Vehicle Movement and Speed Estimation using a Cost Effective Camera

by

Deepali D. Kamat

A Project Report Submitted

in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in

Computer Science

Supervised by Dr. Thomas Kinsman

Department of Computer Science

B. Thomas Golisano College of Computing and Information Sciences

Rochester Institute of Technology

Rochester, New York

December 2016
Dedication

I would like to dedicate this project to my parents and my advisor, Dr. Thomas Kinsman for encouragement and support throughout the course of this project.
Acknowledgments

I would like to extend my sincerest gratitude towards Dr. Thomas Kinsman for giving me the opportunity to work on this project and also for his patience and timely encouragement. I would also like to thank my colloquium supervisor, Dr. Leon Reznik for his feedback and input at each milestone and also all my peers who have provided me with valuable feedback at each stage of the project.
Abstract

Vehicle Movement and Speed Estimation Using a Cost Effective Camera

Deepali D. Kamat

Supervising Professor: Dr. Thomas Kinsman

Technology is on the rise and the automotive industry is no stranger to this. One such development is the onset of driverless vehicles. Driverless cars, or autonomous vehicles, are hitting the road \cite{1,2} however, the current system of these vehicles is comprised of expensive and complex systems including radar, LiDAR, sonars, GPS and motion sensors. The motivation behind this project is to explore computer vision techniques and experiment with cost effective algorithms related to autonomous driving. This involves estimating the speed of a vehicle, and road lane detection using inexpensive dash cam. Operations are performed over a set of consecutive frames from the video to determine lane positions and the speed of the vehicle.
# Contents

1. Dedication ........................................................................................................... 2

2. Acknowledgments ................................................................................................. 3

3. Abstract .................................................................................................................. 4

4. Introduction ............................................................................................................ 7

5. Design .................................................................................................................... 8
   i. Data Collection and Preprocessing ................................................................. 8
   ii. Optical Flow using Lucas Kanade ................................................................. 9
   iii. Canny Edge Detection .................................................................................. 10
   iv. Hough Line Transform ............................................................................... 11
   v. Vanishing Point Detection .......................................................................... 12
   vi. Speed Estimation ....................................................................................... 13

6. Implementation ..................................................................................................... 15
   i. Lane Detection ............................................................................................... 16
   ii. Speed Estimation .......................................................................................... 18

7. Results and Analysis ............................................................................................. 20

8. Conclusion .............................................................................................................. 23
List of Figures

2.1 Speed estimates from frame differences ....................... 13
3.1 Vanishing point in an image frame .............................. 17
4.1 Canny Edges over an image frame .............................. 20
4.2 Hough lines ....................................................... 21
4.3 Histogram of average offsets over 200 frames ............... 22
Chapter 1

Introduction

This project explores cost effective ways to detect lanes and computes the speed of any moving vehicle by using computer vision techniques and applying them to videos taken by a dash cam placed on the dashboard of the moving vehicle. The videos are processed offline. These videos have a frame rate of 30 frames per second (fps). Feature selection involves extracting white lines and using them as a basis for detecting lanes. One of the methods experimented with speed estimation was Optical Flow using the Lucas Kanade \[5,8\] algorithm. However, the desired results were not met due to certain points moving out of frame. The experiments have been performed using OpenCV with a python API, and the help of the PyCharm IDE.
Chapter 2

Design

The experiments for this project have been focused on videos taken while driving on the highways as they have more uniform speed and the crashes on highways are more fatal. The assumption is made that the lane markers on highways are 30 feet apart and the vehicles need to maintain an optimum speed of 55mph.

The Imaging Chain goes as follows:

- Data Collection and preprocessing
- Optical Flow (Lucas Kanade)
- Canny Edge Detection
- Hough Line Transform
- Vanishing Point Detection
- Speed Estimation

2.1 Data Collection and preprocessing

The data was collected while driving across highways around having a speed limit of 55mph according to the National Maximum Speed Law (NMSL)\textsuperscript{[9]} These videos were re-sized to a resolution of 1920 by 1024 pixels to ensure uniformity and better viewing of results. Gaussian Blurring is used for de-noising.
2.2 Optical Flow using Lucas Kanade

This algorithm, proposed in 1981\cite{5,8} uses local information surrounding the point of interests in an image. This however, can at times be a disadvantage if the points move beyond the frame of the image captured. This method is applied in sparse context as it only takes into consideration the local information relative to the local window surrounding the points of interest.

Feature tracking is used to determine optical flow. We consider flow as a process of matching local pixels. Small areas are considered around the feature points and a matching factor is considered in the sequence of images in the video. Hence, for every point \((x, y)\) a box representing a window is taken around the points. The pixels within the window region are considered and the related energy function is minimized. This method also uses the brightness constancy assumption. The original image is divided into smaller sections. Then a least square fit of the optical flow constraint equation with weights is applied. This method suffered from points going out-of-frame which could not be mapped or stored due to lack of storage and the high amounts of computation involved.\cite{1}
2.3 Canny Edge Detection

Canny edge detection, developed by John F. Canny is a popular edge detection algorithm.\textsuperscript{[10]} It involves the following stages:

1. Noise Reduction:
   The image frames are subjected to Gaussian blurring over a 7x7 window.

2. Image intensity Gradient Calculation:
   The blurred or smoothed images are then subjected to a Sobel filter kernel which provides us with the first derivative in the horizontal ($G_x$) and vertical ($G_y$) directions. These can be used to calculate the edge gradients and direction for each pixel in the image frame.

3. Non-Maximal Edge Suppression:
   This is performed to eliminate any undesired pixel values that may not be constituent of the edges. In order to do so, each pixel is evaluated for whether or not it is the local maximum in its surrounding window neighborhood in the gradient direction. After non-maximal edge suppression, only the strongest edge gradient remains.

4. Image Thresholding:
   This operation decides which edges among the calculated edges are actually edges and which need to be removed. Two values: minimum and maximum are taken and all the edge values that do not fall within the range of the minimum and maximum values are discarded. Edges must have at least a maximum edge strength to be considered an important edge to retain. These edge points are then iteratively extended using a connected-component analysis until the edge strength drops below the minimum edge threshold selected.
2.4 Hough Line Transform

Hough transforms are used to determine shapes in any image, provided the shape is represented in mathematical form. In this project, we detect lane markings using the Hough Transform. A line is represented using the formula \( y = mx + b \) or in parametric form as \( \rho = x \cos \theta + y \sin \theta \) wherein \( \rho \) is the orthogonal distance between the origin and the line and \( \theta \) is the angle between the orthogonal line and the horizontal axis calculated in the anti-clockwise direction. This direction is decided depending on the representation of the coordinate system. Lines passing below the origin, possess a positive \( \rho \) angle and an angle value less than 180. If the line passes or is above the origin, an angle less than 180 is taken instead of greater, however, the \( \rho \) value considered is taken negative. For vertical lines having 0° along with horizontal lines have a 90° value.

As mentioned above, lines can be represented in the terms of \((\rho, \theta)\). An array to accumulate the values of rho and theta are created where \( \rho \) is represented as rows and \( \theta \) is represented as column values. The values for rows and columns are decided based on desired pixel accuracy. The maximum possible value for \( \rho \) is the length of the diagonal of the image from the image sequence. For each point \((x,y)\) in the image line, apply \( \theta \) values ranging from 0 through 180° to evaluate the corresponding \( \rho \) values. The same is performed for all the values and the higher values are voted upon. The cell with highest votes in the end which denotes the line in the image at a distance of \( \rho \) from the origin and at angle \( \theta \)°.
2.5 Vanishing Point Detection

Under perspective projection, parallel lines in three-dimensional space project to converging lines in the image plane. The common point of intersection is called *the vanishing point* and may eventually belong to the line at infinity of the image plane in the case of 3D lines parallel to the image plane.\(^{[12]}\) This point is the intersection of projection of a set of parallel lines in space on that of the plane of the image frame. Vanishing points are detected by using a voting algorithm on Hough lines and increasing the vote if the line passes through a pixel. The area is then averaged out to obtain the point with the highest votes for pixel crossings. The coordinates of the point are returned and the point obtained is the vanishing point.
2.6 Speed Estimation

Using known, National Standards for lane markers as image fiducials, and the known frame rate of the camera, we can estimate the speed by matching frames in a looped manner. In order to do so we first create a look-up table with the speeds at each frame match rate.

Given the frame rate of $1/30^{th}$ of a second, and knowing the distance between dashes in the road, we can estimate the vehicle’s speed. This speed estimation takes advantage of the repetitive pattern of road stripes on a standard road. The method is to compare the current frame in the video feed to a queue of past frames, and see where the best match occurs. Once if the best match occurs between $N$ frames, the speed can be computed, assuming that dashed moved forward one dash over $N/30^{th}$ of a second.
Figure 1 depicts a graph of the estimated speed, based on the number of frames between frame matches. The method is not particularly accurate, but takes advantage of the known frame rate and the availability of the road dashes as a viable signal.

These speeds have been calculated keeping an ideal highway road in mind where the vehicles have the maximum speed limit of 55mph and the lane markings have the following specifications. The assumption is that the lane markings are at a consistent distance of 30 feet apart.

- feet per hour = 290400
- feet per minute = 4840
- feet per second = 80.6667
- feet per frame = 2.6889

- frames for each road mark = 11.1570
- time per frame in minutes = 5.5556e-04
- time per frame in hours = 9.2593e-06
- frames for each road mark = 11.1570

For each input video frame, the current frame is compared to the past 20 frames to find a best match. A circular queue of image frames is used to constantly maintain the past 20 frames. The speed is then estimated from this best match. This is a motion-from-known structure technique in which we utilize the known pattern on the road.
Chapter 3

Implementation

The Vehicle movement and speed estimation is done in two separate parts. The evaluations are performed on stored data collected using a cost effective dash cam. The first part uses vanishing point detection methodology and is used to detect lanes based on lane markers and the vanishing point helps understand the direction the vehicle is headed. The second part, involving speed estimation, is based on the motion from structure methodology, in which we estimate speed by matching frames.
A. Lane detection

Methodology: Vanishing point detection

The data is preprocessed by resizing each frame in the image sequence to a fixed resolution. Each frame undergoes Gaussian blurring to reduce unwanted noise. The images then undergo Canny edge detection in which any unwanted non-edges get discarded and only ‘maximal’ edges get prominence as mentioned in the explanation in section 1.3. In some cases the Gaussian blurring might be avoided as it removes too much detail from the image frame. This is done depending on the resolution of the camera.

The detected edges are then fed into a Hough line detector, which fits a set of line segments into a set of pixel points which were detected by the Canny edge detector. The Sobel edge Detector is applied, which helps in calculation the horizontal and vertical edge gradients. This can be performed after eliminating the top half from the skyline and the sky edges are not of importance with respect to the project. Similarly, the left-most and right-most regions of the image are cropped out to emphasize the region-of-interest directly in front of the vehicle.

Once the vertical and horizontal lines are calculated, we compute the magnitude and angles. The lines outside the range of 0 to 45 are eliminated depending on which lane we are focusing. This is done by searching through different angles the edges might be present at. Consider a value of 22°. This serves as the difference or iteration between angles (Δ). Using Δ, we calculate the minimum and maximum angles. The angles that lie outside the range of the minimum and maximum angle values are discarded. We consider the top five percent
of the edge strengths as having the important edges that we require. For each of those angles, we compute the magnitude and eliminate the lower values. We set up Boolean values of pixels for all the points that lie at the correct angles that have magnitude greater than the threshold value. For all the points obtained, form equations of the line and cast votes for all the points that lie within one pixel of the line. We use the length of the line as a degree-of-confidence metric for the line, and longer lines are given more votes.

The voting for the vanishing point has some pointing accuracy, so we fix that by finding the average number of votes for all pixels within a radius of 10 pixels. Once this is done, we scan the values and collect votes for regions of that value of the radius, and find the point having the maximum number of votes. This point represents the vanishing point.

Figure 2: Example of Vanishing point in an image frame
This procedure is repeated across all frames. The vanishing point when observed from the oculus point helps us understand in which direction the vehicle is moving and when the vanishing points moves or shifts more than a considerable amount, we understand, that there may be a divergence or the vehicle might be taking a turn respectively.
B. Speed Estimation

Methodology: Motion from structure

The image sequence is read in and is queued up 20 frames at a time. For each set, the last frame inserted into the queue is compared with the rest of the frames in the queue. The frames are compared backwards through time and a sum of absolute differences is calculated across all of them. The frame with the least difference is stored and then a new frame is inserted and the same procedure is repeated. When comparing backwards, we leave out the extreme end frames as end frames will have very little difference between them. The returned offset values are averaged over all the matched sets and the averaged value is returned. The returned value represents the frame number. This number is looked up in the table formed as mentioned in section 1.6 and the corresponding speed is the speed of the vehicle.

The estimation follows uses two basic speed and motion formulas:

\[
\text{speed} = \frac{\text{distance}}{\text{time}}
\]

\[
\text{time} = \frac{\text{distant number of frames between matches}}{\text{frame rate}}
\]

The frames are matched using a circular queue and the national standard for lane markers as fiducials. The frame rate of the camera is 30fps.
Chapter 4

Results and Analysis

The experiments helped obtain vanishing points, which help estimate the movement or the direction in which the vehicle was going.

The following figure displays some of the Canny edges detected over an image frame from the image sequence taken from the video file:

![Canny edges example output](image)

*Figure 3: Example output for Canny edges over an image frame*

These edges were then fed into a Hough line detector to obtain the vanishing point.
Hough Lines are used to detect the vanishing point across the frames. The vanishing point depends on the point of observation as well and in certain cases depending on the number of votes, there can be more than one vanishing point.

With the assumptions that the lane markers are stable, the speed is estimated across all frames and averaged to get the frame number with the most likely match. In one of the experimental videos, the vehicle was at an average speed between 62mph to 63mph and the average speed estimated for the same video is 63.634mph. A total of 200 frames were taken from the video and the frames were matched.
Figure 5: Histogram of average offsets over 200 frames
Chapter 5

Conclusions

Multiple videos were recorded using a $27 dash cam and experiments were performed using OpenCV with Python. The experiments helped demonstrate the possibility of using a cost effective system for lane detection and speed estimation.

While there exists driver assistance technology and driverless vehicles are approaching the market[1],[2], this project displays a possibility and potential for similar technology in an after market product for existing vehicles, older models and vehicles in developing countries.

Future work for this project involves using Optical Flow successfully for speed estimation. It would also aim at making use of vanishing points for other image understanding possibilities such as bridge height estimation and collision avoidance. Other possibilities would include trying to detect road signs, such as stop signs and speed limit postings.
Bibliography


