

# Technologies for Safe & Efficient Transportation

THE NATIONAL USDOT UNIVERSITY  
TRANSPORTATION CENTER FOR SAFETY

Carnegie Mellon University

UNIVERSITY of PENNSYLVANIA

---

Up-to-date city maps for modeling,  
planning, and assistive technologies

---

FINAL RESEARCH REPORT

Christoph Mertz

Contract No. DTRT-13-GUTC-26

## **DISCLAIMER**

The contents of this report reflect the views of the authors, who are responsible for the facts and the accuracy of the information presented herein. This document is disseminated under the sponsorship of the U.S. Department of Transportation's University Transportation Centers Program, in the interest of information exchange. The U.S. Government assumes no liability for the contents or use thereof.

# Up-to-date city maps for modeling, planning, and assistive technologies

Christoph Mertz ([cmertz@andrew.cmu.edu](mailto:cmertz@andrew.cmu.edu))

## Contents

Introduction .....	3
Sidewalks.....	4
Detection of sidewalks in aerial images.....	4
Detection of tactile patches.....	4
Classifying Stop Signs with Neural Decision Tree .....	5
Night Time Stop Sign Detection .....	6
Detect and read traffic signs .....	8
Semantic segmentation .....	9
Road detection.....	9
Lane detection and multiple classes.....	10
Localization .....	12
Summary and Conclusion .....	14

## Introduction

Digital maps are important for many different aspects of intelligent transportation. They are needed to model traffic patterns, to plan infrastructure upkeep, or for navigation for different traffic participants. These maps not only need to contain roads and all the transportation relevant objects like lane markings and traffic signs, but also their state of repair, compliance with regulations, and suitability for various users. The last point is particular important for people who use wheelchairs, they not only need to know if there is a sidewalk but also if the sidewalk is wide enough and well maintained. Traditional methods to create such maps are manual surveying or surveying vehicles that make use of specialty sensors. These methods are often cost prohibitive to keep maps up-to-date. Our proposed approach is to use inexpensive sensors on a fleet of vehicles that drive on the road for other purposes. We built on our experience with creating maps of road damage and stop signs. In this work we expand our detection to other traffic signs and lane markings and estimate retro-reflectivity of signs. We also want to detect damage, vandalism and vegetation overgrowth. We paid particular attention to gather information for traffic participants other than drivers by detecting sidewalks and tactile patches.

## Sidewalks

### Detection of sidewalks in aerial images

We want to be able to detect all the sidewalks in a city. A good candidate for the raw data are aerial or satellite images as they are readily available from web sites like Google Maps. We trained a CNN (Multinet<sup>1</sup>) using Google Map satellite images and a geographical data set from Washington D.C. Department of Transportation as the ground truth labels (Figure 1 left and middle).



Figure 1 An aerial image of a D.C. intersection (left) and the corresponding mask image (middle) that represent geographical information of the intersection. In the mask image, white, gray, and black pixels represent sidewalks, roads, and background respectively. On the right is an example of a classified test image. Red (blue) means high (low) probability of sidewalk.

After the CNN was trained, we tested it on new aerial images. A typical result can be seen in Figure 1 (right). The classifier can consistently find the sidewalk. The largest source of errors are trees. The reason is that most times trees are background, but in some cases, they can be above the sidewalk or above the street.

### Detection of tactile patches

The same CNN used in the previous section was also used to detect tactile patches in street level images. In this case, we had to label the images by hand.



Figure 2 Detection of tactile patches are indicated with bright green. Most of the classification is correct. On the left some areas are missing (missed detection) and on the right there are a few false detection.

Some typical results can be seen in Figure 2. The accuracy of the detection is about 90%.

---

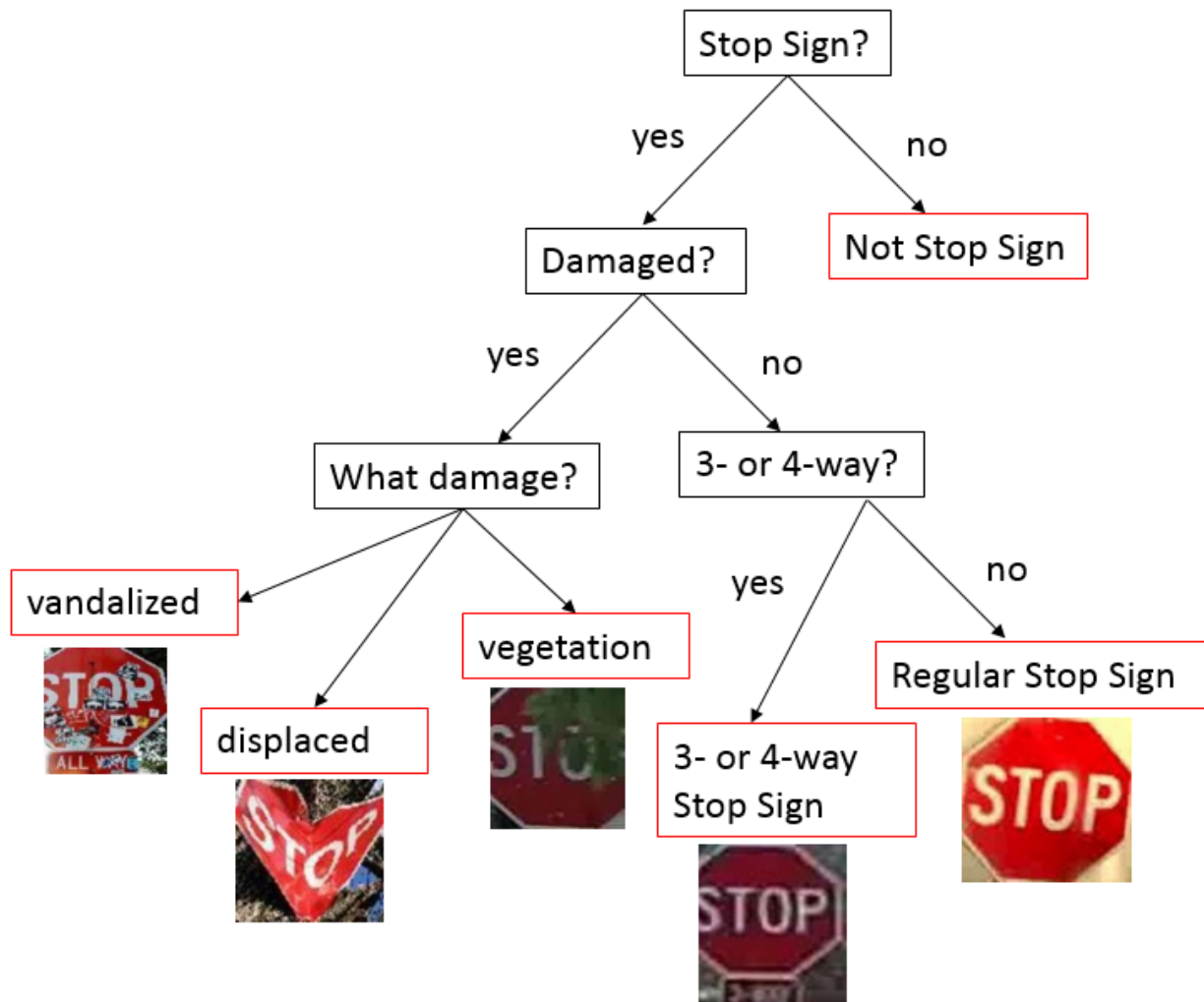
<sup>1</sup> Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., & Urtasun, R. (2016). MultiNet: Real-time Joint Semantic Reasoning for Autonomous Driving. arXiv preprint arXiv:1612.07695.

## Classifying Stop Signs with Neural Decision Tree

We want to be able to detect stop signs and classify their damage, i.e. we want to distinguish following 6 cases:

1. Not Stop Sign
2. 3- or 4-way Stop Sign
3. Displaced Stop Sign
4. Vandalized Stop Sign
5. Vegetation Covered Stop Sign
6. Undamaged Stop Sign

Usually one trains one neural detector with all the available classes. We found that a single neural net to categorize all the classes performs poorly, it especially lost accuracy with the first class (not Stop Sign). We therefore tested a Neural Net Decision Tree. The idea is that one first determines stop sign vs. not-stop sign and then determine the sub-categories. This is illustrated in following decision tree, where each decision made by a separate neural net:



The resulting confusion matrix is as follows:

		predicted					
		not	3/4 way	displaced	vandalized	vegetation	undamaged
true	not	518	0	19	5	0	10
	3/4 way	0	170	0	5	0	5
	displaced	0	0	128	0	0	0
	vandalized	0	24	15	108	4	42
	vegetation	0	0	0	0	0	0
	undamaged	0	19	94	47	68	1847

		predicted					
		not	3/4 way	displaced	vandalized	vegetation	undamaged
true	not	94%	0%	3%	1%	0%	2%
	3/4 way	0%	94%	0%	3%	0%	3%
	displaced	0%	0%	100%	0%	0%	0%
	vandalized	0%	12%	8%	56%	2%	22%
	vegetation	n/a	n/a	n/a	n/a	n/a	n/a
	undamaged	0%	1%	5%	2%	3%	89%

The system is able to predict most of the signs with about 90% or higher accuracy. It has difficulties detecting the vandalized stop signs. There were not enough stop signs with vegetation overgrowth in the test set.

## Night Time Stop Sign Detection

Besides being undamaged it is important for a traffic sign to have good retro-reflectivity. The retro-reflectivity can be measured directly with a calibrated light source and light meter or the adequate brightness of a sign can be evaluated by an inspector driving at night. Such methods are tedious and costly. We tested if we can detect traffic signs at night and determine if it is bright enough when illuminated by a standard headlight.

First, we need to detect the traffic sign, in our case a stop sign. When we use a detector that is only trained on day images, we get a 92% detection/30% false positive rate. Training with night images improves the rates significantly to 98%/5%. In Figure 3 is an example of a false detection and a correct detection.



Figure 3 Stop sign detection at night. On the left is one failure case where a red light is mistaken for a stop sign. On the right is a correct detection.

Once we have detected the stop sign, we can determine its brightness in the image. When the stop sign is directly in front of the vehicle it will be brighter, because the headlights shine directly on it. The further the stop sign is to the side, the darker it will be. Since we drive past the signs and record a video we have many images (frames) of the sign at various angles.

Figure 4 shows two examples. One has a stop sign with good retro-reflectivity and the other is faded. For both we show the intensities in the RGB channels for various viewing/illumination angles (expressed as percentage distance from the edge of the image).

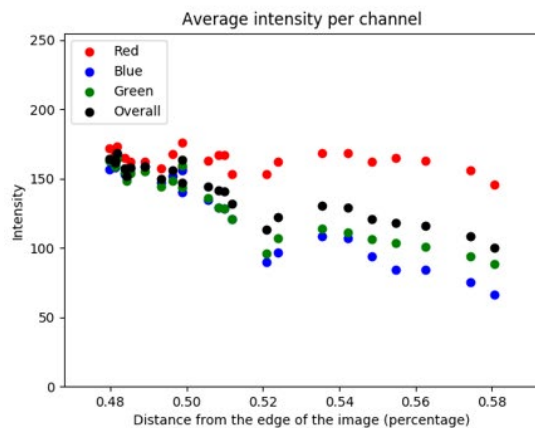
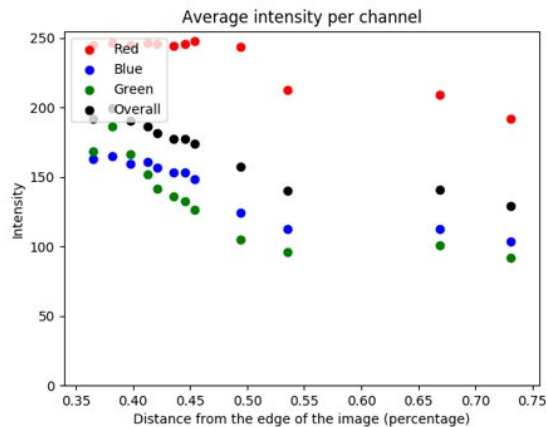


Figure 4 Retroreflectivity of the traffic sign are indicated by their brightness. On top right is a sign with good retroreflectivity. The intensity of the red channel is high, almost to the point of saturation (value around 255). On the bottom right is a faint stop sign. The intensity in the red channel is much lower, around 150.

One can see intensity dropping for larger angles as expected. More importantly, the absolute intensity is significantly lower for the faded stop sign. The drop is most pronounced in the red channel. This of course is expected since the stop sign is mostly red.



## Detect and read traffic signs

Many traffic signs have text on it. From simple texts like “STOP”, “25 mph”, to many words describing parking regulation or appealing to good driving behavior (Figure 5).

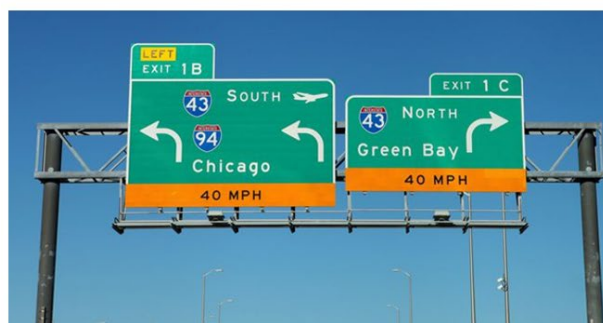


Figure 5 Traffic signs with text.

To detect and understand these signs the object detectors methods described above will not work. One needs text detection and recognition. Most common are two step methods. In the first step the algorithm finds a box in the image that contains text. In the second step the text is read (optical character recognition, OCR).

We used the Foo & Bar method CRNN<sup>2</sup> to detect and recognize text. Their method used quadrangle regression network for text detection, and then used homography to transform quadrangle regions to rectangles and finally CRNN for recognition. We applied this method to overhead traffic signs and got the results shown in :

### Reading signs:



### text detection

EXIT 1B  
Green Bay  
43 NORTH  
EXIT 1C  
SOUTH  
Chicago  
40 MPH  
40 MPH

### word recognition

e---xx-i--t--m--i---b--- => exitmib  
g--r-e-e-n---b--a---y-- => greenbay  
s-----n--o--r-t--h--- => snorth  
e-----xx---i---t--- => exit  
s-----o---u---t---h--- => south  
c---h---i--cc--a---g---o-- => chicago  
44-----o-----m---p---h-- => 40mph  
1---o-----m---p---e-- => 10mpe

Figure 6 Reading of traffic signs. On the left is a typical overhead sign. The detected text boxes are in the middle and on the right are the recognized words.

Some of the text it recognizes correctly (“Greenbay”) while it misreads others (“snorth” instead of “North”). It has difficulties with numbers and letters that indicate an exit or road number (“1B”). That

<sup>2</sup> <https://github.com/bgshih/crnn>



kind of texts were not sufficiently represented in the training data. We therefore implemented scripts that can create synthetic texts on traffic signs in real images (Figure 7).



Figure 7 A synthetic road sign with text on a real image.

These images can then be used to better train a classifier that can read traffic signs.

## Semantic segmentation

In semantic segmentation each pixel in an image is classified. In the next two sub-sections we will discuss our work on road detection with MultiNet and detection of many classes with another

### Road detection

The deep network MultiNet<sup>1</sup> can do joint classification, detection and semantic segmentation via a unified architecture where the encoder is shared amongst the three tasks. By sharing the encoder the network is faster than others. In the schematic below the encoder is the CNN-VGG16, which is then used by the fully convolutional networks (FCN):



We trained the network with labeled data from the KITTI dataset<sup>3</sup> and with our own labeled data. This combined data set gives the best results. Our own data was not large enough and the KITTI data set by itself did not generalize completely to our data. The KITTI data was recorded in Germany whereas our data is from Pittsburgh and its surrounding. Two example semantic segmentations are shown below:

<sup>3</sup> [http://www.cvlibs.net/datasets/kitti/eval\\_road.php](http://www.cvlibs.net/datasets/kitti/eval_road.php)



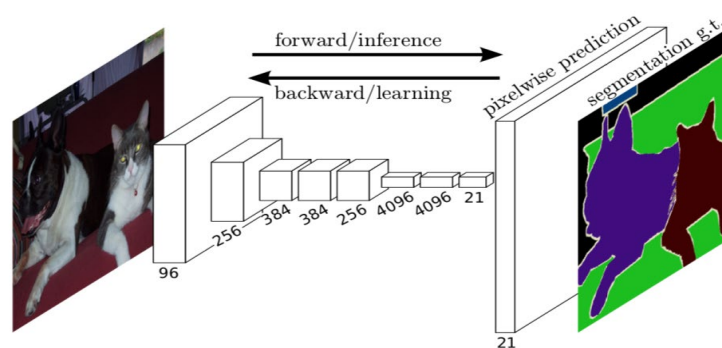
Figure 8 Semantic segmentation of roads. Red (blue) indicate high (low) probability of being road.

We evaluated the results and achieved an average precision of 92%.

### Lane detection and multiple classes

We are interested in lane markings. They are important for drivers to recognize the lanes of the road. They are also a significant maintenance item, on heavily traveled roads like the PA Turnpike they have to be repainted twice a year. We want to detect them and in the future we also want to determine any damage and loss of retro-reflectivity.

A schematic of a fully convolutional networks (FCN)<sup>4</sup> is shown below



<sup>4</sup> Tutorial, CVPR15 Caff , Jon Long, and Evan Shelhamer. "Fully Convolutional Networks."

In past years many datasets have been published with semantically labeled images for training and testing, but none of them had lane markings labeled. Only recently became the Mapillary dataset<sup>5</sup> available which has lane markings labeled separately from roads.

We wanted to explore the best way to train a lane marking classifier. Mapillary dataset has 66 different classes. The question is if training with only two classes (lane marking and not lane marking) or all 66 classes simultaneously would give better results. Figure 9 shows the outcome.



*Figure 9 Top is a raw image. The middle image has the detected lane markings colored in green. The detector was trained with two classes. The bottom image shows the result of a detector trained with 66 classes (sky=blue, vegetation=green, road=purple, lane markings=white, etc.).*

The top is the raw image. In the middle image the predicted lane markings are colored in green. It gets most of the lane markings with a few false detections on the right side. The lower image shows the result with all 66 classes. It can predict broad classes like sky and vegetation fairly well, but does perform poorly in detecting lane markings.

Figure 10 shows three more classification results of the 66 classes.

---

<sup>5</sup> <https://blog.mapillary.com/product/2017/05/03/mapillary-vistas-dataset.html>












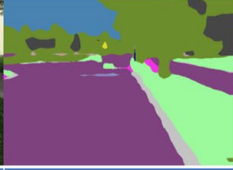
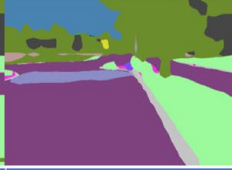
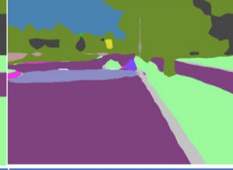
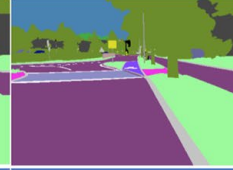
Input image	Epoch 30	Epoch 90	Epoch 150	Ground truth
				
				
				
<b>Training</b>	loss: 0.4985   acc: 0.8874	loss: 0.3482   acc: 0.9071	loss: 0.2847   acc: 0.9144	<b>Ground truth</b>
<b>Validation</b>	pixel accuracy: 0.872545	pixel accuracy: 0.879810	pixel accuracy: 0.881637	<b>Ground truth</b>

Figure 10 Three examples of semantic segmentation. Raw images are on the left and the ground truth is on the right. The results are shown for three different amounts of training (30, 90, and 150 epochs). The pixel accuracy for training and validation is on the bottom. The detector was trained for 66 classes (sky=blue, vegetation=green, road=purple, lane markings=white, etc.).

In summary we find that for common classes like sky, vegetation and road one gets good results when training a model for all classes at the same time. However, for rare and more intricate classes like lane markings it is much better to train each class by itself. Of course, this will have the tradeoff that training and predicting will take significant more computational time.

## Localization

The detection of objects for inventory and assessment needs to be accompanied by the location of the objects. An estimate within 5-10 m is achieved by GPS. This is good for many cases. But often one wants to have better localization. One likes to know which lane the vehicle was traveling in and for high accuracy maps localization within about 1 cm is required. With Structure From Motion (SFM) algorithms it is possible to accurately align images with each other while at the same time creating a 3D point cloud of the objects. We surveyed available open source SFM tools and found ORB-SLAM2<sup>6</sup> and COLMAP<sup>7</sup> the best performing and most user-friendly candidates. ORB-SLAM2 is able to run online, i.e. it is designed to analyze a video stream in real time. COLMAP's strength is its robustness to changes in viewpoint, illumination, camera and even works with images taken at different times.

A good SFM reconstruction from a video that was taken while driving in a loop (Figure 11). Also shown is a satellite view of the same area.

<sup>6</sup> Real-Time SLAM for Monocular, Stereo and RGB-D Cameras, with Loop Detection and Relocalization Capabilities: [https://github.com/raulmur/ORB\\_SLAM2](https://github.com/raulmur/ORB_SLAM2)

<sup>7</sup> <https://colmap.github.io/>





Figure 11 Localization and 3D reconstruction with COLMAP. The locations of the individual frames are marked in red. The 3D points are small black points. At the bottom is a satellite image of the traversed area. The orange arrows indicate corresponding buildings.

The SFM was able to close the loop and give a good representation of the path. The loop was traversed two times and one can see that the two passes are aligned with each other.

Unfortunately, most of the time the SFM was not able to close the loop and instead got a result like the following. The green arrow indicate the series of frames that should have been connected.



Figure 12 Example of unsuccessful loop closure. The green arrow indicates the frames that should be next to each other.

We suspect that the reason for this problem is the rolling shutter of the smartphone camera. It introduces systematic distortions into the images which the standard SFM is most times not able to overcome. Overall, we found that COLMAP is somewhat better at dealing with this problem than ORB-SLAM2.

Rolling shutter could be dealt with in hardware or in software. The hardware solution is to use a global shutter camera. But these cameras are not available in smartphones and in general they have less resolution than rolling shutter cameras. It is also possible to correct rolling shutter effects in software. This is a tedious and complicated method which can currently only be applied when the images are in a video sequence.

## Summary and Conclusion

The fields of deep learning and 3D reconstruction from images have been rapidly developing. We have been updating our detection and analysis tools to take advantage of these new techniques. We showed that it is possible to detect and analyze a broad range of road infrastructure, from traffic signs during day or night, lane markings, sidewalks, or tactile patches. It is possible to read traffic signs and determine if they are damaged. 3D reconstruction is a useful tool to locate images in relation to each other. With all these tools it is now possible to create a detailed map of the road infrastructure using images captured with a smartphone.