LAB-3 Kalman filtering for Object tracking

Kush Gupta and Sergio Avello Largo

I. INTRODUCTION

This lab report contains routines for foreground extraction, blob extraction and Kalman filtering for the given different video sequences. The code was written in C++ using the OpenCV library and designed on Ubuntu using Eclipse IDE. All the tasks performed on the provided videos, were captured from a still camera and those videos belong to the AVSA Lab 3 dataset. The algorithm relies on the foreground segmentation mask obtained for each frame with the MOG2 [1] background subtraction module provided by OpenCV. Firstly, the foreground mask is obtained for each frame of the video sequence, and then, a blob for the object that we are interested in is extracted using the connected component analysis from the foreground segmentation mask. Later, the coordinates of the extracted blob's center is used as an observation in the Kalman filter[2] algorithm for computing the trajectory of the object.

II. METHOD AND IMPLEMENTATION

A. *Foreground Segmentation*

Initially, for the provided video sequence we created a copy of the current frame and then converted it into gray scale. After that, we used an open cv function gaussian blur with a kernel size of 7X7 in order to remove noise from the image. Then, the foreground mask(fgmask) was obtained using the Opencv's MOG2 background subtraction method which segments the foreground and the background. The MOG2 method performs background subtraction using the approach of gaussian mixture model (GMM) and provides a foreground mask (fgmask) where the values 255, 127 and 0 correspond to foreground, shadow and background, respectively. In order to remove the shadow from the fgmask we thresholded the fgmask with a value of 250. To enhance the target object we used morphological opening of the image (erosion followed by dilation) using a structuring element of size 3x3. Morphological opening is used for removing small objects and thin lines from the fgmask, while preserving the shape and size of target object in the image.

B. *Blob Extraction*

For blob extraction we wrote a extractBlob function. In the extractBlobs function we used the foreground mask and the OpenCV's built-in function floodFill, in order to apply the sequential Grass-Fire algorithm for each of the foreground pixels. This algorithm goes through the whole input foreground mask from the top left corner to the bottom right corner and acts when a foreground (255) pixel is found: it firstly labels the pixel in the output image and secondly sets to zero that same pixel in the input image. Then, the neighbours of this "burnt" pixel are checked (4 or 8, depending on the parameters given for the connected components analysis), and any of those neighboring pixels being also foreground pixels will be treated in the same way (they are labeled in the output image and burnt in the input image), and so on. When no more neighboring foreground pixels can be found, the label is incremented and the normal scanning (from top left to bottom right) of the new modified foreground mask resumes, until the next foreground pixel is encountered, labeled, burnt, and its neighbours are investigated.
The Grass-Fire algorithm results in coordinates of the left corner and the width and length of all the blobs containing the objects existing in the frame. Moreover, we implemented the removeSmallBlobs function to remove all blobs whose area was smaller than (MIN_WIDTH X MIN_HEIGHT). This function was controlled by the two parameters, namely MIN_WIDTH and MIN_HEIGHT. Finally, using the output of the Grass-Fire algorithm, the center of the biggest blob in the frame is computed.

C. *Kalman Filter*

The Kalman Filter is an algorithm that generates an estimation of one or more variable(s) based on a set of measurements. It has two main stages: Prediction (using the previous state) and Correction (using the observations).
In order to implement this algorithm in our code, the OpenCV KalmanFilter class was referred. That allowed us to fill the different matrices that rule the process in a very efficient way and then use its member functions predict() and correct() to complete the Prediction and Correction stages. Our Kalman Filter approach has been designed in order to run in two different modes: 'cteVelocity' and 'cteAcceleration' just with changing one line of the code. This implies that for each mode the matrices are different.

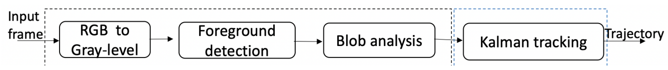

Fig. 1: The figure above shows the block diagram of the pipeline used to implement object tracking using Kalman filter.

III. CODE DESCRIPTION

In the submitted zip file there are 3 folders (T31, T32,T33) with the corresponding sources files for each task. Inside the src folder of each task, the file with the word main in its name acts as the main file for the project, where the different parameters can be modified.

The main file of T31 shows the results of merging task 3.1a and 3.1b without changing any parameter for the video 'singleball.mp4'. For T32, running the main file shows the obtained results for the video 'video2.mp4' with the best parameter configuration. For T33, after running the main file the results for the video 'abandonedBox_600_1000_clip.mp4' with the best parameter configuration can be seen.

A. Running the code

In order to compile, link and run each task, it is only needed to run make in the terminal inside the src folder of each task and then run ./(built object).

IV. DATA DESCRIPTION

A. LAB3.1

For task 3.1, only the one video 'singleball.mp4' was provided. It contained 45 frames. The video was captured from a still camera. In the video a ball moves from left to right. The ball was occluded by a box from frame 27 to 33.

B. LAB3.2

For task 3.2, four video sequences were provided namely video2.mp4, video3.mp4, video5.mp4 and video6.mp4 and they had 111,128,114 and 131 frames respectively. All the video sequences were captured from a still camera and all the videos had noise in them. The video sequences 2,3,5,6 had different motion patterns of the ball to be tracked using the Kalman filtering algorithm. In the 2^{nd} video the green ball moves from top left to right and in the 3^{rd} video initially, the green ball moves towards the wall and then it bounce backwards. In the 5^{th} video the green ball bounces very quickly. In the 6^{th} video the green ball goes from left to right, and gets occluded for some frames by a box and it finally stops after bouncing backwards from the door.

C. LAB3.3

For this task, there were also four videos. However, they were quite different compared with those provided for the previous two tasks. abandonedBox_600_1000_clip.mp4 (401 frames) and pedestrians_800_1025_clip.mp4 (226 frames) show a simple scenario with static background, there is only one object (bicycle or woman) moving during the whole video. On the other hand, the video boats_6950_7900_clip.mp4 (951 frames) is really complex, with a lot of different moving objects in the upper part of the background, the moving water in the lower part of the background and a boat as the main object of the video. A special case is the video streetCornerAtNight_0_100_clip.mp4 (90 frames), because that was recorded using gray scale format.

V. RESULTS AND ANALYSIS

A. Task 3.1

Figure 2 shows the result obtained after merging tasks 3.1a and 3.1b without changing any parameters and applying it to 'singleball.mp4'. The configuration is the following: mode='cteVelocity', learning_rate=0.01, cv::getStructuringElement(cv::MORPH_RECT, cv::Size(3,3)), MIN_WIDTH=10 and MIN_HEIGHT=10.

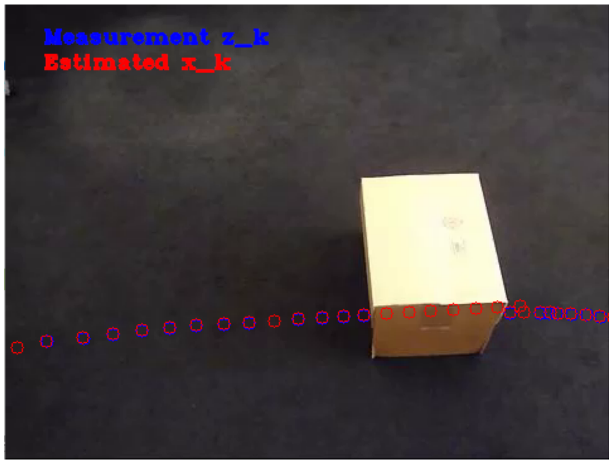

Fig. 2: Result for task 3.1

B. Task 3.2

In task 3.1, we used a value of 10 for both MIN_WIDTH and MIN_HEIGHT as the ball to be tracked was small. However, for task 3.2 the object (ball) to be tracked was bigger in size compared to the one in task 3.1. So, MIN_WIDTH and MIN_HEIGHT were set to 50 each. Also, different sizes of structuring elements were tried: 3x3 5x5 (as the object to be tracked was bigger than earlier), to find out the variations in the tracking results. For the purpose of tracking, we used two modes: 'cteVelocity' and 'cteAcceleration' to analyze which model yields best results for the given size of structuring element and the measurement covariance noise.

Run	Model	Structuring element	Measurement cov noise
1^{st}	Constant Velocity	3x3	20
2^{nd}	Constant Velocity	3x3	22
3^{rd}	Constant Acceleration	5x5	22
4^{th}	Constant Acceleration	5x5	24
5^{th}	Constant Velocity	5x5	24

TABLE I: The table showing the different set of parameters used during the 5 runs to analyze all the videos in task 3.2, in order to visually determine which parameters yields the best results.

During the 1^{st} run and the 2^{nd} run, we used constant velocity model with a structuring element of 3x3 size and the constant for the measurement covariance noise was set to 20 and 22 respectively. The tracking results for the 2^{nd} video was accurate and for 5^{th} video, result was reasonable but for the videos 3^{rd} 3 and 6^{th} 4 there were some wrong tracking predictions due to noise in the foreground mask.

For the 3^{rd} and the 4^{th} run the mode was changed to 'cteAcceleration', the structuring element was increased to a 5x5 size and the constant for measurement covariance noise was set to 22 and 24 respectively. It can be observed that the

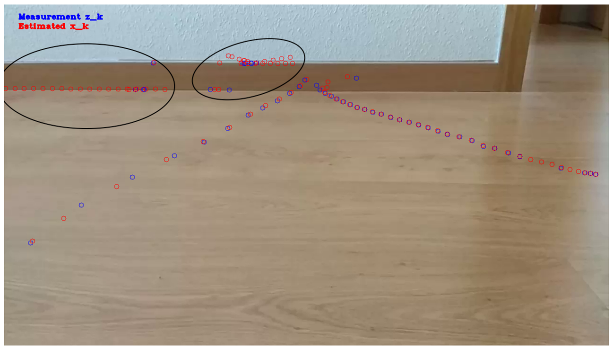

Fig. 3: Result for 1st run, for the 3rd video. The predictions marked in black circle were wrong due to the noise in the foreground mask

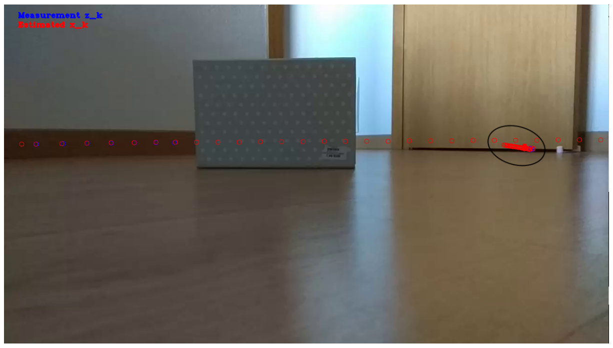

Fig. 4: Result showing tracking for 6th video sequence during the 2nd run. [In right of image] Since the ball was not properly extracted in the foreground mask, we observed missed detection's marked with black circle.

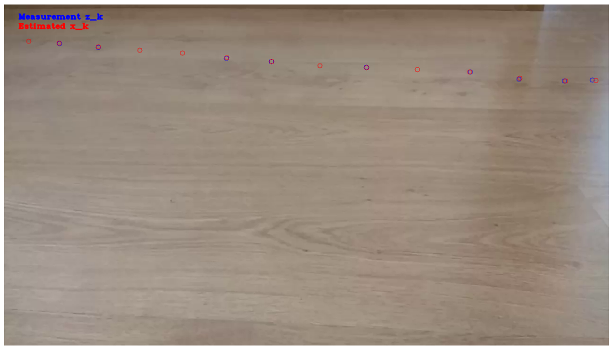

Fig. 5: Result for 4th run, for the 2nd video, as it can be seen in the figure the model correctly tracks the target object (green ball). This configuration yields the best result for the task 3.2, 2nd video.

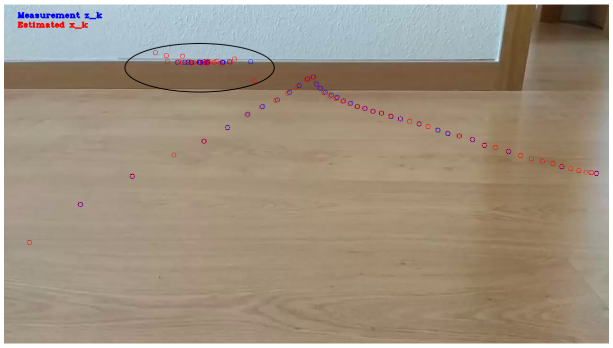

Fig. 6: Result showing a little improvement in tracking for 3rd video sequence at the 3rd run. [In top left of image] we observed mis-tracking of the object due to some noise still present in the foreground mask(marked in black circle).

results for 2nd 5 and 5th videos slightly improved and the results for 3rd video 6 improved a little but it still had wrong predictions due to the presence of noise in foreground mask. In the 6th video 7, as the ball was not fully detected in the foreground mask, the wrong detection was still there. For task 3.2, 2nd video, the best parameter configuration is the following: mode='cteaVelocity', learning_rate=0.001, cv::getStructuringElement(cv::MORPH_RECT, cv::Size(5,5)), MIN_WIDTH=50 and MIN_HEIGHT=50 and MEASNOISECOV_DEF=24 5.

For the 5th run we used 'cteVelocity' mode with a structuring element of 5x5 size and the constant for measurement covariance noise was 24. During the run, we did not observed much improvement in tracking results when compared to the last two runs, for any of the provided video sequences in task 3.2.

C. Task 3.3

First of all, an evaluation on the videos for this task was performed with the same parameter configuration of task 3.1 just to see how it looked like. The trajectory prediction and the measurements are fairly reasonable for the simple videos, however, regarding the more complex videos the outcome was not accurate at all. Figure 8 shows these results.

The first idea to improve these results was just to change the mode of the Kalman Filter from 'cteVelocity' to 'cteAcceleration'. Since the videos are quite complex, we decided to add more complexity to the computation of the predictions and corrections but still keeping fixed (like in task 3.1) the remaining parameters. As we can see in the figure 9 the results slightly improved in terms of correlation between predictions and measurements, nevertheless, they are far from what we expected, so we determined to adjust the parameters based on the characteristics of each sequence.

1) abandonedBox_600_1000_clip.mp4: The main problem of this video appears in the first frame because there is measurement in the top left corner that determined the performance of the algorithm. Therefore, the blob extraction should be improved. After some tuning the best result is shown on Figure 10 with the following configuration: mode='cteAcceleration', learning_rate=0.01, cv::getStructuringElement(cv::MORPH_RECT, cv::Size(4,5)), MIN_WIDTH=9 and MIN_HEIGHT=11.

2) boats_6950_7900_clip.mp4: This video is really complex as it was explained in the data description section. We tried to modify the measurements extraction to improve the results. After some tuning the best

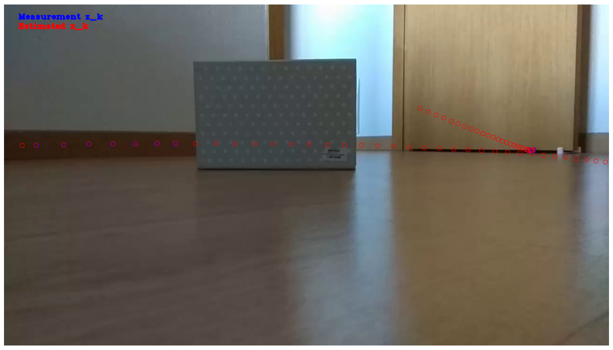

Fig. 7: For the 6^{th} video sequence at the 4^{th} run. We observed mis-tracking of the object(ball) once the ball strikes backwards as the ball is not fully detected in the foreground mask. Hence the model favours prediction more than the measurements due to absence of measurements.

result is shown on Figure 11 with the following configuration: mode='cteAcceleration', learning_rate=0.05, cv::getStructuringElement(cv::MORPH_ELLIPSE, cv::Size(7,5)), MIN_WIDTH=15 and MIN_HEIGHT=60. The predictions improved a lot, the detections of the background objects were solved but the algorithm still struggles when the boat changes direction.

3) streetCornerAtNight_0_100_clip.mp4: For this sequence we needed to avoid the first wrong measurements. After some tuning the best result is shown on Figure 12 with the following configuration: mode='cteVelocity', learning_rate=-1, cv::getStructuringElement(cv::MORPH_RECT, cv::Size(6,6)), MIN_WIDTH=30 and MIN_HEIGHT=20. This configuration outperformed the previous ones.

4) pedestrians_800_1025_clip.mp4: The result for the previous configuration was already really accurate. We managed to get some small improvements with the following configuration: mode='cteAcceleration', learning_rate=0.03, cv::getStructuringElement(cv::MORPH_RECT, cv::Size(4,4)), MIN_WIDTH=10 and MIN_HEIGHT=10. The final result can be seen in Figure 13.

VI. CONCLUSIONS

In conclusion, by smoothing out noise from the image in the pre-processing step, post processing the foreground mask(fgmask) obtained by the OpenCV MOG2 background subtraction module and thresholding to remove the shadow and by using the morphological operation, we were able to achieve good tracking results for the provided video sequences. The tracking results for task 3.1 were very good, despite the fact that the ball was occluded for some frames. For task 3.2, we were provided with 4 video sequences to analyze. The target object in all the 4 sequences were a green ball however, the motion pattern for the ball was different in each video. We tried many different configurations of the parameters and found that the constant acceleration model with a structuring element of size 5x5 and a covariance

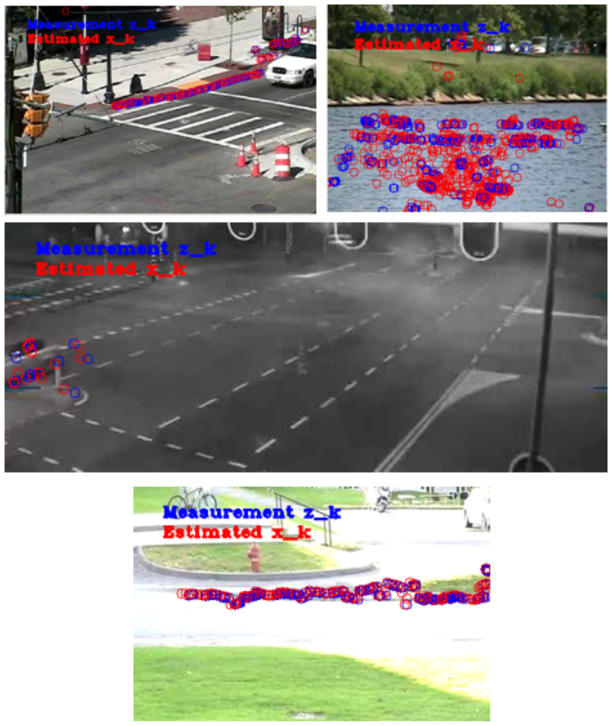

Fig. 8: Results for abandonedBox_600_1000_clip.mp4 (top left), boats_6950_7900_clip.mp4 (top right), streetCornerAtNight_0_100_clip.mp4 (center), pedestrians_800_1025_clip.mp4 (bottom).

observation noise of 24 gave us pretty decent results instead of the noise and jitter in the video sequences. Finally, in task 3.3 we faced very challenging sequences in terms of complex background, multiple objects... However, overall, we manage to solve the vast majority of these issues by tuning most of the parameters of the algorithm.

On the other hand, While working on this project we realised that an accurate foreground mask is a key aspect for object tracking and for further video surveillance analysis.

VII. TIME LOG

Coding: Around 12 hours for merging the code and debugging the errors.
Testing: Around 7 hours of testing the routine to find the optimal set of hyper parameters for different tasks and then analyzing the behaviour of the algorithm.
Report: It took us approximately 7 hours to write the report, take the screenshots of relevant frames,adding the results and formatting them. Proof reading the report to eliminate the possible typo errors.

REFERENCES

[1] OpenCV reference manual http://docs.opencv.org/2.4.13.2/modules/refman.html.
[2] R. E. Kalman, "A new approach to linear filtering and prediction problems" transaction of the asme journal of basic," 1960.

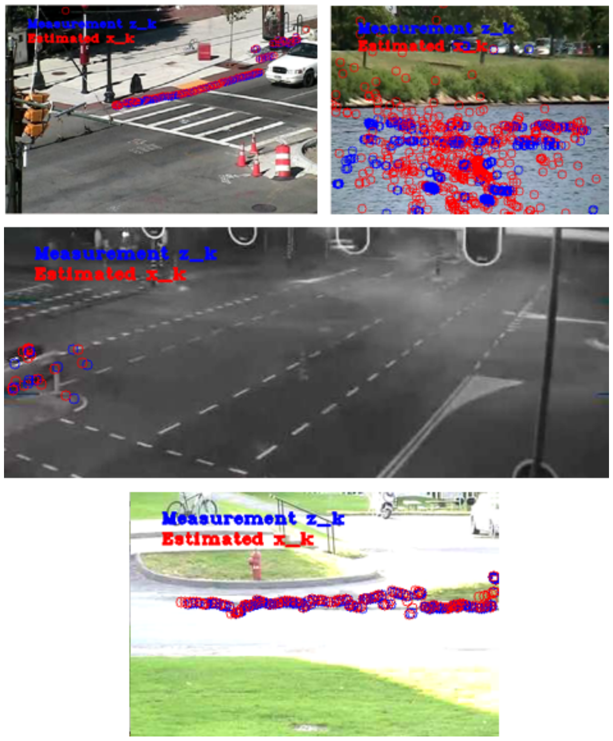

Fig. 9: Results for abandonedBox_600_1000_clip.mp4 (top left), boats_6950_7900_clip.mp4 (top right), streetCornerAtNight_0_100_clip.mp4 (center), pedestrians_800_1025_clip.mp4 (bottom).

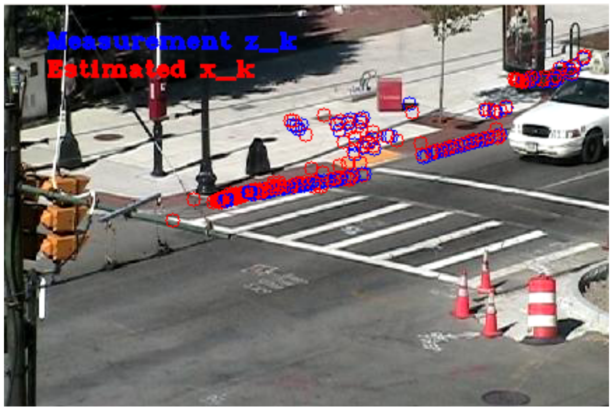

Fig. 10: Measurements and states for abandonedBox_600_1000_clip.mp4.

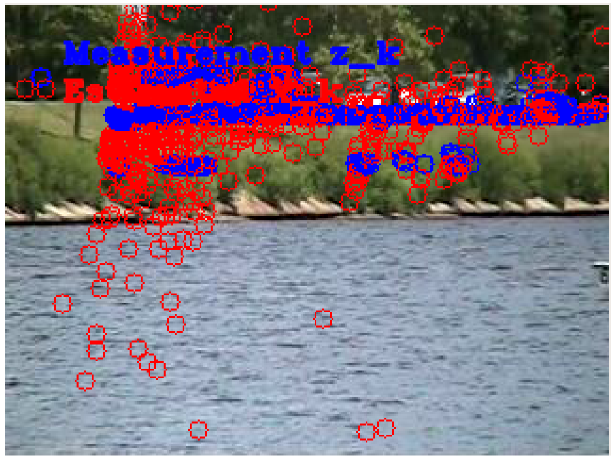

Fig. 11: Measurements and states for boats_6950_7900_clip.mp4.

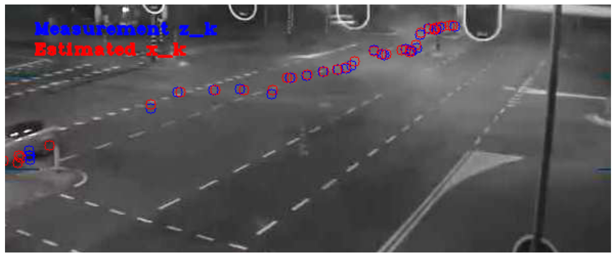

Fig. 12: Measurements and states for streetCornerAtNight_0_100_clip.mp4.

VIII. APPENDIX

A. *Difficulties during individual work*

1) Task-3.1a: The main challenge for task 3.1a was to obtain a good foreground mask because that mask was later used for blob detection. Since, the provided video sequences contained noise and jitter, it was difficult to obtain a foreground mask without noise. However, using the gaussian smoothing and morphological operations, we were able to obtain a decent mask to work with.

2) Task-3.1b: At first the understanding how the Kalman Filter worked was quite confusing. In addition, it took some time to get used to work with the the OpenCV KalmanFilter class.

B. *Modifications during code merging*

We modified the output of task 3.1a just to adapt the format of the coordinates of the center of the blob to the measurement format required for task 3.1b.

C. *Time log for collaborative Work*

1) Coding: Session 1: Six hours Initially, Sergio was acting as driver, and Kush was navigator for the first hour and then we switched the roles each hour. During the first session we were struggling to create a new project in Eclipse and to make the build configurations for that. After successfully creating the new project, we were able to merge the two codes, connecting all the blocks in the pipeline as mentioned in figure1.
Session 2: Six hours In the second session, Kush was acting as driver, and Sergio was navigator for the first hour and then we switched the roles each hour. During this session, we worked on the task 3.2 to found the best parameters for the analysis of the video sequence video2.mp4.
Session 3: Four hours In session 3, Sergio was acting as driver, and Kush was navigator for the first hour and then we switched the roles each hour. During this session, we worked on the task 3.3 in order to find the best parameters for the analysis of the video clip abandonedBox_600_1000_clip.mp4.

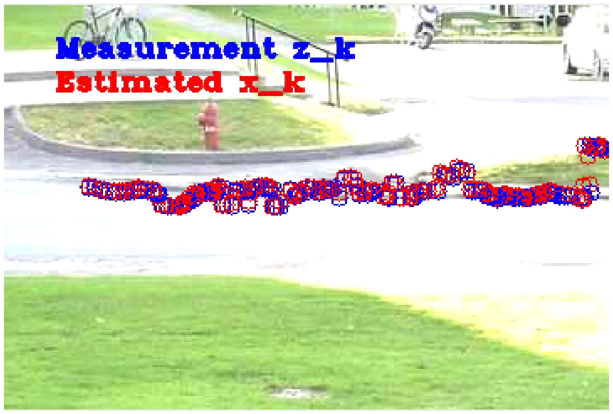

Fig. 13: Measurements and states for pedestrians_800_1025_clip.mp4.

2) Report: Session 1: 7 hours During this session, Kush was acting as driver and Sergio was the navigator for the first hour and then we switched the roles each hour. We wrote all the parts for the report and proof read it in the end.

D. Overall impression

Overall, the proposed methodology for collaborative work was interesting. Using the methodology we experienced that our partner can point out the mistakes we're doing while writing the code or while writing the report. Also, debugging the errors together was quite efficient. However, the major draw back of the approach was to find a suitable time window between the partners to work together.