IN4393 Computer Vision - Final Project

Road Sign Detection on Google Streetview

Patrick Brand (4330676) Willem-Jan Meerkerk (4142160)

July 2017

1 Project description

The aim of this project is to develop a system that is able to detect road signs on Google Streetview data. These images consist of 360 degree panoramic images, always corresponding to a known location on earth. There are many applications possible. For example it can be useful for a road maintenance company to have an automatically generated map with all the positions of road signs in a certain region.

We created a live application where a user is able to navigate over Google Maps, with a corresponding Google Streetview panorama to explore locations. The web application acquires the current image from the Google Streetview API and performs sign detection on this image. The detected road signs are marked by a boundingbox and the classificatin result of each boundingbox is shown in the live application. See Figure 1 for a screenshot of our final application.

The challenge in this Computer Vision project was two-fold: (1) find the location of road signs in an arbitrary Google Streetview image, and (2) classify the found road signs to a known set of possible road signs. In this report we will explain the algorithms we have used to solve the problem, mention our own implementations, describe the experiments we performed to evaluate the system and show the results of these experiments.

2 Algorithms

To localize the road signs, we use an approach where we extract image patches using color segmentation on the input image (ROI finding). In this project we initially limited ourselves to circular road signs, to make the problem doable in the scope of this course. The next step is to use circle detection to find whether or not there is a road sign in an image patch, and if so, to determine the precise location of the road sign in that image patch. The final step is to use a classifier to predict which particular road sign is found.

2.1 ROI finding

The first step of the algorithm is to determine Regions of Interest (ROI) that could contain traffic signs. These ROIs are found by converting the input image to the HSV domain and then applying several thresholds. An example of the binary segmentation can be seen in the middle picture of Figure 2. This idea is similar to the segmentation methods that Takada and Katto use in their work [11] and the work of Brkic was used as a general overview for color based segmentation [2]. After the segmentation the patches are labeled and a bounding box around each label is specified which acts as mask for the Regions of Interest. An example can be seen in the right-most picture of Figure 2.

2.2 Sign localization

Several methods were tried to detect whether or not a road sign is present in the extracted ROIs. Some of them are: RANSAC circle and ellipse fitting, morphological watershed, adaptive threshold, circle or ellipse detection via Hough transform, and approaches with convex hull. The best approach is to use the Hough transform [1] for detecting circles, because that method provides a circle center and radius for each road sign. Based on these parameters the circular road sign is cropped to remove the background.

Given an arbitrary image patch, it is first transformed to the HSV domain and a threshold on the H and S channels is applied. The patch is segmented independently for red parts and blue parts, since this improved the separation of road signs from other objects, for example a blue road sign on a red background. The thresholds are detected experimentally on positive and negative examples of image patches, see Table 1. Next, some morphological operations are performed to improve the binary red and binary blue images. Then the boundaries of the binary objects are extracted, which are used as input image for the Hough transform.

The image patch is transformed to the Hough domain, from which a maximum of 5 circles are extracted, by conditioning on for instance minimum distance between peaks and radius. The detected circles are found using the `hough_circle` and `hough_circle_peaks` methods from skikit-image [12]. Image patches where the Hough circle detection returns no results are rejected. It is possible that multiple circles with a slightly different center or radius are returned for the same road sign. Therefore, a clustering algorithm was implemented, which divides the detected circles into clusters, separated by a minimum distance of 20 pixels. For each cluster, the circle with the maximum radius is selected as the final result of the circle detection.

In the end, the image patches are cropped to the individual road signs by a mask based on the detected center coordinates and the radius of a circle. In Figure 3 of the appendix all the steps, from a single patch to the cropped image, of the sign localization algorithm are summarized.

2.3 Sign classification

Our idea was to detect traffic signs by using Histogram of Oriented Gradients (HOG) as features for training of a Linear Support Vector Machine (SVM) classifier, in a similar fashion to the Dalal and Triggs person detector [3]. Since traffic signs contain very distinct high contrasts between shapes and figures, we expect the HOG descriptor to capture these elements that define a certain traffic sign very well. The work of Dalal and Triggs showed good results of a Linear SVM classifier trained on HOG features, which was therefore chosen for this project. Besides SVM a Random Forest classifier was also trained on the HOG descriptor for comparison. The classifiers from the scikit-learn library were used for this project [8].

Initially, clean images were gathered for all the supported Dutch road signs. The images are downloaded from the internet [13], cropped and resized to the same size of 75x75 pixels. HOG features were generated for these images, using 9 orientations, 8 pixels per cell and 3 cells per block. See Figure 4 for a full representation of the clean dataset.

In order to train a classifier that was more robust, training data that contained samples of each class in various environments and of various quality was needed in order to better represent real world examples. A combination of subsets of the Belgian Traffic Sign Database (BelgiumTS) [6] and the German Traffic Sign Recognition Benchmark (GTSRB) [10] were used for this purpose.

3 Own implementation

For this project, we selected Python as programming language. Most of the implementation is done from scratch, starting with a simple web server using Flask [9]. This web server serves the main web page, retrieves images from the Google Streetview API and returns results for the road sign detection. The functionality of the web page (in HTML and Javascript) is programmed by ourselves. As we have described in the previous section, all steps from a Streetview image to final classification result are written by ourselves, with methods for finding image patches, finding circles, circle clustering, training of classifiers and evaluation. We did use image processing methods from the scikit-image library [14]. On top of this, we have selected to implement the algorithm for generating HOG features ourselves, and to compare it with the implementation of scikit-image. The scikit-learn library provided the classifiers that were trained on the HOG descriptors [8].

3.1 HOG features

The concept of the Histogram of Oriented Gradients descriptor as described by Dala and Triggs [3] consists of four major parts. At first the first order gradient of the image should be calculated in both

the x and y direction. We implemented a simplified approximation of this step by convolving the image with a $\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$ filter and with the transpose of this filter as described by S. Mallick [5]. Based on these two gradients, the angle and magnitude of each gradient can be calculated for every pixel location of the image.

This gradient image is then divided in (NxM) shaped cells and a weighted histogram is calculated for all gradient values per cell. The histogram generally consists of 9 bins representing angles from 0 to 180 degrees. We implemented linear interpolation of the weights, as each angle's contribution to the two surrounding bins is based on a linear interpolation of the corresponding magnitude between the two bins. Therefore an angle of 75 degrees with a magnitude of 1 contributes for $\frac{1}{4}$ to the bin with angles in the range from 60 to 80 degrees and for $\frac{3}{4}$ to the bin with angles in the range from 80 to 100 degrees [7].

At this point we have one histogram per cell. In order to attain a better invariance to illumination, shadowing and edge contrast, the histograms are normalized [4]. This normalization step is performed per group of cells called blocks. Blocks generally consist of (PxQ) cells and, unlike cells, are defined in a sliding window kind of fashion. This means that there is an overlap between blocks. Because of this step, most cells are present in the descriptor multiple times, although they are normalized by a different set of surrounding cells. We implemented 'L2' normalization of a vector that consists of a concatenation of all histograms in a block and we perform this operation on all blocks.

The final step consists of merging all normalized histograms into one big feature vector, which is called the HOG descriptor.

4 Experiments

The experiments are separated in two parts, namely: Sign localization, where the extraction of road sign patches from images are tested, and sign classification, where the recognition of the traffic sign is tested. We created our own dataset, since no dataset with annotated road sign regions on images of Google Streetview images was available. For the classification part there was no dataset of Dutch road signs, so a dataset was created from subsets of both a dataset of German road signs (GTSRB) [10] and a dataset of Belgian road signs (BelgiumTS) [6], where the subsets were selected based on the amount of provided samples and correspondence to their Dutch counterparts.

4.1 Sign localization

In order to perform an empirical test of the sign localization, we manually recorded 120 images from Google Streetview using our web application, resulting in images like the left-most image in Figure 2. Each image contains at least one circular road sign and the images are taken at different locations and under different conditions. This data was obtained very early in our project, not knowing any outcome of our algorithms. Therefore we can assume that our sampling data is unbiased and independent. Our first experiment was to test the full process of sign localization (ROI finding & circle detection) on these images. An image is localized correctly if all the circular road signs are detected (i.e. a bounding box is shown).

Our second experiment was to test the circle detection. Therefore, we automatically saved all the individual image patches of the 120 images (result of ROI finding), resulting in 662 files. For example, the right-most image in Figure 2 resulted in 2 image patches. We divided the image patches in a positive set containing one or more circular road signs) of 121 samples and a negative set (containing no road signs) of 541 samples. Then we applied our circle detection like in Figure 3 and checked for each image patch if the correct number of road signs are detected. In all these experiments, the number of true positives (TP), false positives (FP) and false negatives (FN) are recorded.

4.2 Sign classification

We trained three different classifiers on two different sets of features. A Linear SVC classifier and a Random Forest classifier is trained on the HOG descriptor from SKImage and one SVC classifier is trained on our own implementation of the HOG descriptor. The HOG descriptors are calculated by taking cells that consist of 8x8 pixels, the histograms consist of 9 bins and the blocks of 3x3 cells. The

normalization technique of the SKImage HOG descriptor is L2-Hys (Lowe-styleclipped L2 norm) [3]. The normalization technique of our own implementation of the HOG descriptor is L2 normalization.

A stratified 5-fold cross validation is performed on each classifier, where the following measurements are taken: confusion matrix, accuracy, per class precision, and per class recall for each fold. Since our dataset is highly unbalanced, we can't assume that each class is represented equally well by a normal 5-fold cross validation. Therefore we ensure that the ratio between the amount of train and test samples is as equal as possible for all the classes by performing stratification.

5 Results

5.1 Sign localization

The results of the empirical experiment are summarized in Table 2. For each experiment, we calculated the error rate (number of incorrect detections divided by the total number of files). We see that the sign localization has a large error of 0.5667 on the full Streetview images. The circle detection has an error of 0.3719 on the positive examples, and an error of 0.0481 on the negative example. These measures were calculated strictly. Many Streetview images contain multiple road signs. When not all road signs are correctly detected (i.e. 2 of the 3 road signs), the image is considered as incorrect. Furthermore, the number of false positives (FP) and false negatives (FN) is relatively high. These values were also calculated strictly. It often occurred that a bounding box was placed in the neighbourhood of the road sign, but not exactly at the correct location. These cases count as a FP and a FN.

5.2 Sign classification

The results, shown in Table 3 and Figure 4 of the appendix, are summarized by averaging over all 5 folds, the results of the individual folds are available on our Github page. Pictures of the road signs that belong to each class name can be viewed in Figure 4 (a) of the appendix.

5.2.1 SKImage HOG descriptors LinearSVC classifier

The results Table 3 show that the precision values of this classifier are generally between 0.88 and 0.97, with the exception of classes **A01-80** (0.82) and **F05** (0.75). The recall values of this classifier are generally between 0.90 and 1.00, with several exceptions which are all classes that belong to the speed limit signs. Classes **A01-80** and class **F05** stand out the most, where **A01-80** has both a relatively low precision and recall and **F05** has the lowest precision, but the highest recall. From confusion matrix shown at Figure 4 (a), it is shown that class **F05** also contains very few samples. The overall accuracy score of the classifier is 0.92.

5.2.2 SKImage HOG descriptors RandomForest classifier

The results Table 3 show that the Random Forest classifier has in general very high precision, with many values above 0.95. The classifier scores the lowest precision on class **A01-80** (0.77). The recall scores of the classifier are mostly above 0.90 as well, where most values below 0.90 are again classes that are speed signs. The overall accuracy score of the classifier is 0.94.

5.2.3 Our HOG descriptors LinearSVC classifier

The results Table 3 in the appendix shows that the Linear SVC classifier trained on our HOG descriptors has high precision scores as well (generally above 0.91 and many above 0.92). The classifier has the worst precision for **A01-80** and **F05**, both with a score of 0.87. The recall scores are generally above 0.92, with **A01-80** as the exception which has a recall score of 0.86. Most of the recall scores are above 0.97. The overall accuracy score of the classifer is 0.95.

6 Conclusion

The results of the color-based patch extraction from images are far from optimal. There are quite a lot of False Negatives compared to the amount of True Positives, which shows that the patch extraction is nearly fifty-fifty. One reason for the bad results is that we didn't spend much time in fine tuning the HSV-thresholds. Besides that, extracting patches purely based on color is complex, due to the variation in for instance lighting conditions, and since similar colors are widely present in other objects than road signs. Another approach would be to train the classifier to learn to detect the shape of traffic sign and distinguish between traffic sign and no traffic sign. A sliding window approach over the HOG descriptor of the raw input image could then be used in order to detect the position of possible signs. A downside is that all color information is then discarded, so segmentation is mostly based on shape.

The circle detection definitely improves the results of the ROI finding based on color segmentation. Most of the negative image patches are rejected. However, it is still hard to detect all the circles correctly using the Hough transform, since this depends on a good patch with edges and fine tuned Hough radii. It often occurred that the detected circle is displaced by a few pixels, which negatively affected the cropping and classification of the road sign. A more general approach for road sign localization, not depending on circles, would be better.

Both the SVC classifiers and the Random Forest classifier seem to perform quite well when trained on the HOG descriptors. The confusion matrices show that class **F05** contained very few samples, far less than the amount of samples of the other classes, which could explain why both SVC classifiers scored low on precision for that class . The confusion matrices also show that speed signs (A01-30 to A01-120) are mostly predicted as another speed sign by all classifiers, when classified wrongly. This indicates that the used HOG descriptors did not capture the details of the distinct numbers on the signs well enough. This should definitely be looked into in future work. It is possible that a different set of parameters for the HOG descriptor could give a better result, however other methods for distinguishing the numbers should be considered as well.

Random Forest seems to have a higher overall precision when compared to the SVC classifier that was trained on the same set of descriptors, which means that the Random Forest classifier is more 'exact' in its classification. The Recall scores of both classifiers is quite similar. It is noticeable that the random Random Forest classifier is the only tested classifier that had a very high recall for class **F05** where the other classifiers seemed to struggle with that class.

Training on our own HOG descriptors resulted into better overall classifier than when the same classifier is trained with the HOG descriptors extracted by SKImage. Both precision and recall scores are in general higher than their counterparts of the SKImage based classifier. It is notable that the classifier trained on our features seem suffer less from the problem of wrongly classifying speed signs as other speed signs than the other classifiers, although the same conclusion holds here as well.

The detection of road signs on Streetview images in the webapp is inaccurate and unstable to say the least. If a road sign gets detected correctly, a small movement of the camera will often result in a different (wrong) prediction of that same sign. The dataset that was used to train the classifier is unbalanced and does not contain pictures acquired from Google Streetview. Conditions that appear on Google Streetview images like stitching artifacts are therefore not represented in our dataset. Hence, the classifier could not be trained on such conditions. For future work we would like to create a dataset that has a better balance and represents the 'real world conditions' of Streetview better. Besides that we think that classifying on a set of frames of a sign would increase the accuracy as well. This could be simulated by locking on to patch with a sign and extracting frames after small changes of camera position and on different levels of zoom. Another welcome addition would be to append color information to the hog descriptor, so that the wrongly classifying **C01** as **E01** will become less likely.

For future work, it might be interesting to train a neural network to detect and/or classify road signs based on both color and shape information, since they are proven to be fairly good at object detection.

Evaluation

We truly enjoyed working on this project, however we feel that the task of detecting and recognizing road signs in Google Streetview images is a very complex task. The amount of time and effort that has been put into making the segmentation and classification parts to produce the results that are shown in this report, combined with the effort of manually creating a dataset for segmentation, is tremendous. In hindsight we feel that limiting the scope of our project to either the road sign detection part or the road sign recognition part would have been a better choice, given the amount of time that was available for this project. Nevertheless, we learned a lot from this project, and even though the results are not that great, we believe that this proof of concept demonstrates that Computer Vision techniques have the potential to tackle the problem of recognizing traffic signs in Google Streetview data.

References

[1] Dana H Ballard. "Generalizing the Hough transform to detect arbitrary shapes". In: *Pattern recognition* 13.2 (1981), pp. 111–122.

[2] Karla Brkic. "An overview of traffic sign detection methods". In: *Department of Electronics, Microelectronics, Computer and Intelligent Systems Faculty of Electrical Engineering and Computing Unska* 3 (2010), p. 10000.

[3] Navneet Dalal and Bill Triggs. "Histograms of oriented gradients for human detection". In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on.* Vol. 1. IEEE. 2005, pp. 886–893.

[4] *Histogram of Oriented Gradients.* `http://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html`. Accessed: 2017-07-01.

[5] Satya Mallick. *Histogram of Oriented Gradients.* `http://www.learnopencv.com/histogram-of-oriented-gradients/`. Accessed: 2017-07-01.

[6] Markus Mathias et al. "Traffic sign recognition—How far are we from the solution?" In: *Neural Networks (IJCNN), The 2013 International Joint Conference on.* IEEE. 2013, pp. 1–8.

[7] Chris McCormick. *HOG Person Detector Tutorial.* `http://mccormickml.com/2013/05/09/hog-person-detector-tutorial/`. Accessed: 2017-07-01.

[8] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[9] Armin Ronacher. *Flask, a micro webdevelopment framework for Python.* URL: `http://flask.pocoo.org/`.

[10] J. Stallkamp et al. "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition". In: *Neural Networks* 0 (2012), pp. -. ISSN: 0893-6080. DOI: `10.1016/j.neunet.2012.02.016`. URL: `http://www.sciencedirect.com/science/article/pii/S0893608012000457`.

[11] Ryoki Takada and Jiro Katto. "Traffic sign recognition by distorted template matching". In: *Consumer Electronics (GCCE), 2014 IEEE 3rd Global Conference on.* IEEE. 2014, pp. 416–418.

[12] Scikit-image development team. *Circular and Elliptical Hough Transforms.* URL: `http://scikit-image.org/docs/dev/auto_examples/edges/plot_circular_elliptical_hough_transform.html`.

[13] Verkeersbordenoverzicht.nl. *Overzicht van verkeersborden in Nederland.* URL: `http://www.verkeersbordenoverzicht.nl/`.

[14] Stéfan van der Walt et al. "scikit-image: image processing in Python". In: *PeerJ* 2 (June 2014), e453. ISSN: 2167-8359. DOI: `10.7717/peerj.453`. URL: `http://dx.doi.org/10.7717/peerj.453`.

Appendix

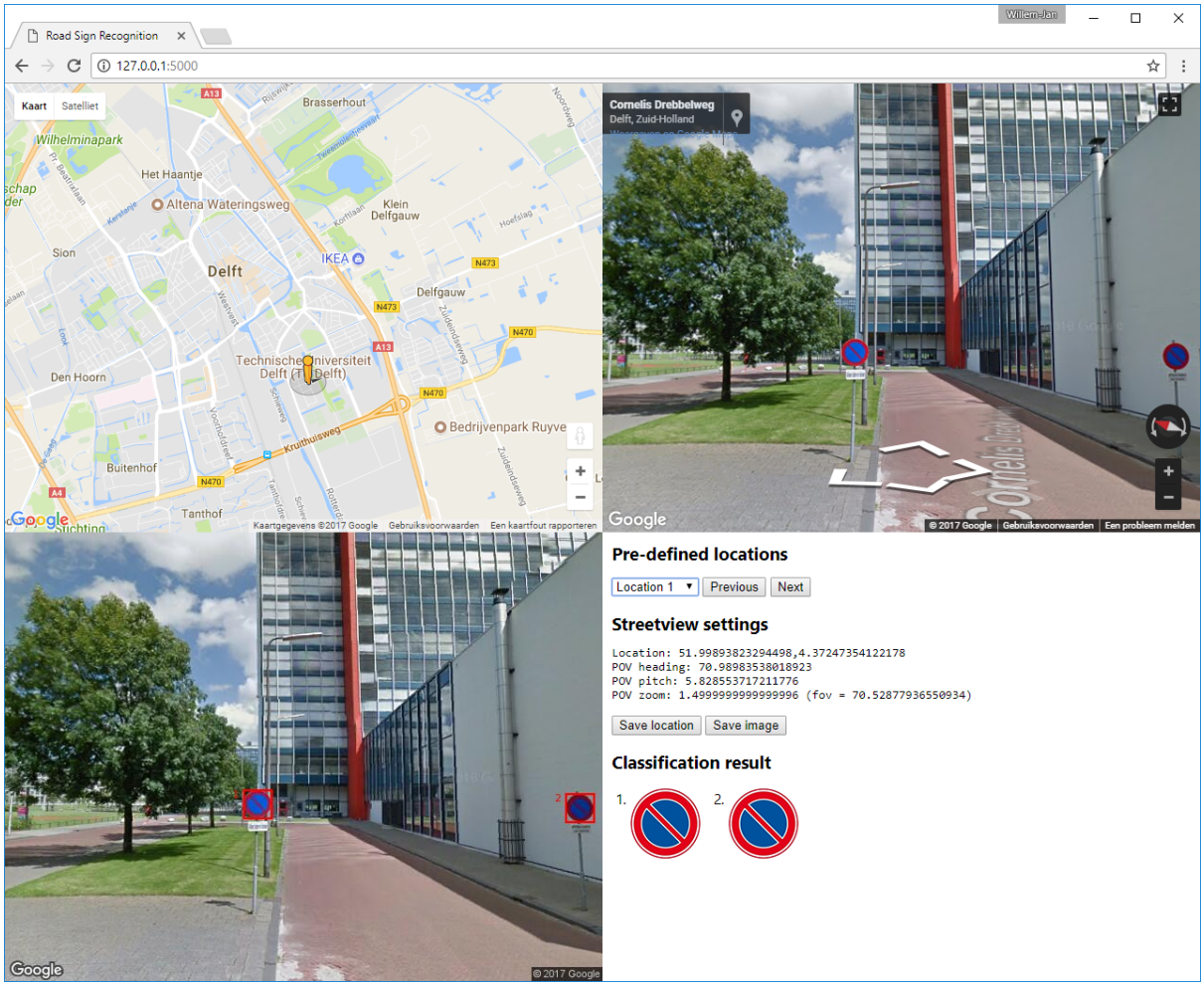


Figure 1: Screenshot of our final application. First row: live Google Maps (left) and corresponding Streetview panorama (right). Second row: resulting image after road sign detection (left) and parameters + classification result (right).

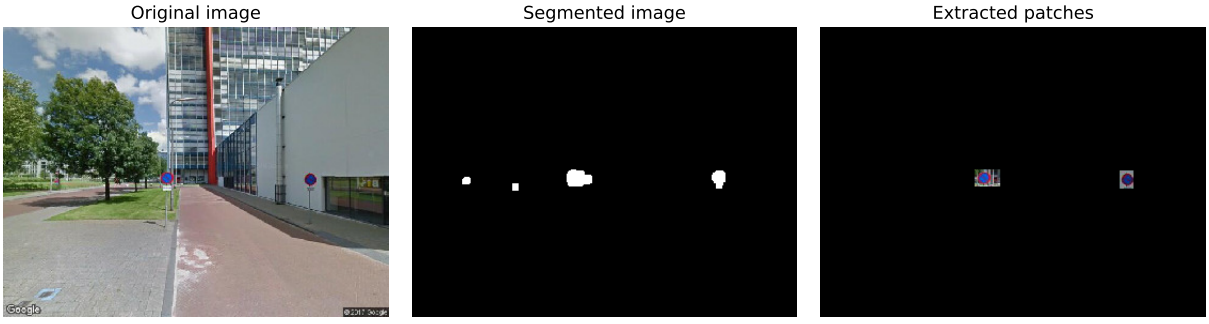


Figure 2: Step-by-step visualization of segmentation, resulting in extracted image patches.

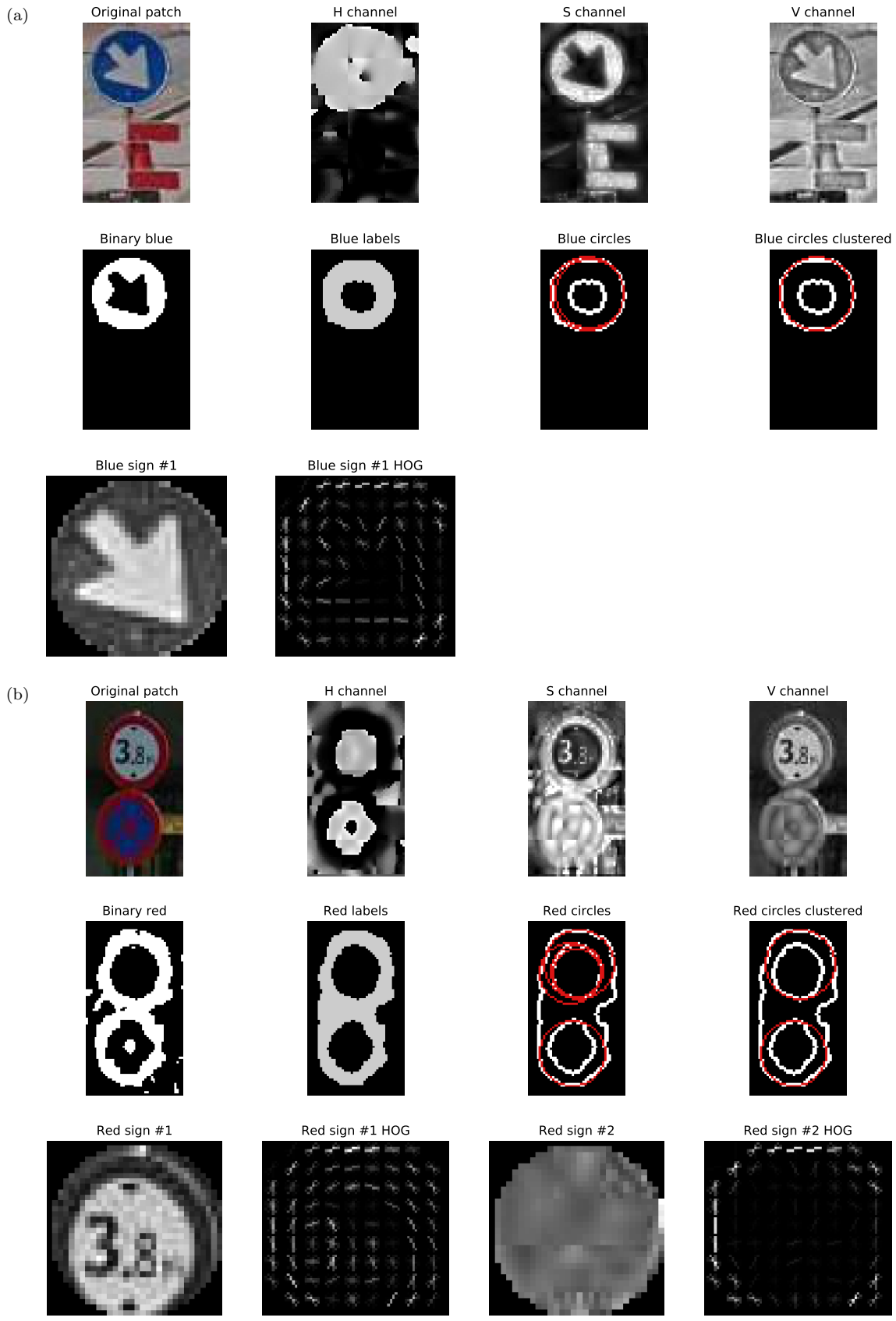


Figure 3: Step-by-step visualization of circle detection on (a) one blue road sign in a single image patch, and (b) two red road signs in a single image patch.

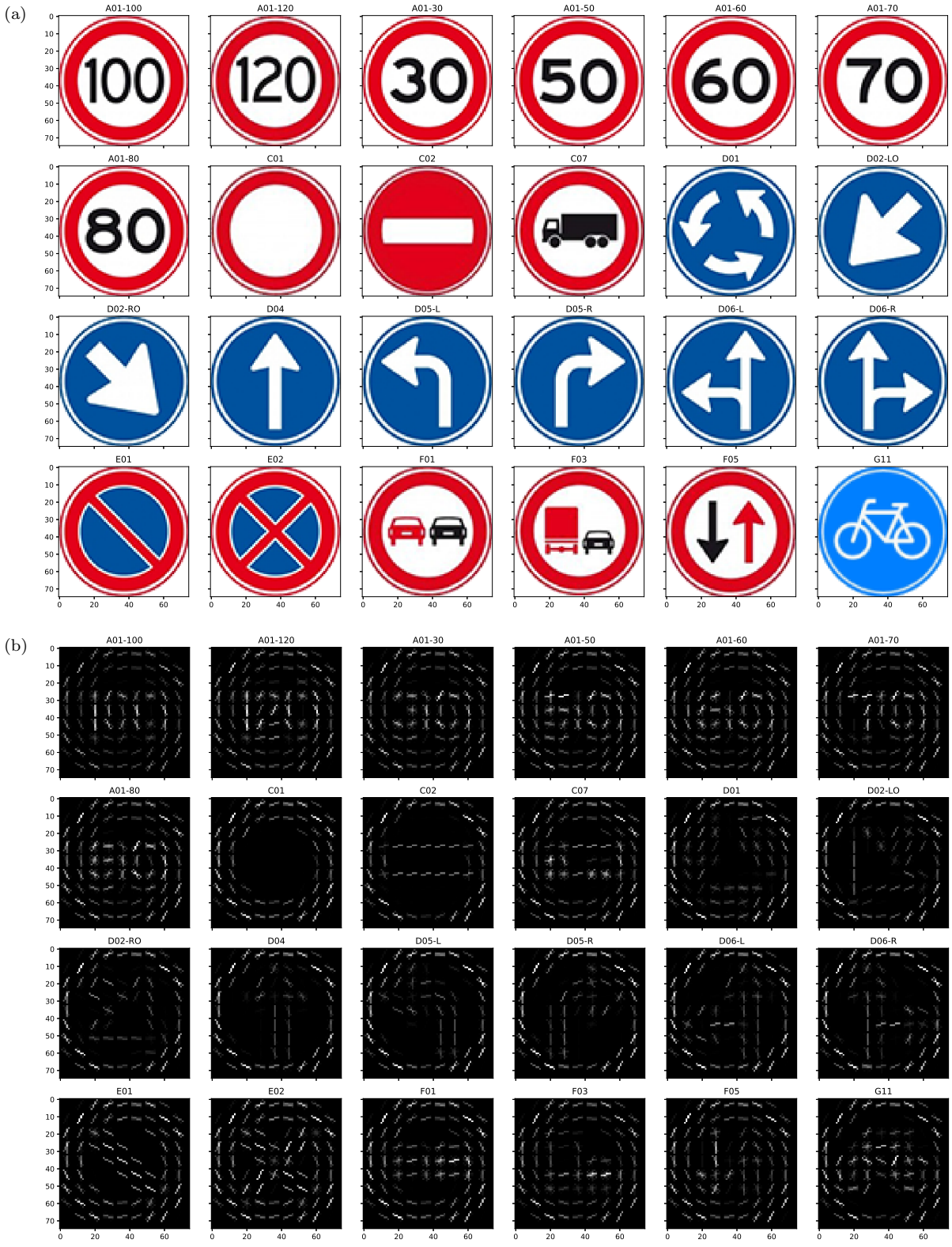


Figure 4: Visualization of our road sign dataset, with (a) clean images and (b) their corresponding HOG features.

	Hue	**Saturation**
Binary red	$H < 0.05$ or $H > 0.80$	$S > 0.30$
Binary blue	$H > 0.55$ and $H < 0.65$	$S > 0.40$

Table 1: Binary thresholds for detecting red and blue segments in image patches. The H and S values are normalized between 0 and 1.

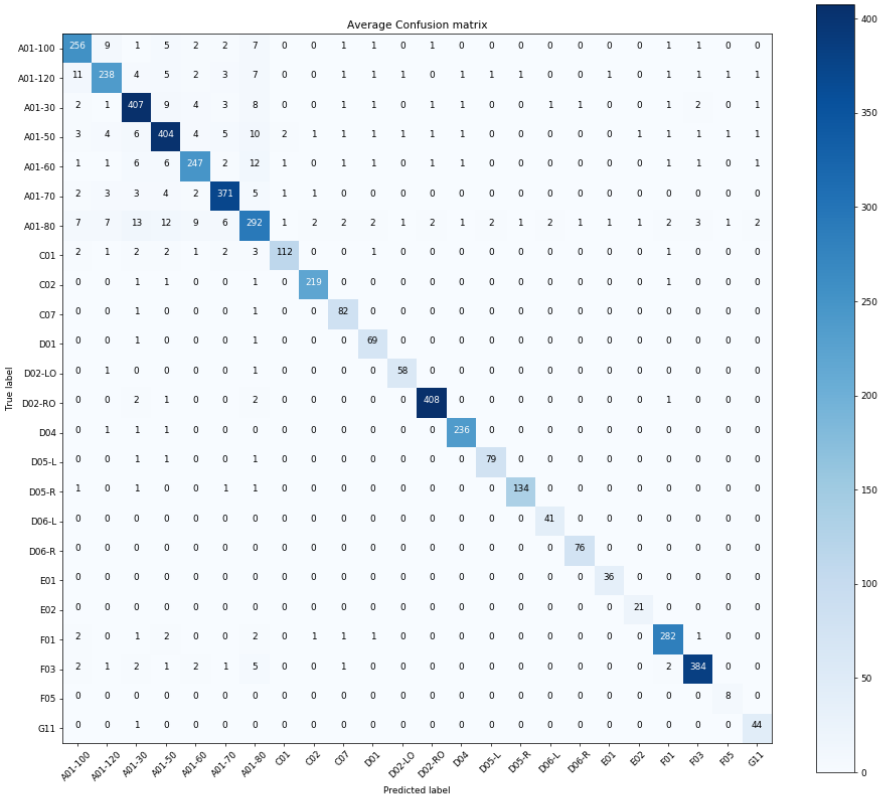
	I	$P_{positive}$	$P_{negative}$
#files	120	121	541
#correct	52	76	515
error	0.5667	0.3719	0.0481
#signs	171	129	-
TP	97	82	-
FP	36	17	26
FN	74	47	-

Table 2: Performance of road sign localization on streetview images and streetview patches. I is the set of 120 streetview images (640x480px), containing at least one road sign. $P_{positive}$ is the set of 121 extracted image patches containing one or more signs, and $P_{negative}$ is the set of 541 extracted image patches containing no road sign.

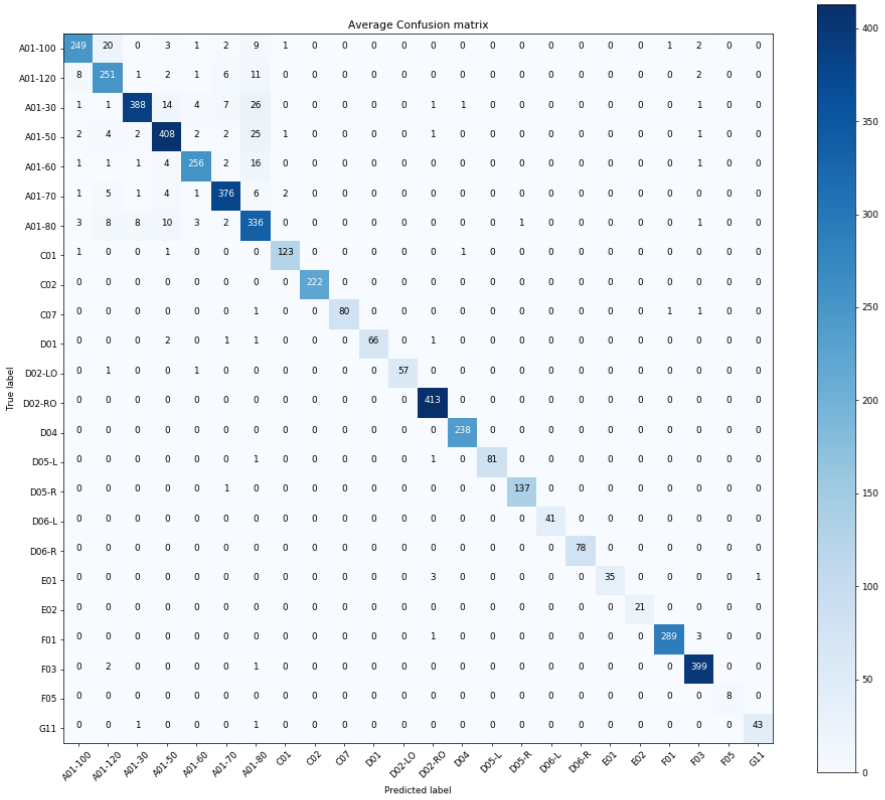
	SKImage SVM		**SKImage RF**		**Our SVM**	
	Precision	*Recall*	*Precision*	*Recall*	*Precision*	*Recall*
A01-100	0.88	0.89	0.94	0.87	0.92	0.93
A01-120	0.89	0.84	0.86	0.89	0.91	0.92
A01-30	0.90	0.92	0.86	0.87	0.94	0.93
A01-50	0.89	0.90	0.91	0.91	0.92	0.92
A01-60	0.90	0.88	0.94	0.91	0.94	0.93
A01-70	0.93	0.94	0.94	0.95	0.96	0.97
A01-80	0.82	0.79	0.77	0.90	0.87	0.86
C01	0.95	0.89	0.96	0.98	0.98	0.98
C02	0.98	0.98	1.00	1.00	0.99	0.99
C07	0.90	0.97	1.00	0.95	0.97	0.99
D01	0.88	0.95	1.00	0.91	0.95	0.98
D02-LO	0.95	0.97	1.00	0.95	0.98	0.99
D02-RO	0.98	0.98	0.98	1.00	0.99	0.99
D04	0.98	0.98	0.99	0.99	0.99	0.99
D05-L	0.95	0.94	1.00	0.96	0.98	0.98
D05-R	0.98	0.97	0.98	0.99	0.99	0.99
D06-L	0.92	0.98	1.00	0.98	0.94	0.97
D06-R	0.96	0.98	1.00	0.99	0.98	0.99
E01	0.91	0.92	1.00	0.89	0.96	0.95
E02	0.87	0.94	0.98	0.97	0.92	0.97
F01	0.96	0.96	0.99	0.98	0.98	0.98
F03	0.97	0.95	0.97	0.99	0.98	0.98
F05	0.75	1.00	1.00	1.00	0.87	1.00
G11	0.89	0.97	0.99	0.94	0.97	1.00
$ClassifierAccuracy$	0.92		0.94		0.95	

Table 3: Results of the Linear SVC and Random Forest classifiers that were trained on SKImage HOG descriptors and the results of the Linear SVC classifier that was trained on our HOG descriptor.

(a)


Average Confusion matrix
True label
Predicted label
A01-100 A01-120 A01-30 A01-50 A01-60 A01-70 A01-80 C01 C02 C07 D01 D02-LO D02-RO D04 D05-L D05-R D06-L D06-R E01 E02 F01 F03 F05 G11
400
350
300
250
200
150
100
50
0

(b)


Average Confusion matrix
True label
Predicted label
A01-100 A01-120 A01-30 A01-50 A01-60 A01-70 A01-80 C01 C02 C07 D01 D02-LO D02-RO D04 D05-L D05-R D06-L D06-R E01 E02 F01 F03 F05 G11
400
350
300
250
200
150
100
50
0

(c)

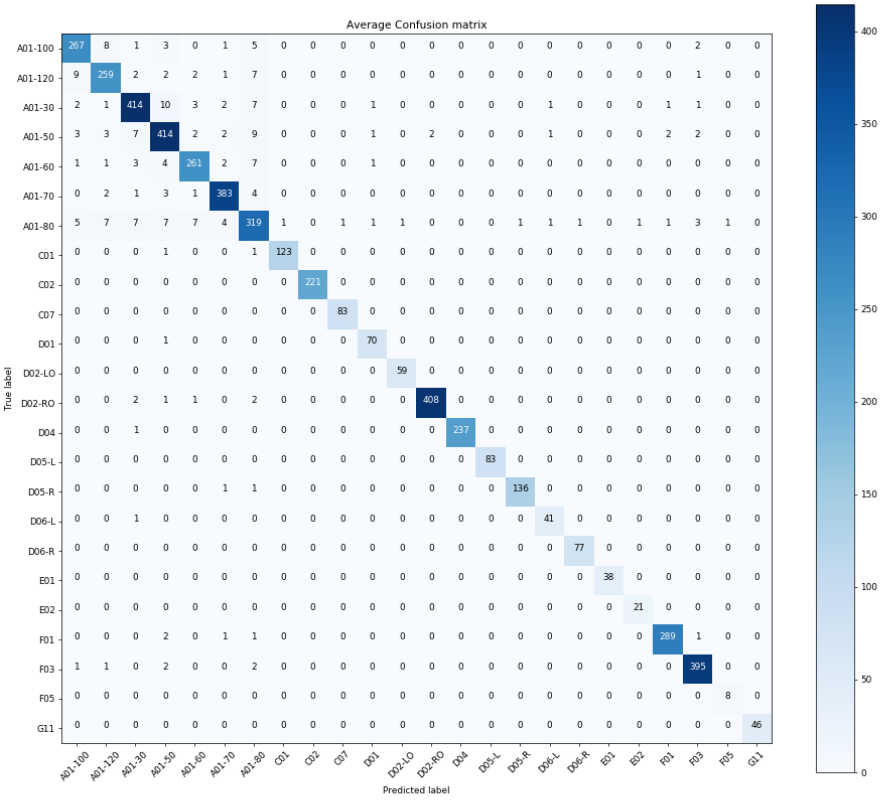


Figure 4: Confusion matrices of classifier evaluation of Linear SVC classifier trained on SKImage HOG descriptors (a), Random Forest classifier trained on SKImage HOG descriptors (b) and Linear SVC classifier trained on our own HOG descriptors (c).