EEE: 515 - Machine Vision - Homework 3

Praveen Paidi
Student ID: 1225713099
Fall 2023

December 2, 2023

Problem status

1. **Problem 1 : Completed**

2. **Problem 2 : Completed**

1 Problem 1

1.1 Part 1

I took 15 images of checkerboard for camera calibration.The following are the some of the images taken from phone camera used for the camera calibration.

Figure 1: Checkerboard image

Figure 2: Checkerboard image

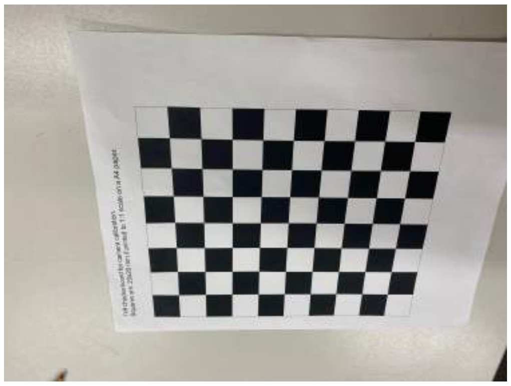
Figure 3: Checkerboard image

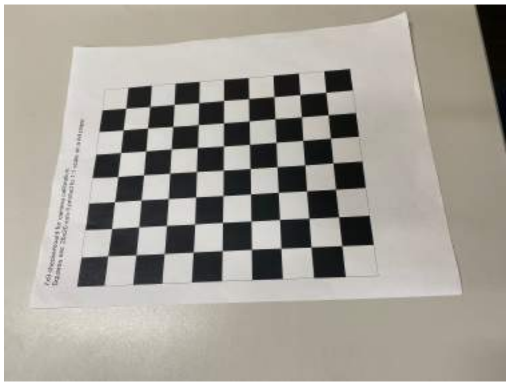
Figure 4: Checkerboard image

Checkerboard is of size 7x9 for camera calibration.
Hyper parameters:
Number of iterations: 35
Error epsilon = 0.001

I used internal function in listing 1 for corners deetction and drawing of corners.

Listing 1: Corners

```
cv.findChessboardCorners(gray, (7,9), None)
cv.drawChessboardCorners(img, (7,9), corners2, ret)
```

Images with corners being drawn is shown in the following images.

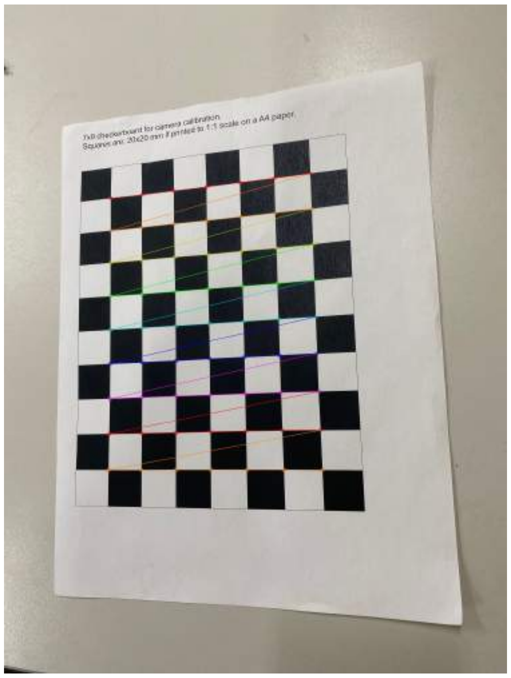
7x9 checkerboard for camera calibration.
Squares are: 20x20 mm if printed to 1:1 scale on a A4 paper.

Figure 5: Checkerboard image Corners

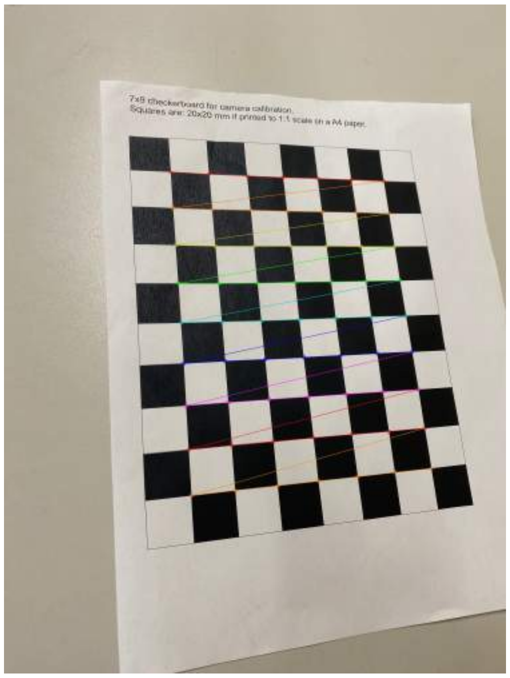
7x9 checkerboard for camera calibration.
Squares are: 20x20 mm if printed to 1:1 scale on a A4 paper.

Figure 6: Checkerboard image Corners

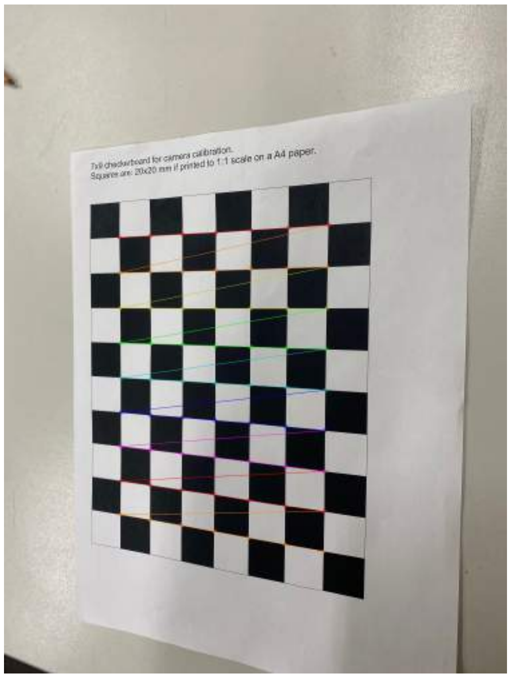
7x9 checkerboard for camera calibration.
Squares are: 20x20 mm if printed to 1:1 scale on a A4 paper.

Figure 7: Checkerboard image Corners

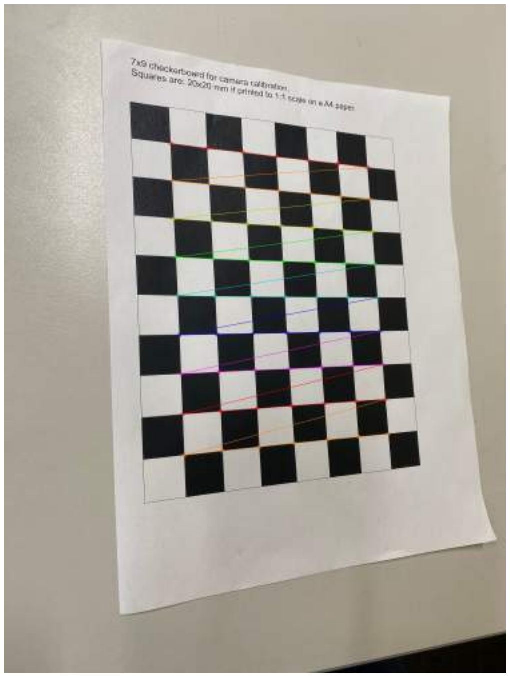
7x9 checkerboard for camera calibration.
Squares are: 20x20 mm if printed to 1:1 scale on a A4 paper.

Figure 8: Checkerboard image Corners

Now once I established 2D corners points on image and respective points in 3D. I calculated camera calibration matrix using listing 2.

Listing 2: Corners

```
cv.calibrateCamera(objpoints, imgpoints, gray.shape[::-1], None, None)
```

The resulting intrinsic matrix is

$$\begin{bmatrix} 1.50843078e+03 & 0.00000000e+00 & 7.60167641e+02 \\ 0.00000000e+00 & 1.50272747e+03 & 1.00963246e+03 \\ 0.00000000e+00 & 0.00000000e+00 & 1.00000000e+00 \end{bmatrix} = \begin{bmatrix} 1508.43 & 0 & 760.16 \\ 0 & 1502.72 & 1009.63 \\ 0 & 0 & 1 \end{bmatrix}$$

From camera matrix $\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$

Comparing we get $f_x = 1508.43$
$f_y = 1502.72$
$c_x = 760.16$
$c_y = 1009.63$

Reprojection error:
Projected the obtained 3D corner points and compared to obtained 2D corner points and I performed least square error.The resulting error is 0.0722.

Reference: Calibration link
Other Aspects: Hyperparameters can be tuned for different grid size and can vary number of iterations.

1.2 Part 2

For this, I used SIFT descriptors with FLANN based matcher and ratio test. The following are some of the images of object.

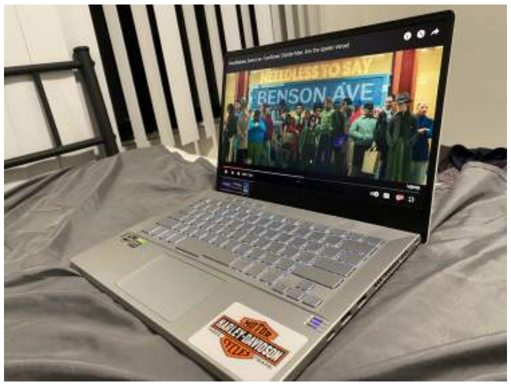
Figure 9: Object image

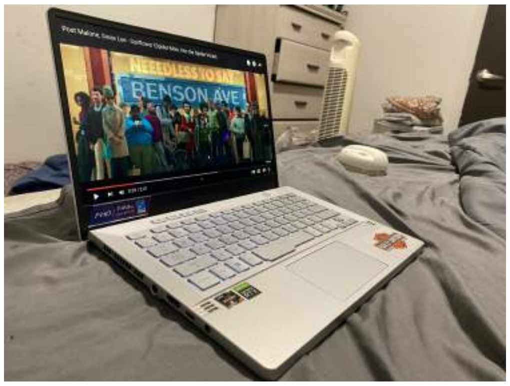
Figure 10: Object image

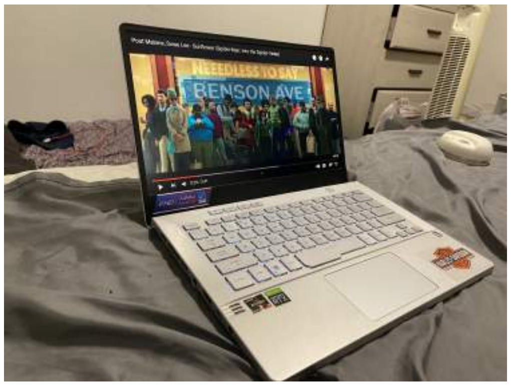
Figure 11: Checkerboard image

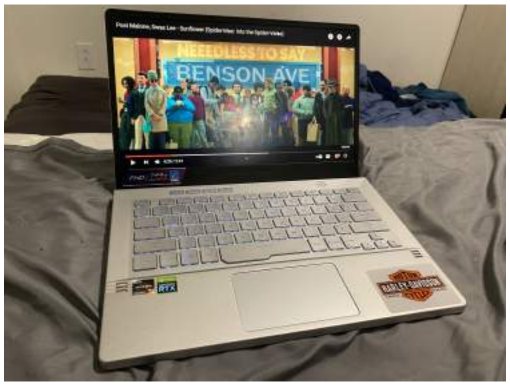
Figure 12: Checkerboard image

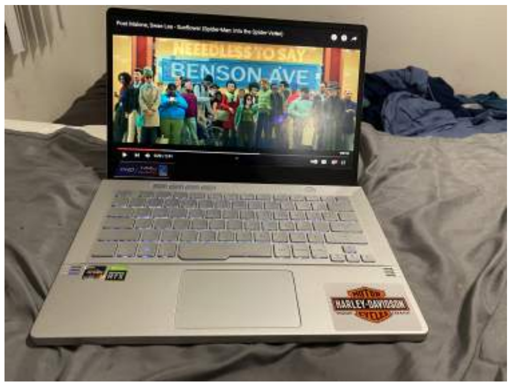
Figure 13: Object Image

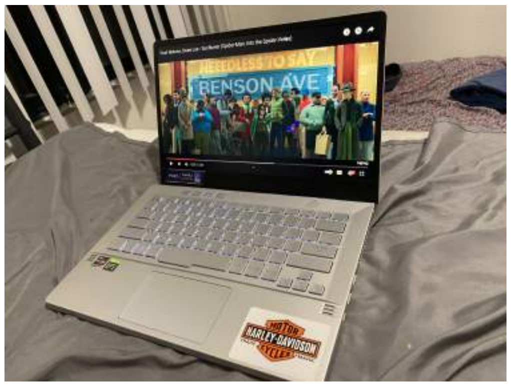
Figure 14: Object Image

The two images picked are fig 9 and fig 10 for epipolar lines display.Listing 3 shows important functions being used for Fundamental matrix.

Listing 3: Fundamental Matrix

```
kp1, des1 = sift.detectAndCompute(img1,None)
kp2, des2 = sift.detectAndCompute(img2,None)
flann = cv.FlannBasedMatcher(index_params,search_params)
F, mask = cv.findFundamentalMat(pts1,pts2,cv.FM_LMEDS)
```

Fundamental Matrix:

$$\begin{bmatrix} 6.17023998e-08 & 1.73991185e-07 & -2.07777209e-04 \\ 1.93216993e-07 & -1.79895439e-08 & 1.62073572e-04 \\ -3.74992546e-04 & -8.75455444e-04 & 1.00000000e+00 \end{bmatrix}$$

The following is cv function for epipolar lines drawing shown in listing 4.

Listing 4: Epipolar Lines

```
lines1 = cv.computeCorrespondEpilines(pts2.reshape(-1,1,2), 2,F)
```

The resulting image is shown in figure 15:
Epilines are converging at a point outside the images. That meeting point is the epipole.

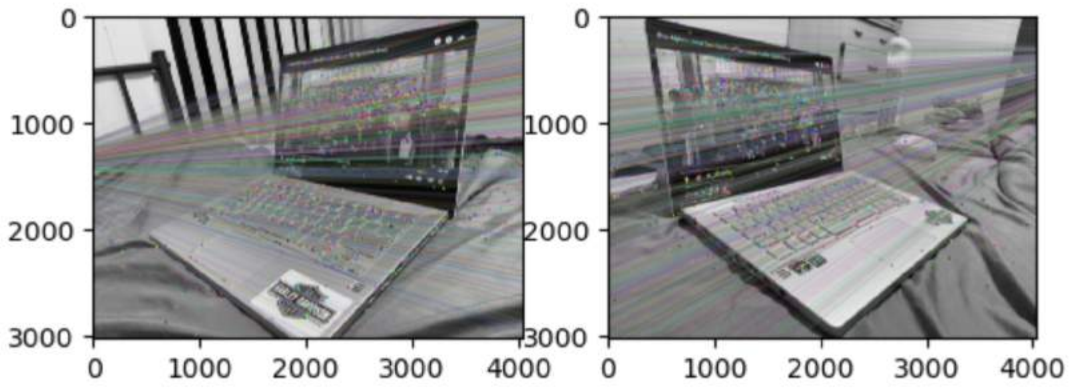

Figure 15: Epipolar lines

Other aspects: Out of the images I used, these form better visual representation.For better epipolar intersection, images with many non-planar points can be used.

Reference Epiploar lines link

1.3 Part 3

Essential Matrix:
The relationship between the essential matrix (E) and the fundamental matrix (F) given the camera calibration matrices (K1 and K2) is expressed as $E = K_2^T.F.K_1$
As the both are taken by same camera applying $E = K^T.F.K$
The resulting Essential matrix is

$$\begin{bmatrix} 0.14039538 & 0.39439632 & 0.02231589 \\ 0.43797662 & 0.04062381 & 0.43697519 \\ -0.20063673 & -1.14411028 & 0.13588962 \end{bmatrix}$$

2 Problem 2

2.1 Part 1

Implementation of kalman filter:

$x_k = x_{k-1} + w_k$ with w have noise variance 0.00001.

$z_k = x_k + v_k$ with v has noise variance 0.0001.

The simulation involved a true constant voltage of 0.44 Volts. Both process noise (with a variance of 1e-5) and measurement noise (with a variance of 0.0001) were considered. The Kalman filter was utilized to predict and update the voltage estimate in each time step. The noisy measurements, generated using a normal distribution centered around the true voltage with a standard deviation corresponding to the measurement noise, were visually depicted alongside the ground truth and the output of the Kalman filter

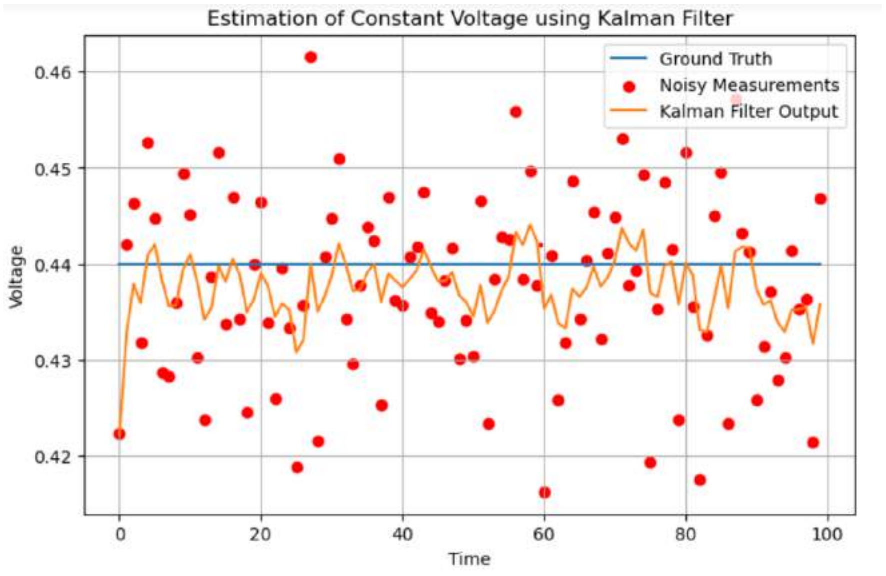

Figure 16: Checkerboard image

Cux code in listing 5.

Listing 5: Kalman Filter

```
x_k = x_k_minus_1
P_k = P_k_minus_1 + noise_var_process
# Update step
K_k = P_k / (P_k + noise_var_measurement)
x_k = x_k + K_k * (z_k - x_k)
P_k = (1 - K_k) * P_k
# Update variables for next iteration
x_k_minus_1 = x_k
P_k_minus_1 = P_k
```

Increasing noise in the process (w_k) will lead to a higher variance in the predicted state (x_k), causing the Kalman Filter to rely more on measurements rather than the predicted state. This might result in more sensitivity to measurement noise.The predicted is becoming sensitive being shown in figure 17

Increasing noise in the measurements (v_k) will cause the Kalman Filter to give less weight to the measurements, relying more on the predicted state. This might lead to smoother estimates but might also introduce more lag in responding to changes in the true voltage.Low repsonse being shown in figure 18.

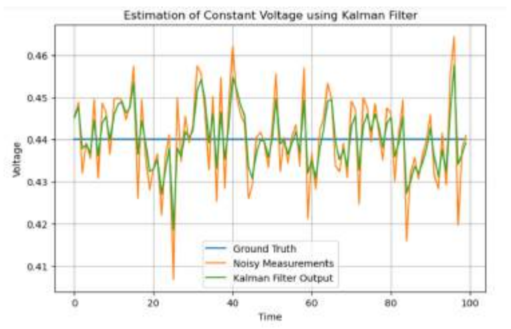

Figure 17: Process Noise Increase

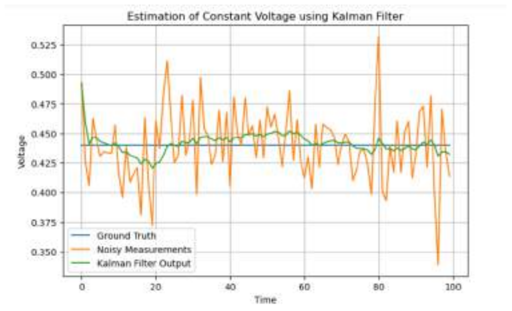

Figure 18: Measured Noise Increase

OTHER ASPECTS

Some Other Plots:

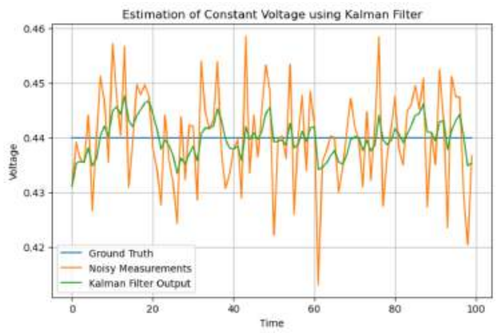

Figure 19: Kalman filter predicted values plotted with respective ground truth and measurement values(Same as figure 16 except scatter plot)

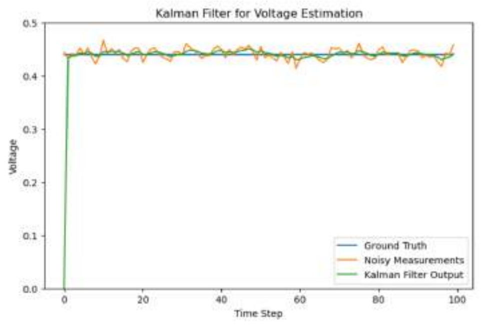

Figure 20: Kalman filter values starting from initial value 0 as there is no prediction available for beforehand for first time step.Each value at each time step being shown.

2.2 Part 2

Tracking 1D object $\begin{bmatrix} x \\ \dot{x} \end{bmatrix}$

$X_k = A.X_k + W_k$; $Z_k = H.X_k + V_k$; A = $\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$

W has Diagnol covariance matrix = 0.00001. V has Diagnol covariance matrix = 10.

Crux code in listing 6.

Q and R being Process noise and measurement noise in the code.

Listing 6: Kalman Filter

```
X = np.dot(A, X) + np.random.multivariate_normal([0, 0], Q)
measurement = np.dot(H, X) + np.random.multivariate_normal([0], R)
X = np.dot(A, X)
P = np.dot(np.dot(A, P), A.T) + Q
K = np.dot(np.dot(P, H.T), np.linalg.inv(np.dot(np.dot(H, P), H.T) + R))
X = X + np.dot(K, (measurement - np.dot(H, X)))
P = P - np.dot(np.dot(K, H), P)
```

The following are the results of the kalman filter being shown in below figures.
If X = [0 0] as intial state

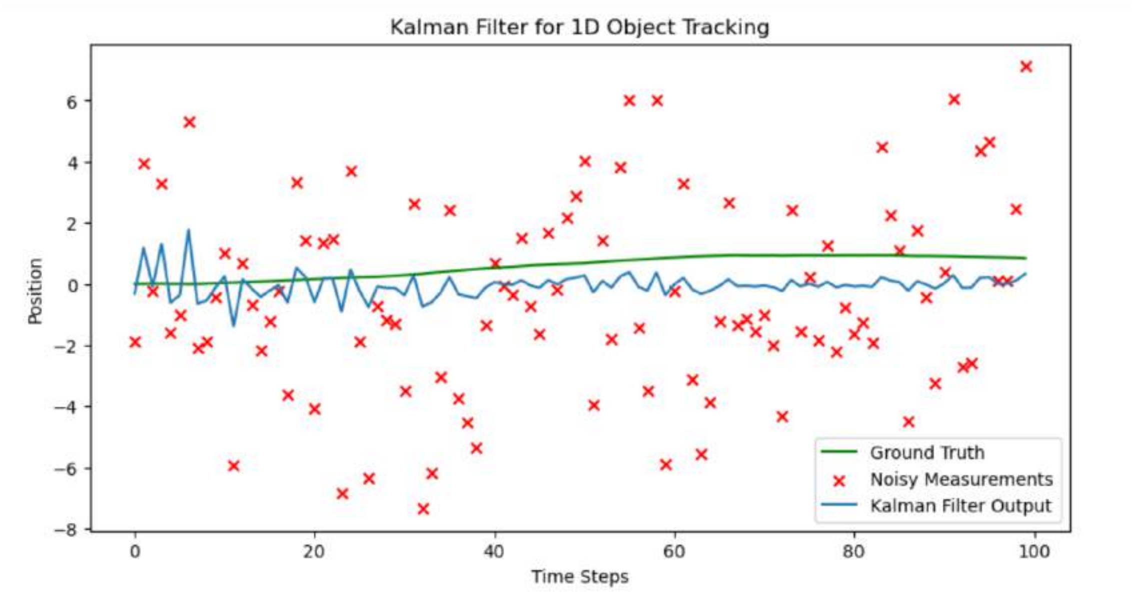

Figure 21: Position Kalman filter

If X = [0 1] as intial state

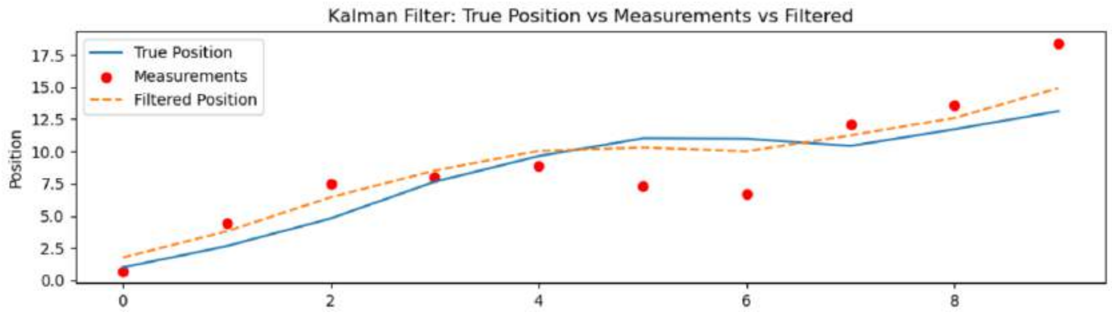

Figure 22: Position Kalman filter

Effect of process noise:
Consequently, the Kalman Filter might rely more on measurements to correct these uncertain predictions, potentially leading to increased reliance on measurements and quicker adaptation to changes.

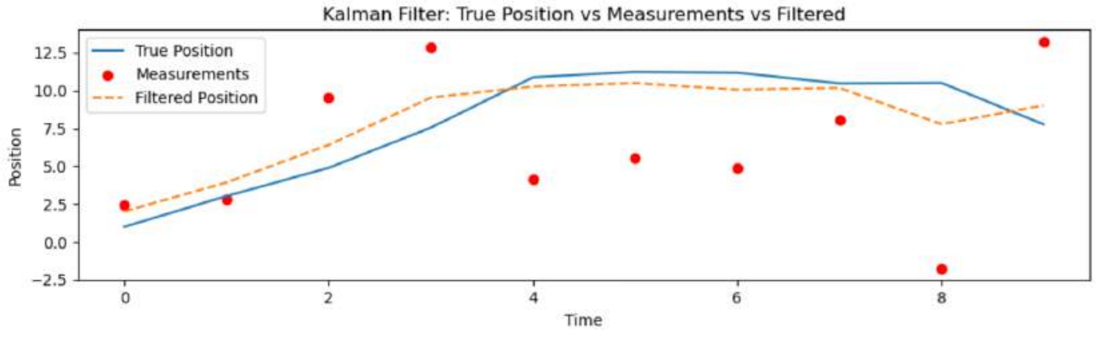

Figure 23: HIGH V

Effect of Measured Noise:

It makes filter to largely ignore measurements, resulting in slower adaptation to changes and a higher reliance on the model's predictions.

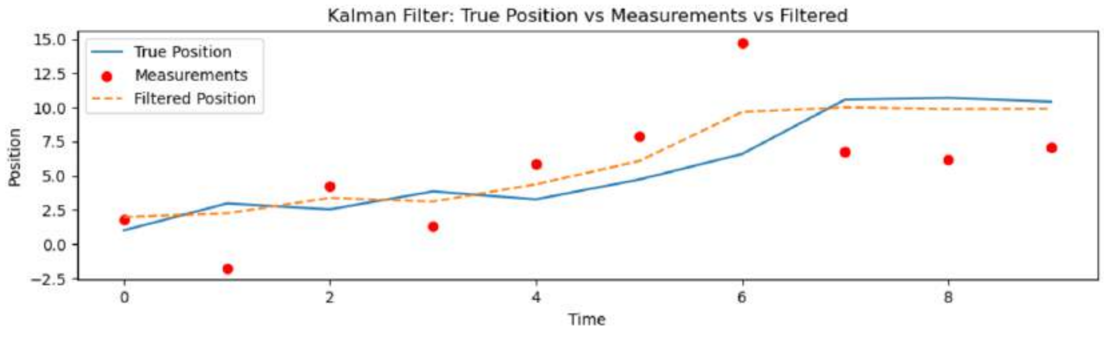

Figure 24: HIGH W