Adaptive HSL Filters and Inverse Perspective Transforms in Lane Detection for Autonomous Driving*

Hannah Mason[1], Landon Bentley[2], Joe MacInnes[3], Rahul Bhadani[4] and Tamal Bose[4]

***Abstract*— A wide variety of lane-detection algorithm relies on supervised machine learning algorithms that require intensive computational power. In a situation where computing power is limited, performance is degraded and/or strong assumptions are made for lane-detection. As a result, such algorithms don't work in generalized situations. The work presented here is an experience report of the lane detection algorithm for the autonomous driving experiment that describes how adaptive HSL (hue, saturation, and luminosity) filters are used in tandem with a perspective transform to detect lanes in a variety of lighting and quality conditions, without using supervised learning algorithms. This algorithm converts an RGB image input to an HSL image and uses the median brightness of the road to estimate how bright the lanes are. Once the lanes are extracted, they are fed into an inverse-perspective transform pipeline, giving a bird's eye view of the lanes, appeared in front of the car. The resulting image can be fed to lane following algorithms, Model-Predictive Controller block, or any generalized lane-following controller that requires a priori to be updated on the fly.**

I. Introduction

For full automation of self-driving vehicles, it is pertinent to note that a vehicle will require itself moving from point A to point B in an orderly fashion in par with traffic regulations and driving laws. This requires an accurate localization of vehicles in the order of centimeters with respect to the road and more specifically the lane on which it is supposed to drive. Even the most sophisticated instruments such as Global Navigation Satelite Systems with IMU fusion fail to provide such accuracy in some cases such as GPS-denied environment, city locations with high GPS-obstructions and due to other conditions such as multipath error and atmospheric bias [1]. This task of lane-keeping can be improved by lane-detection algorithms on the fly and making required corrections locally and returning results without significant delay. The presented work is an experience report that focuses on this last leg improvement in self-localization of the autonomous vehicle (AV) based on low-cost sensors that don't require intense computing power such as one required by supervised learning algorithms utilized for making predictions.

Contribution

The most notable part of this work is detecting lanes without any pre-trained features and demonstration of the algorithm in less than ideal situations such as unclear/faded lane-markings, curved roads and doesn't make any assumptions that road surface needs to be flat. This is important for testing an autonomous vehicle navigation scenario in the resource-constraint environment or where the pre-trained model of lane-feature is not suitable and/or available. The algorithm was tested in a series of experiment, as a part of Research Experience for Undergraduates (REU) program [2], with the University of Arizona self-driving testbed called as CAT Vehicle Testbed [3] on the city of Tucson's road in residential neighborhoods, parking lots and highways all having varying road features.

II. Related Work

It is trivial to say that lane-detection for end-to-end autonomous driving is not a new concept. However end-to-end driving is a multitude of tasks performed in a dynamic environment with a wide variety of constraints, all requiring the AV to perform computation in the regime of sample time of the vehicle control loop. The trivial task of end-to-end navigation goes back in the early 90s when it was more of a concept than implementation [4], limited to specific kind of test robots in laboratory environment [5] or assumptions had been made for a certain kind of lanes [6], [7]. We found an early work in characterizing statistical properties of lane-marking, done by Kluge and Johnson in [8]. They used the histogram of image intensity for bright lanes with a fairly bright background that resulted in a bimodal distribution. However, that was an early stage in the era of intelligent vehicles and no attempt for automated driving was shown. As computing power developed, researchers started using more computationally expensive solutions such as ALVINN, a neural-net based solution [9], [10] for autonomous driving. However, authors used simulated road generators to train neural-net and they mentioned that training on a real road is logistically challenging and must be presented with a large number of road conditions. Some other work in this area using learning algorithms [11]–[17] also suffer from similar kind of problem of training algorithms with a large number of datasets that are logistically and financially inefficient. Later in the millennium, academia and industry saw a surge in the autonomous driving research that was fuelled by DARPA

*This research is supported by the National Science Foundation as a part of CAT Vehicle Research Experience for Undergraduates (REU) program through award 1659428.

[1]Hannah Mason is with Department of Electrical & Computer Engineering, Lipscomb University in Nashville, Tennessee, USA. hgmason@mail.lipscomb.edu

[2]Landon Bentley is with Department of Computer Science, The University of Alabama, Tuscaloosa, Alabama, USA. lcbentley@crimson.ua.edu

[3]Joe MacInnes is with Department of Computer Science, The College of Wooster, Wooster, Ohio, USA. jmacinnes19@wooster.edu

[4]Rahul Bhadani and Tamal Bose are with Department of Electrical & Computer Engineering, The University of Arizona, Tucson, Arizona, USA. {rahulbhadani, tbose}@email.arizona.edu

Urban Challenge in 2007. One way to follow lane during the grand challenge was to follow lane magnets as mentioned in [18]. CMU's BOSS vehicle in the urban challenge used forward-looking LiDAR to detect road-edge detection for developing the lane-keeping system. However, this kind of method of detecting road-edge may not be very beneficial when a road consists of multiple lanes. Nevertheless, their method obtained promising results for a single lane with the feature extraction and classification algorithms. Stanford's Junior was using RNDF (Road Network Data File, containing a digital map of the road network coupled with aerial imagery) provided as a part of the competition with modification in lane-detection based on probabilistic measure. For vehicle localization with respect to the road, their vehicle was using LiDAR and infra-red sensors. Due to prior information about road networks in terms of RNDF file, the problem of self-localization of the vehicle was simpler during the Urban Challenge. In commercial vehicles, lane-detection and self-localization need to be carried using affordable sensors, most common of them are camera-based. Combining this with a need to have an extremely low false alarm rate for lane-departure, even after more than a decade of Urban Grand Challenge, lane-detection and following, the problem is still not trivial. A detailed background of lane marking detection using different sensing modalities can be found in [19]. A common approach for lane-marking detection has been the use of inverse perspective transform. In [20], authors demonstrated the use of perspective transform on synthetic images for lane-detection. In [21]–[23], results were demonstrated to work with real road images. Although findings of this method presented in these papers were satisfactory, when we implemented these methods with our autonomous vehicle Testbed, we found out that the algorithm faltered upon providing input of faded lanes and/or when there were other vehicles of yellow and white colors. A recent implementation of similar kind of algorithms discussed in [24], [25] also suffered from the same problems. A related implementation for lane detection has been provided by researchers from the University of Utah [26]. However, their algorithm produced a surface plot, not an image, which prevents further processing. We found out that it took approximately 16 seconds to produce an output image on a regular personal computer with a processing power equivalent of Intel Corei5. Our project mentioned in this paper is built on the top of these two works to create a more efficient algorithm that is capable of working in faded lanes, curved lanes and robust to visual noise captured by camera due to external agents such other moving vehicles.

Rest of the paper is organized as follows: Section III provides preliminaries for inverse perspective transform. Section IV describes our approach followed by implementation and any issues we encountered. In the end, we provide conclusion and lessons learned as a part of this project.

III. Mathematical Preliminaries

The angle of view under which the scene is acquired from the front camera mounted on the car suffers from perspective transformation. This mandate the vision engineer to design their lane-detection algorithm to take into account the perspective transform effect when processing images in order to weigh each pixel according to its information content.

In the configuration where a front camera is mounted on the top of a vehicle, as shown in Figure 1, with a fixed angle θ with respect to longitudinal direction provides images that are perspective transformed. With this prior information, we can determine pixels appeared in the captured image with respect to the world coordinates to obtain a bird's eye view image. This transformation is known as the inverse-perspective transform.

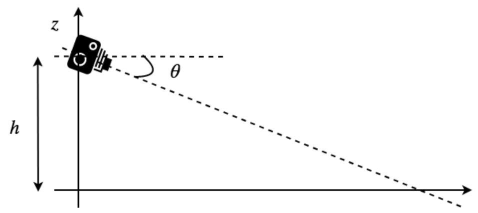

Fig. 1: An illustration of front camera mounted on the vehicle for lane image capture.

Let's assume in camera's coordinate system, each pixel is represented by $\vec{p_c}$. Inverse transform mapping finds a transform T that transforms every pixel $\vec{p_c}$ into world coordinates (i.e. bird's eye view) that we call $\vec{p_w}$:

$$\vec{p_w} = T \cdot \vec{p_c} \tag{1}$$

While mathematics behind the inverse perspective transform can be found in [27], we took a different approach to calculate the inverse perspective of the image. We calculated $M = T^{-1}$ a transformation that converts a bird's eye image to perspective image using control points (specified in terms of source and destination matrix). With the help of this transformation matrix, we use warping to convert perspective images to bird's eye images.

In order to so, we first identify the region of interest (ROI) on the acquired image. The region of interest is specified on a unit square as shown in Figure 2. We use four coordinate points (A_1, A_2, A_3, A_4) to specify the ROI. Let's call them the source matrix. We also specify four coordinate points (P_1, P_2, P_3, P_4) for bird's eye view image, again on the unit square. Let's call them destination matrix. Every point, e.g. A_1 has x and y coordinate such that $A_1 \equiv (A_{1x}, A_{1y})$. Based on these two sets of coordinate points, we calculate a transformation matrix that is used to warp perspective image into a bird's eye image. We use acquired image's original size as a scaling factor for source matrix, while a fractional scaling of acquired image's original size for destination matrix. We explain it in detail in the following paragraphs.

Let the acquired image's height and width are h_y and h_x. Further, let source matrix be

$$\mathcal{S} \equiv \{(A_{ix}, A_{iy})\} \quad i \in [1, 4]$$

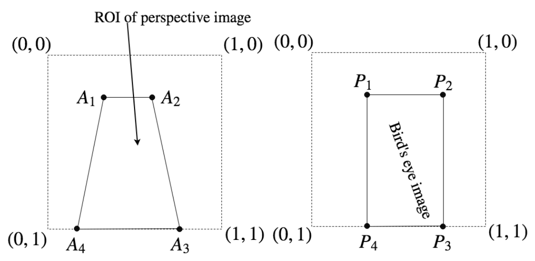

Fig. 2: Source and destination matrices for calculating transformation matrix for warping the perspective image to bird's eye image.

and destination matrix be

$$\mathcal{D} \equiv \{(P_{ix}, P_{iy})\} \quad i \in [1, 4].$$

Let's denote dashed letters for scaled matrices e.g.

$$\mathcal{S}' \equiv (A'_{1x}, A'_{1y}) \equiv (h_x \cdot A_{1x}, h_y \cdot A_{1y}) \tag{2}$$

As mentioned earlier, we use fractional scaling for destination matrix:

$$\mathcal{D}' \equiv (P'_{1x}, P'_{1y}) \equiv (\rho h_x P_{ix}, \gamma h_y P_{iy}) \tag{3}$$

where ρ and γ are fraction of acquired image dimension we want to use to scale the destination matrix before computing transformation matrix.

Let's assume M be the transformation matrix that converts bird's eye to a perspective image:

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \tag{4}$$

For perspective transformation, we have $m_{33} = 1$. A perspective transformation is obtained as follows [28]:

$$\begin{aligned} P_{xi} &= \frac{m_{11}A_{xi} + m_{12}A_{yi} + m_{13}}{m_{31}A_{xi} + m_{32}A_{yi} + m_{33}} \\ P_{yi} &= \frac{m_{21}A_{xi} + m_{22}A_{yi} + m_{23}}{m_{31}A_{xi} + m_{32}A_{yi} + m_{33}} \end{aligned} \tag{5}$$

Since, we identify (A_{xi}, A_{yi}) and (P_{xi}, P_{yi}) first, we are required to calculate M. This is done by solving a system of linear equations:

$$\begin{bmatrix} A_{x1} & A_{y1} & 1 & 0 & 0 & 0 & -A_{x1}P_{x1} & -A_{y1}P_{x1} \\ A_{x2} & A_{y2} & 1 & 0 & 0 & 0 & -A_{x2}P_{x2} & -A_{y2}P_{x2} \\ A_{x3} & A_{y3} & 1 & 0 & 0 & 0 & -A_{x3}P_{x3} & -A_{y3}P_{x3} \\ A_{x4} & A_{y4} & 1 & 0 & 0 & 0 & -A_{x4}P_{x4} & -A_{y4}P_{x4} \\ 0 & 0 & 0 & A_{x1} & A_{y1} & 1 & -A_{x1}P_{y1} & -A_{y1}P_{y1} \\ 0 & 0 & 0 & A_{x1} & A_{y1} & 1 & -A_{x1}P_{y1} & -A_{y1}P_{y1} \\ 0 & 0 & 0 & A_{x2} & A_{y2} & 1 & -A_{x2}P_{y2} & -A_{y2}P_{y2} \\ 0 & 0 & 0 & A_{x2} & A_{y2} & 1 & -A_{x2}P_{y2} & -A_{y2}P_{y2} \\ 0 & 0 & 0 & A_{x3} & A_{y3} & 1 & -A_{x3}P_{y3} & -A_{y3}P_{y3} \\ 0 & 0 & 0 & A_{x4} & A_{y4} & 1 & -A_{x4}P_{y4} & -A_{y4}P_{y4} \end{bmatrix} \begin{bmatrix} m_{11} \\ m_{12} \\ m_{13} \\ m_{21} \\ m_{22} \\ m_{23} \\ m_{31} \\ m_{32} \end{bmatrix} = \begin{bmatrix} P_{x1} \\ P_{x2} \\ P_{x3} \\ P_{x4} \\ P_{y1} \\ P_{y2} \\ P_{y3} \\ P_{y4} \end{bmatrix} \tag{6}$$

Then every pixel $(C_x, C_y) \in \vec{p_c}$ can be transformed to bird's eye view pixel $(W_x, W_y) \in \vec{p_w}$ using the warp transformation:

$$(W_x, W_y) = \left(\frac{m_{11}C_x + m_{12}C_y + m_{13}}{m_{11}C_x + m_{12}C_y + m_{13}}, \frac{m_{21}C_x + m_{22}C_y + m_{23}}{m_{11}C_x + m_{12}C_y + m_{13}} \right) \tag{7}$$

IV. Approach and Implementation

In this section, we detail our approach and implementation of inverse perspective transform used for lane detection. We begin with color filtering required for adaptive HSL in order to identify lanes.

A. *Color Filtering with Adaptive HSL Thresholds*

First, we converted the acquired image in RGB color-space to an HSL image. For implementation, we used OpenCV's `inRange` function in python to extract the lanes using upper and lower limits on the HSL channels of the image. While yellow lanes mostly have the same hue, even when faded or in the shade, the lightness of white lanes can change drastically. As mentioned earlier in the section II, previous works such one mentioned in [24], couldn't identify faded lanes or lanes in the shade. When we reduced the lower lightness limit to account for faded lanes, the algorithm would struggle with images with clearly defined lanes. Bright parts of the road that weren't actually lanes were being registered as lane markings. By altering the limit to account for faded lanes, it no longer worked well for clearly defined lanes - a problem we wanted to fix.

Thus a better method was needed to alter the lower lightness limit based on the overall lightness of the image to reduce the false positive. We performed an analysis on a collection of images with faded and non-faded lanes, the results of which are below in Table 1. In this analysis, we calculated the median lightness of each picture and noted down the average lightness for each category. Since the sky and car hood tended to throw off the lightness median, we cropped out two fifths and the lower fifth part of the image prior to their analysis.

Category	**Average of the Median Lightness**
Well Defined Lanes	107.305
Faded Lanes	73.256

TABLE I: The average of the median brightness values for the images in each category.

While the average lightness produces a similar result to the median, some experimentation revealed the median lightness to be a more accurate reference for troublesome images. For example, a completely white or completely black car in the image has a greater impact on the average lightness of the image than on the median lightness. Thus the median lightness was more reliable for our purposes of identifying lane markings.

The images with faded lanes had significantly lower median lightness values than the images with clearly defined lanes. After some trial and error, 1.8 times the median was found to be an optimal lower lightness threshold for bright images and 1.5 times the median was chosen to be a good lower threshold for the faded and dark images. Images with a median lightness of less than 75 were treated as faded, otherwise, they were treated as bright. We also noted that in an image with bright lanes, the lanes were more saturated than the rest of the image. However, when the lanes were faded, they were frequently the least saturated part of the image. Thus, we used a lower saturation bound of 90 for identifying bright lines whereas a higher saturation bound of 90 for darker images. Examples of this method (after applying inverse perspective transform, discussed in IV-B) can be seen in Figure 3 and Figures 4 to 6.

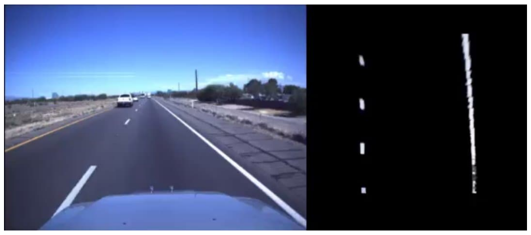
Fig. 3: The application of the adaptive color filtering on a bright image.

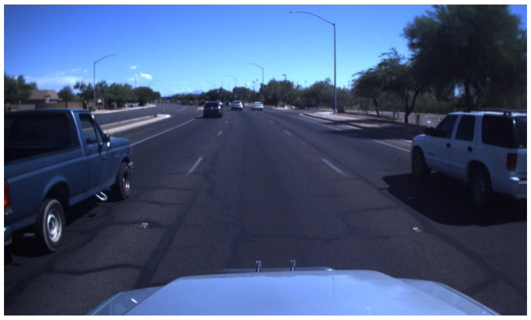
Fig. 4: An image with faded and curved lanes.

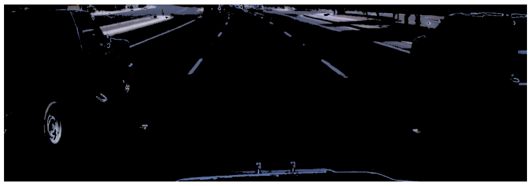
Fig. 5: The adaptive HSL filters applied to Figure 4. While there is still some visual noise in the image, the majority of the road that is not lane markings is masked out. When the inverse perspective transform is applied, the rest of the visual cancels out.

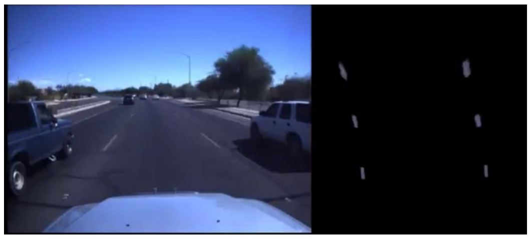
Fig. 6: The final result of the algorithm applied on the image in Figure 4. Note that the lanes are accurately detected and the algorithm was also able to capture the slight curve of the road.

B. Applying Inverse Perspective Transform

Previous works discussed in section II were unable to handle curved lanes. Additionally, the program left unwanted noise in the image which was difficult to consistently filter out. We reproduced inverse perspective transformation based on previous results in MATLAB and can be seen in Figure 7, Figure 8, and Figure 9.

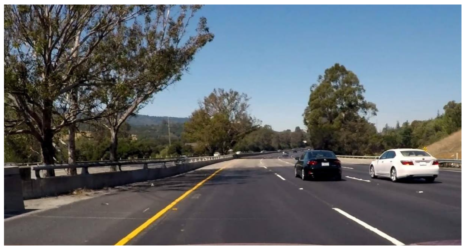
Fig. 7: A sample image to be run through the MATLAB algorithm.

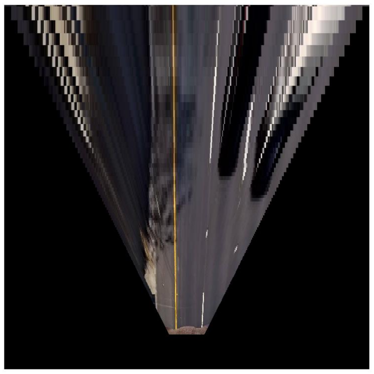
Fig. 8: The perspective transform of the image in Figure 7.

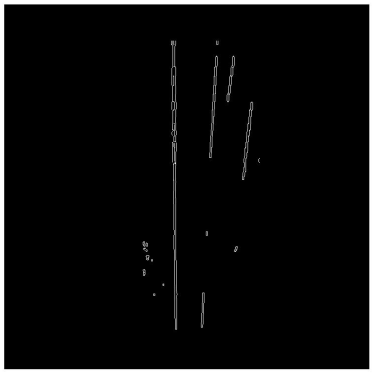
Fig. 9: The result of the color filters and Canny edge detection on the image in Figure 8. Note the amount of white pixels in the image that are not caused by the lane, and that it detects lanes that the car is not in.

Choosing Mapping Points to Calibrate the Perspective Transform: Previous works in this direction were dependent on the camera angle. Each time the camera is put on the car, the source and destination matrix as discussed in section III must be adjusted accordingly. If the camera is always mounted on the car, this is not an issue after the first calibration. But if the camera is constantly being taken down and remounted on the car, this becomes somewhat of an issue, as experienced frequently during Field Operation Tests

(FOTs). To calibrate, the car was put at one end of a straight lane, that was at least 40 meters long and 3.7 meters wide, which matches interstate regulations for lane width in the United States. An image was taken from the camera and 4 points were chosen as the source matrix. Two points were about 40 meters in front of the car, slightly to the right and left of the lanes. Two points were right above the hood of the car, also slightly to the right and left of the lanes. When applying the inverse perspective transform, they should result in a straight-looking lane. The points were adjusted until that was the case. This process can be seen in Figure 10, where the corners of the trapezoid represent the four points chosen. Figure 11 is an image taken from the same camera position, but where the lanes are farther to the side than they were in the calibration photo. Note that the perspective transform still works on this image. Figure 12 also shows that this calibration works on curved roads.

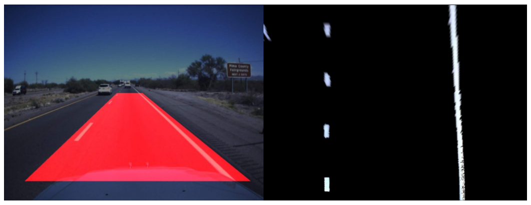

Fig. 10: An ideal calibration image, where the trapezoid corners represent the four points for the source transform. The left side is the original image with the trapezoid overlaid, and the right side is the resulting perspective transform. Note how the lanes are straight in the perspective transform.

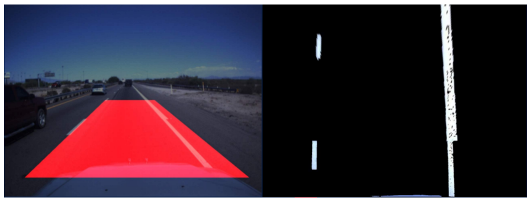

Fig. 11: The perspective transform, calibrated in Figure 10, applied to another image from the same camera position. Note that although the lanes are not fully encapsulated in the trapezoid, the lanes are still detected and shown in the perspective transform.

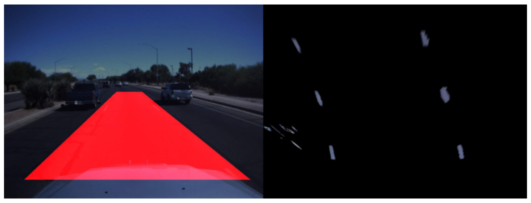

Fig. 12: The perspective transform, calibrated in Figure 10, applied to another image from the same camera position. Note that the curve of the road is maintained in the perspective transformed image.

Curvature in the Road: Once the camera is calibrated, it can handle some variances in lane width and curvature of the road, as seen in Figure 11 and Figure 12. Unlike the previous work, which maps the points to the four corners of the destination image, our program maps the points to slightly inside the four corners of the destination image. This slight change allows the program to handle curves in the road efficiently, without allowing noise in the image.

Scaling: Note that the resulting image seems to squash the y-axis down. We found out this was a nuisance when translating the detected lanes from the resulting bird's eye image to physical waypoints for lane-following [29]. Thus scaling coefficients ρ and γ as mentioned in Equation (3) had to be calculated for use with a physical car. We used a calibration image, seen in Figure 13 to determine a suitable value of ρ and γ, where we had marked fixed increments on the road in front of the car. The perspective transform of this image can be seen in Figure 14.

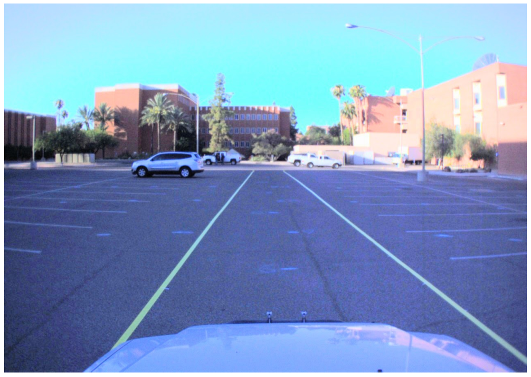

Fig. 13: An image used for finding the scaling coefficients. Note the chalks marks on the ground in front of the vehicle in 3 meter intervals.

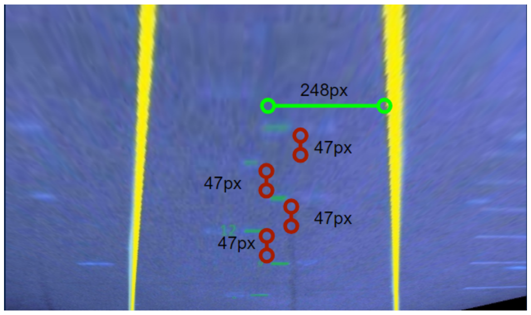

Fig. 14: The perspective transform of the image in Figure 9. The chalk marks and lanes were drawn over digitally before the transform, to make them easier to read in the resulting image.

From Figure 14, we found that the scaling coefficients were about 47 pixels per 3 meters vertically, and 248 pixels per 1.85 meters horizontally. Note that these scaling coefficients will vary with image dimensions. While these scaling coefficients will change each time the camera is calibrated, we found that the change was small and was able to be quickly corrected by trial and error, rather than requiring an entirely new calibration image.

C. Using the Final Image for Lane Following

After calibrating our algorithms, we finalized source matrix and destination matrix as

$$\mathcal{S} \equiv \{(0.565, 0.527), (0.463, 0.527), (.12, .9), (.91, .9)\}$$

$$\mathcal{D} \equiv \{(.8, .2), (.2, .2), (.2, .8), (.8, .8)\}$$

with $\rho = 0.8$ and $\gamma = 1.0$. The full pipeline for obtaining bird's eye image with clear lane-markings is shown in Figure 15. Ideally, our method should be used with a car with a fixed camera. While suitable results were accomplished with the University of Arizona's CAT Vehicle, which required the camera to be remounted frequently, a fixed camera should provide more consistency. The resulting bird's eye images obtained from this method were given to another program which fits a series of waypoints from the extracted lanes; see our work [29] and [30] for more detail. These waypoints were scaled using the aforementioned coefficients and sent to a steering controller to drive the car. A live demo was held where the car was successfully able to follow a curved lane. We released a video reporting the overall experience of the REU program along with the results from this paper in [30].

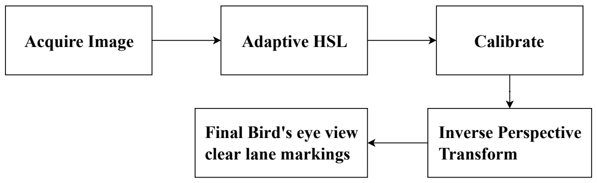
Acquire Image
Adaptive HSL
Calibrate
Inverse Perspective Transform
Final Bird's eye view clear lane markings

Fig. 15: Overall pipeline used for obtaining bird's eye view of clear lane marking.

V. Conclusion and Future Works

In this work, we present a proof-of-concept for real-time lane detection in autonomous driving with low computation overheads. Our method doesn't require any training, testing and is remarkably good for situations that do not require mounting-remounting of the front camera (which is more likely to the case with commercial vehicles). It works well with curved lanes as well as faded lanes. As a part of the future work, we are hoping to extend this work to a scenario where there are no lane-markings present on the road.

Acknowledgement

This research was supported by the National Science Foundation under award 1659428 and 1446435. Additionally, we would like to thank Nancy Emptage for logistics and facilitating the program.

References

[1] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*. Artech house, 2005.
[2] National Science Foundation, " REU Site: Cognitive and Autonomous Test Vehicles," 2018. [Online]. Available: https://www.nsf.gov/awardsearch/showAward?AWD_ID=1659428
[3] R. Bhadani, J. Sprinkle, and M. Bunting, "The CAT Vehicle Testbed: A Simulator with Hardware in the Loop for Autonomous Vehicle Applications," in *Proceedings 2nd International Workshop on Safe Control of Autonomous Vehicles (SCAV 2018), Porto, Portugal, Electronic Proceedings in Theoretical Computer Science*, vol. 269, 2018.
[4] E. Dickmanns, "Computer vision in road vehicles–chances and problems," in *ITCS-Symposium on Human Factors Technology for Next-Generation Transportation Vehicles, Amalfi, Italy*, 1986.
[5] H. Frohn and W. Von Seelen, "Visocar: An autonomous industrial transport vehicle guided by visual navigation," in *Robotics and Automation, 1989. Proceedings., 1989 IEEE International Conference on*. IEEE, 1989.
[6] G. Adorni, M. Bertozzi, and A. Broggi, "Massively parallel road/lane detection," in *Proc. th Jut. Conference on Applications ofAdvanced Technologies in Transportatzon Engmeermg AATTE,(Capri, Italy)*, 1995.
[7] J. Manigel and W. Leonhard, "Computer control of an autonomous road vehicle by computer vision," in *Industrial Electronics, Control and Instrumentation, 1991. Proceedings. IECON'91., 1991 International Conference on*. IEEE, 1991.
[8] K. Kluge and G. Johnson, "Statistical characterization of the visual characteristics of painted lane markings," in *Intelligent Vehicles' 95 Symposium., Proceedings of the*. IEEE, 1995.
[9] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, 1989.
[10] T. Jochem, "Vision based tactical driving," Ph.D. dissertation, Carnegie Mellon University, 1996.
[11] Z. Sun, G. Bebis, and R. Miller, "On-road vehicle detection using gabor filters and support vector machines," in *14th International Conference on Digital Signal Processing, 2000*, vol. 2. IEEE.
[12] X. Wen, L. Shao, Y. Xue, and W. Fang, "A rapid learning algorithm for vehicle classification," *Information Sciences*, vol. 295, 2015.
[13] D. Lieb, A. Lookingbill, and S. Thrun, "Adaptive road following using self-supervised learning and reverse optical flow." in *Robotics: science and systems*, 2005.
[14] S.-Y. Oh, J.-H. Lee, and D.-H. Choi, "A new reinforcement learning vehicle control architecture for vision-based road following," *IEEE Transactions on Vehicular Technology*, vol. 49, no. 3, 2000.
[15] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015.
[16] M. Bojarski, D. Del Testa, D. Dworakowski, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.
[17] R. Gopalan, T. Hong, and R. Chellappa, "A learning approach towards detection and tracking of lane markings," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, 2012.
[18] U. Ozguner, C. Stiller, and K. Redmill, "Systems for safety and autonomous behavior in cars: The darpa grand challenge experience," *Proceedings of the IEEE*, vol. 95, no. 2, 2007.
[19] 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, 2014.
[20] Y. Shu and Z. Tan, "Vision based lane detection in autonomous vehicle," in *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, vol. 6. IEEE, 2004.
[21] M. Bertozzi and A. Broggi, "Gold: A parallel real-time stereo vision system for generic obstacle and lane detection," *IEEE transactions on image processing*, vol. 7, no. 1, 1998.
[22] Y. Wang, E. K. Teoh, and D. Shen, "Lane detection and tracking using b-snake," *Image and Vision computing*, vol. 22, no. 4, 2004.
[23] K.-Y. Chiu and S.-F. Lin, "Lane detection using color-based segmentation," in *Intelligent Vehicles Symposium, 2005. Proceedings. IEEE*. IEEE, 2005.
[24] K. Ficici, *Simple Lane Detection*, Hackster.io, 2018. [Online]. Available: https://web.archive.org/web/20190112194639/https://www.hackster.io/kemfic/simple-lane-detection-c3db2f
[25] *Curved Lane Detection*, Hackster.io, 2018. [Online]. Available: https://web.archive.org/web/20190112194729/https://www.hackster.io/kemfic/curved-lane-detection-34f771
[26] E. Johnson and R. Hamburger, *Computer Vision Class Project*, University of Utah, 2007. [Online]. Available: https://web.archive.org/web/20190112195129/https://pubweb.eng.utah.edu/~hamburge/
[27] H. A. Mallot, H. H. Bülthoff, J. Little, and S. Bohrer, "Inverse perspective mapping simplifies optical flow computation and obstacle detection," *Biological cybernetics*, vol. 64, no. 3, 1991.
[28] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.
[29] L. Bentley, J. MacInnes, R. Bhadani, and T. Bose, "A pseudo-derivative method for sliding window path mapping in robotics-based image processing." Tucson, Arizona: CAT Vehicle Research Experience for Undergraduates, 2019. [Online]. Available: http://dx.doi.org/10.13140/RG.2.2.35628.10885
[30] H. Mason, J. MacInnes, and L. Bentley, *Hannah Mason CAT Vehicle REU 2018*, 2018. [Online]. Available: https://www.youtube.com/watch?v=fAslkEbnBQI