The Pennsylvania State University
The Graduate School
Department of Electrical Engineering

ACTIVE CONTOUR MOTION PLANNING IN THE
INVERSE PERSPECTIVE MAPPING FRAME

A Thesis in
Electrical Engineering
by
Adam J. Nedorezov

© 2016 Adam J. Nedorezov

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2016
The thesis of Adam J. Nedorezov was reviewed and approved* by the following:

David J. Miller  
Professor of Electrical Engineering  
Thesis Advisor

William E. Higgins  
Distinguished Professor of Electrical Engineering

Kultegin Aydin  
Professor of Electrical Engineering  
Head of the Department of Electrical Engineering

*Signatures are on file in the Graduate School
ABSTRACT

This work explores the use of active contours for the purposes of motion planning in a highway environment. Traditionally active contours have been used in image segmentation applications. The gradient descent approach used with active contours parallels motion planning algorithms that use potential fields to optimize routes. By utilizing active contours additional constraints can easily be applied between nodes on the contour allowing for control over the smoothness of a planned route. These motion paths are planned in the Inverse Perspective Mapping frame to mimic planning with a bird’s eye view of the scene. Our results showed that our motion planning algorithm has moderate performance that makes it better suited to be used as one algorithm in a set of algorithms that collaborate through some voting scheme to determine the correct solution.
LIST OF FIGURES

Figure 3-1: Original Image ......................................................................................................17
Figure 3-2: Canny edge detector run on grayscale version of original image .........................17
Figure 3-3: Accumulator of mapped edge pixels.....................................................................18
Figure 3-4: Detected line segments from the Hough Transform .............................................18
Figure 3-5: Candidate vanishing points created by perspective line intersections ..................20
Figure 3-6: Sequence of MCT localizing on vanishing point ..................................................24
Figure 3-7: Error plots of vanishing point estimator ................................................................26
Figure 3-8: X & Y Error plots of vanishing point estimator ....................................................28
Figure 4-1: Example perspective image to be transformed .....................................................31
Figure 4-2: Example from Figure 4-1 after IPM transformed image applied ...........................31
Figure 4-3: Example roadway scene prior to road surface removal ........................................34
Figure 4-4: Example from Figure 4-3 after road removal algorithm .......................................34
Figure 4-5: Wet road surfaces causing misclassification of non-road surfaces .......................36
Figure 4-6: Simulated perspective image .................................................................................39
Figure 4-7: Simulated BEV image of scene in Figure 4-6 .......................................................40
Figure 4-8: IPM Transformed image of Figure 4-6 for comparison to Figure 4-7 ..................40
Figure 4-9: IPM transformed image of Figure 4-3 ..................................................................41
Figure 4-10: All obstacle detections overlaid on IPM image ..................................................44
Figure 4-11: Filtered obstacle detections overlaid on IPM image ...........................................45
Figure 5-1: Demonstrating convergence issues of traditional snakes. Source: Xu and Prince [24]......................................................................................................................50
Figure 5-2: Initialization of active contour over binary labyrinth ...........................................52
Figure 5-3: Figure of the active contour taking a shortcut through a labyrinth wall ...............52
Figure 5-4: Weighted sloped applied over image area to add motivation towards the endpoint ..................................................................................................................................53
Figure 5-5: Active contour solution with weighted slope applied.................................54

Figure 5-6: Close up of vector field for labyrinth wall replaced with weighted slope ..........55

Figure 5-7: Weighted wall pixels allow active contour to slip over wall segment .............55

Figure 5-8: Weighted wall pixels are insufficient to force contour off of wall segment .......56

Figure 5-9: Weighted wall pixels force solution into intersection of wall segments ..........57

Figure 5-10: Clockwise rotation around labyrinth wall segments ..................................58

Figure 5-11: Clockwise rotation along wall edges allowing solution to backtrack around wall segment. ..................................................................................................................59

Figure 5-12: Another example of a clockwise rotation along wall edges allowing solution to backtrack around wall segment.................................................................59

Figure 5-13: Simple labyrinth case unaffected by clockwise rotation addition...............60

Figure 5-14: Clockwise wall edge rotation applied to Counterclockwise labyrinth ...........61

Figure 5-15: Reversing the weighted slope with a clockwise wall edge rotation .............61

Figure 5-16: Counterclockwise wall edge rotation applied to Counterclockwise labyrinth ....62

Figure 5-17: Reversing weighted slope and counterclockwise wall edge rotation ..........62

Figure 5-18: Conversion of IPM image into edge map for GVF snake algorithm ...............64

Figure 5-19: Expansion buffer applied to side by side obstacles....................................65

Figure 5-20: Comparing obstacle edge rotational directions ........................................67

Figure 5-21: Example scene generated in Matlab.............................................................70

Figure 5-22: Test Case 1 results plots..............................................................................75

Figure 5-23: Test Case 1 sequence plots........................................................................75

Figure 5-24: Test Case 2 results plots..............................................................................77

Figure 5-25: Test Case 2 sequence plots........................................................................77

Figure 5-26: Test Case 3 results plots..............................................................................79

Figure 5-27: Test Case 3 sequence plots........................................................................80

Figure 5-28: Test Case 4 results plots..............................................................................81
Figure 5-29: Test Case 4 sequence plots ................................................................. 82
Figure 5-30: CamSeq01 data set result plots ........................................................ 85
Figure 5-31: CamSeq01 data set sequence plots ................................................... 86
Figure 5-32: seq05VD data set result plots .......................................................... 87
Figure 5-33: seq05VD data set sequence plots ...................................................... 88
LIST OF TABLES

Table 1: Tabulation of performance metric for closed loop simulation ................................................ 83

Table 2: Tabulation of performance metric for real world video sequences ........................................ 88
ACKNOWLEDGEMENTS

First and foremost, I would like to express my thanks to my supervisor Dr. Benjamin B. Thompson who has guided me throughout my thesis. He has supported my efforts with his endless patience and incredible insights. Our weekly meetings have marked the steady progress to thesis completion and even a few circuitous but enlightening paths.

Secondly, I would like to express my gratitude to Dr. David J. Miller for serving as the committee chair and providing perceptive feedback that helped shape the discussion points in this thesis.

I also would like to thank Professor William E. Higgins for willingly agreeing on being a committee member.

I would also like to acknowledge the gracious financial support received from the Applied Research Laboratory at Penn State which helped immensely during the writing process.

Finally I want to thank my family for inspiring me to pursue my graduate education and providing their unending support.
Chapter 1  Motivation

In this paper, we propose a vision based, motion planning algorithm for a self-driving car on highways that utilizes a single camera. There are a lot of tradeoffs in using a single camera design. For instance, a single camera makes acquiring precise range information challenging but the algorithms tend to be less computationally intensive. There is also no guarantee that a single algorithm or approach can or will always make the right decision. In fact Google, a company well known as a leader in the development of self-driving cars, reported its first at fault crash in July of 2016 [1] after seven years and well over a million miles driven.

To help minimize the possibility of incorrect decisions, it would be useful to use a set of motion planning algorithms which would each be used to derive their own motion planning solution from different sensor resources. These solutions could then be compared to one another in a voting schema to increase the confidence that the final solution is the right decision. Our primary focus in this paper will be on our proposed algorithm and its performance. It was designed to use minimal resources (i.e. a single camera) and to not be as computationally intensive as stereo-vision or LIDAR systems can be. Because of this, our algorithm would be an ideal candidate as a secondary or tertiary algorithm in the set of voting algorithms used to confirm a final solution.

Our motion planning vision algorithm is composed of three subsystems: a vanishing point detector, a situational awareness system, and a path planning system. The vanishing point detector provides the car with a desired travel point in highway environments. The situational awareness system determines where obstacles such as cars and trees are in relation to the car and camera. Lastly the motion planning system maps a path from the camera to the vanishing point while avoiding obstacles identified by the situational awareness system.
1.1 Motion Planning Problem Statement

We may describe the problem of path planning as one of determining a feasible route through an image or video feed of an autonomous vehicle traveling along a highway environment. The autonomous vehicle is a car capable of acquiring images or video which are used to compute navigational steering commands through the environment without human input. A highway environment is defined as a roadway system that has more than one travel lane in a given direction. Feasible routes are considered as ones which are able to avoid obstacles such as traffic, trees, potholes, etc. while still remaining within the designated travel lanes.

1.2 Application of Motion Planning

The use of automated assistant driving algorithms is gaining popularity with many automotive brands. A shining example is the Lane Keeping Assist technology pioneered by the Toyota Motor Corporation. This system is able to recognize lane markings on the roadway, warn the driver when the vehicle starts to deviate from its lane and even assist the driver to steer and keep the vehicle on course in the center of the lane [2]. Technologies like this however still have a long way to go before the system will be able to plan a route around an obstacle in the current travel lane. These obstacles can be slower moving vehicles, fallen trees or debris off the back of a truck that has yet to be cleared off to the side of the road. In our work we will focus solely on slower moving vehicles that are ahead of the autonomous vehicle.

1.3 Proposed Solution

The primary purpose of our motion planning algorithm is a cruise control application that allows the autonomous vehicle to pass slower moving traffic. Using a single camera as our only
sensory input, we tried to reduce the needed features to the essentials that were either a direct result or byproducts of an algorithm associated with using a single camera. For instance, instead of having a separate costly sensor, such as LIDAR, that can provide range and relative velocity between the vehicle and other objects, we utilized the Inverse Perspective Mapping to estimate range based on known camera properties and the pixel location of an object in the transformed image. Additionally, this transform produces a streaking effect that we utilized during the classification phase to identify what areas of the image should be avoided as obstacles.

We broke down the problem of motion planning in a roadway scene into three components. The first is the problem of determining the region in the image the vehicle is traveling towards. The second problem is to identify the road surface and any obstacles that may hinder the vehicle's ability to continue traveling in its current lane. The final component is to plan a path through the roadway scene along the road surface that will guide the vehicle around obstacles and towards the goal point some look ahead distance in front of the vehicle.

The following work presents the development of three algorithms designed to work together and provide guidance commands for an automated vehicle traveling in a highway environment without human input. The algorithms rely solely on a single camera feed and have no a priori knowledge of any GPS mapping of roadways or ranges to objects through other sensors such as LIDAR.

In Chapter 2, we present some of the previous work that has been done in the areas of vanishing point estimation, road surface and obstacle detection, and active contours. We present our design for a vanishing point estimator in Chapter 3. In Chapter 4, we expand upon existing techniques for road surface detection and build a classifier to identify obstacles from a transformed image. In Chapter 5, we first simplify the motion planning problem as a binary labyrinth and then expand the results to real images, video sequences and closed loop simulation. We present our conclusions and recommendations for future work in Chapter 6.
Chapter 2  Literature Review

2.1  Overview

We performed the following literature review to evaluate relevant work on motion planning in highway environments. Numerous examples that are related to the three components of the motion planning problem are provided. These components are vanishing point estimators, road surface and obstacle classifiers, and motion planning algorithms.

2.2  Vanishing Point Estimation

The vanishing point in an image is able to provide a lot of geometric clues about a scene which are useful for determining one’s orientation within a scene or a potential steering direction. This is because these geometric clues are typically derived from the edges of roadways, lane markings or other man-made structures and how these edges converge towards the horizon due to the perspective effect of a forward looking camera.

Coughlan and Yuille [3] use a Bayesian model to combine knowledge of the Manhattan world three-dimensional geometry with statistical knowledge of edges in the image to derive the orientation of the camera to the scene captured. In roadway scenes this can be converted to the direction of travel of the roadway. In addition to compass direction, obstacles can be detected by checking an object’s alignment with the Manhattan world model.

While this algorithm was designed to work for both indoor and outdoor scenes, the angle estimates were worse by 5° for outdoor scenes as compared to indoor scenes, bringing the overall estimate error to 10°. Given that the minimum highway traffic lane width is 3m [4] and a look ahead distance of several car lengths (~13.5m), then a desired measurement error of half a lane
width at the look ahead distance results in a maximum angle error of 6.3°. This desired error limit is less than the stated 10° angle estimate error by Coughlan and Yuille for outdoor scenes.

Another approach is to convert the edge information into line segments by way of the Hough transform and analyze the regions where they meet. Almansa et al. [5] computed the probability of a segment passing through candidate vanishing regions to determine the three prominent regions that reflected the three axes of the camera. The desired angular resolution can be tuned to any level of fidelity at the expense of requiring more line segments and increasing the detection threshold. Additionally the candidate vanishing regions were all distributed around the image ignoring the center of the image. Given that in a highway scene our desired vanishing point is more likely to be contained within the image, a new sampling schema for the candidate regions would be required.

As an alternative to using pure edge features Kong et al. [6] utilized Gabor filters to extract texture orientation which was then used in a soft-voting scheme to localize the vanishing point by choosing the candidate with the highest score. The soft-voting scheme was used to adjust the weights within voting regions around vanishing point candidates and dampen the effect of pixels large distances away from the candidate point. The estimated vanishing point is then used as a strong clue for localization of the road region. The strength of this algorithm is in its ability to identify road regions in desert, winter or wilderness scenes where the road is an unmarked pathway and the Gabor filters are able to use the texture to identify road boundaries. The soft voting scheme creates a local voting region scheme that prevents distant points from skewing the estimate.

The previous works focused on using the vanishing points to determine camera orientation relative to the scene, and obstacle and roadway localization. Vanishing points have also been shown to be useful in steering of a robot as demonstrated by Schuster et al. [7]. By
computing the turn and tilt of the robot for a given vanishing point, they were able to compute the rotation necessary to align the robot to travel parallel to known guide lines.

### 2.3 Road Surface and Obstacle Detection

The problem of object classification is a field of study all to itself. For humans, obstacle classification in a roadway scene boils down to what looks like an object that should not or cannot be driven over and how fast that object is moving relative to the car. These features can be deduced from ranging sensors, such as LIDAR or Ultrasonic, through stereo-vision systems or even single cameras. Each sensor has their strength and weaknesses as well as requiring compatible algorithms to manipulate the data.

The current popular choice for road surface and obstacle detection by autonomous vehicles is Light Detection and Ranging known as LIDAR. LIDAR is a type of surveying technology that uses focused laser light to calculate the distances to an object by measuring the time it takes for the light to travel from the sensor to the target and back. While this provides the most accurate measurements it is currently susceptible to interference from spurious signals. As the density of these autonomous vehicles on the road increases, the use of an active system like this considerably increases the mutual interference between these LIDAR sensors [8]. Jonathan Petit, a principal scientist at Security Innovation has been able to successfully demonstrate overloading the LIDAR sensor with a significantly larger number of obstacles than there really are by spoofing a copied signal to the LIDAR system [9]. Possibly by encoding or encrypting the pulsed signal, the risk of mutual interference may be reduced.

In terms of estimating relative motion to the sensors, LIDAR can simply compute this from the range information. However, to be able to identify an object, it must first stitch the point cloud data together into reasonable estimates of an objects shape which requires some
sophisticated algorithms and some nontrivial processing power. By combining the point cloud
data with video images, reasonable estimates for object shapes and locations can be made, which
is highly desirable data from a motion planning standpoint.

Stereo-vision techniques utilize disparity maps to locate differences between a pair of
images taken by cameras with a known displacement. By computing this delta between images
distances to objects can be acquired with reasonable resolution. This has most successfully been
demonstrated by NASA with their lunar rovers [10]. Using the disparity map to determine an
object’s distance to the rover, corresponding grid cells of their navigation planner would be
blocked off. However, the addition of a “bird’s eye” viewpoint was also included in their path
planning to provide more information about the rover’s surroundings. Additionally the
computation speed of creating the disparity map limited the rover’s travel speed to a little over
1m/s.

Since video cameras are a passive system, there is a near non-existent chance of mutual
interference between multiple stereo-vision systems. There is however still the potential of
external interference in the form of bright lights pointed directly at the cameras saturating the
image and effectively blinding the system. This form of interference can be more easily resolved
through the use of image filters or changes in camera parameters such as exposure time.

Similar to the LIDAR system, relative motion information can be easily computed from
the pixel data generated by the stereo-vision system. Yet, this also has the necessity for a
sophisticated method for combining the points of data into objects that can be classified as
obstacles.

LIDAR and stereo-vision techniques can be computationally intensive or costly to set up.
And nowadays higher and higher resolution camera have proliferated into nearly every type of
digital device. For this reason there are also a variety of single camera techniques that have been
applied to road surface and object detection. These techniques have ranged from image
transformation techniques like Inverse Perspective Mapping (IPM) and temporal differencing like optical flow for video sequences from a single camera. While single camera systems may not have the resolution of LIDAR or stereo vision systems, they are still capable of providing reasonable warnings about obstacles relative to the camera and would best serve as a secondary system to validate the LIDAR or stereo vision system’s results.

IPM methods transform the original image into an estimation of the top down view of the scene. Bertozzi et al. [11] first used IPM in combination with a stereo camera system; however instead of computing the traditional disparity map, they used a more direct differencing between the two IPM transformed images which fed a polar histogram to detect the triangular regions of non-overlap. These triangular regions were their clues to the obstacles’ existence. Others such as Maud et al. [12] and Jiang et al. [13] used the IPM transform for lane tracking. Nieto et al. [14] were able to stabilize the IPM image through the use of the vanishing point. This was achieved by using the vanishing point location to estimate the slope of the road and its relative position to the camera to adjust the IPM transform parameters. This resulted in more consistency from frame to frame of the IPM image because it was removing the small oscillations introduced by the camera mounting and car suspension as it travels. Tuohy [15] utilized the IPM to estimate ranges to the obstacles using the relationship between the projective transformation and camera properties.

The uses of Optical Flow methods for obstacle avoidance have revolved around the idea of looking for discrepancies between the expected motions of a pixel from frame to frame in a video sequence. Usually there is the assumption that either the camera or the obstacles are static. For example, Zingg et al. [16] use optical flow measurement to guide a Micro Aerial Vehicle (MAVs) down corridors of a building using the assumption that the only motion should be that of the MAV and therefore the delta between computed and expected flow rates provide range estimates to the walls allowing the vehicle to center itself in the corridor.
Mallot *et al.* [17] tried simplifying the optical flow computation by going to the IPM transformed frame as this would regularize the flow field to a single vector making it easier to detect obstacles by identifying regions that deviate from the regularized vector. Braillon *et al.* [18] explored the use of a theoretical optical flow vector for pixels to be used as a comparison to the actual flow value. To reduce the number of computations, they used a pixel similarity metric between the two adjacent frames to identify higher priority regions where the flow deviations would be more likely.

Because the optical flow methods relies on identifying regions with large deviations of between expected and actual flows, it can be difficult to identify obstacles that are moving with the expected flow. Additionally, thresholding the flow delta becomes challenging if the scene contains both fast and slow moving obstacles. The delta can’t be too small or a small error or slight jitter would be perceived as obstacles. And if the deltas are normalized then the slower moving regions would be ignored for the faster regions.

2.4 **Motion Planning and Active Contours**

The goal of a motion planning algorithm is to connect a starting location with a goal location by mapping a path through the configuration space that will allow the vehicle to avoid collision with known obstacles. There are a variety of methods that have been explored throughout the years such as Grid-Based Search, Support Vector Machines, and Potential Fields.

2.4.1 **Grid-Based Search**

Grid-Based Search algorithms discretize the motion of the vehicle into short line segments which are created by overlaying a grid on the configuration space. Traversal from the
starting location to the goal location is achieved by traveling along adjacent grid points so long as the line segment between points is clear of obstacles. Optimization of the path can be adjusted to minimize path length or minimize energy usage as demonstrated by Ganganath et al. [19] through adjusting the cost function for grid line segments. Wang et al. [20] were able to introduce heading and curvature constraints by using an expanded control set which introduces arced segments between neighboring grid points instead of the traditional 4-connected neighbors.

There are several downfalls to grid-based methods. The first is being able to determine an appropriate grid size. Using too fine a resolution for the grid size, the search space becomes very large. Using too coarse a resolution, the algorithm will have trouble finding paths through narrow regions of the configuration space. Grid-based approaches also suffer from exponential growth when moving to higher dimensions; however, the problem of motion planning in a highway environment can be done in two dimensions keeping the search space manageable. Lastly, the grid discretization of possible motions for vehicles as vertical and horizontal segments does not translate well to automobiles.

If the grid constraint is removed, these algorithms start sharing more similarities with Graph Traversal techniques. The grid points become nodes placed throughout the configuration space and branches between nodes represent allowed motion through the space. The difficulty with graph traversal methods lies with how to automate the placement of these nodes in an unknown area. Given that obstacles are moving and blocking random regions of the roadway, ensuring that the selected graph nodes are optimally situated to prevent travel too close to an obstacle or will allow for an optimal solution can be difficult to determine a priori. These Graph Traversal techniques are more suited to path planning at a higher level, such as the traveling salesman problem, where the automobile needs to plan which roads or highways should be taken in order to reach the final destination.
2.4.2 Support Vector Machines

An interesting approach to the motion planning problem is to treat it as a classification problem where obstacles to the left of the line segment between the starting and goal locations are considered one class and the right side obstacles are considered another class. This approach was demonstrated by Miura [21]. A Support Vector Machine (SVM) was trained to classify the two groups of obstacles. The resulting boundary line between the two groups defined a potential travel path for the vehicle from the starting location to the goal location. Since the boundary line transforms from a straight line to a curved path around the obstacles in order to create the separating boundary between the two classes, there is no guarantee that the separating boundary is smooth. The shortfall results in planned motions that are not preferable for safety, stability or comfort of the passengers in the vehicle. There is also not much ability to constrain the boundary solution to a roadway. Approaches such as this appear more optimal for scenarios in wide open spaces or where there is no defined roadway, such as off-roading, driving through a desert or on the moon.

Many techniques like to treat the planned motion path as if the vehicle is a point model. In order to avoid having additional checks to ensure the vehicle extent does not collide with an obstacle along the path, a common approach is to enlarge the obstacles by some safety factor. This creates enough clearance for the vehicle to travel along the resulting path assuming point model dynamics and removes the need to check for collisions directly. So for example, if there was a case where there were two vehicles driving side by side, instead of the motion planning solution trying to squeeze between those two vehicles, the boundary expansion would cause those two vehicles to be treated as a single obstacle that needs to be planned around. However if a gap did exist between the two side by side vehicles after the expansion from the collision buffer, then the motion planning would have a valid solution to drive between the two vehicles without
concern of collision. In order to avoid breaking traditional driving rules it would be preferable to ensure that the boundary expansion results in the obstacles occupying the entire lane width.

2.4.3 Potential Fields

Potential fields are another interesting approach to the motion planning problem. The configuration space is converted into a potential field by placing attractive and repulsive forces around goals and obstacles respectively. Then by treating the vehicle as a point in this potential field, a trajectory can be mapped out that leads the vehicle towards the goal.

A simple point based approach was taken by Galceran et al. [22] where the road surface was converted into a U-shaped driving corridor where the center of the road was a minima and obstacles and road edges were a maxima. The vehicle was then a point that was started at one end of the corridor and used the potential field to influence the steering of the vehicle. Since the effect of the potential field is being computed locally for the location of the vehicle in the corridor, there is no guarantee that the resulting steering vector will navigate completely around an obstacle or reach the destination.

Warren [23] took the approach of breaking the initial path from the starting point to the goal into several segments and then used a gradient decent approach to move the line segment endpoints to regions away from obstacles. Since the path segments were treated as piece wise linear the path does not guarantee a smooth solution. However with more line segments a smoother solution could be obtained at the expense of computation time.

In fact Active Contours expands on the idea of using more points and adds an ability to control the smoothness of the overall path which made it attractive for our work. Active Contours have typically been used for image segmentation problems. They are basically a spline that deforms based on the energy field created by edges in the image. Xu and Prince [24] developed a
Gradient Vector Flow (GVF) to replace the energy field which allowed the field to expand from the edge boundaries more into uniform regions to provide influences to the snake at further ranges as well as help the snake reach up into concave regions. It is these features that appealed to us for the basis of our motion planning algorithm.
Chapter 3  Vanishing Point Detection

The task of autonomous motion planning requires a destination to reach. GPS coordinates provide a path from a more global perspective, but in an image of a roadway the vehicle requires a pixel location to guide its steering. Using the parallelism of the edges of the roadway and lane markings typical in a roadway scene we observe that they become convergent in the image plane due the perspective projection [25]. This phenomenon has been utilized in many approaches as discussed in Chapter 2.2 to localize the vanishing point. The common steps of these approaches are to extract edge information from the image, convert those edges into line segments and then use inferencing or voting techniques to pinpoint the vanishing point.

For humans, a common technique is “aim high in steering” where the driver scans the roadway 20-30 seconds ahead of their current location [26]. This allows the driver to get a better understanding of their surroundings and plan ahead. Visually this focal point is relatively similar to the forward vanishing point in the camera image. By using this focal vanishing point and the knowledge that the bottom center of the image is where the camera is located, we are able to define the starting and goal locations in the image and can generate a travel path between these two points that is restricted to the road surface and navigates around obstacles such as other vehicles or fallen debris.

3.1 Proposed Vanishing Point Tracker

Our proposed vanishing point tracker has three stages. The first stage is to find the vanishing lines in the image. These lines are predominantly roadway edges, lane markers, building edges and other various man-made objects with straight edges. Because of the
perspective projection, these lines will inevitably intersect representing candidate locations for the vanishing point. Hence, the second stage of the algorithm is to acquire candidate vanishing point locations by computing all possible intersections of the lines extracted from the Hough transform. The final stage of our proposed algorithm is to use a Monte Carlo Tracker (MCT) to localize the forward vanishing point from all of the candidate line intersections.

<table>
<thead>
<tr>
<th>ALGORITHM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perform “Edge Detection” to create binary image of object edges</td>
</tr>
<tr>
<td>Perform “Hough Transform” on binary edge map</td>
</tr>
<tr>
<td>Extract five peaks from the Hough parameter space</td>
</tr>
<tr>
<td>Create listing of all possible line intersections from extracted line segments</td>
</tr>
<tr>
<td>Update the MCT with the new candidate locations</td>
</tr>
</tbody>
</table>

### 3.1.1 Finding Perspective Lines

Perspective lines are straight lines that run along the straight edges of objects and converge at a vanishing point. A commonly used feature extraction technique to extract line information from an image is the Hough Transform. While not all lines are necessarily perspective lines, a significant number of the straight edged objects found in the roadway scene, such as roadway and lane marker edges, tend to follow perspective lines.

The Hough Transform uses a parametric form for the equation of a line as shown in Equation (1). The \((x, y)\) location of edge pixels in an image are converted into potential \((\rho, \theta)\) pairs. In the \((\rho, \theta)\) parameter space the resulting curve is sinusoidal. By overlaying all these sinusoidal curves and keeping track of the \((\rho, \theta)\) pairs in an accumulator matrix, the cells with the
largest number of hits represent straight lines where a large number of edge pixels are lying along
the same line.

\[ \rho = x \cos(\theta) + y \sin(\theta) \]  

(1)

There exists some minimum number of peaks that will provide a good distribution of
perspective lines whose intersections provide sufficient candidate vanishing point locations. We
found that the use of five of these peaks proved sufficient in our testing as this provided ten
candidate locations. As the number of peaks is increased, the number of computations required to
find the vanishing line intersections increases which can reduce the time required to localize the
vanishing point. If the number of peaks is decreased, there will be a large reduction in the number
of candidate vanishing point locations requiring more time in order to be able to localize the
actual vanishing point.

Figure 3-1 is a sample highway scene chosen to demonstrate the Hough transform. The
first step is to obtain the edge information from the original image by converting to grayscale and
applying the Canny edge detector. The Canny edge detector uses the intensity gradients of an
image and looks for locations where there are large deltas between adjacent pixels as this
signifies an edge of an object due to a noticeable color change. As Figure 3-2 shows, this
provides a binary output of the edge pixel locations. The \((x, y)\) location of these edge pixels are
converted into \((\rho, \theta)\) pairs and added to their respective bins in the accumulator whose resolution
is 1 pixel for \(\rho\) and 1 degree for \(\theta\). The resulting accumulator is displayed in Figure 3-3. The five
largest peaks are identified by the magenta circles. Finally the five \((\rho, \theta)\) pairs are converted back
to \((x, y)\) locations and drawn as magenta lines in Figure 3-4 to denote the perspective lines.
Figure 3-1: Original Image

Figure 3-2: Canny edge detector run on grayscale version of original image
Figure 3-3: Accumulator of mapped edge pixels

Figure 3-4: Detected line segments from the Hough Transform
3.1.2 Creating Candidate Vanishing Points

Using the five perspective lines found with the Hough transform, candidate locations for the forward vanishing point can be estimated by computing every possible intersection between the perspective lines. Treating the two lines like a system of linear equations the intercept point between the two lines can be found by stacking the equation from (2) into matrix form as in (3) and solving for $[x \ y]^T$. Since we limited the number of perspective lines to five, there is a maximum number of ten potential intercept points assuming no two lines are parallel.

\[
[a_1 \ a_2] \begin{bmatrix} x \\ y \end{bmatrix} = b \quad (2)
\]

\[
Aw = b \quad (3)
\]

Continuing with the example started in Section 3.1.1, Figure 3-5 shows the candidate vanishing points created by the perspective line crossings as cyan crosses. In this image the candidate positions cluster in the center of the current travel lane, however this is not always the case. As we’ll discuss further in Section 3.2, when the Hough transform is unable to identify vanishing lines from one side of the image, whether due to occlusions or poor lighting conditions, only vanishing lines from one side of the roadway are selected and cause the candidate positions to drift towards that side. The consequence of this is that the vanishing point alone is not enough to keep the autonomous vehicle in its current lane. It will require some additional motivation that we discuss in Section 5.3.1.
3.1.3 Localization

A common localization approach for line intersections is the Least-Squares Line Intersection method which given a set of lines minimizes the perpendicular distance from the solution point to all the lines. However, given that the edges of objects in the image may fluctuate from frame to frame due to various lighting conditions and occlusions, the Hough transform will not guarantee the same line segments are consistently selected. This variation in selected perspective lines can induce large jumps in the vanishing point location frame to frame.

To combat the frame to frame jitter of the Least-Squares Line Intersection approach filtering of the vanishing point location is necessary. A Monte Carlo Localization approach was used because of its ability to combine both the localization and filtering aspects in one process. Additionally while this thesis is only concerned with a single vanishing point, the Monte Carlo Tracking (MCT) approach used has the potential to be expanded to localize and track multiple vanishing points of the same image. The single vanishing point we are tracking provides us with
the direction the roadway is headed and hence the location the autonomous vehicle is interested in traveling towards. The other vanishing points are more useful for augmented reality type applications as they provide orientation clues for placing a virtual building in alignment with the reset of the scene.

The MCT approach is a particle filter localization method that begins by uniformly sampling the search space with state estimates called particles. As candidate vanishing point locations are provided, the particles begin converging towards the vanishing point using recursive Bayesian estimation. Spurious candidate points are effectively filtered out as they do not maintain a presence for multiple frames and are unable to draw a significant number of particles toward it. Since information exists about how the camera is going to move from frame to frame by virtue of it being rigidly mounted to the autonomous vehicle, the particles can be adjusted through a control term to account for this motion, increasing the likelihood the particles will be where the measurements are predicted to show up.

The MCT is governed by two equations as described by Dellaert et al. [27]. The first is the Predictive PDF shown in (4). This updates the particle locations based on motion of the camera to the highway scene. The current state estimate is denoted by $x_k$ and the previous state estimate is denoted by $x_{k-1}$. Known control inputs, denoted as $u_{k-1}$, can be known heading changes between tracker updates which can be converted into expected movement of the particle estimates. All the measurements are denoted by $Z^k = \{z_k, i = 1..k\}$, and represent the vanishing line intersections which are our measurements of where the vanishing points lie.

In cases where a sharp turn is being made by the vehicle the vanishing point can drastically shift its location in the image. However by using a combination of high frame rates (i.e. greater than 2 fps) and slow turn rates (i.e. no hard turns at fast speeds for comfort of the passengers), the vanishing point’s motion in the image will be minimal from frame to frame, allowing the motion model, $p(x_k|x_{k-1}, u_{k-1})$, in the Predictive PDF to be ignored and set to 1.
The second equation governing the MCT is the Posterior PDF (5) which is used to update the particles’ positions with the candidate position using a measurement model. Our chosen measurement model was the Gaussian distribution in (6). Particles that are closer to the candidate position are given a higher weight so that during the resampling phase new particles are more likely to be drawn from the locations of the higher weighted particles.

\[
p(x_k|Z^{k-1}) = \int p(x_k|x_{k-1}, u_{k-1})p(x_{k-1}|Z^{k-1}) \, dx_{k-1} \tag{4}
\]

\[
p(x_k|Z^k) = \frac{p(z_k|x_k)p(x_k|Z^{k-1})}{p(z_k|Z^{k-1})} \tag{5}
\]

\[
p(z_k|x_k) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left( -\frac{1}{2}(x_k - z_k)^T \Sigma^{-1} (x_k - z_k) \right) \tag{6}
\]

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.

c. \(p(z_k|x_k)\): is the measurement model. This is probability that the measurement comes from the robots estimated new location.

d. \(p(x_k|Z^{k-1})\): the predictive PDF of (1).

e. \(p(z_k|Z^{k-1})\): is the measurement likelihood that the new measurement is probable given the old set of measurements.

For our MCT, we initialize by uniformly distributing 1000 particles over the entire image. Because of the high frame rate limiting the expected movement of the vanishing point, we were able to skip the predictive step of the MCT. The next step is the update phase which compares the particle states to the candidate vanishing point locations. Particles closer to the candidates are weighted as more important. The last step is to resample the particles states according to the new distribution defined by the current weighting of the particles.
Using the mean of all the active particles an estimate of the actual vanishing point is realized. Since the MCT is initialized with a uniform distribution and updates every video frame by taking small steps for particle movement, it took an average of 1.3 seconds (40 frames) to get the majority of the sample points to converge bringing the location variance to be within our pixel tolerance for the vanishing point location. This was for a video sequence with a frame size of 960x540. The time it takes for the tracker to converge can vary with the size of the image and the number of samples.

Figure 3-6 shows a sequence of images of the MCT localizing on the vanishing point over several frames. The magenta lines are the perspective lines, the cyan crosses are the candidate vanishing point locations for that frame and the yellow dots are the current state estimates of the particles. At $t = 0$ the particles are uniformly distributed over the image. At $t = 0.3$ the particles have made a significant shift toward the cluster of candidates near the center of the image. At $t = 0.966$ the particles are tightly clustered at the vanishing point.
3.2 VP Tracker Performance

To assess the performance of the Vanishing Point Tracker we hand labeled the vanishing point for each frame of several video sequences. Video sequences were used since the Monte Carlo Tracker (MCT) makes use of the intersections between the vanishing lines created by the Hough transform of the image. If the same image was constantly used then the same vanishing lines would be chosen and the MCT would only be able to localize the general region of those vanishing line intersections instead of localizing the vanishing point with any precision. Having a sequence of frames allows for a larger variation of vanishing lines to be found and the consistency of line intersection locations between frames becomes a useful indicator that a vanishing point is more likely to exist there.
We obtained three of these video sequences with a commercial car dash cam mounted inside our vehicle on the center of the front windshield while the fourth video was a sequence of city driving taken from the work of Fauqueur et al. [28]. These video sequences consisted of a nominal straight three lane highway, a two lane highway with a slight curve, a single lane road in a city and a two lane road during rainy weather. The dash cam videos had an image size of 960x540 and the Fauqueur video had an image size of 480x360.

We compared the labeled vanishing point location with the resulting pixel location from the MCT. A plot of the overall pixel error for each data set is shown in Figure 3-7. The error is the distance between the MCT estimate and the ground truth labeling. The MCT settled into a steady state error within 50 frames (~1.67 seconds at 30 fps video) for each of the data sets. The straight highway scenario settled the quickest followed by the curved highway, then the city driving scene in the Fauqueur data set and finally the rainy data set which required the most time to localize the vanishing point.
In both the rainy and curved roadway sequences we noted much higher estimate variance as compared to the straight highway sequence. We found that in these sequences the Hough Transform would tend to select vanishing lines that were primarily from one side of the image. In the curved roadway case, the curve in the road was to the right but the selected vanishing lines were primarily on the left side where the curve on the roadway edge was less perceptible than the curve on the right side of the roadway edge. This caused the MCT estimate to drift towards the left side where most of the line intersections were occurring. This effect can be noticed in the X Error plot of Figure 3-8. The orange red line denoting the curved highway video remains negative throughout the sequence implying that the vanishing point estimate is to the left of the ground truth labeled point.
For the rainy sequence, the vehicle starts in the left lane and has a hard time detecting vanishing lines on the right side of the image. This is partially due to the wet roadway surface being more reflective causing the roadway edges to become blurred and partially due to the video sequence taking place in a city scene where objects such as overhead powerlines and traffic lights generate stronger edge lines that are chosen over the roadway edges resulting in less useful vanishing lines. The purple line in the X Error subplot of Figure 3-8 indicates for the first half of the video the MCT estimates were to the left of the ground truth labeled point and then halfway through the video sequence the estimate becomes biased towards to the right of ground truth. This coincides with the vehicle starting in the left lane and seeing overhead powerlines that had stronger edge lines than the right edge of the roadway. Halfway through the video sequence, the vehicle does a lane change to the right lane and the adjacent vehicles obstruct the view of the left roadway edges. Towards the end of the video sequence, the guardrail to the right of the vehicle generates the majority of the vanishing lines and draws the MCT estimate to the right of the ground truth labeled point increasing the overall error.

The Y Error subplot in Figure 3-8 shows that the MCT estimates were very consistent in the vertical direction once steady state was achieved. The initial large error for the Fauqueur video sequence stems from the fact that the video starts with the vehicle making a turn and having to go up a hill, placing the vanishing point in the upper right corner of the image well away from the center of the image where the MCT tracker is initialized. The tracker is initialized towards the center because of the uniform initialization of the particles over the entire image.
Unfortunately we were unable to conclude if our vanishing point estimator is able to achieve angle errors under the desired 6.3°. This stems from the fact that our ground truth labeling only provides a pixel location in the image and has insufficient information to convert this point into an angle. We reviewed the MCT estimates for each frame of the four video sequences. For a majority of the frames during the steady state periods, the MCT estimates stayed between the lane markings as the error plots above indicated. For the other frames of the steady state periods the MCT estimates would exceed the lane markings of the current travel lane by more than half which would indicate that the angular error would be nearly double of our desired threshold. These large cross range errors tended to be around frames where one side of the roadway was occluded by adjacent vehicles.
While our vanishing point estimator may not have satisfied our 6.3º angular threshold for vanishing point localization, it does a reasonable job of localizing the region in the image where the vanishing point is located. As we found out during our closed loop simulations there was little need for high precision of where the destination anchor point was located. Since it is the first half of the identified motion solution that gets used to generate the steering solution, the second half of the motion solution is used to help ensure continuity in a travel path from frame to frame of the video sequences.
Chapter 4  Inverse Perspective Mapping (IPM)

Because of the perspective effect, any path planned in the original camera image would not generate uniform vehicle turn commands along the path due to the difference in information density of pixels in the middle of the image versus those at the bottom. The idea of information density of pixels relates to how objects at further ranges occupy fewer pixels than they would occupy if they were located closer to the camera. Looking at Figure 4-1 there are two cars ahead of the camera a white and a dark gray. In the real world these two cars take up about the same space. However, because of the perspective effect the dark gray car which is further away is only taking up about half the pixels of the white car. The same effect can be seen with the dashed lane markers the ones closer to the camera appear longer and use more pixels than the ones further away. Due to this effect, locations closer to the vanishing point can be said to be more informationally dense than locations closer to the camera. Therefore a planned solution in the perspective frame would not generate uniform steering commands as the vehicle progressed along the path towards the vanishing point.

In order to make the information density more uniformly distributed among all the pixels a geometrical transformation called Inverse Perspective Mapping (IPM) proposed by Bertozzi and Broggi [11] was used. This geometrical transformation has the effect of making it appear as though the scene has changed from the perspective view as shown in Figure 4-1 into a top down view as seen in Figure 4-2. Comparing the lane markings in the latter figure, it can be seen that the dashed lane markings are now more uniform in length as compared to the their counterparts in Figure 4-1. These lane markings are also now parallel with one another instead of converging towards the vanishing point.
Figure 4-1: Example perspective image to be transformed

Figure 4-2: Example from Figure 4-1 after IPM transformed image applied
4.1 Background

By performing the IPM transformation and knowing other information about the camera optics, the nonlinearity in distances are removed allowing for simple calculations of distances to objects as demonstrated by Tuohy [15]. This allows autonomous vehicles to know the relative position of nearby obstacles which is useful for navigating around those obstacles and enforcing physical constraints, such as minimum turn radii and trailing distances to the potential navigation solutions.

An interesting side effect of the IPM transformation is that any object that has any extent up off the ground plane becomes distorted by spreading vertically in the transformed image. The closer the object is to the horizon the more the object gets distorted by the transformation. In addition the object appears to extend to infinity because it occludes the region behind the object from the camera. So when the transform is filling in the occluded roadway region, it uses the colors of the object instead. By identifying the areas with this distortion, the autonomous vehicle can use this as an additional clue to localize obstacles on the road that need to be avoided.

Bertozzi and Broggi [11] used a flat road hypothesis in their stereo vision setup to identify overlapping regions between the image pairs as a generic obstacle detector. An alternative approach taken by Tuohy [15] was to use the IPM geometric relationship to map the height of high intensity pixels in the IPM transformed intensity map into a distance to non-road surface objects.

4.2 IPM and Obstacle Detection

Our approach to obstacle detection expands upon Tuohy’s approach [15] by developing a streak detector to classify non-road surface objects as obstacles instead of relying solely on the
high intensity pixels from the IPM transformed intensity map. If we were to simply rely on the high intensity pixels as our classifier for obstacles, then any pixel whose color does not match the selected sample region gets called an obstacle. This includes things painted on the roadway such as lane markings. While it’s good to know where the lane markings are, they are not obstacles in the sense that the vehicle cannot traverse over them. To prevent the motion planning algorithm from trying to weave in between lane markings, it is desirable to exclude them from being classified as an obstacle in order to generate smoother paths.

Our streak detector utilizes the inherent distortions of the objects created by the IPM transformation. Since obstacles occlude the regions behind them relative to the camera, the IPM transformation populates those regions with the obstacle instead because of the “flat road” assumption. The result is that the IPM transform populates those regions behind the obstacle with the colors of the obstacles making it appear as though the obstacle is being stretched to infinity.

### 4.2.1 Overview of road removal algorithm

The road removal algorithm by Tuohy [15] provides a simple approach for classifying which regions of the image are road surface and hence drivable by the vehicle. A small region in front of the vehicle, as highlighted in magenta in Figure 4-3, is utilized as a representative for all roadway in the image. An average pixel intensity value for each color channel is calculated from this patch. Then a binary image of non-road pixels is created for each color channel by setting a pixel to 1 everywhere the pixel intensity falls outside of a 13.7% range from the average intensity value for that color channel and 0 otherwise. The three resulting binary image are then merged to create a new RGB image as demonstrated in Figure 4-4. In this image all the black pixels are classified as roadway, while each of the colored pixels represents some object which does not fall
within the intensity range for a corresponding color channel. So the cyan regions indicate objects whose colors do not agree with the green and blue intensities of the representative roadway patch.

![Original Image Prior to Road Surface Removal](image)

**Figure 4-3:** Example roadway scene prior to road surface removal

![Touhy Road Surface Subtraction](image)

**Figure 4-4:** Example from Figure 4-3 after road removal algorithm
There are several interesting drawbacks to using this algorithm. These drawbacks are primarily related to the region used to compute the average pixel intensity for the road surface. This region is highlighted by the magenta square in Figure 4-3. The first issue is related to obstacles, primarily cars, which have the same color as the roadway. In these cases a majority of the car is misclassified as roadway. An example of this can be seen by the black patch on the side and back of the truck in the left half of Figure 4-4. Fortunately most vehicles have taillights, glass or create shadows so there will be some small regions where the color delta will prevent it from being classified entirely as roadway. Although it is preferable for the entire obstacle to be identified, these smaller regions can be sufficient to nudge the motion planning solution away from this area.

A second concern with this classification approach is related to choosing the appropriate size for the sample region. If this region is too small then one of the lane markings can make up a significant portion of this region. So during a lane change when the lane markings will pass through this defined region, it can cause only other lane markings to be classified as roadway and treat the actual roadway as a giant obstacle. For this reason some form of memory in a path planner is needed to supplement during the few frames of missing road surface. On the other extreme for sample region sizing, if the region is too large, then the presence of more non-roadway objects is likely which will shift the average intensity for each color channel to be far from the actual roadway color leading to more misclassifications all around.

Another interesting difficulty with the Touhy classification approach is the case of a wet road surface. When a road surface is wet it becomes slightly reflective so that cars, tail lights, traffic lights, etc. are reflected on the wet roadway. These color distortions allow the algorithm to misclassify the reflected lights on the road surface as obstacles because they have changed the road surface color pixel value. This would cause the motion planning algorithm to create complicated routes trying to avoid obstacles that don’t exist. Examples of such misclassifications
are demonstrated in Figure 4-5. The upper subplot shows the original image of the wet roadway. The lower subplot shows the result after the Touhy road surface subtraction is applied. In reality there are no obstacles located before the traffic light. It is just the reflections of the green light on the roadway and hood of the car that is being stretched and reflected creating regions that are classified as obstacles instead of remaining black and being called roadway.

**Figure 4-5: Wet road surfaces causing misclassification of non-road surfaces**
Despite these drawbacks we found that the path planning algorithm was able to generate reasonable motion paths, although perhaps not optimal paths. The spotty coverage of roadway colored obstacles still helped define an overall region that needed to be avoided. We found that using a sampling region close to the vehicle that was about as wide as a travel lane and about as long as a dashed lane marking allowed the majority of the sample region to be filled primarily with roadway colored pixels.

### 4.2.2 Overview of IPM algorithm

The Inverse Perspective Mapping (IPM) Algorithm transforms the perspective image of the camera into a view commonly referred to as a top down view or a Bird’s Eye View (BEV). By using the camera properties or a calibration procedure, a mapping is created that provides the ability to determine the range to an object on the ground plane. This provides useful information when trying to determine a path through the scene.

The IPM algorithm maps the perspective image into a world-coordinate system through equations (7), (8) and (9) as derived by Bertozzi et al. [11]. The viewing direction is defined as two separate angles $\bar{\theta}$ & $\bar{\gamma}$. The $\bar{\theta}$ parameter is the camera angle with respect to the horizon while the $\bar{\gamma}$ parameter is the camera angle with respect to the forward direction. The $\alpha$ parameter denotes the camera’s aperture which describes the camera’s field of view. The camera’s resolution is represented as $m \times n$ where $m$ is the number of rows and $n$ is the number of columns in the image.

The $(l, d, h)$ parameters represent the camera’s position with respect to the world-coordinate system. Typically these parameters can be physically measured but they can also be estimated by placing a known calibration pattern, such as a uniform grid, in front of the camera and solving for the $(l, d, h)$ parameters using the image coordinates and known pattern spacing.
Because the camera is rigidly mounted to the autonomous vehicle, these camera parameters are able to remain constant allowing all the mappings, interpolation weights, and indices to be saved to reduce real-time computation. To generate the interpolation weights and indices for the IPM mapping, we used the barycentric local coordinates approach. Each of the original image pixels locations are mapped to the IPM frame using equations (7), (8) and (9). Then a new grid of points representing the IPM image we wish to generate is overlaid on those transformed points. Barycentric interpolation takes the three closest transformed pixels to the new grid pixels and determines what percentage each of the transformed pixels will contribute based on the area of the triangle formed by those three closest points. This approach was used instead of bilinear approaches due to the non-square grid nature of the transformed pixels making it difficult to select four suitable neighbors.

To demonstrate the performance of the IPM transformation to an actual BEV image, Figure 4-6 shows a scene using the perspective view from a simulator we developed to perform closed loop testing of our algorithms. Using the same scene from Figure 4-6, Figure 4-7 is taken
with a BEV perspective for comparison to Figure 4-8 which is the IPM transformed version of Figure 4-6.

The most noticeable difference in the IPM image versus the BEV image is that the red objects representing other vehicles are greatly distorted in the IPM image. The IPM algorithm is using a “flat road” hypothesis which makes the assumption that everything in front of the camera is lying flat on, or painted on the road and not extending upwards. However, since the red obstacles do in fact extend out of the ground, occluding the scenery behind it, the algorithm is tricked into thinking that is what the ground looks like at those farther ranges. It is precisely this artifact that we make use of for our obstacle detector in the next section.

![Simulated Perspective Image](image)

Figure 4-6: Simulated perspective image
Figure 4-7: Simulated BEV image of scene in Figure 4-6

Figure 4-8: IPM Transformed image of Figure 4-6 for comparison to Figure 4-7
Applying the IPM transformation to a perspective image from the real world in Figure 4-3 we get Figure 4-9. Again we can see the vertical streaking caused by objects that extend out of the ground plane. But, in general the transformed image gives the correct impression that that is how the scene would appear if it were from the bird’s eye view. Another interesting artifact that can be noticed in the transformed image is that at the top of the image it appears more pixelated and blurry. This is because there is not enough pixel resolution in the original image near the vanishing point to properly reconstruct this section of the scene as detailed as the sections that are closer to the camera.

Figure 4-9: IPM transformed image of Figure 4-3
Using 960x540 sized images we had an increasing amount of pixelization towards the top of the images, and limited the interpolation ranges to a maximum of 300ft down range and ±34ft cross range which covers several lanes of traffic on either side of the autonomous vehicle. The 300ft down range viewing distance creates a maximum autonomous vehicle speed of 82mph when using the 2.5 second reaction time limit as found for the 85th percentile of drivers determined in the work by Wortman and Matthias [29]. Since current driverless vehicle laws require a human driver to be behind the wheel to take over for the vehicle, in order to increase the maximum vehicle speed for the autonomous vehicle, an increase in the down range viewing distance would be needed. To increase the down range viewing distance, higher resolution images can be used to improve the resolution at further viewing distances. However the increased number of pixels will require more processing power to be classified and require more processing power.

4.2.3 Obstacle Detection (Streak Detector)

The simple classification of roadway pixels by the road removal algorithm in section 4.2.1, while effective, calls every object that is not the color of roadway an obstacle which is not quite true in some cases. As can be seen in Figure 4-4 the lane markers are not marked as roadway due to their bright white color. However the autonomous vehicle can certainly drive over these lane markings if a lane change or quick avoidance maneuver is needed. This required us to develop an obstacle detector that sorts the non-road surface pixels into objects that follow the “flat road” hypothesis and those that do not.

The obstacles that do not follow the “flat road” hypothesis are typically other vehicles, road signs, guard rails, etc. They have the tell-tale feature of being streaked across the IPM image. This streaking artifact is best seen when observing how the red boxes in Figure 4-6
become long slightly trapezoidal red streaks in Figure 4-8. This is the primary feature used by our obstacle detector.

Our “Streak Detector” starts by converting the IPM transformed road removal image into a binary image by taking a logical OR of the three channels of the IPM road removal image. This identifies all pixel locations that are not roadway. Using 8-connectivity connected-component labeling a binary image is created. These connected components represent the individual potential obstacles.

For each of these individual potential obstacles we use the dimension of its bounding box to compare the height-to-width ratio. Streaked obstacles have a higher height-to-width ratio since the obstacle occludes the roadway behind it causing the IPM transformation to stretch the obstacle into that space instead of flat road behind it. We found that a 3:1 ratio for height-to-width gave the best performance. A minimum width constraint of 30 pixels (corresponding to the 6 inches for the lane marker width) is also used to check if the potential obstacle is really a lane marker or an object sticking up out of the ground plane. This pixel amount will vary depending on image size and camera properties used in the IPM transformation.

<table>
<thead>
<tr>
<th>ALGORITHM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Take logical AND of IPM road removal image</td>
</tr>
<tr>
<td>Label the connected components of newly created binary image</td>
</tr>
<tr>
<td>Measure the bounding box around each connected component</td>
</tr>
<tr>
<td>IF ( bounding box is at least 3x taller than wider AND bounding box has minimum width of 30 pixels ) THEN</td>
</tr>
<tr>
<td>Mark connected component as a non-flat obstacle</td>
</tr>
<tr>
<td>ENDIF</td>
</tr>
</tbody>
</table>
Applying the streak detector algorithm to Figure 4-10 we can see in Figure 4-11 that the short dashed lane markers are no longer considered non-roadway. All the grass on either side of the roadway and the car in front of the camera retain the classification as a non-flat obstacle. Interestingly the right most solid lane marker, circled in yellow, maintains the non-flat obstacle classification as well since the slight slant causes the width of the bounding box to grow beyond the threshold. This slight slant is a side effect of not using finely tuned camera properties with the IPM transformation. However, since this lane marker is near the edge of the road surface and solid lane markers are used to indicate to the driver to not cross, the detector’s decision to classify it as a non-flat obstacle, while not syntactically correct, becomes a useful classification result for the purposes of the motion planning that will be discussed in the next chapter.

Figure 4-10: All obstacle detections overlaid on IPM image
4.3 Obstacle Detection Performance

To assess performance of our obstacle detector we ran our algorithm against two data sets. The first was a data set we created using the closed loop simulator we explain in Section 5.2. Since it is a simulated data set we are able to easily extract the ground truth labeling of each object in the image. The second data set we used is the labeled roadway images used in work of Fauqueur et al. [28]. Although this video sequence is primarily a roadway scene through a town, the labeled data set clearly identifies each pixel as belonging to either roadway, pedestrians, cars, buildings, lane markings, etc. Of the available labels we grouped the “Road” and “LaneMkgsDriv” labels into a non-obstacle group and all other labels into an obstacle to avoid group.
Since our algorithm performs the classification in the IPM transformed frame we were required to transform the labeled image into the IPM transform using the nearest neighbor approach to interpolation instead of the barycentric interpolation described in Section 4.2.2. This is to avoid the blending together of multiple classes that would have occurred from the interpolation step. While this does pixelate the boundaries of the classification edges, these are a small enough percentage of the overall image that the pixilation can be ignored.

### 4.3.1 Performance against the simulated data set

This data set was generated by the closed loop simulator we designed to test the overall performance of the combined motion planning solution. A more detailed explanation of the simulator is provided in Section 5.2. The scenario is a vehicle traveling down a three lane highway with red boxes representing other vehicles in the two adjacent lanes. The corresponding ground truth labeled video sequence was auto generated by turning the light sources off so that the colors on all the surface objects became flat. Then the RGB color space was converted to indices and saved as the ground truth labels.

For each video frame we compared the obstacle detector output to the ground truth labels by computing the true and false positives and true and false negatives. From there we were able to compute several performance factors. The three performance metrics used were precision, recall and accuracy. The precision metric is corresponds to how many of the pixels the detector called obstacles were in fact ground truth labeled obstacles. The recall metric denotes how much of the ground truth labeled obstacles the obstacle detector was able to identify. The accuracy metric measures how many of the pixels were correctly labeled. For the simulated data set we achieved a precision of 75.1% with a recall score of 68.1% and an overall accuracy of 70.7%.
4.3.2 Performance against the Fauqueur data set

This data set contained 101 images, each with a ground truth labeled counterpart. As with the simulated data set, we ran our obstacle detection algorithm on each image and compared the ground truth labeling to the obstacle detector output. The resulting precision score was 39.3% with a recall score of 59.1% and an overall accuracy of 31.6%.

A lot of the low scoring stems from the decision to use the Tuohy road surface removal algorithm as the basis for the obstacle detector. Non-uniform lighting conditions create patches in the roadway where the color is lightened or darkened well beyond the intensity threshold of the sample region being used to identify the color of the roadway. Additionally, the obstacles were a lot more patchy and had a significant number of holes. However, after applying a guassian filter before proceeding with the roadway substraction the performance nearly doubled so that the precision score became 59.1% the recall score became 94.4% and an overall accuracy of 64.5%. This large performance improvement seems to come from the smoothing filters ability to flatten out the non-uniform lighting conditions throughout the image so that the road surface removal algorithm will result in more uniform patches of road surface instead of individual pixels speckled throughout the image.

While the coverage of the entire obstacle is not always achieved using this approach, this does not seem to have a major effect during the motion planning stage. The reason for this is because the pixels closer to the camera are more likely to be classified correctly due to less color distortion and being closer to the roadway sampling region. Additionally, the extent of the obstacles influence is quite large, so small patches covering the obstacle effectively fill in the entire region of the obstacle and guides the motion planning solution away from the obstacle.
Chapter 5  Motion Planning with Active Contours

5.1  Proposed Motion Planning Algorithm

We present our active contour approach in which the IPM transformed image from Chapter 4 is used to create an energy function. The obstacles detected by the streak detector in Chapter 4 are used to define regions on the energy map for the active contour to avoid. The focal vanishing point found in Chapter 3 is used as a destination anchor point for the active contour. The other anchor point is chosen as the bottom center pixel which represents the camera position in the IPM frame.

5.1.1  Background on Active Contours (Snakes)

Active contours have more commonly been used for image segmentation in biomedical images. It is a deformable spline whose movement is influenced by minimizing an energy function through gradient descent that is most commonly generated by edge features from the image. These edges can be used either as an attractive or repulsive force. The moniker of “Snakes” has been given to active contours due to their resemblance to snakes slithering as the forces are applied to the deformable spline.
With traditional snakes the energy function is composed of an internal and external term as shown in Equation (10). The internal energy term shown in Equation (11) relates to the contour itself, controlling the tension between points and the rigidity or smoothness of the curve with the $\alpha$ and $\beta$ parameters respectively. The $\gamma$ term represents the contribution of the internal energy to the overall energy and is referred to as the viscosity parameter. The contribution of the external energy to the overall energy is represented by the $\kappa$ term. The external energy term is generated by the image either through the grayscale values or image gradients as shown in Equations (12) and (13) respectively. Smoothing of the image can used to reduce the noise within the image before conversion into the external energy term as in Equations (14) and (15). This has the effect of blurring the boundary and increases the ranges from the boundary where the snake would be affected at the expense of losing definition of that boundary edge.

$$E = \int \gamma E_{\text{int}}(s) + \kappa E_{\text{ext}}(s) ds$$  \hspace{1cm} (10)

$$E_{\text{int}} = \frac{1}{2} (\alpha |x'(s)|^2 + \beta |x''(s)|^2)$$  \hspace{1cm} (11)

$$E_{\text{ext}} = I(x,y)$$  \hspace{1cm} (12)

$$E_{\text{ext}} = -|\nabla I(x,y)|^2$$  \hspace{1cm} (13)

$$E_{\text{ext}} = G_{\sigma}(x,y) * I(x,y)$$  \hspace{1cm} (14)

$$E_{\text{ext}} = -|G_{\sigma}(x,y) * I(x,y)|^2$$  \hspace{1cm} (15)

An issue with traditional snakes is non-convergence to concave regions. Figure 5-1 was used by Xu and Prince [24] in their paper to describe this problem. In the leftmost subplot the active contour is initialized around the U-shaped object. There are several more curves superimposed showing the sequence of steps used to conform to the object shape. The middle subplot shows the potential force field used to influence the snake. The difficulty of the U-shaped
object is the concave region. As the snake progresses into the contour, it is forced to split across the boundary due to the opposing forces trying to pull the snake towards the edges. A close-up of these forces is shown in the rightmost subplot. The region where the snake stops progressing into the concave region is where the force vectors are equally opposing each other preventing the snake from stepping further into the concave region as desired.

Figure 5-1: Demonstrating convergence issues of traditional snakes. Source: Xu and Prince [24].

Xu and Prince [24] proposed a Gradient Vector Flow (GVF) to replace the external energy term. The GVF field is used to introduce a new non-irrotational external force field that points to the object boundary when very near to that boundary and greatly extends into the previously homogeneous image regions by filling it in with information taken from the boundaries. The net effect is that the snake can now be captured from longer ranges and is more easily forced into concave regions. Equation (16) is the GVF vector field. Equation (17) describes the energy function that is minimized by the vector field in (16). The \( f \) term in (17) is the edge
map as derived from the image. This can be any of the four original external energy equations found in equations (12) through (15).

\[
\mathbf{v}(x, y) = (u(x, y), v(x, y))
\]  

\[
E = \iint \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla f|^2|\mathbf{v} - \nabla f|^2 dxdy
\]  

5.1.2 Abstraction as binary labyrinth

We first abstracted the problem of motion planning in a roadway to solving a binary labyrinth. The walls of the labyrinth were used to represent the obstacles such as vehicles and guard rails. The labyrinth had a single entry and exit point representing the vehicle’s current location and its desired destination. The snake is initialized as a straight line between these two points as in Figure 5-2. The \( \alpha \) and \( \beta \) parameters were initially set to 0.3 and 0 respectively to allow the contour to be easily deformable, allowing it to wrap around the obstacles more easily. The resulting snake solution would center the snake along the labyrinth pathways while taking perpendicular shortcuts through walls as in Figure 5-3. These shortcuts are due to the snake being initialized across a wall and encountering a similar problem as the concave region where the gradient vectors are equally opposing one another preventing further motion.
Figure 5-2: Initialization of active contour over binary labyrinth

Figure 5-3: Figure of the active contour taking a shortcut through a labyrinth wall.
To motivate the snake to slide off the wall and towards the goal point, a weighted slope (similar to that of Figure 5-4) was applied to the gradient vectors. This did create a more direct path between the start point and the end point; however, the contour did not remain centered between the walls and instead was more likely to hug the wall edge. The repulsive forces of the walls prevented the snake from being able to cross the wall segment and find a valid solution as seen in Figure 5-5.

Figure 5-4: Weighted sloped applied over image area to add motivation towards the endpoint.
Figure 5-5: Active contour solution with weighted slope applied.

In order to overcome the repulsive force of the wall the weighted slope was only applied to the labyrinth walls instead of being broadly applied to every point. Figure 5-6 shows how the gradient vector field of the wall corner has now been replaced by the vector field of the weighted slope at all points along the walls. This allowed the snake to slip off corner boundaries as shown in Figure 5-7. The stalemate between equally opposing gradient forces was broken by replacing those forces with the weighted slope allowing the contour to continue to move and find a way around the wall instead of through it.
Figure 5-6: Close up of vector field for labyrinth wall replaced with weighted slope

Figure 5-7: Weighted wall pixels allow active contour to slip over wall segment
Unfortunately simply applying the weighted slope to wall segments still failed in cases the snake was required to backtrack in order to travel around a wall segment that greatly extended from the initialized straight line snake. Because the weighted slope is pointing only in the direction of the goal, there is little to no gradient forces motivating the snake to travel perpendicular to the wall in order to travel around it. In Figure 5-8 we show the resulting solution when there is not enough upward force to get the snake to travel around the vertical wall. In Figure 5-9, we provide another example where the snake is required to backtrack both upwards and to the right but the forces at the wall are not significant enough to overcome the overall forces drawing the snake towards the intersection of walls instead.

Figure 5-8: Weighted wall pixels are insufficient to force contour off of wall segment
A common adage for solving labyrinths is to follow along one side of the walls of the maze and eventually you can make your way through the labyrinth. This approach is referred to as a wall follower and only applies to simply connected mazes which have all their walls connected together or connected to the entrances. Since we are using simple small scale labyrinths which are simply connected we took inspiration from the wall follower rule and applied a clockwise rotation to the gradient function around the labyrinth walls in addition to the weighted slope on the wall themselves. The idea behind this rotation is to induce the snake to travel along the labyrinth wall edges in a similar manner to a wall follower solution.

To create the clockwise rotation, we took the pixel difference in the x and y directions of the original binary image. If the x difference was positive, the y gradient was set to +1 and set to -1 if the x difference was negative. Similarly for the y difference, if it was positive, the x gradient was set to -1 and +1 if the y difference was negative. Flipping of the signs would result in a counter-clockwise rotation. Figure 5-10 depicts the modified gradient field with a clockwise rotation around the walls. As can be seen the top edges of the walls have a rightward direction,
the left wall edge travels downward, the right wall edge travels upward and the bottom wall edge travels to the left. We also found it necessary to make the vectors a uniform magnitude. This allowed the snake to better center itself in the channel between walls because the forces from the wall now reached further and it also reduced the number of iterations for the snake to converge to a solution since larger steps are taken.

This clockwise rotation provided the missing motivation to the snake to backtrack around the obstructing wall segments as shown in Figure 5-11 & Figure 5-12. These images correspond to the same labyrinths from Figure 5-8 & Figure 5-9 respectively where without the rotation applied to wall edges, the solutions were stuck in locations were the vector fields were pulling equally in opposite directions. Figure 5-13 shows that the addition of the wall edge rotation does not impact the solution for simple cases.
Figure 5-11: Clockwise rotation along wall edges allowing solution to backtrack around wall segment.

Figure 5-12: Another example of a clockwise rotation along wall edges allowing solution to backtrack around wall segment.
However, there are still points of failure as seen in Figure 5-14. This failure here is that the weighted slope applied to the walls is pushing the snake into regions of the labyrinth where the clockwise wall edge rotation is pushing the snake in the opposite direction, thus effectively pinning the snake by creating equally opposing vectors. By changing the direction of the weighted slope to be from the goal to the starting point, the snake is able to break the stalemate between those equally opposing regions and solve the labyrinth as shown in Figure 5-15.
Figure 5-14: Clockwise wall edge rotation applied to Counterclockwise labyrinth

Figure 5-15: Reversing the weighted slope with a clockwise wall edge rotation

Yet again simply changing the direction of the weighted slope is not sufficient for all cases. Certain labyrinths will also prefer a counterclockwise rotation along the wall edges instead.
of a clockwise one. In Figure 5-16, we show a labyrinth whose solution requires a counterclockwise wall edge rotation with a weighted slope from the starting point to the ending point, and Figure 5-17 shows a solution with a counterclockwise wall edge rotation with a weighted slope from the ending point to the starting point.

Figure 5-16: Counterclockwise wall edge rotation applied to Counterclockwise labyrinth

Figure 5-17: Reversing weighted slope and counterclockwise wall edge rotation
Therefore to find a valid solution for the binary labyrinth problem we propose that all four combinations of weighted slope direction and rotation direction are run in parallel and whichever results in the shortest valid path with no wall crossings should be selected as the winning solution. For a majority of the mazes our selection of $\alpha = 0.05$ and $\beta = 0.001$ for a 150 point snake and 2000 iterations performed reasonably well. However, there were instances where the $\alpha$ and $\beta$ parameters needed to be adjusted to allow for less tension and rigidity between points along the contour making it easier for the contour to deform round more complicated wall patterns. These more involved patterns sometimes also required that the number of iterations the active contour algorithm is performed to be increased. While our chosen parameters do not guarantee a solution for all labyrinths, it had sufficient success with the simpler labyrinths which are more representative of highways scenes transformed into the IPM frame that we felt the active contour approach was worth pursuing.

5.1.3 Applying active contour motion planning to IPM images

Using the active contour concepts for binary labyrinths explored in Section 5.1.2, we applied them to the IPM images with identified obstacles from Section 4.2.3. The IPM obstacles were treated similarly to the labyrinth walls. The starting location was the bottom center pixel of the IPM image. This represents the closest point to the camera location and the centerline of the vehicle. The destination location is the vanishing point pixel from Section 3.1.3 mapped into the IPM frame. The starting point was fixed to ensure the solution can be traversed by the vehicle from its current position.

The conversion of the IPM obstacle image to the edge map used in the GVF snake algorithm simply creates an image where a 1 is used if the pixel belongs to an obstacle and a 0
otherwise. An example of this conversion is shown in Figure 5-18. The left subplot show the IPM image with the obstacle detections highlighted in red. On the right is a binary image where the white are the pixels belonging to obstacles and the black are drivable pixels. The red line bisecting the image down the center represents the initialized snake which goes from the camera to the vanishing point.

In Section 2.4.2, we discussed the case of two vehicles driving side by side and how a path planning algorithm might try to set a route that travels in between the two vehicles. While this may intimidate or unnerve most passengers due to close proximity of nearby fast moving vehicles, the computer is not so adversely affected. As explained earlier, the common technique is
to expand the size of the obstacle by a buffer amount that ensured a safe and comfortable separation between the autonomous vehicle and the other vehicles as it is passing. For this reason we used an expansion buffer size of 17 pixels which represents approximately half of a vehicle's width given our parameters for IPM transformation and it’s pixel to distance mapping. This also expanded the vehicle obstacles to be approximately the full width of the travel lanes so that vehicles traveling adjacent to one another would appear as a large horizontal obstacle. We demonstrate the adjacent obstacle example in Figure 5-19 by generating the scene in the simulator we developed and explain in Section 5.2. The left subplot shows the original obstacles classified in the IPM frame and the right subplot is after the 17 pixel expansion buffer is applied to the obstacles. The obstacles now occupy both travel lanes removing the gap between the adjacent vehicles preventing the motion planning algorithm from trying to find a route in between them.

Figure 5-19: Expansion buffer applied to side by side obstacles
As we discussed in Section 5.1.1, there are several parameters that affect the performance of the active contour. The active contour parameters $\alpha$, $\beta$, $\gamma$, and $\kappa$ parameters control the tension, rigidity, internal and external forces respectively. We found that for the labyrinth problem we required very small values ($\alpha = 0.05$, $\beta = 0.001$, $\gamma = 1.0$ and $\kappa = 0.96$) in order to allow the contour to deform and backtrack on itself. However in our simulation environment and the real world video sequences we found we needed to have larger values, ($\alpha = 1.0$, $\beta = 0.25$, $\gamma = 0.25$ and $\kappa = 5.0$), to generate smoother paths towards the endpoint.

In Section 5.1.2 we explained our development modifications to the gradient vector field. One modification was to add a weighted slope to the labyrinth walls to point towards or away from the endpoints. Throughout our development we found that the direction of the weighted slope point had little effect on the final path since the roadway environment did not create as complicated labyrinths which in turn reduced the scenario where the obstacle gradient field created a stalemate forcing the contour to get stuck on an obstacle. We also relaxed the constraint about needing the solution to not pass through an obstacle at far ranges. This is because due to the streaking effect of the IPM transformation obstacles may occlude the vanishing point making no feasible path from the camera endpoint to the destination endpoint. Intersection with obstacles is pushed back as far down range as it can be with an increased effort to make sure no obstacles are within the closest 50 feet to the vehicle.

The other modification to the obstacles gradient field was to add a rotational element around the wall edges which could be either clockwise or counter-clockwise. While we suggested in Section 5.1.2 that both orientations be run in parallel and the shortest path be selected for the solution, in the roadway scenario, we found that the directionality of the rotational vectors around the obstacle edges could be set by which lane the vehicle is currently traveling in. This primarily helped in cases where two obstacles were in adjacent lanes and created a large obstacle. For
example if the autonomous vehicle is in the middle lane of a three car highway with another car in front, a clockwise rotation would entice the autonomous vehicle to plan a route that passes on the left lane. A counter-clockwise rotation entices the autonomous vehicle to pass in the right lane.

A demonstration of the different rotational directions for the obstacle edges is shown in Figure 5-20. Both subplots started with the same initial conditions. The only difference was the directionality of the rotation applied to the gradient vectors around the obstacle edges. The left subplot shows the result with a clockwise rotation and the right subplot shows the result with a counter-clockwise rotation. The resulting path is highlighted in green.
As this algorithm is intended to run against a video sequences there is some minimum number of iterations that the active contour algorithm needs to be run for in order to arrive at a solution. The longer the time delta between frames the more iterations are needed as there is a larger delta between frames and the active contour needs more time to minimize the new energy function. In our simulated environment we used a frame update rate of 3 frames per second and only needed 25 iterations between frames to effectively minimize the active contour to the energy function. This is primarily because the solution does not change much from frame to frame. The real world data sets used frame rates of 30 frames per second. We still used 25 iterations when there was a faster frame rate. This was partially due to the noise of real images creating ragged obstacles and while not optimal we kept the extra iterations since they did not move the solution once the minimum in the energy function was found. For cases were we did not run against a video sequence but were interested in a one off solution, we used 100 iterations to allow the contour more time to search the space and settle at the energy function minimum. The starting solution for each frame was the solution from the previous frame except for the first frame where the solution is initialized as a straight line from the camera endpoint to the destination endpoint.

When using faster frame rates, the algorithm is better able to adapt to slower moving or stopped obstacles. Since the IPM transformation only provides range information about the obstacle some other considerations need to be made to estimate the speed of the surrounding obstacles. Our obstacle classifier provides an instantaneous classification of pixel with no consideration of the previous frames. This makes it difficult to track an obstacle through the frames and estimate the closing speed between it and the autonomous vehicle. However with faster frame rates the amount of pixels an obstacle moves between frames is reduced, effectively slowing the obstacle down relative to the motion planning algorithm. This allows the motion planning algorithm to react gently to faster moving obstacles instead of generating large steering
commands to avoid the fast approaching obstacle when frame rates are slower and the obstacle appears to take large jumps.

5.2 Closed loop simulation of motion planning algorithm

To simulate our motion planning algorithm we developed a closed loop simulation in Matlab. This allowed us to analyze how the motion planning solution evolves as the vehicle travels along the suggested route. An additional benefit of working with the artificial environment is the ability to generate a ground truth labeling of the pixels in the images. This enables us to create data sets to assess the performance of the obstacle detector and of the motion planning algorithms.

We used Matlab’s ability to render perspective images to generate a simulated highway scene using simple 3D shapes with surface objects. Figure 5-21 is an example of the highway scene rendered. This is a 480x640 sized image. We used red cubes to represent cars and other obstacles on the highway, cylinders and cones for trees, and rectangles for lane markers. The highway scene is a mile long stretch of a six lane highway.
For simplicity we modelled a constant speed of 29 m/s or 65 mph, a common speed limit on US highways. We used a constant vehicle speed as we intend this algorithm to be used for obstacle avoidance during cruise control on a highway. The primary objective of our algorithm is to create steering commands to navigate around obstacle while remaining at speed. We recognize that there will be cases where there is no solution that will allow the autonomous vehicle to pass around the obstacles and will require the vehicle to slow down or stop. A potential clue that could be used in future work to determine the need to slow down is if any obstacles are approaching the region around the roadway sampling box used by the obstacle detector. Since this sampling region defines the color attribute of the roadway, it is desired to keep it clear of obstacles. So if obstacles are approaching the sampling region then the autonomous vehicle could be slowed down by some factor relative to the distance between obstacle and sampling region.
To convert the resulting active contour path into a steering command, we created a steering vector by taking the delta between the starting endpoint to a point on the contour approximately 15 meters down range. We chose a look ahead distance of 15 meters as this is roughly three car lengths and provides a good cushion should the vehicle need to plan to come to a complete stop.

The video feed rate in our simulator was set to 3 frames per second with the entire motion planning algorithm executed for each frame. We chose 3 frames per second as this roughly corresponds to how long it would take a vehicle traveling at 100mph to cross the look ahead distance of 15 meters.

The car obstacles are always centered on one of the three right side travel lanes. The selected lane for each car obstacle depends on the scenario being tested. For a majority of the tests a car obstacle is placed in the current travel lane, traveling 5 mph less than the autonomous vehicle, forcing the motion planner to steer into one of the unoccupied lanes.

In order to ensure a contiguous solutions from frame to frame, the snake solution was initialized with the solution from the previous frame with the only modification being to update the vanishing anchor point with the new vanishing point location estimate. This was to prevent cases where from frame to frame the solution would flip flop on the direction it should take to move around an obstacle. We noticed that after the rotational forces were applied to obstacle edges, there was a lot more consistency in the solution such that if it were completely reinitialized for each frame the direction the solution would take would be the same as the previous frame. However, we chose to keep initializing the new solution with the previous frames solution as this reduced the number of iterations that the active contour algorithm needed to be run.
5.3 Motion Planning Performance

We assessed the performance of the motion planning algorithm against both closed loop simulation and real world video sequences. While our algorithm is able to provide and act upon steering commands in the simulated environment, it is a challenge to do the same in the real world due to lack of assets. In order to gauge our motion planning algorithms performance in both the simulated environment as well as for real video sequences, we developed a performance metric that would be applicable for both cases. This entails using a feature calculated from our motion planning result and the ground truth labels.

Our performance metric is computed for each frame by identifying the down range value that our motion planning algorithm was able to reach without encountering an obstacle. These values are referred to as the solution path ranges. We compared these values against the down range value of any encountered obstacle if the autonomous vehicle were to have continued straight ahead. These are referred to as the direct path ranges. We than tabulate the occurrences of the following three cases to gauge performance. The first case is the number of frames where the solution path was able to provide a solution further than the direct path. This indicates that following the solution path will allow the autonomous vehicle to navigate around an obstacle directly ahead. The second case is the number of frames that the solution path resulted in solutions with less distance than the direct path. This indicates that solution path failed and is steering the autonomous vehicle towards an obstacle instead of away from one. The last case is the number of frames that both the solution path and the direct path result in the same distance indicating that the solution path was unable to find a route that would navigate the autonomous vehicle around the upcoming obstacle.
5.3.1 Performance of the closed loop simulation

Using the closed loop simulator described in Section 5.2 we ran our algorithm against several scenarios. These cases were chosen because they represent common driving scenarios in a highway scene. They primarily consist of having to pass slower moving vehicles that are in front of the autonomous vehicle. For each test case we have provided a brief description of the scenario, a brief overview of the path taken by the autonomous vehicle and the score our motion planning algorithm earned against our performance metric.

For our first test, the autonomous vehicle and the obstacle vehicle are starting in the rightmost lane which requires the autonomous vehicle to pass in the left lane. Figure 5-22 shows the results for this test case while Figure 5-23 shows several snapshots from the video sequence. Starting with the upper left subplot in Figure 5-22, this is a view of the initial scenario setup. The autonomous vehicle started 18 meters behind obstacle vehicle. The upper right subplot is a trajectory plot for this test case. The blue line represents the autonomous vehicle, the red line is the obstacle vehicle and the black dashed lines represent the lane markings. From this subplot we can see that the autonomous vehicle immediately starts maneuvering into the middle travel lane. This was due to a combination of the solid lane marking being along the entire right side of the image and the counterclockwise rotation on the obstacle edges in the gradient vector field. Towards the end of the run the autonomous vehicles starts drifting towards the right. This is partially due to the noise in the vanishing point estimator and to the fact that the gradient vector field has no forces that try to align the autonomous vehicle towards the center of a lane. As an idea for future work, we suggest taking an existing lane detector algorithm and converting that result into a gradient field that could be merged with the active contours’ external forces.

The lower left subplot of Figure 5-22 plots the frame by frame analysis of our motion planning algorithms performance using our performance metric. The blue line with circular
markers indicates the solution path which is the unobstructed down range distance achievable if our motion planning solution is used. The red line with crosses is the direct path measurement which indicates the distance to an obstacle if the autonomous vehicle were to continue straight ahead. We can see that for the first two frames the direct path encounters the obstacle vehicle roughly 85 feet ahead. However the solution path shows that for those few frames it is able to identify a route that will navigate the autonomous vehicle around the obstacle vehicle. This suggested path is shown in the leftmost subplot of Figure 5-23. The point where the solution path intersects an obstacle is marked by a little cyan cross. The downrange value of this cross represents the solution path range performance metric. The subplots in Figure 5-23 show the planned path in the IPM frame. The giant red blobs represent all the identified obstacles that need to be navigated around. The green lines represent the motion path that our algorithm has suggested and the cyan vector is the resulting steering command. From left to right these subplots show the sequence of the first test case starting with the initial frame, turning to steer into the left lane, traveling down the left lane, and passing the obstacle vehicle that’s in the right lane.
Figure 5-22: Test Case 1 results plots

Figure 5-23: Test Case 1 sequence plots
The second scenario is quite similar to the first case, both in setup and performance. The difference here is that this time both the autonomous and obstacle vehicle are in the left most travel lane requiring the autonomous vehicle to pass in the right lane. Once again a view of the scenario can be seen in the upper left subplot of Figure 5-24. The upper right subplot shows the trajectory of the autonomous vehicle. This time both vehicles are starting in lane 3. However, now because of the clockwise rotation, due to the autonomous vehicle being so far to the left, and the solid lane markings all along the left side the active contour is pushed towards the right lane instead of trying to move to the left like in the previous test case. In the lower left subplot, the performance metrics are nearly identical to those of the first test case. In the first frame the direct path encounters the obstacle vehicle roughly 85 feet away and the solution path is able to identify a route all the way to the end of the IPM frame. Snapshots of these solution paths are shown in Figure 5-25. Once again from left to right, the order of events are, the first frame, the initial turn towards the middle lane, traveling down the middle lane and passing the obstacle vehicle that is in the left lane.
Figure 5-24: Test Case 2 results plots

Figure 5-25: Test Case 2 sequence plots
The third scenario was chosen to demonstrate the adjacent obstacle case described in Section 2.4.2. In this scenario we have two obstacle vehicles in adjacent travel lanes and are interested in showing that the motion planning algorithm does not try to pass between the obstacle vehicles. The upper left subplot in Figure 5-26 shows how the scenario was initialized with the autonomous vehicle starting 18 meters behind the rightmost obstacle vehicle. The trajectory plot is in the upper right subplot and shows the autonomous vehicle (blue line) starting in lane 1, the leftmost lane, and immediately does a maneuver that takes it to the rightmost lane, lane 3. Although in the trajectory plot it appears the blue line intersects with the red obstacle vehicle line, this does not happen in the simulation since the obstacle vehicle has already moved ahead in time and the red line is just showing the entire position history for that obstacle vehicle.

The lower left subplot has a more interesting performance plot compared to the first couple of test scenarios. In this subplot we have a lot more cases where there exist direct path ranges. This is because of the presence of the additional obstacle vehicle so that after the autonomous vehicle has turned away from the obstacle vehicle directly ahead, there is still an obstacle in the adjacent lane. Since we have a small number of iterations per frame of the active contour algorithm it’s not until the fourth frame that active contour is able to move off the front edge of the obstacles. On the fourth frame the motion planning algorithm is able to find a route that is able to get further down range than the direct path could get by roughly 25 feet. The upwards slope after the 2 second mark is due to the autonomous vehicle straightening out in the leftmost travel lane. The obstacle being detected as this point is the solid yellow double line separating the lanes of oncoming traffic. As the vehicle is turning away from traveling to the leftmost lane the intersection between the double yellow lines and the direct path becomes larger and larger. However the solution path is able to identify a route all the way to the end of the IPM image much earlier than the direct path.
The first and fourth frames are displayed as the two left subplots in Figure 5-27 respectively. In the first frame, because of the large cross range extent of the combined obstacle, the limited number of iterations used by the active contour algorithm was not enough to get the contour around to the side of the obstacle. By the fourth frame the planning algorithm was able to identify that it could get into the leftmost travel lane. This was mostly due to the clockwise rotation applied around the obstacle edges. Without this rotational force, the contour would have gotten stuck in the v-shaped cutout between the obstacles leading to the autonomous vehicle to try and squeeze between the two obstacle vehicles. The two right subplots are snapshot from when the autonomous vehicle is passing the obstacle vehicle that’s in the middle lane. At this point it becomes similar to the two previous test scenarios.

Figure 5-26: Test Case 3 results plots
In the final scenario, obstacle vehicles are located in all three travel lanes. In this case the autonomous vehicle must perform multiple lane changes in order to pass all four vehicles. The upper left subplot of Figure 5-28 shows the initial setup of the test scenario. The upper right plot shows the overall trajectory taken by the autonomous vehicle. The first maneuver was to switch to the middle lane to pass the obstacle vehicle that was directly ahead of the autonomous vehicle. As the autonomous vehicle approaches the obstacle vehicle in the middle lane, the decision to return to the right most lane is made due to the counter-clockwise rotation and the fact that obstacle vehicle in the leftmost lane is blocking sight further into the leftmost lane. There is a slight rightward bump around the 600 meter mark. This was due to the close proximity of the obstacle vehicle in the middle lane as the autonomous vehicle was approaching it. The middle right subplot of Figure 5-29 shows how narrow the gap between obstacle vehicles is and how the middle lane obstacle pushed the solution slightly to the right before steering back into the middle lane to get around the final obstacle vehicle.
The lower left subplot of Figure 5-28 shows the performance of the motion planning algorithm for this test scenario. The first two samples on the direct path represent the distance to the obstacle vehicle in the rightmost lane. The direct path sample between 1 and 8 seconds are the distance to the obstacle vehicle in the middle lane. The direct path samples between 8 and 25 seconds represent the distance to the farthest obstacle vehicle in the rightmost lane. When the solution path range is well above the direct path range, it correlates to when the autonomous vehicle can see past the adjacent obstacle vehicle in the IPM frame.

Figure 5-29 contains four snapshots from the fourth test scenario. These snapshots from left to right are of the first frame, passing the first leftmost obstacle vehicle, passing the middle obstacle vehicle, and passing the last obstacle vehicle respectively.
We tabulated the number of instances that the solution paths had a down range measurement greater than, less than and equal to the direct path down range measurement in Table 1. Only frames where either the solution path or direct path encountered an obstacle were valid for tabulation to avoid inflating the equal cases when both the solution path and the direct path did not encounter any obstacles throughout the IPM frame. In all four cases there were no points where the solution path range was less than the direct path range. In the first and second test scenarios there were only two opportunities where obstacles were blocking the path forward and in both instances the motion planning algorithm was able to identify a solution that would travel further than the direct path.

In the third test scenario, for 80% of the opportunities our motion planning algorithm was able to outperform the direct path ranges while the fourth test scenario was only able to outperform the direct path ranges 56% of the time. Overall our motion planning algorithm was
able to identify a path that would travel further than if the vehicle had driven straight 62% of the time.

<table>
<thead>
<tr>
<th>Solution Path vs Direct Path</th>
<th>Scenario 1</th>
<th>Scenario 2</th>
<th>Scenario 3</th>
<th>Scenario 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Equal</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>33</td>
</tr>
<tr>
<td>Greater than</td>
<td>2</td>
<td>2</td>
<td>12</td>
<td>42</td>
</tr>
<tr>
<td># Opportunities</td>
<td>2</td>
<td>2</td>
<td>15</td>
<td>75</td>
</tr>
</tbody>
</table>

A final remark about the motion planning algorithm is about its performance against slower moving obstacles. Currently regardless of the obstacles speed the vector field around the obstacle has a fixed range away from the object edges. As we’ve seen from our close loop simulations this works well when the speed delta is small (i.e. within 5 mph). However with larger speed deltas (i.e. greater than 20 mph) the autonomous vehicle gets much closer to the obstacle and may not always have enough time to maneuver out of the way. This can be mitigated if a vertical boundary extension was used in a similar manner to the collision boundary expansion used to expand obstacles to occupy the entire width of a travel lane. If the vertical boundary extension was related to the relative speed between obstacle and autonomous vehicle then the autonomous vehicle would react to the obstacle from further away.

5.3.2 Performance against real world video

In this section we discuss the results of performing our motion planning algorithm on real world video sequences. Unlike the closed loop simulation performed in 5.3.1, the planned steering command by our motion planning algorithm is not acted upon. This testing is primarily
to assess our motion planning algorithms performance against real video sequences. To isolate its performance we bypass the obstacle detection step and just use the ground truth labeled data to identify non-roadway. By doing this we simulate an ideal detector that is not influenced much by lighting and shading of the natural world. As we showed in Section 4.3.2, these two features introduced regions that were falsely classified as obstacles and more than halved the accuracy of our classifier when no pre-smoothing was applied to the image. The presence of these false obstacles would cause the motion planning algorithm to perform unnecessary evasion maneuvers.

The first real world video sequence we used was the same Fauqueur data set we used in Section 4.3.2 which contained 101 labeled image frames and consisted of driving down a street in the city of Cambridge. We repeat the same analysis as we performed for the closed looped simulation in Section 5.3.1. The left subplot in Figure 5-30 shows the first frame of the video sequence. It shows that the camera is in the left lane behind two bicyclists with two approaching vehicles in the right lane. The right subplot is a plot of the performance metric for each of the 101 frames. Using the real world images, there were not as clear of a pattern as was seen in the performance plots of the closed loop simulation. Of the 96 frames where the performance metric was applicable, 67 frames showed the solution path to have traveled further than the direct path while 21 frames did worse. A majority of those 21 frames occurred in the beginning of the video sequence where a slight gap between the cyclists happens to line up directly in front of the camera giving the direct path a clear line of sight down the entire look ahead distance.
Figure 5-30: CamSeq01 data set result plots

The upper row of subplots in Figure 5-31 show the original frames from the video sequence while the lower row of subplots show the motion planning results around the red obstacles in the IPM frame. The cyan crosses in the IPM images indicate the point along the solution path that it intersects an obstacle while a magenta cross is the point where the direct path solution intersects an obstacle. When a cyan or magenta cross is missing, the corresponding solution did not intersect with any obstacles. In the first frame, the motion planning algorithm is able to plan a route that would take the vehicle around the two bicyclists in the left lane. However as the vehicle continues to travel straight it becomes harder for the algorithm to plan a route between the bicycles and the second oncoming car. This causes the down range distance our solution path is able to achieve to get shorter and shorter before it intersects with an obstacle. After the second oncoming car passes in the right lane, around the 60th frame, a new route is identified that will allow the solution path to reach the end of the 300 foot look ahead distance (middle right subplot). The rightmost subplot occurs when the vehicle begins maneuvering into the right lane and plans to turn the vehicle to the left a little to avoid driving straight onto the sidewalk on the right side.
Our second real world video data set is identified as seq05VD and also comes from the work done by Fauqueur. While this sequence is well over three minutes, the ground truth labels only exist for every 30th frame. For this reason we had to reinitialize our active contour each frame instead of using the previous solution since the frames used are not temporally adjacent. The left subplot in Figure 5-32 shows the first frame of the video sequence and consists of driving through another city. The right subplot once again plots the performance metric comparing where the down range of our solution path intersects with an obstacle to the down range distance the direct path achieves before intersecting an obstacle. Since a majority of the sampled video frames had a clear path ahead of the vehicle, there were only 29 frames out of the available 179 frames that our performance metric was able to be applied to. Of those 29 frames, only 13 frames showed the solution path to have an improvement over the direct path while it had worse performance for 5 of those frames. For the other 11 frames the two solutions had similar ranges.
but were unable to make it all the way to the end of the look ahead distance without intersecting an obstacle.

Figure 5-32: seq05VD data set result plots

Figure 5-33 shows snapshots of the motion path found for several frames in the video sequence where the top row of subplots are the original video frames and the bottom subplots are the solution path around the red obstacles plotted in the IPM frame. In the leftmost subplot we see that the solution path tries to cut through the obstructed region around 200 feet down range. This is due to a combination of insufficient number of iterations of the active contour algorithm which would have allowed it to continue stepping away from the obstacle, and the fact that the vanishing point estimate is reinitialized to be at the center top of the IPM image. This re-initialization is needed since the frames are not temporally adjacent and the vanishing point tracker is not designed to localize from a single frame. The middle right subplot shows that the obstacles create an impassible barrier around the 200 foot down range mark. This barrier is a result of the sidewalks appearing to converge and the upwardly sloped hill not following the flat world assumption used by the IPM transformation. In scenarios like this where there is an impassible wall the autonomous vehicle should slow down since it is unable to find a steering only solution to navigate forward. The final subplot shows the planning results for a slightly
curving road. The direct path solution hits the road curve on the right near 250 feet down range while the solution path is able to move with the curve until it is drawn into the obstacle near the vanishing endpoint because it was once again reinitialized to the top center of the image since the ground truth labels are not temporally adjacent.

Figure 5-33: seq05VD data set sequence plots

Once again we compiled the comparisons of the solution path to the direct path in Table 2. For both video sequences the majority of the time the solution path was able to find routes that would allow the vehicle to travel further than if the vehicle had continued straight ahead. Overall our motion planning algorithm was able to identify a path that would travel further than if the vehicle had driven straight 64% of the time.

Table 2: Tabulation of performance metric for real world video sequences

<table>
<thead>
<tr>
<th>Solution Path vs Direct Path</th>
<th>CamSeq01</th>
<th>seq05VD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Less than</td>
<td>21</td>
<td>5</td>
</tr>
<tr>
<td>Equal</td>
<td>8</td>
<td>11</td>
</tr>
</tbody>
</table>
Chapter 6  Conclusions and Recommendations

6.1 Conclusions

In this thesis we proposed a vision algorithm for self-driving cars on highways that utilizes a single camera. Its simplicity lends it towards being included in a set of other simple algorithms which are used to vote and confirm a final motion planning result. The approach for our algorithm is threefold: a vanishing point detector, a situational awareness system, and a motion planning system.

In Chapter 3 we developed the vanishing point tracker by tracking the intersections of perspective lines with a Monte Carlo Tracker (MCT). This estimated the pixel location of the forward vanishing point creating the desired steering destination for the autonomous vehicle. We were able to show that after the initialization phase of 50 frames (or 1.67 seconds) the MCT was able to maintain an overall pixel error below 50 pixels. After reviewing the MCT estimates for each frame in the video sequences, we concluded that the MCT was unable to meet the 6.3° angle error threshold derived from a 3 meter lane width and a 15 meter look ahead distance. The MCT approach is still viable for localizing the general region that the vanishing point exists in, which turns out to be sufficient for the motion planning algorithm in Chapter 5.

In Chapter 4 we utilized the Inverse Perspective Mapping (IPM) to convert the perspective image into a bird’s eye view type image which created a more uniformly distributed data density of the roadway scenes. The conversion to the IPM view induced a streaking effect for objects that were not flat on the ground plane. This feature was used as part of an obstacle
classifier to assist in identifying objects that a trajectory would need to be planned around. After comparing the labeling of our obstacle detector with the ground truth labeling of both simulated and real world data, we found that we had an accuracy of 70.7% against simulated data and 64.5% against a real world video sequence. The real world video sequence originally had an accuracy of 31.6% due to the decision to use the Tuohy road surface detector which struggled under lighting variations more prevalent in the real world video sequences. However after applying a Gaussian smoothing filter to the image before applying our obstacle detector, we were able to double the accuracy of the detector.

In Chapter 5 we first abstracted the motion planning problem as a binary labyrinth which showed that active contours are able to navigate through a winding environment after some adjustments to the external energy function. These adjustments included adding a force in the direction of the desired endpoint to provide motivation for the contour to move away an obstacle in the direction of the goal, clockwise and counter clockwise rotations around the obstacle edges to facilitate backtracking and weaving through obstacles, and making the vectors in the gradient vector field uniform length to extend the influence of an obstacle reach. We then transitioned to a closed loop simulated environment to assess the algorithms ability on an artificial highway scene. We were able to show that the active contours were able to create a motion profile that allowed the vehicle to successfully navigate the obstacles.

Lastly, we applied our algorithm to a video sequence of real roadway images where the active contours provided a path through the scene to the estimated vanishing point. The active contours were able to provide reasonable travel trajectories resulting in an improved down range metric of 64% which has comparable performance to that of the simulated data set. On its own our motion planning algorithm has insufficient information to be solely reliable for planning a route through a roadway scene. By combining our simple single camera approach with other
approaches that utilize other sensors, these algorithms can combine their results and determine through voting which planning solution is the best overall.

6.2 Future Work

There were several topics of further study that were identified through our research. The first is expanding the Monte Carlo Tracker from the vanishing point detector to detect all three vanishing points instead of just the one ahead of the camera. This would involve expanding the sample region to be beyond the image boundaries and increasing the number of extracted lines and intersections needed to localize the vanishing points.

In Chapter 4 we discussed the streaking effect of objects due to the Inverse Perspective Mapping (IPM) transformation. Another noticeable effect was an oscillation that was introduced at further ranges due to the car suspension after encountering bumps or potholes on the roadway. Developing some method for video stabilization in the IPM frame would reduce this effect and make frame differencing techniques more applicable, such as those used in optical flow techniques. Additionally we saw that the IPM transform required decent knowledge of the camera parameters in order to properly remove the perspective effect from the image. Perhaps incorporating some method to recalibrate the IPM parameters using a projected grid onto the roadway surface could aid with improving the classification of non-flat objects.

The primary motivation of our research was to focus on single camera techniques that would be lower cost as compared to LIDAR and stereovision systems. Additionally it could work in conjunction with those systems to provide a secondary verification to other detection methods. In the case where our approach is used in conjunction with a LIDAR or stereovision system, the potential to utilize the range measurements could improve our approach to obstacle detection by minimizing the lighting effects which we have seen degrade the performance of our algorithm.
In Chapter 5 we noted that the autonomous vehicle would not always maneuver to the center of the travel lane. As discussed this is due to the the vanishing point not guaranteed to be in the center of a travel lane due to bias from occluded perspective lines. However a more robust alternative might be to leverage the research done on travel lane detection and use the regions classified as travel lanes to create a peak in the external energy to guide the active contour towards the center. This may also help with getting the vehicle to change lanes in a timely manner.

6.3 Final Remarks

In this thesis we have proposed a single camera based motion planning algorithm utilizing the Inverse Perspective Mapping transform for obstacle detection and active contours for motion planning. We have shown that under favorable lighting condition our obstacle detector is able to accurately identify non-drivable obstacles.

Our approach to motion planning transformed a technique normally applied to image segmentation problems and repurposed it as a motion planner. It was able to demonstrate successful navigation through a simulated environment and provided comparable performance metrics to real world video sequences. Its simplicity and low cost make it an ideal algorithmic candidate to be included among a set of algorithms which vote on a final planning result.
Bibliography


