Applied Video Sequence Analysis Lab 3 "Kalman filtering for object tracking"

Mohamed Hussein Fathy Mohamed Hassan
Sanjay Kumar

1. INTRODUCTION

In this laboratory report, we perform background subtraction, Blob extraction, and Kalman filter for single object tracking. We implemented and analyzed the behavior of the tracker, firstly on TOY DATA, then, on REAL DATA. We tested the effect of different parameters value in order to fine-tune the parameters for each video to get the best tracking results on the provided datasets. The main purpose of this laboratory is to show the implementation of Kalman Filter using two different motion models for prediction: a constant velocity model and a constant acceleration model. During the implementation of these tasks, we used C++ with OpenCV library. For the evaluation and analysis part, we used given different video sequences(datasets). The tracker program has two parts: blob extractor, then the filtered blobs are feed to the tracker for the measurement step. and the output from the prediction motion model for the tracker prediction step.

2. METHOD

The tracking approach used in this laboratory session is the Kalman Filter which supports two models: constant velocity and constant acceleration.

Kalman Filter: Kalman Filter is one of the most famous filtering algorithm used is many fields. It is optimal estimator for linear systems and Gaussian errors. Kalman Filters are comparatively simple and require small computational power, due to this reason it is used in many applications such as using vision camera to perform real time image processing for vehicle tracking. This algorithm consists of two steps: 1) Prediction 2) Correlation as shown in figure(1).

A. Constant Velocity: In this model, it is supposed that the value of velocity our object of interest does not change during tracking. For this constant velocity model, the only variables that we considered are the position and speed of the object along x and y axis.

B. Constant acceleration On the other hand, in this model we supposed that the positions and velocity of the tracked object can change, but their acceleration remain constant. here we add two extra variables acceleration in x and y axis.

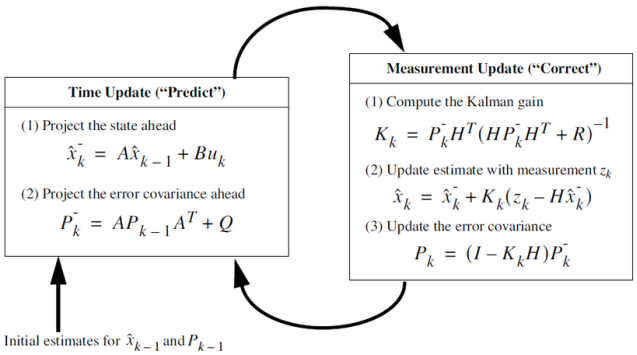

Figure 1. Kalman Filter Steps

3. IMPLEMENTATION

In this section we will briefly discuss about the Kalman filter algorithm. We will implement single object tracker which consists of two parts. 1) Measurement Extraction
2) Kalman Tracking

The code consist of main and two classes which are KalmanTracker and ShowManyImages. KalmanTracker is the main class that contains the tracking loop pipeline, the OpenCV KalmanFilter class, blob extraction, and trajectory plotting. finally, The main function responsible for creating an instance from an KalmanTracker class and initialize it. then call the method to start the tracking pipeline.

Kalman filter pipeline steps: 1) Foreground extraction. 2) blobs extraction. 3) Kalman filter algorithm.

Step1: Foreground extraction: Foreground extraction is used to find the non-stationary objects in the frame to track them, we use the method ::ExtractForeground() for that. which contains OpenCV's BackgroundSubtractorMOG2 for the substation and in order to use this function we need to provide it with four parameters: history – Length of the history, varThreshold – Threshold on the

squared Mahalanobis distance between the pixel and the model, detect shadows – If true, the algorithm will detect shadows. learningRate – The value between 0 and 1 that indicates how fast the background model is learned. and we will change these values besides the Kalman filter parameters in task 3 to fine-tune the tracker. then apply a morphological opening operation on the resulting output from the previous step to remove noise.

Step2: blob extraction: This foreground mask from the previous step is passed to the ::ExtractBlob() method which detects the biggest blob depend on our x, y size constraints, which means that the returned blob from this method is only one blob with the largest area size.then the center point of these blobs will be used by Kalman filer for the measurement step.

Step3: Kalman filter: we used the Kalman filter class from OpenCV, which contains two methods Prediction() , and Update(const cv::Point measurement). we provide the update method with extracted blob center point and the Prediction step predict the next move for the blob depend on the motion model that we are using Constant Velocity or Constant Acceleration model.

4. DATA

there are 3 tasks (task 3.1, 3.2 and 3.3.) contains 9 video sequences. In 3.1 baseline video is given for testing the Kalman Filter. While video sequences in task 3.2 are given for toy examples, and video sequences in task 3.3 are more challenging and represent real life scenarios.

1) Video sequence for task 3.1:
In **"singleball"** baseline video sequence, the ball rolls on surface and passes behind the box and leaves the frames and in some frames the ball is not seen. This video sequence helps us to understand how good the algorithm predicts the position of ball.

2) Video sequences for task 3.2 (Toy Data):
In **"video2"** sequence, we see a tennis ball comes from one side and leave from other side of frame. Ball rolls fast on the floor during its whole trajectory.
In **"video3"** sequence, we see a tennis ball rolls on the floor which strike to the wall, bounced back and left the frame.
In **"video5"** sequence, we see a tennis ball fall on floor (not rolls on floor) after few bouncing it stops slowly. We can see ball has vertical motion and also noise.
In **"video6"** sequence, we see a tennis ball rolls on floor which passes through a box and then stopped just little hitting the door. We can see ball has horizontal motion.
3) Video sequences for task 3.3 (Real Data):
In **"abandoned-Box"** video sequence, we can see it is more challenging than previous video sequences. This video sequence has background with many details. A man ridding a cycle passed behind the car and stop at crosswalk waiting for traffic light. The difficulty of this video sequence is the lost of measurement when rider/cycler stops.
In **"boats"** video sequence, we can see a boat enters in the frame, which turns around in the opposite direction. The main difficulty with this Boat sequence is that it has connected with the dynamic background, which makes the extraction of blobs challenging.
In **"pedestrians"** video sequence, we can see a women enter in the frame from the left side. But here main thing to notice is that this video sequence is taken under strong illuminates condition and cause shadow of the walking woman.
In **"streetCornerAtNight"** video sequence, we can see a fast moving car with front lights open enters into frame. The main difficulty with this sequence is that, first noise in all frames may be due to bad quality of camera or due to lighting condition, second object motion is super-fast, and last the front light of car.

5. RESULTS AND ANALYSIS

1.Analysis of Baseline:
The goal of this task, is to analyze the effect of Kalman components for toy video sequences with fixed measurement extraction parameters. learningrate= 0.001, morphsize=3, varThreshold= 16 and history = 50, minimum width and height for blob extraction = 50, kernelsize=3 , we can notice in figure(2) and figure(3) that the constantAcceleration motion model show better performance than the constantVelocity in predication especially when the object disappeared.

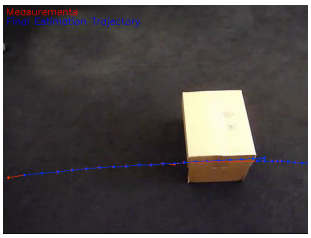

Figure 2. constantVelocity

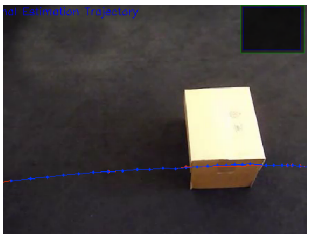

Figure 3. constantAcceleration

2.Analysis of toy data:
The goal of this task, is to analyze the effect of Kalman components for toy video sequences with fixed measurement extraction parameters. learningrate= 0.001, morphsize=3, varThreshold= 16 and history = 50, minimum width and height for blob extraction = 50, kernelsize=3

In **video2 sequence**, with above initial parameters we compare the performance of constantVelocity and con-

stantAcceleration methods. During Analysis we observed that with constantVelocity model, the estimation and the predictions are both near to the measurements because the velocity of the model is constant. And with constantAcceleration model, some predictions are little bit away from the measurements as we can see in figure(4). To conclude, we saw final trajectory can track and estimates the position of the ball quite well. Also, the final and measurement trajectories are not perfectly linear because of the reflection of ball on floor. As shown in figure(5).

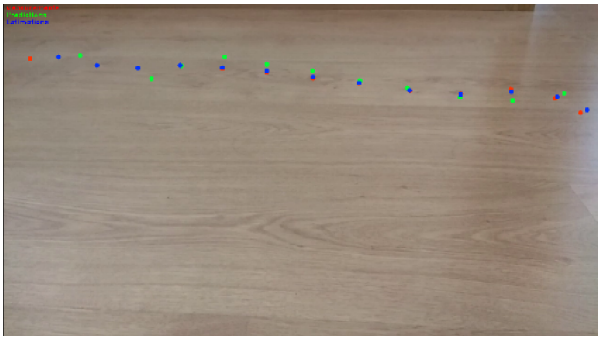

Figure 4. constantAcceleration

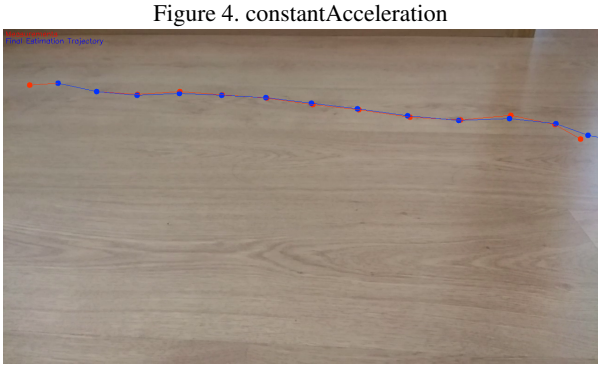

Figure 5. constantVelocity

In video3 sequence, we observed that ball enter into frame, strike the wall and return back, which result a change in moving direction. We observed constantVelocity model can track the ball successfully until it leaves the frame. As we can see in Figure(6), both measurement and final trajectory lines are close to each other but the prediction almost fail when the ball hit the wall which can affect the final trajectory in order to overcome this we increase the trust in measurement by reducing the measurement noise in R matrix (value=30).
On the other hand, in constantAcceleration model we observed the strange behavior of the final trajectory, that the ball leaves the frame and the prediction trajectory stays in the frame, this made the final trajectory curved back around the corner of frame. As we can see in Figure(7).
To conclude, constantVelocity model performed better than constantAcceleration model. As the final trajectory of constantVelocity is leaving the frame perfectly. Also in both models, the trajectory before hitting the wall was almost linear but after hitting the wall, the ball velocity starts to decrease and also trajectories are not linear.

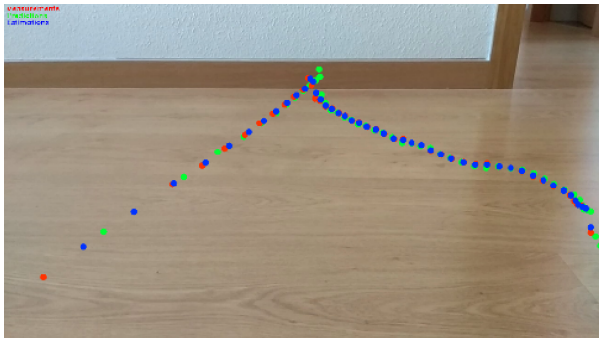

Figure 6. constantVelocity (R =30)

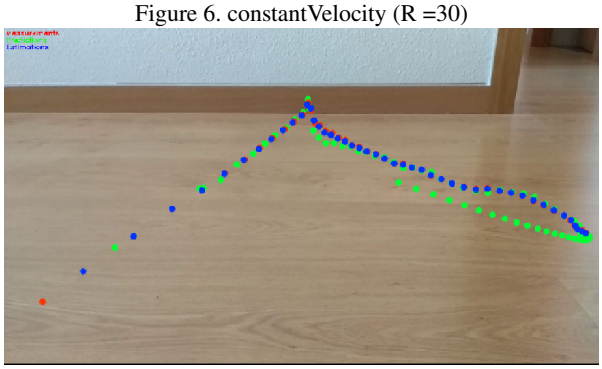

Figure 7. constantAcceleration

In video5 sequence, the ball falls from the top on the floor and bounce serval time. First of all, we significantly increased the measurement error value (R=1000) to make the estimation almost depend only on the prediction step, so we can compare between constantVelocity and constantAcceleration performance. We observed that constantAcceleration are more close to the measurement values than the constantVelocity as we can see in figure(8). But when we decreased the measurement error values (R=30), we noticed the both models handle the bouncing quite well, but constantAcceleration is performing well due to the decay of velocity each bounce. As we can see in Figure(9).

In video6 sequence, we observed that at some point object disappears and reappears again because of rigid box. When ball disappeared, the estimation trajectory is only depending on prediction step. Now we noticed that constantAcceleration shows the strange behavior in the trajectory of predicted step which make a curve by leaving the linear trajectory. We can see in the Figure(10).

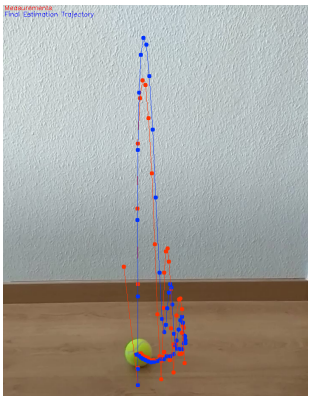

Figure 8. constantVelocity(R=1000)

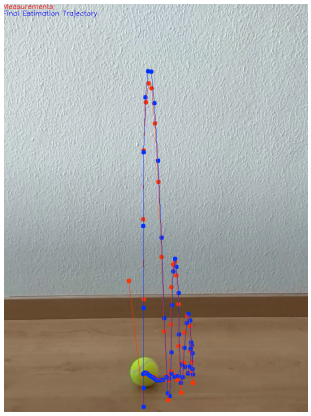

Figure 9. (R=30)

But on the other hand constantVelocity showed a good estimated linear trajectory during the disappearance of ball. But when the ball appeared again we noticed the prediction trajectory was quite far from the measurement input, but the update step (correction) handles this error and the final trajectory show the good tracking. We can see in Figure(11).

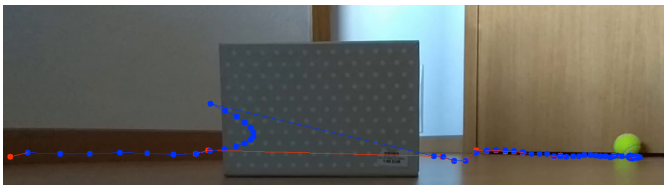

Figure 10. constantAcceleration

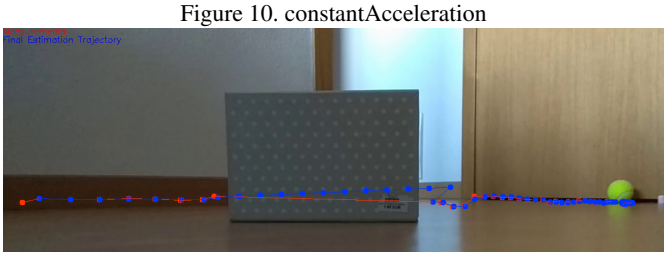

Figure 11. constantVelocity

3.Analysis of real data:

For the **abandonedBox** 'video, there are some challenges such as speed of cycler is not constant and the main challenge is at the end cycler stops and wait for the traffic light so we have to deal with static object and also there is partial occlusion with car.

In order to overcome these challenges, we fine-tuned the detection parameters using blob size = (15,15) also we reduced learning rate equals to 0.0005 in order to overcome the stationary object in the foreground blob detection. And we can see the effect of using 0.001 learning rate in figure(12) and figure(13).

We noticed from observing output trajectories of both models constantVelocity and constantAcceleration are resulting quite good tracking performance. However, the ground truth cycler trajectory is linear but our estimation trajectories are not perfectly linear. Moreover, when the detection is failed to detect the cycler we could noticed that the constantVelocity motion model can give a better estimation result than the constantAcceleration as we can see in the figure(14) and figure(15).

Figure 12. constantVelocity LearningRate = 0.01

Figure 13. constantAcceleration LearningRate = 0.01

Figure 14. constantVelocity LearningRate = 0.0005

Figure 15. constantVelocity LearningRate = 0.0005

This **"boat" video** is more challenging than previous videos. The challenges are: the biggest blob we can detect is the sail part, but the sail part is changing its size specially when the boat starts to turn around, and also the boat motion trajectory is not linear so we can expect a bad performance form the kalman filter and also from the motion model that we are using which is constantVelocity and constantAcceleration.

First of all, we are expecting the bad performance specially from the prediction step in order to avoid that we will increase the trust in the measurement step by reducing the values of R=10, learning rate = 0.0005 and blob size = (15,15). Secondly, by observing the behavior of constantVelocity and constantAcceleration we noticed that before the boat turn around the trajectory of both models was not perfectly linear but not that bad. However, the constantVelocity model shows more robust tracking behavior, but when the boat turned around both of the motion model failed to track during the turning, but when boat start to move in the opposite direction the prediction trajectory start to produce better results. As we can in both the models in Figure(16) and Figure(17).

Figure 16. constantVelocity

Figure 17. constantAcceleration

For the **"streetCornerAtNight" video**, there are some challenges like car entering in the frame very fast with front light on, the car size is changing during its movement towards the camera, and also due the dark environment the strong front light of the car is detected as foreground which will produce some noise in the estimated trajectory. After fine tuning the detection parameters we set the blob size (35,35), morph size = 3, learning rate = 0.001.
By observing the behavior of constantVelocity and constantAcceleration we noticed that both model is almost giving the same results. However, constantVelocity is more linear. Also, we can notice that the measurement values and prediction values are quite near to each other, so if we increase or decrease the values of R matrix it will not going to significantly affect the final estimated trajectory. As we can see in Figure(18) and Figure(19), also we can observe the error in trajectory if using small blob size = 15 in figure(20) For

Figure 18. constantVelocity

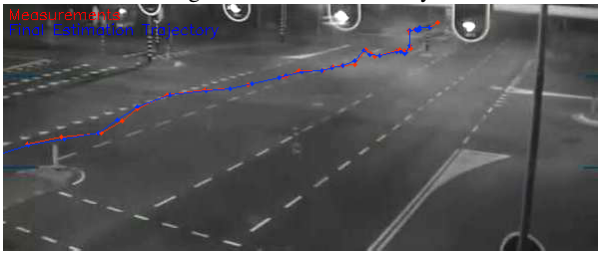

Figure 19. constantAcceleration

the **"pedestrians" video**, there are some challenges such as the pedestrian shadow, non linear motion for the pedestrian during walking. After fine tuning the detection parameters we set the blob size (15,15), morph size = 5, learning rate = 0.001.
By observing the behavior of constantVelocity and constantAcceleration we noticed that both model is almost giving

the same results but we can see that both constantVelocity and constantAcceleration became not robust when the tracked object start to make non linear motion. As we can see in Figure(21) and Figure(22)

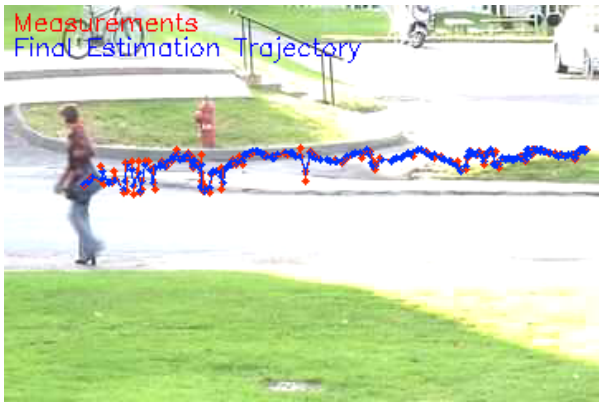

Figure 21. constantVelocity

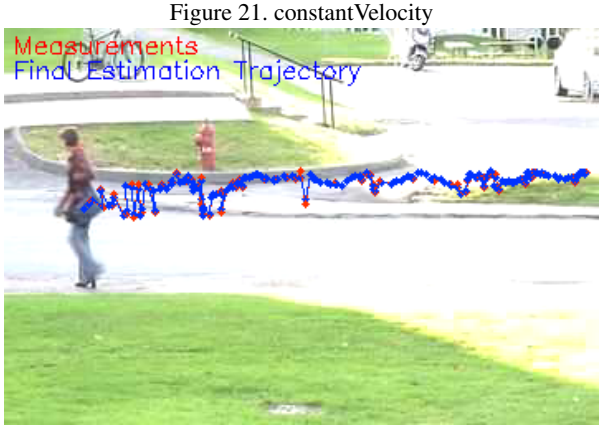

Figure 22. constantAcceleration

6. Conclusion

In conclusion, the Kalman filter is not a general-purpose for all the tracking problems, but it works fine for tracking linear motion objects. and both constant velocity and constant acceleration motion models can work with high efficiency if we fine-tuned the initialization parameters. But find the correct parameters are customized and task-specific for the required video sequence, especially if the sequence has difficulties such as occlusions, shadows, angle change,

and so on. but overall Kalman filter works pretty well for tracking linear motion object with possible Gaussian noise.

7. TIME LOG

the amount of time spent on each aspect of the project is detailed.
1) Code implementation: 8 hours, Writing the Kalman filter pipeline code.
2) Evaluation on toy dataset: 6 hours.
3) Evaluation on real world data: 7 hours.
4) Report: 8 hours, writing the Report.

References

[1] Welch, Greg, and Gary Bishop. "An introduction to the Kalman filter.": 127-132.