OBSTACLE DETECTION ON SIDEWALKS WITH A SINGLE CAMERA

Luca Rettenberger
Ravensburg-Weingarten
University of Applied Sciences

Project Report for Computer Vision
Supervised By: Prof. Dr. Stefan Elser

July 3, 2020

ABSTRACT

Obstacle detection requires stereo vision most of the time, so two cameras to extract 3D data from digital images. If only one camera is available most approaches will not work. StixelNet is a deep convolutional network approach which tries to detect obstacles with only a single camera. This approach is evaluated in two ways. First the network is reimplemented and trained on the provided autonomous driving dataset. Then StixelNet is evaluated both on the provided dataset and a new dataset which is generated especially for this work. The generated dataset is used to evaluate whether the network can also be used to detect obstacles from a different domain if it has been trained on the autonomous driving dataset (without retraining). The results show that the performance drops significantly in the unknown domain but remains reasonable. This leads to the conclusion that the network may be suitable to detect obstacles with high accuracy in a similar domain with some retraining.

1 Introduction

Obstacle detection is a very important subcategory of computer vision. Especially for the visually impaired, a cheap and robust solution for detecting obstacles can have a great improvement in quality of life, as studies have shown that being visually impaired (especially in higher ages) leads to a less fulfilling life [1, 2]. Visual impairment is a global problem which hits poor countries the hardest, since medical treatment in such countries is often not satisfactory [3]. Many of the visual defects people have to deal with in poor countries could have been prevented if treated by skilled personnel [3]. A person who cannot afford to have their visual impairment treated will most certainly not have the possibility to acquire a dedicated device for helping them navigate in everyday life. Fortunately there are many mobile phone users, even on poor continents [4, 5]. So developing a system which detects obstacles on a sidewalk with a single camera (in this case most certainly a mobile phone camera) could be a first step into the direction of helping visually impaired people who cannot afford additional devices.

Most of the research work in the field of obstacle detection is either done in the area of mobile robots [6, 7] or autonomous driving [8, 9, 10]. Few studies explore the possibilities for helping the visually impaired with such systems. [11] developed a walking guidance system for the visually impaired which uses stereo vision for detecting obstacles. This system not only uses two cameras for estimating depth but also employs additional sensors for obstacle detection. [12] established a smart walking stick which uses multiple sensors for detecting obstacles. [13] developed a similar system which uses

computer vision approaches to detect paths and possible approaching obstacles. All of those solutions either use multiple sensor inputs or stereo vision (two cameras) to detect obstacles. Fortunately there are some promising approaches in autonomous driving which try to solve the problem of detecting obstacles with just a single camera. Most notably the StixelNet approach developed by Levi et al. [14] (and its improved version in [15]). StixelNet has shown that it is possible to detect obstacles just from the input of a single camera mounted to a car.

This work evalutes the StixelNet approach for obstacle detection on roads on a well-known dataset and tries to find out if it is possible to transfer it to the domain of sidewalks without training it on new labeled data. Since there is no open source implementation of StixelNet we implement and train it by ourselfs.

2 Methods

In this section the setup for the conducted experiment is explained in detail. First of all the Stixel-World representation is introduced in section 2.1. Then the network architecture used in this work is presented in section 2.2. After that the two used datasets are presented in section 2.3 and section 2.4. Then the used preprocessing steps on the datasets are explained in section 2.5. Next eplains Section 2.6 the loss function used in this work. Finally, are details on the training and used parameters given in section 2.7.

2.1 Stixel-World Representation

The Stixel-World representation was introduced in [9] to estimate the free space in front of vehicles captured by camera images. This is done by dividing up each individual image into narrow rectangles called ”stixels”. Each stixel is vertical to the ground and has its lower edge where the first obstacle relative to the camera is located [9] . Stixels are often used to estimate objects and hence the stixel covers the found object beginning at the lower edge and ending at the upper edge [16], but since this work is concerned with obstacle detecting only the lower edges of the obstacles are of interest and hence only they are estimated. Figure 1 shows an example image with its corresponding stixels.

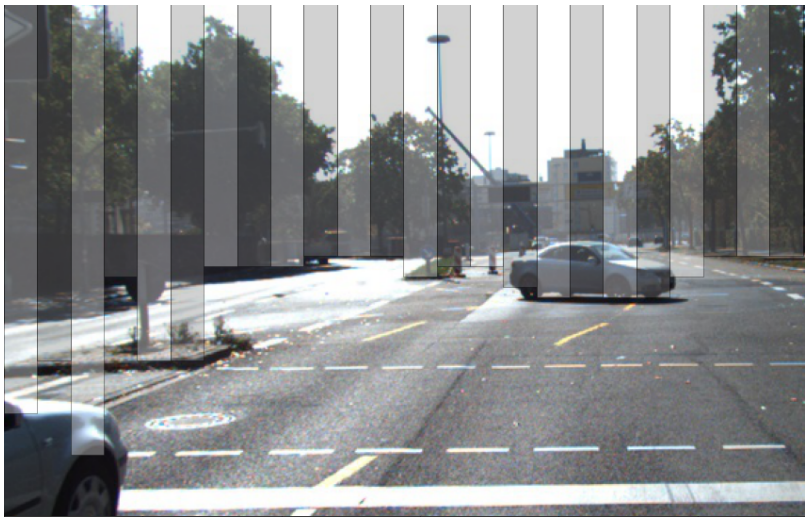


Figure 1: Example image for the Stixel-World representation. The found stixels are marked as slightly transparent gray and black bars. The area under each stixel indicates the suspected free space in front of the vehicle found by the StixelNet network. The car on the far left shows that StixelNet is not able to find obstacles which are so close that the lower edge of the object is not visible.

2.2 StixelNet

The StixelNet approach builds on top of the Stixel-World representation. The ”StixelNet” is a convolutional neural network (CNN) used to find the bottom pixel of an obstacle in a given stixel [14].

The peculiarity of the StixelNet approach is that it uses a single RGB image as input, which is rather unsual since tasks like obstacle detection can be solved much easier with stereo vision as with two cameras depth estimation is much simpler. The StixelNet network expects an image of dimensions $(w, h, 3)$, where w is the width, h is the height and 3 are the three RGB-channels [14]. The network expects the input image to be a vertical stripe of a source image, in which an obstacle should be detected. Consequently, determines w the width of each stripe. In this work w will always be set to 24 and h always to 370 (as in the original StixelNet paper [14] aswell). The network outputs a vector of size n. The output represents a probability distribution P which indicates where an obstacle is expected in the input stripe. If $n = h$ the output will be the probability $P(p)$ for each pixel p to be a lower edge of an obstacle, if $n < h$ the output will be the probability for each bin b of size $\frac{h}{n}$. In this work n will always be set to 50 (like in the original StixelNet paper [14]), which means we will have 50 bins of size $\frac{n}{h} = \frac{370}{50} = 7.4$. This alleged reduction of accuracy not only accelerates training but also enables the network to detect lower edges of obstacles which span multiple pixels (like it will almost always be the case).

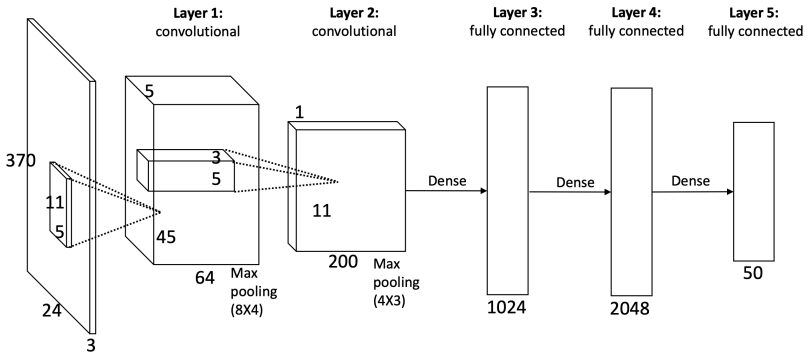


Figure 2: Sketch of the StixelNet architecture. The input is an single RGB-image of size 370×24 [14].

The architecture of the StixelNet is sketched in figure 2. Since StixelNet is a CNN it uses a combination of convolutional layers, each followed by a pooling layer. More precisely there are two convolutional layers followed by three fully connected layers. Both convolutional layers evaluate the input at each pixel position, so a stride of 1. The first convolutional layer uses 64 kernels and the second one 200 [14]. Both max-pooling layers apply the max filter to all non-overlapping subregions of the input [14]. The hidden fully connected layers have 1024 and 2048 neurons [14]. The last fully connected layer is the output layer of size n (which is 50 in this work).

2.3 Train Dataset

The dataset used in this work to train the StixelNet network is a subset of the raw data of the KITTI Vision Benchmark Suite [17]. Even though this dataset is usually meant for stereo vision, it can also easily be used for training the StixelNet network by just taking the image of a single camera rather than both. The KITTI dataset also provides the point cloud output of a Velodyne laser-scanner for each image. From this the ground truth (position of an obstacle) can be generated for an image by looking for abrupt changes in depth. [14] generated such ground truth data from the KITTI dataset by using a sophisticated technique which results in reliable values for obstacles. The used method is highly reliable but not complete, the ground truth, on average, covers about 25% of the whole width of the image (ideally for each column of width = 1 pixel a ground truth value should be set) [14]. To prevent too similar images in the dataset, only every fith frame in a sequence is labeled and used for training. The labeled data contains 386.225 ground truth points on 6.968 distinct frames. On average there are about 55.28 ground truth points in one frame. Figure 3 shows four example frames in which the found ground truth points have been visualized as red dots.

2.4 Sidewalk Dataset

To test the capability of the trained network to operate on data from the domain of "sidewalks" a new labeled dataset for testing is generated. It contains three short sequences with obstacles on sidewalks. The first is a speed camera, the second several trash bins and the third a trash bag. The images of the

Figure 3: Example frames from the KITTI dataset with obstacle labels visualized with red dots.

sequences are captured with a mobile phone. Unlike the KITTI dataset used for training the network this dataset has no automatically generated ground truth but contains hand labeled points. The dataset contains 3.802 ground truth points on 140 distinct frames. On average there are about 30.4 ground truth points in one frame. 15 of the 140 frames do not contain any ground truth points, since there are no lower edges of obstacles on those frames. There are 57 frames for the speed camera, 49 for the trash bins and 33 for the trash bag. Figure 4 shows some example frames from all three obstacles in the dataset.

Figure 4: Example frames from the dataset generated for this work to evaluate whether the StixelNet architecture is able to detect obstacles on a sidewalk.

2.5 Data Preprocessing

Both datasets are given with the frames and the corresponding ground truth points for each frame. The network however expects the input to be stripes rather than the whole frames. Because of that each frame is split into a set of stripes of size $(w_s, h_s, 3)$ so that the network can be trained on those stripes (in this work $w_s = 24$ and $h_s = 370$). The frames may need scaling so that each frame has the same height h_s . Each frame F with the dimensions $F_w \times F_h$ is split into $k = \lfloor \frac{F_w}{w_s} \rfloor$ non-overlapping stripes. The upper left coordinates for all stripes i in the set of all stripes S is defined as:

$$S(i) = (w_s * i, 0)\ ,\ i \in [0, k]. \tag{1}$$

We only need the upper left coordinates, since we know the height and width will always be h_s and w_s respectively.

For each ground truth point in one stripe the value of the most central point on the horizontal axis is used as the ground truth value of the stripe for training, since the network is trained with one true value only and not with a set of possible correct values. So for a set of points $P = \{(x_0, y_0,), (x_1, y_1) \dots (x_n, y_n)\}$ belonging to a frame F the subset of points which belong to each stripe $S(i)$ is selected by filtering P to only consider the set of points P_i which are within the stripe in the horizontal coordinate:

$$P_i = \forall (x_j, y_j) \in P,\ \ w_s * i < x_j < w_s * (i + 1). \tag{2}$$

The used ground truth value G for stripe i is then selected from P_i by taking the member which is the nearest to the horizontal center of the stripe, i.e. with the smallest distance to the center $w_s * i + \frac{w_s}{2}$:

$$G = min(abs(x_k - (w_s * i) + \frac{w_s}{2})),\ \ k = 0, 1 \dots |P_i|, \tag{3}$$

where $|P_i|$ is the length of P_i and x_k is the horizontal coordinate of $P_i(k)$. The vertical coordinate y_k of G is used as ground truth value for the respective stripe. Only stripes which contain at least one point are considered for training, if a stripe contains no point it is removed from the set of stripes. This whole process is visualized in figure 5.

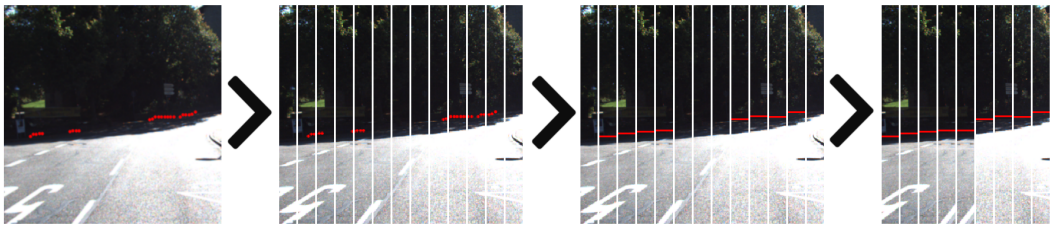


Figure 5: The process of generating the stripes for training visualized. In the first step the frame is split into stripes. Then the ground truth value for training is calculated with the given points in the stripes. Lastly are the stripes deleted which do not have any ground truth value.

This process results in 118.233 unique stripes with ground truth labels for the training dataset (KITTI) and 891 for the sidewalk dataset. For more details see table 1.

Dataset	**KITTI**	**Sidewalk**
# Frames	6.968	125
# Stripes	355.369	3.376
# Stripes with Label	118.233	891
Label/Stripe Ratio	0.33	0.26

Table 1: Details for the two used dataset in this work. **# Frames** is the number of total unique frames. **# Stripes** is the number of stripes generated from the unique frames. **# Stripes with Label** is the number of stripes which have a ground truth label. **Label/Stripe Ratio** is the ratio between all stripes and those who have a label.

The data augmentation steps in this work are limited to horizontal flips. This is because the ground truth values of the stripes are very prone to error if a stripe is augmented. The only (simple) safe augmentation technique are horizontal flips, because the most center point will still be the nearest to the center if the image is flipped and hence the ground truth will not change if a stripe is flipped horizontally.

2.6 PL-Loss

The used loss function in this work is newly developed by [14] especially for the use in StixelNet. The ground truth value $\hat{y}$ is used as an argument for the function $P(y)$ which linearly interpolates the output bins of the network. $P(y)$ is defined as:

$$P(y) = a_i \frac{(c_{i+1} - y)}{c_{i+1} - c_i} + a_{i+1} \frac{(y - c_i)}{c_{i+1} - c_i}, \tag{4}$$

where a_i and a_{i+1} are the outputs of the neurons i and $(i+1)$ [14]. The loss is defined as $-logP(y)$, which is the log loss of the linear interpolation $P(y)$ [14]. This loss function is called piecewise-linear probability loss (PL-Loss) [14].

PL-Loss has the advantage that it does not discretize the ground truth value $\hat{y}$ and hence can use this continuous value of $\hat{y}$ to train the network in such a way that the continuous nature of this problem is not lost in the bins. if $\hat{y}$ lays between two bins the loss function will output a small value if both bins are considered to be probable (i.e. if both bins have high values). If $\hat{y}$ lays exactly in the middle of two bin centers, both bins are equally important, the nearer $\hat{y}$ is to one of the bins the more important that bin gets.

2.7 Training

The network is solely trained on the stripes of the KITTI dataset. The dataset is split into train- and test-data. 80% (94586 stripes) of the stripes are used for training and 20% (23647 stripes) for testing. The dataset is **not** randomly split since randomly sampling frames would result in successive frames being both in the train- and test-dataset which will lead to overfitting (since successive frames will always be very similar). The 23647 stripes for testing are successive frames taken of a random position in the whole dataset.

We use the PL-loss and the Adam optimizer with a learning rate of 0.001. We train the network for 400 epochs on a batch size of 1000 stripes. After each epoch the network performance is tested on the test data. If the network performance increases from one epoch to the next (on the test data) the weights of the network will be saved. As a measurement for goodness of the network on the test data the value of the loss function is used. The network usually reaches the optimal configuration of weights long before the 400'th epoch.

3 Results

In this section the results for both datasets are presented. In the first step our trained implementation of the StixelNet is evaluated on the KITTI dataset in section 3.1 to analyze if the network performs as expected. After that the network is evaluated on the Sidewalk dataset in section 3.2 to analyse how well the network transfers its knowledge to the unknown domain.

To evaluate the models performance on both datasets we follow the approach of [14]. Firstly we take the output of the network $\tilde{y}[S(i)]$ for each stripe $S(i)$ in a set of stripes S to be evaluated. Then we calculate the interpolated value $\tilde{y_p}(i)$ with the $P(y)$ function of PL-Loss (see section 2.6): $\tilde{y_p}(i) = P(\tilde{y}[S(i)])$. With that we compute the absolute pixel-wise error δ between the estimation $\tilde{y_p}(i)$ and the ground truth value $\hat{y}(i)$ for the stripe. We do that for all stripes in the set of stripes S:

$$\delta = abs(\hat{y}(i) - \tilde{y_p}(i)),\ for\ i = 0, 1, \dots |S|, \tag{5}$$

where $|S|$ is the length of S. Now we can use δ to calculate the fraction of samples within an error threshold ϵ to get an accuracy rate E within that threshold. Formally we can calculate E by dividing the number of values in δ below the threshold ϵ by the number of all error values $|\delta|$:

$$E = \frac{1}{|\delta|} \sum_{i=0}^{|\delta|} \delta(i) < \epsilon. \tag{6}$$

Equation (6) expects that $\delta(i) < \epsilon$ yields "1" if the condition is true and "0" if not. After that we plot E against ϵ with increasing values for ϵ to get a "ROC-like" curve which shows how accurately the network estimates the ground truth values. To get a single-value error measurement we calculate the area under the curve (AUC). [14]

3.1 KITTI Dataset

We evaluate the KITTI dataset on the 23647 test-stripes which the network has not seen during train time. Figure 6 shows the plot for the error rate E against ϵ values in the range of $0 \leq \epsilon \leq 50$ with a step size of one (i.e. $\epsilon = 1, 2 \dots 50$). We achieve a AUC value of roughly 0.862, which is slightly worse than the score of 0.87 which the original authors of the StixelNet achieved. However, do we not know how the original authors rounded in their AUC value in [14]. Also does the plot given by the authors in [14] indicate that they did not use a step size of one for the plot, which would reduce the comparability of the two scores.

3.2 Sidewalk Dataset

We evaluate the Sidewalk dataset on all the 891 labeled stripes, since this dataset was never used during train time. The dataset is split into three sub-datasets only containing the frames of one of the three obstacles in the dataset. This is done to evaluate how well the network detects the different obstacles. Figure 7 shows the plot for the error rate E against ϵ values in the range of $0 \leq \epsilon \leq 100$ with a step size of one (i.e. $\epsilon = 1, 2 \dots 100$) for all three sub-datasets and the whole dataset. Figure 8 shows the plot with the same data but with ϵ values in the range of $0 \leq \epsilon \leq 50$.

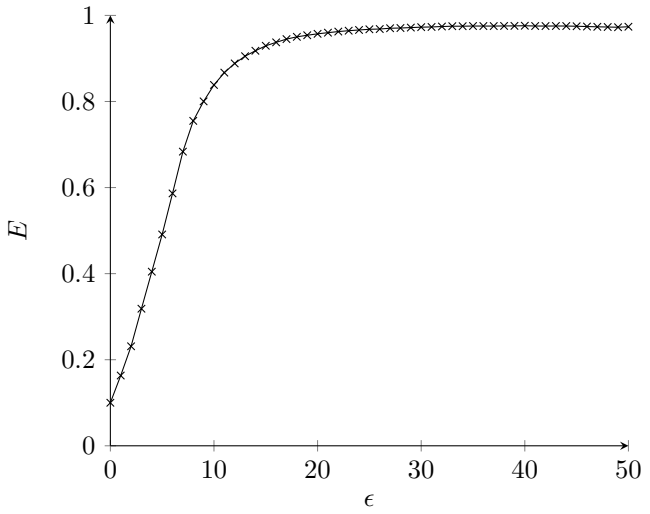


Figure 6: The error rate for the trained network on the KITTI test-data. The error rate E is plotted against ϵ in the range $0 \leq \epsilon \leq 50$. The area under the curve is 0.862.

4 Discussion

The AUC-value on the StixelNet dataset shows that the network was successfully trained like intended by the original authors. The slightly worse performance probably stems from marginally different weights. The results on the Sidewalk dataset show less clear results. StixelNet seems to have problems to detect some obstacles which may occur on sidewalks. The speed camera was not identified very well. This is probably due to that the speed camera was permanently installed into the sidewalk and hence was very different from what one could expect on a street (see figure 9a and 9b).

The performance seems reasonable if one evaluates the results of figure 7. Especially trash bags and trash bins are detected with pretty good confidence, if one considers that the network has never been trained on sidewalk data. On the maximum error threshold ϵ are trash bags detected with 89% confidence, speed cameras with 73% confidence, trash bins with 96% confidence and any obstacle (so all classes of obstacles) with 87% confidence. However is the maximum error threshold of 100 pixels very high, considering that the images are only 370 pixels high. It is obvious from these plots that in all classes the confidence still increases much at $\epsilon \approx 100$. This means that the error rate is often at around 100 pixels.

If one evaluates figure 8, where the maximum threshold is set to 50 pixels, the performance drops significantly (which is expected if we only allow the predicted value to be away half as many pixels from the ground truth). On $\epsilon = 50$ are trash bags detected with 74% confidence, speed cameras with 50% confidence, trash bins with 81% confidence and any obstacle (so all classes of obstacles) with 71% confidence. The test-data on the KITTI dataset has a confidence of 96% on $\epsilon = 50$. So there is still a big gap between the KITTI dataset and the Sidewalk dataset. The performance is still promising, especially on the trash bins and trash bags.

One significant problem, which gets evident on the Sidewalk dataset, is that the frames used for training are very unilateral. The ground truth values are all within a very small field of values. This is because the dataset was labeled with the help of a Velodyne laser-scanner mounted to a car. This means that the captured area will always be in the lower half of the frame. This has let to the problem that obstacles in the Sidewalk dataset are only detected very late (when they are in the lower half of the frame). Figure 9 shows that all obstacle are not detected until they are near to the camera. This is no problem if the camera is in a fixed position (like mounted to a car) but if the camera is held in a hand the obstacle may be near even if it is in the upper half of the frame. This lowers the achieved accuracy, because obstacles in an upper area of a frame will not be detected (like the middle image in figure 4).

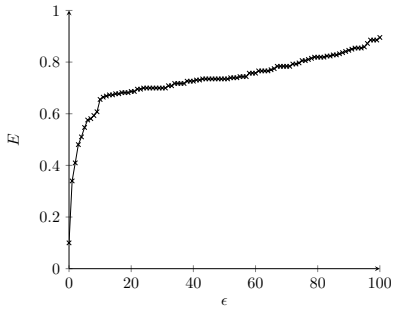
Also is the network not able to detect an obstacle if it is so near, that the lower border is not visible. This problem however is an inherent problem of the StixelNet approach and does also occur in the domain of autonomous driving.

The results show that the StixelNet network is able to detect obstacles of a slightly different domain (in this example the one of sidewalks) even if it has been trained on autonomous driving data. This can be done without retraining or fine-tuning the network on the new data. The performance is far away from the performance on the trained domain but under certain conditions the network can detect obstacles of objects it has never seen before.

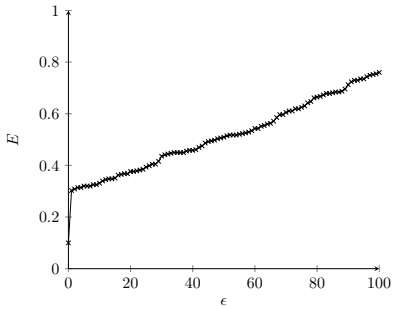
5 Conclusion

In this work we reimplemented and evaluated the StixelNet network. We showed that the network is able to detect obstacles on sidewalks even if it has been trained on a dataset which only contains images from the domain of autonomous driving. The performance on the sidewalk data is far from perfect but has great potential. One next logical step would be to fine-tune the network on the sidewalk data to enable it to detect certain obstacles which until now cannot be detected. For that the dataset generated for this work, which contains frames of obstacles on sidewalks, would need to be extended.

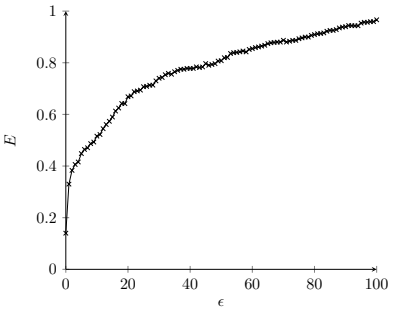
The Sidewalk dataset and StixelNet implementation can be found at: `https://github.com/lrettenberger/obstacle-detetion-on-sidewalks-with-a-single-camera`.



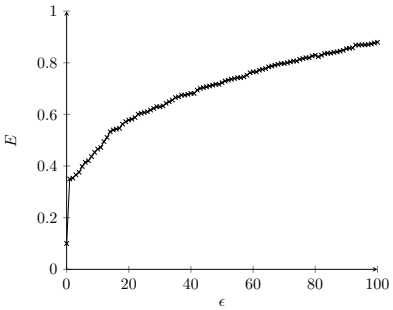
(a) Trash Bag. AUC: 0.734



(b) Speed Camera. AUC: 0.513

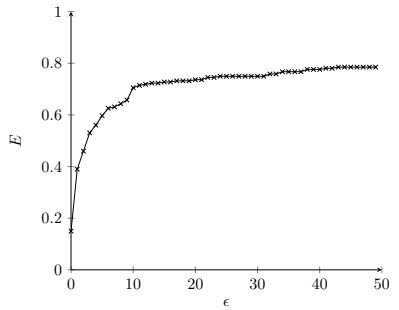


(c) Trash Bins. AUC: 0.798

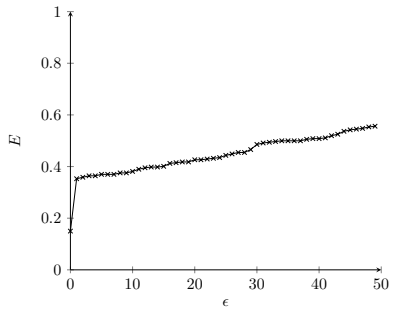


(d) All Obstacles. AUC: 0.682

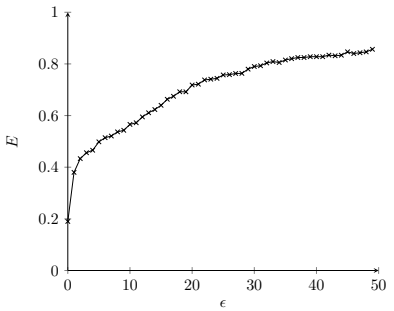
Figure 7: The error rate for the trained network on the Sidewalk-dataset. The error rate E is plotted against ϵ in the range $0 \leq \epsilon \leq 100$. For each plot the type of considered obstacle and the area under the curve (AUC) is given.



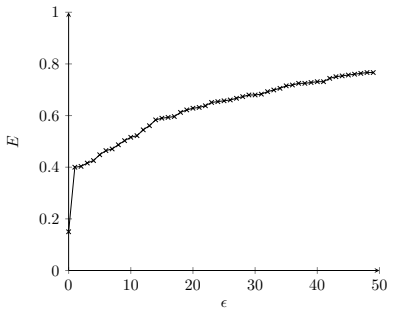
(a) Trash Bag. AUC: 0.649



(b) Speed Camera. AUC: 0.388



(c) Trash Bins. AUC: 0.637



(d) All Obstacles. AUC: 0.562

Figure 8: The error rate for the trained network on the Sidewalk-dataset. The error rate E is plotted against ϵ in the range $0 \leq \epsilon \leq 50$. For each plot the type of considered obstacle and the area under the curve (AUC) is given.

(a) Speed Camera far away.

(b) Speed Camera near.

(c) Trash Bin far away.

(d) Trash bin near

(e) Trash Bag far away.

(f) Trash Bag near.

Figure 9: Examples of far away and near frames of the three categories of the Sidewalk dataset. The red lines show that the network detected a possible obstacle in the respective stripe. The darker the red, the higer the confidence. The examples show that trash bags and trash bins are detected if they are near but not if they are far away. Speed cameras are not detected with high confidence.

References

[1] ILDEBRANDO APPOLLONIO, CORRADO CARABELLESE, LODOVICO FRATTOLA, and MARCO TRABUCCHI. Effects of sensory aids on the quality of life and mortality of elderly people: a multivariate analysis. *Age and ageing*, 25(2):89–96, 1996.

[2] Laurence G Branch, Amy Horowitz, and Cheryl Carr. The implications for everyday life of incident self-reported visual decline among people over age 65 living in the community. *The Gerontologist*, 29(3):359–365, 1989.

[3] World Health Organization et al. Global data on visual impairments 2010. *Geneva: WHO*, pages 1–5, 2012.

[4] Vodafone. Africa: The impact of mobile phones. *The Vodafone Policy Paper Series*, (2), 2005.

[5] Sherwin I DeSouza, MR Rashmi, Agalya P Vasanthi, Suchitha Maria Joseph, and Rashmi Rodrigues. Mobile phones: the next step towards healthcare delivery in rural india? *PloS one*, 9(8), 2014.

[6] Iwan Ulrich and Illah Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *AAAI/IAAI*, pages 866–871, 2000.

[7] Aniket Murarka, Mohan Sridharan, and Benjamin Kuipers. Detecting obstacles and drop-offs using stereo and motion cues for safe local motion. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 702–708. IEEE, 2008.

[8] Sebastian Ramos, Stefan Gehrig, Peter Pinggera, Uwe Franke, and Carsten Rother. Detecting unexpected obstacles for self-driving cars: Fusing deep learning and geometric modeling. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 1025–1032. IEEE, 2017.

[9] Hernán Badino, Uwe Franke, and David Pfeiffer. The stixel world-a compact medium level representation of the 3d-world. In *Joint Pattern Recognition Symposium*, pages 51–60. Springer, 2009.

[10] Gabriel L Oliveira, Wolfram Burgard, and Thomas Brox. Efficient deep models for monocular road segmentation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4885–4891. IEEE, 2016.

[11] Sang-Woong Lee, Seonghoon Kang, and Seong-Whan Lee. A walking guidance system for the visually impaired. *International Journal of Pattern Recognition and Artificial Intelligence*, 22(06):1171–1186, 2008.

[12] Laehyun Kim, Sehyung Park, Sooyong Lee, and Sungdo Ha. An electronic traveler aid for the blind using multiple range sensors. *IEICE Electronics Express*, 6(11):794–799, 2009.

[13] João José, Miguel Farrajota, João MF Rodrigues, and JM Hans Du Buf. The smartvision local navigation aid for blind and visually impaired persons. 2011.

[14] Dan Levi, Noa Garnett, Ethan Fetaya, and Israel Herzlyia. Stixelnet: A deep convolutional network for obstacle detection and road segmentation. In *BMVC*, pages 109–1, 2015.

[15] Noa Garnett, Shai Silberstein, Shaul Oron, Ethan Fetaya, Uri Verner, Ariel Ayash, Vlad Goldner, Rafi Cohen, Kobi Horn, and Dan Levi. Real-time category-based and general obstacle detection for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pages 198–205, 2017.

[16] David Pfeiffer and Uwe Franke. Efficient representation of traffic scenes by means of dynamic stixels. In *2010 IEEE Intelligent Vehicles Symposium*, pages 217–224. IEEE, 2010.

[17] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.