A REAL WORLD DETECTION SYSTEM
Combining Color, Shape and Appearance to Enable Real-time Road Sign Detection

Peng Wang[1], Jianmin Li[1] and Bo Zhang[1]
[1]*Department of Computer Science and Technology, Tsinghua University, Beijing, P.R.China*
wang-p06@mails.tsinghua.edu.cn,{lijianmin,dcszb}@tsinghua.edu.cn

Keywords: Object Detection, Road Sign, Haar-wavelet, Cascade Detector, HSV Color Space, Contour, RANSAC

Abstract: Although specific object detection has undergone great advances in recent years, its application to critical real-time circumstances like those in automated vehicle controlling is still limited, especially when facing strict speed and precision requirements. This paper uses a hybrid of various computer vision techniques including color space analysis, Haar-wavelet cascade detector, contour analysis and RANSAC shape-fitting, to achieve a real-time detection speed while maintain a reasonable precision and false-alarm level. The result is a practical system that out-performed most rivals in an automated vehicle contest and an indication of feasible CV application to speed critical areas.

1 INTRODUCTION

1.1 Motivation

The real time road sign detection system introduced by this paper was motivated as a subsystem of an automatic driving system for vehicles. Road sign detection is a crucial component of automated vehicle controlling, especially when one wants the car to have certain 'intelligence' to sense its environment like a human being does and to decide its movement based on the encircling environment other than external helps like GPS. Three challenges arise in designing such a real time object detection application which cannot be fulfilled together by a single existing computer vision algorithm. They are the strict requirement of speed, of precision and the diversity of target objects.

When building a subsystem used by a vehicle, in order for the car to travel at an acceptable velocity, the detection system must run at a speed much higher than typical CV algorithms can usually achieve, while at the same time subject to a limited onboard computation power.

Also, in order that the car can rely on the output of the detection subsystem to make subsequent decisions, such a subsystem must strive to miss road signs as few as possible while give false detections as few as possible.

Current object detection algorithms mostly deal with one specific target object category. Algorithms like the cascade classifier (Viola 2004) described in Section 2 were designed for a single kind of visual objects such as human faces, pedestrians or cars. In the contrast, our targets, the road signs, are a conceptual category which, in fact, consists of very different visual objects. Our detection task hence lies in between specific object detection and general object detection. Though we can principally train a detector for each kind of road signs and use them simultaneously to tackle the multi-category problem, the speed and computational resource constraints do not allow for such an amount of detectors.

1.2 Outline

To tackle all these three challenges, we designed a combination of various computer vision techniques to utilize color, shape and appearance information all together. The design principle of our system is to rely on a stable detection algorithm, which may be time consuming, to act as the main detector, while utilize various kinds of pre-processing and post-processing stages to shrink the area of regions performed on by this main detector as much as possible. The reduction in regions of interests compensates the slow speed of the main detector, and thus makes the realtime-ness of the whole system possible. For orthogonality, these pre and

post processing stages should exploit information different from that used by the main detector.

We designed a pipeline architecture to combine these pre and post processing stages and the main detector. The pipeline of our road sign detection system consists of 4 stages, as shown in Figure 1. At the first stage, color space analysis is used to spot approximate regions of interest, regions having the potential to contain road signs. After that, the potential regions are tested against contour analysis to shrink the number and size of candidate regions. At the third stage, a Haar-wavelet cascade detector plays the major role of detecting road signs. At the final stage, these detected signs (with their bounding boxes) are checked by a RANSAC shape-fitting algorithm as a post-validator.

Section 3 describes each stage in details. Section 4 describes some additional optimization techniques to further boost the system's speed. Experiment results and performance in a real world contest are shown in Section 5.

Figure 1: The 4-stage pipeline

2 RELATED WORK

As a particular instance of specific object detection in the field of computer vision, road sign (or traffic sign) detection has been studied for years in the literature. Existing methods can be approximately classified into two categories: color-based and shape-based. All methods in (Broggi 2007, Escalera 1997, Escalera 2003) use thresholds within certain color-space to pick up the pixels that comprise the target object, and utilize image processing techniques to derive the final bounding boxes from these identified pixels and regions. The differences between them lie in the different color space they use and various post-processing procedures.

The biggest problem of color-based methods is the poor distinguishing power and weak robustness of sole color information, and hence the difficulty in distinguishing road signs from noises like cars and buildings in a similar color, and in adjusting the parameters to different weathers and lighting conditions.

Among the shape-based methods, most of them borrow ideas and techniques from the field of human face detection. Researches in recent years showed that compositions of simple features like Haar-wavelet (Papageorgiou 2000) turned out to have great advantage in speed while do not suffer much from precision drop. Above basic Haar-wavelets, Bahlmann (2005) proposed colored Haar-wavelets, which split a color image into 7 channels and extract Haar features in each channel. Lienhart (2002) introduces diagonal Haar-wavelets along with their fast computing method, giving Haar-wavelets a bigger expressive power regarding to diagonal changes. Barczak (2005) discusses the possibility that a Haar-wavelet at arbitrary orientation can be expressed as a linear combination of two basic, that is to say, orthogonal Haar-wavelets, which they called 'pair of equivalent features'.

Given that all of these methods use Haar-wavelets as the building blocks, the classifiers they utilize are different. Authors in (Viola 2004) introduced the canonical Cascade Classifier, which has multiple stages with each stage containing a weaker classifier. Each stage classifier rules out some false samples subject to the constraint that it must let most true samples pass. A sample that makes it through all stages will finally be considered positive. The beauty of this architecture is that it can exclude a large volume of false samples at earlier stages, which consist of the cheapest and fastest classifiers. Such a feature makes cascade classifier particularly suitable for situations where there are much more false samples than true samples, of which object detection is one.

Besides that, in attacking road sign detection problem, Soetedjo (2005) proposes 6 local features to form an ellipse detector, which is used then to detect road signs. Paclik (2000) uses Laplace Kernel methods as classifiers. Bahlamann (2005) uses LDA classifiers.

The problem of shape-based methods is that they are all aimed for one specific kind of visual object. For example, the Viola cascade classifier introduced in (Viola 2004) was originally designed to detect human faces. It can be trained to detect other kinds of objects, but not a mix of them. If we force the training samples to contain various kinds of objects, the output will be a detector poor in both hit rate and false-alarm rate. If we train a detector for each kind of objects, the computational resources required during detecting will be overwhelming, as mentioned in Subsection 1.1.

3 PIPELINE

3.1 Color Space Analysis

Many color spaces can be used for color space analysis, and among the most widely used are RGB, HSV and Lab. Our system selects HSV color space because it is the most coherent with the intuition of human conception. HSV has three channels, namely Hue, Saturation and Value. The total space spanned by these three channels can be presented by a HSV Cone, as shown in Figure 2.

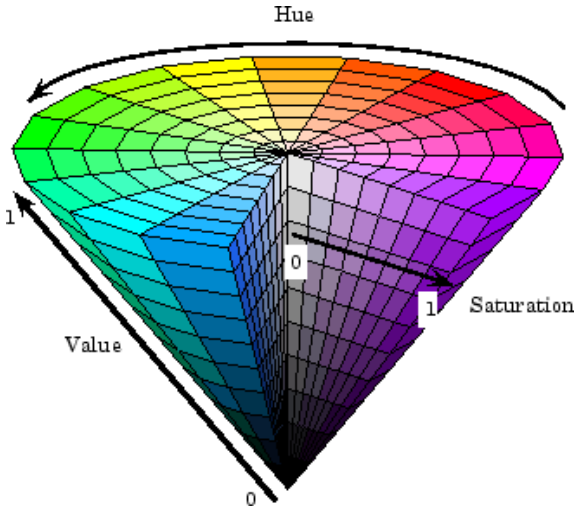

Figure 2: The HSV Color Space. Copied From *Matlab Documentation*

Converting formula from RGB to HSV is expressed as Equation (3-1).

$$\begin{cases} V = M \\ S = \dfrac{M - m}{M} \\ H = \begin{cases} 60h, & h \geq 0 \\ 60h + 360, & h < 0 \end{cases} \end{cases} \tag{3-1}$$

where

$$M = \max(R, G, B)$$

$$m = \min(R, G, B)$$

$$\mathrm{h} = \begin{cases} \dfrac{G - B}{M - m}, & M = R \\ 2 + \dfrac{B - R}{M - m}, & M = G \\ 4 + \dfrac{R - G}{M - m}, & M = B \end{cases}$$

After converting an image into HSV color space, we should inspect whether our target objects, namely road signs, are contained in a distinctive region in that space. Rendering an image captured from the real world road in each of the H, S, V channel, we found that road signs stand out prominently in H and S channels, but not so much in V channel. This indicates that H and S channels are more useful in distinguishing our target objects. Via setting upper and lower thresholds on these two channels, we can approximately pick out pixels that belong to a road sign. These pixels will connect with each other to form irregular regions, which, after certain image processing techniques like Erode, Dilate, Open and Close, can be used to calculate bounding boxes that most tightly contain them. These bounding boxes form the regions of interests (ROIs) for the next pipeline stage.

3.2 Contour Analysis

Computer Vision toolset like OpenCV usually provides some contour analysis tools, which can be used to extract contours from a binary image, to match a contour against a template contour, etc. Contour analysis can be helpful in reducing possible detection candidates when the target objects are simple shapes, such as rectangles, circles or ellipses. Most road signs are in this case, so contour analysis can be applied here.

A road image after color analysis can be sent to contour extracting algorithm. An example output is shown as Figure 3. Contour analysis is based on the binary image (mask image) output of color thresholding. Contour matching algorithms usually take as input two contours and output a real number indicating the extent to what they match. A threshold can be put on this number to rule out candidates whose contours are too far away from the wanted contour. Contour algorithms do not use sliding window, thus is much faster than algorithms that are performed in a sliding window manner, like the cascade detector in stage 3.

Figure 3: A mask image output by color thresholding (left), the contour extraction result (middle), and result after removing small contours (right)

This stage can be viewed as a filter or validator of the previous stage. Some of the ROIs output by the previous stage will be rejected by this stage, because the mask (pixels selected by color thresholding) in that ROI does not have a wanted contour. But this rejection may be mistaken in situations where a sign is merged into its background with a similar color and the mask's contour represents the contour of the background rather than the sign. Thus a decision of whether to turn on this stage should be made for specific signs and specific conditions.

3.3 Haar-wavelet Cascade Detector

3.3.1 Cascade Classifier

The basic intuition of a cascade classifier is that when false samples are much more than true samples and most of the false samples can be easily precluded, it is very economical to firstly use some simple and cheap classifiers to exclude most of the false samples, leaving only the hardest ones to complex and expensive classifiers.

Every stage of a cascade classifier is an individual classifier, with more complexity and stronger classifying ability at higher stages. During operation, samples travel through each stage from lower to higher, and whenever a sample is classified as false at a stage, it won't go on to the next. A sample passing through all the stages is regarded as true.

Via such design, the average computation spent on each sample can be largely reduced. Suppose the probability for a sample to pass a stage is$p$, then the average number of stages a sample can pass is:

$$1 + p + p^2 + \cdots + p^{N-1} = \frac{1 - p^N}{1 - p} \tag{5-1}$$

where $N$is the number of all stages.

Lower stages have smaller computation complexity than higher stages. Suppose the computation complexity at stage n is n, then if we apply the most complex classifier, namely the N-th stage classifier, directly to every sample, the average computation for each sample will be N. However, if we use cascade classifier, the average computation for each sample will be:

$$1 + 2p + 3p^2 + \cdots + Np^{N-1} = \frac{1 - p^N}{(1 - p)^2} - \frac{Np^N}{1 - p} \approx \frac{1}{(1 - p)^2} \tag{5-2}$$

Next let's consider the precision of a cascade classifier. Suppose each stage's hit rate (the proportion of true samples that are correctly classified as true over all true samples) is h, and each stage's false-alarm rate (the proportion of false samples that are mistakenly classified as true over all false samples) is f, then the cascade classifier's overall hit rate H and false-alarm rate F is:

$$H = h^N \\ F = f^N \tag{5-3}$$

The above equation implies that as the number of stages gets larger, both hit rate and false-alarm rate will drop. In order for the overall cascade classifier to have a high hit rate and low false-alarm rate, h should be extremely high, while f needn't to be very low (since it will drop quickly thanks to the power operation).

We use 'cascade detector' and 'cascade classifier' interchangeable in this paper, because once we have a cascade classifier, we can do a sliding-window scanning on the whole image, at each position extracting a local patch and sending it to the classifier to decide whether we have a detect at that position. In this way we will have a cascade detector. Because cascade detector is used in a sliding window manner, it is the biggest time consumer of the whole system, and the major way of speeding up is thus to reduce of area of regions this sliding window is performed on, namely the ROIs. And that is what Stage 1 and 2 are mainly aimed for.

3.3.2 Stage Classifier

The design of each stage's classifier is similar to that in (Viola 2004). A stage classifier is composed of a set of single-feature Haar classifiers. A single-feature Haar classifier contains a Haar feature (described below) and a threshold. It matches the sample under classifying against its Haar feature to obtain a match score, and compares this score with its threshold to decide whether it classifies this sample as true or false. The stage classifier assigns a weight to each of its single-feature Haar classifiers, and uses the weighted average of the outputs of its little member classifiers to form its overall output.

The training of the stage classifier involves selecting a handful of single-feature Haar classifiers as its member classifiers from a large pool of possible single-feature Haar classifiers (due to the large number of possible Haar-wavelets), and in the

meantime deciding the threshold of each member classifier and the weights assigned to them. This procedure is accomplished by AdaBoost (Freund 1995) algorithm. Usually AdaBoost is used as a composer to generate a strong classifier from a bunch of weak classifiers, but it can also be regarded as a feature selecting algorithm, to pick out several good features (or single-feature classifiers) from a large candidate pool.

We use basic Haar-wavelets in our system. A basic Haar-wavelet can be one of four patterns as shown in Figure 4. For the first pattern (the top-left one on the right of Figure 4), six parameters are needed to describe it, namely the 4-tuple (left, top, right, bottom) to describe the outer rectangle, one number to describe the 'splitting position', and one number to indicate which side is black (and/or how black it is). The match score of a Haar-wavelet against a sample is the sum of pixel values in the region of the sample covered by the 'white' area of the Haar wavelet, minus that covered by the 'black' area.

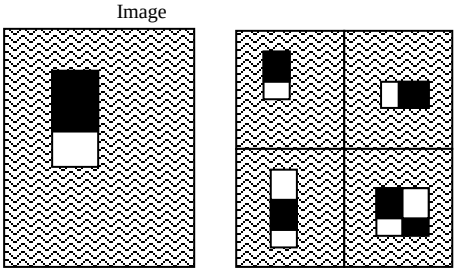

Figure 4: Basic Haar-wavelets

The benefit of basic Haar-wavelet comes from a fast calculating method of its match score, called Integral Image Method. Let $I(x,y)$denotes the pixel value at position (x,y), and $S(x,y)$ denotes the sum of pixel values within the rectangle $(0,0,x,y)$, then the sum of pixel values covered by rectangle (x_1,y_1,x_2,y_2)can be calculated as:

$$S(x_1,y_1,x_2,y_2) = S(x_2,y_2) + S(x_1,y_1) - S(x_1,y_2) - S(x_2,y_1) \quad (5\text{-}3)$$

Therefore, for a sample image, once we calculated and stored its integral image $S(x,y)$, we can compute its match score against any basic Haar wavelet in constant time.

3.3.3 Training of Cascade

The training algorithm of the overall cascade classifier is the same as that in (Viola 2004). The algorithm is described in Algorithm 1.

Input: *PosSamples*
Input: *NegSamples*
Input: *Fmax*: The maximum tolerable overall false-alarm rate
Input: *fmax*: The maximum tolerable stage false-alarm rate
Input: *hmin*: The minimum tolerable stage hit rate
Output: *Cascade*: The cascade classifier

```
Cascade = {}
F := 1
while F > Fmax do
  Stage = {}
  f := 1
  while f > fmax do
    Use Adaboost to find a single-feature
      Haar classifiers Haar that best
      classifies {PosSamples,
      NegSamples}
    Stage += Haar
   Tune Haar's threshold to make
      Stage's hit rate larger than hmin
    f := Stage's false-alarm rate
  end
  Cascade += Stage
  F := F * f
end
return Cascade
```

Algorithm 1: Training of cascade classifier

3.4 RANSAC Shape-Fitting

When we get some points that are believed to be generated from the edge of certain shape, in principle we can recover the generating shape (i.e. its parameters) from the information provided by these points. The perquisite is that we must presume of what category that shape is. For example, if we assume that these points come from a circle, only three points are needed to calculate the parameters of the circle, namely the coordinate of the center and the radius. A regular ellipse (long and short axes parallel to x and y axes correspondingly) needs 4 points, and an arbitrary ellipse needs 5. Typically we will have much more points than theoretically needed. RANSAC (RANdom SAmple Consensus) (Fischler 1981) is a method that exploits this redundant information to improve the precision and stability of shape fitting. It randomly selects points that are mathematically enough to calculate the shape parameters, and repeat this procedure certain times to get multiply sets of calculated parameters. The final values of parameters are decided by a voting among these calculated parameter sets.

In addition to gaining the values of shape parameters, RANSAC can also be used to check our presumption of shape category. For example, if we assume the generating shape is a circle, but the calculated parameter sets differ too much from each other, i.e. the statistical deviation exceeds some criteria, then we should forgo our previous assumption and claim that the shape would not be a circle. RANSAC is used in this way as a shape validator in our system. When our target road signs are ellipses (circles views from some perspective) and the fitting results of a detect deviate too much, we drop that detect.

4 OPTIMIZATION

The four-stage pipeline is still not enough to meet the strict speed requirement of automatic vehicle controlling. Three main problems regarding speed exist. They are the high variance of the size of ROIs returned by color space analysis in different environments (due to the color of the background), the dramatic drop of speed of the cascade detector when false-alarm rate is high, and the diversity of the target road signs.

When the background falls within the range of color space thresholds, a large region of interests will be returned by the color space analysis, leading to a big drop of speed of the total system. To stabilize the size of regions of interests, we implemented some tracking mechanism. A naïve implementation of a tracking mechanism could be that when a sign is detected and ensured (for example by a consecutive series of appearances), subsequent detection could be performed only on its neighborhood, with a thorough detection every several frames to allow for new signs. An advanced mechanism can utilize some tracking algorithms such as Markov chains or particle filters to predict next appearing position of a sign based on its previous observations. For our purpose, a naïve mechanism is good enough.

According to Equation (5-2), the speed of a cascade detector is determined by the probability of a sample passing a stage. If the system's total false-alarm rate is high, indicating that this probability is high, the speed of the cascade detector will drop sharply. And since cascade detector is the main time consumer of the whole system, the system's speed will also suffer much. As a result, we prefer lowering false-alarm rate to lifting hit rate during parameter adjusting, because a low false-alarm rate can serve both to precision and speed, and a moderate hit rate is somewhat tolerable in object detection, since the detection is measured object-wise, that is to say, the detection of an object should be considered succeeded as long as one of its many appearances is detected.

The diversity of the target road signs can be viewed in Figure 5 of Section 5. Intuitively there are four major categories, namely the red circular signs, the yellow triangular signs, the blue circular signs, and the blue rectangular signs, and some singular ones (like the red octagon sign). A cascade detector is based on the target object's whole appearance, including its inner texture and patterns, not just its contour shape, so theoretically we should train one cascade detector for each sign in Figure 5, leading to 25 cascade detectors in total, and run them all in detecting. The speed requirement cannot afford such an amount of computing, thus we trained one cascade detector for each of the four major categories, and one cascade detector for each singular sign, resulting in 9 cascade detectors in total. Multiple detectors also open the possibility of parallelization, and hence a large effort of our development was devoted into parallelizing our pipeline using techniques like OpenMP, which will not be covered in details in this paper.

5 EXPERIMENT RESULTS

To measure our system's precision and speed, we conducted a suite of experiment on a set of videos captured from real road. The set consists of 33 pieces of videos with solution 1280*960, containing totally 11345 frames. About 20% of the frames contain one or more road signs. A road sign is one of the 25 target road signs used in this experiment, as shown in Figure 5. Note the diversity of the colors, shapes and appearance patterns of these signs, which put a significant challenge on the system's precision and speed, as stated in Section 4. We thoroughly annotated the bounding boxes of the signs on all frames, and used them as ground truth for testing. We did the annotation deliberately in a way to make detection tasks 'hard' in that a sign was marked as long as it could be discerned by human eyes, some of them being only several pixels in size, and some partly occluded. Usually targets that are too small or occluded are not required in a detection task. The hardness of the dataset makes the current hit rate figures low but leaves space for future improvement. The training samples were extracted from a different set of videos other than this 33-video testing set, making the training data and testing data uncorrelated.

Figure 5: Road signs used in this experiment

For measurement, we defined several criteria. A detect (the bounding box of a sign) is treated as 'correct' or 'hit' is it intersects with a ground truth rectangle, and the area of intersection is larger than both 80% of the area of the detect and 80% of the area of the ground truth. A series of consecutive appearances of the same sign is annotated as an 'object' in the ground truth files. The testing dataset has 61 objects in total. An object's hit rate is defined as the proportion of the number of frames where it is hit over the number of frames where it appears. The overall hit rate of the whole dataset can be defined in two ways. In one way, it can be defined as a weighted average of the objects' hit rates, using the number of frames of each object's appearance in ground truth as the weights. (This definition can be equivalently stated as the proportion of the number of the ground truth bounding boxes been hit over the number of all ground truth bounding boxes, since the same sign appears no more than once in each frame). The second way is to define it as the simple arithmetic mean of the object's hit rates, which eliminated the domination of objects that appear in much more frames than others. We reported results in both definitions. The overall number of false-alarms is the number of detects that do not hit any ground truth. We reported it in the form of false-alarms per frame (number-of-false-alarms / number-of-all-frames, number-of-all-frames = 11345 in this experiment). We reported the speed of the system as Frames Per Second (FPS), which is measured over the whole running duration on the whole testing dataset. The speed is measured on a quad-core Intel i7 CPU with 2.8GHz main frequency.

In order to vary the trade-off between hit rate and false-alarm rate to get a ROC curve, we adjust the parameter of Minimum Neighbours, which is defined as the following. Because the detection is carried out in a sliding window manner, adjacent positions will trigger similar detection output, and hence the neighbouring positions of a sign would all tend to report detects. We can use this phenomenon to enhance our detection stability, requiring that a detect is acknowledged only if the number of neighbouring positions reporting detects exceeds a specified threshold. Adjusting this threshold, we can trade-off between hit rate and false-alarm rate (low threshold favours high hit rate while high threshold favours low false-alarm rate).

5.1 Full System Performance

Turning on all four stages of the pipeline, and adjusting Minimum Neighbours from 1 to 7, we have results listed in Table 1.

MN	HR1	HR2	FA	FPS
1	**0.586**	**0.475**	**0.132**	**7.86**
3	**0.572**	**0.467**	**0.131**	**7.87**
5	**0.557**	**0.455**	**0.131**	**7.58**
6	**0.532**	**0.432**	**0.138**	**7.18**
7	**0.516**	**0.419**	**0.136**	**7.90**

Table 1: Performance of the full system. MN means the Minimum Neighbors parameter. HR1 means hit rate in the first definition. HR2 means hit rate in the second definition. FA means false-alarms per second. FPS means frames per second.

In Table 1 the header MN means the Minimum Neighbors parameter. HR1 means hit rate in the first definition. HR2 means hit rate in the second definition. FA means false-alarms per second. FPS means frames per second. The first observation is that hit rate in definition 2 is usually lower than definition 1, indicating that the system performs better on some dominating objects in the testing data. These objects are some long and well captured signs that are clear and stable in the videos. On the contrary, many objects in the testing data are minor objects which are far and small and whose appearing sequences are short. Some of them were not detected at all. As the MN parameter rises, hit rates in both definitions go down, indicating that it's harder to fulfill a detecting criterion. However, the false-alarm rate doesn't drop much in return for the sacrifice of hit rate, and it even increases when MN goes above 5. This means that in practice setting MN to 1 is good enough for both hit rate and false-alarm rate. The speed (FPS) doesn't change so much among the settings, considering randomness of system environment. A FPS of about 8 is a conservative estimate, which doesn't take advantage of the optimization techniques like tracking described in Section 4, whose results will be reported in Subsection 5.5. A false-alarm rate of about 0.13 is good enough to be based by further process such as consecutive appearing validation during tracking and validation during recognition.

5.2 Effect of Contour Analysis

Turning off the Contour Analysis stage and thus letting all the ROIs output by the previous stage to reach the next stage, we got results shown in Table 2.

MN	HR1	HR2	FA	FPS
1	**0.782**	**0.729**	**0.244**	**3.834**
3	**0.777**	**0.723**	**0.247**	**3.845**
5	**0.764**	**0.714**	**0.248**	**3.748**
7	**0.733**	**0.676**	**0.255**	**3.960**

Table 2: Performance with Contour Analysis stage turned off. Meanings of the headers are the same as Table 1.

Note the increase of hit rate, but also the increase of false-alarm and the drop of FPS. It can be explained in that as a validating or filtering stage like the Contour Analysis stage is removed, more ROIs will be scanned in the cascade detector stage, leading to a higher hit rate as well as a higher false-alarm rate. More importantly, the increment in the number of ROIs will dramatically increase the computation amount for the cascade detector, which uses a sliding window procedure. As the FPS was nearly halved as a result of the removal of Contour Analysis, and dropped to an intolerable level, we can prove that the Contour Analysis stage is really a crucial component of the whole system, especially when it is intended to be used in a real-time circumstance. It also justifies our bias stated in Section 4 that in object detection tasks, we should better prefer a low false-alarm rate to a high hit rate, because a moderate hit rate can be tolerable since an object is detected as long as some of its appearances are detected, while a high false-alarm rate will lead to drops in both precision and speed, in systems like ours.

5.3 Effect of Color Space Analysis

The effect of removing Color Space Analysis is very obvious, so we just show one set of results, which are shown in Table 3. (Because the Contour Analysis is based on the output mask image of color thresholding, that stage have to be also removed.)

MN	HR1	HR2	FA	FPS
1	**0.572**	**0.533**	**1.446**	**0.694**

Table 3: Performance with Color Space Analysis stage turned off. Meanings of the headers are the same as Table 1.

Putting the nearly unchanged hit rate and the dramatic boost of false-alarm rate aside, the FPS alone would make the system unworkable. This simple result is enough to show that Color Space Analysis is a fundamental part of our system.

5.4 Effect of RANSAC

Turning off the RANSAC stage, we got results shown in Table 4.

MN	HR1	HR2	FA	FPS
1	**0.583**	**0.485**	**0.154**	**7.93**
3	**0.574**	**0.478**	**0.149**	**7.82**
5	**0.560**	**0.468**	**0.147**	**7.73**
7	**0.517**	**0.428**	**0.154**	**7.70**

Table 4: Performance with RANSAC stage turned off. Meanings of the headers are the same as Table 1.

We can draw these results together with those in Table 1 to compare the differences more intuitively, as shown in Figure 6 for hit rate definition 1 and Figure 7 for hit rate definition 2.

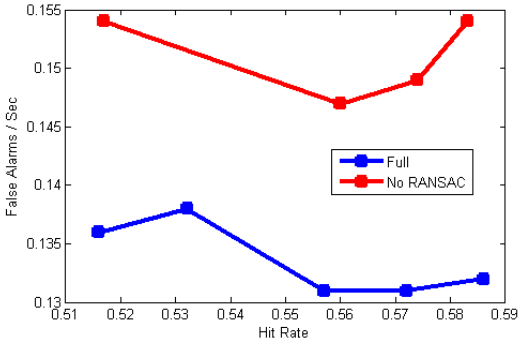

Figure 6: Comparison of precision between full system and that with RANSAC stage turned off. Hit rate uses definition 1. Curves are generated by adjusting MN parameter.

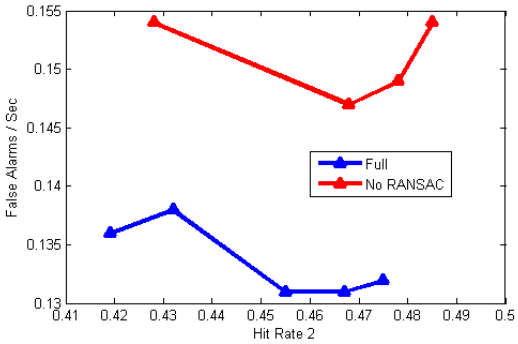

Figure 7: Comparison of precision between full system and that with RANSAC stage turned off. Hit

rate uses definition 2. Curves are generated by adjusting MN parameter.

As can be seen both in Table 4 and Figure 6 or 7, turning off RANSAC stage doesn't affect hit rate much, but increases false-alarm rate by about 15%. Equally saying, introducing RANSAC can lower false-alarm rate by about 9% without sacrificing hit rate. The speed is not affected much either. This proves that RANSAC is a safe and feasible post processing stage for the system, though the influence is not as dramatic as color and contours.

5.5 Effect of Tracking

In addition to the full system tested in Subsection 5.1, we can add simple tracking mechanism described in Section 4 to stabilize the performance and further boost the speed. It constrains the scanning region of each frame to the neighborhood of the detects in the previous frame (and do no detection if the previous frame's detection result is empty). It will perform a thorough scan every several frames (3 in this experiment) to allow for new objects. The results of this addition are shown in Table 5.

MN	HR1	HR2	FA	FPS
1	**0.535**	**0.432**	**0.081**	**11.750**
3	**0.517**	**0.421**	**0.080**	**11.757**
5	**0.509**	**0.409**	**0.081**	**11.580**
7	**0.474**	**0.384**	**0.070**	**11.641**

Table 5: Performance with tracking mechanism added. Meanings of the headers are the same as Table 1.

Surprisingly and happily, both precision and speed enjoy a significant enhancement. The precision is enhanced in that the false-alarm rate drops by more than 60% without the hit rate suffering much. The FPS is increased by about 50%. The reason for this double effects is that by constraining the scanning region to the previous results' neighborhood, we on the one hand reduced the amount of computing and the possibility of false -alarms, and on the other hand ensured that the wanted sign was contained in the ROIs, which may have been mistakenly precluded by Color Space Analysis or Contour Analysis, and thus stabilized the continuous hitting of an already detected target. In this way, hit rate, false-alarm rate and speed can all benefit (or at least not suffer) from the tracking mechanism.

An example detecting result is shown as Figure 8.

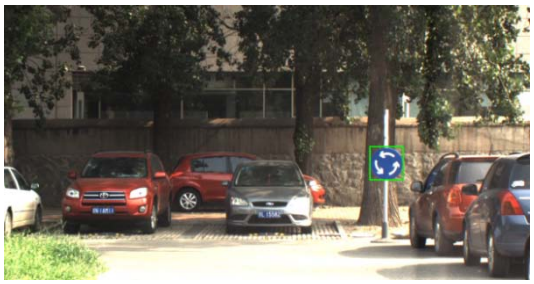

Figure 8: An example of detecting result.

5.6 Experiment Summary

The results of this experiment is conservative in that because the system is highly configurable for each sign or each set of signs to have its own configuration of the pipeline, further fine tuning is possible to enhance the performance on each specific sign (for example some signs are suitable for Contour Analysis while some are not). We used only one setting for all signs in this experiment in order to just show the effectiveness of each stage in principle. Also some additional information could be utilized in specific circumstances, for example, the guarantee that there will be no more than one sign in each frame, or a smaller view or solution of the video, or a narrower candidate pool than the whole 25 signs. In an automatic driving vehicle contest, our system reached an FPS higher than 20 with fine tuning and further information utilized.

6 CONCLUSION

In this paper we presented a system that can fulfill the task of road sign detection in real-time and real world circumstances. By effectively utilizing various computer vision techniques, we proved that though there may not be a single algorithm that can tackle all the challenges in a real world task, a wise selection and hybrid of existing techniques can still produce a feasible and robust application.

REFERENCES

Broggi, A., Cerri, P., et al., 2007. Real Time Road Signs Recognition. In *Intelligent Vehicles Symposium.* IEEE.

Escalera, A., Luis, E., Moreno, M., et al., 1997. Road Traffic Sign Detection and Classification. In *Transactions on Industrial Electronics*. IEEE.

Escalera, A., Armingol, J., et al., 2003. Traffic sign recognition and analysis for intelligent vehicles. In *Image and Vision Computing.*

Viola, P., Jones, M.,et al., 2004. Real-Time Face Detection. In *International Journal of Computer Vision.*

Papageorgiou, C., Poggio, T., 2000. A trainable system for object detection. In *International Journal of Computer Vision.*

Bahlmann, C., Zhu, Y., et al., 2005. A System for Traffic Sign Detection, Tracking, and Recognition Using Color, Shape, and Motion Information. In *Proceedings of Intelligent Vehicles Symposium*. IEEE.

Lienhart, R., Maydt, J.,2002. An Extended Set of Haar-like Features for Rapid Object Detection. In *Proceedings of ICIP*. IEEE.

Soetedjo, A., Yamada, K., 2005. Fast and Robust Traffic Sign Detection. In *Proceedings of International Conference on Systems, Man and Cybernetics*. IEEE.

Barczak, A. L., C., 2005. Toward an Efficient Implementation of a Rotation Invariant Detector using Haar-Like Features. In *Proceedings of IVCNZ.*

Freund, Y., Schapire, R., 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational learning theory.* Springer.

Fischler, M., Bolles, R., et al., 1981. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. In *Communications of the ACM.*