Deep Learning Based Traffic Signs Boundary Estimation

Emir Hrustic, Zhujun Xu, Damien Vivet

2020 IEEE Intelligent Vehicles Symposium (IV)
October 20-23, 2020. Las Vegas, USA

Deep Learning Based Traffic Signs Boundary Estimation

Emir Hrustic, Zhujun Xu, and Damien Vivet

Abstract— **In the context of autonomous navigation, the localization of the vehicle relies on the accurate detection and tracking of artificial landmarks. These landmarks are based on handcrafted features. However, because of their low-level nature, they are not informative but also not robust under various conditions (lightning, weather, point-of-view). Moreover, in Advanced Driver-Assistance Systems (ADAS), and road safety, intense efforts have been made to implement automatic visual data processing, with special emphasis on road object recognition. The main idea of this work is to detect accurate higher-level landmarks such as static semantic objects using Deep learning frameworks. We mainly focus on the accurate detection, segmentation and classification of vertical traffic signs according to their function (danger, give way, prohibition/obligation, and indication). This paper presents the boundary estimation of European traffic signs from an embedded monocular camera in a vehicle. We propose a framework using two different deep neural networks in order to: (1) detect and recognize traffic signs in the video flow and (2) regress the coordinates of each vertices of the detected traffic sign to estimate its shape boundary. We also provide a comparison of our method with Mask R-CNN [1] which is the state-of-the-art segmentation method.**

I. Introduction

Traffic sign recognition and detection have become well-known problems, especially with the increasing interest in autonomous driving vehicles and ADAS (Advanced Driver-Assistance Systems). Traffic signs provides strong semantic information such as speed limitations, directions, dangers etc., which are very useful for decision-making of such systems. From an another perspective, traffic signs may be considered as a very stable and easily detectable object in the environment because they are made by construction as very salient objects which can be used as landmarks. Usually, landmarks are defined as geometric features such as points, lines, or planes. In classical approaches, hand-crafted features like Harris corners [2], SIFT [3] or SURF [4] features are extracted from images and used as landmarks for 3D mapping. However these are not robust under various conditions (weather change, light, fog...), and their track is often lost from one frame to another. In order to overcome this issue, the use of higher level landmarks like traffic signs should be considered. Indeed their static nature, easy shapes and color codes makes them easier to detect, and as a consequence, makes such landmarks more robust than artificial hand-crafted ones. Yet, as we need to get the traffic sign pose in 3D mapping space, the classical detection methods using bounding-boxes are not precise enough. We required a pixel-wise detection in order to detect traffic signs boundaries for a better triangulation. On one hand, pixel-wise traffic signs detection can be seen as a semantic image segmentation problem, several methods tried this approach [5], but they all are computationally, time expensive and not adapted for critical real-time systems. On the other hand, because traffic signs are simple shapes like ellipses, triangles, rectangles or octagons (in the case of European traffic signs), a regression can be made to only estimate their vertices coordinates and as a result to get their boundaries. In this kind of approach, the coordinates of the boundaries corners can be detected across multiple consecutive frames, and the 3D coordinates can then be computed by classical triangulation using camera pose and camera internal parameters. In this article, we propose a framework which splits the problem into two parts: the first one consists in a Convolutional Neural Network (CNN) for traffic sign detection using bounding boxes. These bounding boxes are then cropped from the original image, re-sized and passed to the second part along with the predicted class. This second part is composed of multiple CNNs, one for each traffic sign shape. The activated CNN is given by the previous class prediction. Indeed each shape is composed of a different number of outputs vertices that must be estimated. Each of these CNNs applies regression to the given resized crop and predicts its vertices coordinates. At the end of the process, the shape of the traffic sign is obtained in the image and can be use for additional processing such as 3D reconstruction. Our method consists in a novel approach to regress traffic signs boundaries. To the best of our knowledge, it is the first method using regression on ellipses in order to estimate non-polygonal traffic signs boundaries. Note that this article focus on the detection and regression parts for the boundary estimation only and do not presents the 3D reconstruction part. The rest of this paper is structured as follows. Section II presents the related works of traffic sign detection. Section III explains the methodology used for solving this problem. Finally, section IV presents the results obtained with our approach and comparison with state-of-the-art segmentation approach.

II. Related work

For Semantic SLAM, recognizing and detecting landmarks objects in a frame are crucial steps for mapping and localization. With the increasing interest in Deep-learning these last years, Convolutional Neural Networks(CNN) have achieved state-of-the-art performance for solving such problems. Many Convolutional Neural Networks (CNN) architectures like VGG [6], GoogLeNet [7], ResNet [8] are able to

*This work was supported by the Agence Nationale de la Recherche (ANR-the French national research agency) (ANR-17-CE22-0001-01) and by the French FUI (FUI STAR: DOS0075476 00).

Authors are with Université de Toulouse, ISAE-SUPAERO, DEOS, Toulouse, France. `firstname.name@isae-supaero.fr`

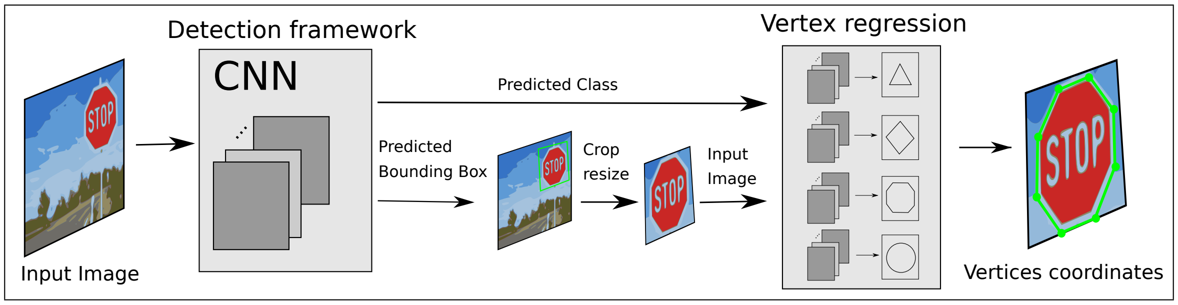

Fig. 1. Our proposed framework for boundary estimation. The detection part is done using *RetinaNet*, the bounding box is cropped from the original image and passed to one of the four regression networks according to the shape of the predicted class.

automatically learn features for object recognition instead of using hand-crafted geometric ones. These architectures are used in many detection frameworks like Faster R-CNN [9], YOLO [10] or RetinaNet [11], they are able to regress a bounding box around each instance of a detected object in a frame. According to the shape of the detected object, a bounding box only gives a rough rectangular approximation of its position in the image. In order to be able to get the 3D pose of an object, a pixel-wise detection is needed. This can be done by Segmentation frameworks like SegNet [5] or Mask R-CNN [1]. However segmentation is computationally complex, time consuming and could fail in some cases where the object to detect is occluded or only appears partly in the image.

An alternative to segmentation would be boundary estimation using template objects [12], this method consists in regressing four vertices (instead of two) of bounding boxes in order to produce parallelogram boxes (Figure 2 (b)), and not rectangular ones. Then find the affine or projection (according to the shape of objects to detect) transformation matrices which transforms regressed bounding box corners to the template corners of the corresponding class. Each template of each class defines its own vertices delimiting the object boundary which relative position to the corners is known. These vertices are then found back in the image using the inverse transformation matrix, which gives the object boundaries back in the image. This method is faster than segmentation, but is only applicable to simple shaped objects and requires a precise database of template objects, moreover, the regression error is propagated through the transformations for the final prediction, finally, the parallelogram-box is supposed to be very accurate to match the model after transformation.

Another approach is based on keypoints detection frameworks such as CornerNet [13] or Stacked Hourglass Networks [14]. They are widely used for human pose estimation and rely on the use of Fully Convolutional Networks (FCNs), using features predicted by the networks as Heatmaps. Coordinates of the keypoints in the image are then deduced from the heatmaps, by selecting features indexes which have the maximal value (*argmax*). The main issue of this approach is that the loss is applied to heatmaps and not directly to coordinates because the error on coordinates can't be backpropagated due to the use of *argmax*. An alternative is to regress directly on coordinates through fully connected (FC) layers, however, it implies a poor spatial generalization because FC are more prone to overfitting [15]. Recently, DSNT layers [16] propose a good compromise between those two approaches, it maps heatmaps to coordinates using differentiable functions with a fully convolutional approach.

The problem with such methods is that they cannot detect circular or elliptical shapes as they are not defined by keypoints. More over segmentation approaches are computationally expensive. Our method propose to estimate both polygonal and non-polygonal traffic signs boundaries in an efficient way.

III. Proposed method

We consider traffic signs (TS) as simple shaped static objects that can be easily detected. This makes them good candidates as landmarks for semantic SLAM. We propose a method to detect their vertices in order to estimate their boundaries. European TS shapes are restricted to four main shapes which are : triangles, rectangles, octagons and ellipses. These shapes can be extended to other regions TS, since our method handles polygonal shapes as well as non-polygonal shapes like ellipses. As shown in Figure 1, our method is divided into two parts, detection part and vertex regression part. The reason we didn't make an end-to-end framework is because of the lack of data for our specific problem. More details will be presented in section IV-A.

For the detection part, we use RetinaNet along with a ResNet50 backbone which is a good compromise between precision and execution speed when compared to other detectors or deeper backbones, as presented in section II. RetinaNet is pretrained on COCO dataset and trained on GTSDB dataset [17]. For the regression part, we compare two methods, one approach we call *direct regression* which consist in regressing directly on vertices coordinates, and the convolutional method which consist in using heatmaps for keypoint detection.

A. Direct regression approach

For the *direct regression* method, we define *four* regression neural networks based on a VGG architecture [6] as detailed in Table I, one for each TS shape (Triangles, Rectangles, Octagons and Ellipses) which defines their respective outputs.

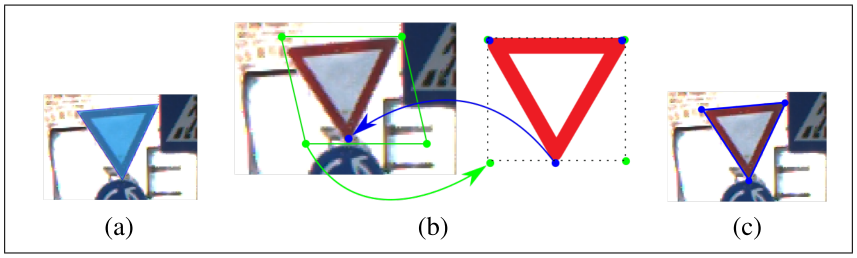

Fig. 2. Different approaches for pixel-wise detection for traffic signs: (a) Segmentation; (b) Boundary estimation using anchors, arrows represents transformations. (c) Our method: Boundary estimation using vertices coordinates. (best seen in color)

TABLE I
REGRESSION NEURAL NETWORKS ARCHITECTURE - *direct regression* METHOD

Layer	Filter size, stride	Output WxHxN
Input	-	96x96x3
Conv	9x9, 1	96x96x64
Conv	7x7, 1	96x96x64
Conv	5x5, 2	48x48x64
BatchNorm	-	-
Conv	3x3, 1	48x48x128
Conv	3x3, 1	48x48x128
Conv	3x3, 2	24x24x128
BatchNorm	-	-
FC	-	256
BatchNorm	-	-
FC	-	256
BatchNorm	-	-
FC	-	Shape dependant

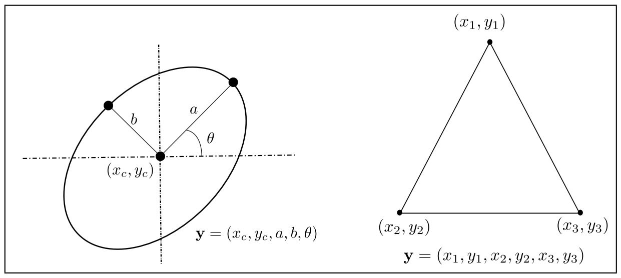
b
a
θ
(x_c, y_c)
$\mathbf{y} = (x_c, y_c, a, b, \theta)$
(x_1, y_1)
(x_2, y_2)
(x_3, y_3)
$\mathbf{y} = (x_1, y_1, x_2, y_2, x_3, y_3)$

Fig. 3. Output vector representation according to the shape : On the left, ellipses are represented by their center coordinates, major axis, minor axis and the angle between the major axis and the horizontal axis. On the right, polygons are represented by their vertices coordinates.

Note that the activated network depends on the predicted class by the detection framework.

As presented in Figure 3, we define the output for polygons as follows:

$$\mathbf{y}_{polygon} = (x_1, y_1, x_2, y_2, \cdots, x_N, y_N)$$

where x_i, y_i are the coordinates of the TS vertices and N is the number of vertices of the considered shape. Ellipses are considered as special cases, and their output is defined as :

$$\mathbf{y}_{ellipse} = (x_c, y_c, a, b, \theta)$$

where x_c, y_c are the coordinates of the ellipse center, a is the major semi-axis, b the minor semi-axis, and $\theta \in [0; \pi[$ the angle between the major semi-axis and the horizontal axis.

A natural loss for polygons vertices coordinates is the Euclidean loss:

$$\mathcal{L}_{euc}(\mathbf{y}, \hat{\mathbf{y}}) = \|\mathbf{y} - \hat{\mathbf{y}}\|_2 \tag{1}$$

where $\mathbf{y}$ is the ground truth coordinates vector and $\hat{\mathbf{y}}$ is the predicted output coordinate vector by the corresponding neural network. For ellipses the Euclidean loss is applicable to the first four parameters (x_c, y_c, a, b) but not for the angle θ. To tackle this problem we split the total loss in two parts, that is to say the total loss is the sum of the first one corresponding to the Euclidean loss for the first four parameters that we call *position* loss and the second one a loss on the angle that we call *orientation* loss :

$$\mathcal{L}_{ellipse} = \alpha_w \mathcal{L}_{position} + \beta_w \mathcal{L}_{orientation} \tag{2}$$

Where α_w and β_w are scaling factors. For θ we consider angles in radian between 0 and π. As angles are defined modulo $[\pi]$ Euclidean loss is not adapted. To tackle this problem, we use the *atan2* function which is differentiable and well defined on the interval $[0; \pi[$. We define our *orientation* loss as following :

$$\mathcal{L}_{orientation}(\theta, \hat{\theta}) = \mid b - a \mid atan2\left(\sin(\theta - \hat{\theta}), \cos(\theta - \hat{\theta})\right)^2 \tag{3}$$

where θ is the ground truth value and $\hat{\theta}$ is the predicted value. There is still a problem when a and b tend toward the same value (special case of the circle when $a = b$) *orientation* loss should be 0 because of the symmetric nature of a circle. This is why we added the term $\mid b - a \mid$ to make angle related loss smaller when the considered ellipse tends toward a circle.

B. Heatmap based regression approach

For the second method, we consider a fully convolutional approach, using heatmap as in Stacked Hourglass Networks [14] for keypoint detection. We design the same way, four regression networks, one per each traffic sign shape. We use ResNet18, removed FC layers and replaced downsampling layers by dilated convolutions to obtain output resolution of 48×48. We also reduce output channels to produce N_S heatmaps. Where N_S is the number of vertices for the shape S which are respectively $3, 4, 8, 5$ for $Triangle, Rectangle, Octagon, Ellipses$. Finally, we apply a DSNT layer as described in [16] to the heatmaps and we get as outputs, coordinates pairs (x, y) for each vertices. Since DSNT layers are differentiable, we can apply the loss directly to the predicted coordinates pairs. The loss is defined as :

$$\mathcal{L}_{DSNT}(\mathbf{y}, \hat{\mathbf{y}}) = \mathcal{L}_{euc}(\mathbf{y}, \hat{\mathbf{y}}) + \mathcal{L}_{\mathcal{D}}(\hat{\mathbf{Z}}, \mathbf{p}) \tag{4}$$

Where $\mathbf{y}$ is the ground truth coordinates vector, $\hat{\mathbf{y}}$ is the predicted output coordinate vector, $\mathcal{L}_{euc}$ is the euclidean loss as in Equation 1, $\hat{\mathbf{Z}}$ is the heatmap output normalized using 2D Softmax, $\mathbf{p}$ is the spherical Gaussian generated using ground truth coordinates and $\mathcal{L}_{\mathcal{D}}$ is the Jensen-Shannon divergence measure. This method works well for polygons since we can define one heatmap per vertex, but there's an issue with ellipses since we have to consider coordinate pairs and not angles and distances. To apply it to ellipses we convert our ellipses representation to the following one:

$$\mathbf{y}_{ellipse} = (x_c, y_c, a\cos\theta, b\sin\theta, -a\sin\theta, b\cos\theta) \quad (5)$$

which corresponds to coordinates of the center and of the points on the semi-axis.

IV. Experiments and results

A. Dataset

Since we consider only European TS we use the German Traffic Sign Detection Benchmark (GTSDB) [17]. This dataset consists of 900 images, 600 for training and 300 for validation/evaluation. Whereas their recognition benchmark GTSRB [18] is much larger $\sim 50k$ images, both of the datasets share the same 40 classes. The idea is to train a detection framework on GTSDB to get a rough detection and classification information (because of the lack of data) and then improve classification, and regress on vertices using networks pretrained on GTSRB.

For the detection part of our framework we tried and compared multiple detection neural networks architectures on the GTSDB Dataset (see Table II). For the regression part, since we didn't find any vertex annotated dataset available, we manually annotated vertices on images taken from the GTSRB dataset, and we used data augmentation on both images and annotated coordinates to produce new ones such as random rotations, translations, scaling and blur. All of the experiments were performed on a computer equipped with a Titan X (Pascal) GPU.

B. Detection framework

We first conducted experiments on RetinaNet [11] with different backbones : ResNet50 , ResNet101 and ResNet152, we also compared it to the Yolo and SSD frameworks. All of the backbones were pretrained on the COCO dataset, and then on the GTSRB dataset with all layers frozen except the last three ones. We then trained the whole framework with each backbone on the GTSDB Dataset which contains 600 images of size 1360×800 on 100 epochs with a batch size of 2 image, and using data augmentation, ADAM optimiser [19] with a starting learning rate of $\alpha = 0.01$, we chose the model who performed best on the evaluation set. We also did the same procedure, on GTSDB dataset with *reduced classes*, we grouped all of the TS in 4 classes according to their shapes : Triangles, Rectangles, Ellipses and Octagons. Finally, we also compared results with a similar detection framework like YOLO and SSD, results are presented in Table II, using the *mAP* (Mean Average Precision) metric.

TABLE II
Detection framework benchmark using different backbones for GTSDB dataset.

Architecture	mAP all classes	mAP reduced classes	time (ms)
RetinaNet-ResNet50	0.67	0.94	58
RetinaNet-ResNet101	**0.675**	**0.943**	89
RetinaNet-ResNet152	0.66	0.94	99
YoloV3-DarkNet53	0.59	0.89	**52**
SSD	0.61	0.89	55

These results shows that the overall mAP increases when we reduce our classes to 4 main classes. This means that our network is able to correctly find good bounding boxes, but often predicts the wrong class for similar traffic signs. This is mainly due to the lack of data in the dataset (600 images).

Bounding boxes are then cropped from the original image, and resized to 96×96 and passed to the regression part of the framework along with the predicted class using the reduced classes.

C. Regression framework

For the regression part, we made our own dataset using a part of the GTSRB dataset on which we manually annotated vertices for polygons and ellipses. 100 images have been annotated per traffic sign shape for the training set, and 20 for the evaluation one, in total 400 images for the training set and 80 for the evaluation set. In addition, we also annotated images from GTSDB to increase the number of images to approximately: 600 ellipses, 400 triangles, 150 rectangles, 120 octagons. We use data augmentation on both images and ground truth vertices coordinates annotations in order to produce new ones to increase the database and prevent over-fitting.

The framework is defined by 4 neural networks with a VGG-like architecture (Table I) which corresponds to the *direct regression* method. The DSNT method consists of 4 neural networks based on ResNet18 architecture where we replaced down-sampling layers by dilated convolutions until we obtain a 48×48 output resolution as described in [16]. All of these networks are trained on our custom annotated dataset during 1000 epochs with a batch size of 16 images of size 96×96, using ADAM with a starting learning rate of $\alpha = 0.01$ and data augmentation. We defined values for Constants $\alpha_w = 4$ and $\beta_w = 1$ defined in Equation 2 by experience, trying multiple values. Results are presented in the Table III, using the *accuracy* metric wich corresponds to the relative mean squared error.

Let's note that it appeared the DSNT provided a small improvement over the direct regression method for polygons, but the direct regression method still gave better results on ellipses. This may be due to lack of data in our dataset, and to the inherent angle and orthogonality information in our vertex representation of ellipses which may be not adapted to describe elliptical shapes. This is a limitation of the DSNT layer, which is applicable only to coordinates pairs which are well-suited for representing polygons, but not for ellipses.

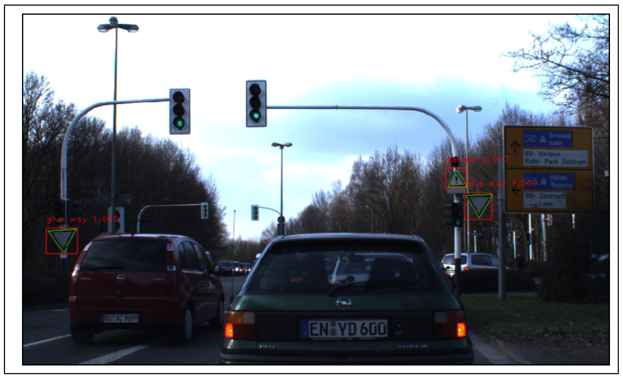

Fig. 4. Resulting image with boundaries drawn on the original image. Detection bounding boxes, classes and scores are represented in red. Theses boxes are cropped, resized and passed to the regression part to find vertices or ellipses and then resized back and drawn to the original image (in green). (best seen in color)

TABLE III
REGRESSION FRAMEWORK BENCHMARK USING *direct regression* AND DSNT METHOD ON OUR VERTEX ANNOTATED GTSRB DATASET.

Method	Accuracy
direct regression-Triangles	0.965
direct regression-Rectangles	0.951
direct regression-Octagons	0.949
***direct regression*-Ellipses**	**0.928**
***DSNT*-Triangles**	**0.975**
***DSNT*-Rectangles**	**0.961**
***DSNT*-Octagons**	**0.951**
DSNT-Ellipses	0.872

Once predictions are found on input images of the detection framework, they are scaled back to the original image size, and boundaries are printed on the original image. See Figure 4 and 5 for some qualitative results. The main boundary regression errors occurs on dark images (e) or on distant objects (g), (i) and (l), this is mainly because we only have few images of this kind annotated in our dataset. We plan to increase our database to improve the detection and boundary estimation.

D. *Comparisons to segmentation methods*

Currently, to the best of our knowledge, there are no available vertex annotated traffic sign datasets, therefore in order to compare our method to segmentation ones, we manually annotated vertices of polygonal traffic signs and elliptical shaped traffic signs in the GTSDB detection dataset. (600 images for the training set and 300 for validation/evaluation sets) From this dataset, on one hand, we deduced the minimal bounding boxes containing our Polygonal and Elliptical traffic signs, we cropped them from the image and resized them to 96×96 and also applied the scaling to vertex coordinates and ellipses corresponding to our traffic signs. We trained our four regression networks using these cropped images and converted coordinates. On the other hand, we also produced binary masks using polygonal and elliptical annotation of traffic signs of our manually annotated dataset and trained a Mask R-CNN instance (pretrained on the COCO dataset) on the same data. Mask R-CNN is trained only on two classes : background versus traffic sign since we're comparing to our regression method. One important fact is that Mask R-CNN will provide a mask that will not necessary be a polygon or an ellipse. Because our goal is to find vertices (or ellipse parametrization) of the considered shape, we still have to approximate it using the provided mask. (see Figure 6). The easiest way to compare both methods is to convert our vertex annotated test set to binary masks that we consider as ground truth. We also do the same for the predicted outputs of our regression networks. We can then compute the IoU between the ground truth masks and the ones predicted by our method or the ones predicted by Mask R-CNN. Results are presented in Table IV. We can see actually that our framework performs slightly better than Mask R-CNN on this dataset, this is probably due to the few amount of data. The main advantages of our method are (1) that it is much faster in execution time, which is important for real-time critical application such as autonomous driving and (2) it can provide very good result with a small amount of data in the learning phase. The execution time is an average prediction time per image (for our framework it corresponds to the entire process: detection + regression time) (3) it can also predict vertices (or ellipse parts) outside of the bounding box (shapes that are only partly in the bounding box). Qualitive results for the comparison are presented on Figure 6 Time is computed only for the full framework and corresponds to the mean number of ms elapsed per image.

TABLE IV
COMPARISON OF OUR FRAMEWORK WITH MASK R-CNN

Method	IoU	Time	FPS
Our framework	**0.823**	**0.071**	**14.08**
Mask R-CNN	0.767	0.275	3.64
Our framework-triangles	0.81	/	/
Mask R-CNN-triangles	0.78	/	/
Our framework-rectangles	0.80	/	/
Mask R-CNN-rectangles	0.77	/	/
Our framework-octagons	0.79	/	/
Mask R-CNN-octagons	0.78	/	/
Our framework-ellipses	0.89	/	/
Mask R-CNN-ellipses	0.73	/	/

V. CONCLUSION

We presented a light framework which is able to detect, recognize and find boundaries of simple semantic objects like traffic signs which can be used as landmarks. Our solution provides very good results compared to state-of-the-art segmentation approach. For this kind of objects, we propose an alternative method to segmentation which is way more computationally complex and time consuming. Moreover, it can be easily adapted to others polygonal or circular shaped objects such as road markings, traffic lights in order to make it more robust to urban environments lacking traffic signs. With such detection over time, the accurate 3D localization of the traffic signs would be possible and would benefit SLAM or urban cartography frameworks. Our vertex

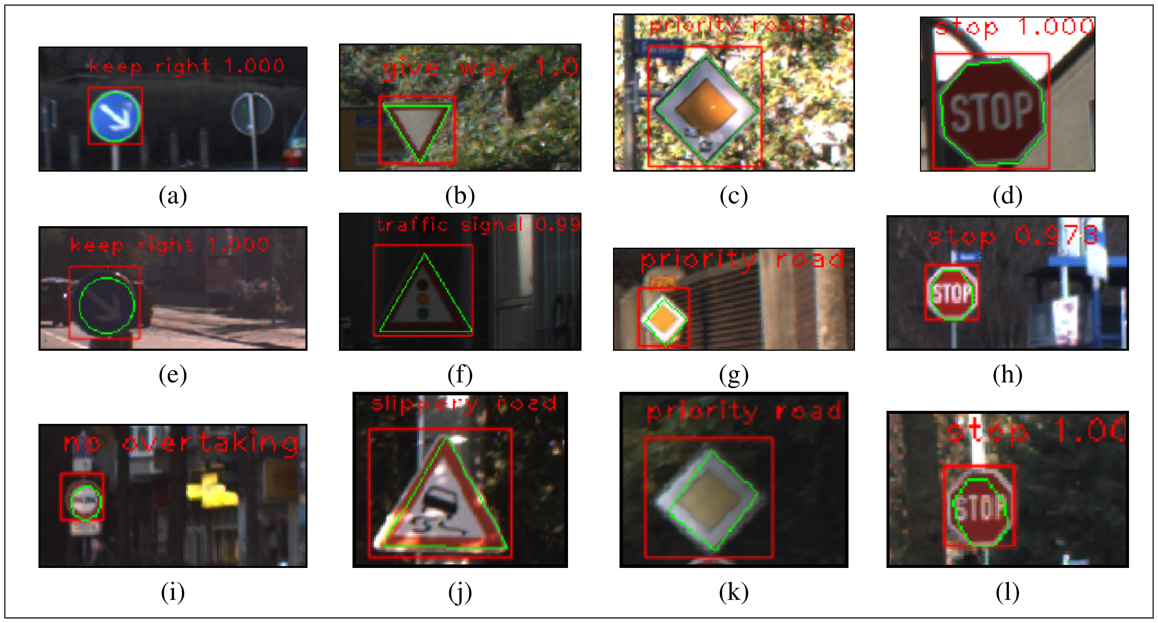

Fig. 5. Image results from the entire framework : In red, bounding boxes predicted by the detection framework and its associated class prediction and score. In green, vertices and ellipses predictions from the regression part. The first row shows some good regressions, the second one ok detections, and the last one, some bad cases.(best seen in color)

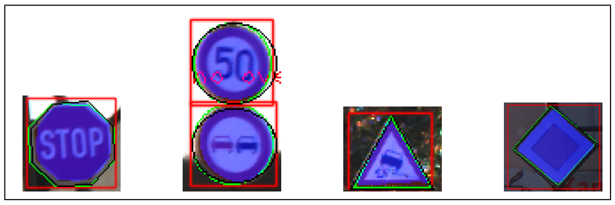

Fig. 6. Comparison with Mask R-CNN. In transparent blue, the mask predicted by Mask R-CNN, in green the shape predicted by our regression networks, and in black the ground truth annotated shape.

annotated traffic signs dataset can be obtained on demand. It is a first step towards an implementation of a *semantic SLAM*. Yet, there are still improvements to be made to this framework. One of the main issue is the lack of a large vertex annotated traffic sign dataset which is required for this framework to work properly. Especially to make it end to end. Next step is to triangulate in 3D the vertices of the detected traffic signs in the video flow in order to get 3D high-level landmarks for the semantic SLAM framework.

References

[1] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969.

[2] C. G. Harris, M. Stephens, *et al.*, "A combined corner and edge detector." in *Alvey vision conference*, vol. 15. Citeseer, 1988, pp. 10–5244.

[3] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov 2004.

[4] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *European conference on computer vision*. Springer, 2006, pp. 404–417.

[5] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[6] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[7] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[8] 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*, 2016, pp. 770–778.

[9] 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*, 2015, pp. 91–99.

[10] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[11] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[12] H. S. Lee and K. Kim, "Simultaneous traffic sign detection and boundary estimation using convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, pp. 1652–1663, 2018.

[13] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 734–750.

[14] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *European Conference on Computer Vision*. Springer, 2016, pp. 483–499.

[15] M. Lin, Q. Chen, and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400*, 2013.

[16] A. Nibali, Z. He, S. Morgan, and L. Prendergast, "Numerical coordinate regression with convolutional neural networks," *arXiv preprint arXiv:1801.07372*, 2018.

[17] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark," in *International Joint Conference on Neural Networks*, 2013.

[18] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608012000457

[19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.