Vision Methodologies on Autonomous ground vehicle

Shubham Praveen Jain, Swati Gupta, Shubh Gupta

EE698G - Probabilistic Mobile Robotics

Abstract

The project deals with utilizing the techniques learned in the course to develop vision based systems for autonomous maneuvering of an intelligent ground vehicle, namely in traversing within lanes, avoiding obstacles and maintaining its heading while navigating in unfamiliar territory.

Keywords: Vision, Intelligent ground vehicle

1. Method

The primary objectives of the vision algorithm are as follows -

- Obstacle Removal
- Lane Detection
- Local and global Occupancy Map updation

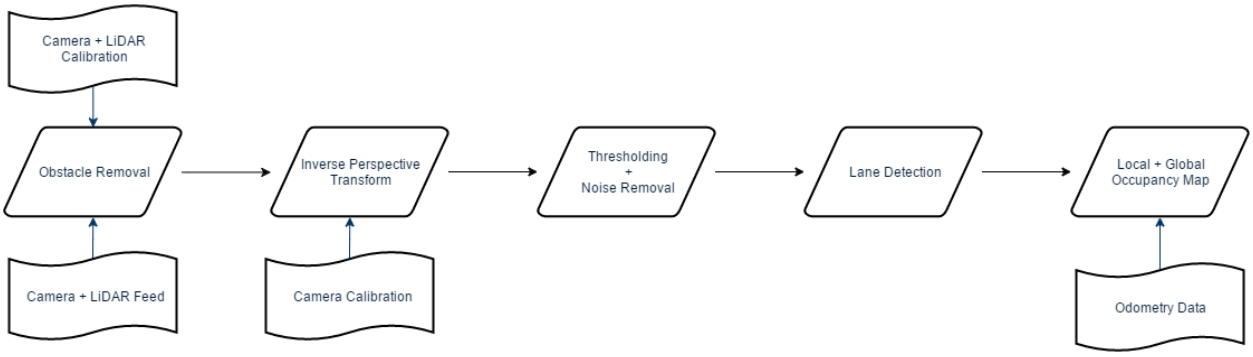

Figure 1: Algorithm of the vision stack

The vision stack is divided into following subsections:

1.1. Camera undistortion and Odometry association

To begin with, we need to procure an image of the space in front of the robot and convert it into a suitable format for further processing. We have used a wide angle camera for this purpose with a field-of-view of 120 degrees. With such a wide field of view, the image produced by the camera is distorted at the edges and hence, needs to be undistorted. Radial distortions arise as a result of the shape of lens, whereas tangential distortions arise from the assembly process of the camera as a whole. Radial correction is of the form

$$x_c = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$
$$y_c = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6)$$

Tangential correction is of form

$$x_c = x + 2p_1 xy + p_2(r^2 + 2x^2)$$
$$y_c = y + 2p_1 xy + p_2(r^2 + 2y^2)$$

Also, the camera intrinsic matrix [1] needs to be determined which is of the form

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

where x, y are camera pixel coordinates and w is the homogenizing term and X, Y, Z are real world coordinates. We use the classical black-white chessboard approach using opensource openCV codes to obtain the distortion coefficients as well as camera intrinsic matrix, and save them to a file. We then use this data while capturing the image to undistort it.
Since the further processing of the image passes it through various nodes, the image might be appreciably delayed when it is to be combined into the global map, and would hence be correlated to false odometry. To avoid this issue, we combine the image with the correct odometry at a previous stage and pass it along as a hybrid message of image data and odometry.

1.2. Lidar-Camera Calibration and Obstacle Elimination

Obstacle removal is an important step because obstacles act as a hinderance to the lane detection algorithm. We perform this step by mapping the data obtained from a 2D LiDAR on the video feed from the camera. In order to do so, we need a transformation matrix from the LiDAR reference frame to the camera reference frame. This transformation depends upon the placement of the sensors on the robot. Hence, we first fix the sensors on the robot and then calibrate them to obtain the required transformation. The rough values of rotation and translation matrices are calculated by physically measuring these quantities. These values are then fine-tuned manually by ensuring that there is a proper overlap of the LiDAR points on the video feed.
Using this transform, we find points on the image corresponding to the obstacles. Using a small patch around these points as an input for the watershed algorithm [2], the foreground is separated from the background. The foreground is assumed to contain all the obstacles. Hence, we replace the entire foreground with zero valued pixels so that they do not interfere with the lane detection algorithm.

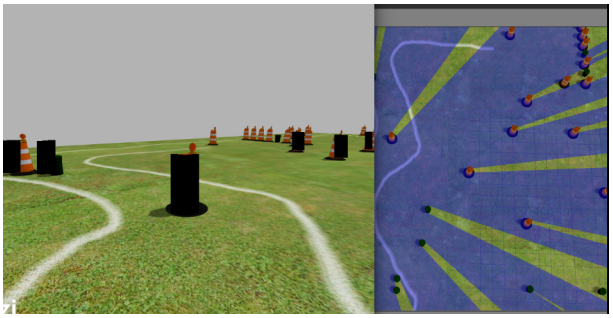

Figure 2: Left - Obstacle removed image, Right - LiDAR scan

1.3. Overhead view transformation

The obstacle-free image has a perspective error because of which the lanes appear to be converging towards each other which makes separation between the right and left lane difficult. Hence, this error is removed by an inverse perspective transform of the image to give a bird's eye view. This transformation is important as it makes path planning easier after the lanes have been detected.

The homography matrix for this transform is obtained by placing a chessboard on the ground plane and obtaining the transform by detecting chessboard corners [3] and comparing them with their true dimensions which correctly aligns corner points of the chessboard as they should appear from an overhead view.

The Overhead view transformation is a rotation about the camera center which converts the real world ground plane into a plane directly in front of the camera and parallel to the image plane. Hence, first all image pixels are mapped to the ground plane, followed by a rotation and conversion back into pixel coordinates. This process is simplified to a multiplication with a perspective matrix.

$$\begin{bmatrix} x' \\ y' \\ w \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

here, x, y, w are homogeneous pixel coordinates and X, Y, Z are mapped real world coordinates of the ground in front of camera.

Figure 3: Left - Original image, Right - Inverse Perspective Transform (Bird's eye view)

1.4. Lane detection

The lane detection algorithm makes two assumptions about the lane - 1) The lane is white in colour, 2) The degree of curvature of the lane is very less hence it can be approximated

by a straight line.
Using the first assumption, a simple thresholding of pixels based on RGB values is used to filter out background points and keep only the lane points. Erode and dilate functions are used to remove noise from this filtered image. Assuming our lanes are straight, Principal Component Analysis [4] is applied on the lane points to find the best fit line passing through them.
In order to make the above algorithm more accurate, lane points close to the previous location of the lane are used for PCA because the frame rate is considerably high as compared to the speed of the robot. Hence, we can assume that the lane will move a very small distance in successive frames.

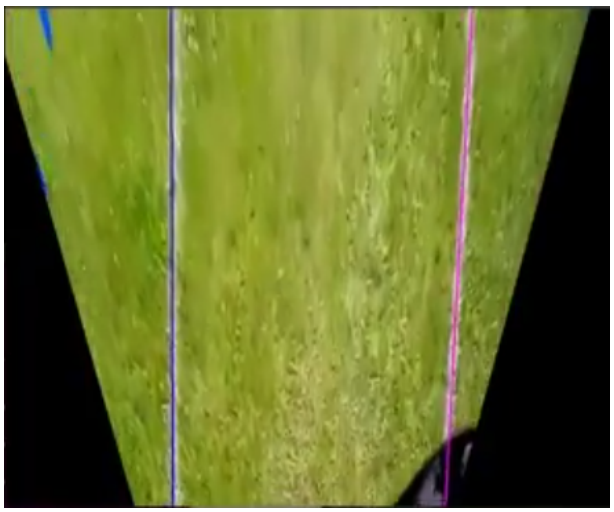

Figure 4: Left lane - Blue line, Right Lane - Pink line

1.5. Heading determination

Motion Planning algorithms require the vision stack to return an approximate heading direction for the robot, so that an efficient path can be planned. This heading can be assumed parallel to the lane direction, as the robot must roughly move along this direction at all times.
Using the output from the lane detection algorithm, we apply canny edge detection [5] , which detects all edges in the frame, in terms of rho(edge length) and theta(slope angle). Assuming that the frame only contains the lanes(as it comes from the lane detection algorithm), edge detection returns edges roughly parallel to the lanes, and an average of all the theta's obtained can thus be taken as the approximate heading. A further averaging with the values obtained in the last 20 frames(sliding window), ensures that the algorithm returns a smooth and jerk invariant output.

1.6. Global Map generation

Once the lane lines are seperated from the rest of the image by the image by the lane detection node, the obtained image must be fused with the occupancy grid containing the lane map. Here we utilize the combined and corrected odometry generated by fusing IMU

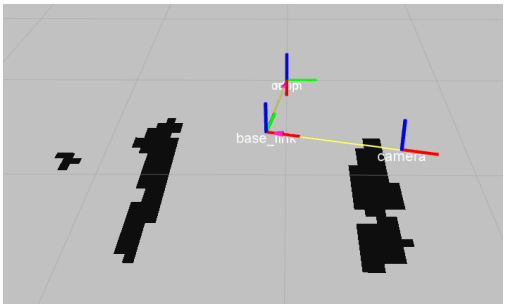

Figure 5: Occupancy Map

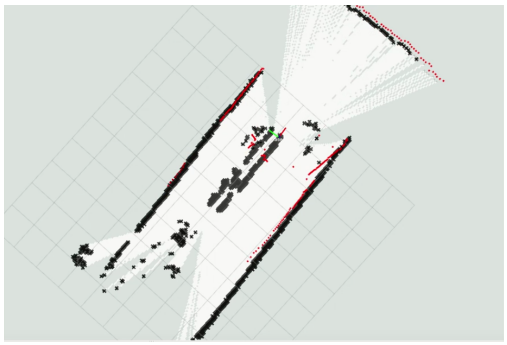

Figure 6: Global map

data and encoder data and corrected by scan matching of lidar data. This odometry had been fused with the image message during an earlier stage and passed along the nodes so as to avoid propagation delays. Utilizing this odometry as well as appropriately scaling the image to the resolution of odometry (determined using intrinsic parameters of camera), we calculate the corresponding rotation and translation matrices and use them to overlay the image over the occupancy grid [4]. Each stack of the image increments that point's probability value, which has a resolution of 0.01.

2. Experiments and Results

2.1. Gazebo simulation

All of the codes were tested on a simulated environment developed in gazebo prior to the live run and were found working as per expectations.
NOTE: The gazebo environment has been developed by first year students in Team IGVC and not by students in the course.

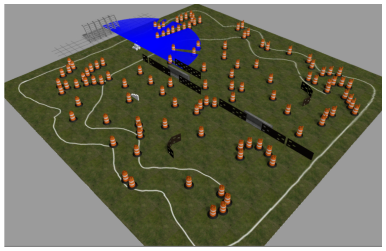

Figure 7: Gazebo simulation

2.2. Live Run

The robot has been tested multiple times on surfaces such as green grass, yellow patched grass and cemented surface and gives satisfactory results in an evenly lighted environment. In case of lighting condition changes, primitive thresholding has to be manually updated to give decent results. In case of large variations in lighting conditions during the run (like sudden shadows or glare from sun), the results are not very satisfactory and work needs to be done on improving that aspect. Factors like camera jerks due to uneven surface also lead

to unfavorable results, but error created by jerks is mitigated once the robot becomes stable again.

3. Conclusion

The technology of autonomous navigation and exploration has an immense range of applications in many fields, and a tremendous scope of further growth and improvement. This project is a small step in this undertaking, with an explicit focus on vision based sensing systems, which have an inherent advantage over other traditional sensing systems like IR, and ultrasonics in that cameras can be used to reduce the overall cost, maintaining high degree of intelligence, flexibility and robustness. We have covered the whole vision stack, starting from undistortion of live video feed, to successful extraction of important parameters like traversable area between the lane, heading, obstacle avoidance and elimination in real time as well as converting them to a form easily integrable with further nodes in the sequence (Mapping, Localisation and Motion Planning). Rigorous testing and a complete practical implementation on an actual robotic system implies that all algorithms are feasible, robust and directly applicable in the real world.

4. Future Work

Our future work will focus on extending the approach to provide fully automated camera and LiDAR to camera calibrations, as well as further generalizing the lane detection algorithms, making them light invariant and robust to a wide variety of environments. Further testing of the algorithms as well as a full integration with motion planning also needs to be performed in order to detect any possibilities of failure in certain boundary conditions, and debug them.

5. References

[1] Monocular Camera Calibration, `http://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration`, 2010. [Online; accessed 2017-04-20].

[2] S. Beucher, F. Meyer, The morphological approach to segmentation: the watershed transformation, OPTICAL ENGINEERING-NEW YORK-MARCEL DEKKER INCORPORATED- 34 (1992) 433–433.

[3] OpenCV Chessboard vision tools, `http://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html`, 2011. [Online; accessed 2017-04-27].

[4] G. Pandey, Probabilistic mobile robotics, Lecture Slides, EE698A, 2017.

[5] J. Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-8 (1986) 679–698.