Vehicle Detection System using Gaussian Mixture Models and Recursive Bayesian Estimators

Anurag Banerjee

Abstract—We implement a system for vehicle detection and tracking from traffic video using Gaussian mixture models and Bayesian estimation. In particular, the system provides robust foreground segmentation of moving vehicles through a K-means clustering approximation as well as vehicle tracking correspondence between frames by correlating Kalman and particle filter prediction updates to current observations through the solution of the assignment problem. In addition, we conduct performance and accuracy benchmarks that show about a 90 percent reduction in runtime at the expense of reducing the robustness of the mixture model classification and about a 30 percent and 45 percent reduction in accumulated error of the Kalman filter and particle filter respectively as compared to a system without any prediction.

1 Introduction

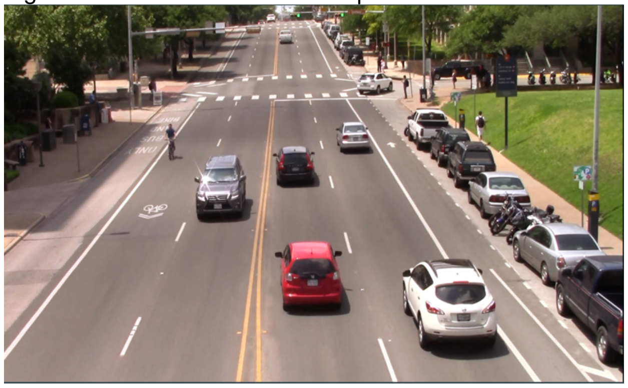
SINCE the rapid growth of transportation networks in U.S. cities, traffic monitoring has become an increasingly important feature for developing modern urban infrastructures. In particular, applications such as traffic congestion analysis, automatic highway accident detection, and even contextual surveillance systems have been widely explored from a variety of different signal processing perspectives [2]. With the appropriate signal processing techniques, one can extract information such as vehicle positions, speeds, or even the total number of cars on the road, etc. Two basic features of any robust traffic detection system are predicated upon detecting vehicles and tracking vehicles from traffic video analysis. Because of the inherent dynamic nature of the traffic monitoring task, no single "best" method exists for approaching these tasks. Traditionally, vehicle detection methods are broken up into adaptive mixture model approaches or motion-based segmentation approaches. Tracking, on the other hand, has seen a majority of its progress from the estimation theory perspective. Within the following paper, we explore the implementation of a traffic detection system using adaptive background mixture models for solving the foreground segmentation problem as well as implementing a recursive Bayesian estimator approach for tracking vehicles based on traffic video.

2 Experimental Setup

For the experimental setup, a Canon VIXIA camcorder was used to record test videos for vehicle detection and tracking tasks. The camera recorded video at a resolution of 1920x1080 at a frame rate of 60 progressive frames per second. All videos were recorded on a stationary camera tripod in order to minimize the need for motion compensation and were recorded at the intersection of Dean Keaton and Speedway as shown below.

In addition, the camera position was decided in order to minimize perspective distortions and reduce the probability of object occlusion in order to improve the performance of the vehicle tracking application. In regards to the processing, all traffic detection tasks were performed in a batch processing manner using MATLAB due to the computational intensity of the adaptive model process. For my final project, We implemented two key components of a basic traffic detection system: vehicle detection, and vehicle tracking using mixture models and a Bayesian Estimator approach as will be described in the proceeding sections.

Fig. 1. Traffic Video Camera Setup



3 Mixture Model Based Segmentation

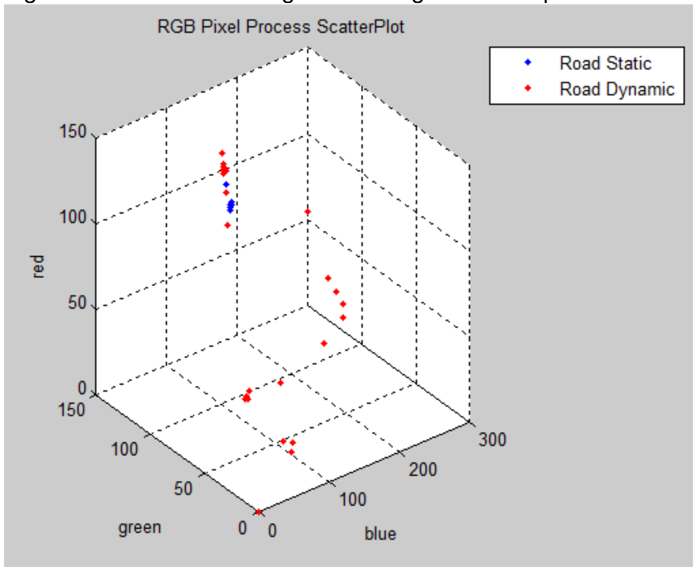
We use adaptive mixture models as the basis for solving the foreground segmentation problem due to its robustness to dynamic scene changes for classifying pixels. Within the following section, we will describe the basics of mixture models and how it applies to traffic detection, the algorithm for classifying foreground pixels from background pixels, and the specifics of updating the models based on the learning rules of the system. Mixture models are probabilistic models with the underlying assumption that, for a given data set, the observations are generated from a mixture of a finite number of distributions with unknown parameters. For our application, we will use an adaptive mixture model in order to cluster or classify background pixels that correspond to static scene elements such as the road and buildings from foreground pixels that correspond to dynamic elements such as moving vehicles[1]. We require stationarity in order to conduct an EM algorithm, so we opt instead to use an approximate K-means clustering approach. Within the video, we treat each spatial coordinate as a pixel process wherein each new frame provides a new pixel observation for that specific coordinate and at each frame we have information about the current observation as well as the past observations as described in the model below

$$\{X_1, ..., X_t\} = \{I(x_0, y_0, i) : 1 \leq i \leq t\}$$

The goal of our foreground segmentation component is to analyze these incoming pixel observations and, based on their relative fitness with our proposed Gaussian mixture model, classify them as belonging to the foreground or background. In particular, the underlying assumption is that for static elements of a video, the mixture model that describes the pixel process will be relatively constant with a corresponding distribution that has low variance and an abundance of prior evidence. In contrast, for moving dynamic objects within the scene, we would expect a poor match with the existing distributions and a corresponding higher variance because of the lack of prior evidence.

As a basic illustration of the concept, we have compiled a 3-d pixel process scatter plot for two pixel locations within a video. Each point in the graph corresponds to the RGB intensity at a specific instant of time for the same pixel location in the video. The blue points correspond to a static background pixel centered on a road while the red points correspond to a road pixel that is crossed by a red car during the course of the video.

Fig. 2. Pixel Process Background Foreground Comparison



As we can see, for pixel processes that do not change much over time, there is a corresponding low variance in pixel observations. On the other hand, pixel processes that are subject to dynamic elements from a moving foreground object will exhibit a noticeably higher variance[6]. Thus, we can be confident that our mixture model process is appropriate for our foreground segmentation task. Now that we have motivated the use of our model for segmentation, we will now go into the details of how the algorithm classifies the pixel observations and how the model is updated over time.

3.1 Algorithm

The basis for matching an incoming pixel observation is testing to see whether it falls within a range of one of the mixture model distributions. For any pixel observation, we can define the probability of observing it as

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

where we define the multivariate normal distribution as

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T \Sigma^{-1} (X_t-\mu_t)}$$

for the RGB pixel value. In our case, K is the number of mixture model distributions that explain each pixel process and is assumed to be constant throughout the video.

3.1.1 Learning Characteristics

The process for classifying an incoming pixel is as follows: we define a pixel observation as matched with a mixture model distribution if it falls within about 1.7 standard deviations of the distribution. If no match is found, we find the model with the lowest fitness, where fitness is defined as w/σ and replace it with one centered around the current pixel observation and a collapsed covariance based upon the current pixel observation. Fitness corresponds to a means by which to classify background pixels based on prior pixel observation evidence.The assumption then becomes pixels that match to models with higher fitness ratings tend to come from static background elements while pixels that match to models with lower fitness tend to come from dynamic foreground elements[5]. This comes directly from the rules that we use to update the models where a match is found shown below:

$$\omega_{k,t} = (1-\alpha)\omega_{k,t-1} + \alpha(M_{k,t})$$
$$\mu_t = (1-\rho)\mu_{t-1} + \rho X_t$$
$$\sigma_t^2 = (1-\rho)\sigma_{t-1}^2 + \rho(X_t-\mu_t)^T(X_t-\mu_t)$$

where α is the learning rate of the system and ρ is a term proportional to the product of the learning rate and the conditional probability of the current pixel observation.

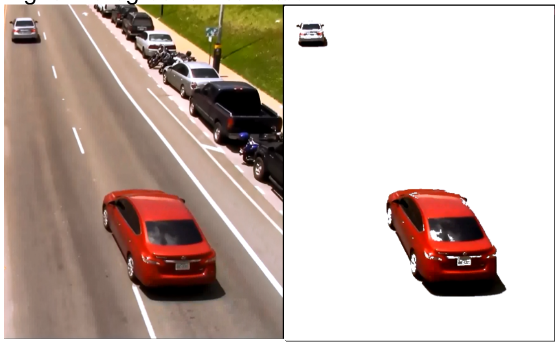
3.1.2 Classification Procedure

Now, as the algorithm proceeds, we can infer that as a model matches with more and more recent pixel observations, it's fitness metric will go up because of it's increased updated weight and decreased updated variance. Meanwhile, for a transient element of the scene corresponding to a moving foreground object, the pixel observations will have little to no prior evidence and correspondingly will be matched with a poor fitness model or noticeably increase the variance of an existing distribution. For a given pixel observation, once the corresponding model distributions are updated, we rank them again according to the updated fitness values and then make the assertion that the first B distributions are assumed to be part of the background process wherein B is defined as

$$B = argmin_b \left(\sum_{k=1}^{b} \omega_k > T \right)$$

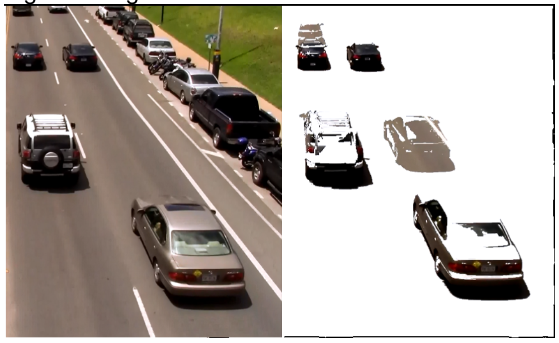
A difficult aspect of the segmentation problem is deciding on parameters for α, the learning rate of the system, and for T, the portion of the video data that should be accounted for as background processes generally. As the learning rate increases, we incorporate moving objects into the background estimation more quickly and can correspondingly collect evidence for background processes more efficiently. This performs well for moving foreground objects that have stopped, such as a car going into park. However, by setting the learning rate too high, we also risk creating an unstable foreground segmentation algorithm that can begin to classify even clear moving foreground objects as part of the background. However, if we are too conservative with our learning rate parameter tuning, then we incur the cost of a much higher convergence time for a stable background model. The net effect is that, the initial conditions of the video can vastly misrepresent what is foreground and what is background respectively. Changing the value of T accordingly changes the models assumptions of the background distribution. In particular, a smaller value of T corresponds to assuming that the background model is unimodal, such as with a static road with relatively constant lighting conditions. However, as we increase T, we begin to assume a multi-modal background distribution [4]. This performs very well for videos whose backgrounds contain some sort of periodic motion such as a stoplight blinking or a waving flag in the wind. However, this also runs the risk for incorporating cars into a background when the traffic patterns are relatively periodic and the cars generally share the same color profiles. For my project, I manually tuned these parameters according to my own subjective quality assessment which I provide an example of below in figure 3 below.

Fig. 3. Foreground Estimation



In this case, the tuning parameters work very well in detecting the vehicles. However, by lowering both the parameters α and T, we see that shadow artifacts come up as in the figure below.

Fig. 4. Foreground Estimation with Shadow Artifacts



This is due to the background distributions not collecting enough evidence to stabilize appropriately resulting in a delayed shadow artifact in the exact shape of the prior foreground object ”following” the moving vehicle temporarily.

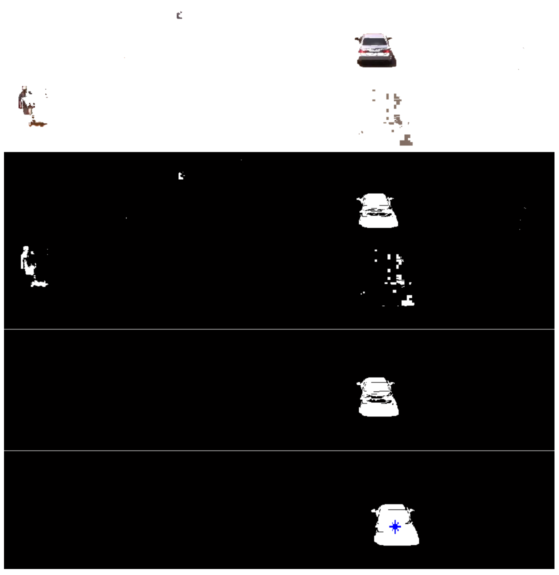
4 Bayesian Estimator Tracking

Once the foreground segmentation problem is solved, we’d like to create a vehicle tracking component for our traffic detector based on a recursive Bayesian estimator. In particular, although we can isolate vehicles with the foreground segmentation procedure from the previous section, the traffic detector has no higher-level correspondence between individual foreground pixels and what vehicle, if any, they may belong to. In light of this, our vehicle tracking system uses both a Kalman filter and particle filter on the connected components of the foreground pixel map and solves the combinatorial optimization assignment problem in order to establish a correspondence between connected components in successive frames. Within the following sections, we will describe the algorithm setup, details of the update equations and model assumptions and the necessity of solving the assignment problem in order to match estimator models between frames.

4.1 Algorithm Setup

We simplify the computation for the Kalman Filter and particle filter equations by calculating the centroids of the connected components in each foreground pixel frame. These centroid locations act as the observation variables of interest in the measurement updates for both our Bayesian Estimator tracking applications. In addition, we use the connected components algorithm to prune the foreground pixel image of components below a preset size threshold. The reason being that these generally will correspond to smaller objects such as pedestrians or camera movements[7]. The remaining connected components in the image can be reasonably assumed to be vehicles then as shown in the connected component pruning process in figure 5 below.

Fig. 5. Connected Component Cleanup



4.2 Kalman Filtering Model

The Kalman Filter is an algorithm based on Bayesian statistical inference in order to produce estimates of some dynamic model in the presence of noise. The Kalman filter has seen broad use in a variety of tracking applications because of its robustness in prediction by incorporating the error covariance of its state estimates in the algorithm procedure. For our purpose, we use the Schmidt modification to the basic Kalman Filter, which breaks up the algorithm into a separate prediction and measurement update step, in order to track the vehicle's position over time. The Kalman Filter is recursive in nature, thus for each proposed Kalman model, we must maintain both the state estimate and the error covariance from the previous stage. For reference, we have included the explicit algorithm computational steps below including the calculations of: K, the optimal Kalman filter gain, e, the innovation process, and Re, the innovation covariance.

$$x_{i+1} = F_i x_i + G_i u_i$$

$$y_i = H_i x_i + v_i$$

$$\hat{x}_{i|i} = \hat{x}_{i|i-1} + K_{f,i} e_i ,$$

$$P_{i|i} = P_i - P_i H_i^* R_{e,i}^{-1} H_i P_i$$

$$e_i = y_i - H_i \hat{x}_{i|i-1}$$

$$K_{p,i} = (F_i P_i H_i^* + G_i S_i) R_{e,i}^{-1}$$

$$R_{e,i} = R_i + H_i P_i H_i^*$$

The most difficult aspect of any Kalman Filtering solution comes from selecting an appropriate model for the system, which may have widespread implications for the accuracy and performance of the algorithm for the specific task. For vehicle tracking, we used a constant velocity model where the system dynamics are modeled by

$$F = \begin{vmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}.$$

Although this model is intuitively simple to understand, it performs well only in a limited number of situations where the cars roughly move at the same speed such as in a highway system. However, the traffic present at a stoplight intersection will be subject to accelerations associated with slowing down for red lights and speeding up for green lights. As a response to the deficiencies of our system dynamics, we played with the process and observation noise assumptions in order to reduce the overall prediction error of the Kalman filter as we will show in Section 5.

4.3 Particle Filtering Model

By contrast, the particle filter is an approximate Bayesian computational model designed to produced estimates of a dynamical system in the presence of noise. As with any other estimator used in the optimal filtering problem, the goal is to estimate the state of the dynamical system at time k given all measurements up to and including k denoted by $y_{1:k}$ Within the context of a Bayesian setting, we must accordingly compute the distribution $p(x_k|y_{1k-1})$ recursively through a similar prediction and update step like the Kalman filtering model. We show the particle filter steps explicitly below. Note that these particle filter algorithm steps are applications of Bayes rule and the Chapman-Kolmogorov equation respectively.

$$\boxed{p(x_k|y_0,\cdots,y_{k-1}) \xrightarrow{\text{updating}} p(x_k|y_0,\cdots,y_k) = \frac{p(y_k|x_k)p(x_k|y_0,\cdots,y_{k-1})}{\int p(y_k|x'_k)p(x'_k|y_0,\cdots,y_{k-1})dx'_k}}$$

$$\boxed{\xrightarrow{\text{prediction}} p(x_{k+1}|y_0,\cdots,y_k) = \int p(x_{k+1}|x_k)p(x_k|y_0,\cdots,y_k)dx_k}$$

However, these conditional distributions are too complex to compute for many applications in general, and so the particle filter is predicated upon using Monte Carlo methodologies in order to approximate these filtering distributions. In particular, we employ sequential importance sampling (SIS) in order to approximate the posterior distribution at a time *k-1* with a weighted set of samples, or particles, and recursively update these particles in order to obtain an approximation to the posterior distribution at the next time step. The underlying premise of SIS is to ensure that these particles and their corresponding weights are updated in order to approximate the posterior distribution at the next time step. For our tracking application, this is done by weighting particles according to their perceived deviation from the current observation and updating the weights of all particles appropriately. By comparison, particle filters are accordingly much more computationally intensive than a Kalman filtering approach. A particular point of emphasis becomes choosing the number of particles for representing the system in order to optimize for available computational resources and to maintain effective sampling diversity.

4.4 Adding and Removing Models

Because the number of vehicles throughout the video is subject to change, we must maintain a pool of filter models in order to estimate and explain the motion of the vehicles. Consequently, appropriate rules must be established for adding new models and removing old ones as the video progresses. We make the assumption that whenever the number of centroids changes between frames, then either a new car has entered or an old car has left the video. For adding a new Kalman model, we must initialize the initial position, velocity, and error covariance of the Kalman model. For adding a new particle filter model, we must initialize the initial position, velocity, generate a set of particles around our initial state, and initialize their respective weights. However, before doing so, we must first determine which centroid is new to the video. This is accomplished by comparing the pool of filter prediction estimates from the previous frame and calculating the distance matrix between each set of predicted points and current observations. Whichever observation has the poorest fitness is assumed to be the new vehicle centroid. For our application, we interpolate that the centroid must have come from the nearest edge of frame and correspondingly initialize its previous position to be the frame edge and the velocity is constructed as a vector between these two points. For removing the model, we use a similar fitness criterion based on the distance between the prediction updates of the previous frame's pool of Bayesian filters and the current observations. Although there are a couple of situations where this model update process will fail, such as when two vehicles are simultaneously entering and leaving the scene, or shadow artifacts that pop up randomly within the middle of the frame, we can account for this by removing models who have poor fitness in the assignment phase of our vehicle tracking system. The initial noise covariances and error covariances are all assumed to be diagonal and have been manually tuned for achieving the best tracking performance.

4.5 Tracking through Assignment Problem

In order to track vehicles within the traffic video, we take the current vehicle location observations, and construct a euclidean distance map with respect to the previous frame's predicted vehicle locations based on the system dynamics of the pool of Bayesian models. If one can establish a correspondence between the current observations and the previous predictions, then one will have effectively tracked the vehicle over time. In order to do so, we must solve the assignment problem, which is a combinatorial optimization problem that seeks to minimize the cost of assigning predicted centroids to current observations. For every frame in the video, a one-to-one correspondence exists that uniquely minimizes the total euclidean distance cost

$$D = \sum_{i=1}^{N} d_{ij}$$

where d_{ij} is the cost of associating the ith predicted centroid with the jth current centroid for N vehicles. For our application, we use the Hungarian algorithm which solves the assignment problem in polynomial time[8].

5 Experimental Results

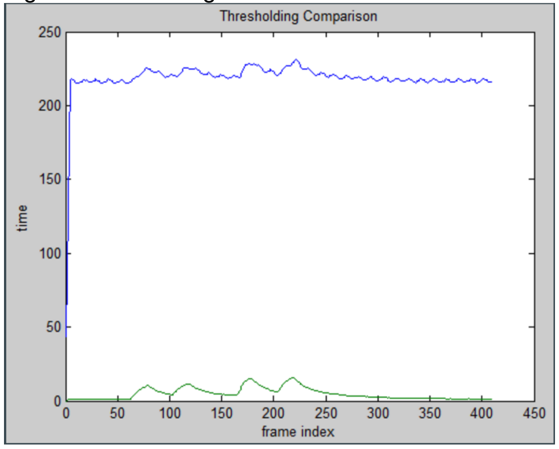
Within the following section, we describe computational runtime improvements for our traffic detection system and an in depth analysis of parameter tuning for the Kalman Filter and particle filter performance for the tracking component of the system.

5.1 Computational Improvements

For computational improvements to our system, we implemented a threshold method for conditionally skipping the adaptive process and a normal distribution calculation modification based on the Mahalanobis distance.

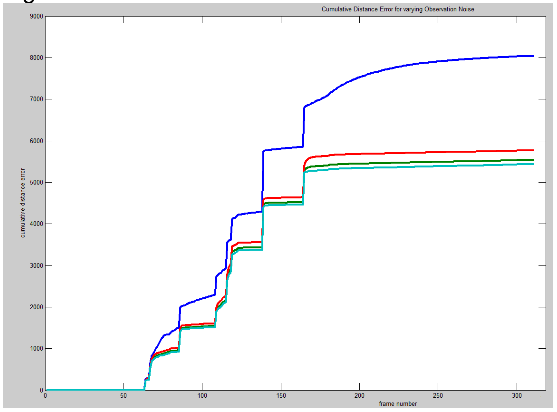
In figure 6, we show the standard algorithm procedure in blue, and the threshold modification in green. In particular, the threshold modification checks to see whether a pixel observation differs significantly from a reference observation 40 frames back. If the change is negligible, we can effectively skip the adaptive model update process. In doing so, our model only accumulates evidence in regions with high pixel observation variance corresponding to moving objects. Although this does save us processing time, we incur a cost of not accumulating evidence for stable background distributions as often which can lead to shadow artifacts as in figure 4 where the background is confused as a foreground object in the wake of a moving object. In order to reduce this effect, I used an additional heuristic comparison based on a reference frame from the beginning of the video to mask out this effect to the point where the connected component cleanup procedure can remove these artifacts usually. Although the threshold method does incur an accuracy hit, in general, it provides up to a 90 percent reduction in processing frames with peak loads coming from scenes with multiple foreground objects.

Fig. 6. Thresholding Method Performance



In addition, we incorporated the Mahalanobis Distance as shown in figure 7 instead of the actual normal distribution probability calculation in order to reduce the time needed for patching pixel observations to the mixture model distributions[10]. During high variance scene changes, the performance improvement ranged from around 30 to 40 percent improvement in processing time.

Fig. 7. Mahalanobis Distance Performance

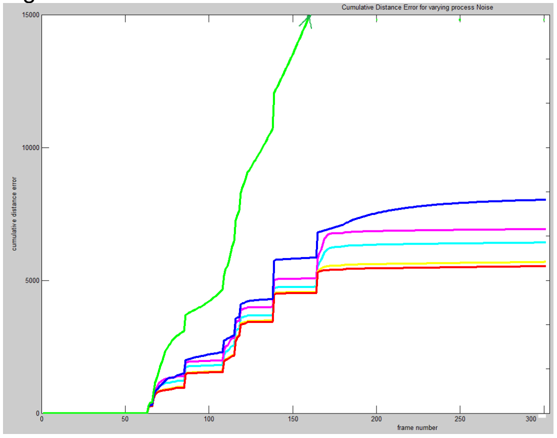


5.2 Kalman Filter Tuning Performance

For accuracy improvements for our system, we tuned the process noise, observation noise, and time step size in order to study the effects on minimizing the accumulated prediction error for the vehicle position over time.

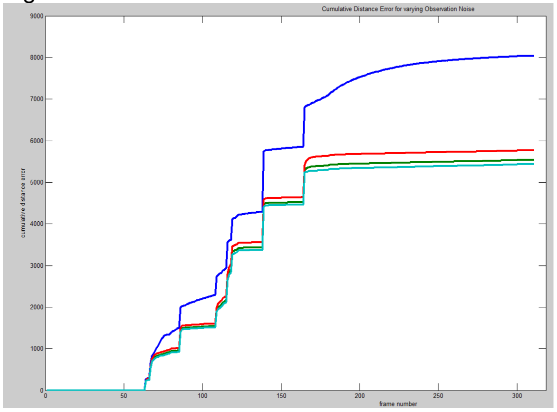
The process noise had the most profound effect on changing the dynamics of the accumulated prediction error. In particular, we found that by assuming no process noise, the Kalman Filter could not effectively explore the state space and thus resulted in a blowup of accumulated prediction error over time. This is because this effectively states that the system is deterministic in nature. By contrast, all of the Kalman models did slightly better than the basic no prediction model with higher process noise magnitudes corresponding to more accurate prediction estimates as shown in figure 8 below.

Fig. 8. Process Noise Variation



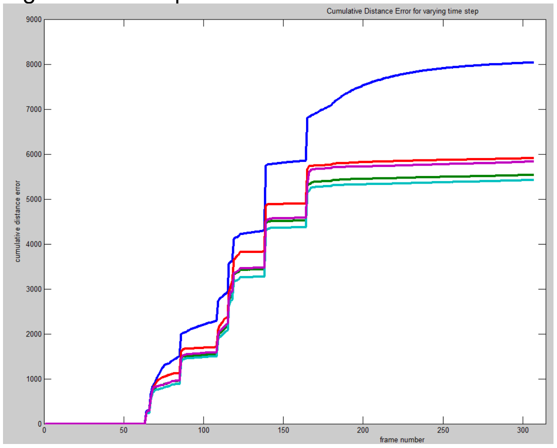
The observation noise variations all performed better in the Kalman model as compared to a basic no prediction setup. However, the observation noise had a relatively small effect with variation among the Kalman prediction approachees. What I found was that no process noise actually performed the best in regards to accumulated prediction error while the models became steadily worse with larger R values.

Fig. 9. Observation Noise Variation



The time step variations all performed better in the Kalman model as compared to a basic no prediction setup. However, the timestep magnitude had a relatively small effect with variation among the Kalman prediction approaches. What I found was that a 2 second timestep actually performed the best in regards to accumulated prediction error while the models became steadily worse in both directions from this local maximum. Perhaps with a more involved estimation technique such as the extended kalman filter, or particle filter, the timestep would have a more profound effect on the accumulated error.

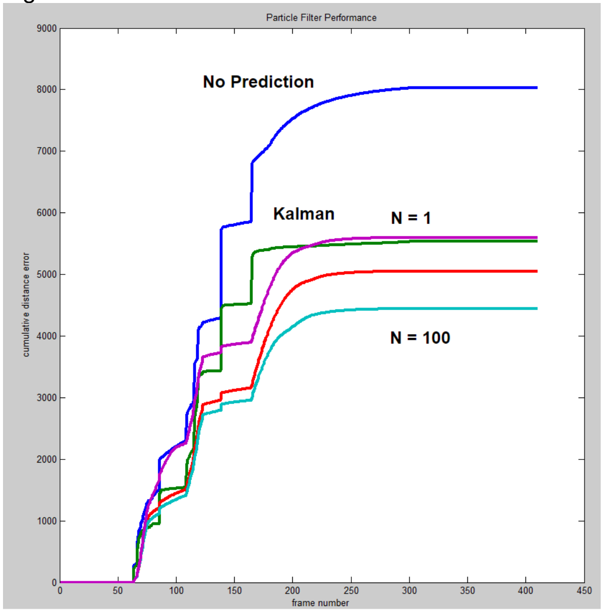
Fig. 10. Timestep Variation



5.3 Particle Filter Tuning Performance

For comparison, we also developed a particle filter solution in order to evaluate the performance benefits between the two recursive Bayesian estimators. In particular, we evaluate a system with zero prediction, the best Kalman filter, and a particle filter with varying numbers of particles from 1 to 100.

Fig. 11. Particle Number Variation



The particle filter seems to outperform both the Kalman filter and basic no prediction model. However, in spite of the obvious performance improvement, we suspect that a degeneracy problem must have arisen in the latter stages of the video due to a meaningful, yet underwhelming accuracy improvement compared to the Kalman filter and no prediction approach. There are a variety of explanations that may have arisen due to both the system model assumptions, such as the Q matrix, or more likely, a lack of sampling diversity. In particular, as the video progresses, only a select few particles may have any meaningful proscribed weights. In order to combat this, we could look perform a resampling procedure based on the effective degeneracy of the system N_{eff} defined by

Fig. 12. Degeneracy Measure

$$N_{eff} \hat{=} \frac{1}{\sum_{i=1}^{N} (w_k^i)^2}$$

6 Future work

Despite the progress made for this project, there are a variety of features that could be added in order to optimize the learning parameters in an unsupervised manner and to improve the robustness of the tracking application of the vehicle detection system. In particular, for our system, we had to manually tune the parameters such as learning rate, background portion, model noise, etc. However, for a tractable system, one would have to come up with a scheme in which the parameters could dynamically change based on some sort of scene analysis, or through a training session based upon subjective quality assessments from human test subjects[3]. For the particle filter implementation, we could tune the performance for process noise variation and explore an aggressive resampling procedure based on the supposed degeneracy of the system. In addition, the constant velocity model for our system was a poor fit overall for modeling the traffic stoplight intersection. As mentioned before, this was largely due in part to the forced vehicle acceleration due to traffic stoplight signals. In order to ameliorate this deficiency, we could come up with a means to switch the dynamics of the system seamlessly between a constant velocity and constant acceleration model, which would capture the dynamics of a traffic intersection very well. A variety of approaches to switching linear dynamical models (SLDM) have been taken, however, an especially promising approach comes from the use of hierarchical Dirichlet processes for hidden markov models. [11] Although this method could in principle discover between a countably infinite number of states, the hidden states for our system would amount to whether the vehicles are undergoing a constant velocity or a constant acceleration. In particular, the distribution of these models can be drawn from a Dirichlet process and solved by using a Rao-Blackwellized particle filter in order to switch between our constant velocity and acceleration models. Although this would incur a significantly increased computational complexity, the dynamics of the system would be very well captured and the accumulated error should in principle be greatly reduced.

7 Conclusions

We implemented a traffic detection system that incorporated both vehicle detection and tracking using Gaussian mixture models and recursive Bayesian estimator approaches. We found that the constant velocity model, although inaccurate, proved to be sufficient for tracking a small number of vehicles. In addition, we found that the mixture model was fairly robust in segmenting foreground objects from static background pixels, but proved to be very computationally intensive without some algorithm modifications based on threshold methods and distribution approximations. Finally, we evaluated the various performance benefits of a no-prediction, Kalman filter, and particle filter tracking system based upon various filter parameters annd found significant accuracy improvements. On the whole, our traffic detector system can serve as a foundation for more complicated traffic monitoring tasks such as those that include more accurate tracking, traffic congestion analytics, and automated accident detection.

References

[1] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on., Fort Collins, CO, 1999, pp. 252 Vol. 2. doi: 10.1109/CVPR.1999.784637

[2] Christof Ridder, Olaf Munkelt, and Harald Kirchner. “Adaptive Background Estimation and Foreground Detection using Kalman-Filtering,” Proceedings of International Conference on recent Advances in Mechatronics, ICRAM’95, UNESCO Chair on Mechatronics, 193-199, 1995.

[3] Klaus-Peter Karmann, Achim von Brandt, “Moving Object Recognition Using an Adaptive Background Memory”, in: V. Cappellini (ed.), “Time-varying Image Processing and Moving Object Recognition, 2”, pp. 297-307, Elsevier Publishers B.V., Amsterdam, The Netherlands, 1990.

[4] Nir Friedman and Stuart Russell. “Image segmentation in video sequences: A probabilistic approach,” In Proc. of the Thirteenth Conference on Uncertainty in Artificial Intelligence(UAI), Aug. 1-3, 1997.

[5] W.E.L. Grimson, Chris Stauffer, Raquel Romano, and Lily Lee. “Using adaptive tracking to classify and monitor activities in a site,” In Computer Vision and Pattern Recognition 1998(CVPR98), Santa Barbara, CA. June 1998.

[6] Li Ying-hong, Tian Hong-fang and Zhang Yan, ”An improved Gaussian mixture background model with real-time adjustment of learning rate,” 2010 International Conference on Information, Networking and Automation (ICINA), Kunming, 2010, pp. V1-512-V1-515. doi: 10.1109/ICINA.2010.5636758

[7] D. Bailey and C. Johnston, “Single pass connected components analysis,” in Image and Vision Computing New Zealand, 2008, pp. 282–287.

[8] F. Luetteke, X. Zhang and J. Franke, ”Implementation of the Hungarian Method for object tracking on a camera monitored transportation system,” Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on, Munich, Germany, 2012, pp. 1-6.

[9] M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp, ”A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking,” in IEEE Transactions on Signal Processing, vol. 50, no. 2, pp. 174-188, Feb 2002. doi: 10.1109/78.978374

[10] Liwei Wang, Yan Zhang and Jufu Feng, ”On the Euclidean distance of images,” in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 8, pp. 1334-1339, Aug. 2005. doi: 10.1109/TPAMI.2005.165

[11] C. Magnant, A. Giremus, E. Grivel, L. Ratton and B. Joseph, ”Dirichlet-process-mixture-based Bayesian nonparametric method for Markov switching process estimation,” Signal Processing Conference (EUSIPCO), 2015 23rd European, Nice, 2015, pp. 1969-1973. doi: 10.1109/EUSIPCO.2015.7362728.