Towards Traffic Sign Recognition on an Embedded System

Goedemé, Toon

January 28, 2008

De Nayer, Wetensch. en Kunst, J. De Nayerlaan 5, 2860 St.-Katelijne-Waver
ESAT-PSI-VISCIS, K. U. Leuven, Kasteelpark Arenberg 10, 3001 Heverlee
Toon.Goedeme@denayer.wenk.be

Abstract

In this paper, we present a method for fast and robust object recognition, especially developed for implementation on an embedded platform. As an example, the method is applied to traffic sign recognition from a forward-looking camera in a car. To facilitate and optimise the implementation of this algorithm on an embedded platform containing parallel hardware, we developed a voting scheme for constellations of visual words, i.e. clustered local features (SURF in this case). On top of easy implementation and robust and fast performance, even with large databases, an extra advantage is that this method can handle multiple identical visual features in one model.
Keywords: *traffic sign recognition, local features, SURF, embedded.*

1 Introduction

The automatic recognition of road signs is one of the key upcoming technologies making cars safer to drive with. An on-board camera installed in the car observes the road ahead. Intelligent computer vision algorithms are being developed that enable the detection and recognition of various objects in these images: traffic lane markings, pedestrians, obstacles, ... Here, we focus on the recognition of traffic signs. Based on recognition results, certain alerts can be sent to the driver, e.g. a warning if the current speed of the car is higher than the speed limit declared in the traffic sign.

Because that the appearance of a certain traffic sign is fixed (even described by law), the detection is quite a bit easier than e.g. the detection of pedestrians. Nevertheless, in real-life experiments substantial appearance variation is measured, mainly due to different light conditions, viewpoint changes, ageing of the traffic sign, deformations and even vandalism. We can conclude that a robust method is needed.

The remainder of this text is organised as follows. Section 2 gives an overview of relevant related work. In section 3, our algorithm is descibed. Some *real life* experiments are presented in section 4. The paper ends with a conclusion in section 5.

2 Related Work

Real-time road sign recognition has been a research topic for many years. This problem is often addressed in a two-stage procedure involving detection and classification. In contrast, our solution is an all-in-one operation which more likely leads to a faster algorithm. An other differece with related work in the field is that our solution does not rely on template matching [13], colour [22], the detection of geometrical basis shapes [6], or canny edges [14]. Moreover, our solution is not limited to certain traffic sign shapes [2]. We use clusters of local image features (SURF) to robustly describe the appearance of the traffic sign.

A few years ago, a major revolution in the object recognition field was the appearance of the idea of local image features [19, 10]. Indeed, looking at local parts instead of the entire pattern to be recognised has the inherent advantage of robustness to partial occlusions. In both template and query image, local regions are extracted around interest points, each described by a descriptor vector for comparison. The development of robust local feature descriptors, like e.g. Mindru's generalised colour moment based ones [12], added robustness to illumination and changes in viewpoint.

Many researchers proposed algorithms for local region matching. The differences between approaches lie in the way in which interest points, local image regions, and descriptor vectors are extracted. An early example is the work of Schmid and Mohr [15], where geometric invariance was still under image rotations only. Scaling was handled by using circular regions of several sizes. Lowe *et al.* [10] extended these ideas to real scale-invariance. More general affine invariance has been achieved in the work of Baumberg [3], that uses an iterative scheme and the combination of multiple scales, and in the more direct, constructive methods of Tuytelaars & Van Gool [19, 18], Matas *et al.* [11], and Mikolajczyk & Schmid [16]. Although these methods are capable to find very qualitative correspondences, most of them are too slow for use in a real-time application as the one we envision here. Moreover, none of these methods are especially suited for the implementation on an embedded computing system, where both memory and computing power must be as low as possible to ensure reliable operation at the lowest cost possible.

The classic recognition scheme with local features, presented in [10, 18], and used in many applications such as in our previous work on robot navigation [8, 7], is based on finding one-on-one matches. Between the query image and a model image of the object to be recognised, bijective matches are found. For each local feature of the one image, the most similar feature in the other is selected.

This scheme contains a fundamental drawback, namely its disability to detect matches when multiple identical features are present in an image. In that case, no guarantee can be given that the most similar feature is the correct correspondence. Such pattern repetitions are quite common in the real world, though, especially in man-made environments. To reduce the number of incorrect matches due to this phenomenon, in classic matching techniques a criterium is used sich as comparing the distance to the most and the second most similar feature [10]. Of course, this practice throws away a lot of good matches in the presence of pattern repetitions.

In this paper, we present a possible solution to this problem by making use of the *visual word* concept. Visual words are introduced [17, 9, 21] in the context of object classification. Local features are grouped into a large number of clusters with those with similar descriptors assigned into the same cluster. By treating each cluster as a *visual word* that represents the specic local pattern shared by the keypoints in that cluster, we have a visual word vocabulary describing all kinds of such local image patterns. With its local features mapped into visual words, an image can be represented as a *bag of visual words*, as a vector containing the (weighted) count of each visual word in that image, which is used as feature vector in the classication task.

In contrast to the in categorisation often used bag-of-words concept, in this paper we present the *constellation-of-words* model. The main difference is that not only the presence of a number of visual words is tested, but also their relative positions.

3 Algorithm

Figure 1 gives an overview of the algorithm. It consists of two phases, namely the model construction phase (upper row) and the matching phase (bottom row).

First, in a model photograph (a), local features are extracted (b). Then, a vocabulary of visual words is formed by clustering these features based on their descriptor. The corresponding visual words on the image (c) are used

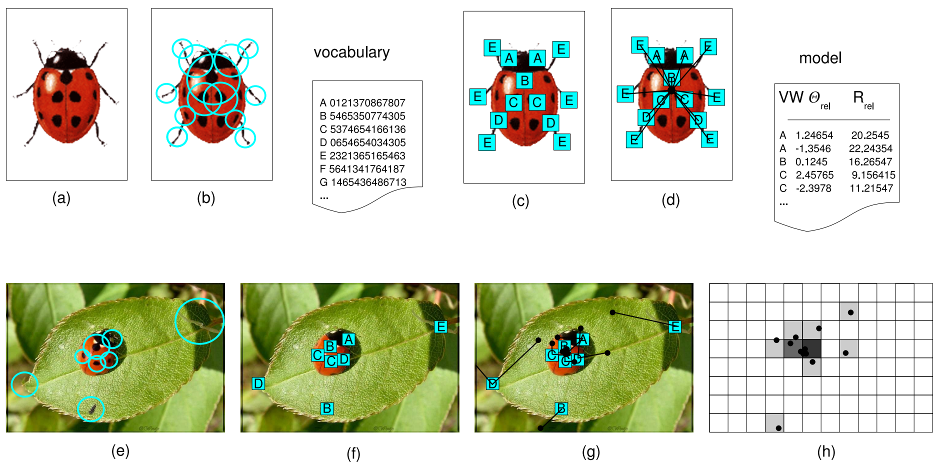

Figure 1: Overview of the algorithm. Top row (model building): (*a*) model photo, (*b*) extracted local features, (*c*) features expressed as visual words from the vocabulary, (*d*) model description with relative anchor positions for each visual word. Bottom row (matching): (*e*) query image with extracted features, (*f*) visual words from the vocabulary, (*g*) anchor position voting based on relative anchor position, (*h*) Hough voting space.

to form the model description. The relative location of the image centre (the *anchor*) is stored for each visual word instance (*d*).

The bottom row depicts the matching procedure. In a query image, local features are extracted (*e*). Matching with the vocabulary yields a set of visual words (*f*). For each visual word in the model description, a vote is cast at the relative location of the anchor location (*g*). The location of the object can be found based on these votes as local maxima in a voting Hough space (*h*). Each of the following subsections describes one step of this algorithm in detail.

3.1 Local Feature Extraction

We chose to use SURF as local feature detector, instead of the often used SIFT detector. SURF [4, 5] is developed to be substantially faster, but at least as performant as SIFT.

3.1.1 Interest Point Detector

In contrast to SIFT [10], which approximates Laplacian of Gaussian (LoG) with Difference of Gaussians (DoG), SURF approximates second order Gaus-

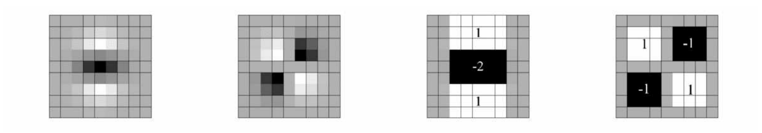

Figure 2: Left: two filters based on Gaussian derivatives. Right: their approximation using box filters.

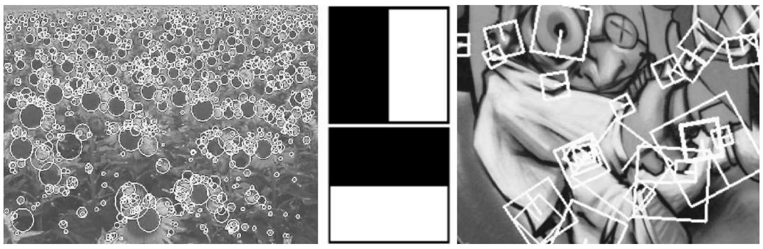

Figure 3: Middle: Haar wavelets. Left and right: examples of extracted SURF features.

sian derivatives with box filters, see figure 2. Image convolutions with these box filters can be computed rapidly by using integral images as defined in [20]. Interest points are localised in scale and image space by applying a non-maximum suppression in a 3×3 neighbourhood. Finally, the found maxima of the determinant of the approximated Hessian matrix are interpolated in scale and image space.

3.1.2 Descriptor

In a first step, SURF constructs a circular region around the detected interest points in order to assign a unique orientation to the former and thus gain invariance to image rotations. The orientation is computed using Haar wavelet responses in both x and y direction as shown in the middle of figure 3. The Haar wavelets can be easily computed via integral images, similar to the Gaussian second order approximated box filters. Once the Haar wavelet responses are computed, they are weighted with a Gaussian centred at the interest points. In a next step the dominant orientation is estimated by summing the horizontal and vertical wavelet responses within a rotating wedge, covering an angle of $\frac{\pi}{3}$ in the wavelet response space. The resulting maximum is then chosen to describe the orientation of the interest point descriptor. In a second step, the SURF descriptors are constructed by extracting square regions around the interest points. These are oriented in the directions assigned in the previous step. Some example windows are shown on the right hand side of figure 3. The windows are split up in 4×4 sub-regions in order to retain some spatial information. In each sub-region, Haar wavelets are extracted at regularly spaced sample points. In order to

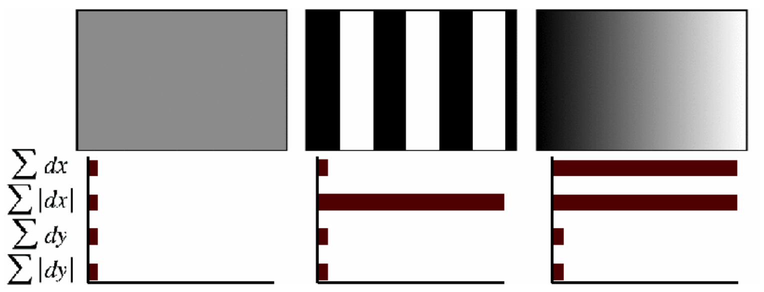

Figure 4: Illustrating the SURF descriptor.

increase robustness to geometric deformations and localisation errors, the responses of the Haar wavelets are weighted with a Gaussian, centred at the interest point. Finally, the wavelet responses in horizontal d_x and vertical directions d_y are summed up over each sub-region. Furthermore, the absolute values $|d_x|$ and $|d_y|$ are summed in order to obtain information about the polarity of the image intensity changes. The resulting descriptor vector for all 4×4 sub-regions is of length 64. See figure 4 for an illustration of the SURF descriptor for three different image intensity patterns.

More details about SURF can be found in [4] and [5].

3.1.3 Visual Words

As explained before, the next step is forming a vocabulary of visual words. This is accomplished by clustering a big set of extracted SURF features. It is important to build this vocabulary using a large number of features, in order to be representative for all images to be processed.

The clustering itself is easily carried out with the k-means algorithm. Distances between features are computed as the Euclidean distance between the corresponding SURF descriptors. Keep in mind that this model-building phase can be processed off-line, the real-time behaviour is only needed in the matching step.

In the fictive ladybug example of figure 1, each visual word is symbolicly presented as a letter. It can be seen that the vocabulary exists of a file linking each visual word symbol with a mean descriptor vector of the corresponding cluster.

3.2 Model Construction

All features found on a model image are matched with the visual word vocabulary, as shown in fig. 1 (c). In addition to the popular bag-of-words models,

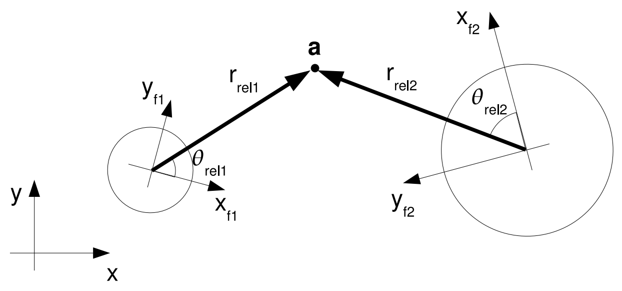

Figure 5: The position of the anchor point is stored in the model as polar coordinates relative to the visual word scale and orientation.

which consist of a set of visual words, we add the relative constellation of all visual words to the model description.

Each line in the model desription file consists of the symbolic name of a visual word, and the relative coordinates (r_{rel}, θ_{rel}) to the anchor point of the model item. As anchor point, we chose for instance the centre of the model picture. These coordinates are expressed as polar coordinates, relative to the individual axis frame of the visual word. Indeed, each visual word in the model photograph has a scale and an orientation because it is extracted as a SURF feature. Figure 5 illustrates this. The resulting model is a very compact description of the appearance of the model photo. Many of these models, based on the same visual word vocabulary, can be saved in a compact database. In our traffic sign recognition application, we build a database of all different traffic signs to be recognised.

3.3 Matching

This part of the algorithm is time-critical. We are spending lots of efforts in speeding up the matching procedure, in order to be able to implement it on an embedded system.

The first operation carried out on incoming images is extracting SURF features, exactly as described in section 3.1. After local feature extraction, matching is performed with the visual words in the vocabulary. We used Mount's ANN (Approximate Nearest Neighbour) [1] algorithm for this, which is very performant. As seen in fig. 1 (*f*), some of the visual words of the object are recognised, amidst other visual words.

3.3.1 Anchor Location Voting

Because each SURF feature has a certain scale and rotation, we can reconstruct the anchor pixel location by using the feature-relative polar coordinates of the object anchor. For each instance in the object model description, this yields a vote for a certain anchor location. In figure 1 (g), this is depicted by the black lines ending with a black dot at the computed anchor location.

Ideally, all these locations would coincide at the correct object centre. Unfortunately, this is not the case due to mismatches and noise. Moreover, if there are two identical visual words in the model description of an object (as is the case in the ladybug example for words A, C and D), each detected visual word of that kind in the query image will cast to different anchor location votes, of which only one can be correct.

3.3.2 Object Detection

For all different models in the database, anchor location votes can be quickly computed. Next task is to decide where a certain object is detected. Because a certain object can be present more than once in the query image, it is clear that a simple average of the anchor position votes is not a sufficient technique, even if robust estimators like RANSAC are used to eliminate outliers. Therefore, we construct a *Hough space*, a matrix which is initiated at zero and incremented at each anchor location vote, fig. 1 (h). The local maxima of the resulting Hough matrix are computed and interpreted as detected object positions.

4 Experiments

For preliminary experiments, we implemented this algorithm using Octave and an executable of the SURF extractor. Figure 7 shows some typical results of different phases of the algorithm. The test images were acquired by taking 640×480 digital photographs at random natural road scenes.

In fig. 6, a number of model photographs are shown. Each of such images, having a resolution of about 100×100 pixels, yielded a thourough description of the traffic sign design in a model description containing on the average 52 features, what boils down to a model file size of only 3.2 KB.

Fig. 7 shows a typical output during the detection stage. In a query image, local features are extracted. These are matched with the visual word vocabulary. Then, for each traffic sign model, for the matching visual words the

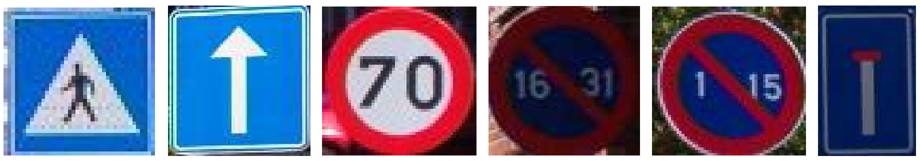

Figure 6: Some model photos from the traffic sign library.

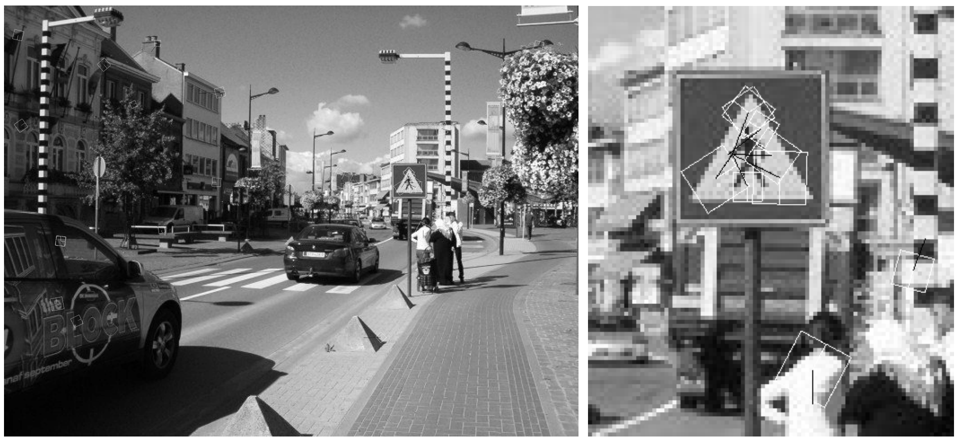

Figure 7: Typical experimental results. In a query image, all matching visual words are shown for the *pedestrian crossing* sign (white squares). From each visual word, the anchor location is computed (black line pointing form visual word centre to anchor). The zoom on the right shows clearly the recognised sign: many anchor lines coincide.

relative anchor position is computed. This stage is visualised in the figure. As can be noticed, in the centre of the traffic sign, many anchor votes are coinciding.

The traffic signs were detected by finding local maxima in the Hough space, for this example visualised in fig. 8. We performed experiments on 35 query images and were able to detect 82% of the trained types. Detection failures were mostly due to the fact that the signs were too far away and hence too small, and severe occlusions by other objects.

5 Conclusions and Future Work

In this paper, we presented an algorithm for object detection based on the concept of visual word costellation voting. The preliminary experiments proved the performance of this approach. The method has the advantages that it is computing power and memory efficient and that it can handle pattern repetitions in the models.

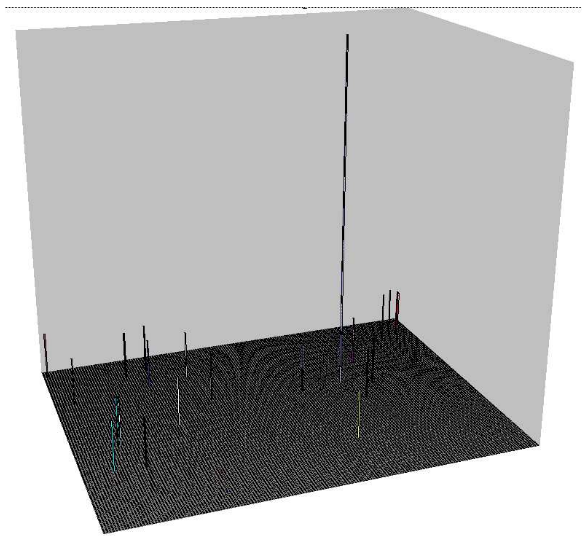

Figure 8: Hough space for the achor positions of the example in fig. 7

We applied this method succesfully on automatic traffic sign recognition.

As told before, our aim in this work is an embedded implementation of this algorithm. The Octave implementation presented here is only a first step towards that. But we believe the proposed approach has a lot of advantages. The SURF extraction phase can mostly be migrated to a parallel hardware implementation on FPGA. Visual word matching is sped up using the ANN-libraries, making use of Kd-trees. Of course a large part of the memory is used by the (mostly sparse) hough space. A better description of the voting space will lead to a great memory improvement of the algorithm.

References

[1] S. Arya, D. Mount, N. Netanyahu, R. Silverman, and A. Wu, "An optimal algorithm for approximate nearest neighbor searching", J. of the ACM, vol. 45, pp. 891-923, 1998, http://www.cs.umd.edu/∼mount/ANN/.

[2] Roberto Ballerini, Luigi Cinque , Luca Lombardi and Roberto Marmo, "Rectangular Traffic Sign Recognition," Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 3617/2005, Image Analysis and Processing ICIAP, pp. 1101-1108 2005.

[3] A. Baumberg, "Reliable feature matching across widely separated views," Computer Vision and Pattern Recognition, Hilton Head, South Carolina, pp. 774-781, 2000.

[4] H. Bay and T. Tuytelaars and L. Van Gool, "Speeded Up Robust Features", ECCV, 2006.

[5] Beat Fasel and Luc Van Gool, "Interactive Museum Guide: Accurate Retrieval of Object Descriptions" in "Adaptive Multimedia Retrieval: User, Context, and Feedback", Lecture Notes in Computer Science, Springer, volume 4398, 2007.

[6] Garcia-Garrido, M.A.; Sotelo, M.A.; Martin-Gorostiza, E., "Fast traffic sign detection and recognition under changing lighting conditions," Intelligent Transportation Systems Conference 2006, pp. 811 - 816, 2006.

[7] T. Goedemé, M. Nuttin, T. Tuytelaars and L. Van Gool, "Omnidirectional Vision based Topological Navigation," International Journal of Computer Vision and International Journal of Robotics Research, Special Issue: Joint Issue of IJCV and IJRR on Vision and Robotics, 2006.

[8] T. Goedemé, T. Tuytelaars, G. Vanacker, M. Nuttin and L. Van Gool, "Feature Based Omnidirectional Sparse Visual Path Following," IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2005, pp. 1003-1008, Edmonton, 2005.

[9] F.-F. Li and P. Perona. "A bayesian hierarchical model for learning natural scene categories." In Proc. of the 2005 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition, pages 524531, 2005.

[10] D. Lowe, "Object Recognition from Local Scale-Invariant Features," International Conference on Computer Vision, pp. 1150-1157, 1999.

[11] J. Matas, O. Chum, M. Urban and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," British Machine Vision Conference, Cardiff, Wales, pp. 384-396, 2002.

[12] F. Mindru, T. Moons, and L. Van Gool, "Recognizing color patters irrespective of viewpoint and illumination," Computer Vision and Pattern Recognition, vol. 1, pp. 368-373, 1999.

[13] A. Rosenfeld and A.C. Kak, "Digital Picture Processing," Computer Science and Applied Mathematics, Academic Press, New York, 1976.

[14] Sandoval, H.; Hattori, T.; Kitagawa, S.; Chigusa, Y., "Angle-dependent edge detection for traffic signs recognition," Proceedings of the IEEE Intelligent Vehicles Symposium 2000, pp. 308-313, 2000.

[15] C. Schmid, R. Mohr, C. Bauckhage, "Local Grey-value Invariants for Image Retrieval," International Journal on Pattern Analysis an Machine Intelligence, Vol. 19, no. 5, pp. 872-877, 1997.

[16] K. Mikolajczyk and C. Schmid, ”An affine invariant interest point detector,” ECCV, vol. 1, 128–142, 2002.

[17] J. Sivic and A. Zisserman. ”Video google: A text retrieval approach to object matching in videos.” In Proc. of 9th IEEE Intl Conf. on Computer Vision, Vol. 2, 2003.

[18] T. Tuytelaars and L. Van Gool, ”Wide baseline stereo based on local, affinely invariant regions,” British Machine Vision Conference, Bristol, UK, pp. 412-422, 2000.

[19] T. Tuytelaars, L. Van Gool, L. D’haene, and R. Koch, ”Matching of Affinely Invariant Regions for Visual Servoing,” Intl. Conf. on Robotics and Automation, pp. 1601-1606, 1999.

[20] Viola, P. and Jones, M., ”Rapid object detection using a boosted cascade of simple features,” Computer Vision and Pattern Recognition, 2001.

[21] J. Zhang, M. Marszalek, S. Lazebnik, and C. Schmid. ”Local features and kernels for classication of texture and object categories: An in-depth study.” In Technical report, INRIA, 2005.

[22] Shuangdong Zhu and Lanlan Liu, ”Traffic Sign Recognition based on Color Standardization, IEEE International Conference on Information Acquisition 2006, pp. 951-955, Veihai, China, 2006.