Object Tracking Using Kalman Filter

Shahin Khobahi

I. Introduction

In this project, we are proposing an adaptive filter approach to track a moving object in a video. Currently, object tracking is an important issue in many applications such as video survelance, traffic management, video indexing, machine learning, artificial intelligence and many other related fields. As we will discuss in the following sections, moving object tracking can be interpreted as an estimation problem. Kalman filter is a powerful algorithm that can be used in the state estimation problems and that is the reason we used this method to estimate and predict the position of a moving object. In the first stage of this project we use the background subtraction method to detect the moving object in the video and then we use the Kalman filter ro predict and estimate the next state of the object.

II. Problem Formulation

A. Object Detection Using Background Subtraction

A video is composed of a series of frames each which can be considered as a 2D signal. So, a video can be seen as a two-dimensional (2D) signal through time. Moreover, there are two types of objects in a video: *steady and moving objects*. Steady objects do not change from frame to frame and can be considered as the background scene. The goal is to detect a moving objects from the steady ones. First let us provide an example: consider that you are looking at a wall and suddenly a birdy fly over this wall. The steady object in this scene is the wall and the moving object is the birds. The bird is in fact disturbing your observation of the wall(background), so in the context of signal processing, this bird(moving object) can be seen as a noise to that background. In other words, in a video, a moving object is like a noise to the background scene which is a fixed signal, and this moving object is adding noise to our observation of that background. Consequently, each frame of the video can be interpreted as a noisy observation of the background. Therefore, the problem is just simply the noise detection in a signal. The following model can be used for our problem:

$$\mathbf{y} = \mathbf{x} + \mathbf{v} \tag{1}$$

Where y is our noisy measurement of x (background signal), and v denotes the disturbance which in fact is our moving object disturbing that background signal x. As it was mentioned earlier, we need to extract the noise v from our noisy signal y (the video). Each frame of the video is a noisy realization of the signal y and we refer to the i-th frame of the video as u_i. Further we assume that the video has N frames. Our approach of extracting the noise from the observations u_i is to first obtain an *estimation* of the background signal $\hat{x}$, then we subtract each observation u_i from the estimated signal $\hat{x}$ to obtain an estimation of the noise at each frame:

$$\hat{v}_i = u_i - \hat{x} \tag{2}$$

Given two deterministic random variables $\{\mathbf{x}, \mathbf{y}\}$, we define the least-mean-squares estimator (l.m.s.e) of x given y as the conditional expectation of $\mathbf{x}$ given $\mathbf{y}$:

$$\hat{x} = E(\mathbf{x}|\mathbf{y}) = E[\mathbf{x}|u_0, u_2, ..., u_{N-1}] \tag{3}$$

For simplicity, we assume that we model x as an unknown constant instead of a random variable. Further we assume that we are given N frames of the video and that is all the information we have. Now, we model our problem as follows:

$$\mathbf{y(i)} = x + \mathbf{v(i)}, \quad i = 0, 1, \ldots, N-1 \tag{4}$$

and we define the column vector $\mathbb{1} = [1, 1, \ldots, 1]^T$. Then,

$$\mathbf{y} = \mathbb{1}x + \mathbf{v} \tag{5}$$

if this is the case, according to Gauss-Markov theorem, the optimal linear estimator (m.v.u.e) of x is:

$$\mathbf{\hat{x}_{mvue}} = \frac{1}{N}\sum_{i=0}^{N-1}\mathbf{y(i)} = \frac{1}{N}\sum_{i=0}^{N-1}\mathbf{u_i} \tag{6}$$

Namely, (6) means that the optimal linear estimator of x given $\{\mathbf{y}(i)\}$, is simply the mean of the samples(measurements). So, in order to obtain an estimation of the background of the video, we take the average of all frames and store it as the background scene. Fig. 1 illustrates 4 frames of a sample video, these samples are in fact 4 noisy measurements of our signal, and that yellow ball which we are trying to track acts as the disturbance to the background of the video(the door and the wall). Fig. 2 provides the background scene that is the result of averaging over all of the frames. Please note that in this project we are assuming that the background does not change, so sample-mean estimator is a good estimation of the background. However, in the case of real-time tracking where the background is not changing, one can feed the average estimator as the frames arrives; evidently, in this case, estimation improves with time.

Now that we obtained $\hat{x}$, we can extract the noise from the signal(video) by subtracting each frame from the background. Fig. 3 provides four realization of the noise(moving object) at different frames. Due to the fact that in our problem we do not care about the energy of the noise - the gray level of an image(pixels) is proportional to the energy or the amount of information it contains (entropy of the image) - we can use *one-bit Compressed Sensing* method to store the noise. That is, we use the following model to store the noise :

$$\mathbf{z_i} = sgn(\mathbf{v_i} - \tau_i), \quad i = 0, 1, \ldots, N-1 \tag{7}$$

Where $\mathbf{z_i}$ denotes the quantized measurement of $\mathbf{v_i}$ with respect to the threshold τ_i. Namely, instead of saving the real value of the measurement v_i, we only save the sign of it with respect to the defined threshold τ_i at that measurement. In practice, one may use an *adaptive* threshold but in our application we use a fixed threshold at all of the measurements. Also, it is worth to mention that in the 1-bit compressed sensing model (7) we lose all of the information about the magnitude of $\mathbf{v_i}$ but as we mentioned earlier we do not care about the energy of the noise, so this model can be used to store the measurements and improve the speed of the tracking process and also it lowers the dimension of the calculations(an image typically has the intensity information of the colors: Red, Green, and Blue and after quantizing the measurements based on (7) we only has one dimension which is known as *binary image*).

Fig. 4, illustrates one estimation of the noise after quantization. It can be observed that we have an anomaly at the bottom right corner of the Fig. 4, which emphesized the fact that this is not an exact realization of the noise(moving object) but instead it is an estimation of the noise and is prone to some errors.

Now that we estimated the moving object, we can easily find the center of the object by inspecting the binary image of each frame and find the area that contains more 1 and choose the larger area as the object. Then we can estimate the center of the area, and store it as the position of the moving object at each frame. Eventually, in Fig. 5, you can see that we detected the moving object at each frame. The yellow circle denotes our detection.

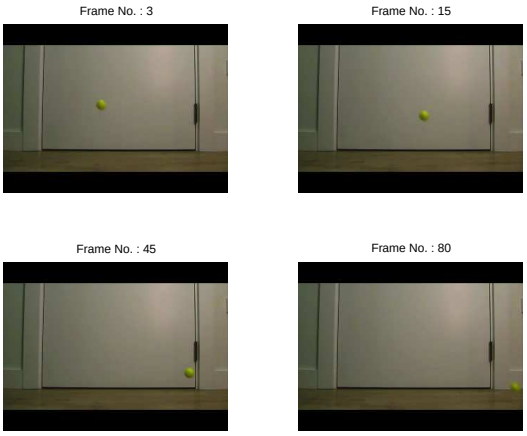

Fig. 1. Four sample frames of the video. In other words, these are four noisy measurements of the background (which is a 2D signal).

B. Kalman Filter

In this section we describe the formulation and system model for Kalman filter.

Intutitively, Kalman filter takes the current state of your system, and makes a prediction based on the current state and current uncertainty of our measurements, and make a prediction for the next state of the system with anuncertainty.

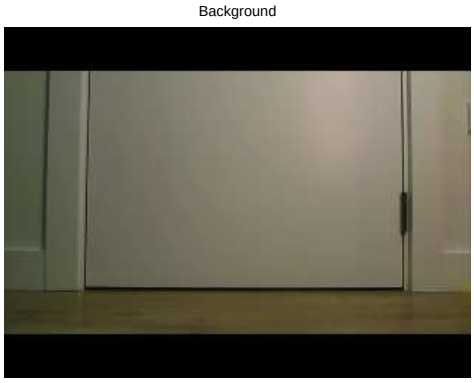

Fig. 2. An estimation of the background signal resulted from averaging over all frames.

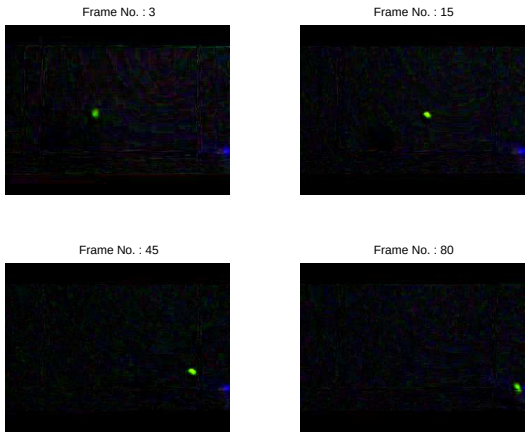

Fig. 3. Four realizations of the noise (moving object) at different frames.

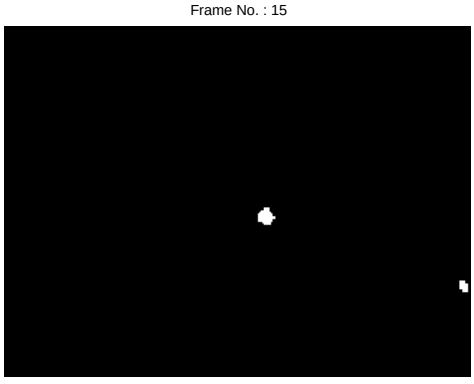

Fig. 4. One realization of the noise after quantization.

Then, it compares its prediction with the received input and correct it self upon the error.

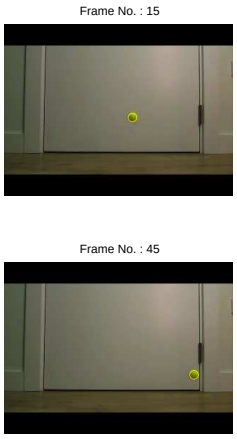

Fig. 5. Four sample frames of the video. In other words, these are four noisy measurements of the background (which is a 2D signal).

First we need to define our state for the Kalman filter. We want to predict the position of a moving object based on the current information of the object. For simplicity we assume a constant velocity model for our problem. The dynamics of a moving object in one-dimension can be described as follows:

$$x_t = \frac{1}{2}aT^2 + v_{t-1}T + x_{t-1} \tag{8}$$

$$v_t = aT + v_{t-1} \tag{9}$$

Where x_t and v_t denotes the position and velocity at time t, and a denotes the acceleration. So, the dynamics of a moving object in one-dimension can be modeled by the position and the first derivation of it. Without losing the generality, we can extend the one-dimensional case to a 2D object and conclude that the dynamics of a two-dimensional object can be described by x, y, $\dot{x}$, and $\dot{y}$.
We define the state X_t with the following variables of interest:

$$X_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} \tag{10}$$

Next, we need to see what is the expected behaviour of our variables when we are going from one state to another. Based on Eq. (8) and (9), we define the following behaviour for the system variables:

$$\begin{aligned} x_t &= x_{t-1} + \dot{x}_{t-1}T + \frac{1}{2}aT^2 \\ y_t &= y_{t-1} + \dot{y}_{t-1}T + \frac{1}{2}aT^2 \\ \dot{x}_t &= \dot{x}_{t-1}T + aT \\ \dot{y}_t &= \dot{y}_{t-1}T + aT \end{aligned}$$

So, the following model can be used to define the state transition:

$$\begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ \dot{x}_{t-1} \\ \dot{y}_{t-1} \end{bmatrix} + \begin{bmatrix} \frac{1}{2}T^2 \\ \frac{1}{2}T^2 \\ T \\ T \end{bmatrix} .a + \mathbf{W_{t-1}} \tag{11}$$

We can formulate (11) as follows:

$$X_t = AX_{t-1} + Bu_{t-1} \tag{12}$$

Where Bu_{t-1} can be seen as the noise(or external force on the acceleration).
In this project, we are observing the position of the moving object. Therefore, we define the following measurement matrix H:

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

The measurement matrix is:

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix} + \mathbf{V_t} \tag{14}$$

Where $\mathbf{V} = [\mathcal{N}(0, \sigma_1^2), \mathcal{N}(0, \sigma_2^2)]^T$ is the measurement noise. Basically, Kalman filter has three noise covariance matrices:

- **Dynamic Noise**: During transition from one state to another, the system can be disturbed by an external force and add noise to the system. An external force can be modeled as a disturbance to the object acceleration in our problem. It contributes to the prediction of the next error covarinace matrix.
- **Measurement Noise**: All of our sensors are prone to noise and consequently will lead to a corruption of our measurements. We refer to this disturbance as the Measurement Noise.
- **Covariance of State Variables**

Assuming that the state variables are independent, we initialize the covariance matrix of state variables as follows. Please note that we can also consider this matrix as *posteriori* error covariance matrix.

$$S_t = \begin{bmatrix} \sigma_x^2 & 0 & 0 & 0 \\ 0 & \sigma_y^2 & 0 & 0 \\ 0 & 0 & \sigma_{\dot{x}}^2 & 0 \\ 0 & 0 & 0 & \sigma_{\dot{y}}^2 \end{bmatrix} \tag{15}$$

Also, we further assume that the measurement noises are independent, then the covariance matrix of $\mathbf{V}$ can be described as:

$$cov(\mathbf{V}) = R = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix} \tag{16}$$

Finally we need to define the covariance matrix of dynamic noise. As it was described earlier, this noise represents the disturbance during transition from one state to another. It can be written as:

$$Q = \begin{bmatrix} \sigma_x^2 & 0 & \sigma_{x\dot{x}} & 0 \\ 0 & \sigma_y^2 & 0 & \sigma_{y\dot{y}} \\ \sigma_{x\dot{x}} & 0 & \sigma_{\dot{x}}^2 & 0 \\ 0 & \sigma_{y\dot{y}} & 0 & \sigma_{\dot{y}}^2 \end{bmatrix} \tag{17}$$

From (11), we can define Q as:

$$Q = \begin{bmatrix} \frac{1}{4}T^4 & 0 & \frac{1}{2}T^3 & 0 \\ 0 & \frac{1}{4}T^4 & 0 & \frac{1}{2}T^3 \\ \frac{1}{2}T^3 & 0 & T^2 & 0 \\ 0 & \frac{1}{2}T^3 & 0 & T^2 \end{bmatrix} \tag{18}$$

We assume that our original tracker (section II.a) is used as the input to the Kalman filter. We define the input vector as:

$$Y_t = \begin{bmatrix} \hat{x}_t \\ \hat{y}_t \end{bmatrix} \tag{19}$$

We defined all of the required matrices for Kalman filter. Now we can use the Kalman filter based on the following algorithm to predict the position of the moving object based on our original tracker (section II.a) as the input to the filter. Kalman filter, has two stages: *prediction* and *correction* :

$$Prediction:$$
$$X_t = AX_{t-1} + Bu$$
$$S_t = AS_{t-1}A^T + Q$$
$$Correction:$$
$$K_{t-1} = S_{t-1}H^T(HS_{t-1}H^T + R)^{-1}$$
$$X_{t+1} = X_t + K_{t-1}(Y_t - HX_t)$$
$$S_{t+1} = (I - K_tH)St$$

III. Results

In order to observe the behaviour of Kalman filter under different circumstances, we considered three different cases to examine the Kalman filter in object tracking. In the following subsections, we examine each of these cases.

A. Scenario 1: Prediction

The first scenario is the case that we are sensing the position of the object every 3 frames and we want to have a good prediction of the position of moving object based on these samples. Fig. 6, illustrates the result in four different frames. The yellow circle is our main tracker(which is used as the input to the Kalman filter every 3 frames) and the black circle is the prediction of Kalman filter. It can be observed that the Kalman filter is tracking the moving object with a very good accuracy.

B. Scenario 2: Prediction In The Presence of Noise

In this scenario, we add a large noise to the input of the Kalman filter. It turns out that the Kalman filter is more robust to the noise than the original tracker. So, if we have our measurements aren corrupted by noise, one can use the Kalman filter to obtain a better estimation than each of the sensors (*data fusion*) because this algorithm is an adaptive filter and is more robust to the noise than each of the sensors. Fig. 7, illustrates this scenario. It can be seen that, the yellow circle is jumping around and is far from the object. However, the Kalman filter has a better estimation of the position. Please note that, a low gain will smooth out the noise but also lowers the speed of Kalman filter (it will detect the changes more slowly).

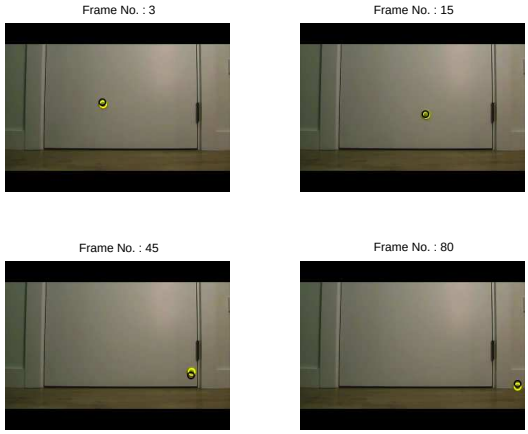

Fig. 6. Scenario 1 in which the Kalman filter tracks the moving object when it is feeded every three samples.

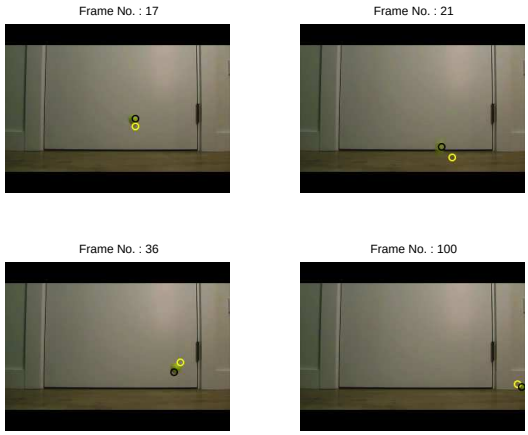

Fig. 7. Scenario 2 in which the Kalman filter tracks the moving object in the presence of a large noise.

C. Scenario 3: Blind Prediction

In this case, we let the Kalman filter to learn for half of the frames and then we did not update the input for the filter. In (10) we defined the dynamic of the system for the constant velocity object. That is, we are not capturing the acceleration of the system. So, we should expect that the Kalman filter can not track the trajectory of the ball because the object is under the gravity and has a negative vertical acceleration. If we want to track the trajectory of the without the input, we must use a more complex system model as follows:

$$X = \begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix} \tag{20}$$

Fig. 8 provides the result of this scenario. As you can see, Kalman filter is not able to track the moving object after cutting the input and it tracks a linear path after that.

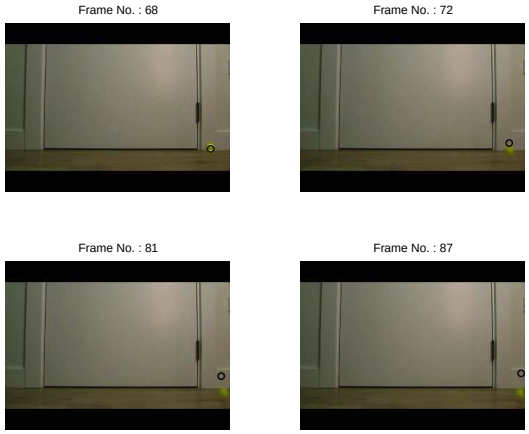

Fig. 8. Scenario 3 in which the Kalman filter blindly track the moving object.

IV. Conclusion

In this project we designed a Kalman filter to track a moving object in a video. In fact, as it was mentioned earlier, a moving object in a video can be seen as a noise to the background scene. So, this project was simply a noise detection based on Kalman filter. The same approach can be used to estimate and cancel out the noise of other signals. As we saw in the scenario 1 and 2, Kalman filter can be used whenever we need to predict the next state of a system based on some noisy measurements. Also, it can be used for sensor fusion as well. It must be mention that this algorithm is defined for linear systems(we used linear algebra). In the case if nonlinear systems, the extended Kalman filter (EKF) which is a nonlinear version of Kalman filter can be used.