LANEMAPPER: A CITY-SCALE LANE MAP GENERATOR FOR AUTONOMOUS DRIVING

An Undergraduate Research Scholars Thesis

by

ANKIT RAMCHANDANI

Submitted to the Undergraduate Research Scholars program at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by Research Advisor: Dr. Dezhen Song

May 2019

Major: Computer Science

TABLE OF CONTENTS

ABSTRACT

LaneMapper: A City-scale Lane Map Generator for Autonomous Driving

Ankit Ramchandani
Department of Computer Science and Engineering
Texas A&M University

Research Advisor: Dr. Dezhen Song
Department of Computer Science and Engineering
Texas A&M University

Autonomous vehicles require lane maps to help navigate from a start to a goal position in a safe, comfortable and quick manner. A lane map represents a set of features inherent to the road, such as lanes, stop signs, traffic lights, and intersections. We present a novel approach to detect multiple lane boundaries and traffic signs to create a 3D city-scale map of the driving environment. We detect, recognize and track lane boundaries with multimodal sensory and prior inputs, such as camera, LiDAR, and GPS/IMU, to assist autonomous driving. We detect and classify traffic signs from the image considering high reflectivity of LiDAR points and further register the locations of traffic signs and lane boundaries together in the world coordinate frame. We have also made our code base open-source for the research community to tweak or use our algorithm for their purposes.

DEDICATION

To my father and my mother

ACKNOWLEDGMENTS

I would like to wholeheartedly thank my faculty advisor Dr. Dezhen Song for his support and time to review my research and thesis. I would also like to thank my PhD mentor, Binbin Li, for being very helpful during the entire research process, letting me experiment with my ideas and helping me review my thesis. This work would not have been possible by their support.

I would also like to thank my family and friends for their unconditional support over my entire research journey.

LIST OF FIGURES

FIGURE Page

1. INTRODUCTION

Autonomous Driving has lately gained a lot of public and media attention because of its incredible potential to transform daily lives. In this thesis, we propose a novel approach to generate city-scale lane maps for autonomous driving. A lane map, at least in the context of this paper, is defined as a three dimensional map of the environment of the vehicle containing the precise location of some important features of the road like lane marks and traffic signs. We utilize multi-modal sensory inputs to extract information related to lane boundaries and traffic signs. We also present a unified system to detect lane marks, detect traffic signs and make a lane map of the surroundings of the car.

Developing a full autonomous driving system is an incredibly humongous task, both in terms of scale and complexity. A large part of the complexity is attributed to the real-time nature of the problem. To save or reduce on-board computation, it is essential to pre-compute as much information as possible. This is where the importance of a lane map comes in. If the vehicle is travelling on a road that has been mapped before, the map already contains all the necessary information about where points of interest like traffic signs and lane marks are located on the road. The vehicle does not need to spend its essential on-board computation time to locate points of interest, but at the most only needs to verify their existence in the vector map. Lane mark detection alone is also very widely used for lane following, lane tracking and lane departure warning. In addition to autonomous driving systems, these applications also have a high utility value in intelligent driving assistant systems which are available in even the contemporary commercial vehicles. Traffic sign detection is also used in driver assistant systems to ensure that drivers don't miss any important traffic signs. If speed limit signs are seen by this system, the system can also warn the driver if he/she crosses the limit.

Lane mark detection is a fairly challenging task because of the following reasons. The fact that lane marks exist in many different shapes and types (dashed rectangles, arrows, solid rectangles etc) (Figure 1.1a) basically makes all shape based approaches either too complex or ineffective. Lane marks are also susceptible to sharp, unpredictable local intensity changes due to shadows or excessive sunlight (Figure 1.1c). These changes yield strong gradients and can be challenging for gradient based approaches. Though more general than just the the lane mark detection problem, the great variation in ambient light (night time driving, cloudy weather, etc) is, in general, both a significant and difficult problem for camera based approaches to deal with (Figure 1.1b).

Traffic sign detection also faces similar challenges. The different shapes (circles, octagons, triangles, etc) and colors (yellow, red, blue, etc) (Figure 1.1d) make shape/color based methods too complex or ineffective as it is hard to find a property common to all traffic signs (one of which is high reflectivity, as used in this research). The variation in ambient lighting is again an issue that needs to be considered. Due to these issues, it is much harder to make hand crafted features for the traffic sign detection problem than it is for lane mark detection.

(a) different types of lane marks (b) variable ambient lighting (c) local brightness changes by shadows (d) different traffic signs

Figure 1.1: Challenging scenarios in detection

In our work, we present novel methods to detect lane marks and traffic signs using both the monocular vision (camera) and light ranging and detection (LiDAR) modalities. In the

lane detection component of our algorithm, we, in addition to detecting, also model the direction of lane marks by spline curves. As would be detailed later, we use a combination of deep learning models, machine learning algorithms and image processing and sensor fusion techniques to get robust lane mark prediction results.

For traffic sign detection, we use clustering algorithms and image processing techniques to detect possible regions where traffic signs may be present. This proposal generation phase only uses LiDAR reflectivity to generate proposals and has very high recall (the ability to detect a traffic sign in the image, if there is one) of traffic signs. To the best of our knowledge, we are unaware of other methods that rely only on LiDAR reflectivity to detect traffic signs. Traffic signs are mandated by law to be made of highly retroreflective materials. Retroreflective materials are materials that are capable of reflecting light back to their source with minimum scattering, irrespective of the angle of incidence of the incoming light ray. This is why reflectivity is a very good measure of detecting traffic signs. Also, our dependence on LiDAR only to detect traffic signs signifies that our algorithm would even work when the ambient light is very low (night time) because LiDAR has its own source of light. We use deep learning methods to classify traffic signs after detection.

Our lane mark detection algorithm has four steps. First, given an image, we run ICNet [1], an image segmentation neural network, on it to know where the road is in the image. Second, we process the segmented image using image processing techniques to generate some predictions of lane marks. The image processing techniques rely on the fact that lane marks are brighter than their immediate surroundings and have roughly constrained shape. In the third step, we consider our predicted lane marks obtained in the last step and only keep regions that have high LiDAR reflectivity, like lane marks are supposed to. Lastly and perhaps most importantly, we consider all LiDAR points and fit one spline curve on the predicted regions on the left of the car and a parallel curve to all regions in the right of the car. We use T-Linkage [2], a robust multiple structure estimation method, to fit the spline

curves and remove all regions too far from these spline curves. The intuition behind fitting a curve on the left and right side of the car is to model the direction of the immediate left and immediate right lane marks of the car. All predicted regions that are sufficiently close to the spline curves are considered to be lane marks. More details regarding the algorithm would be discussed later.

For traffic sign detection, we first project all LiDAR points on top of the image such that each projected point's intensity is equal to its reflectivity and the rest of the image is black. We then use some basic image processing techniques to make this image denser and remove points with very low reflectivity, as we know that traffic signs are supposed to maintain high retroreflectivity by law. We then apply a clustering algorithm that clusters points based on their proximity and similarity in brightness. We merge nearby clusters and remove clusters that are too small. Then we draw a bounding box over each cluster and feed all such cropped regions of bounding boxes to a CNN that predicts all traffic signs in a bounding box. This approach is similar to RCNN [3], but the proposal method is much more specific for this task and the CNN does multi-label classification. Again, more details are discussed later.

Through this research, we believe that we have contributed to the community in the following ways:

1. We develop a novel and robust lane detection algorithm that uses both LiDAR and camera data to make reliable predictions, while most lane mark detection algorithms only rely on camera.

2. We develop a novel traffic sign detection algorithm based only on LiDAR reflectivity data, which we believe is a very strong prior. For all images tested, we have observed a 100% recall of this detection method. To the best of our knowledge, we don't know of other traffic sign detection algorithms that also use LiDAR reflectivity.

3. Our code is open source and would be among very few open source code bases that can create lane maps fully automatically. The code can be accessed at https://github.com/ankit61/LaneMapper.

2. RELATED WORK

Our research is related to lane marking detection and tracking, traffic sign detection and robotic mapping.

2.1 Lane Mark Detection

Lane mark detection has captured the interest of researchers since a long time due to its many applications in lane following, lane departure warning and lane tracking.

Several approaches have been proposed which use different modalities. The most frequently used modality obviously is monocular vision (camera) due to its easy availability and low cost. As will be discussed, camera methods enable the use of many methods ranging from classical image processing techniques to complex neural network based approaches. [4]

Light Ranging and Detection (LiDAR) is another modality commonly used. The greatest benefit of using a LiDAR is that it has its own source of light, making it immune to some signature problems of image based methods like varying ambient light and presence of shadows. The offset to this advantage is the high cost of the sensor, making it fairly unusable if the application area involves use of driver assistant systems in older cars. Since most LiDAR sensors include reflectivity information also, they can be used alone or in combination with a monocular vision based method. [4]

Older methods relied heavily on classical image processing techniques, many domain specific priors and hand crafted features. One example is the very popular work of Bertozzi M. and Broggi A [5]. They used the inverse perspective mapping to remove the perspective effect from the camera image, effectively transforming the image so it appears to be taken from a bird's eye view. Such techniques were usually employed to make the width of lane marks roughly equal at all distances [4]. They then identify low-high-low intensity regions

in the image and apply the geodesic morphological dilation to identify lane marks.

After lane marks are detected, it is important to mathematically model the lane marks so the direction of lane marks can be precisely determined at any point. This is referred to as model fitting. Previous methods [6, 7] have used RANSAC [8], Hough Transform [9] and its generalized versions to describe lane marks.

An important component of many lane mark detection algorithms is also using knowledge of lane mark predictions of the recent past. This helps reduce the computational cost, increase the accuracy and correct incorrect detections [4]. A popular choice to track lane marks is by using Kalman filtering [10, 11]. Some methods [12] also transform the lane mark predictions to the real world. One way to do this is by using visual or LiDAR SLAM algorithms [13, 14]. Other methods [15] use GPS/IMU data to transform to real world coordinates.

Recently, there has also been an interest in detecting lane marks in poor weather conditions [16]. This problem is much harder as the algorithm needs to adapt to to low visibility (fog), high reflectivity on roads (rain, snow) and other challenges.

We propose a lane mark detection algorithm that considers data from both monocular vision and LiDAR modalities resulting in greater robustness, while most existing methods only use camera data.

2.2 Image Segmentation

Image segmentation is a problem that has evoked a great interest in the computer vision community. It has very useful applications in a variety of areas including scene parsing for autonomous driving and robotics, human body part parsing [17], medical imaging and diagnosis. The problem involves finding a label for every pixel in the image, indicating which class does that pixel belong to. Image segmentation is an integral part of our algorithm for lane detection, as would be made more clear in the next chapter. That is why

it is important to review the work related to image segmentation: the performance of the image segmentation method would significantly influence the overall performance.

We studied a variety of image segmentation networks and judged them on their inference speed, accuracy and ease of implementation/availibility of open source implementation in Caffe [18]. We briefly surveyed the following architectures and their implementations: DilatedNet [19], ENet [20], RefineNet [21], PixelNet [22], Global Convolution Network (GCN) [23], PSPNet [24], LinkNet [25], ICNet [1], ERFNet [26], Segaware [27], Tiramisu [28], DeepLab V3+ [29], MultiNet [30]. Of these networks, MultiNet and ICNet caught our attention because they were developed for real time apps. However, we chose ICNet because it was based on the Pyramid Scene Parsing Network (PSPNet) [24], which was one of the strongest image segmentation networks at the time of our survey. ICNet also had an official implementation in Caffe, our preferred framework due its high speed.

Now we briefly shed some light on why ICNet achieves such high speed without compromising accuracy too much. ICNet takes in cascade image inputs, referring to low, medium and high resolution versions of the original image. The architecture is composed of three parts: each is its own image segmentation network run on a different resolution of the input image. Specifically, they run PSPNet on the low resolution image and relatively cheaper (computationally speaking) convolutional networks on the medium and high resolution images. The branch processing the low resolution image has a richer and more complex segmentation network as the computational cost is limited by the small input size. The intuition is to do a high accuracy segmentation on the low resolution image, where computational cost can be controlled and then recover the details lost in the low resolution segmentation map by the segmentation results of the medium and high resolution segmentation maps. The segmentation features of the three branches are combined by a novel cascade feature fusion network that the authors propose. We observed that this fairly

intuitive and low-cost architecture gives good speed and a decent accuracy.

2.3 Object Detection

Object detection is a challenging computer vision problem, which involves finding bounding boxes around all objects of interest in the image. Note that object detection is a much harder and more general problem than localization, which usually refers to finding a bounding box for a fixed number of objects in the image. Unlike localization, object detection problems can have an arbitrary many number of objects in the image, adding to its difficulty. As would be seen in the next section, our method is a specific type of the RCNN object detection method and therefore it is integral to review the work related to object detection. Using a more robust object detection algorithm has the potential to increase both the accuracy and performance of our overall algorithm.

After the rise of deep learning in 2012, one of the earliest object detection network that gained enormous popularity was the Region Based Convolutional Neural Network (RCNN) [3], which is perhaps the most closely related to the object detection network we use in this work. The basic idea was to run some method that could propose a large number of rough bounding boxes (the original paper used selective search). Then, an image classification and localization network was run over all such predictions, which would output the class of the bounding box and the correct bounding box coordinates (as the initial bounding boxes were rough estimates). Though the accuracy of this network was fine, the implementation was computationally very intensive (arising from repeatedly running a convolutional network many times) and highly complex.

This work was followed by SPPNet [31], Fast-RCNN [32] and Faster-RCNN [33]. These following works improved the older RCNN model by changing many aspects including the method for proposing regions, sharing base convolutions to save computation and making the entire model more elegant, easy to implement and end-to-end trainable.

These works showed much better performance with greater accuracy.

From an eagle's view point, the above methods had two separate steps. The first was to propose regions and the second was to classify them. A completely different way to approach the problem was proposed in YOLO [34], standing for You Only Look Once. As the name suggests, this network applies the image through the network only once and outputs the bounding boxes of objects of interest. As would be predictable, this substantially increases the speed of object detection tasks, making real-time object detection problems seem surmountable.

2.4 Traffic Sign Detection

Traffic sign detection refers to the problem of detecting, not just classifying, traffic signs in a given image. There have been highly successful methods for traffic sign classification achieving accuracies as high as 99.46% [35], but detection is a much harder problem as it involves predicting the location of traffic signs, in addition to their class.

Escalera et. al. [36] propose a method to detect traffic signs that depends on a lot of hand crafted features. They take different ratios of RGB channels with each other and threshold these ratios to detect certain colors. They claim that this method is more robust than thresholding in the RGB colorspace and computationally cheaper than converting the image to the HSV space. They then use hand crafted kernels to do corner detection and use different algorithms to detect different shapes of traffic signs. The classification is done by neural networks.

Like the case of all computer vision problems, more recent methods to detect traffic signs rely on deep learning methods [37, 38] to give more accurate results and ease the part of hand crafted feature generation. Zhu et. al. [39] propose a deep learning based approach to detect traffic signs in the wild. They show great performance by only using a vanilla convolutional network which branches off at the end to give different outputs

of interest. Lee et. al. [38] trains a CNN to classify traffic signs and predict its exact boundary. The boundary is estimated by projecting a template traffic sign on top of the input image plane. [40] trains a CNN to detect and classify traffic signs, English characters and Chinese characters using a variant of RCNN.

We propose a method to detect and classify traffic signs in a novel way: our detection algorithm is based only on LiDAR reflectivity, is simple to implement and gives very strong results. We have observed 100% recall on all images we have tested. In this case, recall determines if all traffic signs in a given sequence of images were able to be detected. Our deep learning based multi-label classifier also gives very high accuracy.

2.5 Map Generation

Map generation has become an important part of autonomous driving. To save on board computation, autonomous vehicles usually try to localize themselves on a pre-built map, which contains the location of lane marks, traffic signs and other objects of interest in the segment of the road on which the car is driving. This saves the vehicle the cost of perceiving all objects of interest repeatedly on the fly [41].

For automatic map generation using LiDAR and camera, Zhang et al. [14] propose a state of the art method to combine visual and lidar odometry. Qin et al. [13] propose a monocular vision and IMU based method to estimate states, which is capable of bootstrapping from unknown initial states and can even recover the metric scale, making it very useful in practical scenarios. Bender et al. [42] propose a method to generate a topologically and geometrically complete map of the drivable environment, composed of atomic and interconnected drivable road segments. Dolgov et al. [43] propose a lidar based method to map semi-structured environments like parking lots by building a grid based map of static obstacles and use it to estimate the drivable lanes. Chio et al. [44] propose a lidar based method to detect obstacles and create a local obstacle map in rural

and off-road conditions.

3. APPROACH

3.1 Problem Definition

We assume the vehicle is equipped with a frontal view camera, a LiDAR and GNSS inertial navigation systems, which is the common sensory configuration for autonomous vehicles. We have the following assumptions,

a.1 The camera is pre-calibrated, and the nonlinear distortion of images has been removed.

a.2 All sensor readings are synchronized.

a.3 The coordinate system transformations between any two sensors are known by prior calibration.

In this section, we introduce some notation that would be used in the rest of the thesis:

- $\{\mathcal{L}\}$ defines the lidar coordinate system with x-axis pointing in the vehicle forward direction, y-axis pointing to the left, and z-axis pointing upward. Let $\mathbf{P}_{i,t} = [x_{i,t}, y_{i,t}, z_{i,t}]^T$ be the i^{th} LiDAR point, and $r_{i,t}$ be the the reflectivity of the i^{th} point. $\mathbf{P}_t := \{\mathbf{P}_{i,t}\}$ is the set of LIDAR points at time t.
- $\{\mathcal{I}\}$ defines image coordinate system. Let $\mathbf{I}_t$ be the camera image at time t. Given image pixel $[u, v]^T$, let $h_{\mathbf{I}_t}(u, v)$ be the intensity of that pixel in $\mathbf{I}_t$.
- $\{\mathcal{W}\}$ defines the world coordinate system which overlaps with $\{\mathcal{L}\}$ at the vehicle starting position. That is, it is the same as the coordinate system of the points in $\mathbf{P}_0$.

Denote the left and right lane boundaries in $\{\mathcal{W}\}$ by ${}^{\mathcal{W}}\mathbf{S}_l$ and ${}^{\mathcal{W}}\mathbf{S}_r$ at time t, respectively.

Problem 1. Given current GPS locations, *in situ* camera and LiDAR input, detect and recognize the lane boundaries ${}^{\mathcal{W}}\mathbf{S}_l$ and ${}^{\mathcal{W}}\mathbf{S}_r$, extract traffic signs from both sensing modalities and register in $\{\mathcal{W}\}$.

3.2 Lane Mark Detection

In this section, we are interested to find the immediate left and right lane marks of the egocenteric vehicle. We first identify the road surface on the image using appearance classification, detect lane marks on the road surface and remove lane marks that have low LiDAR reflectivity. We finally fit a spline model using T-Linkage, given the 3D lane marks. The four steps are now described in detail below.

3.2.1 Road Surface Detection

Given an input image, we first find the road surface on the image. This is done by an image segmentation neural network, ICNet [1], pretrained on the CityScapes [45] dataset. Before feeding the image to ICNet, we first convert the image to the YUV color space, apply histogram equalization to the Y channel (corresponding to luminosity) and convert it back to the RGB color space. Due to making the brightness even and thus making shadows less patchy, histogram equalization makes the results of ICNet much better. A significant difference is shown in Figure 3.1.

Let us define $\mathbf{I}_t^r$ to be the image which is the same as $\mathbf{I}_t$, but has intensity 0 for every pixel that is not predicted as a road surface pixel.

3.2.2 Lane mark detection from image data

We first apply morphological dilation on $\mathbf{I}_t^r$. Let $\mathbf{D}_t$ be the image we get after applying dilation with a square n x n structuring element on $\mathbf{I}_t^r$ (During experiments, we set $n = 4$).

$$h_{\mathbf{D}_t}(u,v) = \max_{\forall_{r\in[u-n/2,u+n/2],\ c\in[v-n/2,v+n/2]}} h_{\mathbf{I}_t^r}(r,c) \tag{3.1}$$

(a) Without histogram equalization

(b) With histogram equalization

Figure 3.1: The effect of histogram equalization on ICNet results

The assumption in our algorithm is that lane marks are brighter than their immediate surroundings. More formally, we can say that the difference between the intensity of a point on the lane mark, say l, and the point just outside it, say b, is at least ω (experimentally, set to 20 on a scale of 255). Equivalently, $l - b \geq \omega$.

Now, let

$$h_{\mathbf{S}_t}(u,v) = \begin{cases} h_{\mathbf{D}_t}(u,v) - h_{\mathbf{I}_t}(u,v) & \text{if } h_{\mathbf{D}_t}(u,v) - h_{\mathbf{I}_t}(u,v) > \omega \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

Given that lane marks are brighter than their surroundings, note that the boundary of lane marks would increase by $n/2$ in $\mathbf{D}_t$. Therefore, $h_{\mathbf{S}_t}(u,v) > 0$ if $[u,v]^T$ is a point just outside a lane mark. More generally, $h_{\mathbf{S}_t}(u,v)$ is non-zero when $[u,v]^T$ is at a border of a

object sufficiently brighter than its surroundings. A typical $\mathbf{I}_t$ and its corresponding $\mathbf{S}_t$ are shown in Figure 3.2a and Figure 3.2b respectively.

We now find all closed contours in $\mathbf{S}_t$ by using breadth first search. We reject contours which have very small (10 pixels) or very large area (10,000 pixels). Area is measured by the number of pixels inside a contour. Note these bounds are very loose and so no lane mark, regardless of its shape, would be rejected by these bounds. The bounds only help remove noise. Let the image with these bounded contours be called $\mathbf{C}_t$ (Figure 3.2c).

Then, we find low-high-low intensity regions horizontally in the *original image*. Let $\mathbf{H}_t$ and $\mathbf{L}_t$ be the resultant images after applying the filter $[1\ -1\ 0]$ and $[-1\ 1\ 0]$ respectively. Note that $\mathbf{H}_t$ has high intensity at all spots where the intensity in $\mathbf{I}_t$ shifts from high to low and $\mathbf{L}_t$ has high intensity where the intensity changes from low to high. We then do horizontal peak finding in both $\mathbf{H}_t$ and $\mathbf{L}_t$. Let $\mathbf{I}$ be any image. Then, a pixel $[u, v]^T$ is defined as a horizontal peak of $\mathbf{I}$ if and only if:

$$h_{\mathbf{I}}(u, v) = \max_{\forall_{c \in [v-m/2, v+m/2]}} h_{\mathbf{I}}(u, c) \tag{3.3}$$

where m is a small constant (set to 15). We only retain horizontal peaks in $\mathbf{H}_t$ and $\mathbf{L}_t$:

$$h_{\mathbf{H}_t}(u, v) = \begin{cases} h_{\mathbf{H}_t}(u, v) & \text{if } [u, v]^T \text{ is horizontal peak} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

$$h_{\mathbf{L}_t}(u, v) = \begin{cases} h_{\mathbf{L}_t}(u, v) & \text{if } [u, v]^T \text{ is horizontal peak} \\ 0 & \text{otherwise} \end{cases} \tag{3.5}$$

The intuitive reason for doing peak finding is to get regions in the input image, $\mathbf{I}_t$, that show the sharpest change of intensity. If we keep m small enough, then we can detect the horizontal boundaries of lane marks. Note that peak finding is immune to changes in

ambient lighting in the image. It also doesn't depend on the width of lane marks.

We then get all contours in $\mathbf{C}_t$ that are between the peaks of $\mathbf{L}_t$ and $\mathbf{H}_t$. Such contours are shown in Figure 3.2d. Intuitively, we are finding objects that are brighter than their surroundings and show the sharpest change in intensity. We constrain the width and length of these newly found contours to remove noise. The constraint is so loose that no lane marks of any shape would get affected. We retain contours of width between 5 and 250 pixels and length greater than 10 pixels. There is no maximum limit on length as some lane marks span the entire road. The result after this step is shown in Figure 3.2e. Finally, we just fill in the holes in the final contours and return them as the final lane mark predictions(Figure 3.2f). Figure 3.2g shows predictions overlaid on the input image.

(a) Original input image

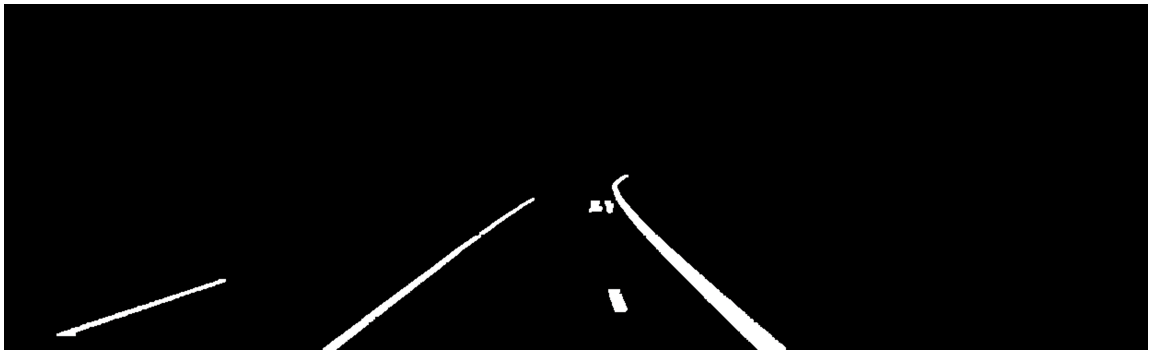
(b) Borders of bright objects in white

Figure 3.2: The Lane Detection Algorithm

(c) Closed contours with bounded area

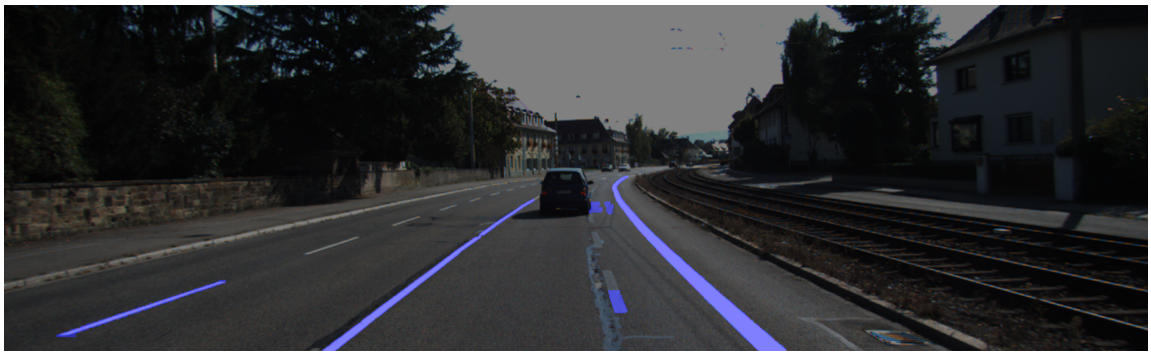
(d) Contours between $\mathbf{L}_t$ & $\mathbf{H}_t$ peaks

(e) Contours with bounded width and height



(f) Final Predicted Lane Marks

Figure 3.2: Continued



(g) Lane marks overlaid on input image

Figure 3.2: Continued

3.2.3 Filter lane mark predictions using LiDAR data

In this step, we first project all LiDAR points, $\mathbf{P}_t$, onto the image plane. We then retain only the LiDAR points that get projected to the road surface found by ICNet. Let the set of such points be $\mathbf{P}'_t = \{\mathbf{P}'_{i,t} = [x'_{i,t}, y'_{i,t}, z'_{i,t}]\}$. Let the reflectivity of $\mathbf{P}'_{i,t}$ be $r'_{i,t}$. Then:

$$\mathbb{P}^*_t = \{\mathbf{P}'_{i,t} \mid r'_{i,t} \geq t_s,\ |y'_{i,t}| < d_w \text{ and } |x'_{i,t}| < d_l\}, \tag{3.6}$$

Here, t_s is chosen by Otsu thresholding on $\forall_i\ r'_{i,t}$ and d_w is half of the maximum expected width of a lane (set to 3m) and d_l is the maximum length of lane marks detected in one frame (set to 20m). Note that all points in $\mathbb{P}^*_t$ have high reflectivity. We are interested in points with high reflectivity because lane marks are generally painted with high reflectivity paints.

Our final predictions of lane marks is a set of LiDAR points in $\mathbb{P}^*_t$ that get projected to regions predicted as lane marks by the algorithm described in section 3.2.2. In other words, a point is considered a lane mark if it was predicted as one by both the image processing algorithm (of section 3.2.2) and the LiDAR processing algorithm defined in this section.

3.2.4 Fit models to describe lane marks

Given $\mathbb{P}_t^*$ from the previous step, we apply T-Linkage [2] to fit spline curves on the points of $\mathbb{P}_t^*$. T-Linkage is capable of automatically identifying and ignoring isolated points and so is robust to some noise in predictions.

It is important to mention that we slightly specialize T-Linkage for our purposes. The algorithm requires a method to sample the given data points. Instead of using random sampling, we sample in such a way that points that are more or less vertical (facing exactly in the forward direction of the LiDAR) are more likely to be sampled together. This method is much more effective than using uniform random sampling as it can ignore more noise in the predictions (in case $\mathbb{P}_t^*$ is noisy). Our inclination towards the forward direction is obviously because lane marks are forward facing.

T-Linkage returns a set of clusters, $\{\mathbf{C}_i\}$, so that points in each cluster can be described by a spline curve. Let $\forall_k\ \mathbf{P}_{k,t}^i = [x_{k,t}^i, y_{k,t}^i, z_{k,t}^i]$ be the set of LiDAR points in $\mathbf{C}_i$. So, $\forall_k\ \mathbf{P}_{k,t}^i \in \mathbf{C}_i$. Let $\mathbf{C}_{max}$ be the largest cluster. So, $\mathbf{C}_{max} = \arg\forall_i \max|\mathbf{C}_i|$ where $|.|$ is just defined as the cardinality of the set. Let the points in $\mathbf{C}_{max}$ be described by the cubic spline curve $\mathbf{S}_{max}$ which has p control points and is defined in terms of a parameter s in the following way:

$$\mathbf{S}_{max}(s) = \mathbf{a_0} + \mathbf{a_1}s + \mathbf{a_2}s^2 + \mathbf{a_3}s^3 \in \mathbb{R}^3, \tag{3.7}$$

where $\forall_{i\in\{0,1,2,3\}}\ \mathbf{a_i} \in \mathbb{R}^3$. Let the control points of $\mathbf{S}_{max}$ be $\mathbf{P}_t^{\mathbf{S}_{max}} = \{P_{i,t}^{\mathbf{S}_{max}} = [x_{i,t}^{\mathbf{S}_{max}}, y_{i,t}^{\mathbf{S}_{max}}, z_{i,t}^{\mathbf{S}_{max}}]\}$. Let's now define a function g which takes two clusters as arguments and returns 1 if the clusters contain points that are on different sides (referring to left/right) of the road and 0 otherwise. We say that a cluster $\mathbf{C}_i$ is on the left if the control point with the smallest x-value of its corresponding spline curve $\mathbf{S}_i$ is on the left (equivalent to $y_{j,t}^{\mathbf{S}_i} > 0$, where $j = \arg\forall_k \min x_{k,t}^{\mathbf{S}_i}$). Given the function g, we define $\mathbf{C}_{opp}$ to be

the largest cluster on the opposite side of $\mathbf{C}_{max}$. So, $\mathbf{C}_{opp} = \arg\max_{\{i|g(\mathbf{C}_i,\mathbf{C}_{max})=1\}} |\mathbf{C}_i|$. Consistent with other definitions, let $\forall_k\ \mathbf{P}^{opp}_{k,t} \in \mathbf{C}_{opp}$, where $\mathbf{P}^{opp}_{k,t} = [x^{opp}_{k,t}, y^{opp}_{k,t}, z^{opp}_{k,t}]$. Let us define a function d_s that measures the distance between a cluster of points and a spline curve: $d_s(\mathbf{C}_i, \mathbf{S}_i) = \frac{1}{|\mathbf{C}_i|} \sum_{\forall_k} \min_{\forall_s} ||\mathbf{P}^i_{k,t} - \mathbf{S}_i(s)||_2$, where $||.||_2$ is the L-2 distance. Let $\mathbf{S}$ be a set of of all spline curves that are just horizontally shifted from $\mathbf{S}_{max}$ and thus are parallel to $\mathbf{S}_{max}$. A spline curve $\mathbf{S}_i$ is said to be horizontally shifted from $\mathbf{S}_{max}$ if and only if $\exists_{\Delta y}\ \forall_k\ (y^{\mathbf{S}_i}_{k,t} = (y^{\mathbf{S}_{max}}_{k,t} + \Delta y))$, where $y^{\mathbf{S}_i}_{k,t}$ and $y^{\mathbf{S}_{max}}_{k,t}$ are just the y-coordinates of the control points of splines $\mathbf{S}_i$ and $\mathbf{S}_{max}$. Then, we define $\mathbf{S}_{opp} \in \mathbf{S}$ to be the spline curve that best fits the points in $\mathbf{C}_{opp}$. More formally,

$$\mathbf{S}_{opp} = \arg\ \forall_{\mathbf{S}_i \in \mathbf{S}}\ \min d_s(\mathbf{C}_{opp}, \mathbf{S}_i), \tag{3.8}$$

This method ensures that $\mathbf{S}_{max}$ and $\mathbf{S}_{opp}$ are always parallel to each other as $\mathbf{S}_{opp} \in \mathbf{S}$. Our final spline curves to predict the immediate left and right lane marks are $\mathbf{S}_{max}$ and $\mathbf{S}_{opp}$.

3.2.5 Lane Boundary Registration

For generating a map of the environment, we use the open source code made available by Qin et al. [13]. The code outputs the rotation matrix and the translation vector of each frame relative to the first frame. Let the rotation matrix and translation vector at time t be ${}^L_W\mathbf{R}_t$ and ${}^L_W\mathbf{T}_t$ respectively. Then we can project the LiDAR point cloud, ${}^L\mathbf{P}_t$ to the first frame ($t = 0$) by the following,

$${}^W\mathbf{P}_t = {}^L_W\mathbf{R}^{-1} * {}^L\mathbf{P}_t - {}^L_W\mathbf{T}_t. \tag{3.9}$$

We use this formula to project lane marks found at time t to the world coordinates (or equivalently, coordinates relative to the $t = 0$ frame).

For traffic sign detection, note that it is not important to know the precise location of

the traffic sign in 3 dimensions at time t as long as we know that there is a traffic sign at that time. Therefore, we don't store/compute the precise location of traffic signs. We only store the times (or equivalently, frame numbers) at which we predict that there should be a traffic sign. We also provide the confidence we have in our belief.

3.3 Traffic Sign Detection

Our traffic sign detection module has two parts: we generate region proposals to extract image regions that may contain the traffic signs given the input image and the corresponding LiDAR scan; we find which traffic signs each region may contain from the region proposals.

3.3.1 Region proposal

As stated before, our region proposal method relies on the simple fact: traffic signs are mandated to have high retroreflectivity by law [46]. Our simple, yet highly accurate, region proposal method only considers LiDAR reflectivity to detect traffic signs. We first project $\mathbf{P}_t$ onto $\mathbf{I}_t$. Let this newly defined "image" be denoted by $\mathbf{I}_t^L$. The pixel intensity of the i^{th} point of $\mathbf{I}_t^L$ is the reflectivity of the projected LiDAR point. If there is no LiDAR point projected on a particular pixel, its intensity is 0. An example $\mathbf{I}_t$ and its corresponding $\mathbf{I}_t^L$ is shown in Figure 3.3a and Figure 3.3b respectively.

Our objective now is to find regions of similar and high brightness in $\mathbf{I}_t^L$. This corresponds to regions of similar and high reflectivity. The reason we find regions of similar reflectivity is because retroreflectivity is a property of material and so a traffic sign should have almost the same reflectivity on its entire surface. To discard low reflectivity regions, we threshold $\mathbf{I}_t^L$ and then dilate it. The dilation is a preparatory step for finding regions of similar and high reflectivity, which ensures high recall of traffic signs in the image. The thresholded and dilated $\mathbf{I}_t^L$ is shown in Figure 3.3c.

After this we run the DBScan clustering algorithm [47] on the image with a custom

way to find distance between two pixels $\mathbf{p}_1$ at (x_1, y_1) and $\mathbf{p}_2$ at (x_2, y_2). If r_i is the reflectivity (also intensity) of pixel $\mathbf{p}_i$, then the distance function, d, is so defined:

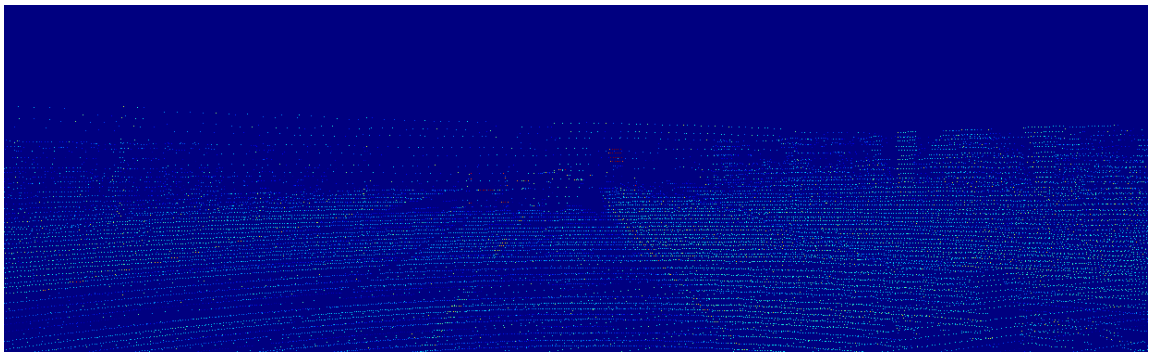
$$d(\mathbf{p}_1, \mathbf{p}_2) = \begin{cases} \infty & \text{if } |r_1 - r_2| > t_r \\ ||\mathbf{p}_1 - \mathbf{p}_2||_1 & \text{otherwise} \end{cases} \tag{3.10}$$

where t_r is the maximum difference of reflectivities we allow (set to 0.75 in our experiments) and $||.||_1$ is the L1 norm.

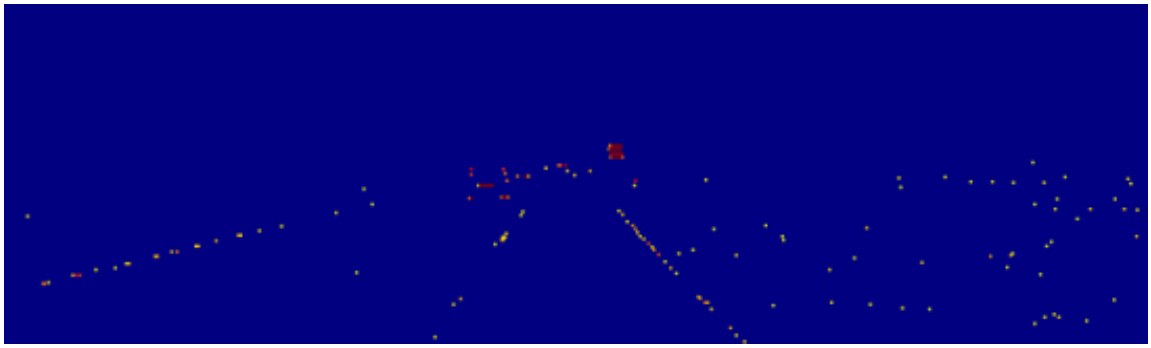
After we get clusters of points with similar and high intensity, we merge all nearby clusters to reduce the final number of clusters and make processing after this faster. We then remove very small clusters (area containing 200 pixels or less), which are small enough that the neural network applied later would anyway not be able to get any meaningful information from that. Lastly, we expand the size of the remaining bounding boxes and return the final expanded bounding boxes. The final predictions are shown in Figure 3.3d.

(a) Original Image

Figure 3.3: Traffic Sign Detection Algorithm



(b) Projected LiDAR points onto the image plane (Jet Color Map)



(c) After thresholding and dilating (Jet Color Map)

(d) Final predictions of traffic signs

Figure 3.3: Continued

3.3.2 Multi-label classification with ResNet

Firstly, we need to understand the need of having a multi-label classification network instead of a regular classification network. There are many cases when one bounding box contains two or more traffic signs. Although one solution would be to increase the

number of bounding boxes and restrict each box to have only one traffic sign, that would compromise the simplicity and the efficacy of the region proposal phase. So, we instead trained a network that could classify all traffic signs in a given bounding box.

We used a ResNet-50 [48] and added a sigmoid layer at the end so the output can be interpreted as probabilities. We then trained the network with a weighted binary cross entropy loss as shown below:

$$f(p,q) = 1/k \cdot \sum_{i \in [1,k]} -(w_1 \cdot q_i \cdot \log p_i + w_0 \cdot (1-q_i) \cdot \log(1-p_i)) \tag{3.11}$$

Here, p, q are k-dimensional vectors such that $\forall_i\, p_i, q_i \in (0,1)$. In practice, p is the output of the neural network and q is the ground truth probability. Therefore, p_i predicts the probability that the i^{th} traffic sign is in the input image. Note that w_1 and w_0 let us choose if we would prefer a higher false positive or a higher false negative. For example, when $w_1 > w_0$, it would mean that predicting $p_i = 1$ is more important than predicting $p_i = 0$, ultimately resulting in a higher false positive rate than a false negative one. Since it is very important that we don't miss a traffic sign, we set $w_1 = 3$ and $w_2 = 1$ during training. We train on a subset of the GTSRB dataset [49] and on some self-labelled images generated by the region proposal phase of our algorithm on the KITTI dataset [50].

An extremely important point that greatly increases the performance of the network is about the data augmentation techniques we use. During training, we apply color jitter to an image with the probability of 20%. Note that color jitter randomly changes the brightness, contrast, saturation and hue of the input image. This transform is very important as it ensures that the network is robust to changes in ambient light and other factors when predicting traffic signs. In our experiments, we allow brightness, contrast, saturation and hue to change by 50% of their original values. If the input image only has one traffic sign then, we also apply one of the following transform to each image with equal probability:

1. Resize: simply resize the image to the size the network expects (52 pixels x 52 pixels)

2. Random Crop: Choose a random 52 x 52 region of the input image

3. Center Crop: Choose the center 52 x 52 region of the input image

We use the idea of random and center crop to make the network be able to identify the traffic sign even when only a part of it is given. This is to prepare for any case when the detection algorithm only detects a part of the traffic sign or the traffic sign is occluded.In the case that there are multiple traffic signs in the image, we can't use random or center crop as it is far less likely that the cropped region would contain all traffic signs. In fact, in such cases, transforms like center and random crop would hurt us by forcing the network to learn some features from the background if the cropped region doesn't contain the traffic signs. Therefore, we only use color jitter and resize if the image has multiple signs.

During testing, we only use the color jitter and the resize transform as we have no prior knowledge about the input image. While testing, we first run our reflectivity-based region proposal algorithm to generate bounding boxes which are likely to contain traffic signs. All cropped parts inside the bounding boxes are then packed into one batch and we run our multi-label classification network to get all predictions. Note that, we don't run the neural network separately for each bounding box, but we run through all bounding boxes in one forward pass, which significantly saves time. In practice, due to the robustness of the detection algorithm, we have never seen an image which produced more than 5 bounding boxes. This means that the forward pass of the neural network is actually a fast process.

We extract LiDAR points that are projected into the traffic sign regions, and apply (3.9) to register the points in $\{\mathcal{W}\}$. Now we have all the elements to build the city-scale vector maps.

4. RESULTS AND FUTURE WORK

4.1 Results

As said before, our final result includes the left and right lane marks registered in the world coordinates. We also augment the final map by traffic signs wherever they are present. The final output of our algorithm on two KITTI datasets is shown in Figure 4.1. Note that the star denotes a traffic sign indicating speed limit 50 which was detected by our algorithm. The red curve denotes the left lane mark and the yellow curve denotes the right lane mark. It can be seen that the results are pretty reliable.

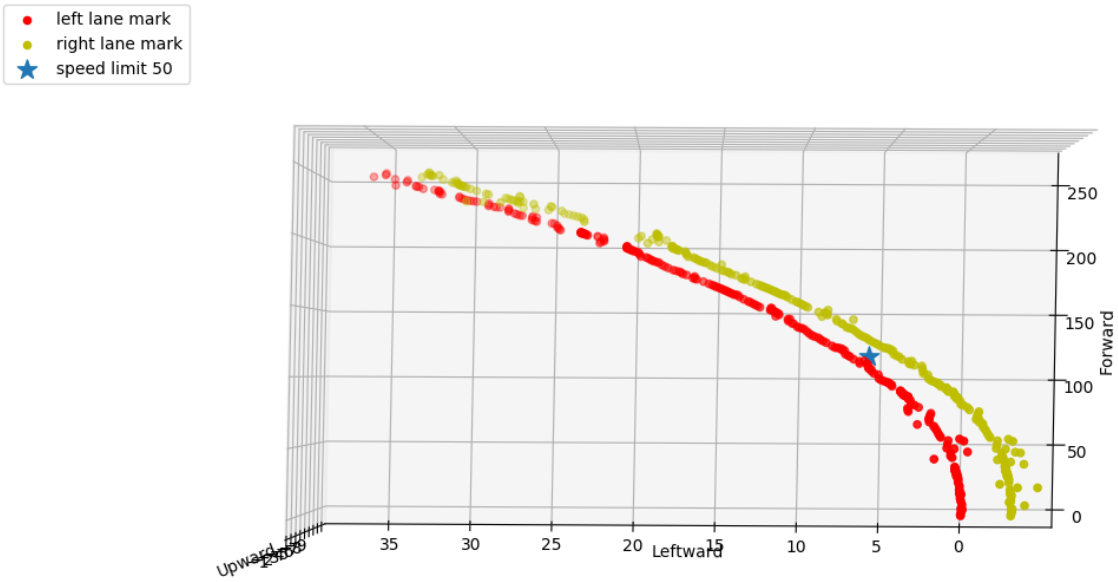
We also share some results of our lane detection algorithm in Figure 4.2. Note that the blue circles on images show the points that were detected as lane marks. The green curves show the spline curves fitted to lane marks.

Finally, we show some of the results of our novel traffic sign detection algorithm in Figure 4.3. Recall that the job of this algorithm is not to detect exactly traffic signs, but to highlight regions of interest in which traffic signs may be present as these proposed regions are processed by a neural network for final confirmation. The regions of interest are shown in green bounding boxes. As may be expected, note that all detected objects are highly reflective.

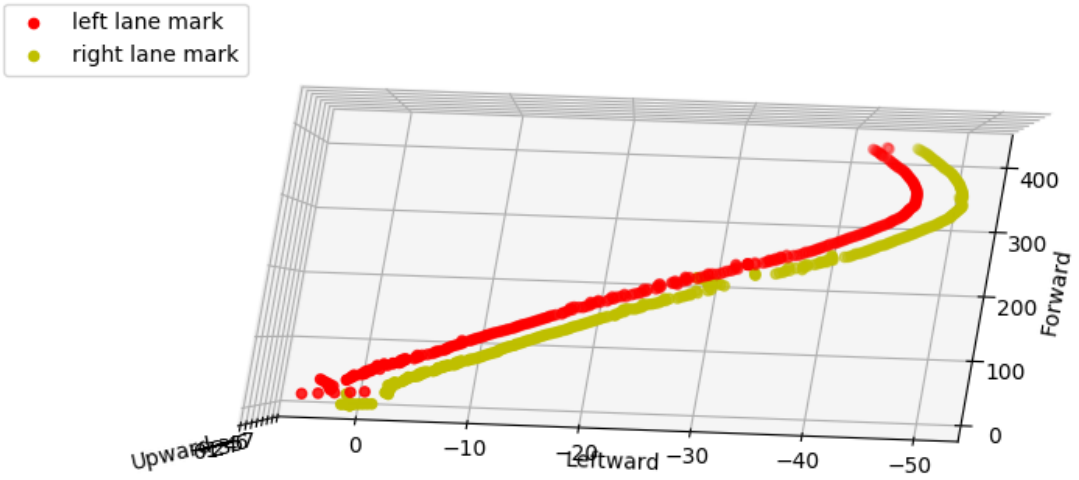
4.2 Future Work

There are many aspects in which our work can be extended:

1. Our traffic sign detection algorithm currently does not have an element of tracking in it. That is, the traffic sign results of the previous frame don't influence the results of the current frame. We can add a component of tracking to our algorithm because if there was a traffic sign at frame $t-1$, then there must also be a traffic sign in frame

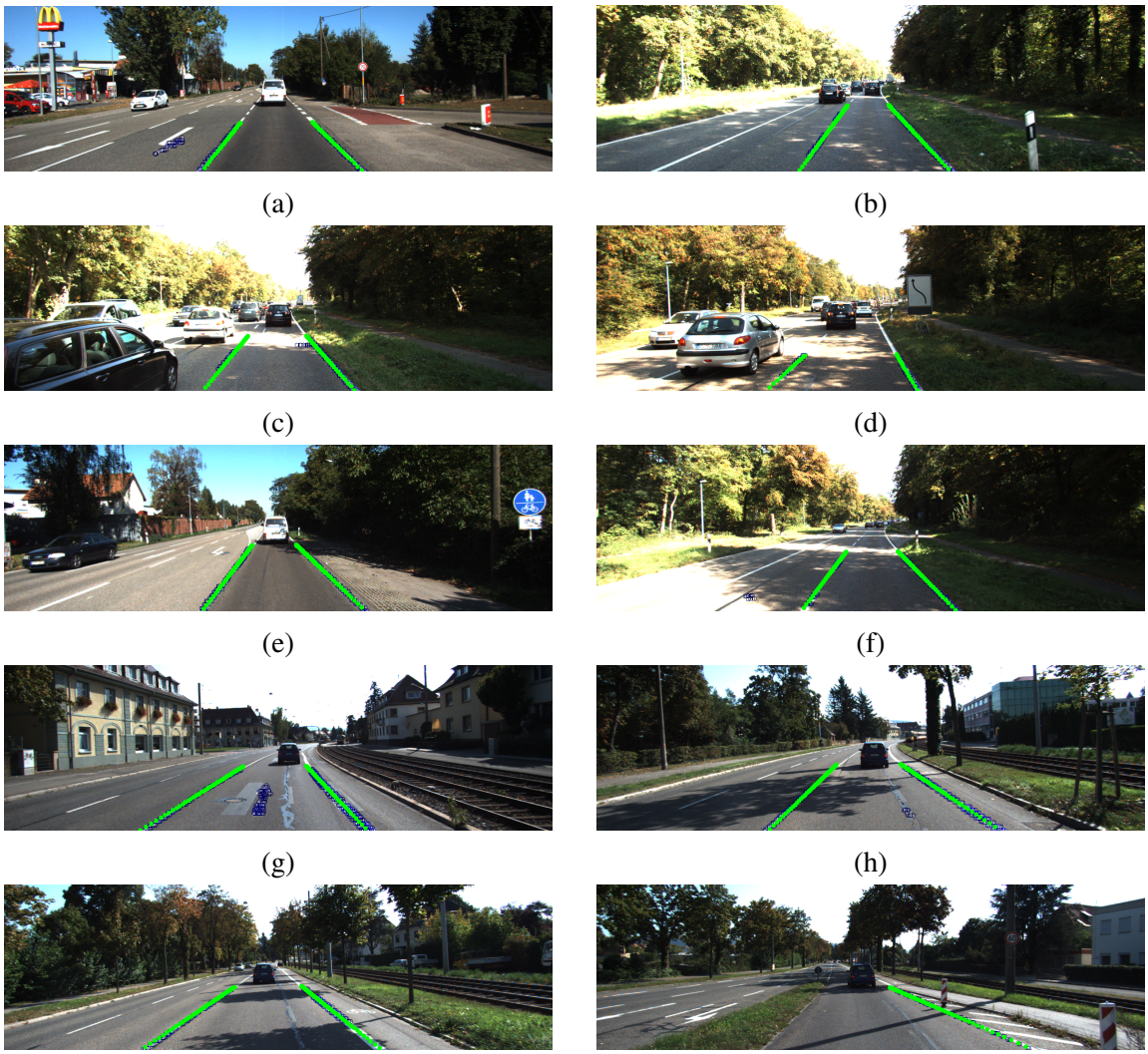


(a) KITTI dataset 2011_09_29_drive_0004



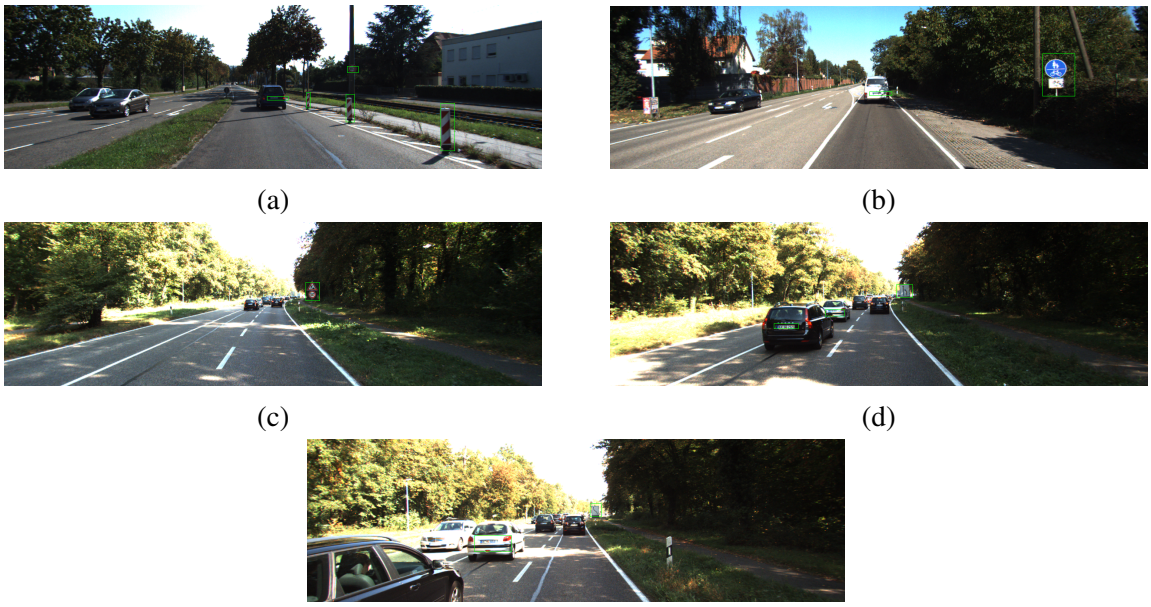
(b) KITTI dataset 2011_09_26_drive_0056

Figure 4.1: Final Results



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

Figure 4.2: Results of the lane mark detection algorithm



(a) (b) (c) (d) (e)

Figure 4.3: Results of the traffic sign detection algorithm

t as long as we don't pass it (which can also be measured by the distance from the vehicle to the traffic sign).

2. There are many cases in which lane marks are not present or too worn out to be detected on the road. Our current algorithm of course can't handle such cases. To deal with such cases, we can generate virtual lane boundaries by tracking lane boundaries from previous frames and making our best guess about the lane marks in the frame they were undetected.

3. We can also add information about the precise location of the traffic signs in 3D to the lane map we generated. This would help vehicles using our map to quickly verify if the traffic sign is present at the location indicated. If not, vehicles can request the map to be updated. This would ensure that the map always stays updated even when traffic signs are removed from some places.

4. As said before, the neural network used for traffic sign detection was trained on the GTSRB dataset [49] with few hand labelled images. The GTSRB dataset only contains images with one traffic sign. However, we would like our algorithm to classify signs even when there are multiple traffic signs in one image. An effort can also be launched in releasing such a dataset. We believe this would significantly boost performance.

REFERENCES

[1] H. Zhao, X. Qi, X. Shen, J. Shi, and J. Jia, "Icnet for real-time semantic segmentation on high-resolution images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 405–420, 2018.

[2] L. Magri and A. Fusiello, "T-linkage: A continuous relaxation of j-linkage for multi-model fitting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3954–3961, 2014.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580–587, 2014.

[4] 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, 2014.

[5] 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, pp. 62–81, 1998.

[6] A. Borkar, M. Hayes, M. T. Smith, and S. Pankanti, "A layered approach to robust lane detection at night," in *Computational Intelligence in Vehicles and Vehicular Systems, 2009. CIVVS'09. IEEE Workshop on*, pp. 51–57, IEEE, 2009.

[7] F. Samadzadegan, A. Sarafraz, and M. Tabibi, "Automatic lane detection in image sequences for vision-based navigation purposes," *ISPRS Image Engineering and Vision Metrology*, 2006.

[8] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[9] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," tech. rep., SRI INTERNATIONAL MENLO PARK CA ARTIFICIAL INTELLIGENCE CENTER, 1971.

[10] J. C. McCall and M. M. Trivedi, "Video-based lane estimation and tracking for driver assistance: survey, system, and evaluation," 2006.

[11] 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)*, pp. 3261–3264, IEEE, 2009.

[12] K. Yamaguchi, A. Watanabe, T. Naito, and Y. Ninomiya, "Road region estimation using a sequence of monocular images," in *2008 19th International Conference on Pattern Recognition*, pp. 1–4, IEEE, 2008.

[13] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[14] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2174–2181, IEEE, 2015.

[15] A. S. Huang, D. Moore, M. Antone, E. Olson, and S. Teller, "Finding multiple lanes in urban road networks with vision and lidar," *Autonomous Robots*, vol. 26, no. 2-3, pp. 103–122, 2009.

[16] C.-Y. Chang and C.-H. Lin, "An efficient method for lane-mark extraction in complex conditions," in *Ubiquitous Intelligence & Computing and 9th International Conference on Autonomic & Trusted Computing (UIC/ATC), 2012 9th International Conference on*, pp. 330–336, IEEE, 2012.

[17] M. M. Kalayeh, E. Basaran, M. Gokmen, M. E. Kamasak, and M. Shah, "Human semantic parsing for person re-identification," *CoRR*, vol. abs/1804.00216, 2018.

[18] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," *arXiv preprint arXiv:1408.5093*, 2014.

[19] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Semantic understanding of scenes through the ade20k dataset," *arXiv preprint arXiv:1608.05442*, 2016.

[20] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *arXiv preprint arXiv:1606.02147*, 2016.

[21] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *IEEE conference on computer vision and pattern recognition (CVPR)*, vol. 1, p. 3, 2017.

[22] A. Bansal, X. Chen, B. Russell, A. Gupta, and D. Ramanan, "Pixelnet: Representation of the pixels, by the pixels, and for the pixels," *arXiv preprint arXiv:1702.06506*, 2017.

[23] C. Peng, X. Zhang, G. Yu, G. Luo, and J. Sun, "Large kernel mattersâĂŤimprove semantic segmentation by global convolutional network," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 1743–1751, IEEE, 2017.

[24] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pp. 2881–2890, 2017.

[25] A. Chaurasia and E. Culurciello, "Linknet: Exploiting encoder representations for efficient semantic segmentation," in *Visual Communications and Image Processing (VCIP), 2017 IEEE*, pp. 1–4, IEEE, 2017.

[26] E. Romera, J. M. Alvarez, L. M. Bergasa, and R. Arroyo, "Erfnet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2018.

[27] I. K. Adam W Harley, Konstantinos G. Derpanis, "Segmentation-aware convolutional networks using local attention masks," in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[28] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio, "The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2017 IEEE Conference on*, pp. 1175–1183, IEEE, 2017.

[29] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *ECCV*, 2018.

[30] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun, "Multinet: Real-time joint semantic reasoning for autonomous driving," *arXiv preprint arXiv:1612.07695*, 2016.

[31] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *European conference on computer vision*, pp. 346–361, Springer, 2014.

[32] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[33] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[34] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[35] D. CireşAn, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural networks*, vol. 32, pp. 333–338, 2012.

[36] A. d. l. Escalera, L. Moreno, M. A. Salichs, and J. M. Armingol, "Road traffic sign detection and classification," 1997.

[37] S. Pei, F. Tang, Y. Ji, J. Fan, and Z. Ning, "Localized traffic sign detection with multi-scale deconvolution networks," *arXiv preprint arXiv:1804.10428*, 2018.

[38] H. S. Lee and K. Kim, "Simultaneous traffic sign detection and boundary estimation using convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, 2018.

[39] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2110–2118, 2016.

[40] R. Qian, B. Zhang, Y. Yue, Z. Wang, and F. Coenen, "Robust chinese traffic sign detection and recognition with deep convolutional neural network," in *2015 11th International Conference on Natural Computation (ICNC)*, pp. 791–796, IEEE, 2015.

[41] R. W. Wolcott and R. M. Eustice, "Visual localization within lidar maps for automated urban driving," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 176–183, IEEE, 2014.

[42] P. Bender, J. Ziegler, and C. Stiller, "Lanelets: Efficient map representation for autonomous driving," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 420–425, IEEE, 2014.

[43] D. Dolgov and S. Thrun, "Autonomous driving in semi-structured environments: Mapping and planning," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3407–3414, IEEE, 2009.

[44] J. Choi, J. Lee, D. Kim, G. Soprani, P. Cerri, A. Broggi, and K. Yi, "Environment-detection-and-mapping algorithm for autonomous driving in rural or off-road environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 2, pp. 974–982, 2012.

[45] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[46] P. J. Carlson and M. S. Lupes, "Methods for maintaining traffic sign retroreflectivity," tech. rep., 2007.

[47] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise.," in *Kdd*, vol. 96, pp. 226–231, 1996.

[48] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[49] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The German Traffic Sign Recognition Benchmark: A multi-class classification competition," in *IEEE International Joint Conference on Neural Networks*, pp. 1453–1460, 2011.

[50] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.