Kalman Filtering for Object Tracking

Mohammad Mohaiminul Islam and Zahid Hassan Tushar

I. Introduction

This report is generated as a guideline and analysis for the developed tracking module based on the Kalman filter [1]. The objective of this work is to track a single moving object in both fabricated and real-world scenarios using two unique models: constant velocity and constant acceleration. In the first stage of development, we implement those models leveraging OpenCV library functions. Later a comparative analysis was carried out between the response of these models in a toy data scenario keeping measurement extraction parameters constant. Finally we tested our models in the real world scenario and attempted to reach an optimal set of parameters for the best performance for each video sequence.

II. Method and Implementation

Our object tracking module can be divided into two subtasks: Measurement Extraction and Tracking. For the first subtask, we have utilized OpenCV library's built-in functions to extract information about the target object while for the second task we have implemented two models and used OpenCV's Kalman Filter function [2] to carry out the computation required for tracking.

A. Measurement Extraction

Measurement of the target object is very crucial to update the prediction of the Kalman filter. We identified the center coordinate of the extracted blob from the current frame as our measurement information. In our case we carried out this measurement in three steps. In the beginning foreground is detected using the Gaussian Mixture Model which gives a quality response in a noisy video sequence. Fortunately OpenCV includes this model as the MOG function [3] and we extracted our foreground mask using this function. However there still remained some noises which can be handled by performing a morphological operation on the foreground mask. For this purpose we used OpenCV's open method [4] with a 3x3 kernel. Finally we extracted blobs from the foreground mask using the OpenCV's floodFill function [5]. Since the scope of this assignment is limited to track a single object in the video sequence, we sorted the blobs based on their size and kept the largest one. After that we computed the center of this blob which was later used as the measurement information for the current frame.

B. Tracking

Object tracking is a very challenging task based on the scenario it is being applied. There can be partial or complete occlusions of the object, similarities between the tracking objects, slow frame rate, and many more. However, the scope of this assignment is confined in tracking a single object, we are considering linear tracking models based on Kalman filter. It provides estimates of some unknown variables given the measurement observed over time. It assumes a Markovian process. In other words, the present state depends only on the previous state, not all other states. The algorithm works in a two-step process. In the prediction step, the Kalman filter produces estimates of the current state variables, along with their uncertainties. Once the outcome of the next measurement (necessarily corrupted with some amount of error, including random noise) is observed, these estimates are updated using a weighted average, with more weight being given to estimates with higher certainty. The weighting factor is known as Kalman gain. The algorithm is recursive. It can run in real-time, using only the present input measurements and the previously calculated state and its uncertainty matrix; no additional past information is required. Using a Kalman filter assumes that the errors are Gaussian [6].

In our "constant velocity" model, we define our state variable x_k and observed measurement z_k as shown in (1) and (2). In the prediction step, we generate the predicted current state $\widehat{x_k}$ and project the error covariance $\widehat{P_k}$as given by (3) and (4).

$$x_k = [x \ \ \dot{x} \ \ y \ \ \dot{y}] \tag{1}$$

$$z_k = [x \ \ y] \tag{2}$$

$$\widehat{x_k} = A \cdot x_{k-1} \tag{3}$$

$$\widehat{P_k} = A \cdot P_k \cdot A^T + Q \tag{4}$$

Where A is the state transition matrix, Xk-1 denotes the previous state, Q is the covariance of the process noise and Pk-1 is the error covariance from the previous state.

In the update step, we compute the Kalman gain, K using (5). We use this gain to update our prediction of the current state as given by (6). The error covariance is also updated in this step using (7).

$$K = \widehat{P_k} \cdot H^T \left(H \cdot \widehat{P_k} \cdot H^T + R\right)^{-1} \tag{5}$$

$$x_k = \widehat{x_k} + K(z_k - H \cdot \widehat{x_k}) \tag{6}$$

$$P_k = (I - K \cdot H) \cdot \widehat{P_k} \tag{7}$$

Where H is the observation matrix, R is the covariance of observation noise and I denotes an identity matrix.

For the "constant acceleration" model, we define our state variable a bit different from the previous model as given in (8). All other steps are similar to the previous model and carried accordingly.

$$x_k = [x \ \ \dot{x} \ \ \ddot{x} \ \ y \ \ \dot{y} \ \ \ddot{y}] \tag{8}$$

During the implementation of the Kalman filter, we used OpenCV's Kalman filter function [2]. We initialize the parameters differently for both models based on their structure.

When we set the covariance of observation to a small value, the gain is high, leading to a dominant effect on the update of the current state. This indicates that we are relying on the observation more. In contrast when the prediction error covariance is low, the gain falls, leading to a state depending on the previous state rather than the observation.

III. Data

As mentioned earlier, in this assignment we have two types of data: one dataset that represents a toy data scenario and the other one represents a real-world scenario [7]. Task 3.2 is based on the first kind of dataset for which we adjusted our model parameters and evaluated the strengths and weaknesses of individual models. Similarly, for task 3.3 we tuned our parameter and ran a comparative analysis on the later kind of dataset.

A. *Toy Dataset*

The dataset is provided by the AVSA lab and consists of five videos: *singleball.mp4*, *video2.mp4*, *video3.mp4*, *video5.mp4* and *video6.mp4*. The challenge in those videos is to track a single ball moving with different velocity, acceleration and occlusion conditions.

B. *Real-world Dataset*

This dataset are composed with the real-world scenario in four separate videos: *abandonedBox_600_1000_clip.mp4*, *boats_6950_7900_clip.mp4*, *pedestrians_800_1025_clip.mp4* and *streetCornerAtNight_0_100_clip.mp4*. The challenge for this dataset is double-folded: firstly extracting the background subtraction model from the noisy dynamic background and secondly using this measurement tracking a real world object.

IV. Result and Analysis

In the following section we have discussed about our result according to each task we were assigned in this assignment.

A. *Task 3.2 :Analysis of Toy Data*

In the assignment we have started to analyze our result with the ***singleball.mp4*** video sequence for the first task with both of our model constant velocity and constant acceleration model. Our result shows that for this specific video sequence constant acceleration model yields better results than constant velocity model for most of the video sequences but there are few cases where the opposite happens e.g. ***video6.mp4***.

1) In Fig. 1 we can see the tracking result on the *singleball.mp4* video sequence with our constant velocity model. Here it can be observed that when the ball is occluded by a box the model starts to predict the possible position of the ball (Blue trajectory) but it goes beyond the occluded region slightly in a wrong direction and when the measurement becomes available again it jumps back. This is because even though the ball decreases velocity inside the box , the prediction goes further due to the fact that the model is considering a constant velocity of the ball.

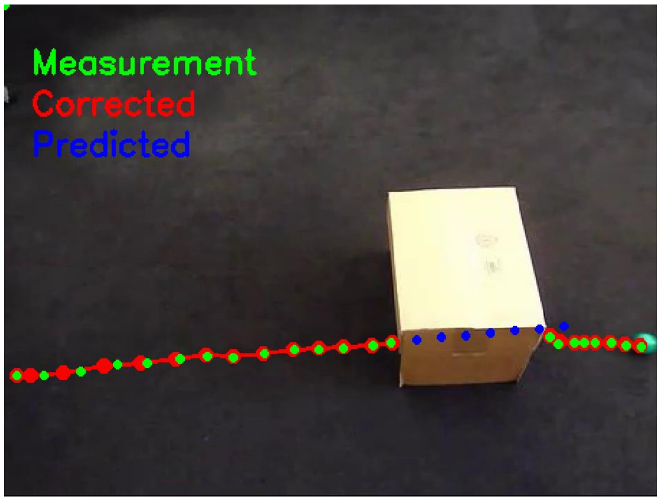

Fig. 1. Tracking of *singleball.mp4* video sequence with Constant Velocity Model (parameters for MOG: learning_rate=0.001, varThreshold=16 and history=50; parameters for Opening: size=3x3, type=MORPH_RECT; parameters for blob extraction: min width=15, min height=15; parameters for Kalman Filter: R Matrix with multiplicative factor 25.0f, P Matrix with Multiplicative factor 10e5)

For the constant acceleration model, this misprediction does not happen as we can see in the Fig. 2. It almost perfectly predicts the position (Blue trajectory) of the ball even when it is occluded. Hence we can say that the constant acceleration model is better.

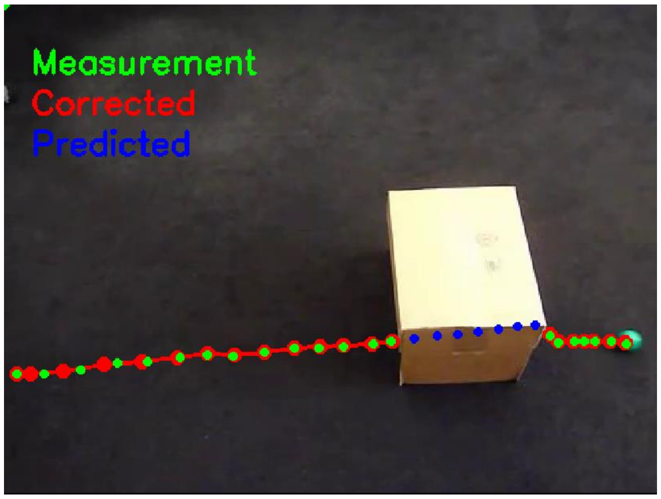

Fig. 2. Tracking of *singleball.mp4* video sequence with Constant Velocity Acceleration (parameters for MOG: learning_rate=0.001, varThreshold=16 and history=50; parameters for Opening: size=3x3, type=MORPH_RECT; parameters for blob extraction: min width=15, min height=15, parameters for Kalman Filter: R Matrix with multiplicative factor 25.0f, P Matrix with Multiplicative factor 10e5)

2) Next on we explored the effect of measurement noise covariance matrix R on our kalman filter model. In the Kalman filter framework measurement noise covariance tells us how much noise we have in our measurement in this case position of the ball. Now for instance, if we increase the value of R then we are not sure about our measurement and consequently Kalman gain K increases thus we rely more on prediction rather than our measurement hence our prediction gets affected. In Fig. 3 we can see the effect. If we increase the R matrix multiplicative factor into 250 we can observe that our prediction is not so correct. This is because when our measurement is not available in other words when the ball is occluded the model predicts (Blue trajectory) the position of the ball. It keeps believing the previous predicted value and due to the high noise covariance it provides wrong prediction but our measurement is not that much noisy in the first place. We can also see that we have a large difference in our measurement between (Green dots) and corrected(red dots) position of the ball due to high R value. We have also explored the performance of our model on a few test video sequences which are provided with this assignment. In this case for video sequence *video2.mp4*, *video3.mp4* and *video5.mp4* both of our models provide nearly the same result but for *video6.mp4* constant velocity mode provides better performance. So we have decided to only discuss the constant velocity model.

In Fig. 4, Fig. 5 and Fig. 6 show the result obtained with a constant velocity model after tuning the Kalman filter parameters. These parameters have been tuned for each of the video sequences individually and mentioned below each figure. Primarily we have played with two parameters of the Kalman filter namely the measurement covariance matrix R and prior error covariance matrix P. The intuition behind tuning these parameters are, when the segmentation is clear and we have a less noisy measurement, it is highly probable that one will get good positioning of the object, so we should trust our measurement more by lowering measurement noise covariance matrix R.

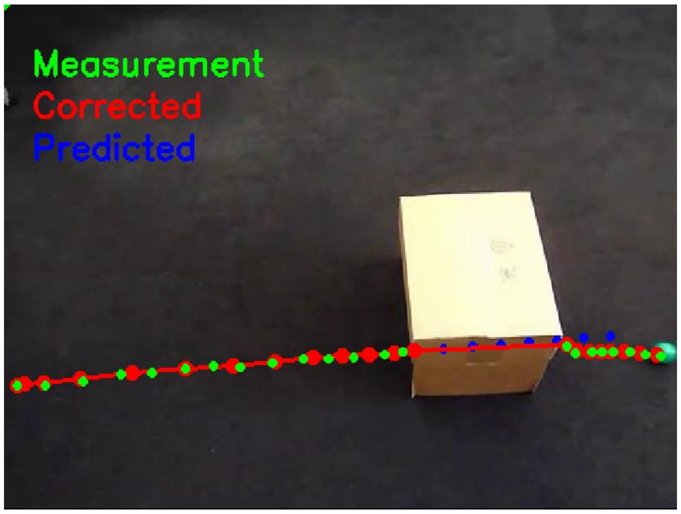

Fig. 3. Demonstration of the effect of increasing measurement noise covariance on video sequence *singleball.mp4* (parameters for MOG: learning_rate=0.001, varThreshold=16 and history=50; parameters for Opening: size=3x3, type=MORPH_RECT; parameters for blob extraction: min width=15, min height=15, parameters for Kalman Filter: R Matrix with multiplicative factor 250.0, P Matrix with Multiplicative factor 10e5)

On the other hand if our we have very accurate object state than we would want to set the state error covariance very low and if we are not quite sure about our object state we would want to set higher value. In these 3 video sequences we have fairly good segmentation due to static background, light condition and no occlusion thus we have less noise in our measurement and we are quite certain about state of the object. So a low value for both R and P matrices works well in this case.

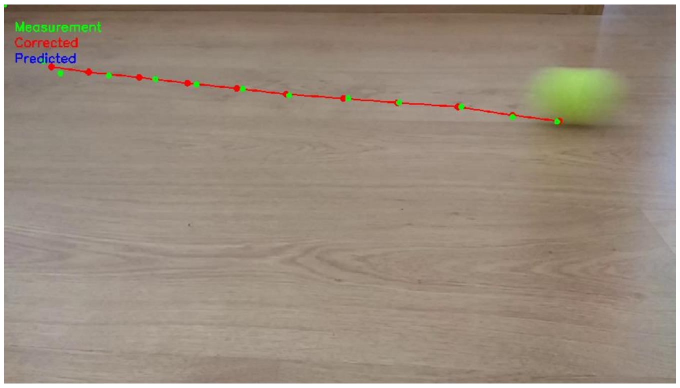

Fig. 4. Tracking of *video2.mp4* video sequence with constant Velocity (parameters for MOG: learning_rate=0.001, varThreshold=16 and history=50; parameters for Opening: size=3x3, type=MORPH_RECT; parameters for blob extraction: min width=50, min height=50, parameters for Kalman Filter: R Matrix with multiplicative factor 25.0, P Matrix with Multiplicative factor 10e2)

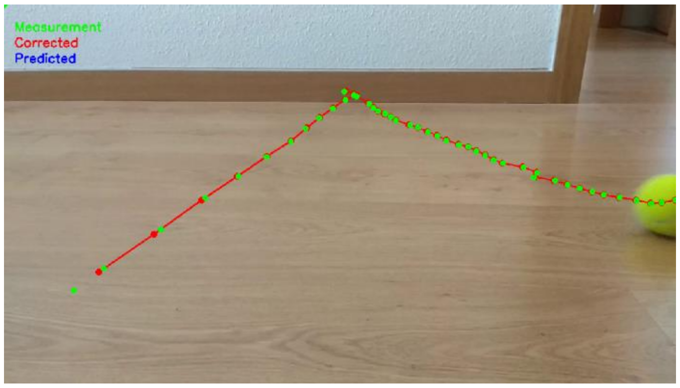

Fig. 5. Tracking of *video3.mp4* video sequence with constant Velocity (parameters for MOG: learning_rate=0.01, varThreshold=16 and history=50; parameters for Opening: size=3x3, type=MORPH_RECT; parameters for blob extraction: min width=50, min height=50, parameters for Kalman Filter: R Matrix with multiplicative factor 15.0, P Matrix with Multiplicative factor 10e2)

However in the case of video sequence *video6.mp4,* from Fig. 7 we can see that there is an occlusion and due to this we don't get the measurement in some frames thus we are not sure about our object state. So want to believe more on prediction than measurement hence a large value for both P and R matrix works well for this particular video sequence.

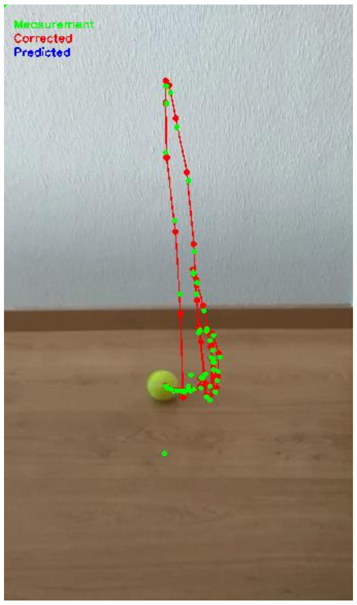

Fig. 6. Tracking of *video5.mp4* video sequence with constant Velocity (parameters for MOG: learning_rate=0.01, varThreshold=16 and history=50; parameters for Opening: size=3x3, type=MORPH_RECT; parameters for blob extraction: min width=50, min height=50, parameters for Kalman Filter: R Matrix with multiplicative factor 15.0, P Matrix with Multiplicative factor 10e2)

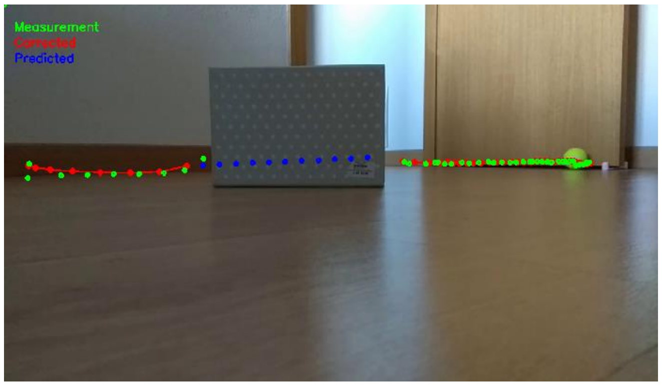

Fig. 7. Tracking of *video6.mp4* video sequence with constant Velocity model. (Parameters for MOG: learning_rate=0.001, varThreshold=16 and history=50; parameters for Opening: size=3x3, type=MORPH_RECT; parameters for blob extraction: min width=15, min height=15, parameters for Kalman Filter: R Matrix with diagonal multiplicative factor 300.0, P Matrix with diagonal Multiplicative factor 10e4)

B. *Task 3.3 :Analysis of Real-world Data*

In this part we have investigated the performance of our model on a real dataset (video sequences) from changedetection.net. While detecting objects in real scenario the main challenge lies in getting proper segmentation from the video sequence in consecutive frames of the video. Hence here we tried to extensively tune the parameters to get as good as possible measurement from the videos. As we are using Gaussian Mixture Model for background subtraction we have experimented with a few parameters such as learning rate, variance threshold for the pixel-model match, number of last frames that affects the current frame of this model and as well as the kernel size of the morphological operation.

In Fig. 8 shows the tracking result of the video sequence *abandonedBox_600_1000_clip.mp4*. The tracking gets complicated when the cyclist stops for a while at the traffic signal. Here if our learning rate is high the cyclist disappears from the foreground mask and our model starts to predict the position considering the object has occluded by something even though it is in the original scene. So we have adjusted (reduced) the learning rate for this particular video sequence to keep the cyclist in the foreground mask for a little bit long. We have also adjusted the variance threshold and history to stabilize the blob detection.

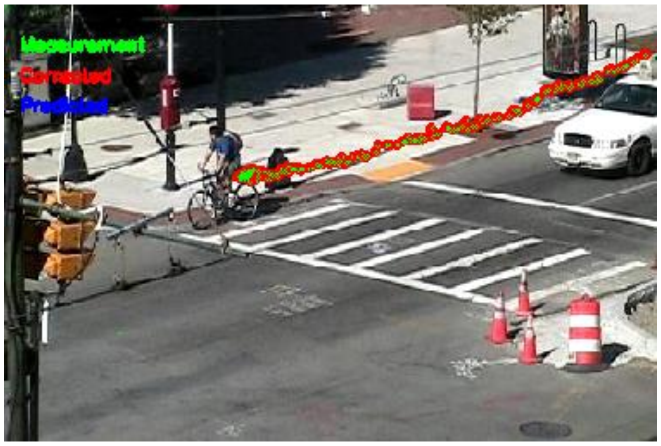

Fig. 8. Tracking of *abandonedBox_600_1000_clip.mp4* video sequence with constant Velocity model. (Parameters for MOG: learning_rate=0.00001, varThreshold=10 and history=100; parameters for Opening & Closing: size=3x3, type=MORPH_RECT; parameters for blob extraction: min width=15, min height=15, parameters for Kalman Filter: R Matrix with diagonal multiplicative factor 20.0, P Matrix with diagonal Multiplicative factor 10e4)

The next video sequence in our test set is from *abandonboats_6950_7900_clip.mp4* in which we are trying to track a boat. This particular video sequence is most challenging to track due to the very dynamic background (water, car and human in background). Also we can observe in Fig. 9 that when the boat changes direction and takes a certain angle the blob detection method detects the sail and lower part of the boat as two separate blobs. These detected blobs are larger in a few frames alternatively hence we get some very noisy measurement at the end of the video sequence. We tried to obtain a reasonable performance by tuning the kernel size of the morphological operation, learning rate and variance threshold.

The challenge in the next video sequence is due to the shadow behind the person. Our background subtraction model encodes the shadow pixels with 127 pixel values in the foreground mask, so to eliminate the shadow we have thresholded our foreground to eliminate all the pixel values with 127. Since the person was moving slowly and there are certain deformations, a low learning rate and variance threshold yielded a better result. In Fig. 10 we can observe the result obtained by our model on *pedestrians_800_1025_clip.mp4* video sequence.

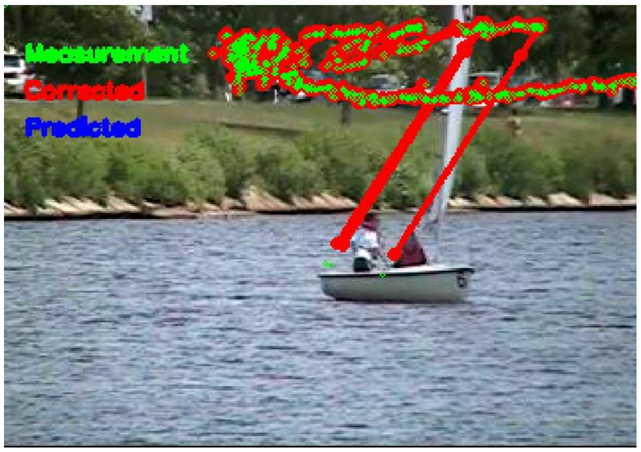

Fig. 9. Tracking of *abandonboats_6950_7900_clip.mp4* video sequence with constant Velocity model. (Parameters for MOG: learning_rate=0.0001, varThreshold=18 and history=50; parameters for Opening & Closing: size=5x5, type=MORPH_RECT; parameters for blob extraction: min width=40, min height=40, parameters for Kalman Filter: R Matrix with diagonal multiplicative factor 20.0, P Matrix with diagonal Multiplicative factor 10e4)

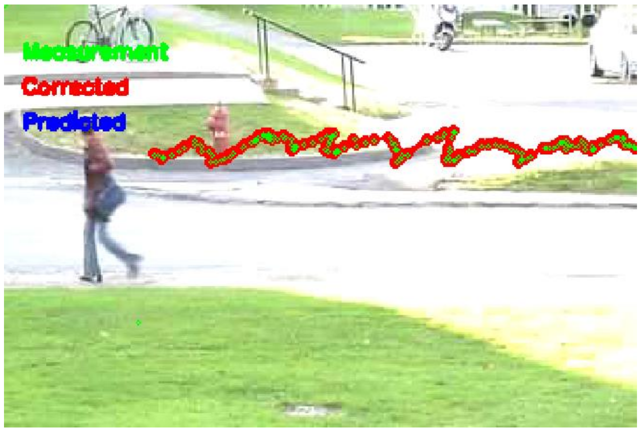

Fig. 10. Tracking of *pedestrians_800_1025_clip.mp4* video sequence with constant Velocity model. (Parameters for MOG: learning_rate=0.00001, varThreshold=10 and history=50; parameters for Opening & Closing: size=4x4, type=MORPH_RECT; parameters for blob extraction: min width=25, min height=25, parameters for Kalman Filter: R Matrix with diagonal multiplicative factor 20.0, P Matrix with diagonal Multiplicative factor 10e4)

The final video sequence in our test data is *streetCornerAtNight_0_100_clip.mp4* where we tried to track a speedy car at a night scene. The main challenge of the video is not getting proper blob segmentation. This is due to the high light exposure of the car's headlight and excessive speed of the car. To cope with fast change we have experimented with a slightly large value of learning rate and variance threshold. Fig. 11 shows the result of tracking the video sequence *streetCornerAtNight_0_100_clip.mp4*.

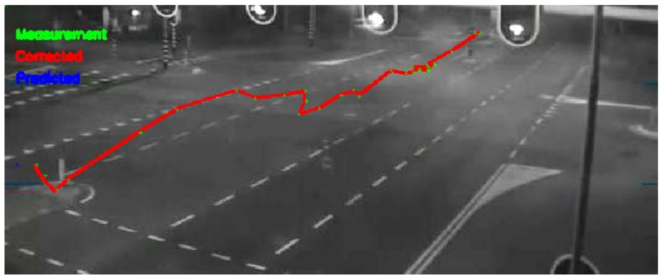

Fig. 11. Tracking of *streetCornerAtNight_0_100_clip.mp4* video sequence with constant Velocity model.(parameters for MOG: learning_rate=0.001, varThreshold=20 and history=50; parameters for Opening & Closing: size=3x3, type=MORPH_RECT; parameters for blob extraction: min width=25, min height=25, parameters for Kalman Filter: R Matrix with diagonal multiplicative factor 20.0, P Matrix with diagonal Multiplicative factor 10e4)

V. CONCLUSIONS

In conclusion, we found that tracking objects is a very tricky task. Extensive parameter selection is required for different application scenarios. While objects in static backgrounds are easy to track despite occlusions, tracking in dynamic background demands more advanced parameter adjustment for background subtraction. The performance of both models are arguable. For some cases one performed better than the other and vice-versa. Further, non-linear models can be explored instead of Kalman filters for better object tracking.

VII. TIME LOG

- **Background study** 5 hours: Understanding the framework provided by the lab and familiarizing with the KalmanFilter class of OpenCV library.
- **Measurement Extraction** 3 hours: Implementing the methods from scratch for blob extraction, refinement and measurement data generation.
- **Tracking model** 7 hours: Designing the program flow, writing the code and solving data type errors in C++.
- **Parameter adjustment and Analysis** 8 hours: 3 hours to reach an optimal set of parameters for toy dataset and 5 hours to tune our models for real-world dataset due to the presence of extreme noises.
- **Report** 8 hours: writing, taking screenshots, proof-reading and editing.

REFERENCES

[1] Kalman, R. E. (March 1, 1960). "A New Approach to Linear Filtering and Prediction Problems." ASME. J. Basic Eng. March 1960; 82(1): 35–45.

[2] https://docs.opencv.org/3.4.1/dd/d6a/classcv_1_1KalmanFilter.html

[3] https://docs.opencv.org/3.4/d7/d7b/classcv_1_1BackgroundSubtractorMOG2.html

[4] https://docs.opencv.org/3.4/d3/dbe/tutorial_opening_closing_hats.html

[5] https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html?highlight=floodfill

[6] https://en.wikipedia.org/wiki/Kalman_filter

[7] AVSA Lab 3 " Kalman Filtering for Object Tracking Dataset"