Kalman Filter for the Real Time Object Tracking

Fu Min
72677006
ShanghaiTech

Abstract

The objective of this project is to gain a deep understanding of Kalman Filter. In this report, I detail a system capable of simultaneously tracking the ball. I use a pre-filtered video sample as input. And I propose Kernel Density Extraction (KDE) as an efficient method to solve multiple problems in object tracking.

1. INTRODUCTION

The kalman filter, known as linear quadratic estimation, is an algorithm that uses a series of measurements observed over time, containing noise (random variations) and other inaccuracies, and produces estimates of unknown variables that tend to be more precise than those based on a single measurement alone. The kalman filter implements a Bayes filter and was invented by Rudolph Emil Kalman in the 1950 [1].

The Kalman filter has numerous applications in technology. A common application is for guidance, navigation and control of vehicles, particularly aircraft and spacecraft. Furthermore, the Kalman filter is a widely applied concept in time series analysis used in fields such as signal processing, econometrics and tracking. Object tracking has a wide range of applications, such as surveillance, sports, medical imaging etc[2].

The Kalman filter uses a system's dynamics model (e.g., physical laws of motion), known control inputs to that system, and multiple sequential measurements (such as from sensors) to form an estimate of the system's varying quantities (its state) that is better than the estimate obtained by using any one measurement alone. The Kalman filter keeps track of the estimated state of the system and the variance or uncertainty of the estimate.

In this regard, this report analyses and discusses the kalman filter object tracking approach in the context of video analysis. It has been exhaustively used in several Object Tracking problems, for instance in [3]. Object Tracking might be complex due to several factors, such as noise in the image frames, scene illumination changes, complex image motion, object occlusions, real-time image processing etc. This paper addresses the three later. Additionally, I also inspect some object detection techniques involving image processing, which are essential for a successful object tracking.

2. METHOD

Object tracking in videos is the task of locating a specific object in a sequence of frames. First, a detection algorithm is required in order to obtain possible object locations. Next, the object tracker is then responsible for following the trajectory of the object in subsequent frames.

2.1. Point tracking

In this project, I consider the point tracking category for object tracking. Point tracking represents the object by a point and estimates the current object location based on estimations in previous frames. That is, there is a dependence between estimations at time k and time k+1. Moreover, these estimations usually are the pixel coordinates of the object position and its motion (e.g. velocity, acceleration). This information is stored in a vector which is referred to as state vector[4].

2.2. Kalman Filter

If the signal and noise are jointly Gaussian, then the Kalman filter is an optimal MMSE estimator [5]. In this approach, the belief at time step n, i.e. s_n, is assumed to be Gaussian distributed, which allows it to be uniquely described by a mean vector $\hat{s}_{n|n}$ and a covariance matrix $M[n|n]$. A standard Kalman filter is a set of equations that implement a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance, when some conditions are presumed[6]. It consists of two steps, namely *prediction* and *update*, each of them making important assumptions.

1) Prediction: In the prediction step, the KF uses previous state to make a first prediction of the current state. In particular, it assumes that the state $s[n]$ is related with $s[n-1]$ through a linear function for all $1 \leq n \leq N$, where N is the duration of the tracking task.This linear function is given by

$$s[n] = A_n s[n-1] + B_n v[n] + u[n] \qquad (1)$$

where we have used the following notation:

- $s[n]$: $p \times 1$ vector representing the state vector at time step n. Typically contains data of interest.
- $v[n]$: $m \times 1$ vector denoting the control vector. Typically contains control input data.
- A_n: $p \times p$ matrix state transition matrix which establishes the relation between two consecutive states.
- B_n: $p \times m$ control matrix. Relates the control vector and the estimated state.
- $u[n]$: $p \times 1$ Gaussian random vector modelling the randomness in this system. It is assumed to be zeromean with covariance matrix $Q[n]$.

In the context of object tracking, where there is no external influence, both the control matrix and control vector can be omitted. Hence, (1)is rewritten as

$$s[n] = A_n s[n-1] + u[n] \qquad (2)$$

As aforementioned, the state is fully characterized by a mean vector and a covariance matrix. Thus, the predicted state is defined by the predicted mean $\hat{s}[n|n-1]$, and the minimal predicted MSE covariance matrix $M[n|n-1]$. Using (2) we can write the prediction equations as

$$\begin{aligned} \hat{s}[n|n-1] &= A_n \hat{s}[n-1|n-1] \\ M[n|n-1] &= A_n M[n-1|n-1] A_n^T + Q[n] \end{aligned} \qquad (3)$$

The system performance depends on the value of A_n and $Q[n]$, whose impact will be later studied.

2) Update: Next, in the update step the KF uses the current measurement, i.e $x[n] = [x_1[n]x_2[n]...x_M[n]]^T$ to correct (or update) the object's state. It is assumed that the relation between the measurement $x[n]$ and the state $s[n]$ is also linear. The observations are modeled using the Bayesian linear model

$$x[n] = H_n s[n-1] + w[n] \qquad (4)$$

where H_n is a known $M \times p$ matrix, x[n] is an $M \times 1$ observation vector, and w[n] is a $M \times 1$ observation noise sequence, and $w[n] \sim N(0, C[n])$.

In this step, the predicted parameters $\hat{s}[n|n-1]$ and $M[n|n-1]$ are corrected. For this purpose, the Kalman gain $K[n]$ is defined, which quantifies the relative importance between the update and prediction step. That is, the higher it is, the more relevant is the measurement for the state estimation, which might occur when the measurement is very certain. Using (4) we write the update equation as

$$\begin{aligned} K[n] &= M[n|n-1] H_n^T (C[n] + H_n M[n|n-1] H_n^T)^{-1} \\ \hat{s}[n|n] &= \hat{s}[n|n-1] + H_n(x[n] - H_n \hat{s}[n|n-1]) \\ M[n|n] &= (I - K[n] H_n) M[n|n-1] \end{aligned} \qquad (5)$$

The recursion is initialized by $\hat{s}[-1|-1]$ =μ_s, and $M[-1|-1]$ =C_s

3. IMPLEMENTATION

I have selected 1 minute recording of the Nintendo Pinball arcade game as my test video and have focussed on tracking the ball. In addition, I deleted some regions from the original video in order to create occlusions. This allowed me for testing the performance of my implemented method when the measurements were very poor. Prior to any tracking algorithm, I pre-process each input video frame n in order to generate the corresponding binary matrix $I_B(n)$ of the same size of the original picture with '1's in pixels that are likely to contain ball and '0' in the rest. This was done using an RGB color interval of the ball, giving the results shown in Figure 1 illustrates.

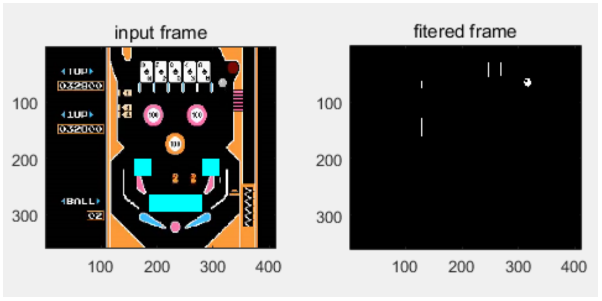

Figure 1. The left image illustrates the n-th input frame with the artificially added occlusions. The right image shows the filtered n-th frame, which shows likely region for the ball highlighted in white.

I propose Kernel Density Extraction (KDE) as an efficient method to solve multiple problems in Object tracking such as: Tuning of noise covariance matrix Q[n] in KF and state estimation.

Next, I applied the 10×10 kernel matrix K to the binary image as

$$I_N(n) = K * I_B(n) \tag{6}$$

where '$*$' stands for the convolution operator and I removed the outer rows and columns of the $I_N(n)$ in order for it to match the dimensions of the original frame. Note that K is a circular kernel which is what I need when the tracking object is round. Next, $I_N(n)$ was used to include the measurement information in the posterior estimation on Kalman filter.

In the following, let me detail my implementations.

In this project, I have considered two different motion models: (i) Constant Velocity and (ii) Constant Acceleration. Moreover, for simplicity I have used stationary matrices for the state transition and the state-to-measurement mapping,i.e.

$$\begin{aligned} A_n &= A \\ H_n &= H \end{aligned} \tag{7}$$

Furthermore, the measurement x_n for the Kalman filter is simply acquired by obtaining the coordinates of the mode of ${I_N}^n$ for all frames.

I shall highlight that I keep the measurement noise covariance C[n] variable. I tune it using the variance of the coefficients of $I_N(n)$. In particular, the higher is the variance (which means that I might have a clear mode in $I_N(n)$) the lower valued we set C[n]. In contrast, when the variance is low I set a high valued C[n]. This way, I have a mechanism to put more weight on the motion model whenever the measurements are poor and vice-versa.

The algorithm of the Kalman Filter is summarized in table 1.

Table 1. KF Algorithm

Algorithm KF ($s[n-1|n-1], M[n-1|n-1], x_k$)
Step 1: Prediction $\hat{s}[n|n-1] \leftarrow As[n-1|n-1]$ $M[n|n-1] \leftarrow AM[n-1|n-1]A^T$+Q[n]
Step 2: Update K[n]$\leftarrow M[n|n-1]H^T(C[n]+HM[n|n-1]H^T)^{-1}$ $s[n|n] \leftarrow \hat{s}[n|n-1] + H_n(x[n] - H_n\hat{s}[n|n-1])$ $M[n|n] \leftarrow (I - K[n]H)M[n|n-1]$

3.1. Constant Velocity Motion Model

For this approach we used a state vector of size 4×1, shown in (8). The first two positions are the pixel coordinates of the object ($p_{n,1}$ and $p_{n,2}$) and the two last are the discrete velocity of the object

($v_{n,1}$ and $v_{n,2}$),

$$s[n] = \begin{bmatrix} p_{n,1} \\ p_{n,2} \\ v_{n,1} \\ v_{n,2} \end{bmatrix} \tag{8}$$

Note that we only consider two dimensions of motion, since we are working with images which are two-dimensional. Moreover, we have $v_{n,1} = v_1$ and $v_{n,2} = v_2$ for all $n = 1, ..., N$.

From physics, for a constant velocity linear model the position p_n can be obtained using the position $p_{n-\Delta t}$, the velocity v and the incremental time difference Δt, the Physics model defines

$$\begin{aligned} p_n &= p_{n-\Delta t} + \Delta t v \\ v &= 1v \end{aligned} \tag{9}$$

In my case, I have that Δt=1. Hence (8) is rewritten as

$$\begin{aligned} p_n &= p_{n-1} + 1v \\ v &= 1v \end{aligned} \tag{10}$$

Since I only observe the position of the object, the size of the measurement vector $x[n]$ is 2×1. So the following defines the measurement from the state

$$x[n] = \begin{bmatrix} x_{n,1} \\ x_{n,2} \end{bmatrix} = Hs[n-1] + w[n] \tag{11}$$

Thus, the state transition matrix is given by

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12}$$

The estimation matrix is given by

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{13}$$

The initial state, i.e. $s[0]$, is characterized by a mean vector $\hat{s}[0|0]$ and a covariance matrix $M[0|0]$. The first is estimated using the first two measurements x_{-1} and x_0.

$$\hat{s}[0|0] = \begin{bmatrix} p_{0,1} \\ p_{0,2} \\ v_{0,1} \\ v_{0,1} \end{bmatrix} = \begin{bmatrix} x_{0,1} \\ x_{0,2} \\ x_{0,1} - x_{-1,1} \\ x_{0,2} - x_{-1,2} \end{bmatrix} \tag{14}$$

and the second is initialized with high coefficients

$$M[0|0] = \begin{bmatrix} 10 & 0 & 0 & 0 \\ 0 & 10 & 0 & 0 \\ 0 & 0 & 10 & 0 \\ 0 & 0 & 0 & 10 \end{bmatrix} \tag{15}$$

Using the algorithm from table 1 the object tracking task can be performed.

3.2. Constant Acceleration Motion Model

This approach considers variability in the velocity according to a constant acceleration value a. The Physics model defines

$$\begin{aligned} p_t &= p_{t-\Delta t} + \Delta t v_{t-\Delta t} \\ v_t &= v_{t-\Delta t} + a\Delta t \\ a &= a \end{aligned} \tag{16}$$

In this regard, the state transition matrix is now given by

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 & 0.5 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{17}$$

The initial estimate is obtained similarly as in the previous case. However, we now need the first three measurements

$$\begin{aligned} \hat{s}[0|0] &= \left[p_{0,1}, p_{0,2}, v_{0,1}, v_{0,1}, a_1, a_2\right]^{\mathrm{T}} \\ &= \begin{bmatrix} x_{0,1} \\ x_{0,2} \\ x_{0,1} - x_{-1,1} \\ x_{0,2} - x_{-1,2} \\ (x_{0,1} - x_{-1,1}) - (x_{-1,1} - x_{-2,1}) \\ (x_{0,1} - x_{-1,1}) - (x_{-1,1} - x_{-2,1}) \end{bmatrix} \end{aligned} \tag{18}$$

Using the algorithm from table 1 the object tracking task can be performed.

3.3. Algorithm Description

The algorithm is been implemented in Matlab using different functions as described.

1) main.m

This is the entry point for the algorithm. The file does the initialization, generates the measurements, and run the Kalman filter and display results.

2) KalmanInit.m

Inputs: update value of Q(param), choose between motion model(mmodel)

Outputs: measurement noise(R), process noise(Q), state transition matrix(A), measurement matrix(H)

This function initializes the parameters of the Kalman Filter using a constant speed or constant acceleration motion model.

3) KalmanPredict.m

Inputs: Previous state mean ($\hat{s}[n-1|n-1]$), Previous covariance($M[n-1|n-1]$), state transition matrix(A), process noise(Q)

Outputs: Predicted mean($\hat{s}[n|n-1]$), Predicted covariance($M[n|n-1]$)

This function performs the prediction script using Kalman Filter.

4) KalmanUpdate.m

Inputs: Predicted mean($\hat{s}[n|n-1]$), Predicted covariance($M[n|n-1]$), state transition matrix(A), measurement noise(R), measurements($x[n]$)

Outputs: Updated mean($\hat{s}[n|n]$), Updated variance($M[n|n]$)

This function performs the update script using Kalman Filter.

5) KalmanAlgorithm.m

Inputs: number of current frame(count), frame after adding the occlusions(vidFrame), Previous state mean ($\hat{s}[n-1|n-1]$), Previous covariance($M[n-1|n-1]$), state transition matrix(A), process noise(Q), measurements(x[n])

Outputs: Updated mean($\hat{s}[n|n]$), Updated variance($M[n|n]$)

This function performs the Kalman Filter algorithm. When we are at count=0, obtain the measure and set the initial position coordinates. When we are at count=1, we can obtain the initial speed as the difference of the position at count=1 and count=0.

If we choose the constant speed motion model. Next, we initialize the initial covariance matrix with an arbitrary value.

If we choose the constant acceleration motion model constant acceleration. When we are At count=2, we can obtain an estimation of the initial acceleration by taking the difference of the velocity at count = 1, count = 0. Next, we also initialize the initial covariance matrix with an arbitrary value.

6) Video-editing.m

Input: current frame occlusions

Output: current frame with occlusions

This function is used to add the occlusions to each frame.

7) imageTranformation.m

Inputs: current frame(original-im), colour threshold to decide the colour filtering(colour-thres), identify a certain color(c-thres),

Output: Binary image(out)

This function is used to Filters the image and transforms it in a binary image. White will represent the most likely regions of target object's pixels.

8) rec-size.m

Inputs: size of the current frame(xp,yp), the Updated mean (centroidx, centroidy), the assumed size(distance)

Outputs: the maximum sizes of the rectangle that will enclose the ball (max-distance-x-K, max-distance-y-K)

The function computes the maximum possible size of the rectangle that envolves the estimate with the input conditions given.

9) KernelFunction.m

The function returns the Kernel matrix.

10) mse.m

The function computes the prediction error, compared to the actual state of the system.

4. RESULTS

In this section I show and briefly discuss the results obtained when using the method explained so far for object tracking. As already highlighted in the previous section, I will use a 1 minute length video which captures a Nintendo Pinball game play. However, I restrict the analysis to the first 15 seconds, since it is enough to gain some insights about the performances of the kalman filter.

In the following, I put videos into the project file where you can see how each method performed:

- KF constant velocity motion model(constant v.mp4)
- KF constant acceleration motion model(constant a.mp4)

Table 2 lists the MSE of the estimations for the first 15 seconds where no particle deprivation happened. To compute the MSE, I use as a reference image processing tools comparing pixel values and removed the artificially added occlusion regions. Furthermore, Figure(4) illustrate the squared error for all frames within the 15 second sequence.

Table 2. Results without occlusion regions

Model	Mean square error
Constant Velocity	9.917
Constant Acceleration	9.918

In addition, Table 3 shows the MSE results when particle deprivation happens. As expected, I observe decrease in the performance. But it's not notable. Furthermore, Figure(5) illustrate the squared error for all frames within the 15 second sequence.

5. DISCUSSION

When occlusion intervals are shown in Figure2 and Figure3 as peaks in the squared error, then constant velocity and constant acceleration performances become poor . This is due to the fact that the particle cloud loses track of the ball and is not able to obtain new accurate measurements. In contrast, without occlusion, the squared errors of the two models are much small all time shown in Figure4. From table 2 and table 3, in general, I observe that the Kalman Filter is less affected by occulusion. One of the keys here has been the tuning of Q[n] when there are occlusions, previously explained. I can assert that kernels can be very helpful in the filtering of the image in order to obtain suitable measurements.

Table 3. Results with occlusion regions

Model	Mean square error
Constant Velocity	36.18
Constant Acceleration	56.69

References

[1] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35-45, 1960.

[2] Alper Yilmaz, Omar Javed, and Mubarak Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006.

[3] Ted J Broida and Rama Chellappa. Estimation of object motion parameters from noisy images. *IEEE transactions on pattern analysis and machine intelligence*, (1): 90-99, 1986.

[4] Duc Phu Chau, Francois Bremond, and Monique Thonnat. Object tracking in videos: Approaches and issues. *arXiv preprint arXiv* :1304.5212,2013.

[5] Kay, Steven M. Fundamentals of Statistical Processing, Volume I: Estimation Theory. PTR Prentice hall, 1993.

[6] G. Welch and G. Bishop, ”An Introduction to the Kalman Filter”, *In Pract.* vol.7, no.1, pp.116, 2006.

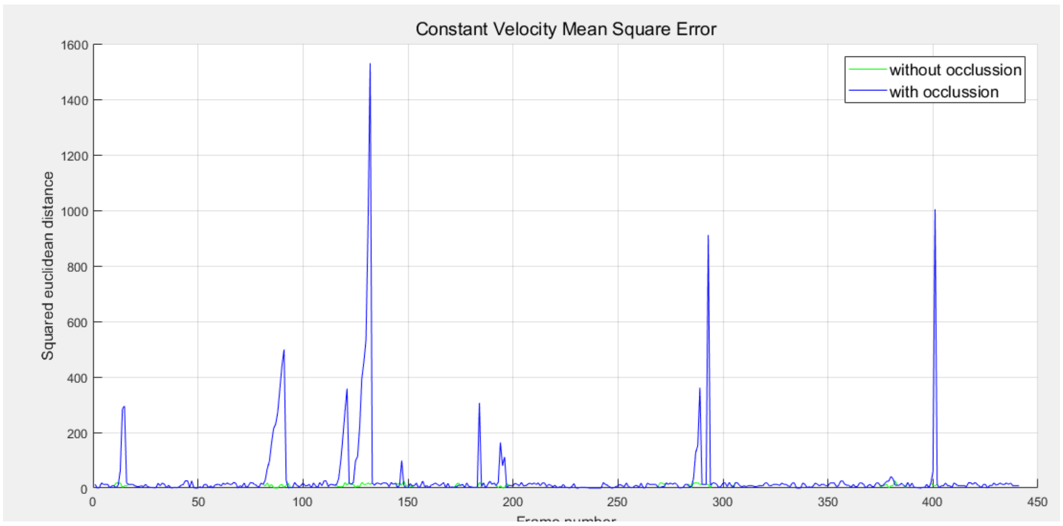

Figure 2. Constant Velocity Mean Square Error

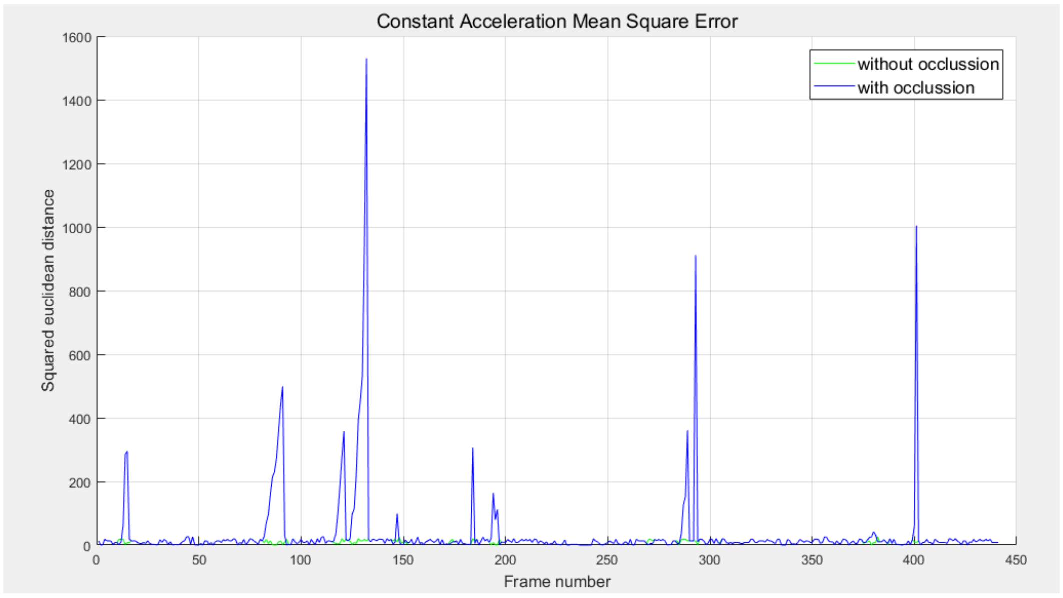

Figure 3. Constant Acceleration Mean Square Error

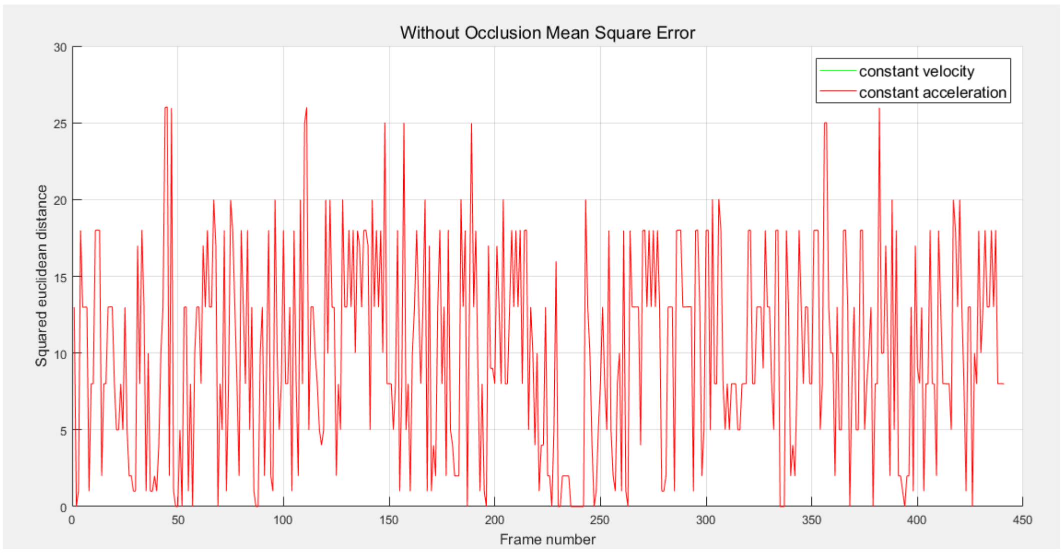

Figure 4. Without Occlusion Mean Square Error

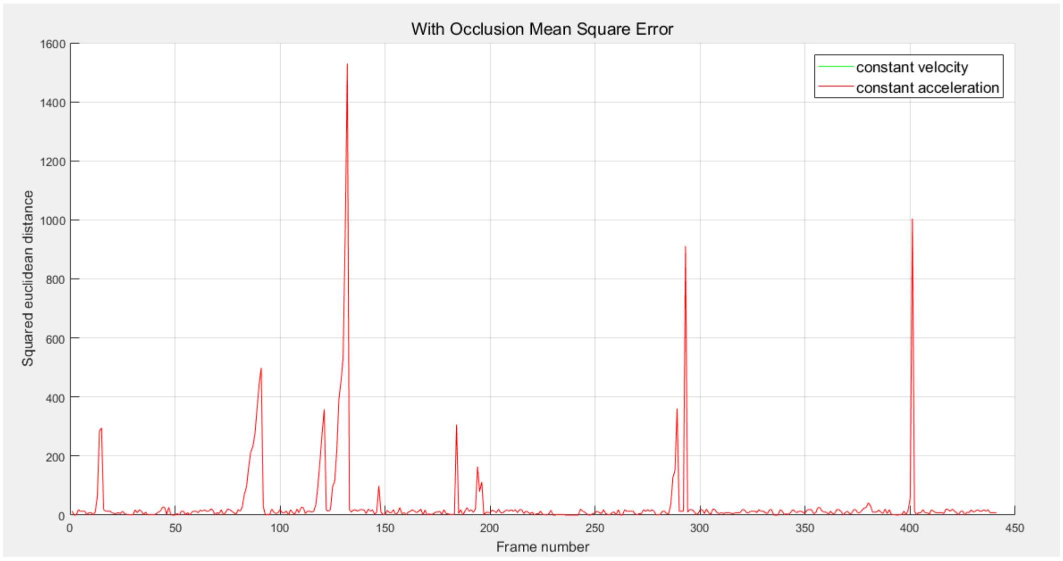

Figure 5. With Occlusion Mean Square Error.