TEXT and rectangles in blue will NOT show on printed copy

Type final title of thesis or dissertation (M.S. and Ph.D.) below . If your title has changed since your submitted an Application for Graduate Degree, notify Graduate Office.

Comparative Analysis of Tracking Algorithms through Video

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment

of the requirements for the degree

Master of Science

by

Name

Michael Lukas

Month degree is awarded May

Year 2015

APPROVAL SHEET

The thesis

is submitted in partial fulfillment of the requirements

for the degree of

Master of Science

AUTHOR *signature*

The thesis has been read and approved by the examining committee:

Please insert committee member names below:

Dr. Donald Brown

Advisor

Dr. Peter Beling

Dr. Gerard Learmonth

Accepted for the School of Engineering and Applied Science:

Dean, School of Engineering and Applied Science

Month degree is awarded May

Year 2015

Print Form

Comparative Analysis of Tracking Algorithms through Video

Presented to

the Faculty of the School of Engineering and Applied Science

University of Virginia

In Partial Fulfillment of the requirements for the Degree Master of Science

(Systems and Information Engineering)

by

Michael Lukas

April 2015

Abstract. Kalman filtering was introduced by R.E. Kalman in 1960 as a way to predict the state of system that was subject to noisy measurements. The measurements were assumed to have Gaussian noise which make the actual state somewhere in the middle but unknown. Since the first implementation the Kalman filter has been used extensively in signal processing and later introduced into object tracking. Using state equations from physics on a moving object a newly predicted state can be estimated based on time between observations the only difference between signal processing and object tracking is working in an additional dimension.

The Particle filter (sequential monte carlo (SMC)) was introduced in 1993 by Gordon et al. in the paper 'Novel approach to nonlinear/non-Gaussian /Bayesian state estimation' which discussed a new way for state space estimation by continuously resampling an estimated distribution and reducing the error based on actual observations. The inherent advantage to this method is that it requires no knowledge of how the object motion needs to be modeled.

In this paper I intend to do a comparative analysis of algorithms on video data and how well they are able to track an object in motion as a saved video or if given the video as though it were sequenced in real-time. Another aspect of exploration is the determination of which algorithm performs better under variable frame rates or how slow can each frame rate be before there is a severe lack of data to track objects in a scene accurately.

Contents

1. Problem Statement

Computational performance for many complex algorithms has been significantly reduced and even some of the most complex algorithms can be iterated through in a fraction of a second. Although technology has allowed for complex algorithms to be instantiated throughout certain fields of study, there has yet to be a performance estimate of computation complexity, real-time processing, or accuracy comparisons between certain video processing algorithms. This thesis work would be a comparative analysis of competing algorithms used for object tracking in a video scene. The video will contain varying degrees of difficulty and a comparison will be made as to how each could possibly be better than the other and under what conditions it applies.

2. Literature Review

2.1. Particle Filter. Gordon, Salmon, and Smith prove in their paper that a bootstrapping filter, now commonly referred to as the Particle filter or sequential Monte Carlo method is superior to the extended Kalman filter due to the fact it does not rely on a model of behavior and its innate ability to predict movement of non-linear functions. The filter works by making initial guesses as to where the point or where the signal is. These guesses are referred to as particles, the particles are worked on by some updating function that predicts where they will be at the current time step. Once the observation of where the point is the particles are all updated according to a weighting scheme that weights points closer to the observed point higher and distances further from the observed point lower. Then the particles are redrawn from this newly weighted distribution and used for the next time step which follows the previous steps to update the point, weight them according to the observed point, then redraw them from the new distribution.

The portion that is missing is that there is lacking knowledge of how the computational load increases as the number of objects increases. This is caused by the continuous sampling and resampling of particles. Each iteration requires the updating of each particle through time then calculating their importance on what was actually observed then resampled to be

used in the next iteration. When passing this through thousands of particles and numerous states the computational effects have yet to be studied. Their examples only follow a single object for their tracking example and only one signal in their one dimensional case. By increasing to multiple objects the calculations increase exponentially and there has been no comparison to see if the algorithm can support multiple objects in a near real-time environment or what happens when only given a sparse dataset, which could occur in a real world application.

The ending of their paper is a call for more research in regards to a "quantitative assessment of filter performance for important nonlinear problems." [5]. Gordon's work relies on one other aspect of their tracking behavior is that they are only looking at a single object as well as not looking at computational complexity and time. By looking at other aspects yet to be explored a new analysis of the filter's performance can be addressed.

2.2. Kalman Filter. In his paper 'A New Approach to Linear Filtering and Prediction Problems'[11], Kalman presents his research for finding an optimal filter that is much more simplified than the Wiener filter as the Wiener filter had too many limitations that did not promote practical implementations. Kalman introduced this concept in order to predict random signals based on noisy measurements and allow the prediction of known signals in the presence of random noise. Although largely Kalman assumes the random noise to be Gaussian in nature. This paper also requires the signal or track to be largely linear in nature, statistical outliers result in a negative Kalman gain causing the covariance and the model to have negative state predictions.

The Kalman filter works in a few steps in a feedback loop. The initial requirement is that the system be modeled by a state transition equation. If the model is too far off the resulting predictions will be far off unless it can be fixed in the covariance calculation update. For a tracking problem as the one to be researched a displacement model would accurately described the model in which the object will be tracked. With an adequate system model we then predict the next state based on the initialization state. This state then has a calculation of Kalman gain this calculation is intended to determine how inaccurate the

state prediction was to the actual observed value. Kalman gain is intended to reduce error in the calculation of the covariance matrix. The covariance matrix is the posteriori error from the state prediction to the observed value. The updated covariance matrix will be used in the next state prediction and should have less error due to the update from the Kalman gain calculation. The limitations of this approach have been noted and other variations of Kalman have been created to circumvent some of the assumptions that have to be made in order to accurately use this method; linearity, Gaussian noise. Although these limitations exist the low amount of computational cost could prove it to be a satisfactory method for tracking objects.

2.3. Real Time Object Tracking. 'Real Time Object Detection and Tracking: Histogram Matching and Kalman Filter Approach' [15] is a paper from 2010 that assesses the feasibility of using histogram segmentation of static video and used in combination with Kalman filtering to track an object in real time. The authors of this paper are using a histogram segmentation algorithm in combination with absolute frame differencing. How this works is by finding an object that satisfies a certain histogram distribution. The distribution is binned in a certain way in that the object is unique to the background or environment. The problem with this technique is the spectra in which they work. Since red, green, blue (RGB) values are not linear there is certain conditions where this technique will assume the object is new based on varying degrees of light intensity. To better compensate the authors should have worked either in hue, saturation, value (HSV) coordinates which are cylindrical or luminosity, a, and b (LAB) values which is an international standard and is a cubic structure thus works better in over saturated environments. The other aspect is that the distribution of colors can also change as the object changes position in reference to the camera. There could be shifts of the histogram and could be seen as a new object if say the object goes from a head on view to a profile view from the camera. Based on the set-up of there algorithm it could cause a re-initialization and loss of previously accumulated track data. The only practical purpose for this would be to process many different videos and trying to follow a specific object through a scene in which the color histogram was previously known.

2.4. Video Based Object Tracking. "Video Based Moving Object Tracking by Particle Filter"[8] introduces two differing methods of identifying an object but rely on Particle filter as the algorithm of choice to predict movement from frame to frame. The first segmentation algorithm the authors introduce is another histogram segmentation algorithm that requires a pre-set histogram or human interaction to determine which object is to be followed through the sequence of video. The authors work in HSV color space which is robust when light intensities change. The other method they are introducing is template matching. This method introduces algorithm complexity as it is susceptible to changes in shape, rotations, camera angle, distance from the camera. The way to address these issues is to have a large repository that these different templates reside and are accessed whilst processing the video. The authors use a correlation factor and probability of the state transition to determine if they are still following the same object. This method is inherently more robust than relying on histogram matching as explained earlier about position to camera, etc. The only requirement is that the amount of templates to match could lead to large computation needs and a repository for how the template will change in time. The authors also use a clean video that has no additive noise. Their template extraction begins with a canny edge detector. The canny edge detector works by looking at differences in intensity gradients[2]. A large gradient increase between two pixels informs the detector that there is likely an edge present. Since the videos being processed in the authors videos have little to no noise the canny detector works as expected. The other problem with canny is if the background and object to be tracked are close in color causing no large gradient change from pixel to pixel. The authors allude to this type of failure as they track the object and have only satisfactory results. Although based on their claims of robustness I feel as though they felt short on tests that run the gamut of robustness, i.e. tracking multiple objects, introducing noise to the video signal, adjusting frame rates, and working in an environment where object and background are similar in color.

2.5. Comparing a Kalman Filter and a Particle Filter in a Multiple Objects Tracking Application[14]**.** The authors of this paper do a comparative analysis

of a Kalman filter against a Particle filter in a real-time application based on video. The basis for this analysis exists to look at single and multi-target applications through video sequences and determining which is the better of the two algorithms. Based on their assessment the particle filter wins out due to how robust it is since it creates other exploitable features for data association. The Kalman filter application uses a probabilistic approach to data association that requires different feature extraction such as Euclidian distance from predicted point to next observed value. This is time consuming an computationally intensive. The particle filter uses multi-modal application to simultaneously use one set of particles to create probabilistic associations to data. Due to this robustness the processing time when objects was four or greater resulted in a computation time that was nearly half that of the Kalman filter application. Based on their findings the application of the Kalman filter would have been best suited for applications only involving a single target while the particle filter was best overall. Although the focus of this paper largely focuses on the front-end data association portions it delves into the the robustness characteristic of the particle filter for real-time processing of multiple objects.

2.6. Performance Comparison of Gaussian-Based Filters Using Information Measures. Tracking objects through non-linear trajectories causes many problems with state estimation as with most algorithms a modeling behavior equation needs to be expected or known. In this paper Vemula [18] uses two different metrics in order to optimize error based on algorithm. The first is the Kullback-Liebler $\mathcal{KL}$ divergence along with root mean squared error. The $\mathcal{KL}$ divergence is an information metric and is described in equation (13) . The value of the $\mathcal{KL}$ divergence is always positive and represents how much a sampled distribution, q, diverges from a reference distribution, p. The paper shows how there is a correlation between lowering the $\mathcal{KL}$ and the smaller the RMSE becomes. They do this dynamically at the expense of computation time to determine the best choice of optimizing parameters for the unscented Kalman, extended Kalman, and particle filters. In order to make a representative distribution of the posterior the authors create this by using a large scale particle filter (10000 particles). This is justified by the general acceptance that

particle filters overall have the lowest RMSE which means they best estimate the posterior distribution. This paper was used a reference for output values as they were also looking at non-linear systems in a continuous context. Based on results shown herein and the values that the authors stated I was able to conclude that my versions were correctly instantiated and were reproducible.

2.7. Particle Filters for Positioning, Navigation and Tracking. Different applications of Kalman and particle filtering are analyzed in this paper. The authors make an association in order to reduce the jittering effect that is sometimes needed in the particle filter. The jittering effect is used when users want to minimize the number of particles used in order to speed up prediction times. Sometimes during this minimization effort the number of particles used will not create enough particles to represent the actual sample density or distribution i.e. the weighting of the particles has many zeros. The jittering effect creates an intermediate step where the particles are resampled with added noise in order to create enough particles that have weights greater than zero and have a sufficient number of these points. Although the authors only utilize the normal Kalman and particle filter they accurately describe the computational complexity which is for the Kalman is on the order of $\mathcal{O}(2n_x^3)$ and the particle filter on the order of $\mathcal{O}(Nn_x^2)$ where in these equations n_x represents the amount of dimensions in the state space estimate. Based on the notion that $n_x = 4$ and $N = 500$ we should see that the Kalman filter is an estimated 62 times faster than the particle filter.

3. Methods

3.1. Data Source. Availability of data was a difficult problem as really the only truly non-linear source would be from simulation of data but would result in the algorithm containing the function with which the data was created from. Another source that would have had real world applications would be the tracking of ships through time. This did not fulfill the non-linear portion and would ultimately not be an adequate representation of a comparison since the normal Kalman filter would provide the best prediction at the lowest

computational cost. The search lead me to something that I had seen everyday during the summer months. That source was how my dogs behaved while at the dog park while retrieving balls. Their pattern was non-linear, their velocity and acceleration were non-constant as well as if a camera were to capture the movement there would be relativistic effects from the fixed position.

3.2. Image Processing. In order to extract a usable dataset the method of processing the images becomes important. Dealing with a single object through a video sequence requires no data association methods and can easily be done. The increasing complexity of multiple objects tracking and prediction creates an entirely different range of problems. How do you associate a previous data point to a prediction and the next data point? Each of these scenarios will be discussed in the following sections.

3.2.1. *Single Object Tracking.* This portion of data extraction was simple since there was no need to associate previous data with current data so the only aspect discussed in this section will account for the method in which the image was processed. The following multiple object tracking section will discuss how data was associated with previous data (i.e. covariance matrices). A simple yet powerful way to distinguish movement in a video sequence is to take the absolute difference between two sequential frames. This works well with registered (still) video. Unregistered video has an added step of feature extraction and will not be discussed in this document. Taking the absolute difference of frames creates a ghost image of what has moved between two frames since there will only be a minutia of movement or difference of all objects.

The difference image as seen above in figure 1 is then further processed in order to elicit the needed information. This can be done in a myriad of ways so I was interested in a computationally inexpensive method that found available data. The approach that was taken was to first threshold the difference image in order to suppress minute changes in intensities or returns. These differences could be something small such as wind causing the grass to move and reflect more or less light. Once the thresholded image is created and

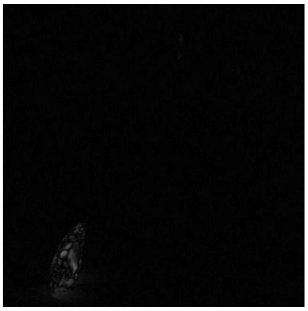
Frame 1	Frame 2	Difference Image
		

FIGURE 1. Example of Frame Differencing

the remainder of the image only contains the larger objects that have changed I applied a black-white conversion of gray values that turned anything gray into white pixels while the background or non-changing pixels were set to remain black. The remaining black and white image is then processed to create a blob shape. The processing is required since some changes from the frame differencing and thresholding could create multiple separate blobs which would later result in multiple objects being tracked that were actually the same object. This is done using a varied combination of built in MATLAB functions such as *imdilate*, *imfill*, and *bwareaopen*. The result of the processing is shown in figure 2 which illustrates the differenced image with the result having two objects in the frame moving since the previous frame.

FIGURE 2. Processed Black and White Image

As can be shown in 2 we can see two distinct objects within the frame. Based solely on the differenced image the ball being thrown was not entirely noticeable. The way to distinguish the objects will be discussed in the next section about multi-object tracking. Since we know that we only want to track the largest moving object in the frame we use the centroid function built in from the *bwlabel* command in MATLAB. This function labels all centroid locations of each unique blob. Since for the single object case we are only interested in the dog movement as such we identify the largest blob and disregard everything else.

3.2.2. *Multi-Object Tracking.* Multi-object tracking required another level of data association. This was achieved by first determining if the centroid was in a previous frame. To do this all previous tracks were searched with then they were compared against the predicted location for the next frame. If the centroid was located within two standard deviations of the predicted value then the centroid was associated with the previous covariance matrix and allowed to continue. If the track was not associated and deemed a new track and the database of tracks would be updated to reflect a new track. The database in this case refers to a structure element that contains the new measured value, the prediction from the previous time step, and also the associated covariance matrix.

3.3. Algorithm Implementation. In the literature review section I looked at work from various authors on prediction theorems such as the Kalman filter and particle filter. The mathematics involved is discussed in their respective papers on how they work along with their proofs. This section will discuss the crux of the formulae along with how code was created in order to discuss specific cases that this paper intends to expound upon.

3.3.1. *Kalman Filter.* The Kalman filter is a type of discrete state space model. This follows that the previous states prior to the current state and the next predicted state do not directly influence the predicted state. The standard formulation for this is

$$x_t = g(x_{t-1}) + u_t \tag{1}$$

$$y_t = h(x_t) + v_t \tag{2}$$

x_t is the unknown state, $g(x_{t-1})$ is the state transition function, y_t are the measurements, $h(x_t)$ is the measurement function, and v_t and u_t are the expected noise. Since I was working in a two dimensional space the state transition function took the form as follows.

$$g(x_{t-1}) = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3}$$

This linear set of equations relates to the change of position formula represented by

$$\text{Change of Pos.} = v * \Delta t + \frac{1}{2} * a * \Delta t^2 \tag{4}$$

$g(x_{t-1})$ only takes into account the velocity vector. To help estimate acceleration we fold this into the u_t portion of the equation and is represented below by the matrix of the form

$$u_t = \begin{bmatrix} \frac{1}{2}\Delta t^2 \\ \frac{1}{2}\Delta t^2 \\ \Delta t \\ \Delta t \end{bmatrix} * Noise \tag{5}$$

$h(x_t)$ is just a position update this matrix helps to calculate the Kalman gain which is how incorrect the state update is from the actual measured value. This value helps to try and reduce the error in the next update through the use of the covariance matrix. The derivations are left in the paper proposed by Kalman [11]. The covariance matrix is calculated as follows

$$P = g(x_{t-1}) * P * g(x_{t-1})^T \tag{6}$$

This is the estimated covariance for the next update step and used to estimate the following location at time t. To estimate how correct the estimate is the Kalman gain is calculated which is something like an averaging factor that weights the estimate with a vague

knowledge of what has happened previously from the covariance matrix. This is calculated as follows

$$K = P * h(x_t)' * inv(h(x_t) * P * h(x_t)' + noise) \tag{7}$$

Now we just combine all the equations together and estimate the new point by the following equation

$$x_t = (g(x_{t-1}) + u_t) + K * (PreviousMeasuredValue - h(x_t) * g(x_{t-1}) + u_t) \tag{8}$$

That refers to the code required to make a prediction of the next time step. The limitations of this is that the model assumes linearity and begins to accumulate errors once the measured values become non-linear. Non-linearity refers to values that cannot be expressed in a linear form from one time step to the next. These are things like polynomial equations that model the behavior. The way to better estimate non-linear models is with the extended Kalman filter.

3.3.2. *Extended Kalman Filter.* The way in which the extended Kalman filter differs from the Kalman filter is the state update model that is used as the predictor for the next state. For the Kalman this model is a linear model which in my case was the change of position equation from basic physics. The model in the extended Kalman does not need to be linear but it does require it to be differentiable. This is because the extended Kalman includes a step that creates another matrix of partial derivatives since the covariance and means of the update model cannot be applied directly. By using the partial derivatives the calculations essentially linearize the current estimate to be nearer the mean and covariance.

The implementation for this was bearings only tracking. This uses trigonometric equations to estimate the position based on some relative point. The Jacobian transition matrix

is listed below.

$$H = \begin{bmatrix} cos(\hat{\theta}) & 0 & sin(\hat{\theta}) & 0 \\ -sin(\hat{\theta})/\hat{r} & 0 & cos(\hat{\theta})/\hat{r} & 0 \end{bmatrix} \tag{9}$$

$$\hat{r} = \sqrt{\hat{x}^2 + \hat{y}^2} \tag{10}$$

$$\hat{\theta} = \arctan(\hat{y}/\hat{x}) \tag{11}$$

3.3.3. *Particle Filter.* Particle filtering is a multi-step process that is rather computationally intensive and can greatly be influenced by how many particles are used. This filtering technique is a sample based variant of Bayes filters which is the over arching type of filtering that contains Kalman filtering. The sampling that is used with particle filtering is to create enough particles that allow samples to resemble the actual probability density function (pdf) distribution as the number of samples increases. The way in which the algorithm works is by creating a random state estimate for the first iteration. This is due to fact that there is no previous estimations on where the object could be located so it makes a random guess over the entire frame. Once the first observation is seen by the algorithm there is an importance weighting performed on the samples. The importance follows that the weights sum to one and are related to how correctly the new state was guessed. These importance weights are then passed into the next iteration where they are used to re-draw a new random set of samples. As the number of iterations increases the importance factors are normalized and tends to represent the actual posterior.

Since we assume the posterior is Gaussian we can update the next iteration by weighting the probabilities this is done for each variable and particle. The update step is

$$P_x = \frac{e^{(\frac{-(\hat{x}-x_N)^2}{(4)}}}{\sqrt{4\pi}} \tag{12}$$

We perform this step for each particle generated and then normalize the probabilities such that they sum to one. Once we have normalized the probabilities the next step is to redraw

particles based on this new distribution. These are the particles that will be used in the next iteration and state update.

3.4. Varied Time. Based on research done with these algorithms there has been little study into where they begin to breakdown based on limited information. For this portion I instituted two different methods for processing. The first was a constant variation in time which would be similar to a camera or video that streams or captures only a certain frame rates. Having a steady consistent track is great but does not always represent what data is actually available. The second variation is when we have inconsistent data. While controlling the seed numbers for the random number generator and creating a list of frames to skip we are able to see performance of each algorithm when not all data is continuous or consistent.

3.5. Metrics. Two metrics were chosen on the overall performance of each algorithm. These metrics are root mean squared error (RMSE) and Kullback-Liebler ($\mathcal{KL}$) divergence. The $\mathcal{KL}$ divergence equation is given below in the multivariate case. The original derivation and proof for the univariate case is given in their paper. The starting univariate equation is given by

$$\begin{aligned} \mathcal{KL}(p||q) &= \int p(x) log \frac{p(x)}{q(x)} dx \\ &= \log \frac{\sigma_2}{\sigma_1} + \frac{\sigma_1 + (\mu_1 - \mu_2)^2}{2\sigma_2^2} - \frac{1}{2} \end{aligned} \tag{13}$$

The multivariate extension takes into account that Σ represents the covariance matrix as well as each μ represents the mean of each given distribution.

$$\begin{aligned} \mathcal{KL} &= \int \left[\frac{1}{2} \log \frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2}(x-\mu_1)^T \Sigma_1^{-1}(x-\mu_1) + \frac{1}{2}(x-\mu_2)^T \Sigma_2^{-1}(x-\mu_2) \right] x p(x) dx \\ &= \frac{1}{2} \log \frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2} tr \left\{ E\left[(x-\mu_1)(x-\mu_1)^T \right] \Sigma_1^{-1} \right\} + \frac{1}{2} E\left[(x-\mu_2)^T \Sigma_2^{-1}(x-\mu_2) \right] \\ &= \frac{1}{2} \log \frac{|\Sigma_2|}{|\Sigma_1|} - \frac{1}{2} tr \left\{ I_d \right\} + \frac{1}{2}(\mu_1 - \mu_2)' \Sigma_2^{-1}(\mu_1 - \mu_2) + \frac{1}{2} tr \left\{ \Sigma_2^{-1} \Sigma_1 \right\} \\ &= \frac{1}{2} \left[\log \frac{|\Sigma_2|}{|\Sigma_1|} - d + tr\left(\Sigma_2^{-1} \Sigma_1 \right) + (\mu_2 - \mu_1)^T \Sigma_2^{-1}(\mu_2 - \mu_1) \right] \end{aligned} \tag{14}$$

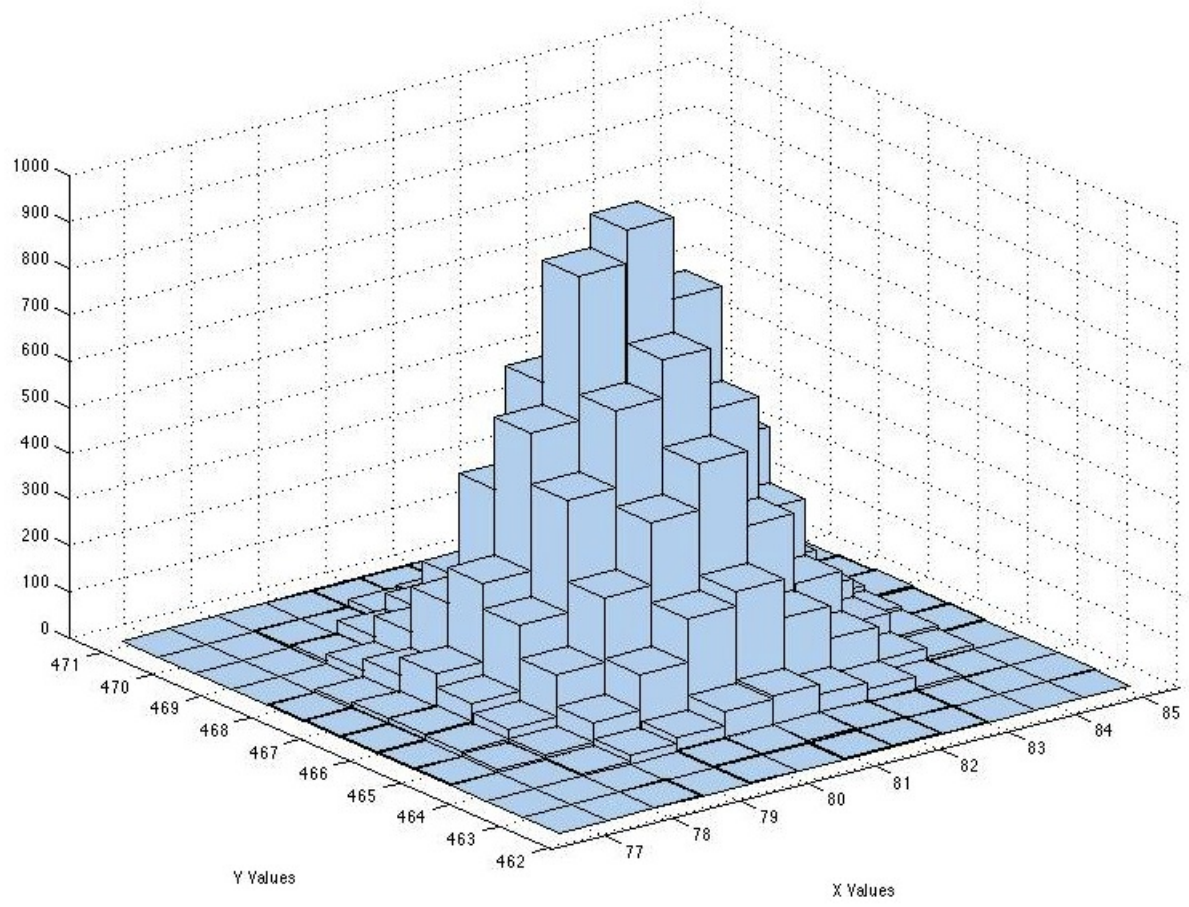


FIGURE 3. Representative Posterior Estimate with 15000 Particles

The theory behind this is that given two distributions p and q that are conditional on the same event space that you can determine how far apart q is from p and give a relative assessment of how much they differ. If $q = p$ then ultimately their divergence will be zero. In order to obtain the reference distribution I used the theory behind "Performance Comparison of Gaussian-Based Filters"[18]. In this paper they make the conjecture that to best estimate the unknown posterior distribution a Monte Carlo simulation is run with many data points in order to create a representative posterior. This is run with a particle filter that uses many particles. Due to the theory with which the particle filter operates as the number of particles approaches ∞ the true posterior distribution is achieved. In order to represent the posterior Vemula[18] used 10000 particles. For our purposes I ended up using 15000. This allowed for a more defined multivariate Gaussian distribution that is shown below in figure 3.

To ensure algorithms were written correctly a comparison between posted results from Vemula [18] and the outputs of my results should align with some differences but have similar patterns and values. Based on $\mathcal{KL}$ and RMSE values in comparison to those posted in Vemula [18] the values were deemed acceptable thus ensuring that the different algorithms were properly adjusted for different noise and tuning variables.

4. Results

Based on affirmation that the algorithms were correctly tuned from a cursory comparison of published results the next section will display the results for the different variations of my application. The continuous case was used as the baseline to ensure a good comparison and proof of concept against other algorithms. The results for the varied time cases is where the trade-offs and benefits will become present.

4.1. Continuous Data Set. This section is dedicated to the single object continuous data set. The data set consists of 240 frames which is roughly 10 seconds of video. When an object is not present it creates a not a number (NaN) value for the centroids command of the video processing. These values are discarded in order to improve processing. After all NaN values have been discarded the resulting data set is 218 usable frame. From this data set the algorithms were run and the following results were seen and were expected for this case based on claims made in other papers.

4.1.1. *$\mathcal{KL}$ Divergence.* The following sets of figures will display how each algorithm looks when looking at the histogram of the distributions based on a creation using the multivariate Gaussian function in MATLAB. The function requires a matrix input of μ_x and μ_y as well as Σ which represents the means of each as well as the covariance matrices associated with the distributions. Since there are only two variables in this distribution the covariance matrix is a 4x4 matrix that is composed as follows

$$\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} \\ \sigma_{xy} & \sigma_{yy} \end{bmatrix} \tag{15}$$

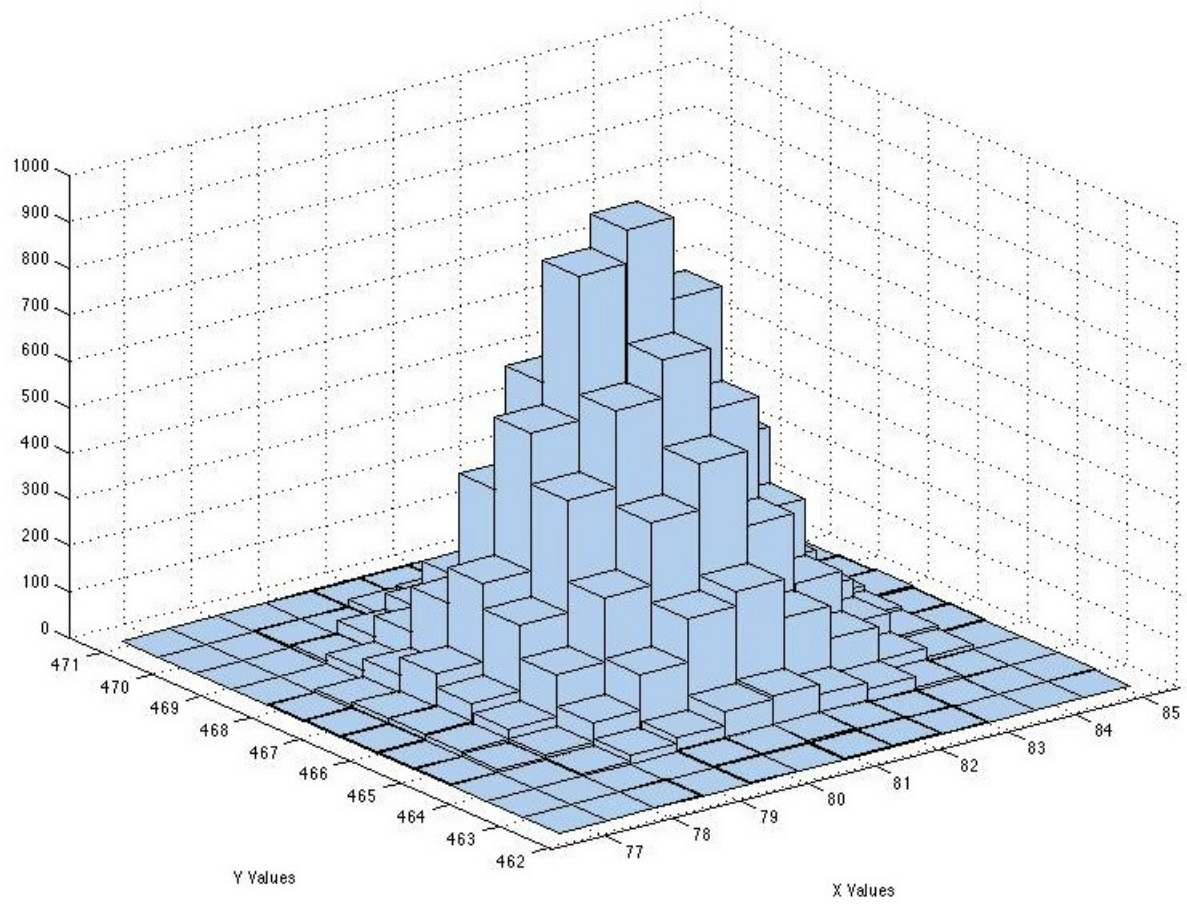


FIGURE 4. Representative Posterior Estimate with 15000 Particles

As was stated earlier the posterior distribution was created from a 15000 particle implementation of a particle filter. This was done because inherently across the board particle filters produce the lowest RMSE of Gaussian filters. As will be shown later the particle filter in this instance also produced the lowest RMSE. This is from the estimation of the unknown posterior distribution since as the amount of particles increase towards infinity the estimated posterior distribution will approach the actual posterior distribution. For fair representation each distribution was binned into 12x12 grids and are shown in figures 4, 5, 6, and 7.

Figure 4 shows the representative distribution. The other remaining distributions are shown as well. Discussed earlier about how $\mathcal{KL}$ divergence works we compare the distributions to assess how closely they measure against the representative sample. Inherently it would be logical that the particle filter run with 100 particles should match closely with the

representative distribution. This happened in practice but it was not a zero and the covariance matrices as well as the associated means were different. Figure 5 shows the distribution of points for the particle filter. It looks sparse and doesn't look like it could actually be a subset of the larger particle filter. This is not the case since the Z axis is what is important or how likely something is. The distribution for the Kalman depicts how close the spread out the covariance is meaning the it would have a flatter distribution unlike the particle which has a tight sharper covariance. The extended Kalman filter has a unique pattern which in the one dimensional case would seem more of a skewed distribution. The distribution is due to how the covariance is propagated and it is taking a linear approach to a time segment.

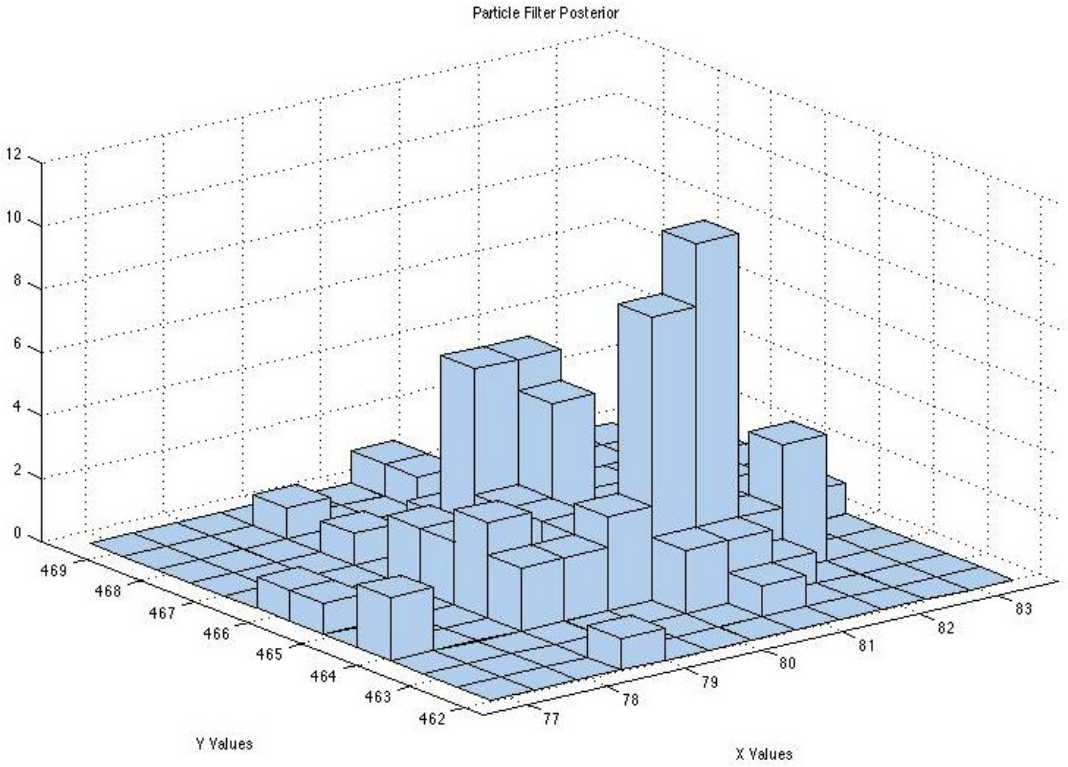


FIGURE 5. Posterior Distribution: Particle Filter

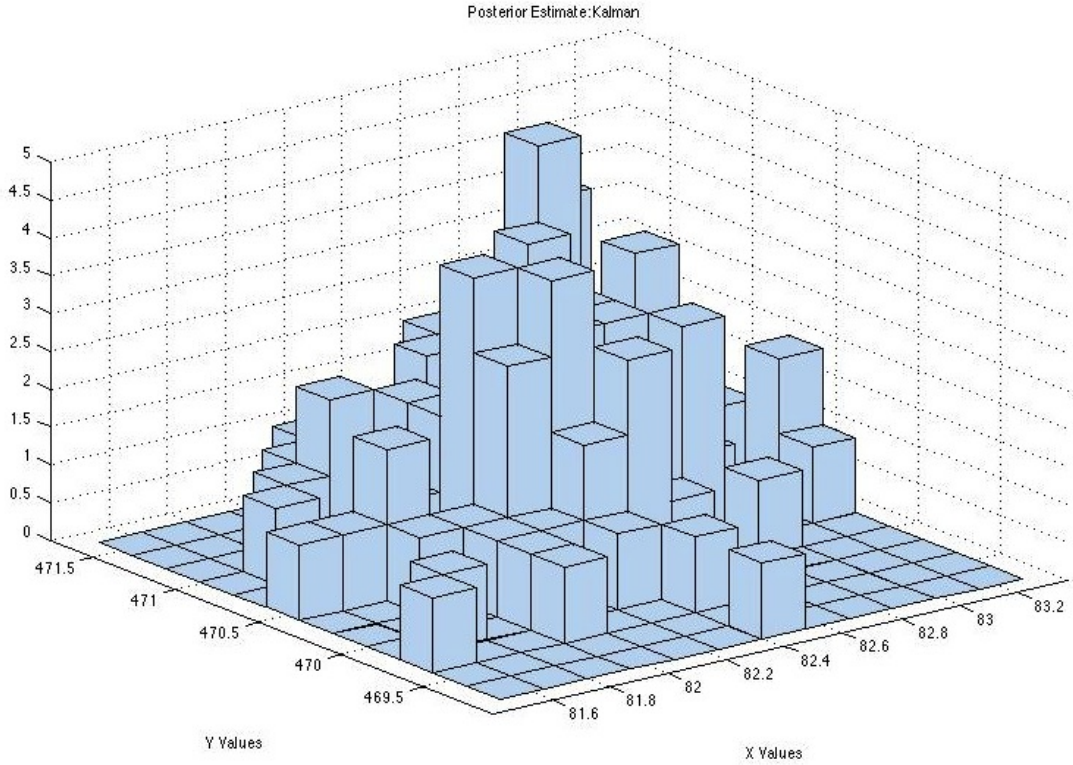


FIGURE 6. Posterior Distribution: Kalman Filter

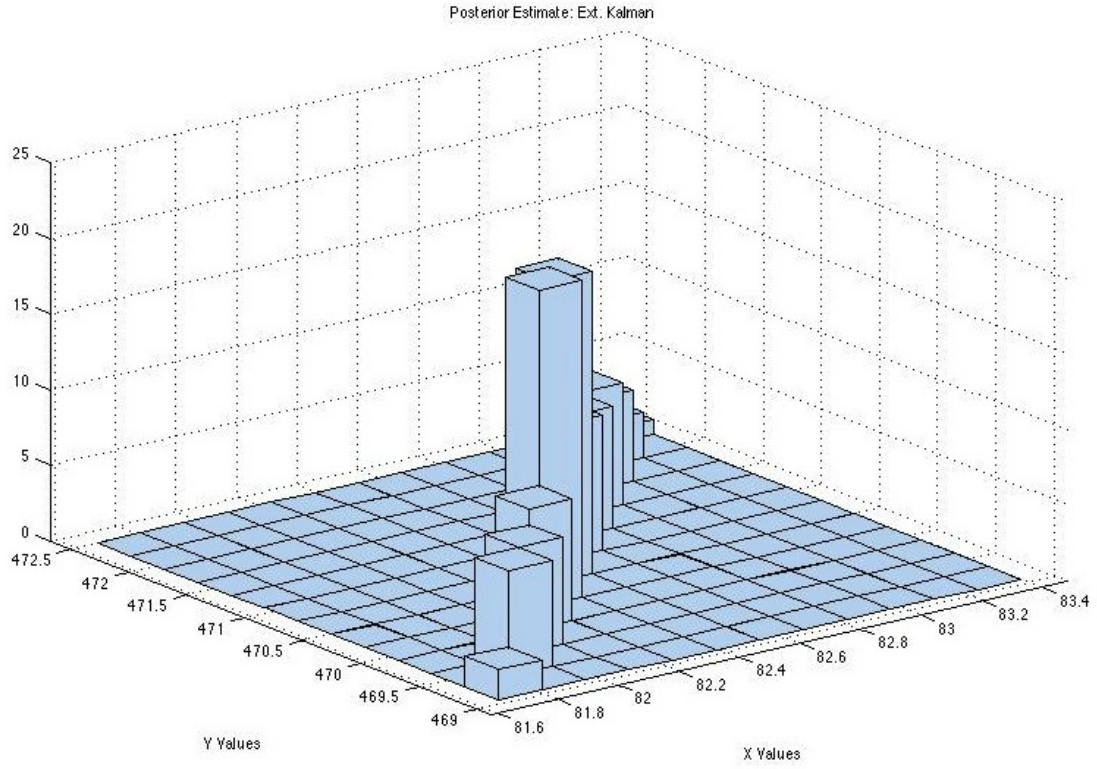


FIGURE 7. Posterior Distribution: Ext. Kalman Filter

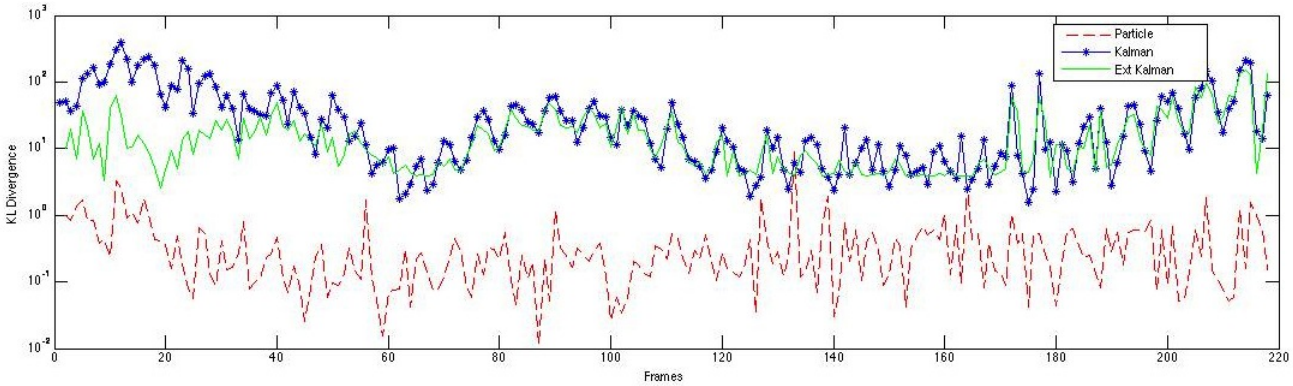


FIGURE 8. $\mathcal{KL}$ Divergence

As can be seen in figure 8 the particle filter has the lowest $\mathcal{KL}$ divergence values across the board. The extended Kalman and Kalman filter have very similar tracks with the extended Kalman marginally being the better solution.

4.1.2. *Root Mean Square Error.* The other measure used for a comparison was RMSE to determine how far off the predictions were from the measured value from the frames.

$$RMSE_{total} = \sqrt{RMSE_x^2 + RMSE_y^2} \tag{16}$$

$$RMSE_x = \sqrt{\frac{1}{n}\Sigma(x_{predict} - x_{meas.})^2} \tag{17}$$

$$RMSE_y = \sqrt{\frac{1}{n}\Sigma(y_{predict} - y_{meas.})^2} \tag{18}$$

For this measure I looked at both cumulative RMSE as well as how the RMSE was effected between individual frames. These measurements were used as a comparison to ensure proper adjustment of relative values and behavior documented in other publishings of these algorithms. As can be confirmed with the cumulative RMSE values figure 9 the particle filter maintains a lower level of overall error with the extended Kalman coming in just above that. The normal Kalman filter has a very steady error rate with minor perturbations in the cumulative case. The region of note in the cumulative case is around the 100th frame where the extended Kalman begins to have less error that the normal Kalman. This is about the point where the track becomes highly non-linear. The initial high peaks in RMSE in the particle and extended Kalman are both known behaviors figure 10. Some of that initial error on the particle filter is because for the first frame it just has to estimate over the entire frame where it thinks it will be resulting in a spike of error. Once that is corrected it achieves a more reasonable guess of future points by creating better guesses from a better set of points with which to choose from. The extended Kalman has an initial error that is relatively high and is due to the modeling equation since it is trying to model the behavior as non-linear while the track in equation is actually linear. It take a number of iterations in order for it to correct itself and once it does has a much lower error rate.

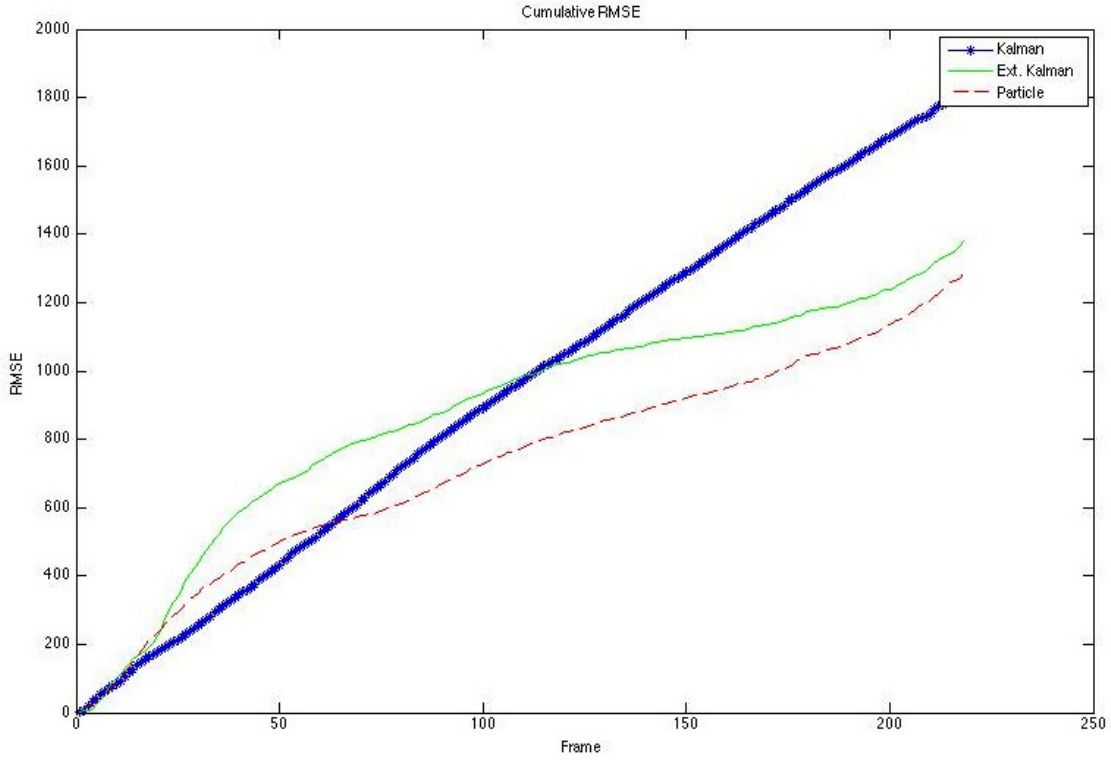


FIGURE 9. Cumulative RMSE: Continuous Data Track

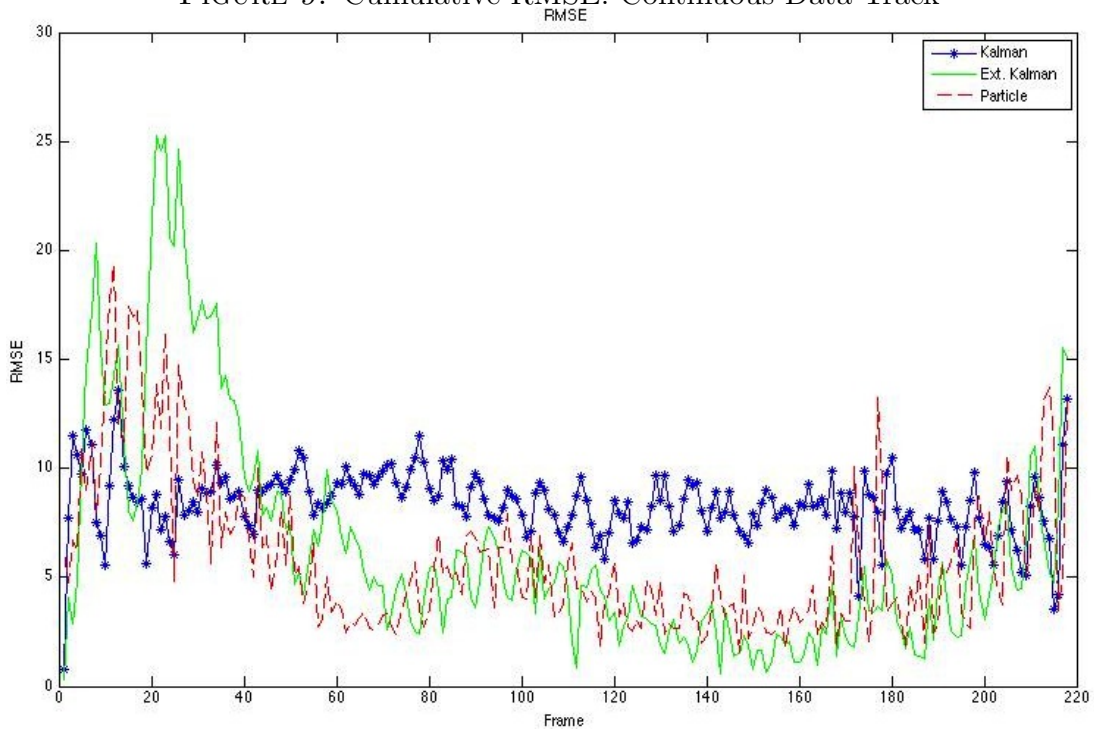


FIGURE 10. Frame by Frame RMSE: Continuous Data Track

4.2. Continuous-Multiple Time Steps. This section contains the results of predicting how RMSE and $\mathcal{KL}$ divergence were effected once the data was allowed to skip various amounts of time. This accounts for continuous time skips from two to six frames since above that the trend is rather apparent. The next section contains the results for random time skips over the video.

4.2.1. *$\mathcal{KL}$ Divergence: 2 Frame Skip.* Using the model distribution which was created from a 15000 particle filter monte carlo simulation the comparison of posterior estimates. As can be seen there is little to no difference between the particle filter and the extended Kalman. The normal Kalman has many differences and as will be shown for the RMSE the difference is related to the amount of error.

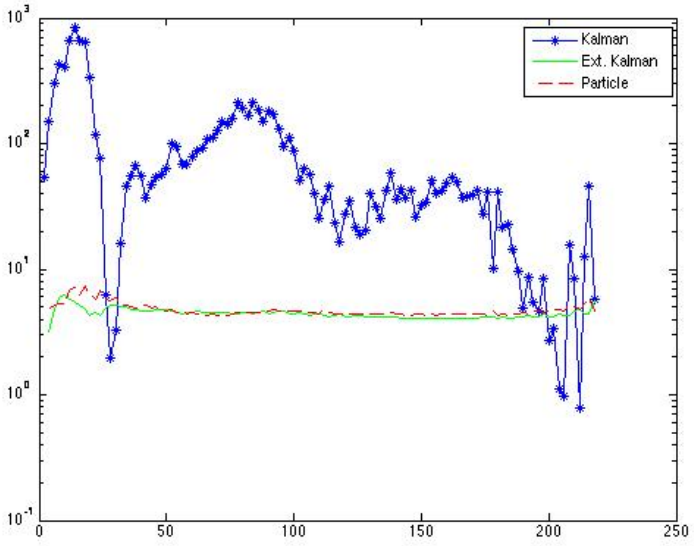


FIGURE 11. $\mathcal{KL}$ Divergence: 2 Frame Skip

4.2.2. *Root Mean Square Error: 2 Frame Skip.* The expectation for figures 12 and 13 are an increase in cumulative RMSE and frame by frame RMSE. These are both shown to be true. The gap between the extended Kalman filter and the particle filter greatly increases as opposed to the gap that was very similar shown in the purely continuous case. One peculiarity is the correlation between the $\mathcal{KL}$ divergence measure and the total RMSE. Based on the conjecture that if the posterior is adequately represented then the estimate of

position should be more accurate. The $\mathcal{KL}$ divergence is nearly identical for the particle and extended Kalman filter as shown in figure 11. This could be impart to having an unchanging tuning variable for the amount of noise expected for the velocity or position. Since I kept those static from instance to instance it could be that the amount of variability could not be properly adjusted or in a reasonable amount of time to correct itself.

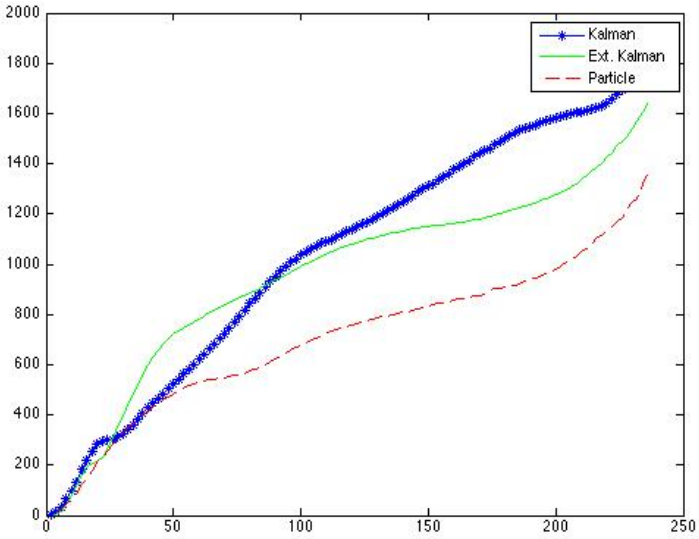


FIGURE 12. Cumulative RMSE: 2 Frame Skip

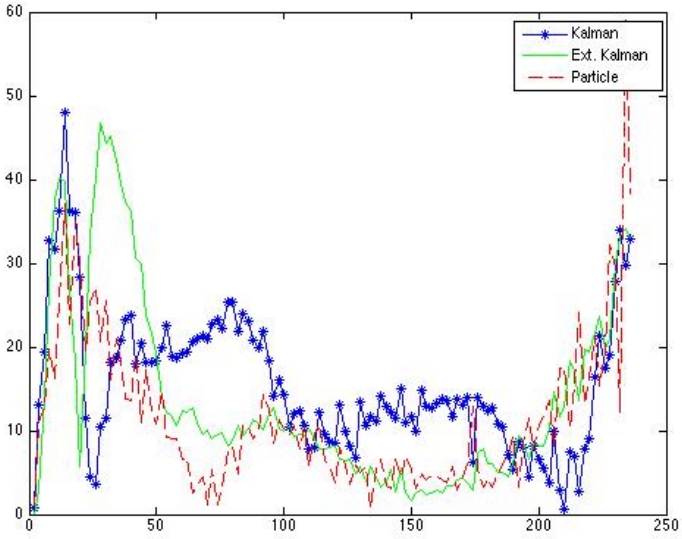


FIGURE 13. Frame by Frame RMSE: 2 Frame Skip

4.2.3. *$\mathcal{KL}$ Divergence: 3 Frame Skip.* Processing every third frame produced similar results as with processing every second frame in terms of $\mathcal{KL}$ divergence. The only thing that threw off results was that the initial values in the covariance matrix of the extended Kalman filter were nearing zero which caused the matrix to be singular or near singular when taking the inverse. That accounts for the initial spike in value in figure 14. This is the reason why there is no cumulative $\mathcal{KL}$ metric since it skewed the results.

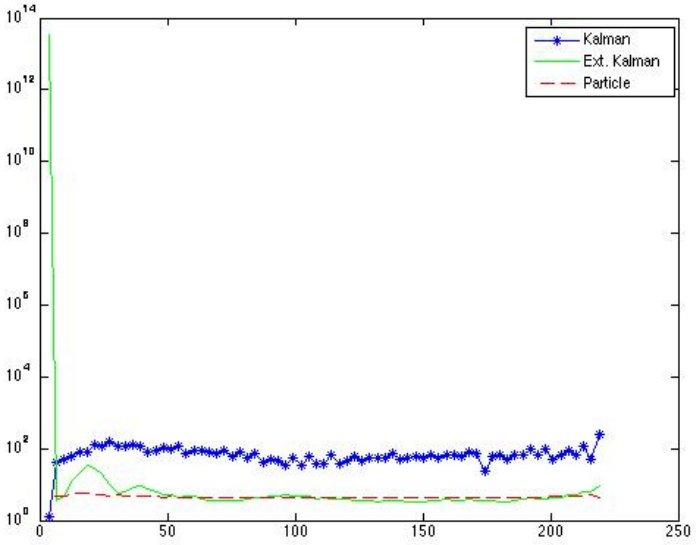


FIGURE 14. $\mathcal{KL}$ Divergence: 3 Frame Skip

4.2.4. *Root Mean Square Error: 3 Frame Skip.* The results for measuring every third frame contrast what was seen in the predictions of every second frame. The $\mathcal{KL}$ divergence is similar though out the entire video disregarding the first value. The supporting evidence is seen in the RMSE for cumulative and frame to frame. As can be seen in figures 15 and 16 they are close with little to no distinction at which outperforms the other. The only other discerning difference would be computation workload and that will be discussed later.

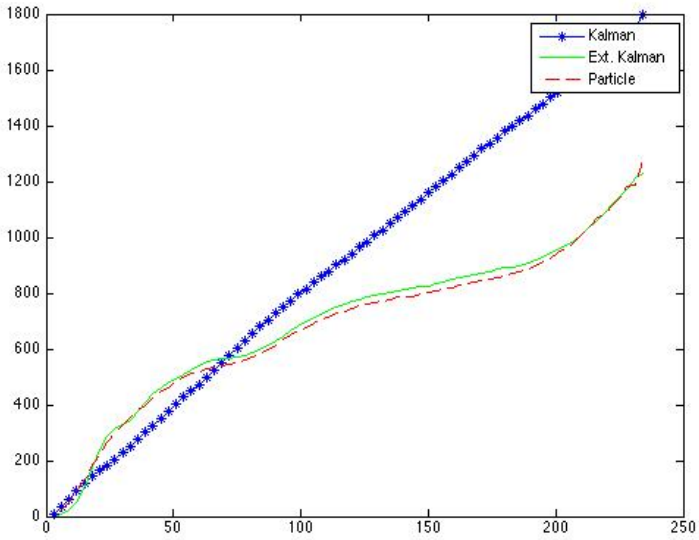


FIGURE 15. Cumulative RMSE: 3 Frame Skip

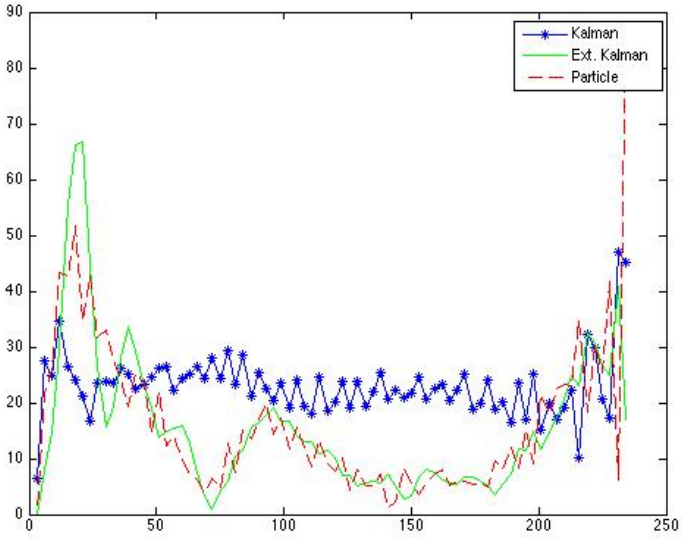


FIGURE 16. Frame by Frame RMSE: 3 Frame Skip

4.2.5. *$\mathcal{KL}$ Divergence: 4 Frame Skip.* Similar to the position estimate of every third frame, estimation of every fourth frame created very similar results with the normal Kalman being relatively out of the picture while the extended Kalman and particle filter have very close results in regards to $\mathcal{KL}$ divergence as shown in figure 17 . This also translates below when talking about RMSE.

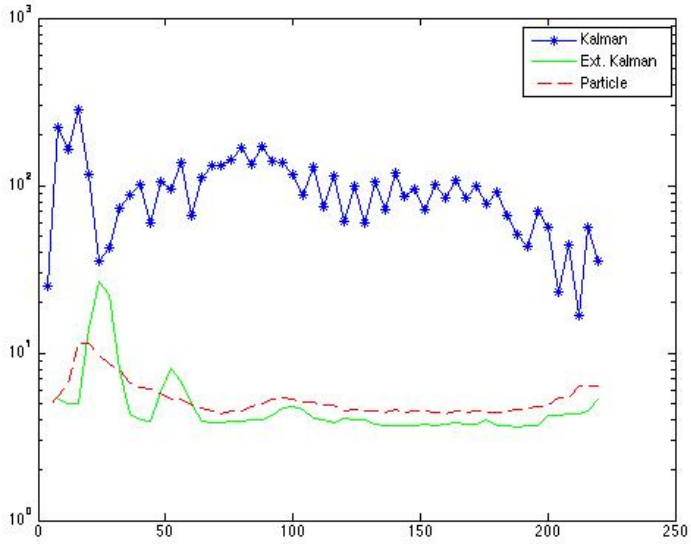


FIGURE 17. $\mathcal{KL}$ Divergence: 4 Frame Skip

4.2.6. *Root Mean Square Error: 4 Frame Skip.* Mentioned above there is very little difference when looking at cumulative RMSE. The frame by frame assessment shows that once the covariance matrix adjusts in the extended Kalman the RMSE becomes small and similar to the particle filter with an end result of no noticeable difference between the performance of each algorithm.

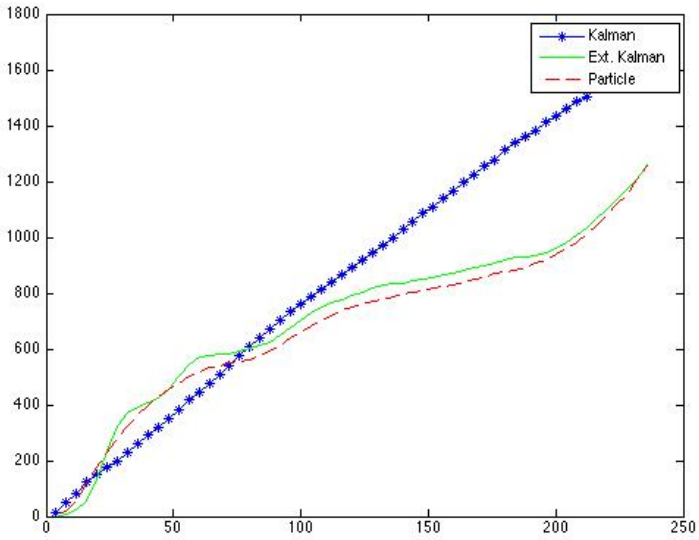


FIGURE 18. Cumulative RMSE: 4 Frame Skip

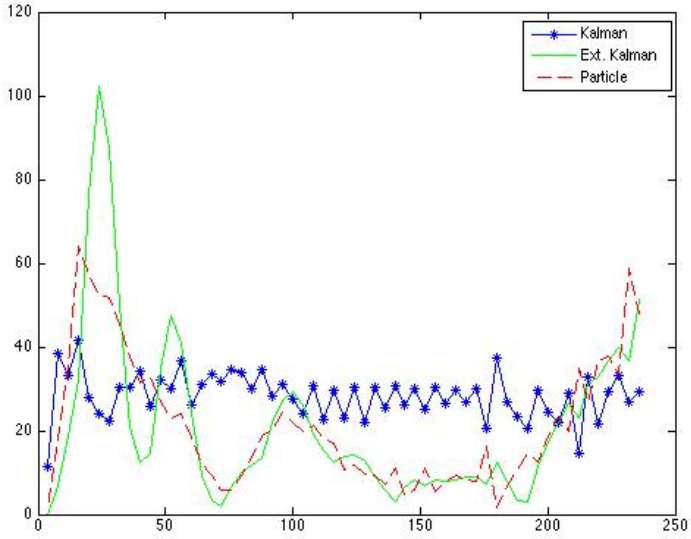


FIGURE 19. Frame by Frame RMSE: 4 Frame Skip

4.2.7. *$\mathcal{KL}$ Divergence: 5 Frame Skip.* Measurements at every third and fourth frames produced results that are to be expected where the $\mathcal{KL}$ divergence measure translates to how far off the RMSE ends up being. Once we do measurements at every fifth frame we see similar results as measuring at every second frame. This is that although the $\mathcal{KL}$ divergence is low the overall RMSE is high.

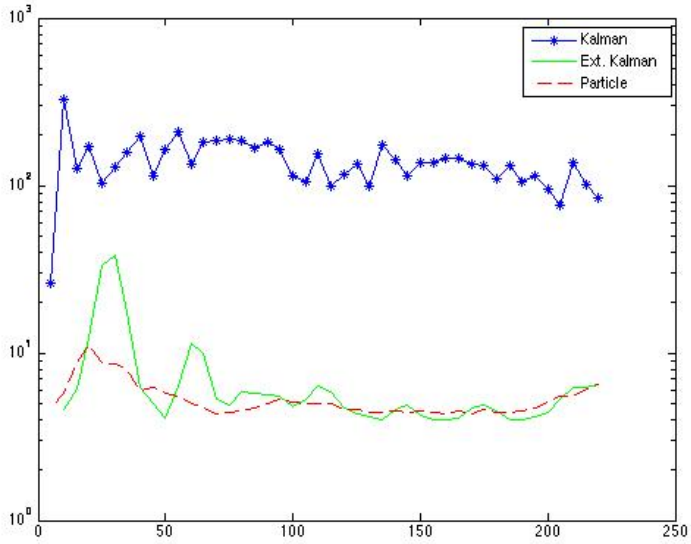


FIGURE 20. $\mathcal{KL}$ Divergence: 5 Frame Skip

4.2.8. *Root Mean Square Error: 5 Frame Skip.* Linearization of the track causes the a small amount of lag in how the algorithm adjusts for error in the covariance matrix. This is shown in the RMSE as it takes five collections to begin to compensate. This is a known feature of the extended Kalman filter. There could be ways to reduce the error by actively adjusting the tuning parameters at the expense of computation time. The result is pretty definitive as shown in figure 21 that the clear winner is the particle filter.

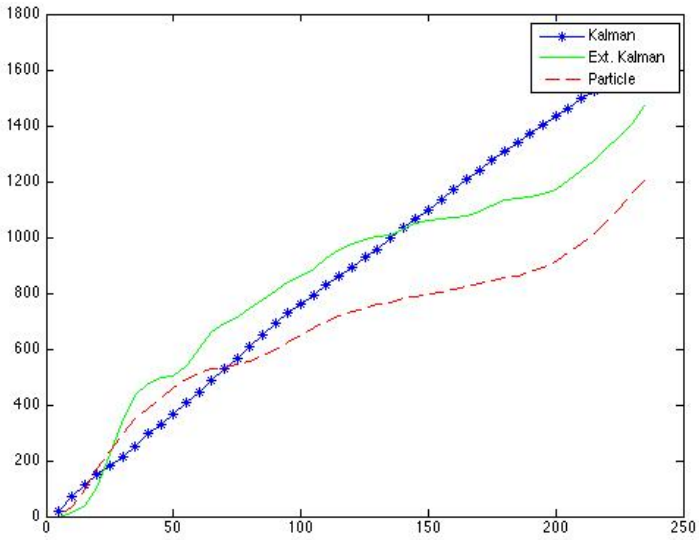


FIGURE 21. Cumulative RMSE: 5 Frame Skip

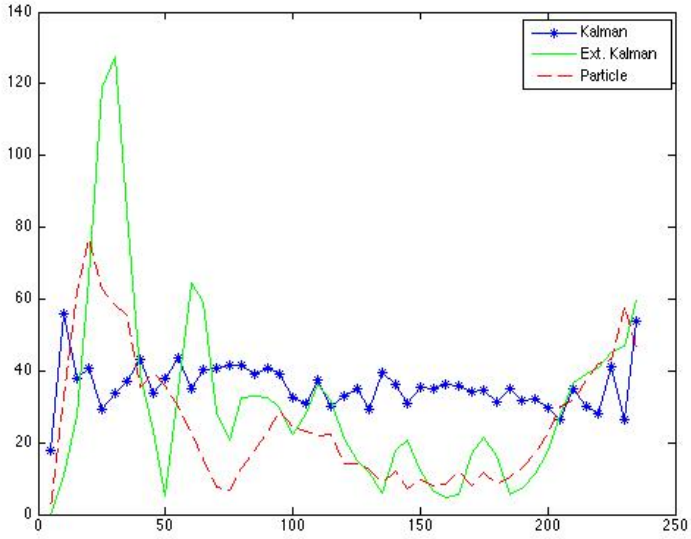


FIGURE 22. Frame by Frame RMSE: 5 Frame Skip

4.2.9. *$\mathcal{KL}$ Divergence: 6 Frame Skip.* Measuring at every sixth frame reproduced similar results as measuring every fifth frame. The same happened at very seventh and with frame so we will only show the results for every sixth frame. The only significant difference is how the correlation between $\mathcal{KL}$ and RMSE begin to diverge.

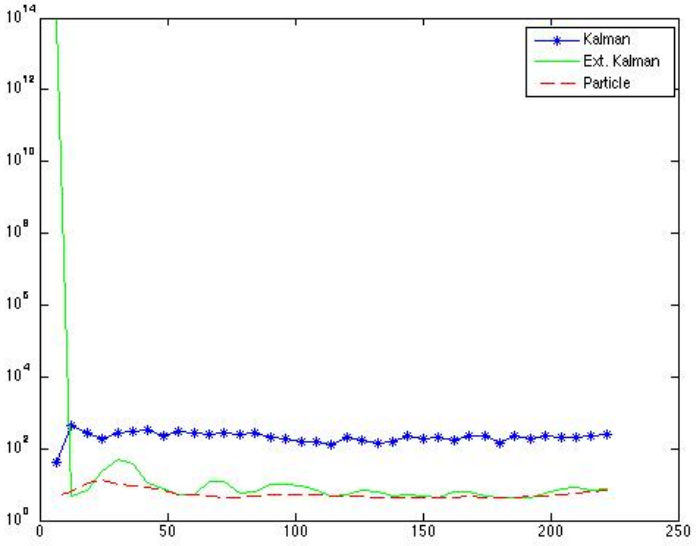


FIGURE 23. $\mathcal{KL}$ Divergence: 6 Frame Skip

4.2.10. *Root Mean Square Error: 6 Frame Skip.* Similar to the previously shown RMSE cumulative and frame by frame the amount of difference begins to show how well the particle filter continues to outperform the other algorithms.

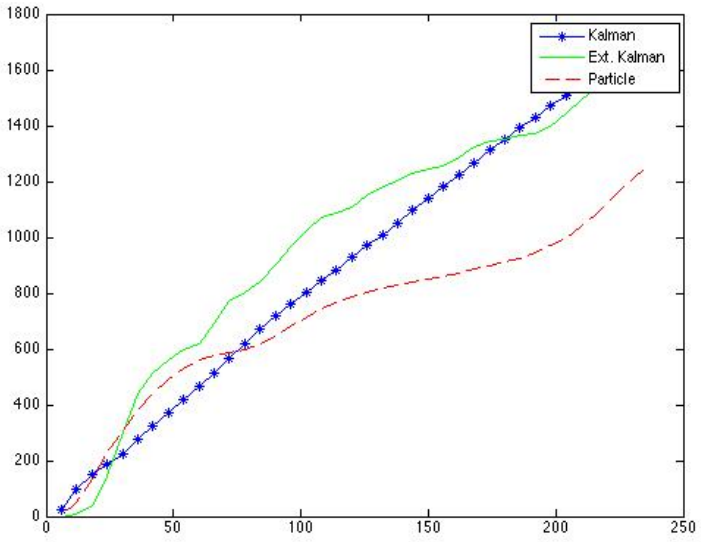


FIGURE 24. Cumulative RMSE: 6 Frame Skip

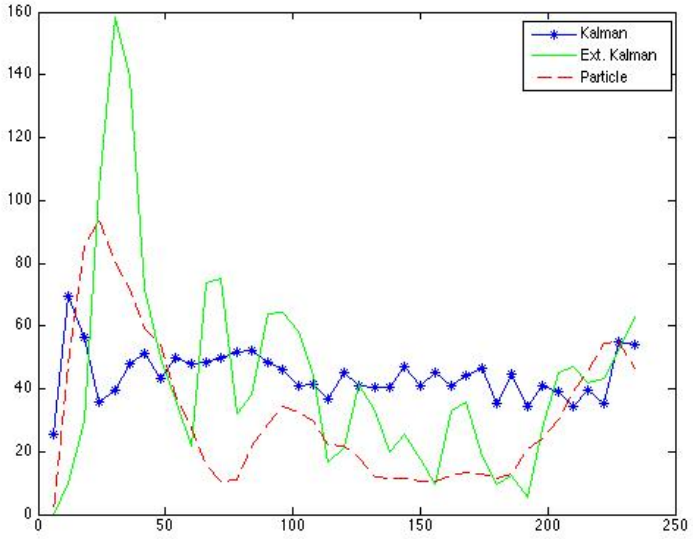


FIGURE 25. Frame by Frame RMSE: 6 Frame Skip

4.3. Variable Data Set. This section contains the results from allowing the data to be manipulated in time. The data was randomly advanced in time using a uniformly draw number between one and eight. This was to test the robustness of each algorithm and how the variation in data would effect the prediction.

4.3.1. *$\mathcal{KL}$ Divergence.* The variable data set that was used did not produce great results since the only relevant metric would be is the cumulative $\mathcal{KL}$ divergence. The problem with this was that for some initial values there would be a singularity or near singularity in the extended Kalman filter covariance matrix when it was inverted. This skewed the average since it was such a large number and caused a great increase to the average. For this reason the average for the 500 samples was left out.

4.3.2. *Root Mean Square Error.* Comparison of inter-frame errors cannot be represented since a transition from frame seven to frame nine would be different from a transition from six to nine. Due to this realization the average cumulative error was looked at. Although provided is an example of what type of behavior was viewed for a single instance.

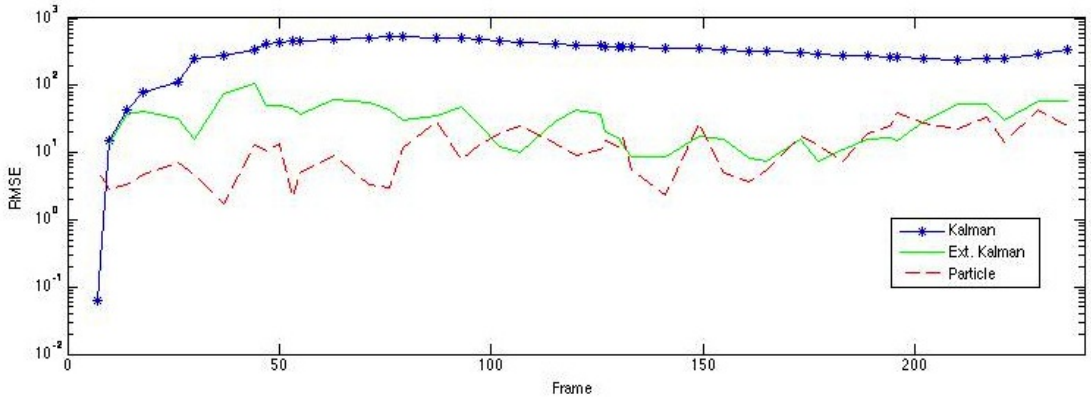


FIGURE 26. Frame by FrameRMSE: Variable Data Track

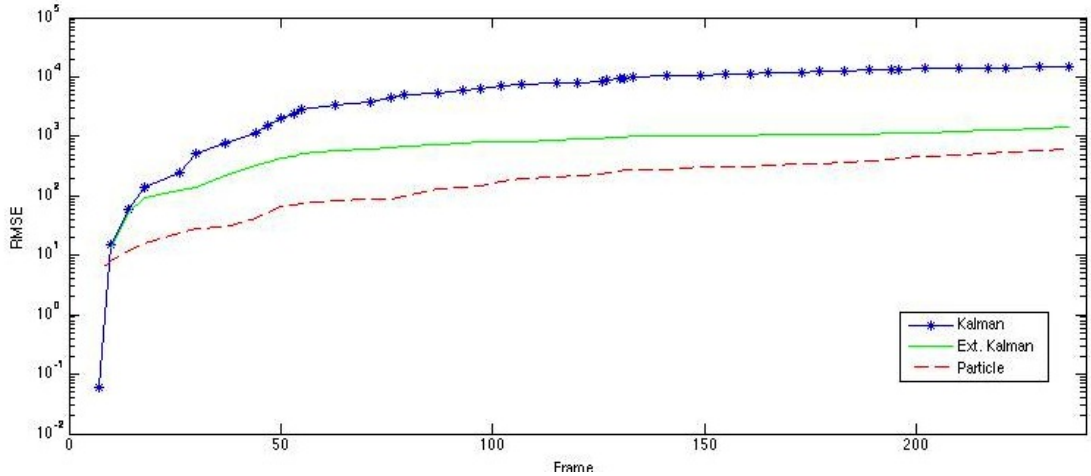


FIGURE 27. Cumulative RMSE: Variable Data Track

As can be seen in figures 26 and 27 the same type of pattern emerges where the particle filter and extended Kalman filter are on par and the particle filter ultimately being better. The normal Kalman experiences larger error rates and overestimates during the times that the track becomes more non-linear in nature. Using the time steps that are jumped between one to eight time steps and taking the average over 500 different times we can see in table 1.

Normal Klaman	Ext. Kalman*	Particle
1.4×10^5	1.1×10^5	9.4×10^4

TABLE 1. Average Cumulative Error

The extended Kalman has an asterisk due to some of the data being miss calculated. There were terms that caused some of the values to create a singularity in the update matrix causing the calculations to go either negative (which all values should be strictly positive) or go into the XX$\times 10^{11}$ neighborhood. These values were erroneous and when discarded created a much lower error rate and is thus reflected in the table value. If the erroneous data was included the average cumulative error was 7.27×10^5 which is not entirely too far off but overall worse than the normal Kalman filter.

5. Discussion

Tracking and prediction of where objects are and where they will be has many applications outside the context of this paper. There has been work at how well objects are tracked using different algorithms but not many tested the robustness of the algorithm. This is a measure how they perform outside of ideal conditions meaning strictly continuous streaming data. There are other things that increase complexity such as unregistered video but that was not considered for this paper but could be looked at in the future. What this paper intended to address was how the algorithms performed when given scenarios such as inconsistent (variable) data, such as time skips as well as consistent time skips. Looking

at two metrics of RMSE and $\mathcal{KL}$ divergence there were comparisons at how well the posterior was estimated as well as how far from the actual measured value was the estimated value.

Results for the consistent multiple time steps presented very interesting results since the $\mathcal{KL}$ divergence for the extended Kalman and particle filters were very similar but for the RMSE measurements the results were not always the same. One would think that if the posterior estimates are similar then the subsequent position estimates will likely fall within the same amount of error. This proved true for taking measurements of every third and fourth frames. The cumulative RMSE's were close and rather indistinguishable but when applied to more frames they began to diverge in the amount of cumulative error. This is perhaps due to the the extended Kalman needing better error estimates within the algorithm since each step that does not have a decent way of correction would cause the error to be systematic and would correct in the covariance but at a cost.

One to one comparison of the variable time steps would not be possible since each sequence of time steps was unique. There was data represented that showed the overall average cumulative error as well as representative output from just one such sequence. As with much of the $\mathcal{KL}$ calculations the extended Kalman had values that caused singularities within the calculations creating errors within the data and improper estimates of the locations. When these erroneous outliers were corrected the results were that the overall best was the particle filter for overall lowest error rate. The $\mathcal{KL}$ divergence was not calculated for this portion as there is not really a good metric with which to compare. Frame by frame comparison and how it relates to frame by frame RMSE would be the best but since each sequence is unique there would not be enough room within this document to show each evaluation.

Based on the research presented above for how different tracking algorithms handle nonlinear tracks with an emphasis on not having all data available shows the clear leader is the particle filter. This is shown throughout for variable sequences, time skipping sequences

and continuous time. The noted downfall is computation time which can be remedied in different instances such as parallel processing or brute force with a lot of computing power. The extended Kalman had similar performance in how well it represented the posterior distribution. This would have led me to believe the position estimate would be closer to the measured value which in some sequences proved true and false in other instances. This might be due to a lack of optimal tuning parameters which at a cost can be determined for each instance. It would be interesting to look at the RMSE and $\mathcal{KL}$ of the frame to frame to see if the RMSE gets lower when there is a three or four frame time skip. In just about every aspect the normal Kalman was outperformed except when it came to pure speed. There were times when the normal could outperform the extended Kalman which could be above six frame time steps. The extended Kalman is widely used and accepted for how well it performs in multiple instances and how computationally complex it is. The only downfall is that there needs to be a model that estimates the track with which to update and predict. The particle filter as has been stated previously does not require this which is a major benefit. Definitively we can say that if you have data collection in the sweet spot of every third or fourth time step then there is no real difference in error rates of the extended Kalman and particle filter. Outside of this sweet spot the particle filter outperforms in the post processing aspect. If we were to consider time versus error the extended Kalman would likely be the better of the three but this would only apply to serial processing of a real-time system.

Future work for development could be done to capitalize on the decrease in computational complexity between the extended Kalman and particle filter. In the paper by Gustafsson [6] the complexity of the particle filter based on the parameters given was around 62 times slower than the Kalman filter. The table below shows the computational complexity of each algorithm where n_x is the amount of state variables that are updated at each time step.

Particle	Kalman	Ext. Kalman
$\mathcal{O}(Nn_x^2)$	$\mathcal{O}(2n_x^3)$	$\mathcal{O}(4n_x^3)$
8000	128	256

TABLE 2. Computational Complexity Measures

Based on the complexity measures listed above as the number of parameters increases there could be a point where they are equivalent but tracking that amount of state parameters is unlikely. In order to exploit the decrease in complexity it could be feasible to create a hybrid algorithm that accounts for when the predictions are likely to be indistinguishable. This would happen when Δt was three or four. The switch would be seamless as it would just reference the last time step and assess the covariance of the previous usage then use that in its prediction. This algorithm would exploit the intrinsic value of both algorithms while maintaining the error that would be comparable to running the particle at all times.

Bibliography

[1] Miodrag Bolić, Petar M. Djurić, Sangjin Hong, "Resampling Algorithms for Particle Filters:A Computational Complexity Perspective",EURASIP Journal on Applied Signal Processing 2004:15, 2267-2277

[2] Bovik, Alan C. The Essential Guide to Video Processing. Amsterdam: Academic/Elsevier, 2009. Print.

[3] DiValentin, L.(2013). Initial Application of Fault Detection Techniques to Cybersecurity Intrusion Detection. Retrieved from http://libra.virginia.edu/catalog/libra-oa:3628

[4] Einicke, G.A.; White, L.B. "Robust Extended Kalman Filtering" IEEE Trans. Signal Processing 47 (9): 2596-2599. 1999

[5] N.J. Gordon, D.J. Salmond, A.F.M. Smith, "Novel approach to nonlinear/non-Gaussian Bayesian state estimation" IEEE proceedings-F, Vol. 140, No. 2, April 1993

[6] Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell,Jonas Jansson, Rickard Karlsson, Per-Johan Nordlund. "Particle Filters for Positioning, Navigation and Tracking", Signal Processing, IEEE Transactions Volume:50, Issue:2, Feb 2002

[7] Steven A Holmes, Georg Klein, and David W Murray, An $\mathcal{O}(N^2)$ Square Root Unscented Kalman Filter for Visual Simultaneous Localization and Mapping, University of Oxford.

[8] Md. Zahidul Islam, Chi-Min Oh and Chil-Woo Lee,"Video Based Moving Object Tracking by Particle Filter", International Journal of Signal Processing, Image Processing and Pattern Vol. 2, No.1, March, 2009

[9] Yan-Bin Jia, *Discrete-time Kalman Filter and the Particle Filter*, Com S 477/577 Notes, https://www.cs.iastate.edu, Dec 11, 2014, Accessed 9 April 2015

[10] S. J. Julier and J. K. Uhlmann. "A New Extension of the Kalman Filter to Nonlinear Systems", Proc. of AeroSense: The 11th Int. Symp. on Aerospace/Defence Sensing, Simulation and Controls, 1997

[11] R.E. Kalman,"A New Approach to Linear Filtering and Prediction Problems", Transactions of the ASME Journal of Basic Engineering, 82 (Series D): 35-45. 1960.

[12] Kennedy, J., Eberhart, R. "Particle Swarm Optimization", Proceedings., IEEE International Conference on Neural Networks, 1995.

[13] Cody Kwok, Dieter Fox, Marina Meila, **Real-time Particle Filters**, Dept. of Computer Science & Engineering, Dept. of Statistics, University of Washington

[14] M. Marrón, J.C. García, M.A. Sotelo, M. Cabello, D. Pizarro, F. Huerta, J. Cerro, "Comparing a Kalman Filter and a Particle Filter in a Multiple Objects Tracking Application", Electronics Department, University of Alcala, Alcalá de Henares, SPAIN

[15] Madhur Mehta, Chandni Goyal, M.C. Srivastava, R.C. Jain. "Real Time Object Detection and Tracking: Histogram Matching and Kalman Filter Approach", IEEE, 796-801 Volume 5, 2010.

[16] Müller, P. 'Monte Carlo Integration in General Dynamic Models' Contemporary Mathematics, 115, 145-163, 1991

[17] Kaare Brandt Petersen, Michael Syskind Pedersen, **The Matrix Cookbook**, Version: November 15, 2012, http://matrixcookbook.com

[18] Mahesh Vemula, Mónica F. Bugallo, Peter M. Djurić, "Performance Comparison of Gaussian-Based Filters Using Information Measures", IEEE Signal Processing Letters, Volume 14, No. 12, Dec 2007

[19] Bin Yu, Tutorial: Information Theory and Statistics, ICMLA 2008, San Diego, Statistics Department, EECS Department, UC Berkeley,

1. First Appendix

```
temp = 0;
p_kal =[];
RMSE_kal = 0;
RMSE_kal_cum = [];
RMSE_kal_f = [];
for i = 2:dt:240
    file = 'dog';
    num = num2str(i);
    ext = 'jpg';
    filename = strcat(file,num,'.',ext);
    img = imread(filename);
    %imshow(img)
    gray_stills = rgb2gray(img);
    num_past = num2str(i-1);
    file_past = strcat(file,num_past,'.',ext);
    img_past = imread(file_past);
    gray_stills_past = rgb2gray(img_past);
    gray_diff = imabsdiff(gray_stills_past,gray_stills);
    %imshow(gray_diff)
    thresh = 0.05;
    bw = (gray_diff >= thresh *255);
    bw2 = bwareaopen(bw,20,8);
    SE = strel('disk',2,4);
    bw3 = imdilate(bw2,SE);
    bw3 = imfill(bw3,8,'holes');
    L = bwlabel(bw3);
    s = regionprops(L, 'Area', 'Centroid');
    centroids = nan(1, 2);
```

```
    if size(s,1) == 0
    else
        area_vector = [s.Area];
        [tmp, idx] = max(area_vector);
        centroids = s(idx(1)).Centroid;
    end
    if isnan(centroids) == 1
        continue
    end
%%
    u = 1;
    if temp == 0
        Q= [centroids(1,1); centroids(1,2); 0; 0]; %initized state--it has
            four components: [posX; posY; velX; velY]
        Q_estimate = Q;
        noise_mag = .3; %process noise
        noise_x = 5;  %measurement noise in the vertical direction (x axis).
        noise_y = 5;  %measurement noise in the horizontal direction (y axis).
        noise = [noise_x 0; 0 noise_y];
        proc_noise = [dt^4/4 0 dt^3/2 0; ...
            0 dt^4/4 0 dt^3/2; ...
            dt^3/2 0 dt^2 0; ...
            0 dt^3/2 0 dt^2].*noise_mag^2; % Convert the process noise (stdv)
               into covariance matrix
        P = proc_noise; % covariance matrix
        A = [1 0 dt 0; 0 1 0 dt; 0 0 1 0; 0 0 0 1]; %state update matrice
        B = [(dt^2/2); (dt^2/2); dt; dt];
        % B = 0;
        C = [1 0 0 0; 0 1 0 0];
        Q_loc_estimate = [];%  position estimate
        vel_estimate = []; % velocity estimate
```

```
        P_estimate = P;
        predic_state = [];
        predic_var = [];
        temp = 1;
    end
    Q_loc_meas = [centroids(1,1); centroids(1,2)];
    Q_estimate = A * Q_estimate + B * u;
    predic_state = [predic_state, Q_estimate(:,1)] ;
    P = A * P * A' + proc_noise;
    predic_var = [predic_var; P] ;
    K = P*C'*inv(C*P*C'+noise);
    Q_loc_estimate = [Q_loc_estimate, Q_estimate(1:2)];
    vel_estimate = [vel_estimate, Q_estimate(3:4)];
    kal_x_loc = Q_estimate(1);
    kal_y_loc = Q_estimate(2);
    % Update the state estimate.
    if ~isnan(Q_loc_meas)
        Q_estimate = Q_estimate + K * (Q_loc_meas - C * Q_estimate);
    end
    P =  (eye(4)-K*C)*P;
    p_kal = cat(3,p_kal, P(1:2,1:2));
    RMSE_kal_x = (centroids(1,1)-kal_x_loc)^2;
    RMSE_kal_y = (centroids(1,2)-kal_y_loc)^2;
    RMSE_kal = RMSE_kal + sqrt(RMSE_kal_x + RMSE_kal_y);
    RMSE_kal_cum = [RMSE_kal_cum;RMSE_kal];
    RMSE_kal_frame = sqrt(RMSE_kal_x + RMSE_kal_y);
    RMSE_kal_f = [RMSE_kal_f;RMSE_kal_frame];
end
```