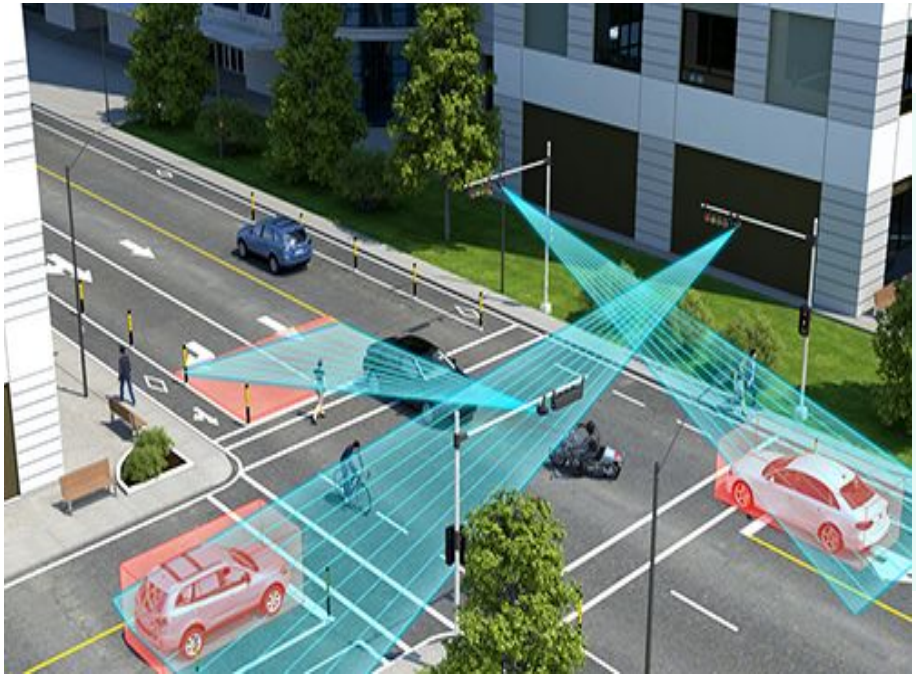
Video Surveillance for traffic monitoring

TEAM 4:

- Alex Vallès Fernández
- Mikel Menta Garde
- Sebastián Maya Hernández
- Pedro Luis Trigueros Mondéjar

Project goal



In this project we created a traffic monitoring system based in computer vision techniques.

We want to perform out the following implementations:

- Segmentation of foreground objects
- Tracking of the cars
- Interpretation of trackings:
  - Speed estimation
  - Car counting

Pipeline

Input video → Video stabilization → Foreground segmentation → Object tracking

Foreground segmentation: *Gaussian adaptive modelling* → *Hole filling* → *Area filtering* → *Specific morphology*

Result

Speed estimation

Vehicle counting

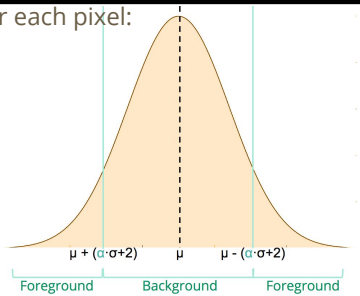
Foreground segmentation

In a first step, the system needs to separate the moving objects from the background.

Video Stabilization: we also have an stabilization method based on a block matching for sequences with high jitter such as *Traffic*.

Gaussian adaptive modelling

For each pixel:



if pixel $i \in$ Background **then**

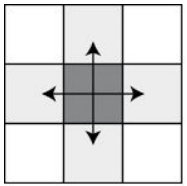
$$\mu_i = \rho \cdot I_i + (1-\rho) \cdot \mu_i$$

$$\sigma_i^2 = \rho \cdot (I_i - \mu_i)^2 + (1-\rho) \cdot \sigma_i^2$$

end if

Hole filling

We perform a hole filling over the mask using *4-connectivity* to fill the foreground areas segmented.



Area filtering

We perform an area filtering to clean the image of noise detections (using number of pixels adapted for each scene).

Specific morphology

We perform some specific morphology to end the cleanup of the objects in each scenario, with the specific steps of:

Pipeline
Closing: *(ellipse)*
Hole filling
Opening

Tracking

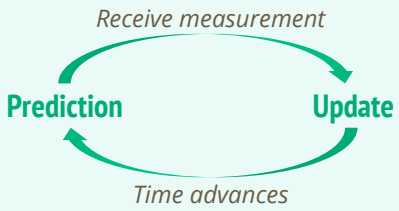
After having segmented the moving objects, the system tracks each of them frame by frame in the video.

Algorithm

For every mask:

1. Get **connected components**
2. For every CC, compute distance to every detected object (from BBox center to **Kalman Prediction**):
   a. Find nearest object inside a given **threshold**
   b. **Update Kalman** filter of the nearest object
   c. If not found, **create new object**

The **Kalman Filter** is based in the following model:



Comparison

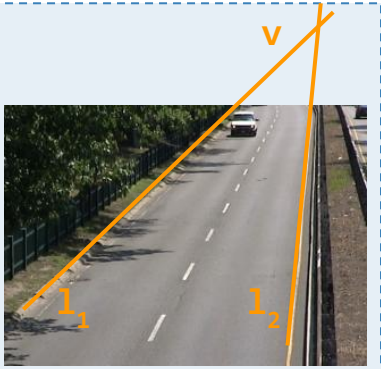
We compared the results with the **Kernel Correlation Filters** algorithm. We see that KCF **performs worse on the tracking** than our results with Kalman (check the car IDs).

The Kernel Correlation Filters uses a gaussian confidence map to determine the position of current frame. If the position is not confidence enough, then Block Mean Shift algorithm is used to get the object again and local features are used to adjust the final position.

Speed estimation

1. Pick 4 points to get two parallel lines `(l1, l2)`
2. Find the **vanishing point** `v` computing the cross product of the parallel lines. $v = l_1 \times l_2$
3. Get homography `H` for an **aerial view** transformation.
4. Hand pick 2 points of a known distance (i.e. discontinuous line)
5. Convert the obtained points using `H`
6. Compute the **scale factor** that relates your image pixels with real meters
7. For every object track:
   a. Get velocity for every pair of positions (transformed with H) using **FPS**
   b. Compute a **weighted average** of the velocities `(w = 0.3)` $vel_k = (vel_k \times w) + (vel_{k-1} \times (1 - w))$

$$H = \begin{pmatrix} 1 & \frac{-v_1}{v_2} & 0 \\ 0 & 1 & 0 \\ 0 & \frac{-1}{v_2} & 1 \end{pmatrix}$$



AERIAL VIEW

Highway

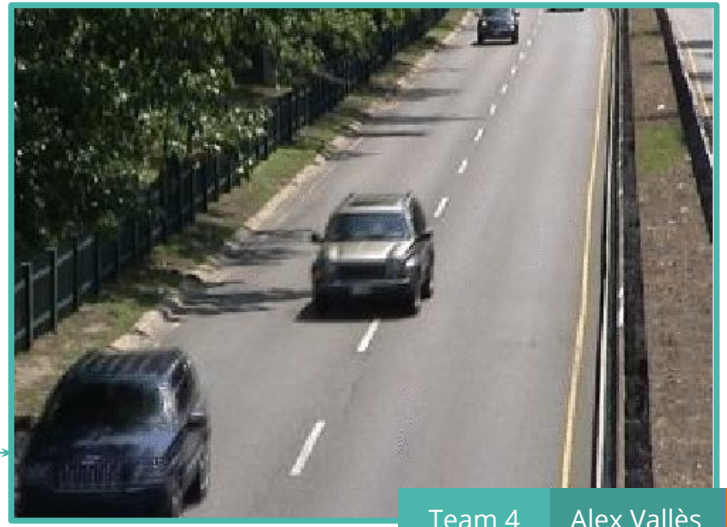
Traffic

Speed estimation

HIGHWAY	
Car ID	Velocity (km/h)
1	90.22
2	107.11
3	95.66
5	89.29
7	79.813
9	80.78
10	82.29
12	81.97

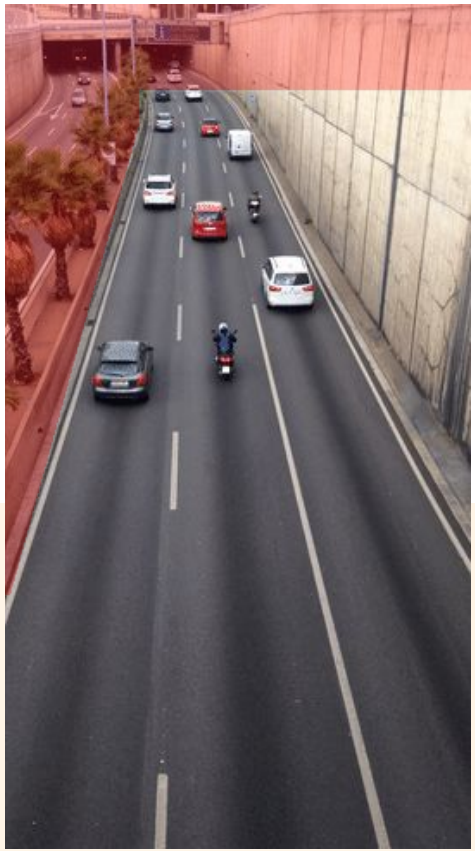
TRAFFIC	
Car ID	Velocity (km/h)
1	59.49
2	65.35

Erroneous tracking threshold: the detected objects that live for less than *5 frames* are not taken into account.



** Dataset from Team 1 of 2017 class*

Our own study



For this study we used this dataset* recorded in the 'Ronda de dalt' of barcelona, because of its:

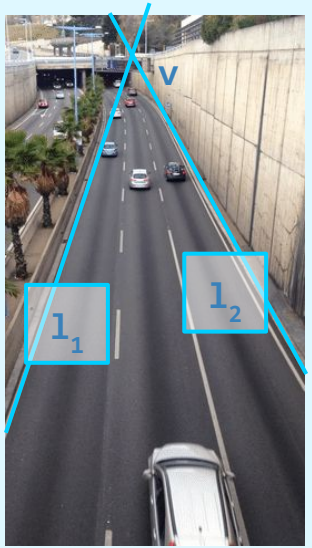
- Centered perspective
- Appropriate height (no occlusions between the cars).

Goals of the study:

- Detect number of cars
- Which ones exceed the speed limit (80km/h).

Region of interest image (ROI) for:

- Discard the cars that go in the opposite direction (to the left of the image)
- Get rid of the noise of the upper part of the image (clouds, leaves, etc.)



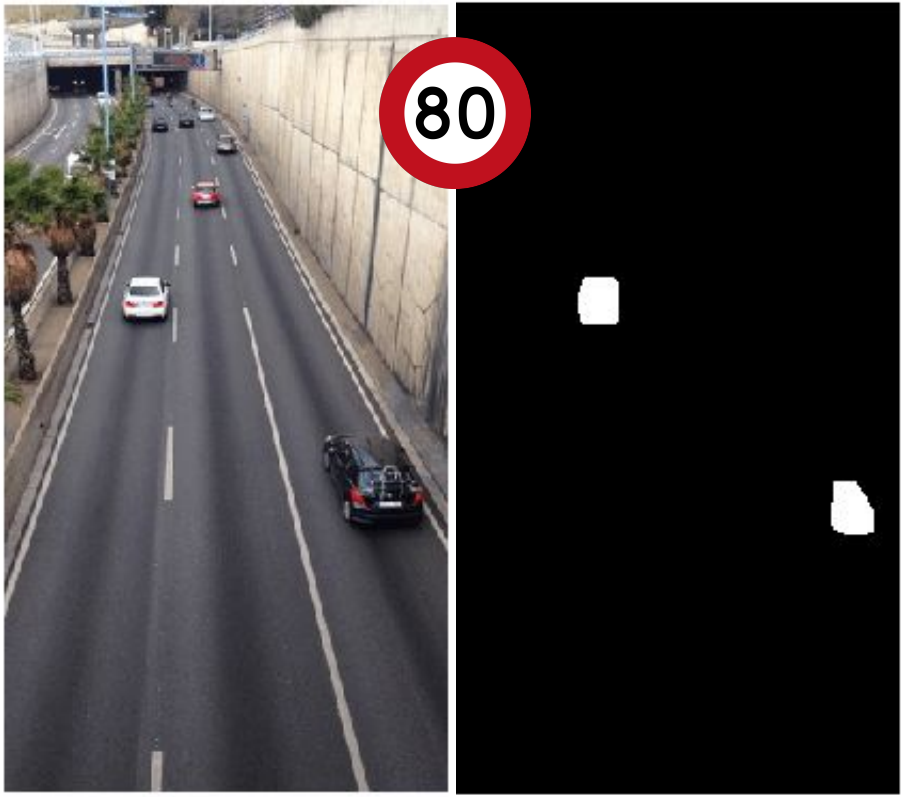
- Kalman filter for tracking
- Aerial view & known distance for speed estimation

Results

COMMENTS:

- Farthest cars are not detected
- Objects with outlier speeds are considered noise.
- In the test sequence of this dataset a total of 8 cars were detected
- 3 of these cars exceed the limit (37.5%)
- Cars in leftmost lanes have higher speeds.

Car ID	Mean Velocity (km/h)
1	84.18
2	68.04
5	95.89
6	69.80
7	61.50
8	73.21
10	115.59
11	79.72



Total number of cars: 8

Conclusions

- Foreground detection depends too much on finding the right parameters for the used dataset
- To make a good detection it is necessary to design a very specific morphology pipeline
- Kalman tracker works very well except when the image has a lot of perspective, then it fails because of the distance threshold
- The speed estimate is quite unstable if you do not do a weighted average to smooth it up a bit.
- If you manage to correctly perform the aerial view transformation the speeds are pretty accurate

THANK YOU!