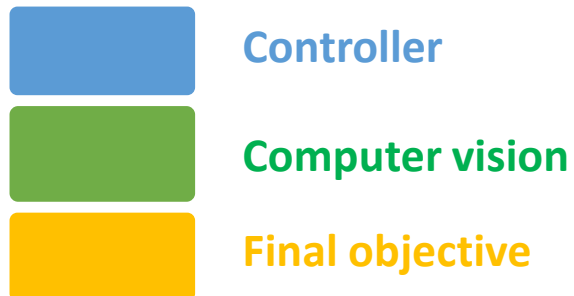
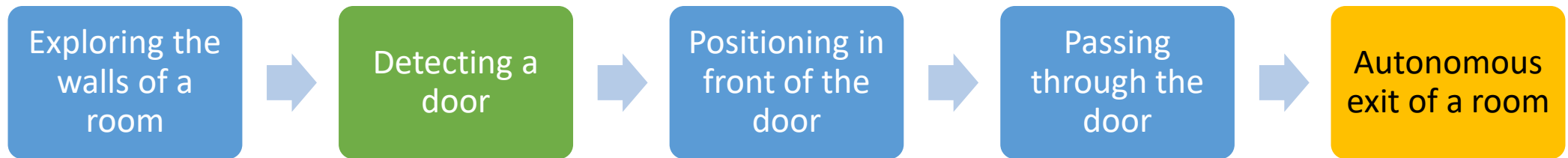


# Debriefing 2/11/17

Master thesis about autonomous drones

Aurian d'Avernas

# Reminder of the main goal



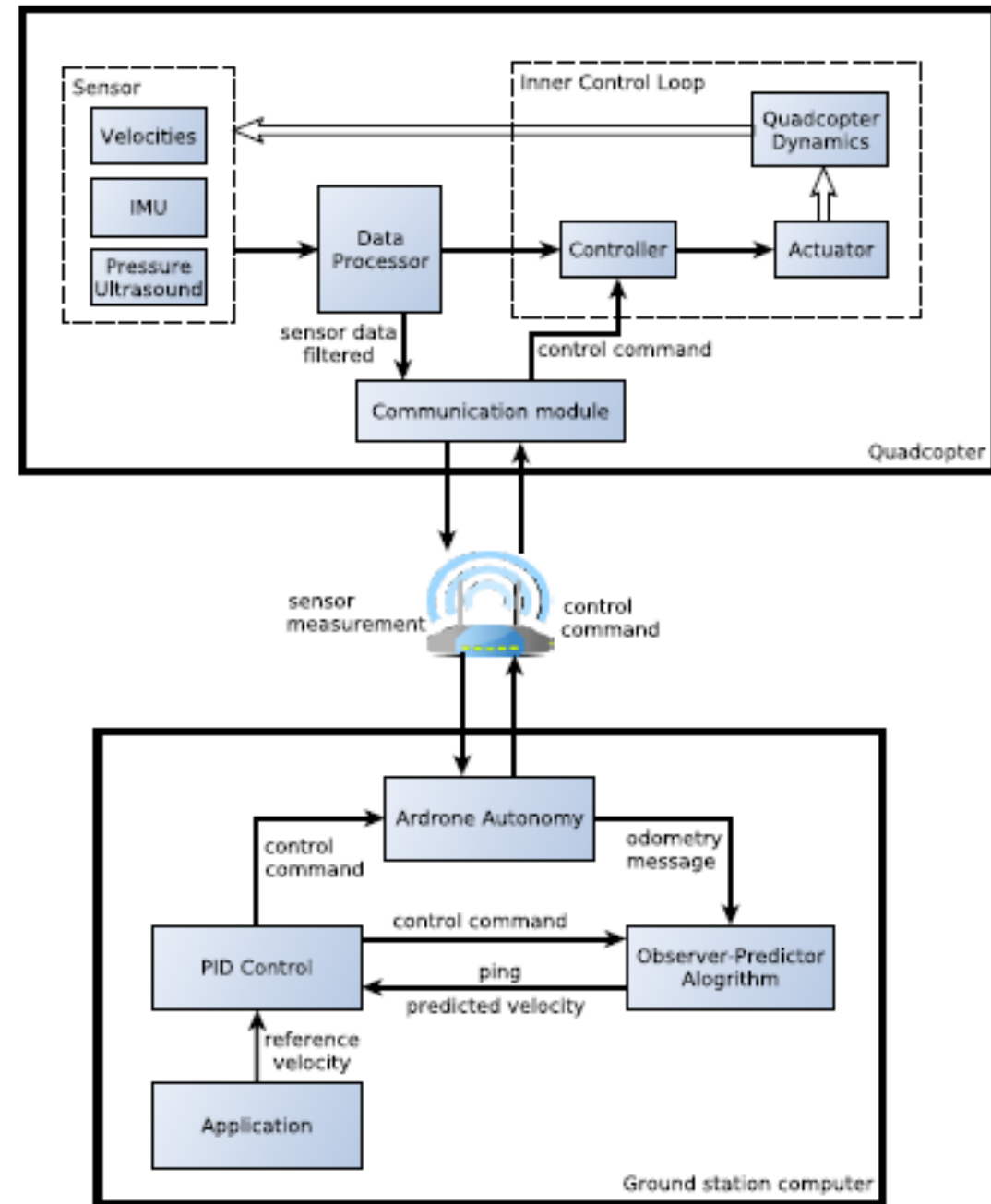
# Controller

- Need for a controller because no position or velocity controller is included in ardrone\_autonomy package when using ARDrone 2.0 with ROS. Only a shaky closed-source inner velocity control loop is present inside ARDrone 2.0.
- UCL 2016 teams implemented a PD controller proper to their own visual pose based on their SLAM process
  - no integral term → accumulating small errors over large distances
  - mandatory to enable SLAM process in order to benefit of their position controller → resources (battery, transmission rate) and time demanding  
→ not adapted for fast exploration of a room
- Choice of a new position controller based on TUM and TUD previous works

# TUD controllers

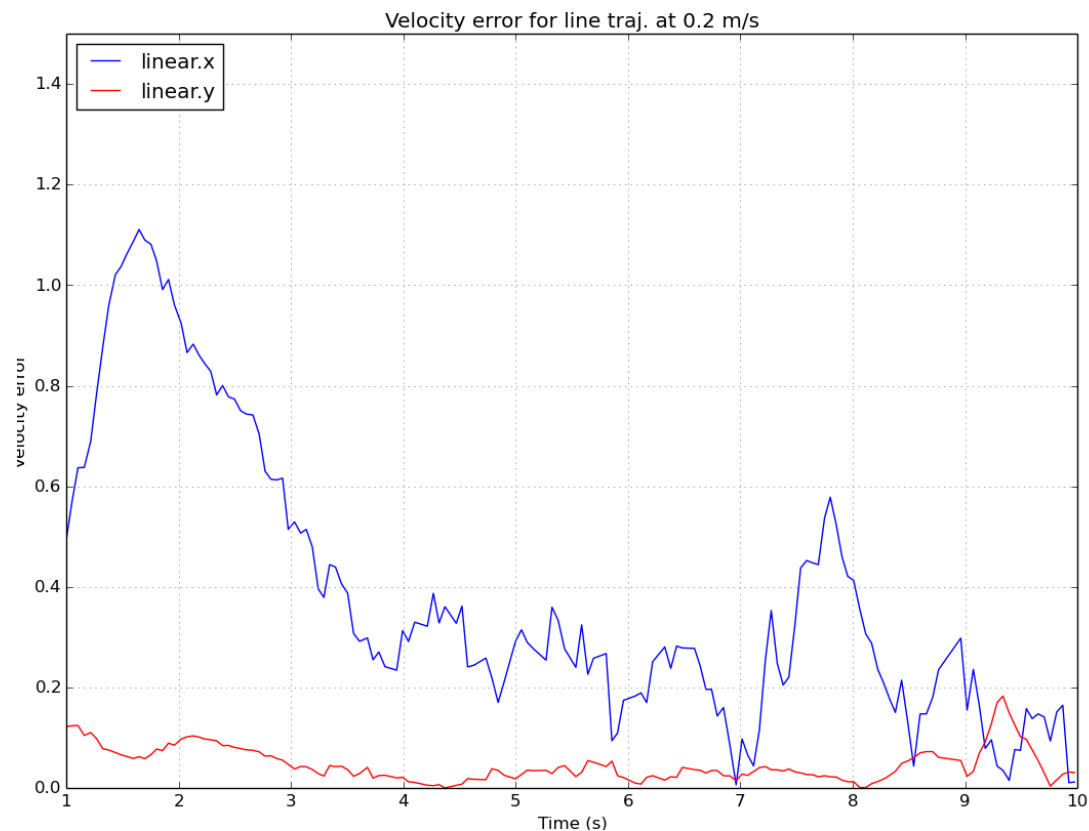
- Two controllers have been adapted and experimented
  - classical PID position controller
  - classical PID position controller with Extended Kalman Filter in order to take into account the transmission delays induced by the wifi communication system used to communicate with the drone
- Both controllers provide good performances, but the addition of EKF shows significant better results
  - slightly better response times for PID with EKF than without EKF
  - far better dampings for PID with EKF than without EKF

# PID controller with EKF

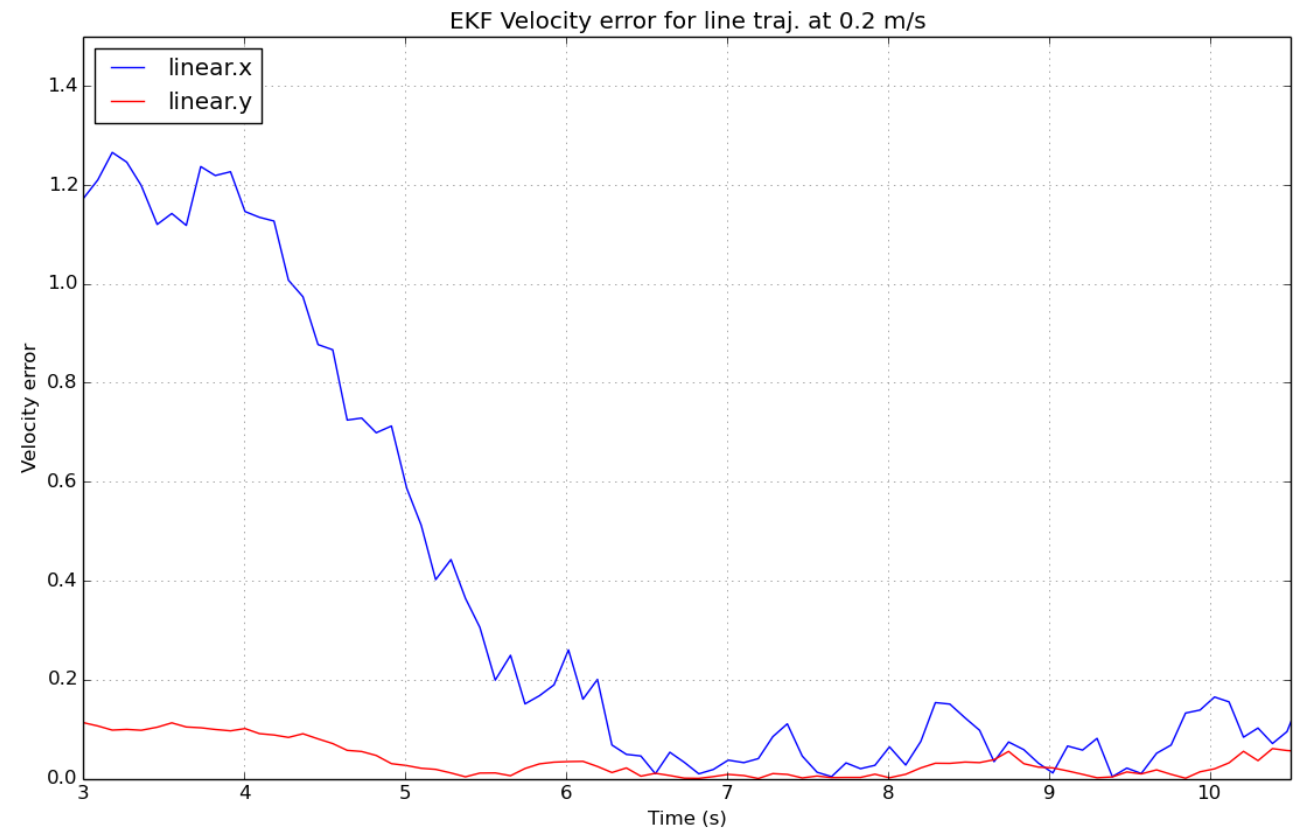


# Controller: comparison classic PID w/ & w/o EKF

**PID controller error after a reference velocity step of 0.2 m/s to travel 2 m**

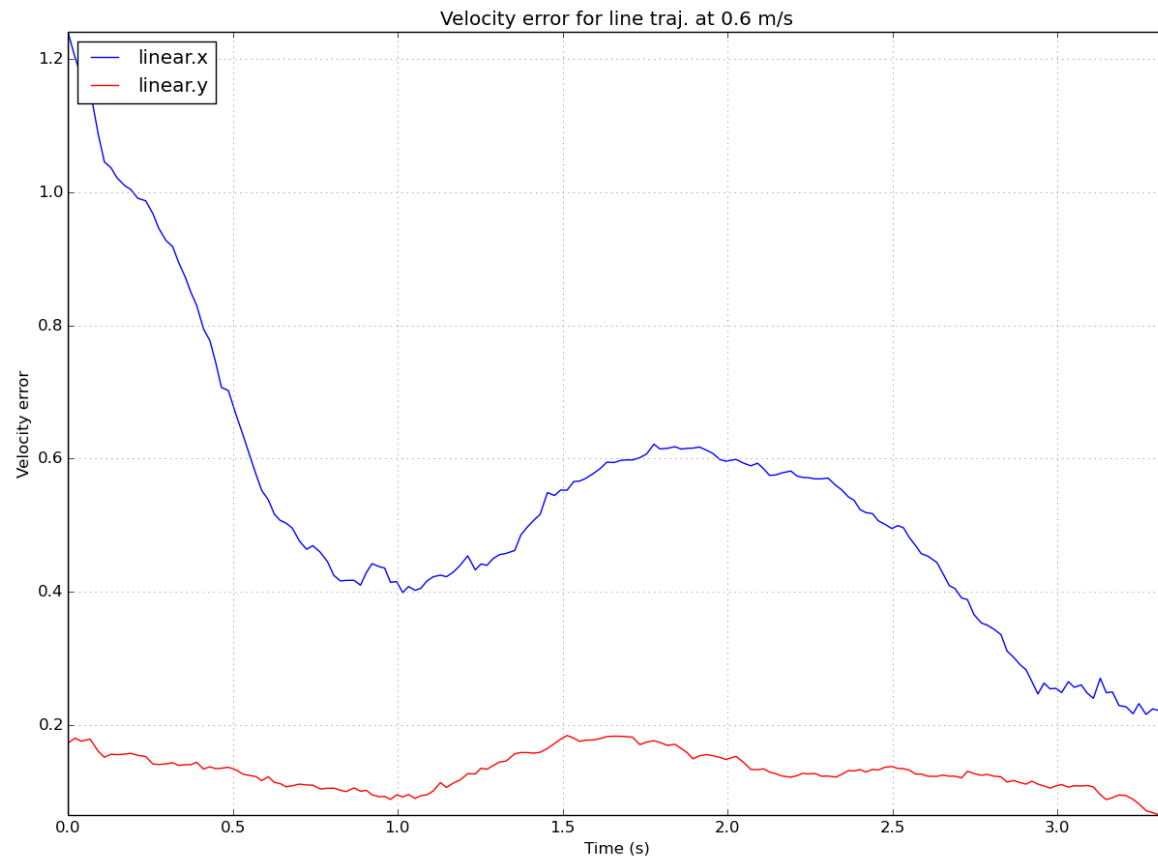


**PID controller + EKF after a reference velocity step of 0.2 m/s to travel 2 m**

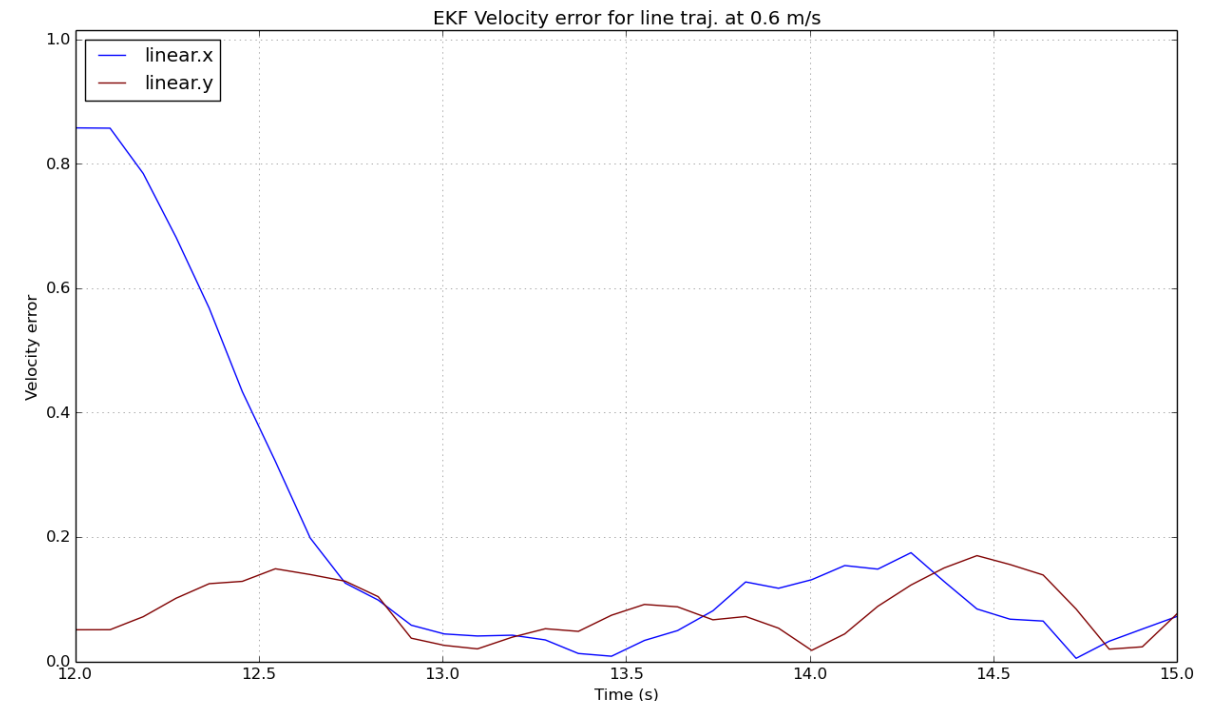


# Controller: comparison classic PID w/ & w/o EKF

**PID controller after a reference velocity step of 0.6 m/s to travel 2 m**

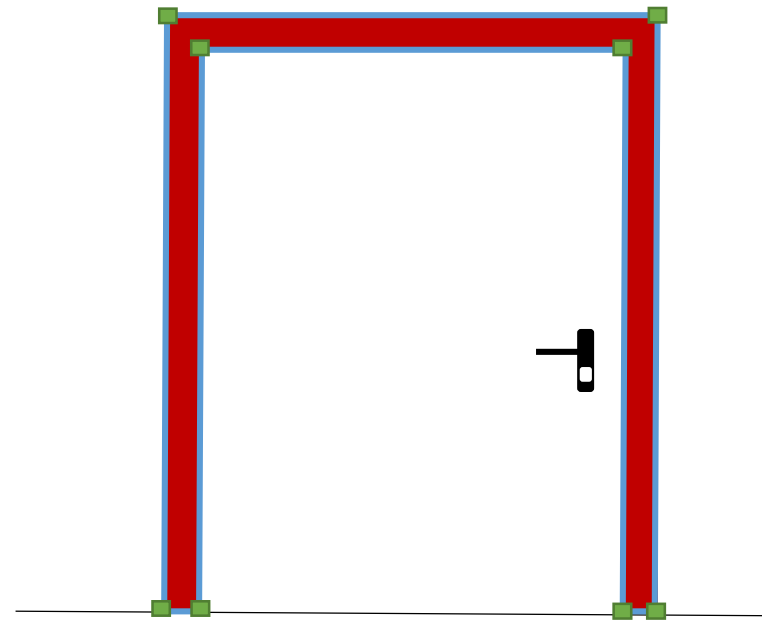


**PID controller + EKF after a reference velocity step of 0.6 m/s to travel 2 m**



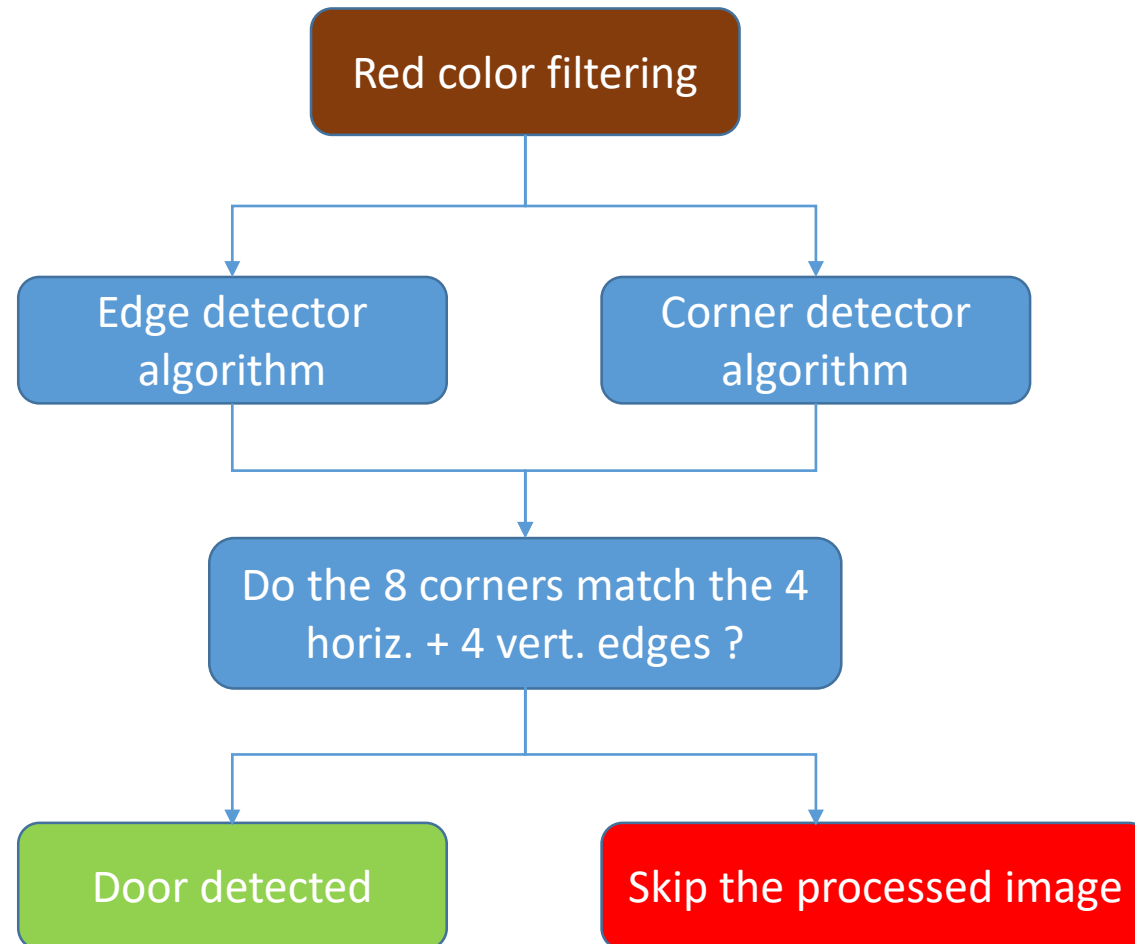
# Computer vision: detection of a door

- Simplifying hypothesis
  - All the doors have a red contour
- The door is described by
  - Red **color** → Color filter
  - 8 **edges** → Edge detector
  - 8 **corners** → Corner detector



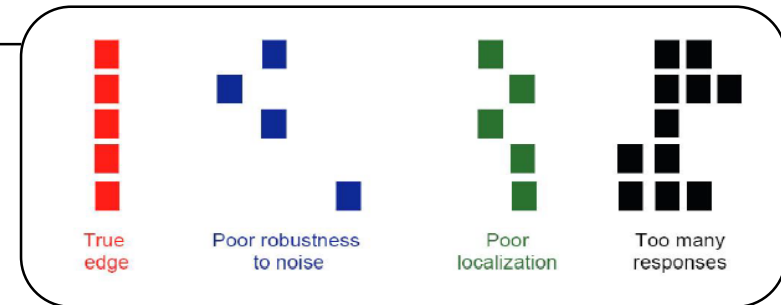


# Computer vision: Main algorithm

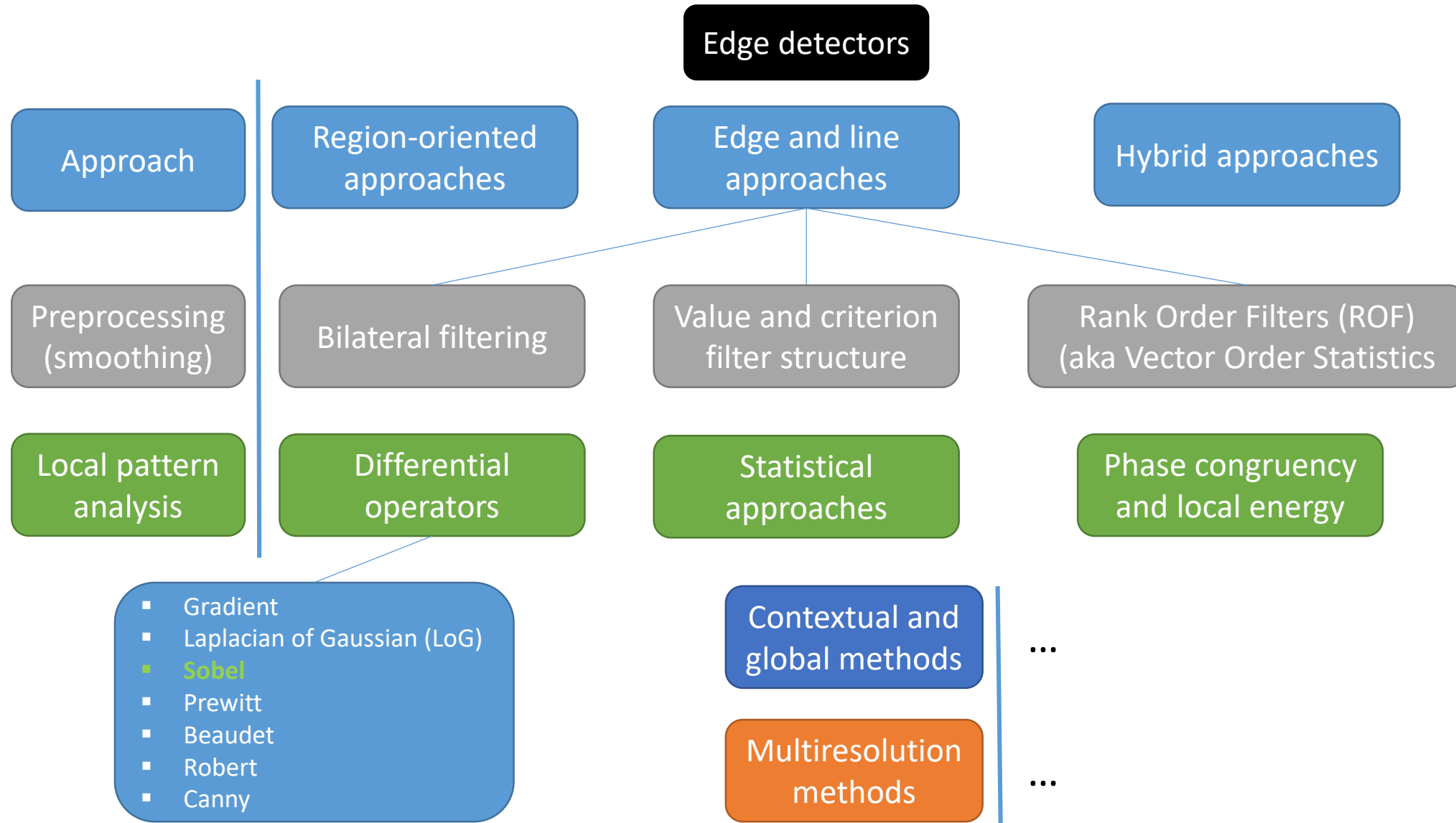


# Computer vision: Edge detectors


- Extracting intensity changes due to
  - geometric events (surface orientation discontinuities, depth discontinuities, color and texture discontinuities)
  - non-geometric events (illumination changes, specularities, shadows, inter-reflections)
- Edge descriptors: direction, strength and position
- Edge criteria: detection, localization and single response
- 3 main operations
  - differentiation : evaluation the desired derivatives of the image
  - smoothing : reduction of the noise and regularizing the numerical differentiation
  - labeling : localizing the edges and increasing SNR by deleting false edges



# Edge detectors - Overview



# Edge detectors - Choice

	Simplicity	Detection of edges and their orientations	Rotation invariant	Sensitivity to noise	Finding correct places of edges	Testing wider area around pixel	Improving SNR, localization and response	Complex computations Time consuming	Malfunction at the corners	Not finding the orientation of edges
Sobel, Prewitt, Kirsch, Robert (First deriv.)										
Zero Crossing (Second deriv.)										
LoG										
Canny										

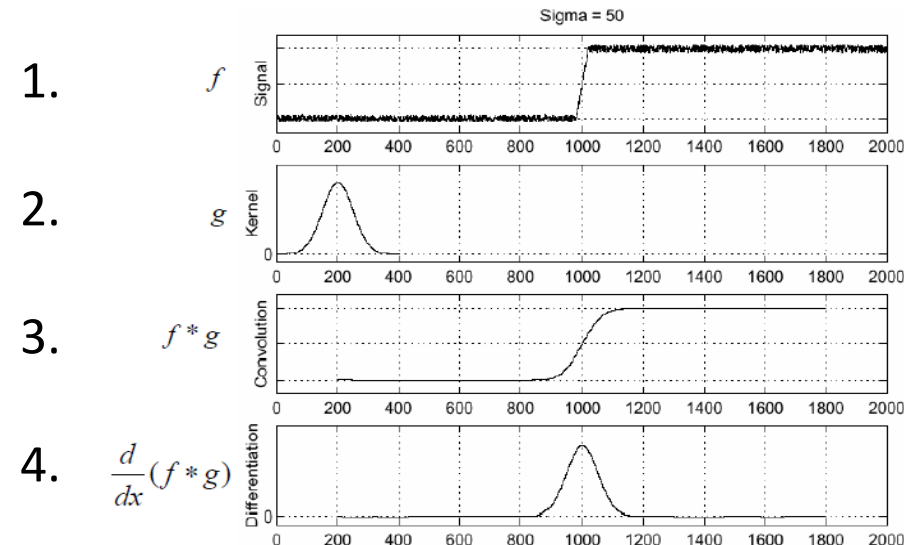
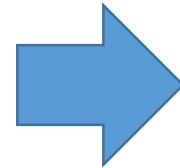
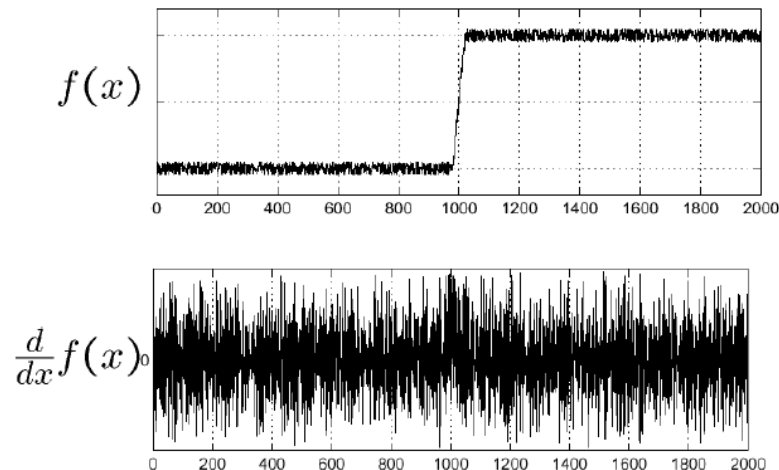
# Sobel's Edge Detector

Prewitt:  $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$  ;  $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel:  $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$  ;  $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts:  $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$  ;  $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

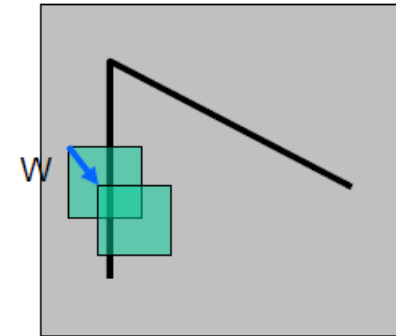
- Sobel is used to find the approximate absolute gradient magnitude at each point in an input grayscale image.
- Sobel kernel different from the others in the sense that it is more suitable to detect edges along the horizontal and vertical axis.



To find edges, look for peaks in  $\frac{d}{dx}(f * g)$

# Computer vision: Corner detector

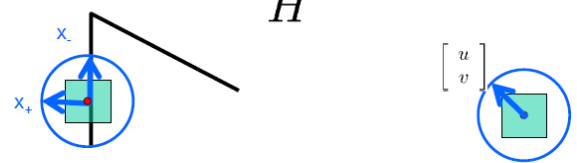
- Corner = junction of two edges, where an edge is a sudden change in image brightness.
- Main reference algorithm is from Harris: intensity change in shifting window: eigenvalue analysis
- Main steps
  1. Color to grayscale
  2. Spatial derivative calculation
  3. Structure tensor setup
  4. Harris response calculation
  5. Non-maximum suppression
- Many Harris-based improved algorithms existing
- Concurrent: Kanade-Lucas-Tomasi operator but Harris provides better repeatability under changing illumination and rotation



# Corner detector: Harris algorithm

## Harris

1. Computing of Gaussian derivatives at each pixel
2. Computing of second moment matrix  $H$  in a Gaussian window around each pixel
3. Computing of corner response function  $R$
4. Threshold  $R$
5. Finding local maxima of response function

$$E(u, v) = \sum_{(x,y) \in W} [u \ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_H \begin{bmatrix} u \\ v \end{bmatrix}$$


Eigenvalues and eigenvectors of  $H$

- Define shifts with the smallest and largest change (E value)
- $x_+$  = direction of largest increase in E.  $Hx_+ = \lambda_+ x_+$
- $\lambda_+$  = amount of increase in direction  $x_+$
- $x_-$  = direction of smallest increase in E.  $Hx_- = \lambda_- x_-$
- $\lambda_-$  = amount of increase in direction  $x_-$

$$R = \det H - k(\text{trace } H)^2$$

With:

$$\det H = \lambda_1 \lambda_2$$

$$\text{trace } H = \lambda_1 + \lambda_2$$