# Using Visual Lane Detection to Control Steering in a Self-Driving Vehicle

Kevin McFall

Department of Mechatronics Engineering, Kennesaw State University,
1100 South Marietta Parkway, Marietta, GA, 30060, USA
kmcfall@kennesaw.edu

**Abstract.** An effective lane detection algorithm employing the Hough transform and inverse perspective mapping to estimate distances in real space is utilized to send steering control commands to a self-driving vehicle. The vehicle is capable of autonomously traversing long stretches of straight road in a wide variety of conditions with the same set of algorithm design parameters. Better performance is hampered by slowly updating inputs to the steering control system. The 5 frames per second (FPS) using a Raspberry Pi 2 for image capture and processing can be improved to 23 FPS with an Odroid XU3. Even at 5 FPS, the vehicle is capable of navigating structured and unstructured roads at slow speed.

**Keywords:** Self-driving vehicle, Hough transform, dynamic threshold, inverse perspective transform, temporal integration, angle control.

## 1 Introduction

It is now generally accepted that self-driving vehicles [1]–[3] in one form or another are the future of automobile transportation. Every major car manufacturer is exploring autonomous driving [4]–[8]. Elon Musk, CEO of Tesla Motors, famously stated in 2014 that Tesla's new set of cars to be unveiled in 2015 will be capable of self-driving 90% of the time [9]. Lane departure warning, adaptive cruise control, and self-parking features are already available on luxury model cars. Understandably, intense research is ongoing to develop algorithms and hardware to make these and more advanced self-driving capabilities sufficiently inexpensive and reliable to be made universally available.

Self-driving vehicles have been pioneered by Google [10] and the DARPA Grand Challenge in 2005 [11]–[13] and Urban Challenge in 2007 [14]–[18]. However, the Google car has access to high resolution 3D maps of the world [19] and DARPA Urban Challenge competitors were provided exact digital maps of the course, enabling navigation with limited onboard perception [20]. Commercial systems do not currently have access to such data, requiring robust sensing systems. A wide array of research addresses lane detection [21], but most focus on a particular subtask with few offering quantitative performance evaluation on the full system [20], and even fewer gathering characteristics essential for navigation [22]. This manuscript

implements all the requisite functional modules for lane detection [20], and its primary contribution lies in extracting steering commands from the algorithm and using them to control a vehicle on an actual road. In order to avoid the safety issues associated with testing on a full-sized automobile, the 4 ft by 4 ft aluminum-framed vehicle pictured in Fig. 1 was used to test lane detection and steering control.

**Fig. 1.** Self-driving vehicle used during field tests.

Modalities for lane and road perception include vision, light detection and ranging (LIDAR), geographic information systems (GPS), inertial measurement units (IMU), vehicle dynamics, and radar [20]. Vision is the most common, but LIDAR offers 3D structure of the environment and an active light source to mitigate issues stemming from shadow and darkness. LIDAR devices have been used extensively in the DARPA Grand Challenges, but high cost currently prevents their use from becoming widespread [20]. Accordingly, this work focuses on low-cost camera sensing instead. Lane detection with cameras is generally classified as using color, texture, or edge features to segment the road surface [21]. Among these, edge methods using some version of the Hough transform are one of the most common [22]–[43].

The main functional modules [20] of successful vision detection systems include: pre-processing, feature extraction, road/lane model fitting, temporal integration, and image to world correspondence. Some research touch only on pre-processing and feature extraction [25], [32], [35], [37], [38] while others apply various road models [22], [24], [31], [42]. A common method for connecting a road/lane model to vehicle position in the real world involves perspective mapping [23], [26]–[30], [34], [36], [40], [41], [43] which translates image lines to their corresponding locations in real space. Perspective mapping is most often used to reject potential lane lines not parallel to each other, or those indicating impossible lane widths. Temporal integration with the Hough transform, using information from previous image frames, is much less common. This technique can be used to identify a vanishing point [24] or reduce the image's region of interest (ROI) [33], [39]. Temporal integration is essential to the algorithm presented here, by effectively reducing the ROI and allowing accurate localization of the vehicle in real space when only a single lane boundary is detected. Another uncommon but useful technique is to identify a so-called virtual boundary, e.g. the location in the image of an undetected lane boundary using the detected boundary on the other side of the lane [22], [28]. The work here combines all these aspects: using boundary positions from previous frames calculated with the inverse perspective transform to limit the search space, and predicting virtual boundary locations when a boundary is obscured or otherwise not detected.

The vast majority of lane detection research evaluates an algorithm offline using a video feed of previously recorded road images, while some are tested in real-time with a human driving the vehicle. Both of these methods assume the vehicle always

travels nearly parallel to the road direction. Deviations of only 5-10° significantly hamper lane detection where one lane boundary can disappear completely from the field of view. With steering controlled autonomously rather than by a human, lane detection can be tested with a full range of representative road images. This work presents a complete system where lane detection is shown sufficient to successfully control steering autonomously in real road conditions.

## 2 Lane Detection

At the core of the lane detection algorithm is the Hough transform, which evaluates a binary edge image by discretizing all possible lines in the image into an accumulator matrix and counting the edge pixels falling on each line [44]. Accumulator entries with large numbers of pixels falling on them are likely candidates for lane boundaries. A detected line is selected from the candidates by searching for local maxima in the accumulator matrix close to the boundary detected in the previous frame. Essential to the algorithm is determining whether the detected line is to be trusted or not. A detected boundary is trusted if either it has not changed significantly from the position of a trusted boundary in the previous frame, or if the distance in real space between left and right detected boundaries is close to the actual lane width. The inverse perspective transform is used to measure the distance in real space between the two detected lines in the image. A previous version of this algorithm [45] using a MATLAB toolbox for the Hough transform [46] successfully identified lane boundaries in 95% of the frames tested during an 8-minute video of highway driving. The algorithm described here modifies the previous one to use the open source computer vision (OpenCV) library [47], [48] in order to implement it in an embedded system. The remainder of this section presents details of the four functional modules of lane detection, where the road/lane model and real world correspondence are covered together with the inverse perspective transform.

### 2.1 Image Pre-processing

The acquired image is first converted to grayscale since color is not used in the algorithm. The top half of the image is cropped out since the vanishing point falls at the image center, and the road bed lies in the bottom half image for a camera with zero roll and pitch travelling on a level road. To further reduce the ROI, the remaining bottom half image is split vertically where the left and right lane boundaries are detected independently in the left and right bottom quarters, respectively.

### 2.2 Feature Extraction

The binary edge image generated with a Canny filter is analyzed using the Hough transform. The `HoughLines` function in OpenCV returns the accumulator entries comprising the most edge pixels above a given threshold. Variations in road scene, image contrast, quality of road markings, etc. significantly affect proper threshold

choice. Allowing `HoughLines` to return too many accumulator entries risks a false positive while too few could miss the actual lane boundary. Returning between 40 and 50 candidate accumulator entries has been found to produce reasonable results for any number of different road situations. The threshold is dynamically updated every frame by decreasing it if fewer than 40 entries are returned and increasing if above 50.

### 2.3 Temporal Integration

Of the 40 to 50 candidate accumulator entries returned by the Hough transform, the actual lane boundary is expected to be one of the strongest lines. However, selecting the strongest line could result in identifying a windshield wiper, an adjacent boundary on a multiple lane road, or otherwise incorrect line.

Each accumulator entry represents a unique line defined by the perpendicular distance $\rho$ from the top left image corner and its angle $\theta$ below the horizontal [44]. The parameters $\rho$ and $\theta$ are used as opposed to the more commonplace slope $m$ and intercept $b$ to avoid slope discontinuities from vertical lines. Converting between $\rho,\theta$ and $m,b$ parameters is straightforward using standard trigonometry.

The first step in determining which accumulator entry to select involves removing obvious outliers. As in [39], candidates are removed from consideration having values of $\rho$ and $\theta$ differing significantly from the line detected in the previous frame. Some processing time could be saved if this step is instead moved to the pre-processing module [33] by negating edge pixels outside the ROI.

Accumulator entries are returned by `HoughLines` in rank order of their strength, which is modified by comparing the Euclidean distances in $\rho,\theta$ space between each remaining candidate line and the line detected in the previous frame. The ranking of each candidate is penalized depending on its distance normalized by the standard deviation of the candidate distances. If all candidates identify essentially the same line, none of their rankings will be modified and result in the strongest being detected. When candidates exhibit large dispersion in distance, however, those farther from the previous lane position are increasingly demoted. Standard deviation rather than mean is used for normalization to account for the expectation of some nonzero distance between detected lines from subsequent frames. The winning candidate is determined by the modified rank, and chosen as the "detected" lane boundary line. Whether the detected line is "trusted" depends on results from the inverse perspective transform.

### 2.4 Inverse Perspective Transform

The road model used assumes driving on a level surface with parallel left and right lane boundaries a known width apart. Comparing detected left and right lines with this model determines whether or not they are trusted as actual lane boundaries.

Camera images follow the central imaging model [44], [45] where a camera is placed at the $XYZ$ origin of the world coordinate axes. The perspective transform projecting point $P$ in real space to its corresponding location $p$ in the image is one-to-one. The inverse perspective transform, however, results in an infinite number of points $P$ corresponding to any given $p$. A one-to-one inverse correspondence between

image lines and world-coordinate lines does exist assuming the real lines lie in the *XZ* plane at a fixed $Y = H$ where $H$ is the height of the camera above ground. The distance $L$ of a line in world-coordinates to the camera and the yaw angle $\phi$ between the line and direction of travel as appears in Fig. 2 are related to the image slope $m$ and intercept $b$ according to

$$L = \frac{H}{\sqrt{\left[2b\tan\left(\frac{1}{2}\psi\right)\Big/ N_c\sqrt{1+1/R^2}\right]^2 + m^2}} \text{ and } \phi = -\tan^{-1}\frac{2b\tan\left(\frac{1}{2}\psi\right)}{mN_c\sqrt{1+1/R^2}} \quad (1)$$

where $\psi$ is the camera's diagonal angle of view, $N_c$ is the horizontal camera resolution, $R$ is the image aspect ratio, and $b$ is measured in pixels. The angle of view is approximately 50° for most cameras.

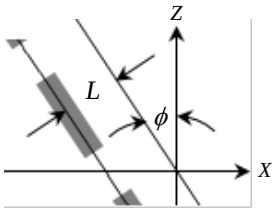


**Fig. 2.** Definition of lane distance $L$ and yaw angle $\phi$ for a camera located at the *XZ* origin and travelling along the *Z* axis.

Detected road boundary lines are trusted if the $L$ distances for both left and right boundaries add to a value close to the actual lane width. If the detected and actual lane widths do not match, detected boundaries will instead be trusted if they have not changed significantly from the previous frame. Detection of lines in the next frame is dependent on boundaries from the current frame; in case only one boundary in the current frame is trusted, the other virtual boundary is predicted using the inverse of Eqn. (1) for an $L$ one lane width away from the trusted line and the same $\theta$. In such a manner, the algorithm continues to operate normally even when one lane boundary is occluded, beyond the field of view, beaten in rank by a nearby spurious line, or detection otherwise fails.

## 3 Vehicle Hardware and Control

Rather than testing vehicle control on a full-size automobile, a smaller but still road-worthy custom vehicle was designed and built. The smaller vehicle is safer to operate and easier to manage. The aluminum-framed design is relatively stiff and still lightweight, with 10" wheels sufficiently large to traverse uneven roads and other obstructions. The front drive wheels control steering as well with each motor driven by an independent motor as in [29], while the rear caster wheels are free to swivel.

Both Raspberry Pi and Odroid XU3 microprocessors are explored to provide image capture and processing. The lane detection algorithm is used to generate a steering

command which is sent via serial USB communication to an Arduino Uno, which in turn transmits pulse width modulation signals to motor controllers for each drive motor. The Arduino also accepts signals from a radio control (RC) receiver. The transmitter allows for remote control of the vehicle with a kill switch and mode control for manual and autonomous modes. Currently, autonomous mode uses throttle commands from the transmitter and steering commands from the lane detection algorithm. Power is supplied to the microprocessor, Arduino, and RC receiver with a 5 V portable cell phone battery charger, while a 12 V, 7 aH lead-acid battery provides power to the motor controllers.

Standard proportional, integral, derivative (PID) control can be used for self-driving vehicles [49]. Currently a proportional derivative (PD) control output

$$u = K_1\left(\tfrac{1}{2}W_d - L\right) + K_2\phi + K_3\Delta\phi \tag{2}$$

is implemented where $W_d$ is the most recent detected lane width, $\Delta\phi$ is the change in $\phi$ from the previous frame, and the $K_i$ are control gains. An integral term may be introduced later if significant steady state errors in $L$ persist. Values for $L$ and $\phi$ are based on trusted boundaries only, taking the average of left and right if both are trusted. Initial testing with only proportional gain was unable to maintain a small yaw angle $\phi$; the control problem is primarily one of angle rather than position control. Unmitigated increases in $\phi$ quickly result in loss of one boundary from the field of view and driving over the other boundary. Addition of the $K_3$ derivative term strengthens steering when turning away from the proper lane direction and dampens it to prevent overcorrection when $\phi$ is still large but improving. The control gains were tuned to achieve as responsive and smooth steering as possible within the constraints of chatter in $L$ and $\phi$ and a relatively slow frame rate. The control output $u$ is mapped to integer values between 25 and 125 to be sent to the Arduino as a single byte via USB. Some lane detection research specifically locates the vehicle in lane [42] or provides steering information [22], [50], but none distinguish between error in position vs. error in angle; any successful control scheme must focus primarily on maintaining the proper heading relative to road direction.

## 4 Field Test Results

The vehicle demonstrates successful navigation on straight paths in the numerous situations appearing in Fig. 3. The gray lines in Fig. 3 are trusted by the algorithm while black indicates lines detected in the previous frame. The top row of numbers report lane distances from the inverse perspective transform with left detected lane boundary (left), right detected boundary (right), and detected lane width (center). The bottom row of numbers displays the yaw angle according to the left (left) and right (right) detected lines, as well as the steering command (center). A command value of 75 represents straight while 25 and 125 request full left and right turns, respectively.

Although relatively simple, the lane detection algorithm is sufficient to traverse stretches of road up to 70 ft long as well as the sidewalks, hallways, and indoor environments in Fig. 3. For a camera level in terms of pitch and roll, the only

necessary calibration involves the height of the camera above ground and the lane width. In fact none of these requirements need be specified with much accuracy when applying a simple, single-step calibration process. If any of the parameters are inaccurate, the detected width will differ slightly from its exact width. Before engaging autonomous mode, the road width parameter in the lane detection algorithm can simply be set to the detected lane width to account for any inaccuracies. No fine tuning of algorithm parameters is necessary; the same set of parameters (other than lane width) were used in all locations depicted in Fig. 3.

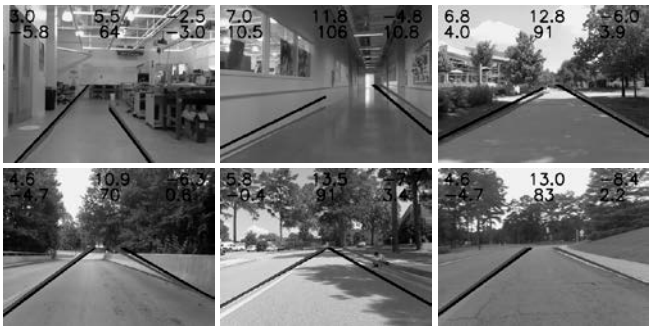

**Fig. 3.** Various scenes successfully navigated by the vehicle including a 5 ft wide artificial lane in a laboratory setting, hallway, sidewalk, and several different road conditions.

By changing only the lane width, the vehicle is capable of autonomous control for lanes of widths ranging from 5 to 14 feet, even in highly noisy environments such as the laboratory setting in the top left image of Fig. 3. The wall and nearby equipment introduce additional lines parallel to actual boundaries. In Fig. 4, such a line (white) is falsely detected, but the 6.4 ft measured distance between the trusted (gray) and detected (white) lines falls outside the threshold for the actual 5 ft lane width. Instead, the right lane boundary is sought in the next frame at a position of the virtual boundary (off-white) determined by the perspective transform prediction of a line 5 ft to the right of the trusted line.

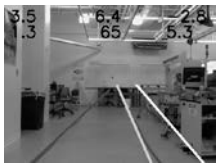

**Fig. 4.** Case where a spurious lane boundary (white) is rejected – the trusted line (gray) is used to predict the location of the rejected lane boundary (off-white) using the perspective transform.

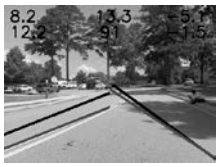

**Fig. 5.** False positive trusted line due to shadow condition.

The lane detection algorithm can still benefit from refinement as is apparent in the trusted right lane boundary in Fig. 5 where the detected line represents a shadow rather than the actual lane boundary. Situations with false positive trusted boundaries generally persist for only a few frames, 6 in this case, before correct identification resumes. The valid trusted left boundary reduces the effect of the false positive by averaging in accurate values for $L$ and $\phi$ with the improper right boundary values. The resulting steering brought the vehicle somewhat closer to the left boundary than desired but still maintained the vehicle safely within the lane.

All field test results used a Raspberry Pi 2 on a Linux operating system for image capture and processing. The 900 MHz quad core ARM cortex in the Raspberry Pi 2 offers a significant improvement over the 700 MHz Raspberry Pi 1. Both versions include support, including a software library, for the Raspberry Pi camera module which connects directly to the board using a ribbon cable. Another single-board computer, the Odroid XU3, is also capable of running Linux and provides even more computing power with a 2.0 GHz quad core Cortex A-15. Both Raspberry Pi and Odroid execute the same Python script, differing only in image capture using the Odroid USB-CAM with the OpenCV `VideoCapture` command rather than the Pi camera and its library. For comparison, the Raspberry Pi is tested both with the Pi camera and the USB-CAM. Table 1 includes frame rates comprised of both image capture and processing for all three microprocessors at two resolutions. Tests have shown that the smaller resolution, with its higher frame rate, offers sufficient resolution for successful boundary detection. Frame rates for the Raspberry Pi are essentially independent of camera selection, and are approximately 5 times slower than using the more powerful Odroid XU3. Future field tests will employ the XU3 exclusively, but have not yet been conducted as existing portable batteries are unable to supply 5 V at the requisite 4 A of current.

**Table 1.** Comparison of frame rates using different microprocessors.

| Resolution | Microprocessor | Camera | Frame rate (FPS) |
|---|---|---|---|
| 352×288 | Raspberry Pi 1 | Pi camera | 2.2 |
| | | USB | 1.8 |
| | Raspberry Pi 2 | Pi camera | 4.5 |
| | | USB | 5.2 |
| | Odroid XU3 | USB | 23 |
| 640×480 | Raspberry Pi 1 | Pi camera | 1.2 |
| | | USB | 0.83 |
| | Raspberry Pi 2 | Pi camera | 2.1 |
| | | USB | 2.4 |
| | Odroid XU3 | USB | 11 |

## 5 Discussion and Future Work

Failure to maintain the vehicle in lane generally does not result from poor lane detection, which is successful even on the curved roads in Fig. 6 as the approximately

straight near-field boundaries dominate the image. Rather, inability of the PD control scheme to react to steering commands is the weakest link. Slow frame rates around 5 frames per second (FPS) using the Raspberry Pi 2 are partially to blame, which will be remedied once the XU3 power supply issue is resolved. Chatter in boundary detection such as jumping from one side of a double marked line to another or from curb shoulder to road edge is also problematic for PD control. An inertial measurement unit (IMU) supplying yaw angle and rate should greatly improve time resolution and accuracy of $\phi$ and $\Delta\phi$. The IMU data and speed from shaft encoders can improve lane detection by incorporating dynamic thresholds for how much a lane boundary is expected to change from frame to frame. Another factor contributing to difficulty with steering control is the inertia required to restore rotated caster wheels when recovering from a turn. Eventually, the vehicle's tank drive steering should be replaced with a steering mechanism similar to that in a standard automobile.

**Fig. 6.** Successful lane boundary detection on curved roads.

**Fig. 7.** Motor and servo controlled actuation of steering (left), acceleration (center), and braking (right) to enable drive-by-wire control of a KIA Optima.

The majority of lane detection research does not supply information about computational speed, which is essential for real-time control of autonomous vehicles. Some algorithms are described as fast [24] or real-time [35] but offer no information about computation time, while others report 10 FPS or less [23], [32], [36]. Frame rates between 35 and 100 FPS depending on image complexity are reported [41] at 640×480 resolution on a 3.5 GHz computer. The 50 FPS in [34] at 640×480 using a 2.4 GHz machine likely does not include image capture time as processing was performed offline. An embedded AMD E-350 1.6 GHz dual-core processor operating at 320×240 achieved 21 FPS [27]. The 23 FPS using the Odroid XU3 at 352×288 is comparable to these published frame rates.

The vehicle and lane detection algorithm presented here is successful on structured and unstructured roads with a variety of features including curbs, shoulders, median dividers, and solid or dashed lane markings. Although currently only capable of traversing straight road sections, improvements are already underway to improve the speed and accuracy of inputs to the PD controller to improve performance. This includes encoder and IMU data, and eventually GPS and a scanning laser range finder. Completely successful self-driving vehicles must rely on multiple modalities such as these rather than relying solely on vision [20].

Field testing has so far been restricted to a 4 ft by 4 ft vehicle, but the system will be ported to a KIA Optima. The Optima has already been modified to be drive-by-wire capable as depicted in Fig. 7 where a motor controls steering with a timing belt connected to the steering column (left), servo motors actuate the accelerator pedal (center), and a cable attached to the brake pedal winds around a motor shaft (right). Once the mature self-driving system is developed on the smaller vehicle, it can be adapted to control a full size automobile.

## References


1. Kayo Mimizuka, "FOCUS: Global race to develop self-driving cars heats up," *Kyodo News International, Inc.*, 18-Oct-2013.
2. Myra Miller, "Self-driving cars could be the future," *Eunice News (LA)*, 16-Oct-2013.
3. T. Pultarova, "Self-Driving Selfcharging Electric Cars Ready to Roll," *Engineering & Technology (17509637)*, vol. 9, no. 12, pp. 10–10, Jan. 2015.
4. Richard Truett, "For GM, the self-driving car of the future is also a thing of the past," *Automotive News*, p. 0040, 15-Sep-2014.
5. Daniel Shane, "Ford moves closer to self-driving cars with high-tech reveal," *ArabianBusiness.com (Dubai, United Arab Emirates)*, 25-Feb-2014.
6. Dee-Ann Durbin, "Honda introduces self-driving car," *Associated Press: Worldstream*, 09-Sep-2014.
7. S. Frizell, "Here Is Mercedes' Outrageous Vision for the Future of Cars," *Time.com*, p. N.PAG, Jan. 2015.
8. D. Leggett, "US: Audi tests 'piloted driving' systems in Florida," *Aroq - Just-Auto.com (Global News)*, p. 8, Aug. 2014.
9. B. Hays, "Musk: Next Tesla cars will self-drive 90 percent of the time," *UPI NewsTrack (Consumer Health)*, Oct. 2014.
10. D. Fox, "Google Car Moves Into Fast Lane," *Discover*, vol. 36, no. 1, pp. 54–54, Feb. 2015.
11. R. Mason, J. Radford, D. Kumar, R. Walters, B. Fulkerson, E. Jones, D. Caldwell, J. Meltzer, Y. Alon, A. Shashua, H. Hattori, E. Frazzoli, and S. Soatto, "The Golem Group/University of California at Los Angeles Autonomous Ground Vehicle in the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 8, pp. 527–553, Aug. 2006.
12. J. McBride, J. Ivan, D. Rhode, J. Rupp, M. Rupp, J. Higgins, D. Turner, and R. Eustice, "A Perspective on Emerging Automotive Safety Applications, Derived from Lessons Learned through Participation in the DARPA Grand Challenges," *Journal of Field Robotics*, vol. 25, no. 10, pp. 808–840, Oct. 2008.
13. S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the DARPA Grand Challenge," *J. Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
14. J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, "Little Ben: The Ben Franklin Racing Team's entry in the 2007 DARPA Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, Sep. 2008.

15. S. Kammel, J. Ziegler, B. Pitzer, M. Werling, T. Gindele, D. Jagzent, J. Schroder, M. Thuy, M. Goebl, F. von Hundelshausen, O. Pink, C. Frese, and C. Stiller, “Team AnnieWAY’s autonomous system for the 2007 DARPA Urban Challenge,” *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, Sep. 2008.
16. J. Sprinkle, J. M. Eklund, H. Gonzalez, E. I. Grøtli, B. Upcroft, A. Makarenko, W. Uther, M. Moser, R. Fitch, H. Durrant-Whyte, and S. S. Sastry, “Model-based design: a report from the trenches of the DARPA Urban Challenge,” *Software & Systems Modeling*, vol. 8, no. 4, p. 551, Sep. 2009.
17. S. Thrun, “Why We Compete in DARPA’s Urban Challenge Autonomous Robot Race,” *Communications of the ACM*, vol. 50, no. 10, pp. 29–31, Oct. 2007.
18. C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. “Red” Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, “Autonomous driving in urban environments: Boss and the Urban Challenge,” *J. Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.
19. G. Erico, “How Google’s Self-Driving Car Works,” *IEEE Spectrum*, vol. 18, 2013.
20. A. B. Hillel, R. Lerner, D. Levi, and G. Raz, “Recent progress in road and lane detection: a survey,” *Machine Vision and Applications*, vol. 25, no. 3, pp. 727–745, Feb. 2012.
21. S. Yenikaya, G. Yenikaya, and E. Düven, “Keeping the Vehicle on the Road - A Survey on On-Road Lane Detection Systems,” *ACM Computing Surveys*, vol. 46, no. 1, p. 2, Oct. 2013.
22. V. Umamaheswari, S. Amarjyoti, T. Bakshi, and A. Singh, “Steering angle estimation for autonomous vehicle navigation using hough and Euclidean transform,” in *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, 2015, pp. 1–5.
23. J. M. Collado, C. Hilario, A. de la Escalera, and J. M. Armingol, “Adaptative Road Lanes Detection and Classification,” in *Advanced Concepts for Intelligent Vision Systems*, J. Blanc-Talon, W. Philips, D. Popescu, and P. Scheunders, Eds. Springer Berlin Heidelberg, 2006, pp. 1151–1162.
24. L. Hua-jun, G. Zhi-bo, L. Jian-feng, and Y. Jing-yu, “A Fast Method for Vanishing Point Estimation and Tracking and Its Application in Road Images,” in *2006 6th International Conference on ITS Telecommunications Proceedings*, 2006, pp. 106–109.
25. P. P. Thittaporn Ganokratanaa, “A Hough Transform Based Lane Detection for Driving System,” 2013.
26. S. Zhou, Y. Jiang, J. Xi, J. Gong, G. Xiong, and H. Chen, “A novel lane detection based on geometrical model and Gabor filter,” in *2010 IEEE Intelligent Vehicles Symposium (IV)*, 2010, pp. 59–64.
27. T. Sun, S. Tang, J. Wang, and W. Zhang, “A robust lane detection method for autonomous car-like robot,” in *2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, 2013, pp. 373–378.
28. Y. Jiang, F. Gao, and G. Xu, “Computer vision-based multiple-lane detection on straight road and in a curve,” in *2010 International Conference on Image Analysis and Signal Processing (IASP)*, 2010, pp. 114–117.
29. G. Pannu, M. Ansari, and P. Gupta, “Design and Implementation of Autonomous Car using Raspberry Pi,” *International Journal of Computer Applications*, vol. 113, no. 9, 2015.
30. D. Seo and K. Jo, “Inverse Perspective Mapping based road curvature estimation,” in *2014 IEEE/SICE International Symposium on System Integration (SII)*, 2014, pp. 480–483.

31. J. Wang, F. Gu, C. Zhang, and G. Zhang, “Lane boundary detection based on parabola model,” in *2010 IEEE International Conference on Information and Automation (ICIA)*, 2010, pp. 1729–1734.
32. S. Habib and M. A. Hannan, “Lane Departure Detection and Transmisson using Hough Transform Method,” *Przegląd Elektrotechniczny*, vol. R. 89, nr 5, pp. 141–146, 2013.
33. J. Wang, Y. Wu, Z. Liang, and Y. Xi, “Lane detection based on random hough transform on region of interesting,” in *2010 IEEE International Conference on Information and Automation (ICIA)*, 2010, pp. 1735–1740.
34. M. Aly, “Real time detection of lane markers in urban streets,” in *2008 IEEE Intelligent Vehicles Symposium*, 2008, pp. 7–12.
35. A. A. M. Assidiq, O. O. Khalifa, R. Islam, and S. Khan, “Real time lane detection for autonomous vehicles,” in *International Conference on Computer and Communication Engineering, 2008. ICCCE 2008*, 2008, pp. 82–88.
36. S. Tan and J. Mae, “Real-Time Lane Recognition and Position Estimation for Small Vehicle,” *Internetwokring Indonesia Journal*, vol. 5, no. 2, 2013.
37. H. Lin, H. Kim, C. Lin, and L. O. Chua, “Road Boundary Detection Based on the Dynamic Programming and the Randomized Hough Transform,” in *International Symposium on Information Technology Convergence, 2007. ISITC 2007*, 2007, pp. 63–70.
38. P. M. Daigavane and P. R. Bajaj, “Road Lane Detection with Improved Canny Edges Using Ant Colony Optimization,” in *2010 3rd International Conference on Emerging Trends in Engineering and Technology (ICETET)*, 2010, pp. 76–80.
39. K. Ghazali, R. Xiao, and J. Ma, “Road Lane Detection Using H-Maxima and Improved Hough Transform,” in *2012 Fourth International Conference on Computational Intelligence, Modelling and Simulation (CIMSiM)*, 2012, pp. 205–208.
40. W. Xiaoyun, W. Yongzhong, and W. Chenglin, “Robust lane detection based on gradient-pairs constraint,” in *Control Conference (CCC), 2011 30th Chinese*, 2011, pp. 3181–3185.
41. J. Huang, H. Liang, Z. Wang, T. Mei, and Y. Song, “Robust lane marking detection under different road conditions,” in *2013 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2013, pp. 1753–1758.
.42. M. P. Batista, P. Y. Shinzato, D. F. Wolf, and D. Gomes, “Lane Detection and Estimation using Perspective Image,” in *2014 Joint Conference on Robotics: SBR-LARS Robotics Symposium and Robocontrol (SBR LARS Robocontrol)*, 2014, pp. 25–30.
43. A. Borkar, M. Hayes, and M. T. Smith, “Robust lane detection and tracking with ransac and Kalman filter,” in *2009 16th IEEE International Conference on Image Processing (ICIP)*, 2009, pp. 3261–3264.
44. P. Corke, *Robotics, Vision and Control*. Springer, 2011.
45. K. McFall and D. Tran, “Visual Lane Detection Algorithm Using the Perspective Transform,” in *Proceedings of the 14th Early Career Technical Conference*, 2014, vol. 13.
46. P. Corke, “Machine Vision Toolbox for MATLAB Release 3,” Jan-2015. [Online]. Available: http://www.petercorke.com/MVTB/vision.pdf. [Accessed: 12-Mar-2015].
47. K. Pulli, A. Baksheev, K. Kornyakov, and V. Erumihov, “Real-Time Computer Vision with OpenCV,” *Communications of the ACM*, vol. 55, no. 6, pp. 61–69, Jun. 2012.
48. S. Brahmbhatt, “Embedded Computer Vision: Running OpenCV Programs on the Raspberry Pi,” in *Practical OpenCV*, Apress, 2013, pp. 201–218.
49. Pan Zhao, Jiajia Chen, Yan Song, Xiang Tao, Tiejuan Xu, and Tao Mei, “Design of a Control System for an Autonomous Vehicle Based on Adaptive-PID,” *International Journal of Advanced Robotic Systems*, vol. 9, pp. 1–11, Jul. 2012.
50. T. Shu, Y. Zheng, and Z. Shi, “Image processing-based wheel steer angle detection,” *Journal of Electronic Imaging*, vol. 22, no. 4, p. 043005, 2013.