Automatic Labeling of Lane Markings for Autonomous Vehicles

Jeffrey Kiske
Stanford University
450 Serra Mall, Stanford, CA 94305
jkiske@stanford.edu

1. Introduction

As autonomous vehicles become more popular, there is an increasing need to reduce the cost of the expensive equipment outfitted on each car. Traditional sensors for autonomous vehicles include an expensive lidar (~$70,000) to accurately determine depth near the car. Generally, if the equipment is damaged, the only solution is to purchase another.

In this project I will build a system to automatically label highway lane markings. These labels will be used to train a deep neural network that will be able to predict lanes using images from inexpensive cameras. Removing the lidar completely from autonomous vehicles is beyond the scope of a quarter-long project. Future projects can use my system to label cars, road signs, and other obstacles in the road as inputs to other machine learning systems.

Existing solutions to this problem such as the Mobileye Lane Departure Warning System are effective at a range of only 30 meters [3]. My goal is to predict lane markings beyond that. I will combine data collected from a Velodyne lidar (HDL-32E), HDR video streams, and a high precision GPS and build a large dataset of labeled lane markings. Eventually this data will be used to train a neural network which will predict lanes from only photos.

2. Related Work

2.1. Review of Related Work

Before implementing my project, my research group was using a traditional computer vision approach to label lanes. The system would look in the bottom 2/3rds of the image and find the two largest white or yellow areas closest to the center of the frame. While this approach correctly labeled lanes in some situations, it failed when there was poor lighting or occlusions. When testing the neural network, it was obvious that there were too many erroneous labels because the system was unable to consistently predict the lanes.

This project implements a system proven to be successful on other autonomous vehicles. Specifically, Professor Sebastian Thrun's research group fused Velodyne point clouds with camera images to find curbs, lane markings, and cars. My project is the first step to implementing many of these systems.

2.2. Main Contributions

Instead of using traditional computer vision with a single camera, this project will fuse inputs from many different sensors. I use position from a high precision GPS, depth from a Velodyne Lidar, and color information from two stereoscopic cameras. This method requires all components to be time-synchronized and calibrated. To calibrate sensors, I use a combination of automatic and manual adjustment. Fusing the data from each sensor into one map allows me to generate highly accurate lane labels for 20-hours of driving data.

3. Technical Information

3.1. Summary

The goal of my project is to fuse three independent sources of data to create accurate labels for lane markings. To do this I need to synchronize timing and calibrate each source with respect to the others. I implement manual and automatic techniques to calibrate each sensor and combine all data into a single map. From this map, I remove points that are not lane markings using thresholding. These points are easy to find because lane markings are highly reflective, and we know the approximate position relative to the Velodyne. Using hierarchical clustering, I can remove noise and ensure there is a one-to-one mapping between labels and lane markings. Using this method, I am able to quickly and accurate label lane markings better than the previous system.

3.2. Building a Dense Map

Velodyne outputs a set of 64,000 data points at 10 Hz, each with a 3D coordinate, a measure of reflective intensity, and the laser index. The Velodyne also reports localization information with accelerometers, gyroscopes, and a low-resolution GPS. Highway lane markings are more re-

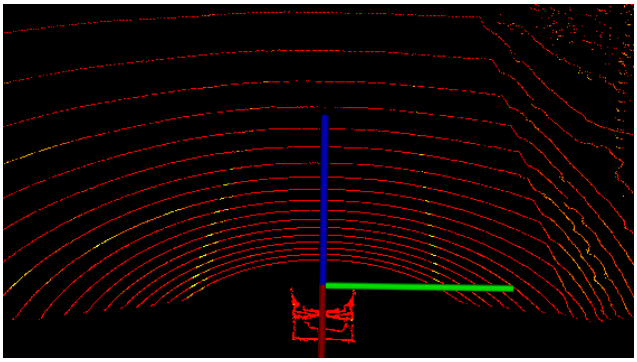

Figure 1. Lanes are reflective so they have a higher intensity value

flective than the surrounding asphalt, so the reported laser intensity value is larger. In Figure 1, green points indicate high reflectivity and red indicates low reflectivity.

Individual Velodyne point clouds are too sparse to label lane markings further than 20 meters away. With a single point cloud the Velodyne can see the closest lane marking, at best. To label lanes farther than that, we need to combine clouds from different points in time into a single map.

With accurate position and time information, we can overlay multiple point clouds into a single view. The GPS unit mounted on the car reports position with centimeter accuracy. A crucial step in combining point clouds is calibrating the GPS to the Velodyne. Figure 2 shows what can happen when the two units are not calibrated with respect to each other.

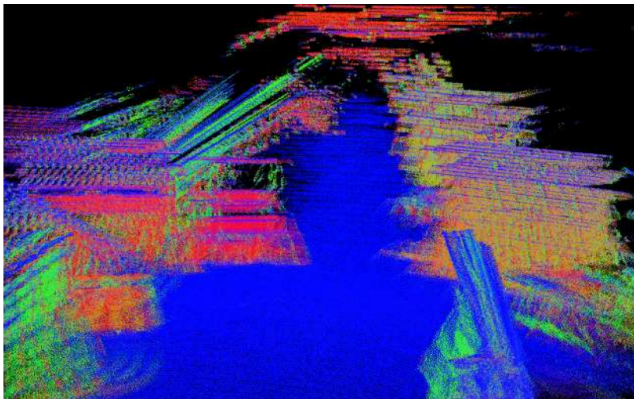

Figure 2. Integrating point clouds with an uncalibrated GPS yields blurry, unrecognizable results[1]

Although there are automated ways of finding this calibration, manual calibration works well if we consider each point cloud relative to the GPS's reference frame. We can make this assumption because both the Velodyne and GPS are rigidly mounted to the car – a change in GPS position is the same as a change in Velodyne position. The system only needs to be calibrated for errors in rotation between the Velodyne and GPS.

Given a point cloud (P), the GPS to Velodyne rotation offsets (R_{calib}), and the change in position given by the GPS($T_{\Delta position}$), we can start overlaying point clouds from different points in time (P^*).

$$R_{calib} * T_{\Delta position} * P = P^* \quad (1)$$

Extending this method to multiple point clouds, we obtain the dense map seen in Figure 3. Using the GPS to Velodyne calibration, we are able to obtain 3D maps of entire highways. This step makes it possible to reliably extract lane markings from the Velodyne's point cloud.

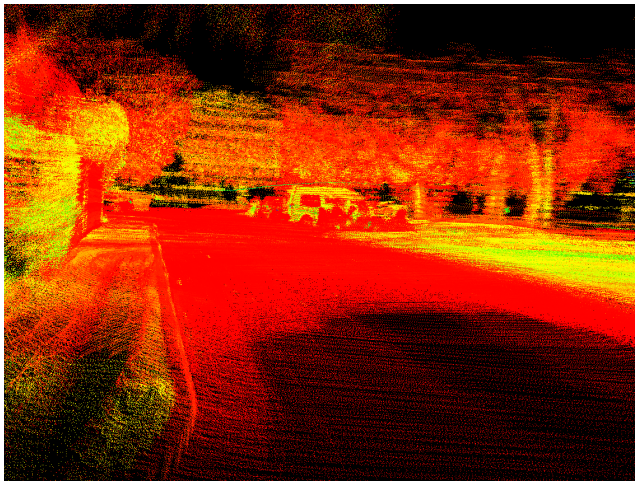

Figure 3. Using a correct calibration produces recognizable objects in the point cloud

3.3. Time Synchronization

Synchronizing the timing between all three data sources is crucial for building the map described in Section 3.2. Synchronizing this system is difficult because each individual component has its own clock. The GPS and Velodyne operate on different GPS units, and the cameras use the computer's UTC time[2]. Synchronizing the clocks is also complicated by different data refresh rates. The GPS reports position at 200 Hz, the Velodyne reports a point cloud at 10 Hz, and the camera's refresh their image at 50 Hz. The system finds the closest Velodyne point cloud and GPS position relative to the camera's timestamp. Since the Velodyne data refresh rate is 5x slower than the camera frame rate, this causes a timing error of up to 80 milliseconds. This disparity means that the point cloud may be up to 2.22 meters away from the actual markings at highway speeds (100 km/h). Although, this does not cause problems in practice because we assume the highways are generally straight.

[1]Image from *Unsupervised Calibration for Multi-beam Lasers*[1]

[2]The computer's UTC time is synchronized with Stanford's NTP server at `time.stanford.edu`

An additional complication of time synchronization is the disparity between UTC and GPS time. GPS time does not take leap seconds into account. Leap seconds occur non-deterministically up to twice each year. Since the start of GPS time (Jan 6, 1980) there have been 16 leap seconds[3]. This means that UTC time and GPS time are misaligned by 16 seconds. Since leap seconds cannot be predicted, the synchronization between the three components in the system will need to be updated when a new leap second is announced. Since this data is being post-processed, a timing misalignment error should be easily caught.

3.4. Camera Calibration and Velodyne Positioning

The system also needs a correct calibration between the cameras with the Velodyne. Determining the rotation and translations between these two components can be done either manually or automatically. Manual calibration is done by projecting the point cloud into the camera's field of view and adjusting the transformation parameters until the points 'line up' correctly with the scene. Specifically, we need to know the x, y, and z translation and the yaw, pitch, and roll rotations of the cameras relative to the Velodyne. The translation parameters are easily measured using a tape measure and the rotation parameters are initially guessed. My results for manual calibration can be seen in Figure4.

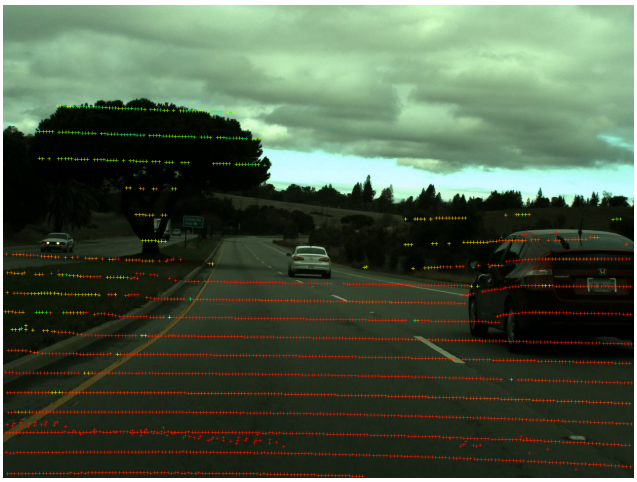

Figure 4. Projection of the Velodyne point cloud using manual calibration

Once I had an initial estimate for the affine parameters, I implemented the iterative algorithm described by Levinson[2]. This algorithm maximizes the correlation between depth discontinuities in the point cloud and color discontinuities in the camera image. Although this algorithm was designed to quickly determine if a calibration is correct, it can also be extended to refine a calibration given an initial guess.

[3]The last one was announced in June 30, 2012

Given an image I calculate the 'edginess' of each pixel. The edginess value of each pixel is determined by the maximum intensity distance between it and its eight neighbors. We will call this new edge image E. Next, apply a smoothing function such that the i, jth pixel's value is defined as:

$$D_{i,j} = \alpha * E_{i,j} + (1-\alpha) * \max_{x,y} E_{x,y} * \gamma^{max(\|x-i\|,\|y-j\|)} \quad (2)$$

This smooths the edges of the image so that the objective function (defined in Equation 4) is rewarded for matching points close to edges[4]. Levinson recommends values $\alpha = \frac{1}{3}$ and $\gamma = 0.98$ for smoothing.

With our new 'edginess' image D, we can implement the second part of the algorithm: finding depth discontinuities in the Velodyne point cloud. Define a new point cloud X where each point i in X is defined as the maximum distance between two horizontally adjacent laser points P. Since the Velodyne is parallel to the ground, we can assume these points are from the same laser at different times in the sweep.

$$X_i = \max(\|P_{i-1}\| - \|P_i\|, \|P_{p+1}\| - \|P_p\|, 0)^{0.5} \quad (3)$$

We remove points from P that have distance values in X less than 30 centimeters to create P^*. This removing points with low distances increases the chance we see an edge at that point in the camera image. We construct objective function J given our initial guess C_g from manual calibration. J is maximized by aligning large depth discontinuities in P^* with high intensity values in D. $D_{i,j}$ refers to the pixel coordinate that point P_p^* projects to.

$$J = \sum_p C_g * P_p^* * D_{i,j} \quad (4)$$

We can attempt to find an absolute maximum for J by performing a grid search across each of the six degrees of freedom in C_g. Iteratively searching this grid will eventually find the calibration C that maximizes J. This method only works if our initial guess is close to the correct calibration – a valid assumption given the manual calibration qualitatively looks correct.

Using the Velodyne-camera calibration obtained from Levinson's method and the GPS-Velodyne calibrations through manual calibration, I am able to combine all sensor inputs. I can now add color information the 3D depth maps shown in Figure3. With all of the sensors correctly calibrated, I can build 3D color maps of a scene. While this is not useful for labeling the lanes, it is easy to verify all three calibrations are correct (see Figure 5.

[4]The inverse distance transform is used in my implementation, but any smoothing function should work

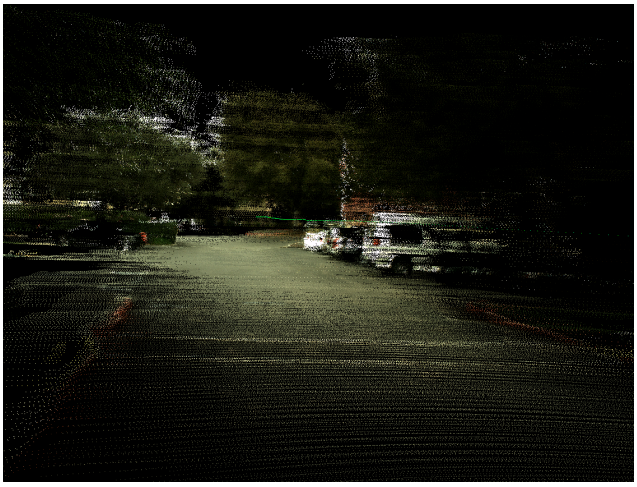

Figure 5. A colored map of the scene. This fuses all three sensor inputs

3.5. Filtering Out the Lane Markings

While I originally believed that I needed to locate the ground plane to find lane markings, they can be isolated from the rest of the point cloud by thresholding. We know that in all of our test data the lanes are 1.5 meters below the Velodyne and approximately 1.5 meters on either side. Lane markings are also highly reflective, so points with intensity value less than than 60 can be discarded[5]. With this filter, I was able to see all reflective markings within about 50 meters of the car. Figure 6 shows my results for a single point cloud.

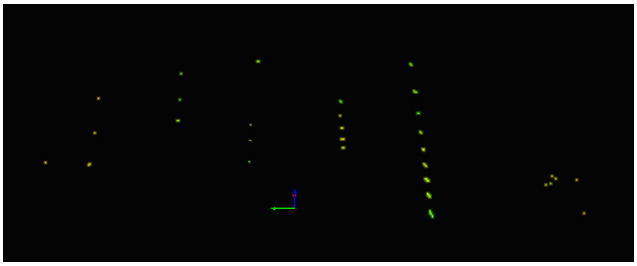

Figure 6. Lanes can be easily filtered from the surrounding asphalt

Using the integrated maps created in section 3.2 and the filters described above, I projected 5 seconds of driving onto one frame of video. Applying the filters to all frames in the map without applying another transformation would cause errors. Specifically, it could remove lane markings from the map if the road curved or changed elevation. To correctly apply the filters to a point cloud that occurs at t_1, I first need to transform the cloud to the position at time t_0 (where t_0 is the point cloud closest to the camera timestamp). Next, I apply the filter and transform back into t_1. Applying the filter after a transformation to t_0 ensures that all clouds are filtered correctly. Figure 7 shows my results filtering out all lanes from the highway map. Alignment errors are within the 2.2 meter range explained in Section 3.3.

[5]For reference, the asphalt has reflectivity between 0 and 5

Figure 7. Lanes are filtered from the point cloud map of a highway

3.6. Clustering Lane Markings

The results in Figure 7 are promising, but also noisy. To improve my labels, I used hierarchical clustering to group lanes together. Hierarchical clustering calculates the distance from one point to all other points. It groups points that are within a certain distance threshold of each other. I weight the distance function to reward points that lie the axis parallel to the direction of travel. This ensures that points from different lanes are not grouped together.

After all points are grouped, I take the median of each group. This places a single point on each lane marking. I plot the projection of the points using the same calibration matrix obtained in Section 3.4. The results of clustering for a single lane is shown in Figure 8.

Figure 8. Points are clustered to give one point per lane marking

4. Experimental Results

Using the Velodyne to label lanes works significantly better than the traditional computer vision approach. Looking for white and yellow clusters on the ground failed with lanes that were far away or had poor lighting. A significant advantage of my implementation is that it can label lanes far into the future with high accuracy. It can also label lanes that are occluded by cars or changes in elevation. This is very important when training the lane predictor because it should be able to predict the lane path even when the markers are hidden.

Unfortunately, I cannot compare my results to a ground truth. Building the a dataset to compare to would require hand labeling the 20 hour corpus. Even with hand labeling, a human label may be very different than my system's label because any label on the lane marker is a correct. Qualitatively, the output of my system performs better than the previous system.

Some example outputs of the lane labeler can be seen at `http://goo.gl/SfF5M1`

5. Conclusion

This project combined three data sources to label lanes in a 20 hour data set. Much of the project involved synchronizing data across independent clocks and calibrating each data source with respect to the others. By combining multiple data sources, I was able to obtain results that far exceeded the accuracy of the previous color-based approach. Other projects can also build off by my system. Given the calibrations I determined, cars and road signs can be detected by simply changing the thresholds and clustering parameters.

References

[1] J. Levinson and S. Thrun. Unsupervised calibration for multi-beam lasers. In *International Symposium on Experimental Robotics*, 2010.

[2] J. Levinson and S. Thrun. Automatic online calibration of cameras and lasers. In *Robotics: Science and Systems*, 2013.

[3] Mobileeye. Mobileye lane departure warning (ldw), Jan. 2014.