Kalman Filter for object tracking

Javier Rico[1,2,3,(✉)], Meaza Eyakem[1,2,3,(✉)]

[1] Pázmány Péter Catholic University, Faculty of Information Technology and Bionics, Budapest, Hungary
[2] Autonomous University of Madrid, Madrid, Spain
[3] University of Bordeaux, Bordeaux, France
jvirico@gmail.com,meazaeyakem1@gmail.com

1 INTRODUCTION

Two algorithms for object tracking based on Kalman Filter [3] are implemented using OpenCV C++ library [2]. First, a Constant Velocity Model [6], and second an Acceleration Model. The strengths and weaknesses of both models are discussed using toy and real video sequences. Each tracking experiment can be divided into 3 steps, (1) foreground mask is generated based on Background Subtraction, morphological opening is applied to filter noise from the background mask. (2) the main Blobs are extracted from the filtered mask using Connected Component Analysis (CCA). From the extracted blobs, the center of the biggest blob is used as input measurement to the Kalman filter. (3) with the main Blob as the input for Kalman Filter, Constant Velocity and Acceleration models for Kalman filter are implemented.

2 METHOD

To track an object from a video sequence we can first extract the blob from the foreground mask generated. However, finding the foreground mask using Background Subtraction method encounters problems when the foreground is being occluded by another object. Moreover, frequently the detection process introduces noise. To solve these problems we use Kalman Filter to propagate the state of the extracted blob through time, facilitating object tracking while the object is occluded, computing the estimation of object position based on measurements and predictions. To do so, we use OpenCV MOG2 subtraction method to extract the foreground mask, we apply some morphological operations, then we extract the blobs using Connected Component Analysis, and finally we implement the mentioned Kalman filter models, Constant Velocity and Acceleration Models.

2.1 Blob Extraction

Blob extraction using Connected Component Analysis (CCA) [5] has been implemented after the foreground mask is obtained using OpenCV MOG2 [4]. To reduce the noisy foreground and false blobs produced, we filter the blobs by their size, keeping only the Blob of interest.

2.2 Kalman Tracking

Kalman Filter is used to estimate the position of the object when measurements fail to track its position. The main purpose of this step is to improve the prediction of the target position based on previous measurements and predictions [1]. The prediction is corrected at each step using the information from the measurements when available.

3 IMPLEMENTATION

The C++ project has been organized trying to maintain the Main function as clean as possible, keeping in Main.cpp just the key parameters to set the different scenarios and facilitate the tuning of the final result. Most of the functionality has been encapsulated into classes when possible.

Following, the list of classes and modules that encapsulate the source code:

- **kalman.cpp/.hpp**: Class that contains all the functionality for Kalman Filter.
- **blobs.cpp**: Module with functionality to perform Blob Filtering and Extraction.
- **fgseg.cpp**: Class to control Background Segmentation.
- **ShowManyImages.cpp**: Module with procedures to allow graphical representation of results.
- **main.cpp**: Controls the main parameters for choosing the scene being analyzed, the Background subtraction process, the Blob filtering and extraction, the set-up of Kalman filter, and the visualization of results. Runs the tracking process of an object given a video or sequence of videos.

3.1 main.cpp

The main.cpp file is divided into five sections, Parametrization, Foreground Detection, Blob Extraction, Kalman Tracking and Results Visualization.

We will proceed to explain the Parametrization sub-sections, which is the part that parametrizes the whole process and therefore susceptible to be modified.

- **Dataset**: with the object *oSRC* we control the set of videos we are using by passing a numeric parameter, from 1 to 4 to target the hardcoded datasets, or 0 to use a video passed as an argument of the application.
- **Foreground Segmentation and Blob Analysis**. With the object *oBGS* we set the method to use for Foreground Segmentation, OpenCV MOG2 has been used, and the morphological operations to perform after the segmentation. In this section we can control also the learning rate for MOG2, the size of the structural element used for Connected Component Analysis, and the minimum blob size to filter small blobs caused by noise.

– **Kalman Initialization**: with the object *oKF* we can control the configuration of Kalman Filter, choosing between Constant Velocity or Acceleration Models. The flag *bFullDebugOn* activates detailed debug messaging from the Kalman process. However, any modifications to Kalman internal Matrices should be done in the section *KALMAN TUNING*, like for example, the Uncertainty Matrix (R).

3.2 Running the code

Use the makefile provided to compile the project. We have implemented oSRC object to ease the experimentation when several datasets and videos are used, by passing a parameter from 1 to 4 to target each of the videos. The URLs to each of the videos used for the present report are specified in **vsrc.hpp**. We set the default value of this parameter to 0 so the application can accept a path of an external video as an input argument when calling **main.cpp** from a command line.

In summary, one can run the code by simply passing the video path as an input argument, or by changing the parameter to get the path from the hard coded paths and simply running the application.

4 DATA

Two datasets from CDNET data-bank [7] are used in the following experiments. We name these two data sets the **Toy data set** and the **Real data set**. Both data sets contain 4 video sequences each with its own tracking challenges. The details of each experiment and the video sequences are discussed in **Results and Analysis** section.

5 RESULTS AND ANALYSIS

The following experiments are analyzed:

– **Experiment 1**: Analysis of how the two Kalman Models implemented handle the video "singleball.mp4", provided to implement and validate the implementation before further analysis of Toy and Real data datasets.
– **Experiment 2**: Analysis of Toy data series, 4 videos, applying Kalman Acceleration Model. We describe the required parameter tuning to fit the Tracking process.
– **Experiment 3**: Analysis of Real data series, 4 videos, applying again Kalman Acceleration Model. We describe the required parameter tuning to fit the Tracking process.

5.1 Experiment 1: Constant Velocity and Acceleration Models

Constant Velocity Model For this model we consider constant velocity, and therefore we can see how the tracking of the ball looses the object when it is accelerating or decelerating.

$$x_k = [x, x', y, y']$$

$$z_k = [x, y]$$

This is clearly observed when the ball is occluded and decelerating a the same time, resulting in a clear offset between prediction and real trajectory that is noticeable when the ball becomes visible again, see Fig. 1c.

Acceleration Model Now we consider acceleration, and therefore we should observe an improvement compared to constant velocity model, specially in those moments when the prediction was missing the ball due to acceleration or deceleration.

$$x_k = [x, x', x'', y, y', y'']$$

$$z_k = [x, y]$$

When we analyze closely the video sequence 'singleball.mp4', we can see how considering acceleration corrects some of the offsets mentioned before, see Fig. 2c.

At the beginning of both models we observe the same offset (see Fig. 1a and Fig. 2a), this is due to the necessity of the Kalman algorithm to build up precision by obtaining several measurements, the model will produce a stable tracking as long as the object keeps a linear trajectory, and velocity and acceleration do not change abruptly, when this happens the Kalman process needs to re-adjust again.

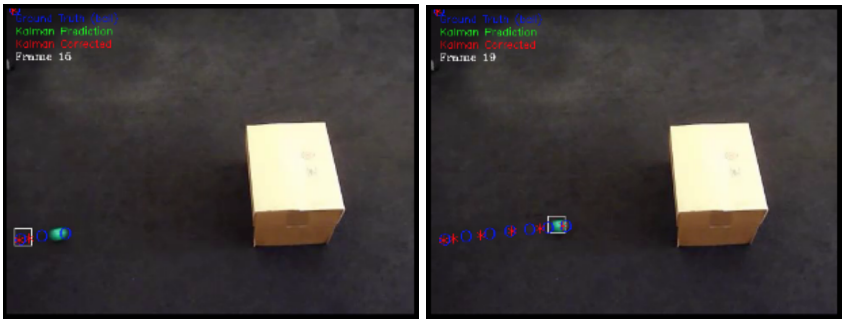
Note on Kalman Filter parameters We have experimented different setting of Kalman Filter, changing the blending factors, the measurement error co-variance, and prediction error co-variance matrices.

We noticed visible differences when using small versus big values of the Uncertainty Matrix (R), since it controls how much we trust the measurements of the system, Blob Extraction measurements in our context, therefore it rewards or penalizes the compromise between prediction and measurement.

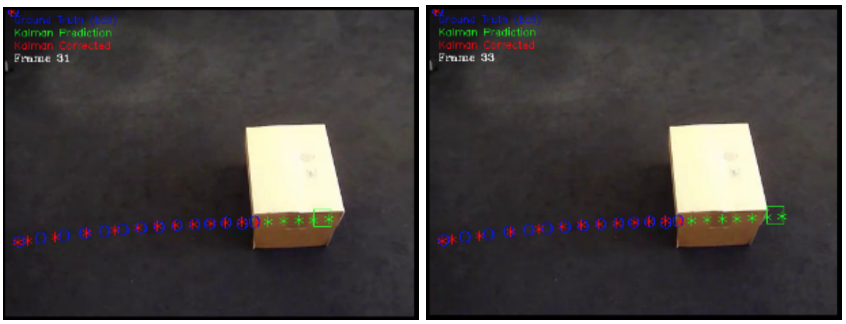
We can see the result of increasing or decreasing the measurements covariance matrix (R) in Fig. 3 (Frame 16 in both images):

5.2 Experiment 2: Toy data series

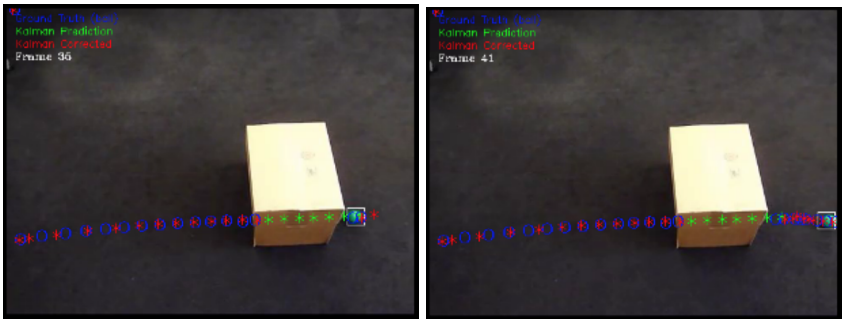
In this experiment we will tune the parameters for the Toy video sequences that consists on a moving tennis ball in four different scenarios.



(a) Ball enters the scene accelerating (*Frame 16, left image*). The tracking process needs several updates to build up precision (*Frame 19, right image*).



(b) The ball is occluded and decelerating (*Frame 31, left*), and the tracking prediction surpasses the real ball trajectory due to constant velocity considered (*Frame 33, right*).



(c) The ball enters the scene again, we can observe the offset here (*Frame 36, left*). The tracking process updates with the new measurements. From this point, the tracking process updates well when the velocity is near constant (*Frame 41, right*).

Fig. 1: Experiment 1 - Constant Velocity Model

Video 1 The first scenario is the simplest of them all, see Fig. 4. On one hand, its characteristics allow for a good Background Subtraction, and no other moving objects are present in the scene, therefore the Blob extraction process doesn't

have much problems identifying the target ball. Further to this, no occlusions are present. On the other hand, the scene presents a textured background with a fast motion of the ball.

As we can see in Fig. 4, the Kalman tracks the object smoothly starting from the entrance of the ball in the scene up to the end. Acceleration Model for Kalman with the same configuration as Experiment 1 is used. Low Learning Rate (*0.0001*) for Background Subtraction. And customization of Blob Extraction, were blob size and shape are adjusted (*min-w = 15, min-h = 15*).

Video 2 The second video sequence presents a challenging Background Subtraction, and OpenCV MOG2 fails to generate a sufficient background/foreground differentiation, see Fig. 5. Therefore Blob Extraction can not identify the ball and Kalman Filter can not track it.

Video 3 In this scenario, where the tennis ball is bouncing up and down, the main problem is presented by the horizontal cornice, see Fig. 6, that at some moments can be identified as the biggest Blob in the image, confusing the tracking process.

We solve this problem by constraining the shape and size of the Blobs to rectangular shapes close to the ball size (width and height between 55 and 250). Additionally, we increase the Morphological operations (opening) to help reduce the fogy foreground mask.

Video 4 The last Toy video sequence to analyse has a moving ball on a fixed background, see Fig. 8. The ball gets occluded by a fixed box during part of the sequence. Here, we do not encounter obvious problems related to Background Subtraction or Blob Extraction.

5.3 Experiment 3: Real data series

In this experiment we will be tuning the parameters of the system to adjust to 4 Real video sequences, see Fig. 9.

Cyclist scene The first scene to analyze, file *"abandonedBoxclip.mp4"*, consists on a junction where a cyclist enters the scene and stops at the crosswalk.

The blob extraction here is unstable, and the fact that the object to track stops and remains static for much of the sequence brings two challenges: (1) Blob is constantly shaking and making short position variations, this makes difficult for Kalman to adjust parameters to keep track of the smooth trajectory of the cyclist in case of occlusions or the temporal disappearance of the blob; (2) the blob extraction must consider that the object stops for long periods, remaining static, without fusing it with the background, for this we could increase the history buffer of the background subtraction part, also decreasing learning rate.

To adjust our system to these 2 challenges, see Fig. 10, we have set the parameters as follows. Use of Acceleration Model for Kalman, with the same configuration as for Experiment 1. We have decreased the Learning Rate (*0.0001*) of Background Subtraction, trying to avoid the problem (2) mentioned before. Regarding challenge (1), we have customized the minimum size and shape of a Blob (*min-w = 15, min-h = 15*).

Sailboat scene The second scene to analyze, file *"boatsclip.mp4"*, consists on a sailboat crossing a lake from one side to the other and coming back, see Fig. 11.

The main challenge this scene presents is during those moments when the ship is in a position relative to the camera where the ship's sail becomes less visible, and blob extraction fails identifying the object for some instants.

The settings of the system for this scenario are: Acceleration Model for Kalman, using the same configuration as Experiment 1. We have maintained a low Learning Rate (0.0001) for Background Subtraction. And we have customized the minimum size and shape of a Blob (min-w = 30, min-h = 65) to avoid some wrong detection that would appear from other objects moving on above the lake, trying to stabilize Blob Extraction, see Fig. 12.

Pedestrians scene The third scene to analyze, file *"pedes triansclip.mp4"*, consists on a pedestrian walking on a park.

We have found two aspects of the scene to be particularly problematic, (1) the pedestrian walks from right to left with small up and down oscillations with each step that make more difficult to the Kalman process to adjust the trajectory of the object. And (2), Background Subtraction process is failing to remove the shadow of the pedestrian, see Fig. 14a, making Blob Extraction to believe there is another object (similar human shape) crossing the scene form some configurations.

The strategy followed here is very similar to the configuration for the Sailboat scene: Acceleration Model for Kalman with the same configuration as Experiment 1. Low Learning Rate (*0.0001*) for Background Subtraction. And customization of Blob Extraction to avoid confusing the pedestrian shadow (2) with the object of interest, blob size and shape adjusted (min_w = 25, min_h = 55), see Fig. 14b.

Highway scene The last scene, file *"streetCornerAtNightclip.mp4"*, presents a car crossing a road at high speed during night.

This scene has the particularity that car lights cause confusion on Blob Extraction, probably because Connected Component Analysis process is not capable of differentiating between the car and the road due to the change in luminosity that the car lights produce in from of it, resulting in a Blob much bigger than it should be if only the car was considered.

No special settings have been applied on this scene, the tracking process follows the car, but a small offset is produced due to that the excessively big

Blob has a center that doesn't correspond with the exact center of the car, see Fig. 15.

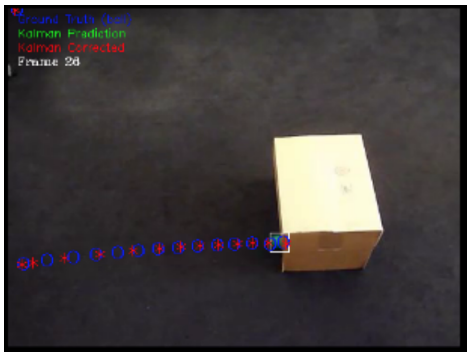
6 CONCLUSIONS

Using Kalman Filter has shown to be an effective tracking method in many scenarios, specially for those where the measurement and state transition models are linear, or close to linear. Acceleration Model has demonstrated to outperform Constant Velocity model in the here experiments. In our context, depending on a measurement source that is in many cases noisy, the capacity of Kalman to tolerate certain level of uncertainty in the measurements is a powerful capability that allows it to succeed in many tracking situations. However, long periods of missing measurements, relying entirely on predictions, can make Kalman kinematic model to accumulate errors that may end up loosing the object being tracked. The fact that Kalman presents more problems with non-linear dynamics is evident as the bounding ball shown in Video 3 (Toy data series). To conclude, it is very clear that the reliability of the measurements (Blob extractor in our context) is key for a fast convergence, specially at the beginning of tracking, and also for Kalman to cope with occlusions and blind periods of navigation.

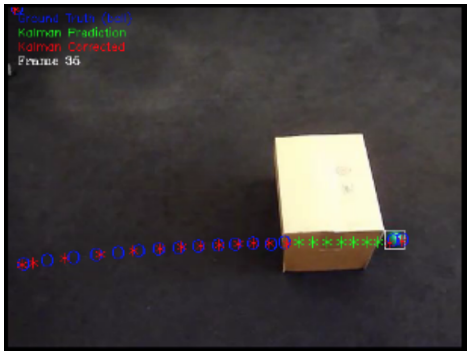
References

1. Funk, N.: A study of the kalman filter applied to visual tracking. University of Alberta, Project for CMPUT **652**(6) (2003)
2. Kaehler, A., Bradski, G.: Learning OpenCV 3: computer vision in C++ with the OpenCV library. " O'Reilly Media, Inc." (2016)
3. Kalman, R.E.: A new approach to linear filtering and prediction problems (1960)
4. Marcomini, L., Cunha, A.: A comparison between background modelling methods for vehicle segmentation in highway traffic videos. arXiv preprint arXiv:1810.02835 (2018)
5. Rosenfeld, A., Pfaltz, J.L.: Sequential operations in digital picture processing. Journal of the ACM (JACM) **13**(4), 471–494 (1966)
6. Schöller, C., Aravantinos, V., Lay, F., Knoll, A.: What the constant velocity model can teach us about pedestrian motion prediction. IEEE Robotics and Automation Letters **5**(2), 1696–1703 (2020)
7. Wang, Y., Jodoin, P.M., Porikli, F., Konrad, J., Benezeth, Y., Ishwar, P.: Cdnet 2014: An expanded change detection benchmark dataset. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pp. 387–394 (2014)

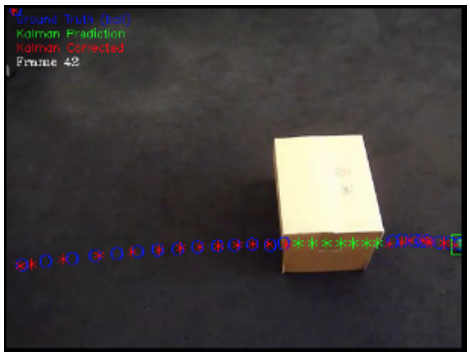
(a) *Frame 16*. Ball enters the scene and is detected.



(b) *Frame 26*. The model needs several measurements to build up precision. Kalman has adjusted to the velocity and acceleration before the ball enters the blind zone.



(c) *Frame 36*. Tracks correctly considering decelerating. Acceleration model behaves well in this scenario.



(d) *Frame 42*. Since no abrupt changes in trajectory or acceleration, the model tracks with precision until the end.

Fig. 2: Experiment 1 - Acceleration Model

(a) *Frame 16*. Higher values of Co-variance Matrix (R) make Kalman to trust less in the measurements and therefore rely more in predictions. Taking more time at the beginning to update its internal kinetic model.

(b) *Frame 16*. Lower values of Uncertainty matrix (R) help Kalman to put less trust in the prediction and update faster with the measurements. Its kinetic model adjusts to the motion much faster than in the previous scenario.

Fig. 3: Uncertainty Matrix (R). R is a diagonal matrix with low and high values of 25 and 100 respectively.

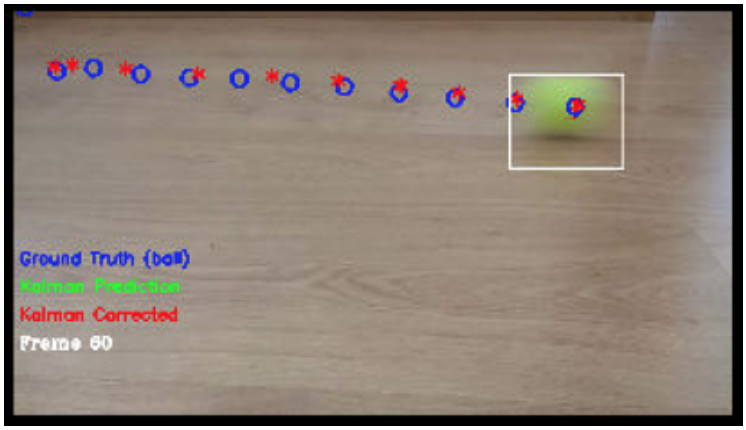


Fig. 4: *Frame 60*. Acceleration Model. Kalman keeps accurate tracking of the ball.

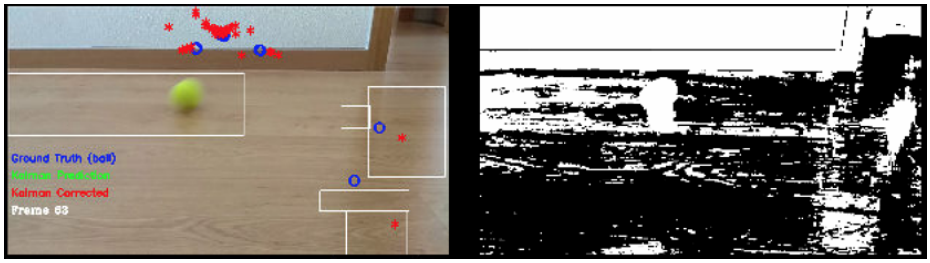
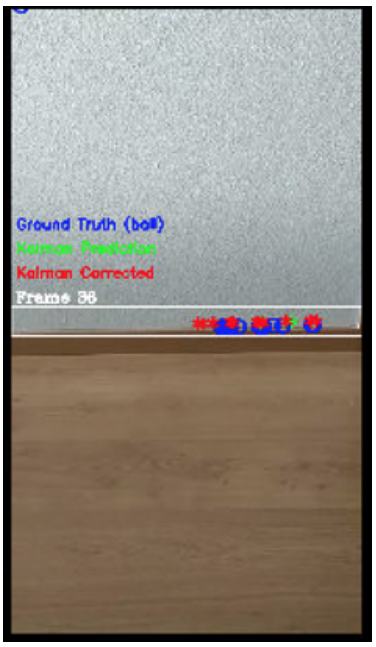
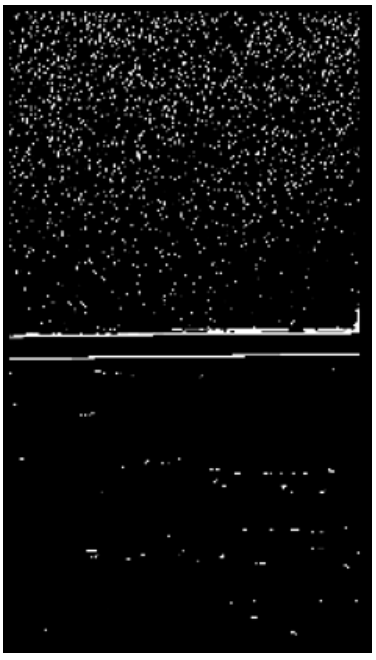


Fig. 5: *Frame 63*. Backgound Subtraction method (*OpenCV MOG2*) fails to generate a sufficient background/foreground differentiation.



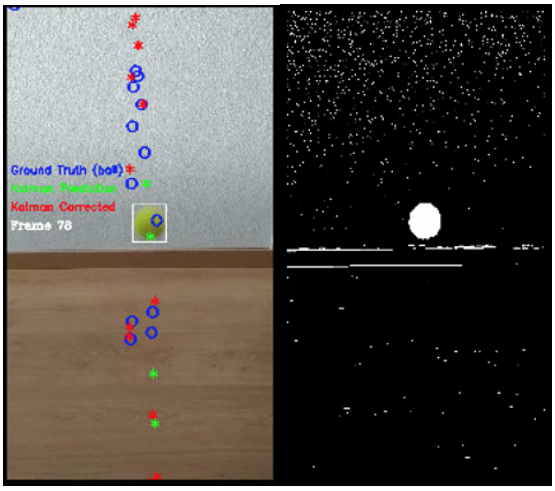
(a) Blob extraction.



(b) Foreground mask.

Fig. 6: *Frame 36*. Artifacts present due to incorrect Blob Extraction/filtering.

(a) *Frames 62, 69, and 109*. The new Blob Extraction constrains eliminate the cornice issue. Notice that the moments when the ball is not detected, Kalman tries to predict the next position, not being always accurate due to the fast changes in acceleration and direction, see right image. This is particularly visible if we take close look at how much vertically spread are the Kalman measurements in comparison to the ground truth measurements.



(b) *Frame 78 and its foreground mask*. Notice that the moments when the ball is not detected (*green asterisks*), Kalman tries to predict the next position, not being always accurate due to the fast changes in acceleration and direction. This is particularly visible if we take close look at how much vertically spreaded are the Kalman measurements in comparison to the ground truth measurements (*blue circles*).

Fig. 7: Experiment 2. Video 2, with constrained blob size and extra morphological operations applied to foreground mask.

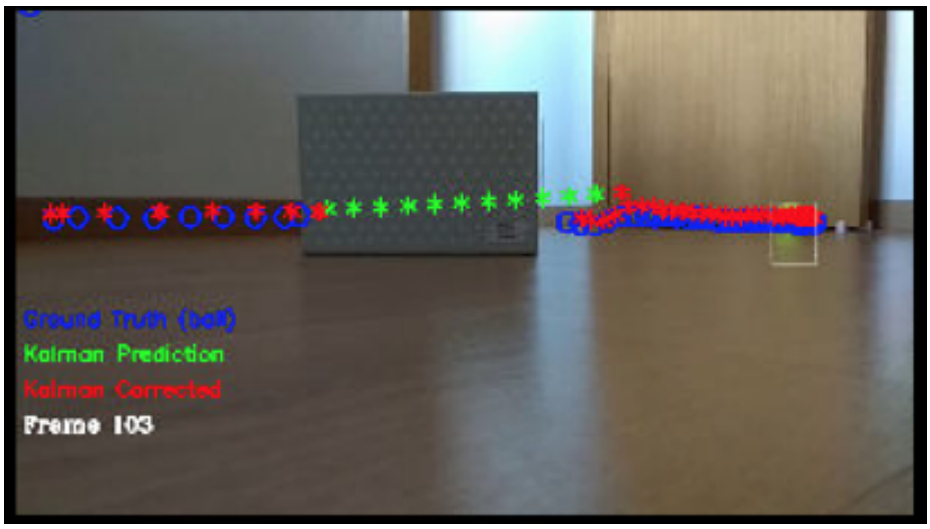
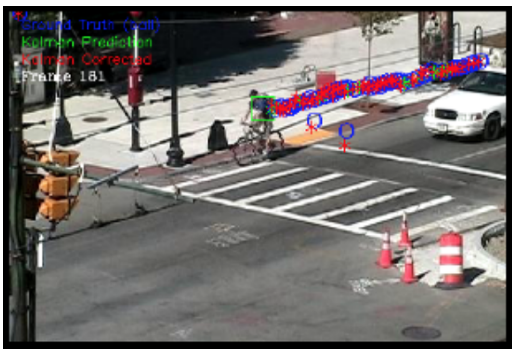


Fig. 8: *Frame 103*. The detection goes smoothly, with a small offset in x and y axis of the image that gets corrected as soon as the ball exists the blind zone.

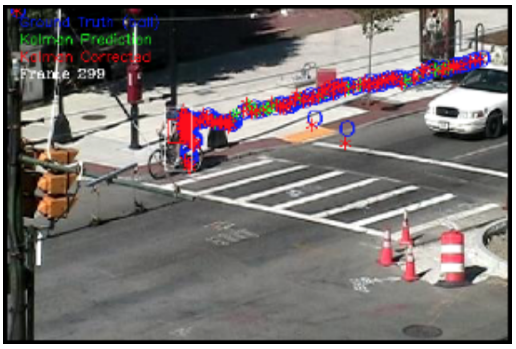
Fig. 9: *Cyclist scene*

(a) *Frame 129*. The tracking process follows the object that enters the scene smoothly.



(b) *Frame 181*. Kalman tracking process fills correctly the moments when the object is not detected (see green marks).

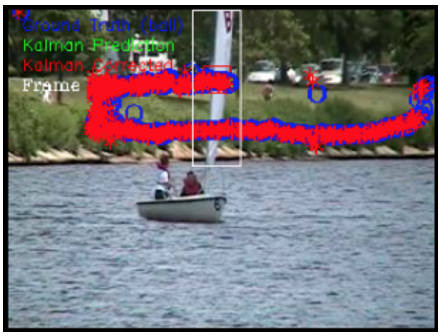
(c) *Frame 233*. Cyclist stops at the crosswalk.



(d) *Frame 299*. The Blob extraction doesn't loose the static object for as long as it is recorded (+60 frames).

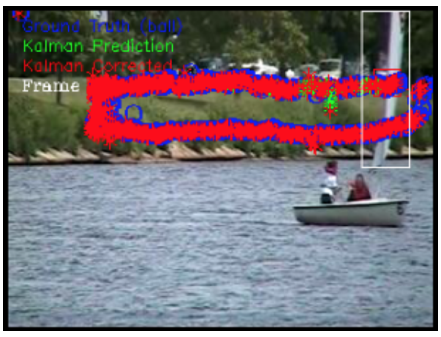
Fig. 10: Experiment 3. Cyclist scene.

Fig. 11: *Sailboat scene*



(a) *Frame 760*. The tracking follows the boat's sail without any problem for most of the video.

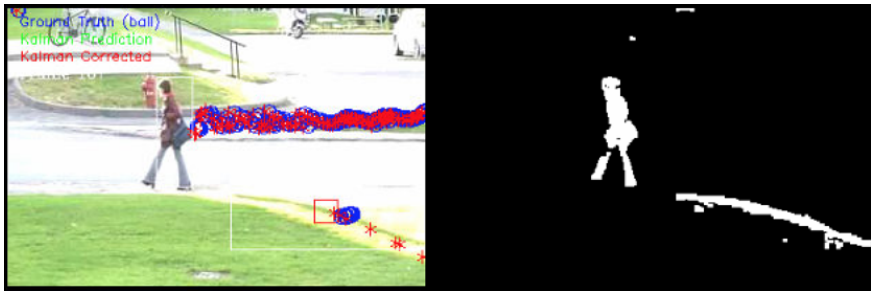
(b) *Frame 870*. At some moments the Blob Extraction fails to locate the sail but Kalman predicts it with small errors.



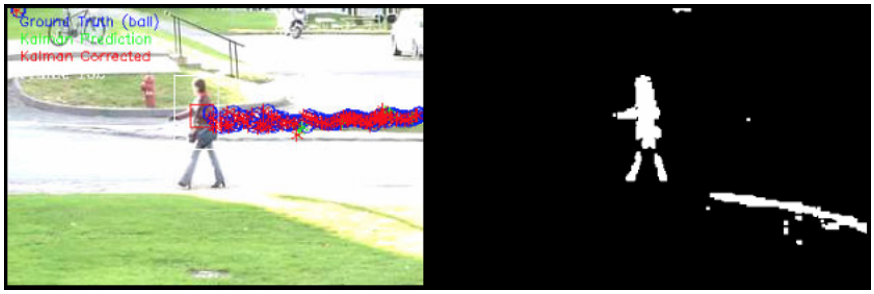
(c) *Frame 945*. The trajectory is completed showing a couple of seconds of small prediction errors due to consecutive Blob Extraction fails.

Fig. 12: Experiment 3. Sailboat scene.

Fig. 13: *Pedestrian scene*



(a) *Frame 101*. Background Subtraction failing to suppress the shadow, causing the algorithm to track the wrong blob (*shadow blob is the biggest*).



(b) *Frame 102*. Blob shape restriction solves the shadow issue.

(c) *Frame 220*. Object tracking with final setting.

Fig. 14: Experiment 3. Pedestrian scene.

Fig. 15: *Highway scene*