Obstacle detection using stereo vision for self-driving cars

Naveen Appiah
Department of Mechanical Engineering
Stanford University
Email: nappiahb@stanford.edu

Nitin Bandaru
Department of Mechanical Engineering
Stanford University
Email: nbandaru@stanford.edu

***Abstract*—Perceiving the surroundings accurately and quickly is one of the most essential and challenging tasks for autonomous systems such as self-driving cars. The most common sensing systems such as RADAR and LIDAR that are used for perception on self-driving cars today give a full 360° view to the car making it more informed about the environment than a human driver. This article presents a methodology to employ two 360° cameras to perceive obstacles all around the autonomous vehicle using stereo vision. Using vertical, rather than horizontal, camera displacement allows the computation of depth information in all viewing directions, except *zenith* and *nadir* which have the least useful information about the obstacles. The Key idea for obstacle detection is to classify points in the 3D space based on height, width and traversable slope relative to the neighbouring points. The detected obstacle points can be mapped onto convenient projection planes for motion planning.**

I. Introduction

Computer vision is one of the toughest problems in Artificial Intelligence (AI) that has been challenging researchers and engineers for decades. This problem of extracting useful information from images and videos finds application in a variety of fields such as robotics, remote sensing, virtual reality, industrial automation, etc. The concept of making cars drive by themselves has gained immense popularity today in the AI world, mainly motivated by the number of accidents that occur due to driver errors/ negligence. Autonomous vehicles are designed to sense their surroundings with techniques such as RADAR, LIDAR, GPS and computer vision. This array of sensors working coherently to observe and record the surroundings constitute the perception module of the vehicle. The next stage in the pipeline is the localization step which stitches together the incomplete and disconnected information obtained from the sensors to identify the position, velocity and other states of the vehicle and the obstacles (including dynamic obstacles). The final module is the planning stage where the vehicle has to decide what it has to do given the situation it is in. The present day research prototypes built by major players in the industry/ academia have LIDAR and RADAR as their primary perception systems. They generally provide a very accurate full 360° view to the vehicle making it more informed about the environment than a normal human driver. The downside to these systems is the cost involved in deploying them. So an option is to use cameras and computer vision techniques to substitute these systems.

It has been shown in several cases that stereoscopic vision can be applied to extract useful information about the surroundings that could assist the navigation of mobile robots [1], [2], [3]. But in most cases the field of view is limited to just the front of the robot. It is very critical that the vision system we are targeting doesn't compromise on obtaining the 360° view provided by LIDAR / RADAR systems. This can be achieved by employing special cameras that capture 360° by 180° image of a scene.

II. Related Work

It is very essential for an autonomous vehicle to accurately and reliably perceive and discriminate obstacles in the environment. To this end, many approaches have been presented for different application areas and scenarios in past years using stereo vision or 2D/3D sensor technologies. Each obstacle detection system is focused on a specific tessellation or clustering strategy, hence they have been categorized into 4 main models [4]: (*i*) probabilistic occupancy map, (*ii*) digital elevation map, (*iii*) scene flow segmentation and (*iv*) geometry-based clusters.

In *probabilistic occupancy maps*, the world is represented as a rigid grid of cells containing a random variable whose outcome can be free, occupied, or undefined (not mapped) [5], [6]. The goal here is to compute the associated joint distribution depending on a set of measurements carried out on a certain discrete set of time moments. *Digital elevation maps* (DEM) is one of the algorithms that try to detect obstacles relying on the fact that they protrude up from a dominant ground surface. The obstacle detection algorithm proposed in [7] marks DEM cells as road or obstacles, using the density of 3D points as a criterion. It also involves fitting a surface model to the road surface. The *scene flow segmentation* or otherwise called as *optical flow* utilizes temporal correlation between different frames of a scene captured by stereo cameras to classify obstacles that are in motion [8], [9], [10]. This method thus naturally handles tracking dynamic obstacles. Finally the *geometry-based clustering* involves classification based on geometric structure of point clouds in the 3D space. The obstacle detection algorithm that will best suit this category is [11] which is based on a search method that clusters points using a double cone model. This algorithm became the basis for the obstacle detection module that went on the

intercontinental autonomous driving efforts by [12]. The real time performance of this approach is thoroughly evaluated in [13] and they were able to achieve a 10 Hz update rate with this approach.

The approach we present in this report is inspired by the method in [11]. Given our objective of obtaining a 360° view, we present a few modifications to this algorithm that will use 360° stereo pair to obtain obstacle information. The remainder of this report is structured as follows. Section 3 talks about the equipment setup that would best suit for this application. Section 4 describes the disparity and depth estimation technique used. Section 5 talks about the pixel-level implementation details of the obstacle detection algorithm. Section 6 discusses the experimental results on a few test scenes. Section 7 outlines possible next steps from this project.

III. Equipment set up

A pair of cameras are required to implement stereo vision. Since the application of this work is to replace existing LIDAR-based perception systems, the system developed in this work needs to detect obstacles all around the vehicle. Two Ricoh Theta[1] cameras were used to capture 360° by 180° spherical panoramas. The Ricoh Theta camera comprises of two opposite-facing 185° fish-eye lens and sensors. The two images are stitched by the Ricoh application to form a 360° by 180° spherical image.

The two cameras were displaced vertically as this would result in loss of information directly above and below the cameras, which are areas, not of interest to us. For the purposes of this work, the camera was mounted on a tripod at one position to capture the first image. A spirit level was used to ensure that the camera was horizontal. The camera was then vertically displaced through a known height on the tripod to the second position to capture the second image.

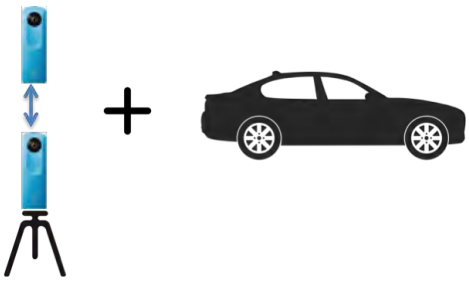


Fig. 1: Vertically displaced Ricoh Theta cameras mounted on a vehicle

IV. Depth Map Estimation

A. Image pair capture

The spherical images captured by the ricoh cameras are in equirectangular format i.e. they are uniformly smapled in azimuth and altitude angles. The azimuth variation is from $-\pi$ to π along the x-axis of the image and the altitude variation is from $-\pi/2$ to $\pi/2$ along the y-axis of the image. The images are of resolution 2048×1024 pixels.

Fig. 2: Image of the scene from the lower camera. It's a 360° by 180° spherical image in equirectangular format

B. Disparity estimation

The disparity for the spherical image pair is the change in the altitude angle $\Delta\theta$ between the two images, since the cameras are displaced vertically. There are several techniques to estimate disparity values such as those outlined in [14], [15], [16]. The procedure adopted in this work involves estimating the optical flow of each point. The optical flow estimation is done using the *optical flow software*[2] developed by [17]. As the cameras are vertically displaced, the optical flow is in the vertical direction. The optical flow at every pixel gives us the disparity value in number of pixels, at that pixel.

To speed up the disparity estimation step, the original image is down-sampled to 1022×512 pixels. To further speed up the computation, the top and bottom portions of the image which don't contain important information for the vehicle are chopped out. An altitude range of -30° to 30° is considered for the process. The resolution of the image so obtained is 1023×171 pixels. The image is further down-sampled to 767×129 pixels, on which the rest of the algorithm is run. The black and white image of the obstacle map obtained as the output of the obstacle detection algorithm is then up-sampled to 1023×171 pixels.

Fig. 3: Chopped and down sampled original lower image. The image is of resolution 767 by 129 pixels with an altitude range of -30° to 30°.

[1] https://theta360.com/en/

[2] http://cs.brown.edu/ black/code.html

Fig. 4: Disparity map generated from optical flow estimation. Brighter pixels correspond to greater disparity and so smaller depth. We observe uniform gradient of disparity along the ground plane, in the bottom half of the image.

C. Depth estimation

The next step is to determine the depth map. This is achieved through some basic geometry and trigonometric transformations applied to the disparity map.

$$\frac{l}{sin\Delta\theta} = \frac{PA}{sin(\frac{\pi}{2} + \theta_1)} \tag{1}$$

where l is the baseline distance, $\Delta\theta$ is the disparity value, θ_1 is the altitude angle of the point with respect to the upper camera center and PA is the distance from the point to the upper camera center. The depth is then calculated as

$$depth = \sqrt{PA^2 + (\frac{l}{2})^2 - 2 \cdot PA \cdot \frac{l}{2} \cdot cos(\frac{\pi}{2} + \theta_1)} \tag{2}$$

Using the above formulae, the depth value for every pixel can be calculated from the corresponding disparity value.

Fig. 5: Depth map generated from the disparity map. Darker pixels correspond to smaller depth.

V. Obstacle Detection

Geometry-based clustering technique has been proposed to detect obstacles in the scene. Before we dive into the obstacle detection algorithm, we need to come up with a definition for an obstacle. It is important to understand and visualize obstacles in the 3D space. Let us consider the most general case of obstacles found above a ground plane and focus our analysis on this case. We later discuss some of the other possibilities of obstacles or spaces in a scene that need to be avoided by a vehicle. Thus, obstacles are points or areas in the scene which are at a height from the dominant ground plane.

Mathematically, we will define obstacles in terms of two distinct points in space in the following way:
Two points P_1 and P_2 belong to the same obstacle and are said to be compatible if

1) $H_T < |P_{2Z} - P_{1Z}| < H_{max}$. The difference between the elevations of the two points is within a range defined by H_T and H_{max}.
2) $((P_2 - P_1) \cdot (P_3 - P_1)/||P_2 - P_1||||P_3 - P_1||) > cos\theta_T$. The point P_3 is obtained by displacing P_1 through H_{max} in the z direction. This condition enforces that the angle between P_2 and P_1 with respect to the z direction or elevation direction is less than a threshold value.

In the above definition, H_T is the minimum height of an object for it to be considered an obstacle. H_{max} is the maximum height of on obstacle. The value of θ_T can be set based on the accepted traversable slope to classify obstacles appropriately.

The definition is illustrated in Figure 6. We construct an upward cone from P_1 in the 3D space based on the values chosen for the three parameters H_T, H_{max} and θ_T. If any of the points in the scene lie in the frustum as shown, those points are classified as obstacle points.

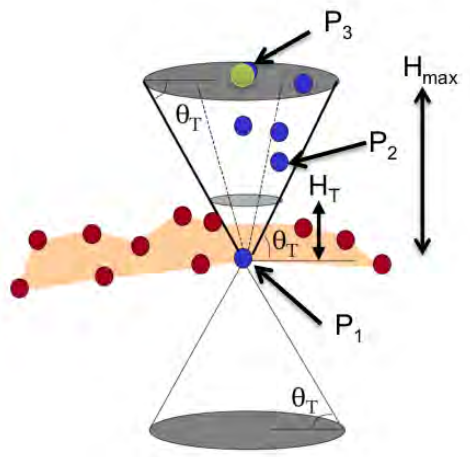


Fig. 6: Cone formed in the 3D space with P_1 at the center. The point P_2, if lies within the frustum formed by H_T, H_{max} and θ_T, is classified as an obstacle point. (Source: [11])

From the depth map generated before, we have information about the depth of all the pixels. All the points or pixels are now 3D points in Azimuth-Altitude-Depth space. The points are transformed from the Azimuth-Altitude-Depth space to the X-Y-Z space.

A naive algorithm would involve examining all point pairs which would have a complexity of $O(N^2)$. The algorithm proposed is more efficient and compares a point with a limited set of points. These set of points are the points contained within the trapezium formed by the projection of the frustum or cone formed above the point (referred to as *base point* from here on) in the 3D space, onto the image plane. The projected trapezium is scaled according to depth of the *base point*. The parameters of the trapezium are

$$h_T = \frac{H}{\pi} \cdot \frac{H_T}{depth} \tag{3}$$

where H is the height of the image and $depth$ is the depth of the *base point*. The first term $\frac{H}{\pi}$ is a scaling factor to transform

the height to the altitude angle and thus, the number of pixels. Similarly,

$$h_{max} = \frac{H}{\pi} \cdot \frac{H_{max}}{depth} \tag{4}$$

Here, h_T is height of the closer parallel side of the trapezium from the *base point* and h_{max} is height of the farther parallel side from the *base point*. The upper left angle of the trapezium is same as θ_T, the threshold angle chosen for the compatibility definition earlier.

We loop through all the points in the trapezium and form a point pair for each point with the base point. If the pair of points satisy the definition of obstacles i.e. are compatible, then we classify that point in the trapezium as an obstacle point. This algorithm has a better time complexity than the Naive alogrithm. Let K denote the average number of points in the trapezium. The complexity is $O(KN)$.

Algorithm:

- Classify all points as non-obstacles.
- Scan through all the pixels, P, in the image.
  - Determine the set of pixels, T_P, in the projected trapezium of P on the 2D image plane.
  - Examine all points in T_P and determine set O_P of points P_i in T_P, compatible with P.
  - If O_P is not empty, classify all points of O_P as obstacle points.

Fig. 7: Black and white image of the Obstacle map. The white pixels correspond to obstacle points.

A. Post-processing

Due to the lack of texture on the ground, we get fewer feature points in those regions and so the estimated disparity values are inaccurate. The disparity gradient is discontinuous in those regions on the ground, resulting in noise in the output. Median filtering and morphological closing operation are applied to the output of the obstacle detection algorithm to close the small holes.

Fig. 8: Black and white image of the Obstacle map after post-processing. The small holes in Figure 7 are filled.

Fig. 9: Original lower image with obstacles overlaid on it. The pixels in yellow correspond to obstacles.

B. Polar Map

The primary application of this work is to detect obstacles so as to plan the motion of the vehicle. It is important to represent the obstacle points in a manner that might be useful for motion planning. A polar map representation of the obstacle points is chosen. The polar map is generated by projecting the 3D obstacle points onto the ground plane. In figure 10, the vehicle is at the center of the circle facing $0°$. The obstacle points are depicted in blue with the radius of the circle indicating the depth from the vehicle. The small patch at $0°$, close to the center corresponds to the white sign board in the scene in figure 3 and the patch at $30°$ corresponds to the black platform. The buildings and trees can be seen further away in the map. From the polar map, we can plan a path for the vehicle, not obstructed by obstacles.

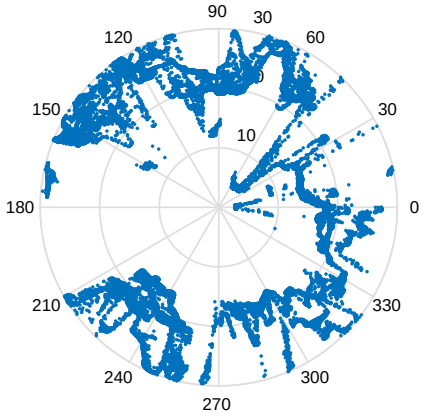


Fig. 10: The agent is at the center of the map, facing $0°$. The blue points correspond to polar positions of the obstacle points around the agent.

VI. Results

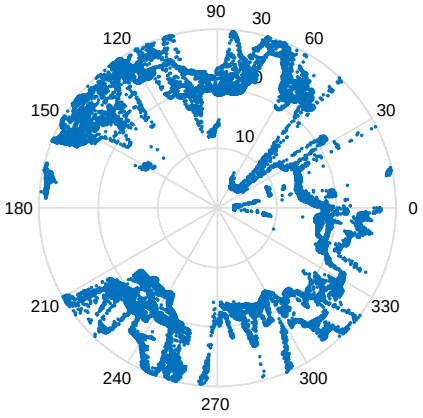
Figure 11 depcits a scene which has a good mix of obstacles at short range and long range. The algorithm detects all the obstacles in scene. The process was tested on various outdoor settings and gave positive results in all the cases. In figure 13, we can observe that the polar map representation accurately captures the cars parked next to each other at angles $315°$ to $35°$. Also, in figure 13, we notice small erroneous detection at $90°$ on the ground. This is due to the

(a)

(b)

(c)

(d)



(e)

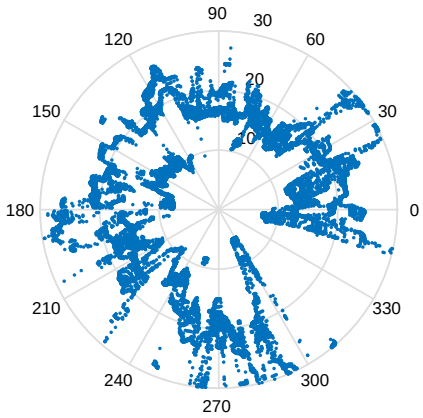
Fig. 11: Scene 1: (a) lower image; (b) depth map; (c) black and white image of obstacle map; (d) lower image with obstacles overlaid; (e) polar map representation of obstacles

(a)

(b)

(c)

(d)



(e)

Fig. 12: Scene 2: (a) lower image; (b) depth map; (c) black and white image of obstacle map; (d) lower image with obstacles overlaid; (e) polar map representation of obstacles

lack of texture on the ground, which results in inaccurate disparity estimation. Thus, the obstacle detection algorithm classifies it as an obstacle point. The accuracy of the obstacle detection is limited by the accuracy of disparity estimation techniques. The programs were developed in MATLAB[3]. The average runtime of disparity estimation is $125s$ and that of obstacle detection is $240s$.

It was mentioned earlier that obstacles above the ground were being considered for this work. But the process developed also works for potholes in the ground. When a point inside the hole is picked, the cone/frustum in the 3D space will contain points on the ground around the pothole. These points will be classified as obstacles which implies that these points will be avoided in the path planning process. Potholes are areas that need to be avoided and the algorithm does exactly that.

VII. Future Work

We have showed the working of an algorithm to detect obstacles. But the challenge always lies in running it real-time on the vehicle with a quick update rate so as to react quickly

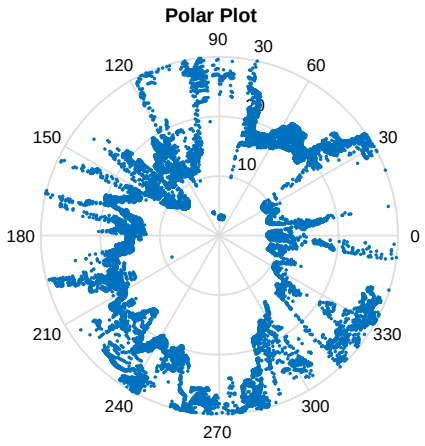
[3]http://www.mathworks.com/products/matlab/

(a)

(b)

(c)

(d)



(e)

Fig. 13: Scene 3: (a) lower image; (b) depth map; (c) black and white image of obstacle map; (d) lower image with obstacles overlaid; (e) polar map representation of obstacles

to changes in the environment. The first among our future steps should be optimizing the run time of this algorithm and possibly explore efficient platforms and parallel architecture if required to run it online. There are other quick depth-estimation algorithms [18], [16] in the literature that are more suitable for real time applications at the cost of reduced accuracy. So a sensible trade off has to be made on the choice of the depth estimation algorithm.

The obstacle detection algorithm is found to be decently robust in detecting obstacles sticking out of the ground. But it does not particularly consider holes or cliffs. Given the depth map and 3D location of points in view, it is easy to build a few more features in the pipeline to seamlessly handle these kinds of obstacles as well. For example, we could classify the road based on the fact that it has a steady gradient in depth value and plan a path for the vehicle only along the definite road, if exists, to avoid falling off cliffs. And of course, besides just obstacles, the vision system should also detect lane markings, sign boards, bicyclists' hand signals, etc to complete the whole perception package of the vehicle.

ACKNOWLEDGMENT

We would like to thank Jayant Thatte, Jean-Baptiste Boin and Dr. Haricharan Lakshman for their guidance through the course of this work. We would also like to thank Prof. Gordon Wetzstein, Kushagr Gupta and the rest of the CS232 teaching staff for teaching the class.

APPENDIX

The work split up among the authors is as follows:

- **Naveen:** Data collection, disparity estimation, depth calculation, obstacle detection algorithm development and implementation, poster and report.
- **Nitin:** Data collection, disparity estimation, obstacle detection algorithm development and implementation, post-processing, poster and report.

REFERENCES

[1] Don Murray and James J Little. Using real-time stereo vision for mobile robot navigation. *Autonomous Robots*, 8(2):161–171, 2000.

[2] Hans P Moravec. Rover visual obstacle avoidance. In *IJCAI*, pages 785–790, 1981.

[3] Don Murray and Cullen Jennings. Stereo vision based mapping and navigation for mobile robots. In *Robotics and Automation, 1997. Proceedings., 1997 IEEE International Conference on*, volume 2, pages 1694–1699. IEEE, 1997.

[4] Nicola Bernini, Massimo Bertozzi, Luca Castangia, Marco Patander, and Mario Sabbatelli. Real-time obstacle detection using stereo vision for autonomous ground vehicles: A survey. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 873–878. IEEE, 2014.

[5] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[6] Hernán Badino, Uwe Franke, and Rudolf Mester. Free space computation using stochastic occupancy grids and dynamic programming. In *Workshop on Dynamical Vision, ICCV, Rio de Janeiro, Brazil*, volume 20, 2007.

[7] Florin Oniga and Sergiu Nedevschi. Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection. *Vehicular Technology, IEEE Transactions on*, 59(3):1172–1182, 2010.

[8] Andreas Wedel, Annemarie Meißner, Clemens Rabe, Uwe Franke, and Daniel Cremers. Detection and segmentation of independently moving objects from dense scene flow. In *Energy minimization methods in computer vision and pattern recognition*, pages 14–27. Springer, 2009.

[9] Uwe Franke, Clemens Rabe, Hernán Badino, and Stefan Gehrig. 6d-vision: Fusion of stereo and motion for robust environment perception. In *Pattern Recognition*, pages 216–223. Springer, 2005.

[10] Philip Lenz, Julius Ziegler, Andreas Geiger, and Martin Roser. Sparse scene flow segmentation for moving object detection in urban environments. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 926–932. IEEE, 2011.

[11] A Talukder, R Manduchi, A Rankin, and L Matthies. Fast and reliable obstacle detection and segmentation for cross-country navigation. In *Intelligent Vehicle Symposium, 2002. IEEE*, volume 2, pages 610–618. IEEE, 2002.

[12] Massimo Bertozzi, Luca Bombini, Alberto Broggi, Michele Buzzoni, Elena Cardarelli, Stefano Cattani, Pietro Cerri, Alessandro Coati, Stefano Debattisti, Andrea Falzoni, et al. Viac: An out of ordinary experiment. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 175–180. IEEE, 2011.

[13] Alberto Broggi, Michele Buzzoni, Mirko Felisa, and Paolo Zani. Stereo obstacle detection in challenging environments: the viac experience. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1599–1604. IEEE, 2011.

[14] Kyung-Hoon Bae, Dong-Sik Yi, Seung Cheol Kim, and Eun-Soo Kim. A bi-directional stereo matching algorithm based on adaptive matching window. In *Optics & Photonics 2005*, pages 590929–590929. International Society for Optics and Photonics, 2005.

[15] HanSung Kim and Kwanghoon Sohn. Hierarchical depth estimation for image synthesis in mixed reality. In *Electronic Imaging 2003*, pages 544–553. International Society for Optics and Photonics, 2003.

[16] Jochen Schmidt, Heinrich Niemann, and Sebastian Vogt. Dense disparity maps in real-time with an application to augmented reality. In *Applications of Computer Vision, 2002.(WACV 2002). Proceedings. Sixth IEEE Workshop on*, pages 225–230. IEEE, 2002.

[17] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of optical flow estimation and their principles. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 2432–2439. IEEE, 2010.

[18] Heiko Hirschmüller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 807–814. IEEE, 2005.