Real time lane tracking using particle filtering in Hough space

Bashar Mengana
TSCRM
880126-0939
Email: mengana@kth.se

Martin Larsson
TSCRM
890525-0034
Email: martinl6@kth.se

***Abstract*—The urge for improved tracking systems is justified in a society where we observe an increasing demand for traffic safety systems. The presented tracking device in this report is decomposed into a lane detector and a lane tracking component. The tracking component consists of a Particle filter as an estimator of the posteriori belief, and lane detection is achieved with image processing operations such as Canny edge detection, Morphological operations and the standard 2D Hough transform.**

Unlike similar reports in the field of lane detection, we focus on point tracking in the Hough space as opposed to line tracking in the spatial domain.

The report offers a way to detect outliers and a way to associate particles to their corresponding clusters as well as suggest light K-means clustering as a way of natural grouping when extracting statistical properties from particle clusters.

***Index Terms*—Lane detection, Lane tracking, Particle filter, Hough Transform, K-means clustering, Outlier detection, Stratified resampling.**

I. Introduction

Society's interest in general lies in the wellbeing of its citizens. Vision based street lane detection is an important factor for automatic driving systems, with which we can reduce the risk of car accidents. The urge for improved tracking systems is justified in a society where we observe an increasing demand for traffic safety systems.

The ability to analyze the camera image of a road in a effective and systematic way is ultimately one of the main appealing difficulties to solve in order to eventually introduce autonomous driving.

The process of autonomous driving can simply be decomposed into two prerequisites, that is, the detection of lanes and the tracking of lanes. Lane detection entails analyzing a single image fed by a camera sensor and to determine the lane stripes. Lane tracking is the process of applying either a non-parametric filter or a Gaussian filter to make use of and capture the temporal knowledge in the stream of images. Tracking with a particle filter in a sense injects robustness in the tracking system - note that we are free to just use a standard Hough transform for tracking. Let us consider the scenario where we omit the filter. Even though our reduced tracking device will be able to track the lane markings, we will not able to keep track of the lane markings as soon as another vehicle obstructs the view of the camera sensor. Thorough analysis show that the particle filter injects the kind of robustness to the tracking device that makes the detection process much more tolerant to occlusion, much more tolerant to noise and certainly more practical than just a standard Hough transform in the sense that it instills a more forgiving nature.

Monte Carlo Methods or Particle Filters allow to approximate arbitrary multimodal probability distributions. In other words, if we are using a non-parametric filter we are able to relax some of the assumptions needed for a Gaussian filter. This is beneficial for research purposes. Consider the case where one needs to quickly draft an estimation filter for some tracking scenario. To begin with, a Gaussian filter like the EKF works only with Gaussian beliefs, so we need to examine if this is the case. A non-parametric filter works out of the box in a sense - we do not need to make sure the posteriori is Gaussian or even linear. Most of the real world scenarios are not linear or does not even need to possess an analytical form. For this reason, we chose to work with a Particle filter and not the EKF.

A number of research works solves the lane detection/tracking problem efficiently, especially for highway like situations [1]–[4]. However, we are going to perform tracking in the Hough space and thus hope to contribute to the field. Section I is a short introduction of lane tracking and discusses related work. Section II explains the main theory regarding the digital image processing needed for lane detection, while Section III explains the most important aspects of lane tracking, such as particle deprivation, the noise model, clustering methods, particle association, the resampling scheme and outlier detection. Sections IV and V explains our experiment and presents the results. Sections VI, VII and VIII concludes the report with a discussion, notes about future work and a concise conclusion.

A. Related work

Unlike the work regarding lane tracking with the statistical Hough transform in [1], we track particles in the standard Hough space, also know as the parameter space, and not in the spatial domain. We provide an intuitive, approach by tracking line parametrization. Paper [1] uses inverse perspective mapping to free the image from perspective effects and to render the lines parallel. However, unlike the work of [4], which does lane detection in urban environments, we mainly focus on lane detection on motor highways. That is mainly due to the non existence of unique models, poor quality of lane

Lane detection
1. Image frame is converted to grayscale.
2. Canny edge detector is applied.
3. Morphological operations are applied.
4. Hough transform is used to extract the line parameters.

Figure 1. Sketch of the image preprocessing.

markings due to wear, occlusions due to the presence of traffic and complex road geometry in urban environments. While we only parametrize straight lines in the Hough space, paper [4] gives an overview of how one could generalize this to splines, apt for urban tracking. Our paper is similar to the work in [2], because we propose a guideline for how one could utilize real time tracking in MATLAB . However, paper [2] gives an overview of how one could handle the image preprocessing in a more refined way. The work in [2] relates to our work, but this thesis differs from our report because it uses a Gaussian filter, unlike our use of a non-parametric filter.

II. LANE DETECTION

The process of lane detection in this system is summarized in Figure 1.

A. Canny edge detection

The main point with the *Canny edge detector* is to create 1 pixel thick edge lines that also are continuous. Compared to other lane detecting algorithms, the Canny edge detector is superior [5]. Canny edge produce a binary image, apt for the Hough transform.

The camera sensor frame preprocessed by the Canny edge detector is initially cropped to half the height in order to cut out the sky from the original frame. In the test data we used (see section IV), some artifacts (edges) of the vehicle were detected by the Canny edge detector and remained in the edge image. In order to remove these artifacts and general image noise we need to apply Morphological operations.

B. Morphological operations

In the current implementation, several morphological operations are applied to the binary edge image just before the Hough transform is used in order to reduce noise and to reduce the amount of vehicle artifacts in the edge image. They are applied in the following order: 3 dilations in order to fill the rectangles that the edges of the line markings make up (see second image in Figure 5), followed by 4 erosions and an "infinite" amount of morphological thinning in order to restore to 1 pixel thick edges and to remove both noise and the discussed artifacts. The resulting image can be seen in the third image in Figure 5.

C. The Hough transform

There are two possible approaches to the line detection problem when using a particle filter: estimating actual pixels in the image that make up the lines in the spatial domain, or estimating the parameters of the lanes in the *Hough space*.

The *Hough transform* is a way of identifying lines in a binary image. The idea is to sample all the lines passing through a particular pixel, extract the parameters of these lines (ρ, θ), and create an "accumulator", or "Hough space", to keep track of the lines encountered. Lines that have been encountered many times produce peaks in the Hough space, and by extracting the corresponding parameters from these peaks, the lane stripes in the lane scene image can be reconstructed. More information about this procedure can be found in [5].

Several other lane detectors use Hough space to find the parameters of the lines [1]–[3]. It is possible to use the *statistical Hough transform* instead, which produces continuous estimates of the line parameters [1].

The regular 2D Hough transform has recently been proven to be a statistically robust estimator for finding lines [6]. The 2D Hough transform has however one main weakness: the probability density function $\hat{p}_{\rho\theta}(\rho, \theta)$ of the parameters (ρ, θ) in the 2D Hough space is estimated using a discrete two dimensional histogram. Therefore the trade off in between the number of bins in the histogram and the number of available observations is crucial. Too many bins for too few observations would lead to a sparse representation of the density. Too few bins would also reduce the resolution in the Hough space and therefore limit the precision of the estimates.

The Statistical Hough transform (SHT), unlike the standard 2D Hough transform that requires binary edge images, works on intensity images and uses kernels to describe the distribution of the Hough variables (ρ, θ) [1]. Since the SHT works with intensity images, we do not need to apply canny edge nor do we need morphological operations to preprocess the image fed to the estimator. Edge segmentation is a sensitive operation, especially in noisy images where some edges may be completely lost. The SHT replace the mentioned discrete 2D histogram of the regular 2D Hough Transform, by a smoother continuous kernel estimate. Thus, the discussed crucial difficulty of carefully choosing the resolution (number of bins) is no longer an issue with SHT.

This report however, uses the regular Hough transform, since we believe the concept of lane detection with the regular Hough transform is both clearer, more intuitive and an apt limitation of the scope of the project.

III. LANE TRACKING

The list in Figure 2 summarize the main loop in the lane detection and tracking algorithm in order to give an overview of the process. The list can mainly be decomposed in a detection component and a a tracking component.

The Particle filter steps are outlined in Figure 3 in order to give a overview of the process. The steps outlined in the list are explained throughout the report.

A. Tracking in Hough space vs. spatial domain

Viewing an image in domains such as 2D Hough space enables the identification of features that may not be as easily detected in the spatial domain. The 2D Hough space should be though of as a parameter space, thus any spatial feature

Lane detection and tracking loop
1. Image frame is acquired from data set or camera
2. Detect lane: see list in Figure 1
3. Track lane: see list in Figure 3
4. Extract statistical properties from each cluster with the aid of light K-Means clustering
5. Plot images as a video stream

Figure 2. Sketch of the lane detector process.

Particle filtering
1. Prediction
2. Associate particles with their corresponding clusters
3. Identify outliers
4. Assign a normalized weight Ψ to each particle
5. Cluster particle set with light K-means clustering
6. Stratified resampling

Figure 3. Sketch of the particle filtering process.

that is decomposable into a parametrization can be mapped to a point in the 2D Hough space. Intuitively, tracking of a straight line compared to tracking of just a single point should prove to need more particles since the elongatedness of a straight line is certainly more spacious compared to just a single point. In theory, a single particle (hypothesis as to where the point is) is sufficient to describe a point in the state space, however, a single point is not enough to represent a 1 pixel thick line. In addition, another valid reason for tracking points in the 2D Hough space and not lines in the spatial domain lies in the representation nature. While we could set every particle to represents an hypothesis of the position and shape configuration (pose, width, height, elongatedness) of the target line our state vector will indeed grow and eventually the curse of dimensionality will hit our tracking problem hard. However, tracking of a point in 2D Hough space requires only two parameters (ρ, θ).

Although the Hough transform has been heavily mentioned in this report which is mainly about tracking and estimation of a lane, the aim of the work reported here is not to use the Hough transform as a tool for tracking. It is rather for tracking features through their parameters as represented by the 2D Hough accumulator.

B. Multiple object tracking

On a typical road, there are at least two lane markings of significant importance. Depending on the situation it may be preferable to keep an estimate of the entire road, or just the current inhabited lane. In either case, it is required to track several different lines. This puts some demand on the filter, and can be addressed in several ways (assuming two lines are to be tracked):

- The state can be extended to track pairs of lines, and thus become 4-dimensional.
- Several parallel filters can be run, one for each line.
- Several clusters of particles can be maintained, each cluster tracking one line.

Extending the state will introduce much greater computational demands, and is significantly more difficult to model. Since the lane boundaries are not independent, a more significant analysis of the state covariance is necessary. For example: the lanes often have a fixed width, and the lines are generally parallel.

Using multiple parallel Particle filters requires that the measurements are associated to the correct filter. A method must be created to identify the "left line" and send it to the "left filter" etc. The Hough transform sorts the detected peaks in order of collector magnitude, and this value is usually not constant through time for a given line. In addition, using n parallel filters uses n times the computational power.

Keeping a small state space but with multiple "true" hypothesis (one cluster of particles for each line on the lane) puts other demands on the setup. The greatest and most critical being, as always, particle deprivation. In this scenario the risk of all particles converging to only a single group is substantial, and critical. The problem of associating detected lines to the correct cluster is not necessarily more complex than when using several filters, except that the computational load is kept low.

C. The motion model

The motion model is used along with the diffusion of the particles in the prediction step.

$$\hat{x}_k^- = \underbrace{\hat{x}_{k-1} + u_k}_{\text{applying motion}} + \underbrace{\mathcal{N}(0, \Sigma_R)}_{\text{diffusion}} \tag{1}$$

In general, when working with for instance robot tracking the odometry information (control signal) u_k is calculated with the aid of sensors like wheel encoders. However, in image tracking, we do not really have access to such odometry information, and we need to make some assumptions on the target we are tracking. In addition to this, lane tracking does not really have a control signal in the same sense. A way to think of the odometry information is to consider the camera movement (i.e. the rotation of the camera).

In contrast to robot tracking, the stripes on the lane will not move according to a control signal in the same sense, however, it is possible to model the control signal as the movement of the camera sensor. We assume that

$$u_t = 0 \tag{2}$$

under the assumption that the camera sensor does not rotate around its axis that is pointing to the sky (in other words, no rotation of the camera).

The particles will only undergo diffusion during the prediction step

$$\hat{x}_k^- = \hat{x}_{k-1} + \underbrace{\mathcal{N}(0, \Sigma_R)}_{\text{diffusion}} \tag{3}$$

and since we have simplified our motion model (we have introduced more uncertainty by assuming that $u_t = 0$, exactly) it makes sense to increase our state noise covariance matrix Σ_R. A larger Σ_R will increase the state discovery and this is important since this basically means that we are able to find the true posteriori belief eventually even if there initially is no particles at that state.

D. The observation model

The observation model is a function of the innovation term ν:

$$\nu = z_k - \hat{z_k} \tag{4}$$

where k is the time step, z_k is the measurements (in our case z_k contains the N line parameters $\{(\rho_1, \theta_1), (\rho_2, \theta_2) \dots (\rho_N, \theta_N)\}$ that are extracted from the preprocessed image frame that have been captured with the camera sensor). The predicted measurement for each particle is denoted $\hat{z_k}$ and is directly available for each particle (each particle estimates and contains a set of line parameters (ρ_i, θ_i) and a weight w_i).

The likelihood $p(z_k \mid x_k)$ for the observation model is given by a Gaussian:

$$p(z_k \mid x_k) = \frac{1}{2\pi \left|\Sigma_Q\right|^{1/2}} \exp\left\{-\frac{1}{2}\nu^T \Sigma_Q^{-1} \nu\right\} \tag{5}$$

where x_k is a particle.

When a particle *predicts* a $\hat{z_k}$ that gives rise to a large innovation term (large difference between prediction and actual measurement z_k), the likelihood $p(z_k \mid x_k)$ will be low. Intuitively this seems correct, since a big innovation term ν should give rise to a low likelihood - the observation given the particle is not likely. The observation would be more likely if the innovation term is small, since this would indicate that our prediction of the measurement is following the correct value.

By using a threshold on either $p(z_k \mid x_k)$ or the squared normalized innovation term also known as the Mahalanobis distance, we are able to find outliers. A higher threshold would indicate that we let more particles through, whereas a low threshold value indicates we are more careful. If a data set contains a lot of outliers, we would want to be more careful.

E. Association, outlier detection and weighting

During data association, the particles receive weight corresponding to the Ψ of the closest measurement, where Ψ is the probability of the measurement given the particle state. Note that the association part is about finding the cluster group that a certain particle correspond to, about associating measurements with particle groups. To determine the weight of the particle, we calculate the likelihood given below (a Gaussian which is a function of the innovation and the measurement variance). This procedure is the same as for a standard particle filter, see equation 5

During data association when only one line is detected, we simply use a threshold (λ_Ψ) on the particle's Ψ-value to determine if the particle is an outlier or not. It is subsequently excluded from the resampling process. This threshold is determined experimentally, by looking at a histogram (of the weights of the particles) illustrating the scenario where only one line was captured by the camera sensor, see Figure 4 for an illustration of this. The threshold is chosen as the line that separates the groups.

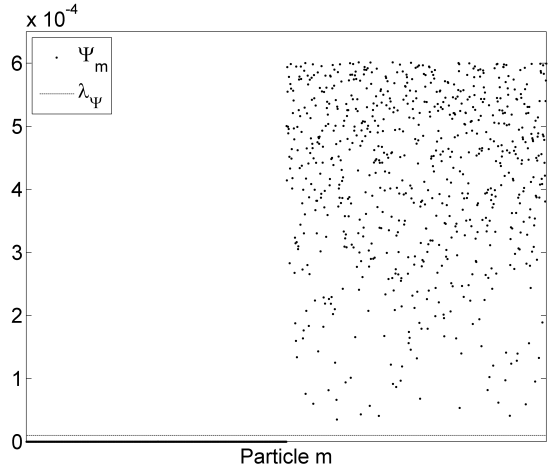

Figure 4. The particle weights plotted for each particle when only one line was detected. The dashed line is the chosen threshold, and is shown to clearly distinguish between the two clusters of particles. The y-axis corresponds to weights, and the x-axis are all the partcles correspinding to these weights.

F. Particle deprivation within clusters

In this report, we introduce the notion of particle deprivation within clusters. Usually, particle deprivation refers to the loss of diversity in a particle set. However, since we are using a single particle filter to track multiple objects we need to consider particle deprivation within groups - loss of diversity in a cluster, not the entire and combined particle set.

In a scenario with a set of particle groups, when one or more of the current groups *lose* particles to a more dominating group, we will have lost representation of a certain object we are tracking (the clustering will be incorrect).

The key of avoiding this lies in finding a good resampling scheme. To investigate how resampling should be done in multiple object tracking with a single particle filter we consider systematic resampling where all groups are fused into a combined particle set and stratified resampling. For instance, if we were tracking two line segments, i.e. two points in the 2D Hough space, we will have two distinct particle groups.

1) Systematic resampling with fused particle groups: This implies *open cluster boundaries*, i.e. particles from one group can easily be *moved* to another group. The resampling step will incorporate the measurement z_k and move particles to higher posteriori probability. In a sense, the resampling step is an implementation of the Darwinian evolution. Particles with low posteriori probability will eventually be removed (not drawn). The risk of all particles being attracted and moved

to a more dominating group where the filter believes the posteriori probability is high, will increase at each resampling. This occurs only when we fuse the particle set (no clustering).

2) Stratified resampling: This is a more refined way of dealing with multiple tracking with a single particle filter. Instead of resampling a fused set of particles, we apply light K-means grouping. For instance, if we are tracking a lane with two stripes, we need two clusters. The resampling process of each cluster is independent of the resampling process of all other clusters. Since we apply clustering each iteration of the particle filter, we are assured that particles within groups that need to diffuse to other groups are still able to do so (the K-means clustering will make sure of this since it clusters according to distance), but the risk of loosing diversity in a particle group decrease significantly.

G. Resampling procedure

The process of resampling needed some tweaking in order to allow the clusters to survive even when lanes are obstructed. Our system was designed to react differently in two distinct cases: when only one lines was detected, and when two lines were detected.

When only one line is detected, it is an indication that one of the lines are being obstructed. In this case, we do not want the particle cluster corresponding to the obstructed line to become deprived of particles. We have implemented our outlier detection to be activated at this time, and identify which particle group does not seem to correspond to the detected line. This group does not take part in the resampling.

However, when two lines are detected we use K-means clustering to identify which particles correspond to which line. These clusters are then resampled individually, so that they do not steal particles from each other.

Every set is resampled using systematic resampling.

H. Light k-means clustering

During the construction of the multiple object tracking Particle filter we were faced with the emerging problem of natural clustering of the grouped particles. The two main problems of clustering we intend to discuss here occurs in resampling, and in the end of each particle filter iteration where we extract statistical properties of the particle groups. The statistical mean serves as an approximation of the peak in the 2D Hough space, and indirectly the line segments that represent the lines of the lane in the image.

The clustering method needs to be computationally light since it functions on all particles, each iteration of the particle filter. We require also that the clustering is in a sense automatic and natural. The K-means clustering method is a viable option, however it suffers from being computationally complex as the number of particles increases. Most of the time is spent on computing vector distances. One such operation costs $\mathcal{O}(M)$, where M is the number of particles. The reassignment step computes $K \cdot N$ distances (K is the number of clusters, and N is the number of attributes), so its overall complexity is $\mathcal{O}(K \cdot N \cdot M)$. In the re-computation step, each vector gets added to a centroid once. For a fixed number of iterations I, the overall complexity is therefore $\mathcal{O}(K \cdot N \cdot M \cdot I)$. Thus, K-means is linear in all relevant factors: iterations, number of clusters, number of vectors and dimensionality of the space. However, even a linear algorithm can be quite slow if one of the arguments of $\mathcal{O}(\ldots)$ is large (M usually is large), therefore we need to approximate the K-means method. A simple approximation results in what can be denoted as the "light" K-means approach - a method that implements sparse matrices, coefficient placement outside of summations and computing $cx - 0.5c^2$ instead of the full Euclidean distance $x^2 - 2cx + c^2$. The speed gain is also obtained with vectorization. Another viable option is mean shift clustering.

We feel the need to stress the influence of the clustering method when extracting the mentioned statistical properties from particle groups. Let us return to the notion of particle deprivation in groups. Note that the quality of a tracking device is associated with its robustness. Particle deprivation in a particle group if not properly addressed is a serious issue. Even though the risk of particle deprivation in clusters can be reduced with for instance infrequent resampling, the particle filter will benefit from a carefully chosen clustering method. Consider the scenario where we have reduced the frequency of resampling to avoid particle deprivation in neither of the current particle groups. However, during some iterations where we do resample, the number of particles in one group decreases severely. In this scenario, we require that the natural clustering method used to extract the statistical properties for both group is able to cluster the groups correctly even though one of the group contains a lot less of particles compared to the other groups. If we succeed at this, then we have in some sense bypassed the effect of particle deprivation in one group, and we are able to *save* the situation and hope that our reduced resampling is able to balance the number of particles again. In a scenario where we have chosen the clustering method poorly, we might end up with one lesser statistical property - due to the merging of the small particle cloud and any large one in its surrounding.

The light K-means approach proves to be efficient in speed and in memory footprint but also, is able to extract statistical properties as we intend it to.

I. The advantages and drawbacks of refined preprocessing

The particle filter we use is fed with measurements z_k that indirectly are obtained by a camera sensor. The raw camera data is immediately converted to a grayscale image, and we apply Canny edge to extract edges and morphological operations such as erosion and dilation to reduce clutter, reduce noise and reduce the amount of pixels that the 2D Hough transform need to process. Note that of the mentioned operations, the Canny edge operation is by far the most time consuming according to available profiler tools in MATLAB.

Since our Particle filter implementation uses no control ($u_k = 0$) and only diffusion in the predictive step, the quality of the tracking will basically depend on the quality of the

update step. The resampling process will incorporate the measurement z_k and the weighted particle set will be an approximation of the posteriori belief $\overline{bel}(x_k)$. This tells us that unless we are able to provide the particle filter with accurate measurements z_k to incorporate at each filter iteration, our particle filter will likely either diverge or converge very slowly. We draw the conclusion that the preprocessing of the image data is a vital step in our particle filter implementation, since our implementation depends mostly on the update step. However, the predictive step is still important for discovery of the state space and in cases where our camera sensor is not able to provide all line segments on a road, one or more of the particle groups estimating *a missing* line segment (in that particular iteration) will only be diffused, i.e. they will not be affected by the update step since the measurement data z_k lacks an actual measurement for *that* line marking. Usually, if we are estimating two line markings, z_k is a $2x2$ matrix, and when one line is missing in an image frame, z_k is $2x1$ vector.

The main advantage of refined preprocessing of camera data is according to the discussion above the benefit of more accurate measurements z_k, and thus better tracking. Unfortunately, a refined and better preprocessing step is associated with being more computationally complex, thus adding to the duration of every time step in the particle filter. Eventually, this will render the particle filter poor in any practical scenario. Speed is one of the most attractive properties of a Particle filter and image operations such as those mentioned above will benefit from being done on a GPU rather than CPU, thus reducing the duration of each Particle filter iteration.

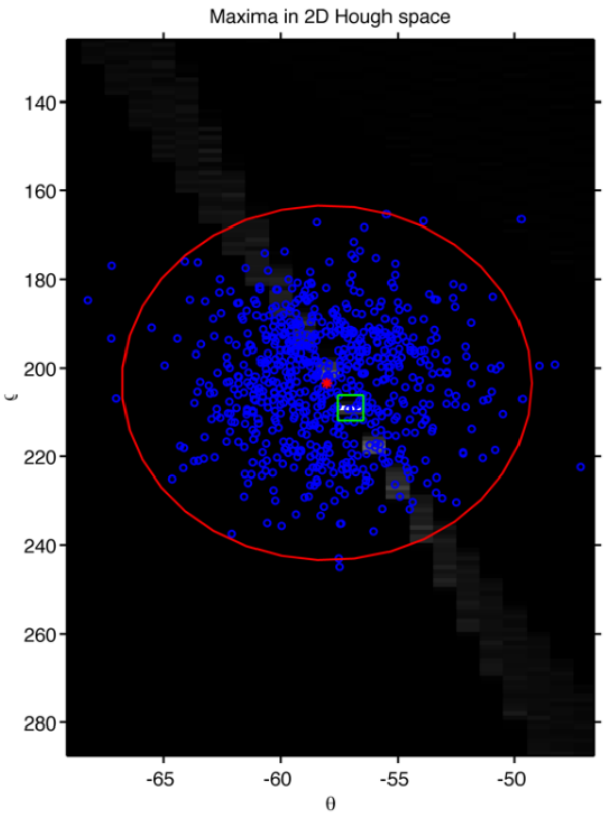

Figure 6. A picture of a cluster and its statistical properties: The blue dots signify particles; the red star the cluster mean; the red line a measurement of the spread; and the green square are the Hough space peaks discussed earlier - also known as our measurments z_k.

IV. Experiment

In order to verify our algorithm and to check whether or not our tracking device performs as intended we preform several experiments so as to scrutinize all the vital components of the device.

Detection and tracking are the most essential subcomponents. Detection is preformed with a set of digital image processing operations, namely Canny Edge detection, Morphological operations and the 2D Hough Transform. The effects of the operations are shown in Figure 5.

Tracking of the lane markings is done in the 2D Hough transform as described. In order to test the tracking of lines we need to make sure we are able to feed the Particle filter with accurate measurements z_k. We should be able to notice that the algorithm is able to keep track of the line parametrization in the Hough space providing that z_k measurements are precise, both Σ_Q and Σ_R are correctly designed, and with an appropriate association and outlier detection. In order to better visualize the tracking we plot the Hough space maxima (green squares in the figures) on top of the calculated means $\mu_{1:N}$ (red dots in the figures) of the particle groups (every group is tracking a line marking of the lane), see Figure 9. A good tracking device should show statistical means $\mu_{1:N}$ of the groups following the measurements z_k correctly - in this case the measurements z_k are actually the Hough space maxima. In other words, we are able to verify good tracking of the lane markings by noticing that the statistical means are on top of the Hough space peaks, as seen in Figure 6. One is also able to verify good tracking by plotting the tracked lines on top of the original image and seeing that the plotted lines are framing the lane.

Association and outlier detection is according to the authors opinion one of the most crucial parts of the tracking device. A poorly chosen outlier threshold or a badly designed association will render the particle filter useless. To examine these components we consider how fast the filter converges and we examine how the outlier detection works for different threshold values. We discuss a way to find an appropriate threshold value λ_ψ by analyzing the particles (as discussed in Section III).

Modeling of the noise means to find the covariance matrices that make the filter perform optimal. To begin with, when the process (motion of the target) is not noisy, and the measurement process is not noisy either, the particle filter is unnecessary. In other words, it might not be necessary to implement and optimize a particle filter if it is known ahead of time that the motion is well behaved. However, this is not the case here. Therefore, we assume that our measurement and process noise covariances are nonzero values. Both Σ_Q and Σ_R can be estimated by analyzing the variation of the maxima in the Hough space. As said earlier, the point is finding the Σ_Q

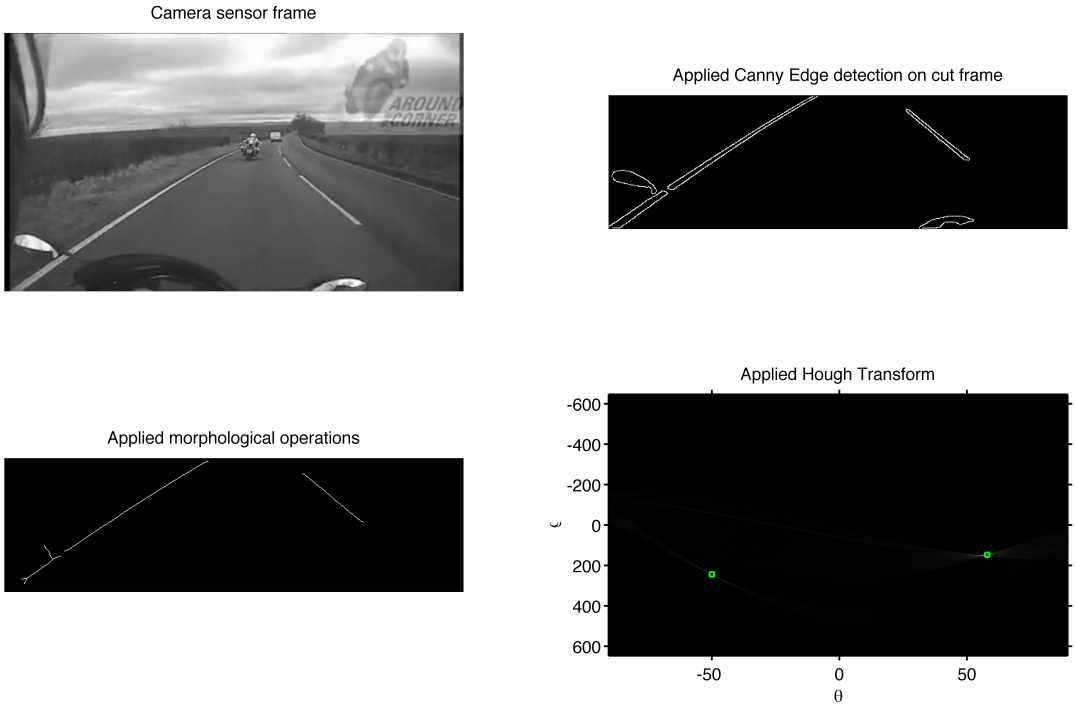

Figure 5. A visualization of the different stages of the preprocessing algorithm.

and Σ_R that gives a good outcome, so even though analyzing the position data is a good idea, if we have some rough idea of Σ_Q and Σ_R, a trial-and-error approach might give a good starting point if we have some prior knowledge of the system. By combining prior knowledge (for instance, we might know that the sensors are bad, and thus can model this as increased measurement noise), and analyzing known position data, we are able to design a good noise model.

In order to examine the tolerance level to noise and to corrupt data(robustness) we inject noise by leaving some of the noisy features in the original image untouched during preprocessing.

One of the most interesting cases of occlusion is overtaking. A Particle filter for lane detection should be able to keep track of the lane markings during an overtake. Lane tracking with only a Hough transform will not be able to keep track of the lane as soon as it is not able to see the lane markings. A Particle filter can overcome this problem. By simulating an overtake we intend to verify that our tracking device is better than just digital image processing. We expect the statistical covariance of the particle group tracking the occluded line to increase in size due to the fact that this particle group is only subjected to diffusion and no resampling (since the measurement we would incorporate to that particle group is missing due to overtake). In order to better visualize this we plot covariance ellipses around each group. We should be able to notice a growing covariance ellipse during occlusion, that, as soon as the overtake process is over, should snap back to its original smaller size. We expect the tracking device to be able to track the lane despite the overtake. To simulate an overtake, the right half of the binary image used in the Hough transform was set to 0 for 50 frames (frames 50 to 100 in the dataset).

V. Results

The performance of our filter is satisfactory. We have managed to tune our parameters to such extent as to produce stable results for our sample data.

A plot of the innovations in θ throughout the whole data set can be seen in Figure 7. As expected, the innovation bears striking resemblance to a Gaussian. The center peak corresponds to the innovation calculated from a certain particle cluster to the line this cluster is estimating, the ”rightmost” peak corresponds to the innovation corresponding a particle cloud and the line this cloud is not estimating, and vice versa. The rightmost peak is larger, as this corresponds to the innovation from the cluster which suffered from outliers.

In the end, we settled on these parameters for the motion model:

$$\begin{cases} \sigma^2_{\rho\rho} = 2.4954 \cdot 10^2 \\ \sigma^2_{\theta\theta} = 9.5456 \cdot 10^2 \\ \sigma^2_{\rho\theta} = 0 \end{cases}$$

and these parameters were chosen for the measurement model:

$$\begin{cases} \sigma^2_{\rho\rho} = 7 \cdot 10^2 \\ \sigma^2_{\theta\theta} = 10^2 \\ \sigma^2_{\rho\theta} = 0 \end{cases}$$

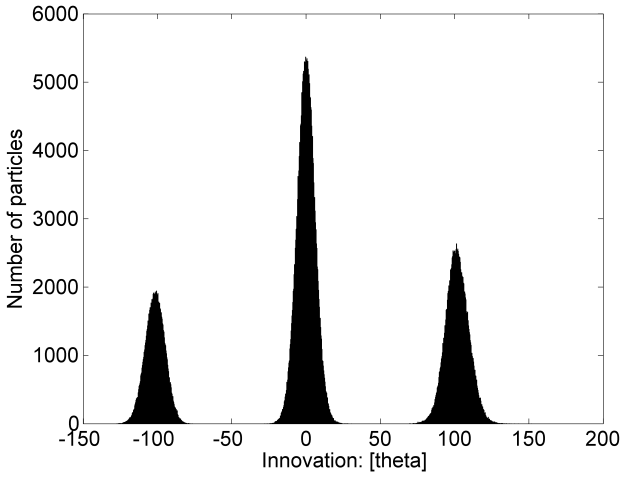

Figure 7. Histogram of innovation throughout the sample data. This data contains every calculated innovation for the entire run. The two peaks correspond to innovations from a particle to the "other" line.

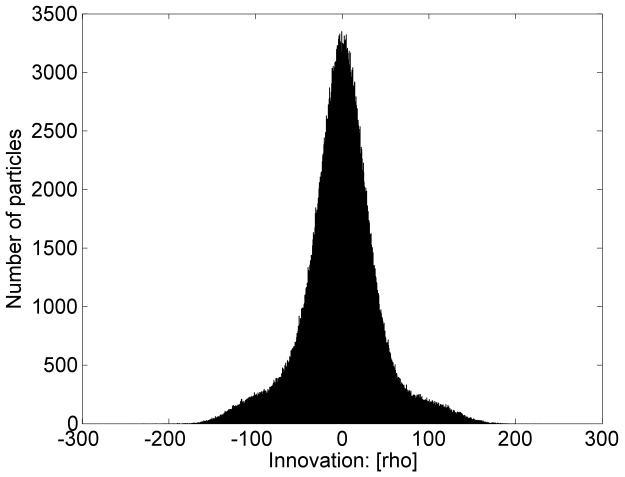

Figure 8. Similar phenomenon as the θ-case (Figure 7), but the values are too similar to produce distinct peaks.

These are estimated as we described in Section IV. The same phenomenon can be seen for ρ in Figure 8, but the ρ-values does not differ significantly between the two estimated lines, which gives this merged peak. The reason they are so close is because the ρ-parameter is similar for the two lines, as can be seen in Figure 9.

The filter displayed a frame rate of around 6.3 frames per second (4.3 frames per second with verbose flag on), which is acceptable as no attempt has been made to optimize the code. Equivalently, the processing time per frame is $6.3^{-1} \approx 150$ ms. For comparison: this gives the filter a "reaction time" of 150 ms, compared to 384 ms for the average human [7].

No analysis has been done with regard to the limitations of the image processing part of this system, apart from the occlusion of lines. Factors such as change in illumination, no road paint etc. is beyond the scope of this project.

To find a suitable threshold for the outlier detection, as discussed in section III, Figure 4 was produced and analyzed. This is a scatter plot of Ψ-values ($p(z|x)$) for every particle when only one line was detected. The left cluster has a very low Ψ, and the rightmost cluster has a much clearer distribution. The value $\lambda_\Psi = 10^{-5}$ was experimentally verified to work, and is shown in the Figure 4 as a dashed line. Both the localization and tracking problems performed satisfactory, the only discernible effect was the difference in the number of particles per cluster.

VI. Discussion

A. Lane detection

The data set used in the simulation is a stream of images captured by a camera sensor filming a real scenario, i.e. the data set is not manufactured but rather real in that sense that it is actually filmed. Therefore, it is not far fetched to claim that our tracking device should be able to preform well in practical scenarios. One of the main goals set out prior the work of this project was to try to create a concept. The basic yet very powerful idea was to create a vital component in a driving assistance system by decomposing the task into lane detection and lane tracking. If we were able to prove that our lane tracking device is capable of keeping track of a lane extracted from real data, then we would be one step closer to this concept. The film associated with this report is in some sense a proof of concept.

It has been shown in [7] that a *simple reaction time* is shorter than a *recognition reaction time*, and that the *choice reaction time* is longest of all. It has also been concluded that simple reaction times averaged 220 ms but recognition reaction times averaged 384 ms [7]. The reaction time for the presented tracking device is about 150 ms, as discussed in section V. It is important to realize that even though the average human reaction time is somewhere between 0.2-0.4 ms we need to take into account the temporal effect. For instance, as human beings we tend to get tired when driving long distances, and that we in addition to the reaction time need time to make a decision - the *real reaction* time will be longer when we take even more factors into account. We claimed in the beginning that society undoubtedly has an interest in reducing the number of car accident. A tracking device such as the one we have presented preforms very well, and this justifies the development of different kind of estimation filters.

Inverse perspective mapping (IPM) is a form of image processing that frees the captured frames from perspective effects. The presented lane detection component has not utilized this. IPM projects the lane markings to parallel lines - this enables us to simplify the tracking process. For instance, consider a lane with two lane markings: IPM gives us the choice of just tracking one of the lines (given we knew the distance between the two lines). In our case, this would imply that we only need to track one line, making the clustering redundant and thus increase tracking speed and reduce complexity. The article [3] gives an excellent description of IPM.

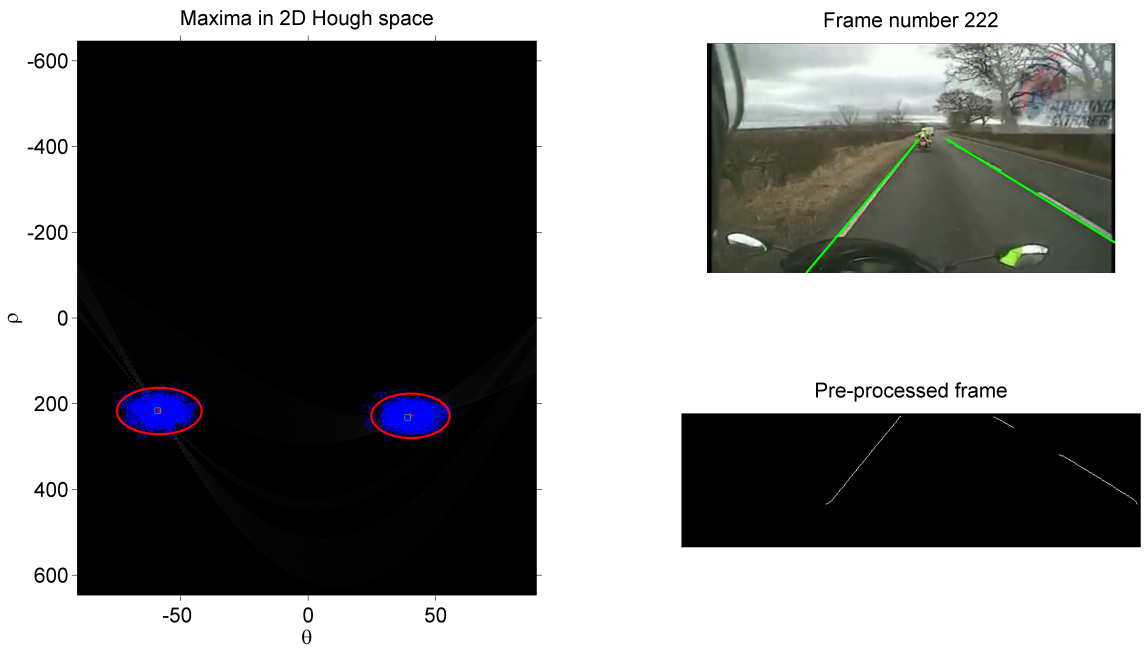

Figure 9. Final frame of the sample data shown with current particle states (blue), cluster means (red stars), cluster covariances (red circles) and detected peaks (green squares).

Our lane detection component does not make any difference in lines. For instance, both a dashed line and a solid line will be registered as a simple straight line with no distinguishing. A refined lane tracker should be able to make a difference between this kind of lines since they usually *carries* different kind of information. For example, a dashed line usually signifies a boundary between lanes, while a solid line should under no circumstances be traversed.

B. Lane tracking

In this setup, a very simple motion model was used. Since the prediction step only contains diffusion, the estimate of a ”disappeared line”, i.e the mean of the particle cluster corresponding to an occluded line, never changes since the particles will simply be spread out equally in all directions. This is by no means an optimal choice of predictor since tracking an occluded line was one of the objectives - more research is needed in this area. However, albeit the simplification of motion model, our lane tracking device performs very well.

Since the orientation of the lanes obviously are dependent, as discussed, this assumption could be used to make more precise predictions of the next position of the lines on lanes. If it was possible to detect and measure curvature of the road, or have access to the pose of the vehicle, more accurate predictions could be supplied. In addition, statistical analysis could be performed on roads to assess a probability distribution of the lane. [4]. These parameters (such as vertical drift etc) could be estimated using an estimation filter, but the data would be more reliable if it was available from the vehicle itself. This should not be difficult to implement, since most modern vehicles contain sensors and electronic interfaces to deliver just this type of data to the manufacturer and car mechanics. In conjunction with a motion model of the car, a change in reference frame can be used to describe the motion of the lanes.

In our case a simple Gaussian was used to estimate the observation probability distribution. However, when using the Statistical Hough transform an alternative description, as presented in [1], is more suitable. It is still based on the Normal distribution, but is directly tied to the resolution and properties of the Statistical Hough Transform.

When we implemented our observation-likelihood function we only used intuition and experimentation as well as analyzing the variance in the data set, to establish a suitable covariance for the observation model. In a real application, a more thorough analysis may be required, depending on the demands on the performance.

The particle filter was chosen for our implementation, because it relaxes the demands on the a posteriori distributions. In our case however, the posterior belief is possible to describe with a Gaussians. Since the process model is linear, this means that the use of a Kalman filter is feasible. The Kalman filter is in itself more computationally cheap than the particle filter. This is useful if the state space is chosen to be large, as mentioned in section III. The particle filter suffers greatly under the curse of dimensionality, and the Kalman filter could be used to ease the computational load. However, this is not necessarily the case when more advanced descriptions of the motion and measurement model are available, and future analysis could show that the posterior distribution is too complex to be estimated as a Gaussian. With the current, simple modeling

applied, the Kalman filter would have been effective, but the versatility of the Particle filter makes the Particle filter a better choice when experimenting with different models.
In order to limit the diffusion of a group during the overtake process, we reduce the size of the process noise covariance. Keep in mind that particle groups corresponding to a missing line are not resampled during overtake since there are no measurements of occluded lines available to incorporate in this scenario. If we did not limit the process noise we would notice a too big diffusion of the particle groups corresponding to an occluded line, causing poor tracking during an overtake. The reason as to why we simply did not use a lower process noise covariance to begin with is that such a low covariance value would render the tracking poor.

VII. Future work

The obvious continuation of this project is of course to estimate the position of the vehicle in relation to the lane edges. Either to be used for automatic control, or simply to use as warning during cruise control - a driving assistance system.

The tracking described in this report would benefit from a more refined association and outlier detection, as well as being written in a lower level language such as C++ with GPU acceleration.

A more refined motion model should be developed for practical applications, and possibly a new state space description.

VIII. Conclusions

This report describes an implementation where particles are divided into clusters - each cluster tracking one point in the Hough space. Using a large state space is cumbersome when using a particle filter, but this implementation proves that it is not necessary to extend the state space when tracking multiple objects using a single filter. K-means clustering has been shown as a useful, computationally cheap method of identifying the different groups of particles. This implementation appears powerful enough to maintain a feasible estimate of a lane, even when parts of the lane are not detectable by the image preprocessing, for instance during overtake.

References

[1] "Combining statistical hough transform and particle filter for robust lane detection and tracking," in *Intelligent Vehicles Symposium (IV), 2010 IEEE*, june 2010, pp. 993 –997.

[2] S. Mills, T. Pridmore, and M. Hills, "Tracking in a hough space with the extended kalman filter," in *The British Machine Vision Conference, Norwich, 2003*, pp. 173–182.

[3] M. Aly, "Real time detection of lane markers in urban streets," in *Intelligent Vehicles Symposium, 2008 IEEE*, june 2008, pp. 7 –12.

[4] S. Sehestedt, S. Kodagoda, A. Alempijevic, and G. Dissanayake, "Efficient lane detection and tracking in urban environments," in *3rd European Conference on Mobile Robots (ECMR 2007)*, pp. 1–6.

[5] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Pearson Prentice Hall, 2008.

[6] A. Goldenshluger and A.Zeevi, "The hough transform estimator," *The Annals of Statistics*, vol. 32, no. 5, October 2004.

[7] D. Laming, "Information theory of choice-reaction times," *Academic Press, London*, 1968.