Adam Nedorezov
EE 554
Final Project Results

Roadway Image Vanishing Point Detection/Tracking Using Monte Carlo Methods

The concept behind this project was derived from the Robot Localization problem solved using Monte Carlo methods in the paper by Dellaert. The difference here is that I am trying to localize the vanishing point in an image instead of a robots position in a building. The sonar measurements made by the robot are now replaced by the intersections between lines that estimate edges in the image. The control input is negative of the cameras movement from frame to frame since it is assumed that the camera is trying to move towards the vanishing point (although as this information was not available for the video sequences used, this was assumed to zero).

The following paper is broken up into three sections to better explain the results and the work done. The first section talks about the Monte Carlo Tracker (MCT); the principles behind it and an example implementation. The second section covers the two approaches explored to provide line estimates of edges in the image in order to get their intersection as the measurements for the MCT. The final section details the results of combining the two components together to create a MCT for a vanishing point in an image.

Contents

Building the Monte Carlo Tracker (MCT)

The investigation and implementation of a MCT was the main focus of this project however it was to be applied to a different type of localization problem than that of the paper by Dellaert. To begin with, a simpler problem, modeled on the robot localization problem, was considered in order to build, test and evaluate the MCT. I started by considering a robot moving in a square room of size 1x1. External sensor measurements of the robot's position were generated as a $\sim N(0, 0.0125)$ around the ground truth position. The control measurements were computed as the difference between ground truth positions between time steps plus some sensor noise modeled as $\sim N(0, 0.00125)$.

The MCT approach is a cyclical process that involves taking samples from the probability distribution describing the robots position, predicting how the distribution is changed by applying the known control step to each of the samples drawn and finally, weighting this newly predicted probability by the received measurements to obtain the a probability density function describing the robots position. Essentially this method assumes the samples are a representative of a possibly highly sophisticated probability density function (pdf) in order to approximate the result of the new pdf after transformations are applied to the original. Below are the equations describing the Monte Carlo Localization/Tracking process:

(1) Predictive PDF

$$p(x_k|Z^{k-1}) = \int p(x_k|x_{k-1}, u_{k-1})p(x_{k-1}|Z^{k-1})dx_{k-1}$$

a. $p(x_k|x_{k-1}, u_{k-1})$: is the motion model. It is the conditional probability of the new position given the old position and some control input u_{k-1}.
b. $p(x_{k-1}|Z^{k-1})$: is the Prior PDF. This is just the Posterior PDF of (2) from the previous time step. This is initialized to be uniform over the entire search space at the first time step.

(2) Posterior PDF

$$p(x_k|Z^k) = \frac{p(z_k|x_k)p(x_k|Z^{k-1})}{p(z_k|Z^{k-1})}$$

a. $p(z_k|x_k)$: is the measurement model. This is probability that the measurement comes from the robots estimated new location.
b. $p(x_k|Z^{k-1})$: the predictive PDf of (1).
c. $p(z_k|Z^{k-1})$: is the measurement likelihood that the new measurement is probable given the old set of measurements.

This coded MCT was tested against several different target trajectories. These included a vertical, a diagonal, a sinusoidal and a figure eight trajectory. Below are

screen shot of the full trajectory taken by the robot and the trajectory estimate of MCT. This is followed by an error plot that shows the error between the MCT position estimate and the ground truth position. The error produced by the tracker is typically below 0.02 units once the robot has been localized.

The code for the Monte Carlo Tracker is called "test_MonteCarloTracker.m".

Typical initialization of the MCT tracker

The MCT is initialized by uniformly sampling over the search region as shown below. Two hundred samples are used.

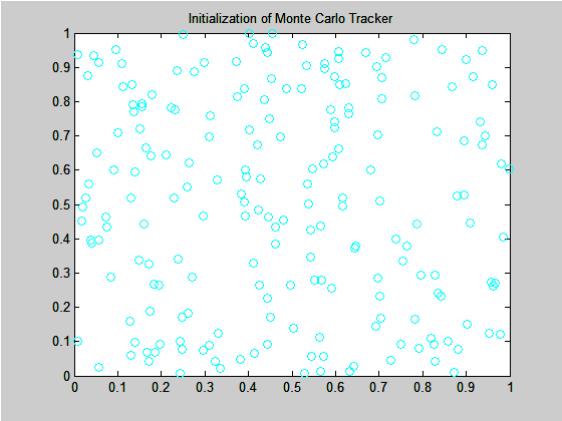

Figure 1: Typical Initialization of the Monte Carlo Tracker

Vertical Trajectory

The robot starts at the bottom of the search space [0.5 0] and traverses vertically to [0.5 1]. After about 60 time steps the MCT position estimate reaches a steady state.

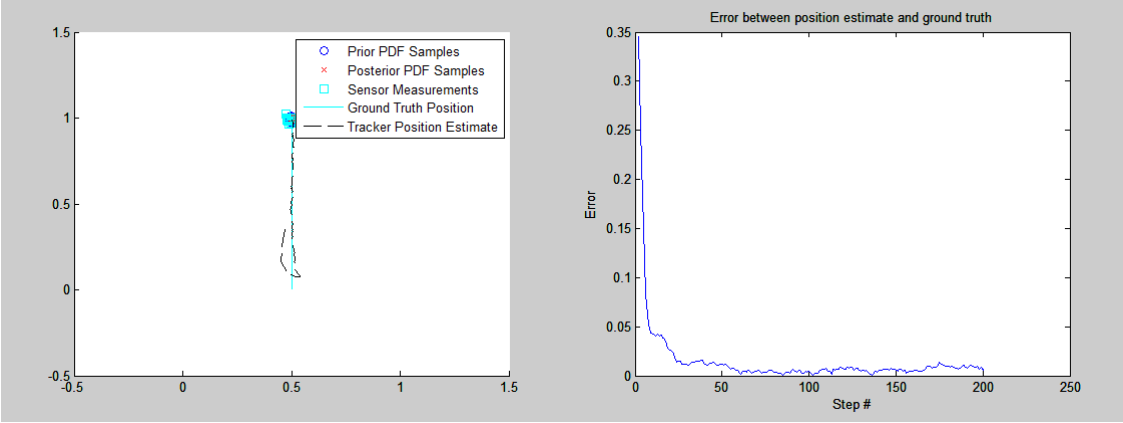

Figure 2: Results for robot with vertical trajectory

Diagonal Trajectory

The robot starts at the bottom of the search space [0 0] and traverses diagonally to [1 1]. After about 25 time steps the MCT position estimate reaches a steady state.

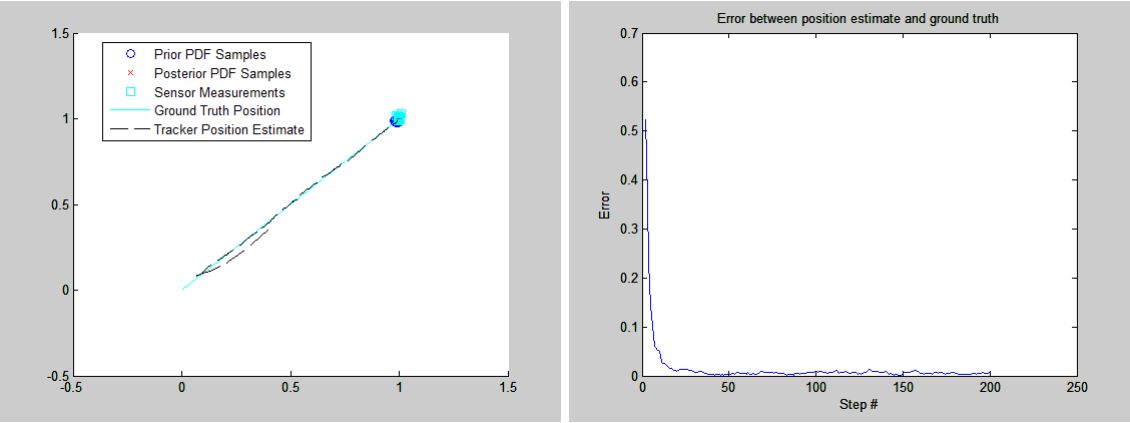

Figure 3: Results for robot with diagonal trajectory

Sinusoidal Trajectory

The robot starts at the bottom of the search space [0.5 0] and traverses sinusoidally to [0.5 1]. After about 50 time steps the MCT position estimate reaches a steady state.

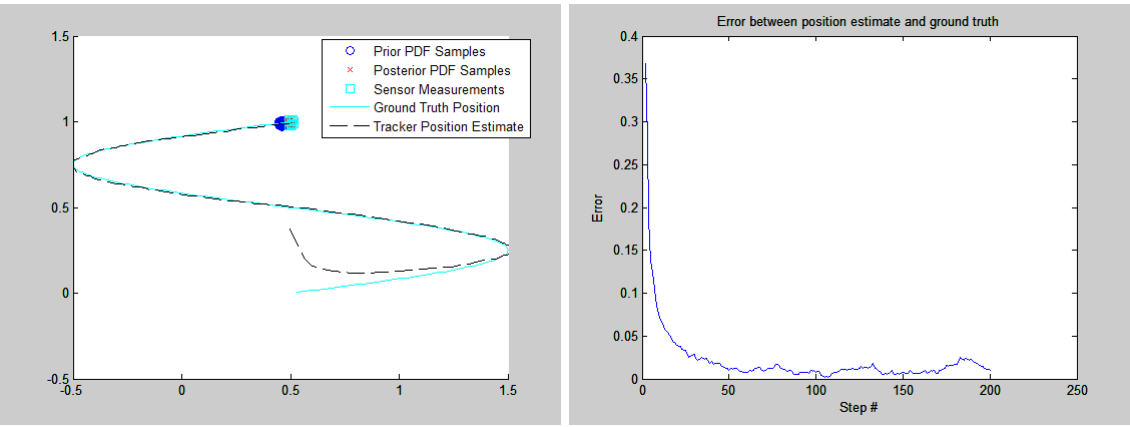

Figure 4: Results for robot with sinusoidal trajectory

Figure 8 Trajectory

The robot starts at the right of the search space [1 0.5] and follows a figure eight trajectory until it returns to the original starting point. After about 90 time steps the MCT position estimate reaches a steady state.

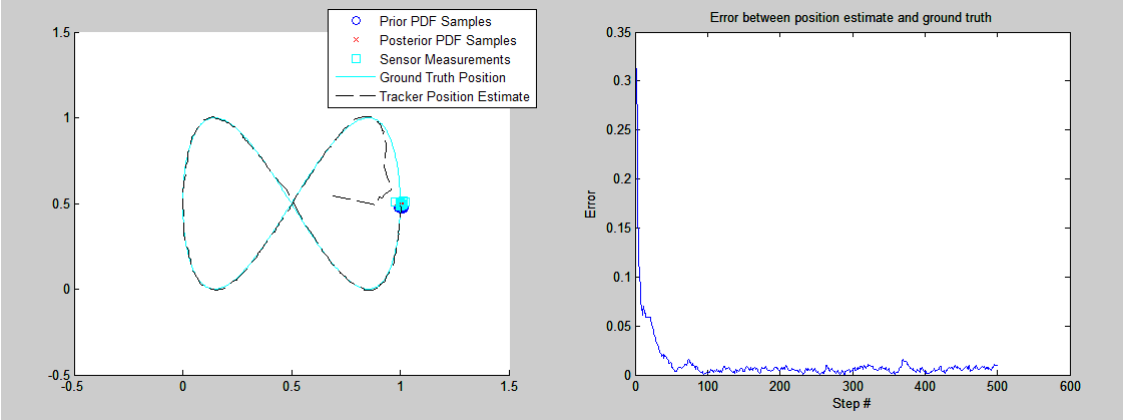

Figure 5: Results for robot with figure 8 trajectory at slow speed

It should be noted that when the robot was made to travel at faster speeds the trackers error began to grow.

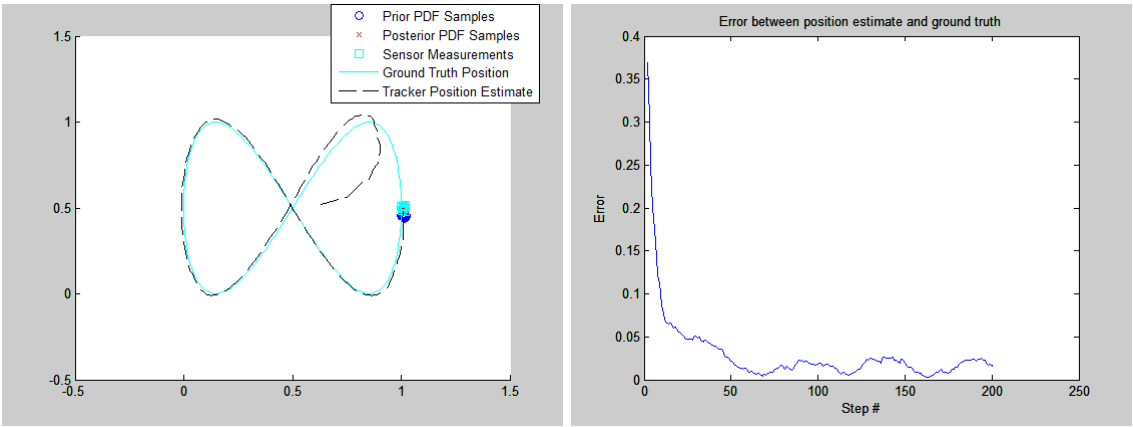

Figure 6: Results for robot with figure 8 trajectory at fast speed

Line Detection in an Image

In order to create the measurements (vanishing point estimates) for the MCT, two methods were explored. The first method was the Manhattan World assumption (Coughlan and Yuille) which maps edges detected in the image to one of the Manhattan xyz axes. The second method was the Hough transform which finds lines that best fits the detected edges in the image. These methods are outlined below and sample results of their implementation are shown.

Manhattan World (MW) Assumption

As already stated this approach tries to map edges in the image to one of three typical xyz axes. In fact these mappings are described by a set of rotation angles which describe the rotation of the camera to the Manhattan structures in the image. The labeling of image edges as an xyz axis is performed by searching over a set of angles and comparing the likelihoods produced by the Bayesian Inference Model developed by Coughlan and Yuille. This Bayesian model is described here in more detail.

(1) $p(Z|X) = \prod_{k=1}^{K} P_{x_k}(Z|X)$

a. x_k is the image coordinate
b. Z is the image
c. X is the given set of angle parameters. (These are the three orientation angles that explain the camera orientation to the Manhattan xyz coordinates. These are the parameters that are searched over)

(2) $p_{x_k}(Z|X) = \sum_{i=1}^{5} p_{x_k}(Z|m_i, X)P(m_i)$

a. m_1, m_2, m_3 are the x, y, z axis respectively
b. m_4 is a random edge direction
c. m_5 is not an edge
d. $P(m_i) = \{0.02, 0.02, 0.02, 0.04, 0.9\}$

(3) $$p(Z|m_i, X) = \begin{cases} p_{x_k}^{on}(Z)p_{x_k}^{ang}(Z|\theta_{x_k}^{i}(X), & i = 1, 2, 3 \\ p_{x_k}^{on}(Z)\frac{1}{2*\pi}, & i = 4 \\ p_{x_k}^{off}(Z)\frac{1}{2*\pi}, & i = 5 \end{cases}$$

a. $p_{x_k}^{on}$ is the probability that the pixel is an edge pixel. In the code I approximate this based on the magnitude of pixels gradient.
b. $p_{x_k}^{off}$ is the probability that the pixel in *NOT* an edge pixel
c. $p_{x_k}^{ang}(Z|\theta_{x_k}^{i}(X)$ is the probability that the angle of the gradient matches the actual Manhattan axes orientation i. In the paper this was implemented as a box function where if the angle difference between the two angles was less than 6° then the probability was high. In the code I use the take the exponential of the difference.

I attempted to implement the MW model (see "test_manhattan") however I believe I have some misunderstanding that prevented me from being able to get my

desired output which was a line equations whose intersections could be used to estimate the vanishing point. After a week of trying to debug the MW code I came across the Hough transform which is described in the next section.

Anyway, the results of my MW model implementation does seem to classify the pixels as either an X, Y, Z edge, an un-aligned edge or a non-edge as demonstrated in figure 8. And while the axis orientation depicted in figure 9 does show the z axis (blue) pointing towards the perceived vanishing point the result was not consistent among other images such as that of figure 10.

Figure 7: Original Image

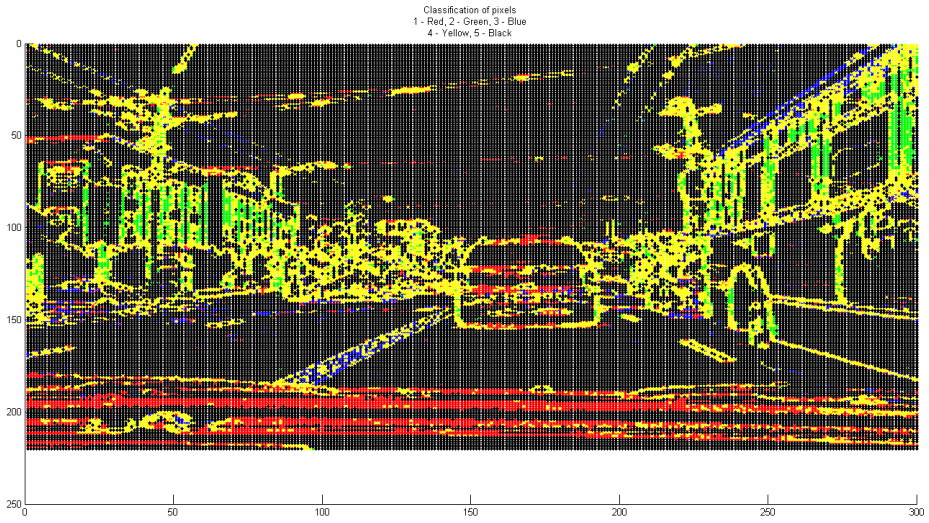

Figure 8: Edge classification for each pixel. The red dots are the X axis, the green dots are the Y axis, the blue dots are the Z axis, the yellow dots are un-aligned edges and the black dots are classified as non-edge pixels.

Figure 9: Displays a regional estimate of where the vanishing points are. The blue lines (z axis) are aligned with the perceived vanishing point in the center of the image. However these appear to be traveling away from the vanishing point and not towards it. Fortunately this does not matter if we are trying to find the intersection of these lines.

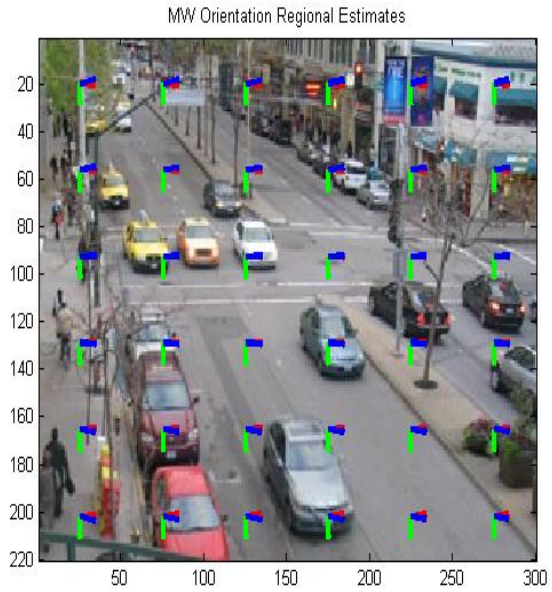

Figure 10: Example of an image where the overlaid axis orientations do not appear to point towards the vanishing points.

The Hough Transform

After struggling with the MW approach I came across the Hough transform built into Matlab. The Hough transform is a voting procedure that estimates the lines that best overlay the edges detected in the image. The added benefits of switching to this method was that it is much less computationally intensive which reduced the time it takes when combined into the full vanishing point (VP) tracker. The other point that can be made is that the results provided by the functions *hough*, *houghpeaks*, and *houghlines* produce line outputs more closely related to the originally intended line search algorithm where line equations would be found that relate to edges in the images.

The Hough transform is a parameter search method over θ and ρ of the equation for a line $[xsin(\theta) + y\cos(\theta) = \rho]$. Each edge pixel (x, y) produces a set of (θ, ρ). The resulting set of parameters are then overlaid for each pixel to create an Accumulator matrix which counts the number of pixels that produce the same set of parameters (θ, ρ). The peak of this Accumulator matrix corresponds to the parameters that produce a line which will overlay the most edge pixels.

In order to implement the Hough transform I modified the input image in the example provided for the *houghlines* function. Below is an example of the results produced by the Hough transform. The green lines are the line estimates produced from the edge information, the yellow and red crosses are the start and stop points of the line estimate and the cyan line is the longest line segment found. After some modifications to the parameters used and the way the line segments are drawn it was easy to compute the intersection points of the lines produced. These intersections were then fed into the MCT.

Figure 11: Line estimation for edges using the Hough Transform.

Putting It All Together

With each of the components tested they were finally combined into a single process in order to create the vanishing point (VP) tracker. To test this code two video sequences were used, the first was a set of images that were broken out of the video form the *CamSeq01* dataset. These images depict a car traveling down a road in Cambridge. The other data set is that of a car driving down a highway and switching lanes multiple times.

For testing, 1000 samples and a pixel variance of 2 were used. In order to get some measure of performance the ground truth position of the vanishing point was quickly estimated by me (perhaps poorly in some cases) by looping through the images once and using the *ginput* function to mark what I perceived was the vanishing point. While these ground truth markings are by no means perfect they do provide a quick way to compare the trackers performance.

The error between the MCT estimate and the ground truth location of the vanishing point was computed by first dividing the x and y error distance by the images x and y dimensions respectively. This was done to normalize the error measurements to the size of the image. Then the sum of squared error was computed to obtain an error percentage.

CamSeq01 Data set

The CamSeq01 Data set is comprised of 101 frames with an image size of [720 960]. Pictured first is the initial frame where the MCT is initialized to uniformly distribute position estimates (yellow dots) all over the image. The lines produces by the Hough transform are magenta, their intersections are the cyan crosses and the green square is the current mean of the position estimates. The following images are snapshots of the MCT at work for the city scene.

Figure 12: Initial frame of CamSeq01 dataset.

Figure 13: Position estimates moving towards vanishing point.

Figure 14: The samples have converged to a vanishing point estimate.

Figure 15: While making a switch in the lanes, it seems that the estimate is dragged to the right by the cars movement. Perhaps if the control input was included (using knowledge about the cars maneuvering, which is unknown for both data sets) this would help move the estimate towards cluster of line intersections. Trials with estimates of this control input were done and did seem to better track the cluster of line intersections.

Figure 16: Final frame of the CamSeq01 data set.

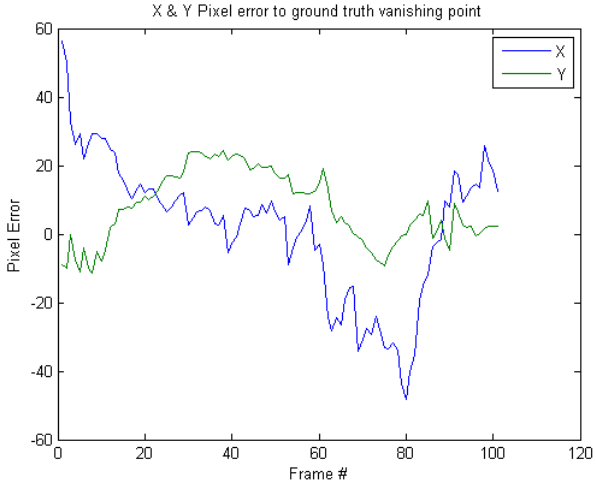

Figure 17: X & Y pixel error plot. The spike occurring around 80 is during the cars maneuver from the left to the right lane. If the control inputs were known then they could be used to predict this movement and reduce the spike in error.

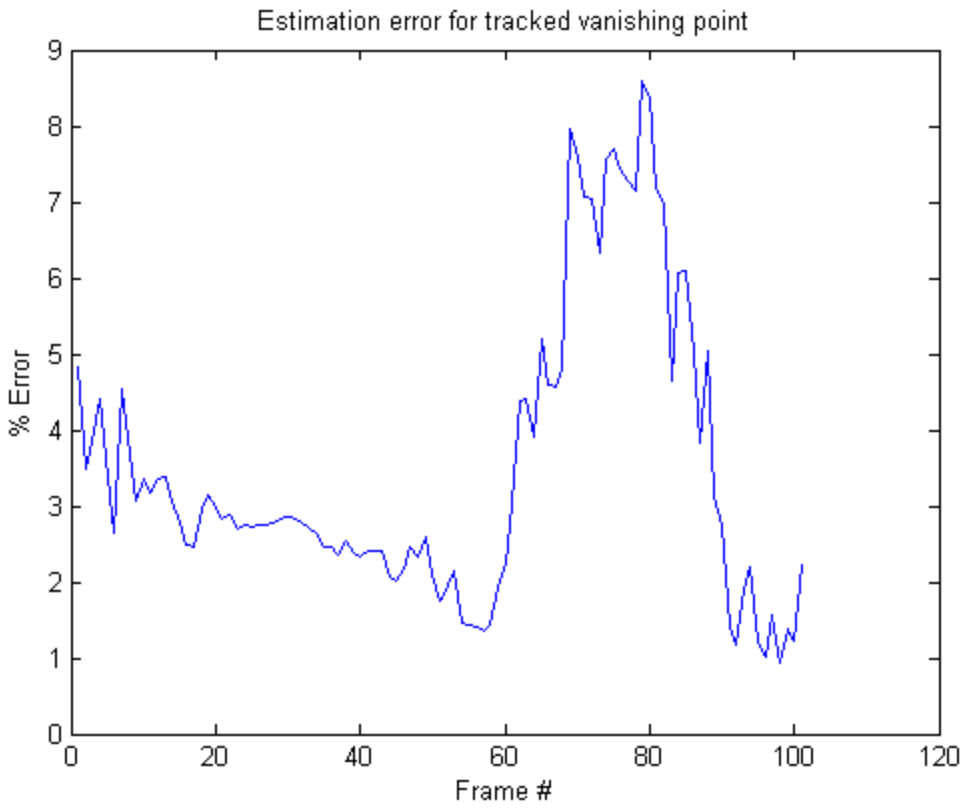

Figure 18: % Error plot. Again, the error spike during frames 70-90 is caused by the car changing lanes.

Highway Diving Video Set

This data set is a video of a car driving down a highway making frequent lane changes. All the markings, colors, and conditions are the same as was for the *CamSeq01* data set. There are 337 frames with an image size of [240 360].

Pictured first is the initial frame where the MCT is initialized to uniformly distribute position estimates (yellow dots) all over the image. Again, the magenta lines are produced by the Hough transform, their intersections are the cyan crosses and the green square is the current mean of the position estimates. The following images are snapshots of the MCT at work for the highway scene.

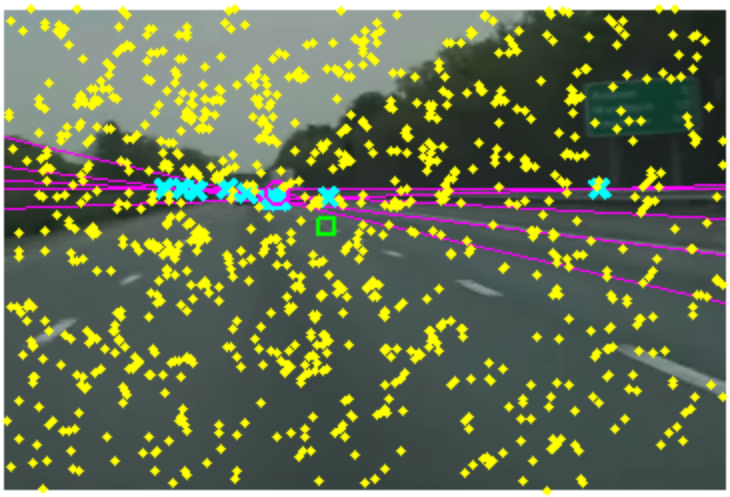

Figure 19: Initial frame of the highway driving video sequence.

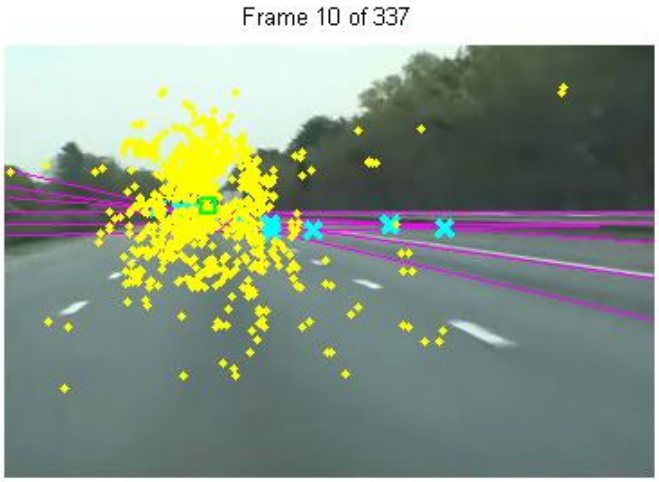

Figure 20: Position estimates moving towards vanishing point.

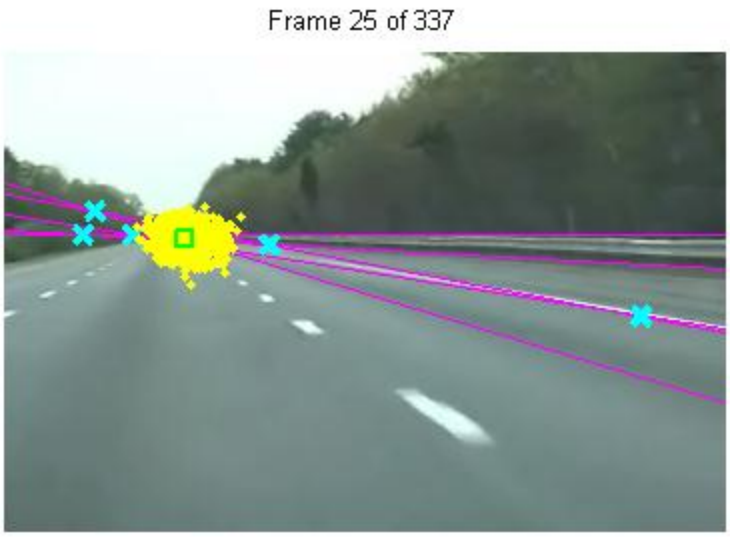

Figure 21: The samples have converged to a vanishing point estimate.

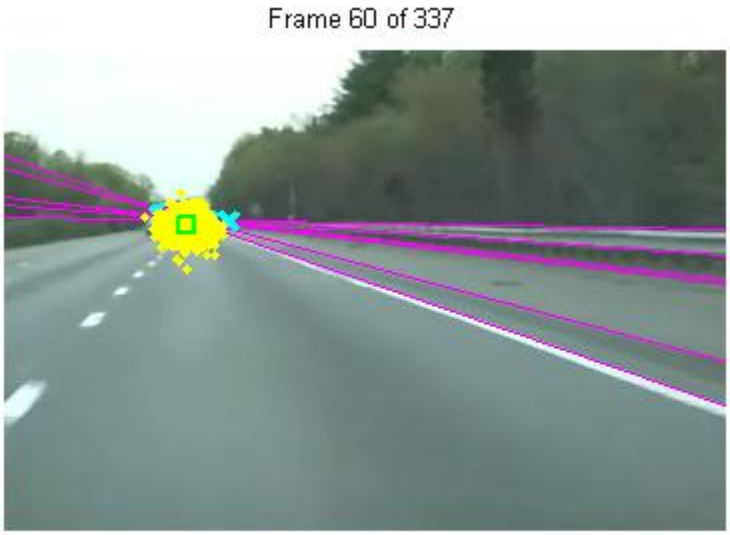

Figure 22: Car moved into rightmost lane

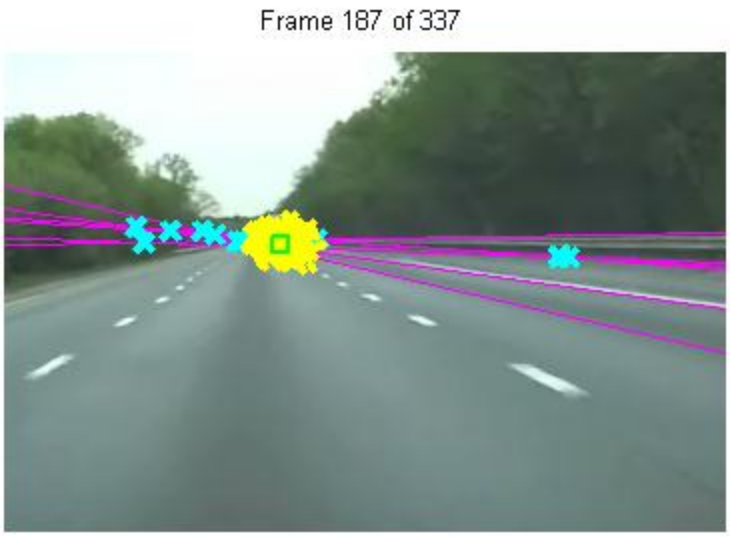

Figure 23: Car moved back into center lane.

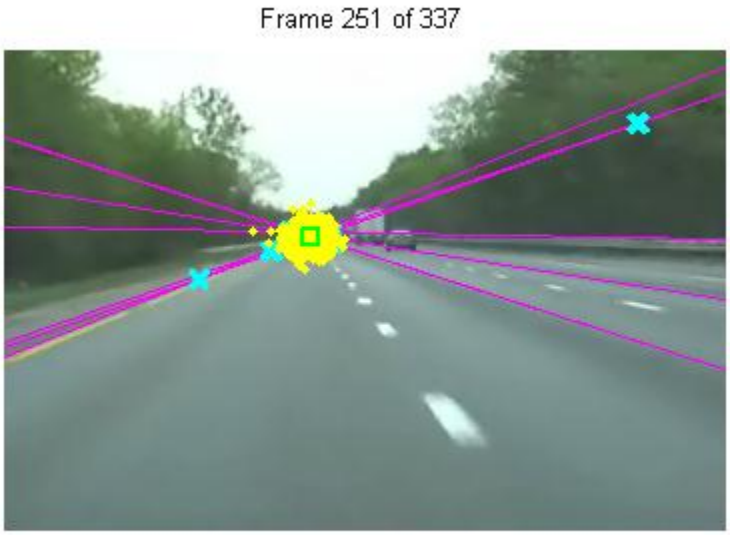

Figure 24: Car moved into left most lane.

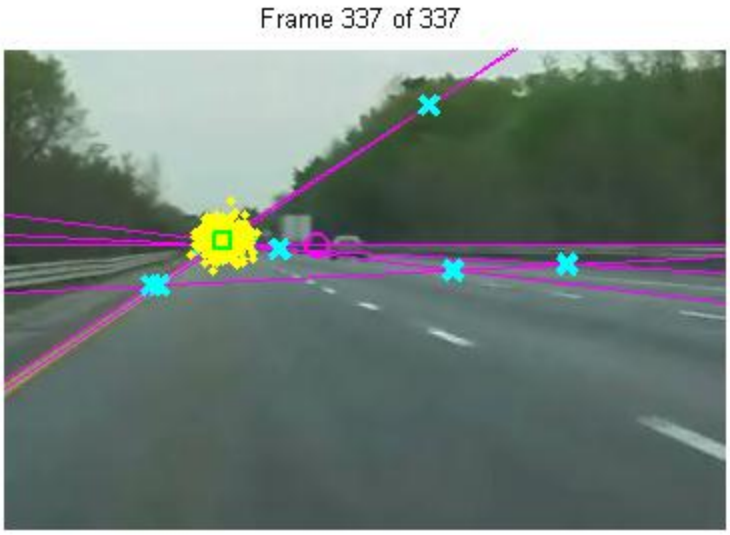

Figure 25: Final frame of the highway video sequence.

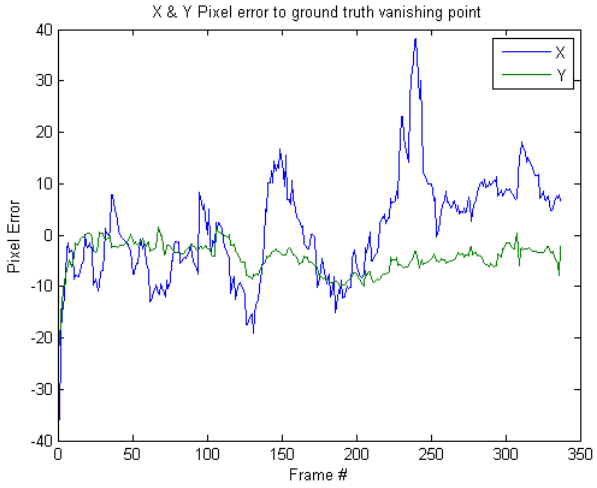

Figure 26: X & Y pixel error plot. Several spikes in the error occur in the x direction. Each of their appearance seems to correlate with the car changing lanes. Again perhaps if the movement of the car was known, this could have been fed into the MCT as the control input and reduced the spikes seen in the error plot above.

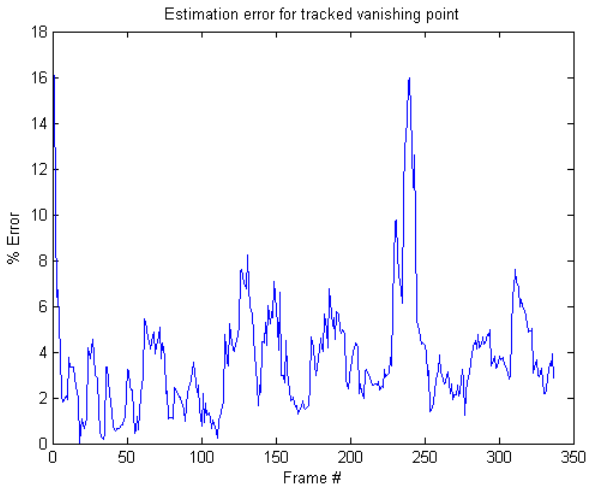

Figure 27: % Error plot. Again, the spikes in error are due to the car changing lanes.

Summary

The final project performed here incorporated topics from the Monte Carlo methods and graphical models covered in class. The localization and tracking of a vanishing point in the images uses Monte Carlo methods to sample the underlying distributions in order to estimate the vanishing point position likelihood. (This concept was based off of the robot localization problem by Dellaert). Intersections of lines were used to create the samples that would update the distributions being estimated by the MCT.

In order to obtain the line intersections, the lines were approximated using two methods. The first method was the Manhattan world assumption which claims that edges in the image can be mapped to one of the real world axis x, y or z. This method involves a Bayesian Inference modeling for which the argmax would provide the classification of the labels. The Bayesian Inference model boiled down to a conditional probability which was broken up into smaller and more manageable conditional probabilities, much like the methods discussed during the graphical models lectures. However due to difficulties in understanding the frame rotations and error in implementing it, the MW model was not ready to be used in the full localization/tracking algorithm. (After fixing what might have been an indexing issue and a misunderstanding about the likelihood comparisons, the results still did not produce the expected output consistently).

The other method of line estimation of image edges is the Hough transform which is a voting procedure where image edges produce a set of possible parameters for a line. The set of parameters that are used the most, determined by the Accumulator matrix, from each edge's parameter set, relates to a line that passes along the most edge pixels. Extracting these parameters, lines are produces whose intersections provide the MCT with measurements to update its estimate of the vanishing point.

The performance of the MCT on the image sequences was reasonable. For the CamSeq01 data set the pixel error was under 9% and for the highway sequence the error was under 17%. These numbers of course are based on the quickly generated set of ground truth locations of the vanishing point which may or may not be accurate. In addition the control input for the predictive phase of the MCT was set to zero for both data sets. Attempts were made at estimating these control inputs and results did seem to improve, however, these were poorly approximated and resulted in additional errors if the an unrealistic set of controls were produced.

Overall it seems that the MCT can be adapted to localize and track a vanishing point in an image. While there are certainly improvements and enhancements to be made as future work, the initial results look promising.

References

CamSeq01 Dataset, Cambridge Labeled Objects in Video. 8 October 2007. April 2012.

Coughlan, J.M. and A. L. Yuille. "Manhattan World: Orientation and Outlier Detection by Bayesian Inference." *Neural Computation* 15 (2003): 1063-1088.

Dellaert, F. "Monte Carlo localization for mobile robots." *IEEE International Conference on Robotics and Automation*. 1999. 1322-1328.

Deutscher, J., M. Isard and J. Maccormick. "Automatic camera calibration from a single Manhattan image." *ECCV*. Copenhagen, Denmark, 2002. 175-205.