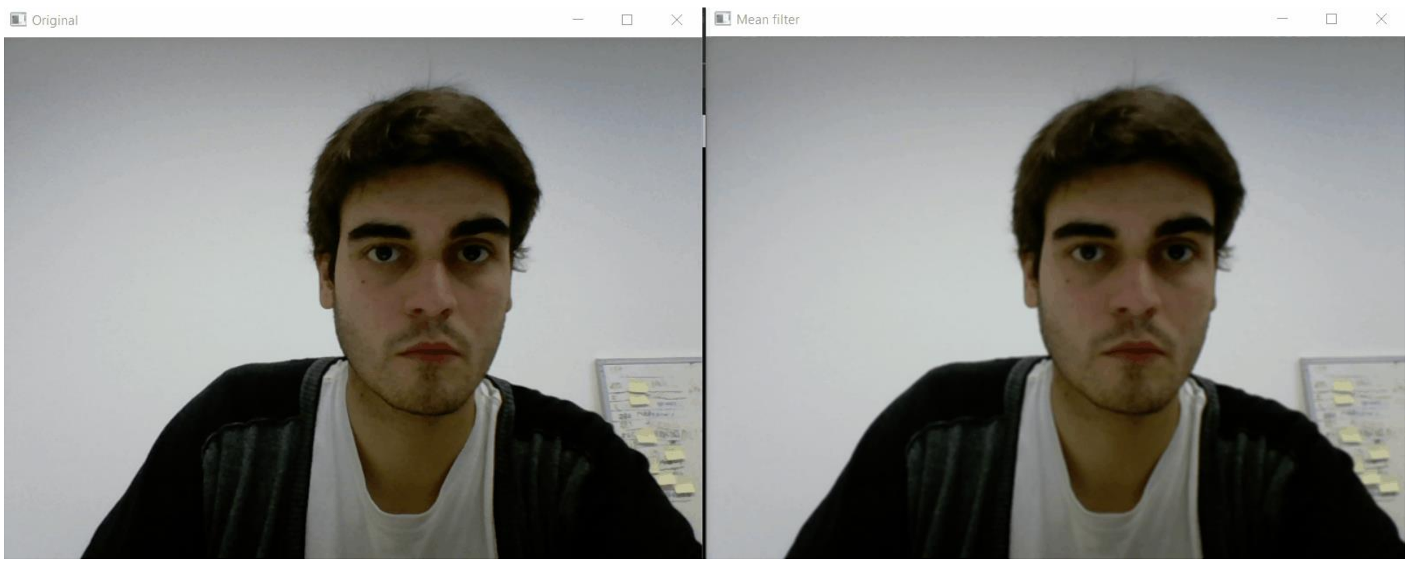
```
void cvFilter2D(const CvArr*
src, CvArr* dst, const CvMat*
kernel)
```

- **src –** input image
- **dst –** output image of the same size and teh same number of channels as *src*
- **ddepth –** desired depth of the destination image, if negative it will be the same as *src.depth()*
- **kernel –** convolutional kernel, a single channel floating point matrix; to apply different kernel to different channels, split the image into separate color planes using *split()* and process them individually.

Exercise 1- Convolution

Mean filter

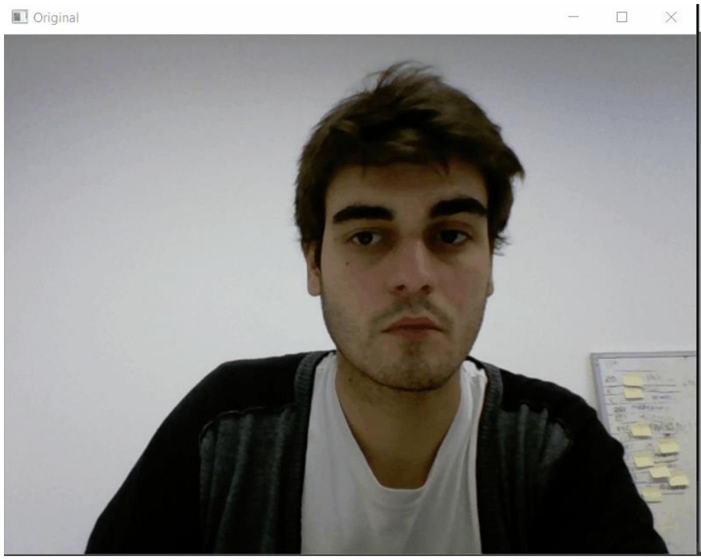
$$K(m,n) = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

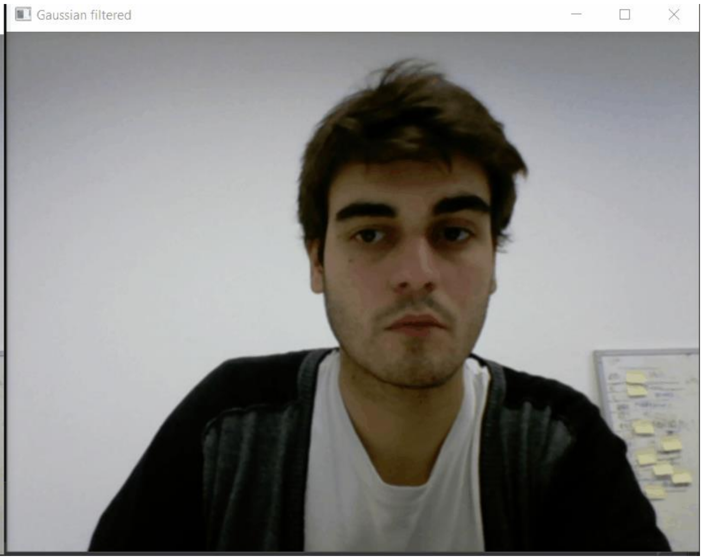


Exercise 1- Convolution

Gaussian blur filter

$$K(m,n) = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

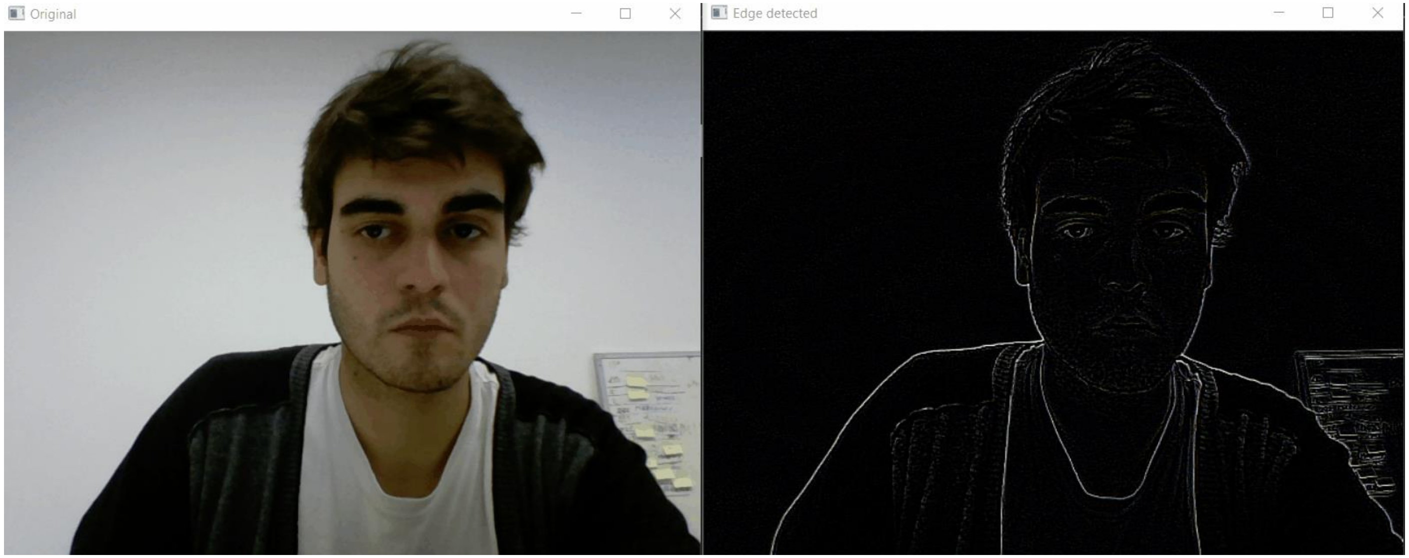




Exercise 1- Convolution

Edge detection filter

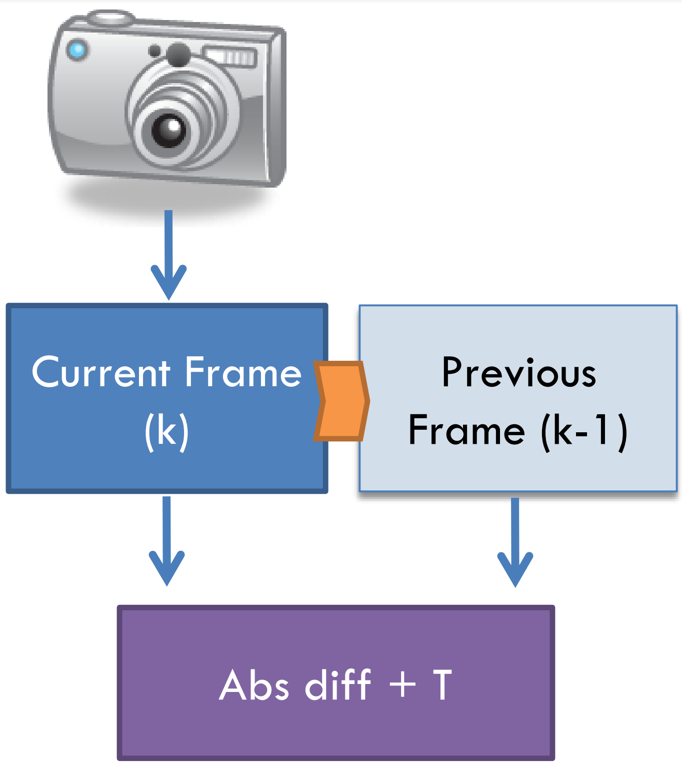
$$K(m,n) = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$



Change detection algorithms

- **Change Detection**: detection of 'meaningful' changes occurring in a scene by processing of images captured at different time instants.
- **Input**: video sequence of monitored scene
- **Output**: '*change mask*' → greyscale image were changed pixel are white (255) all the other black (0).
- **Assumption**: static high frame rate camera.
- **Applications**: surveilance, trafic monitoring…

Two-frame difference



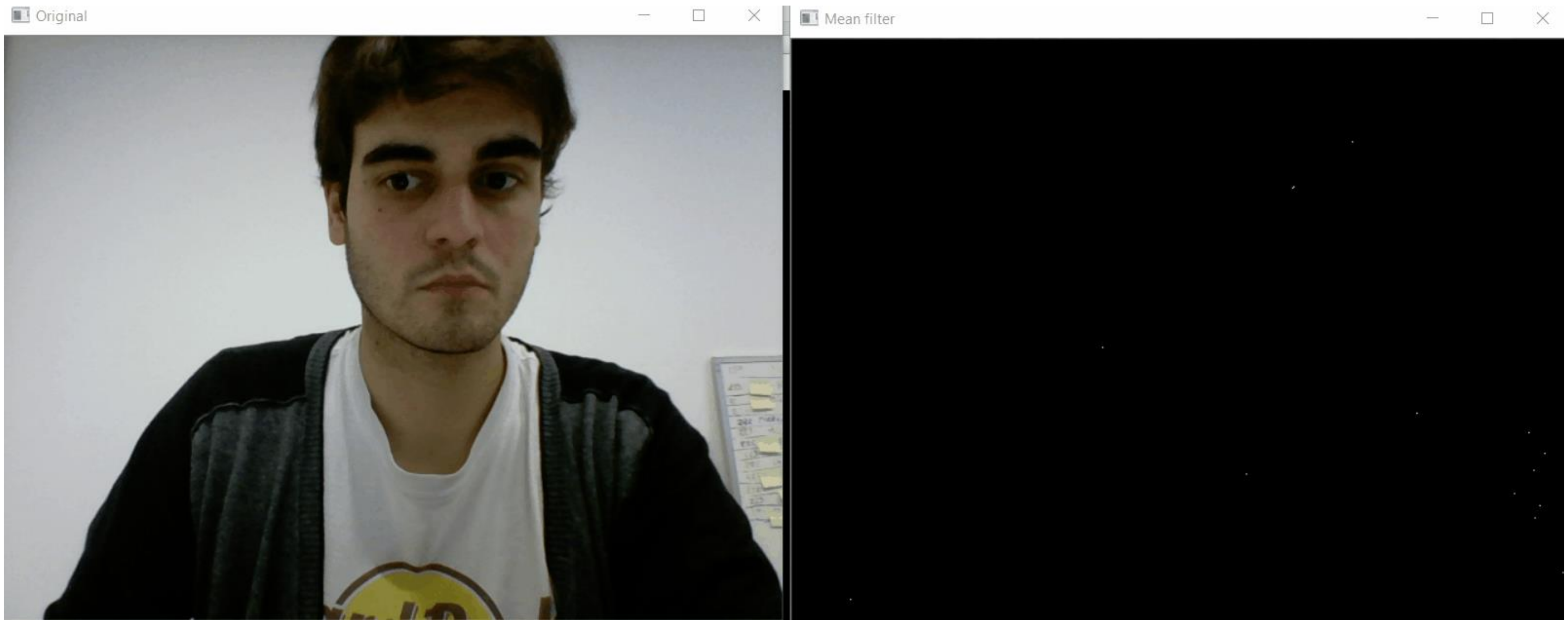
At the end of each loop, the current frame becomes the previous one:

- C→`cvCopyImage(previous, current);`
- C++→`current.copyTo(previous)`

$$P_m(i,j) = \begin{cases} 255 \; if \; |P_{k-1}(i,j) - P_k(i,j)| > T \\ 0 \qquad\qquad\qquad\qquad\qquad otherwise \end{cases}$$

- $P_m(i,j)$: pixel intensity at position (i,j) in the output mask.
- $P_k(i,j)$: pixel intensity at position (i,j) in frame k.
- $P_{k-1}(i,j)$: pixel intensity at position (i,j) in frame k-1.
- *T*: threshold.

Two-frame difference



Threshold = 30