Tracking the Latitude and Longitude of Boats in a Video Feed

Blair Johnson

December 12, 2020

1 Introduction

The primary objective of this project is to create a system that meets the requirements of the US Navy AI Tracks at Sea challenge. The challenge asks competitors to create a system that can generate the longitude and latitude trajectories associated with boats in a camera feed. The competition provided 17 three-minute videos, a log of estimated frame times for each video, and a log containing timestamped GPS coordinates for the boat in the videos. Each video was taken in the same location using the same boat and camera on the same day. The location of the camera was provided, but not its orientation.

2 Gathering Background

Several key points of information were conspicuously missing from the information provided by the competition coordinators. First, the orientation of the camera used to record the videos was not provided. This would be essential information for any optical tracking system. Second, no information that would be useful for traditional CV approaches was provided. Competitors were left to design a solution without precise data on the camera's focal length or field of view. Thus, information about the camera's orientation and configuration had to be derived from the information that was provided.

2.a Orientation

In order to determine the orientation of the camera in the world frame, the area around its known position was explored using Google Earth. Prominent landmarks in the video, such as an airplane hanger on the far right hand side of the frame and a large white building in the center of the frame were identified using Google Earth, and their positions were noted. Using the coordinates of these locations, the camera's global orientation was defined as the vector projecting from the camera along the right hand side of its field of view. The camera's position and orientation were then used to define the 2D homogeneous transformation matrix[1] (1) from the world frame to the camera frame.

$$\zeta_o^c = \begin{bmatrix} \cos\theta & -\sin\theta & \vec{t}_{lon} \\ \sin\theta & \cos\theta & \vec{t}_{lat} \\ 0 & 0 & 1 \end{bmatrix}, \ \theta = \angle \vec{r}_{RHS} = -0.373448408973746\, rad \tag{1}$$

2.b Camera Parameters

With no obvious right angles or parallel lines in the frame, determining the camera's focal length would prove extremely difficult. This eliminated many traditional computer vision methods that would provide exact solutions to the world frame tracking problem from consideration. This left two main options for consideration: approximate heuristic-based solutions, and deep learning solutions. With the limited quantity and quality of data, simpler heuristic solutions were desirable. This class of solutions would create simple mappings from bounding box data to real world positions and would rely on the horizontal field of view of

[1] As the camera resolution was 720p, the system will only be able to resolve boats on a small local scale. This will allow the use of 2D Cartesian coordinates as an approximate for surface longitude and latitude.

the camera to create these mappings. Unfortunately, there were no readily identifiable landmarks on the left hand side of the videos. Instead, a pin was dropped in Google Earth at the location that most resembled the left hand side of the frame. From this vector, the approximate field of view was calculated to be 86°.

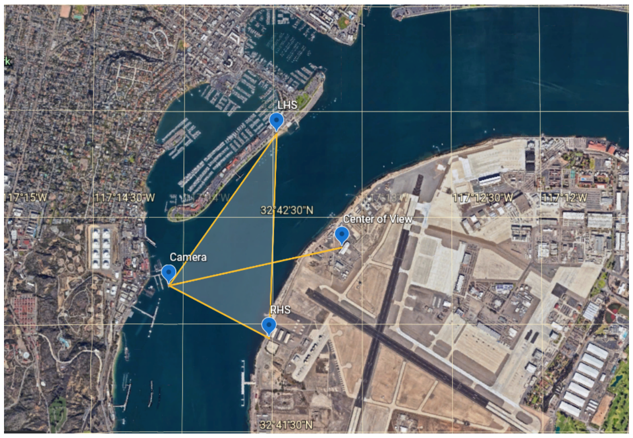

Figure 1: Google Earth project with annotations for camera, right landmark, center landmark, and estimated left hand side position.

3 Labeling and Preparing Data

The first step completed in this project was hand-labeling bounding boxes around the target vessel in each of the videos provided. MATLAB's video labeling tool was used to interpolate hand drawn bounding boxes between frames. The outputs of this hand labeling emulate the outputs of the object detection model that would be used to detect vessels within the frame. A script was written to parse the bounding box objects that MATLAB's video labeling tool outputs and save their attributes to CSV files.

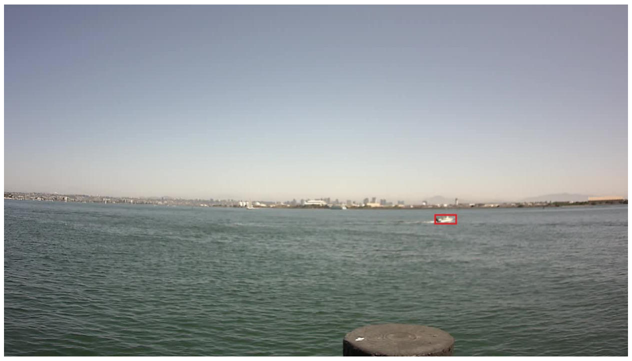

Figure 2: A bounding box rendered around the target vessel.

The next task was to combine the camera and vessel data sources into a single log of times, frames, and locations. This presented two major challenges. The first was the quality and resolution of the data. Both the GPS and frame logs contained timestamps that were only accurate to the second. The camera frame log also contained missing timestamps for the first several seconds of each video. The second challenge was the irregularity of the data. The webcam used in data collection operated with a variable frame rate making it difficult to interpolate between timestamps. The GPS on-board the vessel also collected data at very irregular intervals and at a much lower average rate than the camera. This meant that combining these two data sources would require estimation of the true timestamps to the fraction of a second, as well as correlation of the irregular GPS locations with the video frames.

4 Cleaning and Correlating Data

A script was written to ingest the GPS, camera, and bounding box label data for each video and save a pickle file containing a master record of every frame, the time it was taken, the approximate location of the vessel at that time, and on-screen location, height, and width of the vessel.

4.a Assigning Timestamps

A simple interpolation method was used to assign approximate timestamps to each frame and GPS data point. For every second within each video, an estimated frame rate was generated based on the frequency of identical timestamps. Using this approximate frame rate, repeated timestamps were evenly interpolated within the second in which they were captured. GPS coordinates captured within the same second were interpolated evenly in a similar fashion. The missing timestamps at the beginning of each video were predicted by projecting backwards from the first labeled timestamp using the approximate frame rate estimated at that location.

4.b Correlating Data

In order to identify the relationship between camera data and world frame position, each GPS position within a video run time must be matched to its associated on-screen location. In order to do this, a function compared the GPS timestamps generated within the run-time of each video with that video's frame timestamps. Each GPS coordinate was assigned to the frame that was captured closest to it in time. As frames were captured more frequently than GPS data, the majority of frames had no associated coordinates. At this time, bounding box labels were read and joined to the DataFrame containing the correlated data along the frame index.

5 Positional Regression

With screen and world positions correlated in a single dataset, it was finally possible to begin designing a system that could translate between the two.

5.a Linear Model

The most promising approach explored was the linear model. In this approach, the camera's field of view was represented in a 2D polar coordinate frame. In this frame, every GPS location was represented by an angular rotation from the right hand side of the field of view and a linear translation away from the camera. Taking advantage of two simple relationships, these angle and distance parameters could easily be estimated from a bounding box. First, there was an approximate linear relationship between the on-screen x coordinate as a proportion of the horizontal resolution, and the angle of the corresponding position. The constant of proportionality in this case is the field of view, and this relationship is represented in (2). There was also determined to be an inverse relationship between the on-screen height of the boat and its distance from the camera, represented in (3).

$$\angle \vec{r_b^c} \propto \frac{1 - x_b}{1280} \tag{2}$$

$$\left|\vec{r_b^c}\right| \propto \frac{1}{\mathrm{h_b}} \tag{3}$$

In order to find these proportionality constants, the correlated dataset was filtered to remove frames which did not have a corresponding GPS position, and then linear searches were used to solve the minimization problem in (4).

$$\underset{f \in \mathbb{R}, \alpha \in \mathbb{R}}{\text{minimize}} \quad \sum_{n=0}^{N} \left\| r_{\vec{b_n^o}} - \zeta_o^{cT} r_{\vec{b_n^c}} \right\|^2 \tag{4a}$$

$$\text{subject to} \quad 0 \leq f \leq \frac{\pi}{2}, \tag{4b}$$

$$0.00001 \leq \alpha \leq 0.00005 \tag{4c}$$

Since the boat's distance from the camera and angle from the camera's orientation were independent and could be derived from the boat and camera locations, (4) was broken into two separate minimization problems. The constrained domain of each parameter was searched to arrive at the optimal values of $f = 1.2008$, $\alpha = 0.000021$ which minimized the error in predicting angle and distance[2]. Examining the resulting predictions in Figures 3 and 4, the angle estimator appears to be highly effective, whereas the distance estimator exhibits a "jitter" which hurts its performance as an estimator.

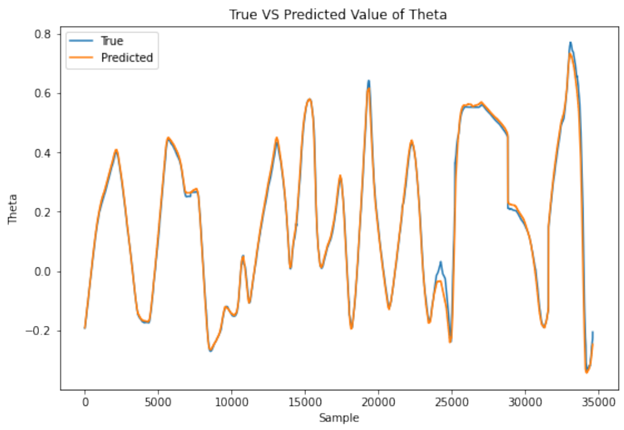

Figure 3: True vs estimated theta values.

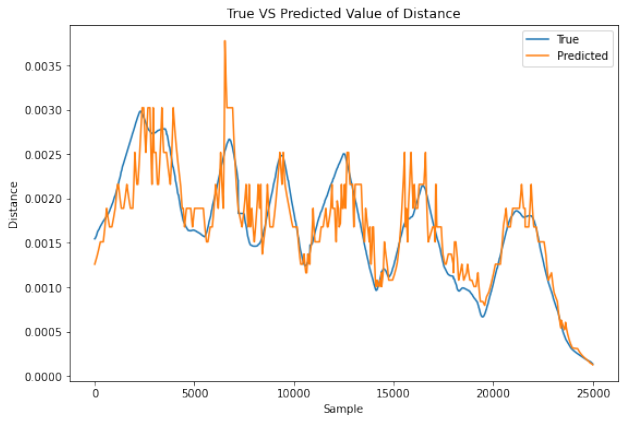

Figure 4: True vs estimated distance values.

Several different methods of filtering will be evaluated as potential solutions for the error in the distance

[2]Towards the end of the dataset, the boat traveled beyond the resolution limitations of the camera. These samples were excluded from the cost function evaluation as they provided no usable boat height data.

estimator. The final[3] equation relating bounding box data to world frame position is (5).

$$\vec{r_b^o} = \zeta_o^{cT} \left[\frac{\alpha}{h_b} \cos\left(\frac{f(1-(x_b+0.5w_b))}{1280} \right), \frac{\alpha}{h_b} \sin\left(\frac{f(1-(x_b+0.5w_b))}{1280} \right), 1 \right]^T \tag{5}$$

5.b DNN Regression

Another method for positional regression that was briefly explored was a deep learning based approach. A small deep neural network was constructed with six layers of 16 nodes, followed by an output layer of two nodes. Linear activation functions were used throughout the network. On-screen x and y positions, as well as relative longitude and latitude positions were normalized, and a 20% validation set was withheld. The model was trained without dropout for 50 epochs on the remaining data with a MSE loss and Adam optimizer. Comparing the trajectories generated by the model, and the true validation trajectories, two main observations were made. First, the model seemed to do a good job of capturing the proper shape of the boat trajectory. Second, the model failed to project its predicted longitude and latitude coordinates into the proper scale. Results were several orders of magnitude larger than the desired values, and required manual re-scaling to be comparable. This method showed some promise, but no more investigation was done as the linear method proved far more reliable.

6 Filtering Regression Results

Although the linear method of positional projection worked well, there was still significant error present in the distance estimation results. Several filtering methods were explored to combat this error.

6.a Moving Average Filtering

The first method that was used to improve results was the moving average filter. A linear search was performed over the range of possible window lengths and found 367 to be the value that minimized the squared error function.

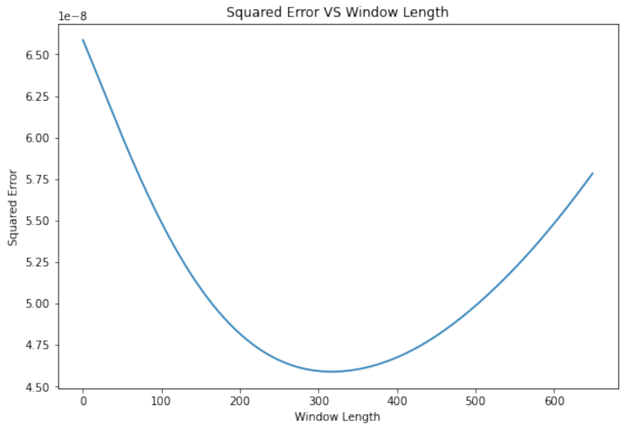

Figure 5: The characteristic convex loss curve of the squared L2 norm.

This method showed clear improvements in the quality of distance prediction. Figure 6 shows the improved distance estimation results.

[3] Note that x_b is the x coordinate of the left hand side of the bounding box, and a $0.5w_b$ (width of bounding box) term has been added to obtain the x coordinate of the center of the bounding box. h_b is the height of the bounding box.

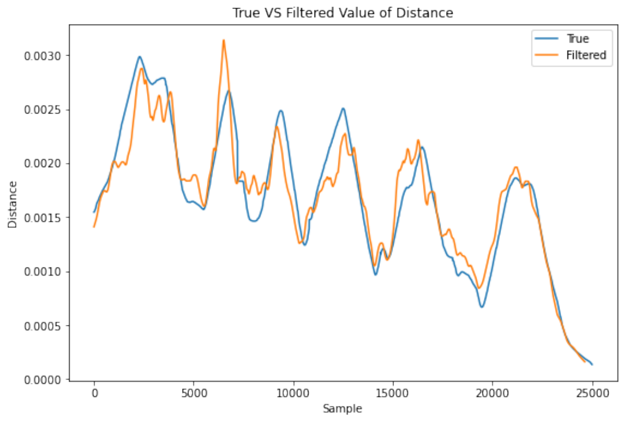

Figure 6: Results of linear distance estimation after moving average filtering with window length 367.

6.b FFT Filtering

An additional method that was explored to improve distance estimation was the use of a lowpass filter via the Fast Fourier Transform. Another linear search was performed to determine the number of terms of the DFT to truncate before reconstruction in order to minimize reconstruction error. It was found that reconstruction from the first 50 terms of the DFT was optimal, but in Figure 7 it is evident that this results in a poorer approximation of the true signal than the moving average method produced.

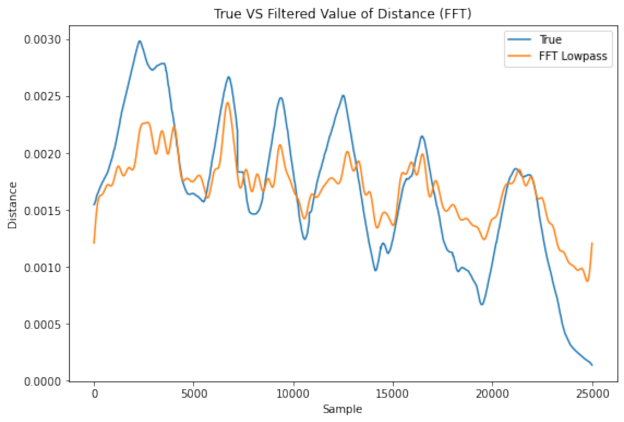

Figure 7: Result of linear distance estimation reconstructed from the first 50 terms of its DFT.

6.c Kalman Filtering

The final method that was explored involved operating on the predicted trajectories produced by the linear model. The Kalman filter seeks to model the approximate system dynamics of the hidden state vector and uses this a priori knowledge to improve a streaming estimate as it receives new observations. The full scope of the Kalman filter's capabilities was not explored, but early work produced a configuration that showed promise in improving the quality of path predictions. Algorithm 1 describes the third order Kalman filter algorithm used. The chosen dynamics model assumes constant acceleration over short timescales, and it clips acceleration values that surpass the highest values found in the training dataset. Initial values for the P_k, the process covariance matrix, were chosen based on reasonable error tolerances in the linear position estimate. These values were estimated in meters, and as such, this implementation of the Kalman filter requires that latitude and longitude values be converted into circumferential meters before ingestion. This conversion will vary with latitude, but it is approximately affine when far away from the poles.

Algorithm 1: The Kalman Filter

```
initialize:
$\hat{x}_0 = [y_{00},\ y_{01},\ 1,\ 1,\ 0.1,\ 0.1]$
$H = I_{6x6}$
$\delta = [21,\ 21,\ 5.2,\ 4.7,\ 1.2,\ 1.2]$
$P_0 = \delta^T \delta$
$R = 1.1 * P$
$\alpha_{max} = 15$
$F_0 = \begin{bmatrix} 1 & 0 & dt_0 & 0 & \frac{1}{2}dt_0^2 & 0 \\ 0 & 1 & 0 & dt_0 & 0 & \frac{1}{2}dt_0^2 \\ 0 & 0 & 1 & 0 & dt_0 & 0 \\ 0 & 0 & 0 & 1 & 0 & dt_0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$
$Q = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \end{bmatrix}$
for $k$ = 1,2...,  $dt_k$ do
    $\hat{x}_{k|k-1} = F_k \hat{x}_{k-1}$
    $P_k = F_k P_{k-1} F_k^T$
    $K = \frac{P_k H^T}{H P_k H^T + R}$
    $P_k = (I - KH) P_k$
    if $y_k$ then
        $\hat{x}_k = \hat{x}_{k|k-1} + K(y_k - H\hat{x}_{k|k-1})$
    end
    if $\alpha_{max} \leq |\hat{x}_{k4:}|$ then
        clip: $-\alpha_{max} \leq \hat{x}_{k4:} \leq \alpha_{max}$
    end
end
```

Another important note about this implementation of the Kalman filter is that it requires observation of the entire state vector. This means that each y_k must contain data about position, velocity, and acceleration in two dimensions. In order to generate these values, both the first and second order discrete derivatives were taken from the observed position vector produced by the linear model. This shouldn't be necessary, but quality results were difficult to obtain without this added information. In theory, the Kalman filter should also be able to generate reasonable position estimates even if no measurements are fed into it. This is important, because the object detection algorithm generating bounding box data from a video cannot be relied upon to identify the boat in every frame. This is especially an issue given the low resolution of the camera in use. If the boat travels too far away from the camera, the algorithm will be unable to resolve its position. In practice, experiments with using the Kalman filter to predict the boat's position in missing frames have been largely unsuccessful. The filter has a tendency to make extremely aggressive predictions about the boat's behavior, launching it off in random directions and producing a very "jagged" trajectory. These issues suggest that there are still problems with the current implementation of the Kalman filter, and further work could improve these results.

7 Analysis and Conclusion

The most successful of the approaches examined for generating world-frame positions from on-screen locations was the linear model. While experimentation showed that the DNN was capable of projecting screen-space coordinates to world-frame coordinates, the results were too inexact to easily work with. In addition to being extremely fast and simple to implement, the linear model showed that it could consistently reproduce approximate trajectories from video data. The linear model also had the added benefit of interpretability, and derived constants could be verified with real-world measurements. While the linear model proved quite effective as an initial step, the high volatility of its distance estimates was a major contributor to the overall system error. Figure 8 depicts an example trajectory, and the raw estimated path produced by the linear model.

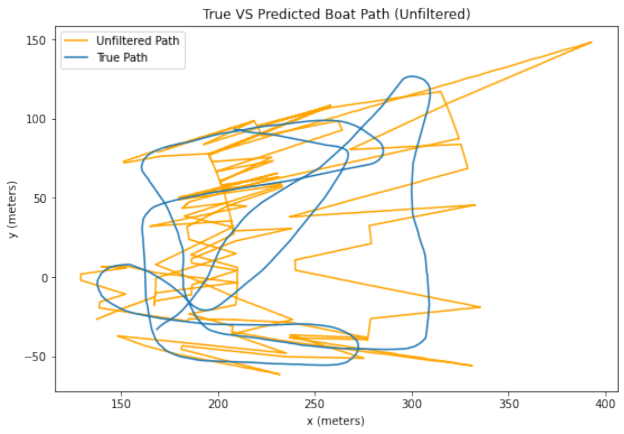

Figure 8: True boat trajectory and the raw trajectory produced by the linear model.

Since the model predicts distance from the height of the bounding box, the resolution in distance is far lower than the resolution in angular position which is produced from horizontal location within the image. This is evident when examining Figure 8. The predicted trajectory is dominated by a handful of discrete distances. This is most noticeable as the boat gets farther away from the camera. As this happens, the height decrease becomes a greater proportion of the overall height of the bounding box, resulting in increasingly spaced out discrete distances. Filtering promises to smooth out these distance predictions and provide a more accurate estimate of the boat's path. The two most effective filtering methods examined were the moving average filter and the Kalman filter[4]. Each of these two filtering techniques exhibits unique characteristics that make them ideal for different use-cases.

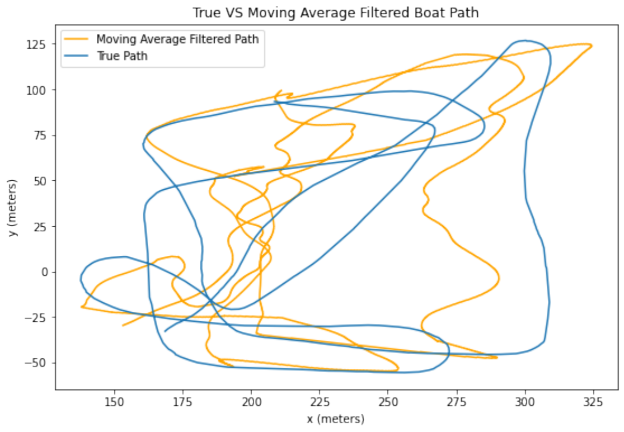

Figure 9: True boat trajectory and the boat trajectory after moving average filtering with a 367 element window.

The moving average filter did a remarkably good job of producing reasonable vessel trajectories from the

[4]The FFT lowpass filter removed some of the jitter from the distance estimates, but strayed too far from the true values to be as useful as the other two filters examined.

linear model output. This filtering technique was the most effective technique when doing an error based assessment. The locations produced by the moving average filter were consistently closer to the true locations of the vessel than any of the other filtering techniques examined. That being said, the moving average filter did a poor job of smoothing out the high frequency ”squiggles” that are characteristic of its outputs. Despite being accurate, these paths are unrealistic for an actual boat to follow and would require many tight turns that the actual vessel does not make. This is where the Kalman filter excels.

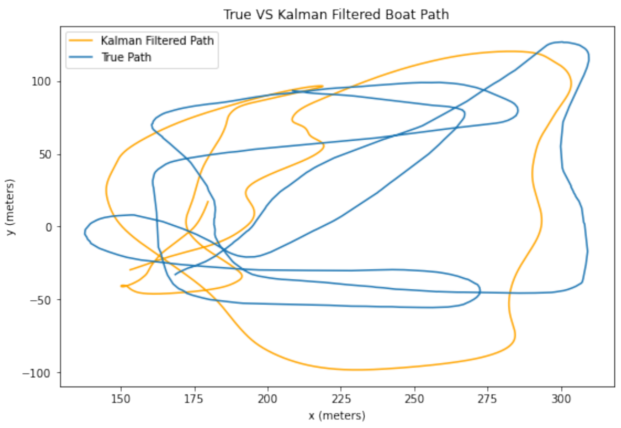

Figure 10: True boat trajectory and the boat trajectory after the Kalman filter was applied.

While the Kalman filter produced vessel trajectories that were consistently farther from the true trajectories than the moving average filter, it did a very good job of producing realistic trajectories from the volatile inputs that it received from the linear model. The trajectories that the Kalman filter produced rarely contained maneuvers that would be difficult or impossible for a boat to take and overall they represented a qualitatively better estimate of the vessel’s path. Additionally, early experiments suggest that Kalman filtering of the moving average filter outputs can further improve the visual consistency of the results. These results suggest that the system could be tuned to the desired use-case of a user. Should an application value minimization of tracking error, then just a moving average filter is necessary. If an application is user-facing and values realistic and ”pretty” predictions, then the Kalman filter can make tracking more presentable and realistic.