ROAD SIGN RECOGNITION

DAQING LI

A THESIS
IN
THE DEPARTMENT
OF
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE AT
CONCORDIA UNIVERSITY
MONTREAL, QUEBEC, CANADA

SEPTEMBER 1998

0-612-39486-7

Abstract

Road Sign Recognition

Daqing Li

A method for recognizing road signs in street scene images is proposed. Previous methods typically perform recognition using simple detection algorithms based on a few major features. They either rely on the detection of a few global features, which is not reliable; or use special algorithm to detect certain predefined shapes, which is not general enough. Further more, there is a lack of effective classification methods to verify the detected object candidates from color image. In our work, we designed a set of algorithms to deal with these problems.

We first developed a fast region growing segmentation method, combined with a 1NN smoothing filter, to work on color and produce robust segmentation results even when the image is noisy. Edges and regions come out naturally as the result of the segmentation, and an image is represented in terms of these basic geometric features. An effective color distance measurement is developed to work with the segmentation.

We then designed a hierarchical feature representation scheme, which organizes model features in a tree structure. The tree nodes are components that form the basic parts of a model, and the tree leaves are primitive physical features that can be mapped directly to basic features in segmented image. This structure is quite expressive in describing the semantic meanings of road sign contents, and can also help to speed up the detection. A matching error model was defined to work with the feature representation scheme.

To deal with the complexity in object detection, we designed an anchor feature and its detection scheme for the use of alignment detection. Alignment transformations are further verified by a unified matching scheme, which integrates feature ordering and performance tactics into one process. The matching process incorporates the hierarchical feature scheme for better performance and accuracy.

We used a modified Hausdorff distance as the final verification method in matching a candidate to the model. An efficient algorithm in calculating the distance using Voronoi diagram based on edge map as point set is developed. This distance is used due to its resilience to translation and noise. We choose to use this method also due to its discerning power with just a few training samples.

The experiment shows very encouraging result.

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Dr. C.Y.Suen, Dr. T.D.Bui and Dr. T.Kasvand for their invaluable support and help during the years of my study.

I wish to thank Dr. Suen for giving me the opportunity to work in such an interesting project, for providing well-equipped lab environment, and for the knowledge learned from his stimulating graduate course works. I want to thank Dr. Bui for his continuous support, guidance and care over the years. His encouragement and care are the most important incentives for me to finish my work. I also want to thank Dr. Kasvand for many helpful technical comments and discussions, and for helping me to make constant progress and to prepare technical reports. Without their support and help, this work would be impossible.

I am also indebted to Dr. Ying Li and Marc Lalonde for their free share of ideas and insight knowledge about the project.

I am also thankful to many other people in CENPARMI and in the Department of Computer Science who helped me in various ways.

Last but not least, I am extremely grateful to my parents and my dear wife Jian, for their unbounded support and care.

Contents

List of Figures

1 Introduction

This thesis describes the research and development result of a road sign recognition project. In this project, we try to build a system that can detect and recognize road signs from color images taken by cameras. The primary intention to build such a system was to use it in moving vehicles as a kind of driving assistant, providing vision-based warning mechanisms for possible key traffic signs, e.g. stop sign, yield sign, speed limit sign etc. However, the techniques involved in this system can be applied to many other object recognition problems.

1.1 The challenge

The major challenge in road sign recognition is that a scene image of a road environment is often very complicated. It may contain a large number of different kinds of natural and artificial objects. Natural objects include trees, grass, sky, mountain, pedestrians and other forms of natural life. Among artificial objects are roads, road markers, cars, traffic signs, and buildings. These objects often overlap over each other, breaking the image into a cluster of locally non-recognizable small fragments. This complication often leads to exponential number of mapping combinations that a recognition system has to check against, and thus greatly degrades the efficiency of a recognition system. (Fig. 1)

Another difficult factor in road sign recognition is the ubiquitous noise in an image of natural environment. Noise may come from the camera, occlusion of objects, rust on the signs, varying lighting conditions, shading and other spurious data. With the presence of noise, features extracted from images are often not reliable – edges may be broken, shading may make the contrast unstable across the image, a portion of an object may be unseen due to occlusion, color may change under different lighting conditions. This unreliability of features has a direct impact on a system's performance since any detection and recognition algorithm has to depend upon the presence of certain features (Fig. 2).

The dimension of an image under process may be quite big. The typical dimension for images in our project is about 600 by 400 pixels, 24 bits of color for each pixel. That is about 720K bytes per image and this amount of information will doubtlessly cost considerable processing time. Insufficient resolution is also a problem. Human eyes can recognize a sign of 10×16 pixels (the direction sign in Fig. 1). However, that size is very small for computer to recognize reliably, especially in noisy environment (Fig. 3)

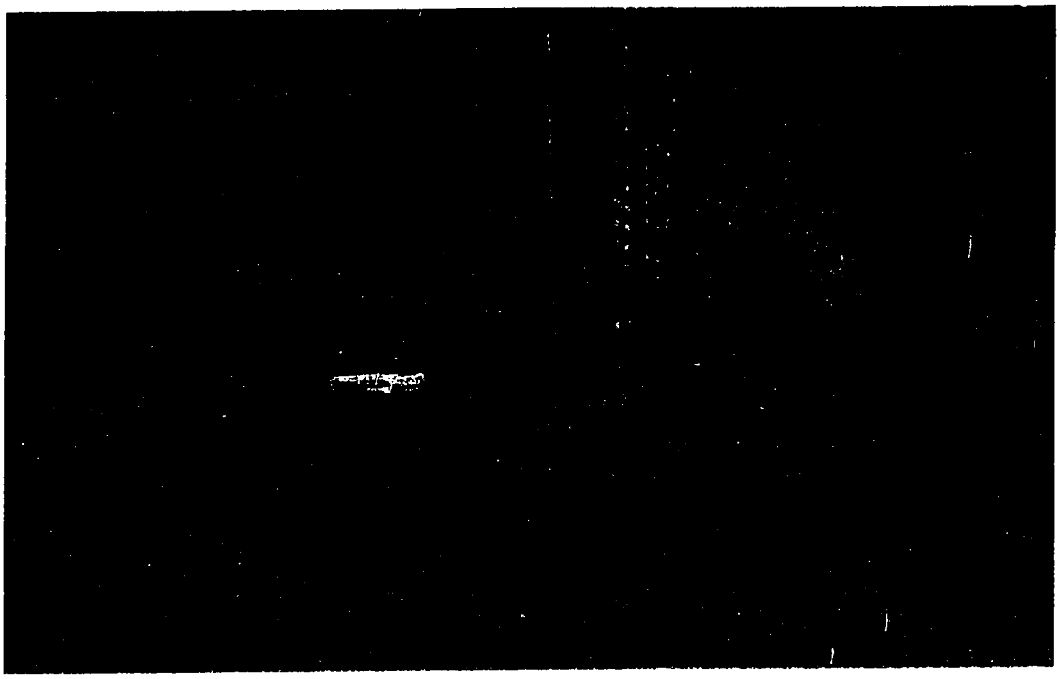


Figure 1: A street scene.

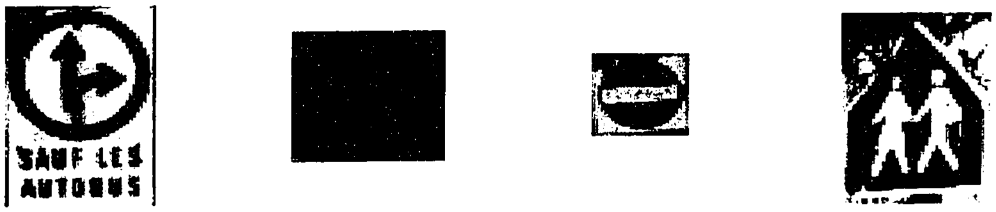


Figure 2: Road signs in different conditions.

Figure 3: Enlarged version of a small sign (original size: 9X15).

The last but not least difficulty is the lack of training samples, or better say, the big object space. Due to the scale of the project, the number of road sign samples available is quite limited. Moreover, the model space is very big. Suppose we use 24 by 24 pixels samples with 24 bit color for each pixel, the model space is overwhelmingly bigger than 24 by 24 black and white models normally used by an optical character recognition algorithm. Given the wide variations a sign could be presented under noisy conditions, the task of collecting a reasonable amount of training samples to cover this huge space is very intimidating. Therefore, the classifier of the system has to make a decision using insufficient training samples, which is a very difficult job.

Although there are many difficulties compared to other possibly easier recognition problems, to be fair, we should point out that road signs are probably the easiest to be recognized in street environment. In other words, if we want computer to recognize objects in streets and don't know how to do that, road sign is the best starting point.

1.2 Previous work

There are quite some reviews in the field of object recognition. Pope [2] did a concise review covering some late development in model based object recognition. Suetens, Fua and Hanson [3] covered a variety of matching techniques. For those who are interested in road sign recognition, please refer to the survey made by Lalonde and Li [1]. Here we will discuss in general terms the techniques that are used in previous work concerning road sign recognition.

Segmentation is usually the first step in object recognition. A commonly used method in segmentation is color thresholding. Bartneck and Ritter [20], Kehtarnavaz et al. [17], Akatsuka and Imai [6] all used this method to pick out the designated color of road sign. Saint-Blancard [7] used camera with a red cut-off filter to enhance the contrast between red and other colors. This method is easy to implement and fast. Its output could be a cloud of pixels of similar color and normally further processing is needed to find out geometrically meaningful subsets. Bartneck and Ritter [20] used a connected component analysis to generate the symbolic description of all color labeled regions. Saint-Blancard [7] used edge detection and following to get the closed contours. Kehtarnavaz et al. [17] also used geometrical analysis of the color patch to decide if a hypothesis is a stop sign.

The drawback of color thresholding is its vulnerability to noise and changes in lighting condition, besides it does not consider geometric relation among pixels. More sophisticated methods for segmentation include edge detection and region growing. Piccioli [4, 5] used an edge detector on gray level image. Priese and Rehrmann [8], Nicchotti et al. [12] used region growing in segmenting color images. Both edge detection and region growing produce result that can be directly used in sign detection without further processing.

Based on the result of segmentation, shape analysis is then used to detect the object candidates. However, one will soon find out that quite often segmentation alone can not produce features that are reliable enough for shape analysis. One solution to this unreliability is to use smaller features. Priese et al. [11] defined basic shapes of traffic sign components (circles, triangles, etc.) with 24-edges polygons describing their convex hulls. At the detection stage, patches of colors extracted by the segmentation algorithm are collected in "object lists", with one list for each sign color (blue, white, red...). All objects are then encoded, and assigned a probability (based on an edge-by-edge comparison between the object and the model) to characterize their membership to specific shape classes. A decision tree is constructed for the final detection and recognition. By using 24-edges polygons, the algorithm can rely on smaller features that tend to be more stable. However, finding out 24-edge polygons more complicated or fragmented (due to noise or occlusion) may be more difficult.

Another way to get around the feature unreliability is to use specialized detector that's good at detecting a particular shape. Method proposed by Bresserer et al. [10] classifies chain-coded objects according to evidence (in the Dempster-Shafer theory) supplied by knowledge sources. These sources are a corner detector, a circle detector, and a histogram-based analyzer. When an unknown shape is presented to the module, each knowledge source studies the pattern and computes a basic probability assignment attached to the feature found. The probabilities are combined with the Dempster rule, and a semantic network builds up the belief for each class that the unknown shape is a member of that class (classes used are circle, triangle and polygon). Here they were using specialized detector for predefined shapes. Interestingly, they still pointed out that the limiting factor is the segmentation quality, which should be improved.

Similar work was done by Piccioli [4]. Clusters of edge-points having the desired shape (e.g., triangular or circular) are extracted. Each cluster is a candidate to be a road sign. The inner region of each candidate is then tested against the date-base of signs by template matching. They used different algorithms to detect triangles and circles. For example, possible circles are detected by matching edges with a set of 10 annuli, three pixels thick, with inner radius ranging from 15 to 24 pixels.

Using specialized detector is more reliable and efficient. The drawback is the number of recognizable objects is limited to the kind of detectors. It's not easy to extend the system to handle other shapes. On the other hand, using smaller feature is more flexible, while it is usually not as efficient due to the usually huge number of mapping combinations.

A final verification step is often necessary to validate the object candidates detected in the image. Interestingly, this step is often omitted in object recognition systems, or if presented, is not powerful or efficient enough to make it practical. The lack of final verification may come from the difficulty in classification in large model space. Zheng et al. [9] trained statistical classifiers on colors, shapes and pictograms of road signs. The result of the classifiers is n-best interpretations of the road sign type and their respective certainties. Saint-Blancard [7] tried programming method, expert system, and neural network in classification.

Some road sign recognition projects developed dedicated hardware to achieve high speed of recognition. Better or even specially designed hardware is definitely very important in building a system of high performance. However, this topic is beyond the scope of this thesis.

The complication of city street environment makes the recognition task more difficult than many other object recognition tasks. As is put by Ulmer, a researcher in Daimler-Benz (Traffic Engineering, Prometheus 1996): "The concept (of their current approach) works really well on highways, even when the car is doing 130 kilometers per hour. However, our computer can't yet cope with city traffic, intersections or oncoming traffic." Indeed, road environment is nonuniformly structured. The most structured environment is highway, then follow several other types of roads, and city streets are the least structured. As observed by Blancard [7]: "The city street environment is the most complex because of superaboundance of different manufactured objects along with natural objects... (it) does not have the same nature as do roads and highways and should be considered independently" These remarks fully reflect the difficulties in road sign recognition in city street environment.

As one can see, there is still much room for improvement in almost every technical aspect in road sign recognition, such as segmentation, feature extraction, detection and recognition algorithms.

1.3 Design criteria and evaluation

Before we start designing our own road sign recognition system, it is important to understand the requirements that define a good solution. There are some commonly accepted criteria in evaluating an object recognition system, as summarized by Grimson [62]:

Efficiency: A method should be efficient. Efficiency is measured in both run time and computation complexity.

Correctness: A method should produce correct result. Correctness is measured in the ratios of false positive (incorrect mapping) and false negative (correct mapping have not been found).

Robustness: A method should degrade gracefully in extreme conditions, such as increase of noise, degradation of lighting conditions, increase of irrelevant data, the decrease of relevant data.

Scope: A method should cover as broad circumstances as possible. Here the circumstances include type of objects and the way the objects are presented. Examples of scope issue are the number of objects to be recognized, whether or how much occlusion and spurious data is allowed, how cluttered an environment can be, how strict the requirement on sensory data quality is etc.

It is often difficult to measure these criteria quantitatively. Some of the criteria can be measured by numbers. A system's efficiency can be measured by the time needed in processing a single image of certain size; its correctness can be measured by the ratio of false positive and false negative. Since all of the criteria are related to each other, none of them can be measured alone meaningfully. For example, since the image quality and complexity used in each system may be different, it's hard to measure the performance across different systems. Even letting different systems work on the same set of images won't necessarily produce reliable evaluation, since some systems perform well on certain kind of images while work badly on others.

Although it is difficult to compare different systems quantitatively, it is still possible, and often useful, to measure some of the criteria numerically within a single system, to provide knowledge about the system's performance in certain context. This will help to evaluate a system and provide evidence to further analyze and improve different approaches.

1.4 Our approach

Most road sign recognition systems consist of several, if not all, of the following major steps: segmentation, model feature representation, sign detection and final recognition. In segmentation, pixels in the image are grouped into larger units, such as edges and regions, according to their geometric and/or photometric relations. This step greatly reduces the volume of information contained in an image, and also represents the image in basic geometric features. Model feature representation process is used to extract and organize the feature information of an object model. Based on image and model representation, detection algorithm is performed to find the candidates of model object in the image. Usually a system searches for possible object candidates that possess enough features similar to that of an object model, then makes the final recognition decision using object matching algorithms. We will follow these steps in our method.

1.4.1 Segmentation

There are many possible ways to do segmentation. Each method has its strength and weakness. Color thresholding is effective in selecting out desired colors, but does not consider geometric relations, and some kind of post-processing is needed to get the final segmentation result. Edge detection captures local change accurately but is susceptible to noise. Besides, there is a lack of edge operators on color image. Region method takes both color and locality into consideration. Region growing method generates both region and edge quite naturally. A fast region growing method is developed to extract both edge and region information from the image. We have also designed a color distance to work with the segmentation. It has been shown that some local smoothing filter would be very useful in bringing better segmentation results. To preserve the local fine details, and also achieve smoothing, we used 1NN filter. The result is quite satisfactory.

1.4.2 Representation

Object models are represented in terms of features. We designed a hierarchical feature representation scheme which organizes features into different levels. Higher level features are more general, lower level features are about finer details. Organizing features into such a structure has several benefits. It is quite expressive in describing different relations among features, and helps more accurate detection and recognition. Another benefit is from its tree structure. Better performance is achieved by using layered information for coarse-to-fine hierarchical searching and verification. Different matching quality models have been studied and a weighted error model is proposed.

1.4.3 Detection

Basic search methods, such as constrained tree searching, geometric hashing and alignment methods are studied. The issue is tradeoff between efficiency and correctness. It has been shown that by choosing the right anchor features, alignment method can achieve polynomial search time. We proposed to use bounding edges of components as anchor features, and developed constraints to detect such features efficiently. Alignment transformations are further verified by a unified matching scheme. To address the performance issue in detection, our matching scheme integrates feature ordering and performance enhancement techniques into one process. Using the feature model designed in our work, the matching process starts from the top-level features and proceeds down to bottom level features, uses the matching error model for quick matching termination.

1.4.4 Recognition

Hausdorff distance is studied as the method to recognize road sign candidates. This distance is chosen due to its resilience to slight translation and distortion, which is an important property for recognition in noisy environments. Another nice property of this distance is its ability to recognize using a small number of training samples. In order to overcome its sensitive response to noise, we proposed a modified version of the Hausdorff distance using edge map as the point set. An efficient algorithm is proposed to compute the distance using Voronoi diagram.

2 Segmentation

In this chapter, we will first give our view of what segmentation means, especially to our task at hand. Then we will establish our color distance measurement that will be used in color image segmentation and object detection. After a brief review of major segmentation methods, we will design and implement a fast region growing segmentation method, with the help of local smoothing filter.

2.1 The definition

Roughly speaking, segmentation is a low-level image representation process which groups image pixels into larger units such as edges and regions. It is low-level because only geometric and photometry properties are used; no assumption is made for underlying semantics of image contents. The purpose of segmentation is to reduce the volume of information presented in an image, and in the meantime, keep all the important information needed for object recognition or whatever purposes it is for.

The classical definition of segmentation [13] is as follows:

Difinition: Segmentation of a discrete image signal f(m, n), where $\{0 \leq m \leq M - 1 \cap 0 \leq n \leq N - 1\}$, is the division of f into disjoint nonempty subareas $f_1, f_2, ..., f_n$ which satisfy some criterion of uniformity E:

(i) $\bigcup_{i=1}^{P} f_i = f$.

(ii) f_i is connected $\forall i$ with i=1,..., P.

(iii) $\forall f_i$ the criterion of uniformity $E(f_i)$ is satisfied.

(iv) $E(f_i \cup f_j)$ is not satisfied for any union of two neighboring f_i, f_j.

This definition is very much region oriented. The first condition simply requires that the segmentation be complete: every pixel in an image must be in a region. The second condition is the topological requirement that regions must be connected. The property, which determines the segmentation, is introduced in the third condition. The fourth condition expresses the maximality of each region in the segmentation.

In light of object recognition, the first condition may not be necessary. We can be satisfied by an incomplete segmentation as long as all the discernible edges and regions are detected. For example, in edge detection, we don't need to segment the image into complete regions; we can also just find out relatively "flat" regions and skip "rugged" terrain.

Further more, this definition is single-layered. The second condition means that the segmentation is defined on a single layer and there is only one explanation of edge and region about the image. This may not be sufficient since objects may present under various conditions; not all of them can be properly presented using a uniform criterion. We may need to use multi-layered segmentation to reveal image features on different scales.

As for the third condition in the definition, obviously the uniformity criterion should be strict enough to avoid over-compression of image information. Stricter criterion may cause over-segmentation, i.e. too many edges and regions are generated, but at least key information is better preserved.

2.2 Use of color

Color provides a key information for human beings to distinguish objects in real world. Although recognition is still possible with gray-level image, color will make signs more distinguishable and easier to find. So almost every traffic sign recognition system found in the literature uses color and acknowledges its importance. The major purpose in studying color space is to find out how to measure the distance between two colors. This measurement is very important to color image segmentation.

Similarly to human eye which has three kinds of receptors (cones) for sensing specific parts of the spectrum, modern cameras perceive color with three sensors, each for a 'primary' color: red, green, and blue. So an image got by a camera is represented by a collection of three coordinate (R, G, B) pixels. The data space containing the pixels is called the color space. Apart from RGB space, there are many ways to represent color depending on the application. Here we discuss two most commonly used color spaces.

2.2.1 RGB color space

Analyzing an image in RGB color space is convenient and tempting since the color coordinates of pixels are supplied by the camera without transformation. Two colors are similar to each other if they are adjacent within a small neighborhood in the RGB space. Another nice property of RGB space is that the mean of two colors corresponds to the mixture of both in human sensation.

The drawback of this space is that two colors that are similar to each other may be far apart in RGB space. The three coordinates are highly correlated [32]. Variations in ambient light intensity have a disastrous effect in RGB values by shifting the clusters of color pixels significantly in the cubic space, while the actual "colors" of these pixels are still the same.

2.2.2 HSV color space

The HSV color model was introduced to approximate the perceptual properties of "hue", "saturation" and "value" [33]. It has the distinctive feature of being similar to the way colors are perceived by humans. Hue is the color aspect of a light stimulation, corresponds to a narrow range in the color spectrum, such as red, green, yellow etc. Saturation is the "purity" of a color, signifies the white content of a stimuli, a "fully saturated color" is a "pure" color, while grays are "desaturated colors". Brightness value is the scale of a color from dim to light; it highly correlates with physical intensity of light. RGB coordinates can be mapped to HSV space with the use of non-linear transformations:

$$max = Maximum(R, G, B)$$

$$min = Minimum(R.G.B)$$

$$V = max$$

$$S = \frac{max - min}{max}$$

$$H = \begin{cases} 60(g-b)/(max-min) & \text{if (r=max)} \\ 60(b-r)/(max-min) & \text{if } g = max \\ 60(r-g)/(max-min) & \text{if } b = max \end{cases}$$

$$\text{If } H < 0, \quad \text{then } H = H + 360.$$

So the range of H will be within $0° - 360°$.

The HSV color space is certainly appealing for color processing because chromatic information is represented by the hue coordinate, and varying light conditions are "absorbed" (to some extent) by the intensity value coordinate.

However, there are problems with this space also. There are singularities in the Hue dimension along the gray-level axis (R=G=B), where small perturbation in the RGB signals may cause strong variations in hue. Besides, both S and H are unstable (undefined) when light intensity (V) is very low.

2.2.3 Color distance

The advantages for RGB space are that it is simple, easy to manipulate, can be used to find out similar colors and calculate mean of two colors. Its drawback is that it can not give correct similarity measure on hues. On the other hand, HSV space can give intuitive measurement of hues, but this measurement is not applicable when the hue is poorly defined, i.e. when the S or V is low.

Here we see a very good chance to combine the merit of these two spaces together. We can use HSV whenever the hues of two colors are both well defined, and use RGB space otherwise. This makes perfect sense in that when color is abundant, HSV space, particularly H (the hue) plays a more prominent role in differentiating different colors. If the hue of any one of the two comparing colors is poorly defined, we can not use hue as a distance measure any more. In this case, we can rely on the RGB distance.

However, neither space is metric space, so the distance measurements only make sense in a small neighborhood. Fortunately, we just need to measure a small neighborhood in segmentation. Any two colors that are too different to be measured effectively within a small color neighborhood can be considered to have an infinite mutual distance.

There are several ways to defined distance in both RGB and HSV space. For RGB, we have Euclidean distance,

$$D_{rgb} = \sqrt{(R_1 - R_2)^2 + (G_1 - G_2)^2 + (B_1 - B_2)^2}$$

absolute distance,

$$D_{rgb} = |R_1 - R_2| + |G_1 - G_2| + |B_1 - B_2|$$

maximum sub-component distance,

$$D_{rgb} = \max(|R_1 - R_2|, |G_1 - G_2|, |B_1 - B_2|)$$

After many tests, we found that maximum sub-component distance gives the best result. It is also very easy to calculate.

In HSV space, we can have similar choices as in RGB space. However, we consider hue as more important than luminance and saturation. Hue plays a bigger role in identifying road sign with certain color, and is more stable than the other two components. Therefore, we define the color distance in HSV as the difference between the hues:

$$D_{hsv} = \begin{cases} |H_1 - H_2| & if|H_1 - H_2| < 180^\circ \\ 180 - |H_1 - H_2| & if|H_1 - H_2| > 180^\circ \end{cases}$$

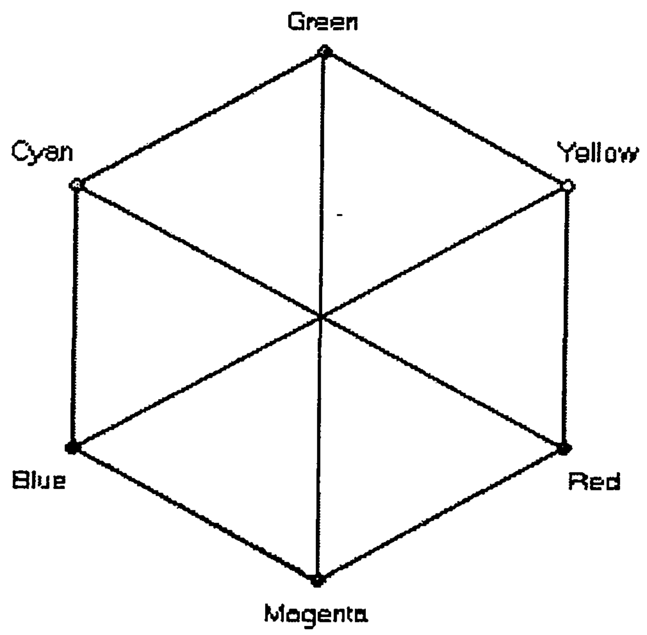


Figure 4: Color hexagon shown hue position for different colors.

As a matter of fact, we will only consider hue difference within a small neighborhood; beyond this neighborhood, hue difference does not make much sense. For example, we can not say that blue is closer to red than cyan (Fig. 4). If the hue difference is too big, we prefer to use RGB distance instead.

Our color distance measure algorithm can be described as follows:

Constants:
I_{min}, S_{min}: minimum value of intensity and saturation that a meaningful hue can be defined.
ΔH_{max}: maximum useful hue difference. Hue differences larger than that will be considered meaningless.

Input:
color 1: defined by RGB and HSV values $R_1, G_1, B_1, H_1, S_1, V_1$.
color 2: defined by RGB and HSV values $R_2, G_2, B_2, H_2, S_2, V_2$.

Output:
D: the distance measure of color 1 and color 2

The algorithm:

```
$D_{hsv} = |H_1 - H_2|$;
if $D_{hsv} > 180°$ then $D_{hsv} = 180° - D_{hsv}$;
$D_{rgb} = |R_1 - R_2| + |G_1 - G_2| + |B_1 - B_2|$;
if $V_1 > I_{min}$ and $V_2 > I_{min}$ and $S_1 > S_{min}$ and $S_2 > S_{min}$ and $D_1 < \Delta H_{max}$
then $D = D_{hsv}$
else $D = D_{rgb}$.
```

2.3 A brief review of segmentation

Segmentation methods can be roughly classified into three categories [14, 15, 16]: pixel classification, edge detection, and region-based methods. We will discuss briefly these segmentation methods and underline the reason for our choice of segmentation method in this work.

2.3.1 Pixel classification

Pixel-based segmentation techniques utilizes classification methods to classify each pixel into classes. The image segments are defined as the maximum connected components of the pixels belonging to the same class. So normally a post-processing step is needed to find out the edges and regions. This kind of technique is parallel since segments are formed after all the pixels in the image have been classified. Pixel classification is performed based on the feature values of the pixels.

Depending on different feature spaces and different classification strategies, pixel classification segmentation methods can be divided into several classes: 1. Histogram thresholding: when the feature space is one-dimensional. This generally applies to gray-level image. 2. Clustering: when the feature space is multi-dimensional, e.g. color image. 3. Other classification methods: such as neural network, polynomial classifier etc.

The classification can be supervised or unsupervised.. In supervised classification, training samples are provided to the classifier by human to establish the mapping. The assumption is that some objects are formed by certain gray-levels/colors. By pinpointing these gray-levels/colors, one can single out the regions that possibly belong to the objects of interest. Examples are color registration and clustering mentioned in [17], the neural network classification methods in [18] and [19], and polynomial classification [20].

Unsupervised classification partitions the feature space based on some criterion func-

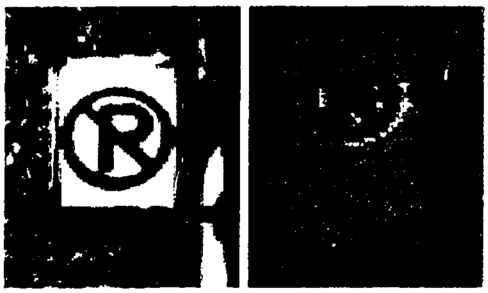


Figure 5: Color/intensity shifts under different lighting conditions.

tions. The assumption behind unsupervised pixel classification methods is that the regions generated after segmentation are formed by pixels of the same (or similar) color, and thus these pixels will form clusters in the feature space. By finding out these clusters, one also finds out the segmentation. Examples are foreground and background classification [21], maximized entropy [22], and some color space clustering methods [23].

Supervised methods are useful in finding out predefined colors, as is popularly used in previous road sign recognition systems, although its performance is not as good in poor light conditions where color might shift (Fig. 5). And for unsupervised methods to success, the following preconditions have to be met: 1. For histogram thresholding, the number of objects contained in the image is known a priori. 2. Objects are represented within ranges of intensity/color which overlap each other by only small amounts. 3. Suitable threshold/cluster can be calculated. In reality, these preconditions are often not fulfilled. In complicated natural environment, the number of objects is unknown and their intensity/color range may overlap to such extent that the intensity/color histogram will look like uni-modal. A street scene is shown in Fig 6, its histograms are shown in Fig. 7). Besides, three-dimensional color clustering is a computationally expensive process [24].

Another major drawback of pixel classification methods is that there is no requirement for good spatial continuation and the resulting segmentation boundaries are very noisy and busy. Edges are often not continued, and regions formed tend to be fragmented.

2.3.2 Edge detection

Edge detection locates points of abrupt changes in intensity values/colors. The segmentation is generally a two-stage process. First every pixel is assigned a gradient value, which can be a complex vector; that is, it can have magnitude and direction.

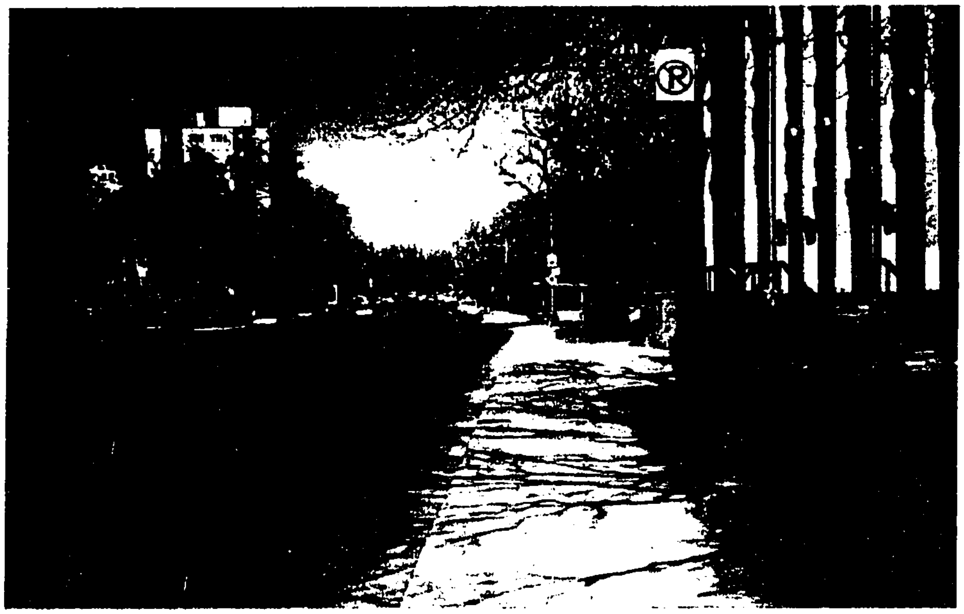


Figure 6: Original image

After getting the gradient images, edges are produced using suitable post-processing tracing methods.

Standard gradient operators

There are a number of standard gradient operators to calculate the gradient edge map of an image, such as Roberts operator, Sobel operator, Kirsch operator, Prewitt operator and the Laplacian operator [25, 26, 27, 28].

The designation of image pixels in the local operator window shown in Fig. 8 can be used to define these operators. For example,

Roberts operator:
$G = \max(|F(6) - F(9)|, |F(7) - F(5)|)$

Sobel operator:
$G = |(F(6) + 2F(7) + F(8)) - (F(4) + 2F(3) + F(2))| + |(F(6) + 2F(5) + F(4)) - (F(8) + 2F(1) + F(2))|$

Pseudo-Laplace:
$G = |F(1) + F(3) + F(5) + F(7) - 4F(9)|$

It has been shown that local gradient operators do not handle noisy image signal very

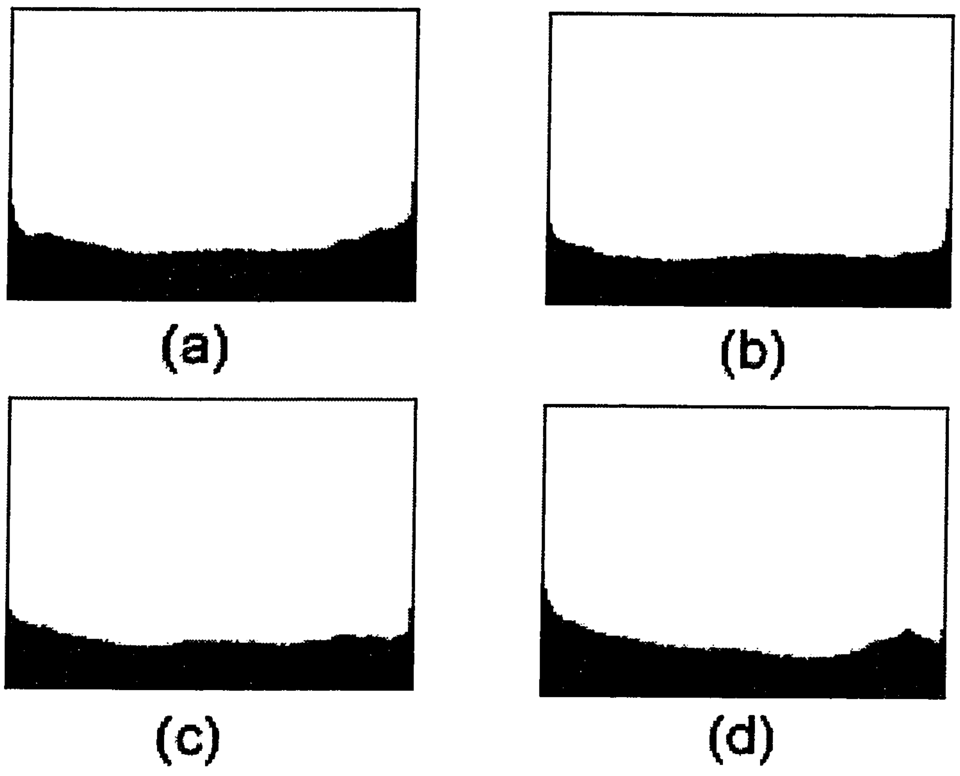


Figure 7: Image histograms: (a) intensity, (b) red, (c) green, (d) blue

6	7	8
5	9	1
4	3	2

Figure 8: Designation of the image pixels in a 3X3 operator window.

well. If we want local gradient operator to handle noisy images, we have to use larger operator windows, which will lead to higher computing costs.

Canny's edge detector

Canny [29] has taken an analytic approach in designing gradient operators. Edge detection is performed by convolving a continuous domain, one-dimensional noisy edge signal f(x) with an anti-symmetric impulse response function h(x), which is of zero amplitude outside the range [-W, W]. An edge is marked at the local maximum in gradient magnitude of Gaussian-smoothed image, which is the convolved gradient $f(x)*h(x)$. The impulse response h(x) is chosen to satisfy several edge finding criteria, such as good detection, good localization and single response for one edge. Canny's method often produces better result than simpler standard local gradient operators do. Deriche [30] provided a simple efficient implementation to approximate Canny's operator.

Second order derivative edge detection

Second order derivative edge detection employs some form of spatial second order differentiation to accentuate edges. An edge is marked if a significant spatial change occurs in the second derivative. Laplacian-of-Gaussian (LoG) filter, a.k.a. Marr-Hildreth filter, is an edge operator that combines a two-dimensional second derivation of the gray value function with a noise-reducing smooth by a Gaussian low-pass filter. Edge pixels are detected at zero-crossings of the final convolution result. LoG filter is well suited for detecting thick edges, which require large operator windows.

Detection using edge models

Image data can be matched to ideal parameterized edge models. If the match is sufficiently accurate at a given image location, then an edge is assumed to exist with the same parameters as the ideal edge model. These methods are generally used in relatively large image windows. An example of such model is straight edge model, which is fully defined by line parameters, as introduced by Hueckel [31].

Color edge detection

The gradient operators described above are designed for gray-level images. Several definitions of edge [34, 35] have been made to extend them to deal with color images. We can define edge on just one color component space, this will ignore the information on other components; or we can perform an OR operation on, or take the vector sum of, the edge maps of all color components. The problem with the OR operation is there may be too many edges or edges found may be too thick; while in case of vector sum, gradients in components may differ in different direction and thus make the overall gradient null.

It is desirable to design a color edge detector directly in the color vector space. An edge detector [36] can be defined as $D = \| X_{max} - X_{med} \|$, where X_{med} is the median color in the neighborhood window of x, and X_{max} is the color in the x neighborhood that has the biggest distance to X_{med}. However, this detector is very sensitive to impulse noise. Or we can use other forms similar to this to reduce the impact of noise, such as $D = min_j(\| X_{n-j+1} - X_1 \|), j = 1, 2, ..., k, k \leq n$ n is the size of the window. This will help reduce the effect of noise but unfortunately there is no formula in choosing the k, and sometimes the edge gradient will be reduced in edges where there is little noise present.

2.3.3 Region-based methods

Unlike edge detection, which finds out dissimilarities in an image, region-based methods try to find out homogeneous regions in an image. Major region-based methods are surface fitting, split-and-merge, and region growing.

Surface Fitting

Surface fitting can be used to achieve segmentation. Pixels in a region are assumed to be able to be fitted by a surface model. Besl and Jain [36] proposed an algorithm that segment a large class of images into regions of arbitrary shape using a piecewise smooth surface model for image data having surface coherence properties. The approximate image data with bivariate functions so that it is possible to compute a complete noiseless image reconstruction based on the extracted functions and regions. This result is impressive for a number of images, but the control structure of this algorithm is quite complex. It is also very computational expensive - segmentation processing time can even shoot up to several CPU hours depending on image complexity and number of surface primitives used.

Li and Shen [37] showed that local moments are related to their surface fitting coefficients by a linear transform. By this property, the surface fitting based feature extraction is converted into a computation of 2D local moments. A recursive method is proposed to compute 2D local moments based on which the surface fitting is implemented. This method is also computationally expensive.

Split and merge

Split and merge is another type of approach to region-based method. The split method for segmentation begins with the entire image as the initial segment. Then it successively splits each current segment into quarters if the segment is not homogeneous enough. There are different methods to define homogeneity. For example, a segment can be considered homogeneous if the greatest distance in color space between two pixels in the segment is small enough. Chen and Pavlidis [38] suggested using statistical tests for uniformity. It requires that there is no significant difference between the mean of the region and each of its quarters.

Because segments are successively divided into quarters, the boundaries produced by splitting often tend to be squarish and artificial. Browning and Tanimoto [39] gave a description of a split and merge scheme where the split and merge is first accomplished on mutually exclusive sub-image blocks and the resulting segments are then merged between adjacent blocks to take care of the artificial block boundaries.

Region growing

Region growing is basically an iterative process of merging neighboring regions into larger regions under certain conditions. It generally has four basic elements:

1. Starting point(s) (seed) of merging: the place (pair of regions) where each merging step takes place.
2. Merging criterion: the homogeneity criterion under which two regions can be merged into one.
3. Merging pair selection policy: decide which neighboring region(s) to merge if there are several pass the homogeneity test.
4. Termination criterion: the criterion when no neighboring region pair satisfies, the merging step stops. In most cases, the termination criterion is just the merging criterion.

Methods differ mainly in how starting point is selected and the merging criterion. Early methods [40, 41] are quite primitive in that they start merging from sufficiently homogeneous small regions, which are found manually or arbitrarily, and do the merging according to the image scan sequence. Later methods [42, 43] find the most similar neighboring regions in the image as the pair to merge. The finding and merging of the most similar pairs are iterated until a termination criterion is satisfied.

There are many different methods in deciding the merging criterion, which is normally a distance measure of adjacent two regions. Kocher [42] defined the distance between two adjacent regions as the sum of square errors between original data and the mean value over the union of the two regions. Gambotto and Monga [43] defined the distance of two regions as the difference of their mean values. The mean value of the new region is then calculated from the previous two merged regions.

Another group of merging criteria is to combine edge information with region growing. The belief is that if merging criteria don't take the continuity of boundaries between regions into consideration, "false" contours at the boundaries of some adjacent regions may be created, i.e., contours are created where gradient magnitude is not high. Brice and Fennema [44] decide that two regions are merged if the boundary between them is weak. The weakness of boundary between two neighboring regions is defined as the ratio of the length of weak boundary (difference of intensity is lower than a certain threshold) to the length of shared boundary. Gambotto and Monga [43] proposed another growing criterion which involved an estimate of the mean gradient over the common boundary of adjacent regions. The decision for merging is made depending on the comparison between the average gradient on the common boundary GC(i,j) and the values of the average gradients GR(i) and GR(j) within the two regions R(i) and R(j).

$$GC(i,j) < F\{GR(i), GR(j)\}$$

where F is an increasing function of both GR(i) and GR(j). So that for a fixed value of GC(i, j), adjacent regions which are very noisy are more likely to be merged than flat regions. Westman *et al.* [45] proposed a region growing method that contains two stages. The first merges adjacent pixels whose difference is below a certain threshold into basic regions. The second step merges adjacent regions whose average boundary contrast is below another threshold. The reason for the first merging step is to form basic segments bigger than just pixels to avoid sensitivity to local variation and noise, thus to achieve some robustness.

Worthy to note is that edge information should be used with caution in region methods. Edge only makes sense when there are more than one pixel in common boundary. By using edge criteria, we may not be able to find out fine structures in the image, e.g. line of one pixel width, which is often necessary in object recognition. Besides, "false" edge is not too much a problem, as we will discuss later.

2.4 Our method

When we design our segmentation method, we have to take our particular task into consideration. As already pointed out, pixel clustering method is not a good choice for the complicated road scene that we are dealing with. Edge detectors are more or less subject to noise disturbance and have difficulty generating coherent boundaries due to the difficulty in edge tracing. Besides, most edge operators are defined on gray-level and can not be easily extended to color images. For edge operators defined on color vector space, they don't have effective low-pass filter to work with them. As far as region-based methods are concerned, surface fitting is computationally too expensive, while split-and-merge methods tend to generate rigid unnatural edges. It turns out that region growing methods provide the best result in extracting region information, and edges are formed quite naturally.

2.4.1 Fast region growing

The region growing algorithm used in this work is as follows:

- Step 1: Every pixel is treated as a starting region
- Step 2: Two neighboring regions, which are most similar to each other in the whole image, are merged into one. Similarity between two adjacent is measured as the color distance between the mean color values of the two regions.
- Step 3: Go to step 2, until the minimum distance between any two neighboring regions is larger than a threshold.

The above algorithm is simple, yet is very powerful (Fig. 11). Here we would like to discuss the alleged shortcomings of region growing. There is the false boundary problem [46] as following:

1. True boundary in the image is not found.
2. New boundary is created where there is no real boundary in the image
3. A boundary is created for a real boundary but the location found is not accurate.

While problem 3 is not often observable and less serious, we don't observe problem 1 in our method if the threshold is chosen to be sufficiently low. Since each region is created within a color difference threshold and the merging always starts from the smallest color difference, the boundary between two adjacent regions will be produced quite naturally, and this is actually one of the strong points of region growing method over ordinary edge detection methods. However, in many cases problem 2 is just unavoidable for region methods. There are cases where two adjacent regions don't have a clear-cut common boundary. We either create a "false boundary" at an appropriate place, even the boundary is weak; or we have to merge these two regions into one. Edge and region are two different concepts and they don't always match each other perfectly in an image. The false boundary is the side effect of the attempt to prevent the chaining effect; and we use the mean color value to enlarge the difference between two regions in case they are connected by some pixels whose color is the middle (a slow ramp). Chaining effect can be seen in Fig 10 where gradient

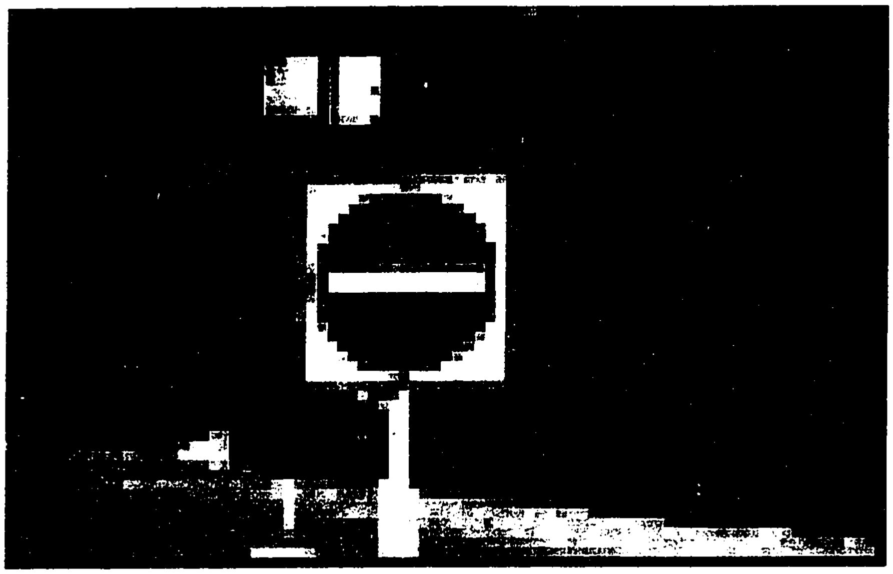


Figure 9: Original image, enlarged.

between pixels are used as merging criterion. The lower-left corner of the image lost its detail because regions are "chained" together. As we shall see later in object detection, some extra edges are not a big problem, at long as the critical information has been preserved.

However, this method is very slow. Because the mean color value over a pixel can change due to a merge, after each merge we have to recalculate all the distances between any two neighboring regions to find out the next merging candidate. Suppose there are N pixels in an image, the merging step is of $O(N)$. For each merge, we have to do $O(N)$ calculations to update the distance values. The total cost will be $O(N^2)$ of color distance calculation. With a normal road image of 600×400, the computation can take over one hour on a SUN Sparc 2.

To speed it up, we modify the step 2 in above algorithm to:

- Step 2: Two neighboring regions, which are most similar to each other in the whole image, are merged into one. Similarity between two adjacent regions is measured as either the distance between the mean color values of the two regions, or the smallest distance between any two adjacent pixels on the boundary of the two regions; whichever is larger.

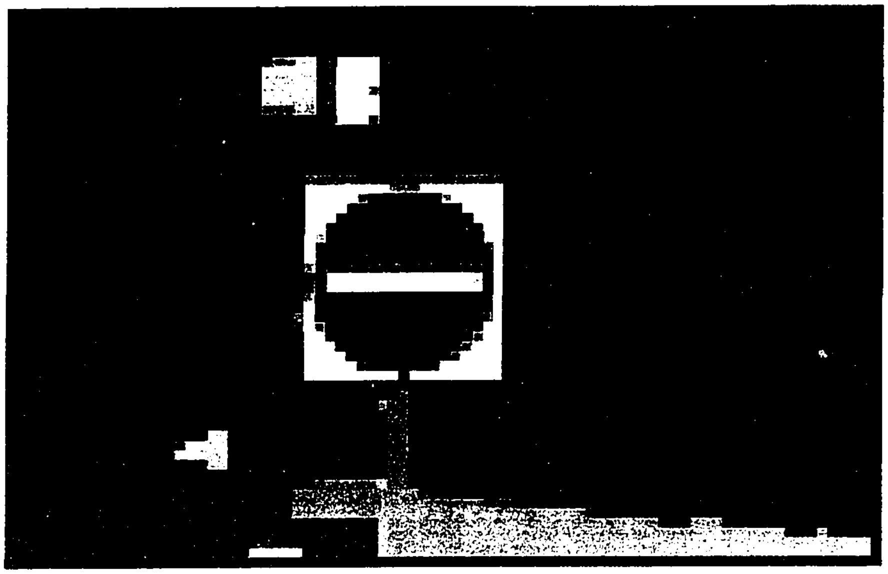


Figure 10: Region growing using pixel gradient merging criterion.

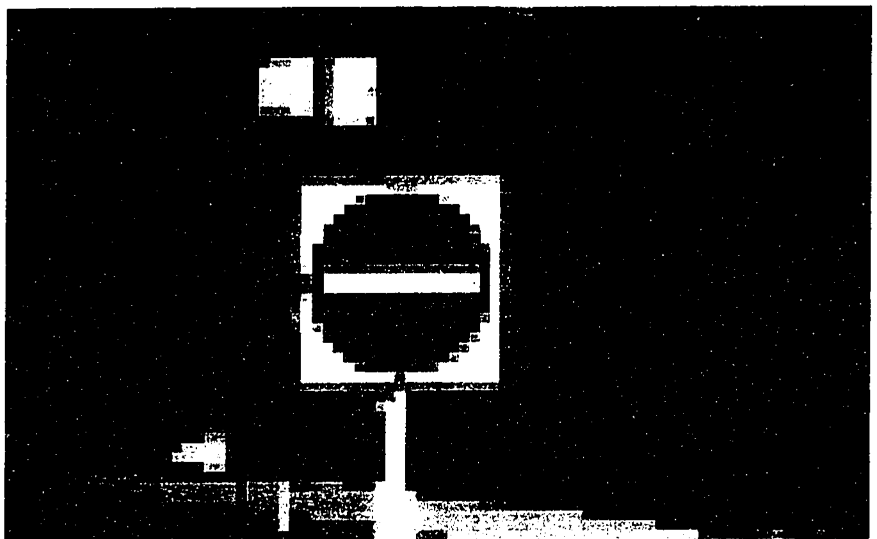


Figure 11: Region growing using mean region color value.

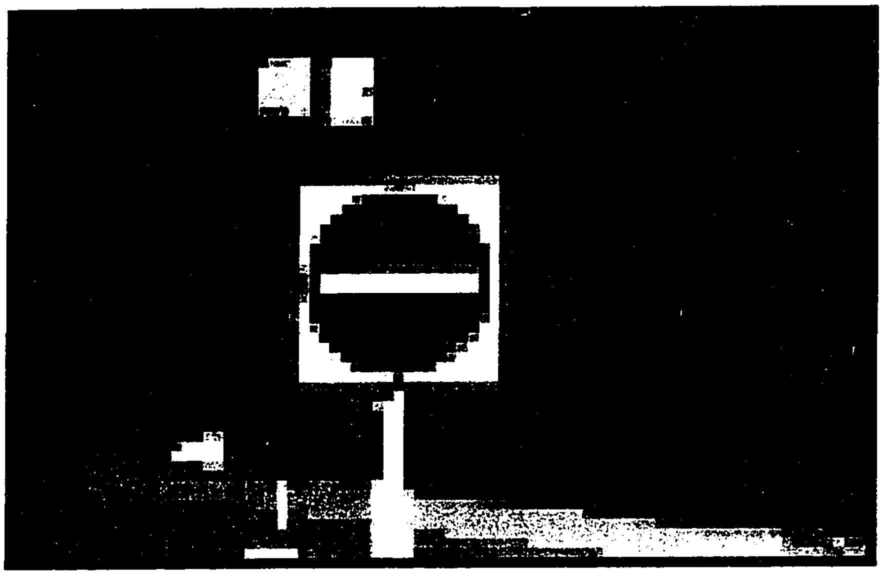


Figure 12: Region growing using approximated mean color value criterion.

It's a good approximation of previous algorithm (Fig. 12), and is very computationally efficient. For each merging iteration, we don't have to perform the expensive calculation to find out the smallest distance between neighboring regions in the whole image any more. The computation cost is $O(N)$ to generate the distance list, and $O(N)$ to perform the merging, totaling $O(2 \times N)$, much smaller than $O(N^2)$. Considering that N can be fairly large, the difference is huge. For the same image of 600 $\times$ 400, it only takes a few seconds to finish the segmentation. The actual implementation of the algorithm as follows:

1. Create a list of adjacent pixel pairs, listed in ascending order of the color distance between the pixels in each pair. So each pair in the list is positioned with a so-called current distance value.
2. If the list is not empty, fetch the first pair of pixels; otherwise go to 7.
3. For the pixel pair fetched, if the current distance of the pair is greater than a threshold, go to 7.
4. If the pixels are in one region already, remove the pair from the list. Go to 2.
5. Calculate the distance between the pixels using mean color of their respective regions. If the distance is greater than the current distance of the pair, update

the current distance to the new one and move the pair up to the appropriate position in the ordered list. Go to 2.

6. Merge the regions containing the pair of pixels into one. Go to 2.

7. Stop.

2.5 Preprocessing

Our segmentation algorithm works beautifully until it meets nasty pictures (Fig. 13). The segmentation result is somewhat disheartening (Fig. 14). Our method is designed to treat every pixel equally, even if a pixel is a noisy intruder. However, the noisy points are very different from normal signal and the region growing method will not merge them into neighboring regions, and consequently, the segmentation result is very fragmented. Noise is ubiquitous due to the equipment limitation, the resolution of the object, or any other reason. We have to employ some kind of method to deal with them, and so preprocessing comes into play.

An important preprocessing technique is smoothing filters that smooth out impulse noises while preserve or even enhance edges. There are several smoothing filters, namely, median [47], KNN [48], SNN [49] and SNF [50].

The median filter replaces the center pixel in a square window with the median gray value of the pixels in the window. K-nearest neighbor (KNN) mean filter substitutes the center pixel in a square window with the mean value of the K first pixels nearest to the center pixel. Symmetric nearest neighbor (SNN) uses the mean value of half the number of pixels in the neighborhood by selecting, from each pair of pixels located symmetrically on oppose sides of the center pixel, the one closer in color to the center pixel. Symmetric Neighborhood Filters (SNF) is an improved version of SNN, which determines for each pair whether the center pixel is the edge on either side, i.e., if the center is an outlier (ridge, valley) with respect to the pair. If it is, then this pair will not contribute to the average of the center. By this improvement, the filter can preserve the lines with one bit width.

As stated, the purpose of smooth filter is to enhance the edge while smooth out the interior of a region. A major concern with our segmentation task is to get rid of impulse noise while keep the fine structure, such as one pixel width lines, as needed in object recognition (especially to recognize objects with low resolution, say 12x12 pixels in size). SNN is not good at preserving the fine structure, while SNF can. However, SNF is not so powerful in getting rid of impulse noise, such as singular pixels. Under the criteria we set here for smooth filter, 1-NN gives the best result

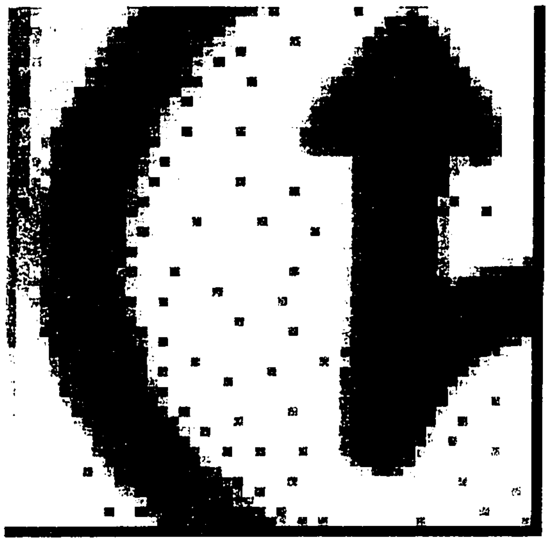


Figure 13: A enlarged portion of a noisy sign.

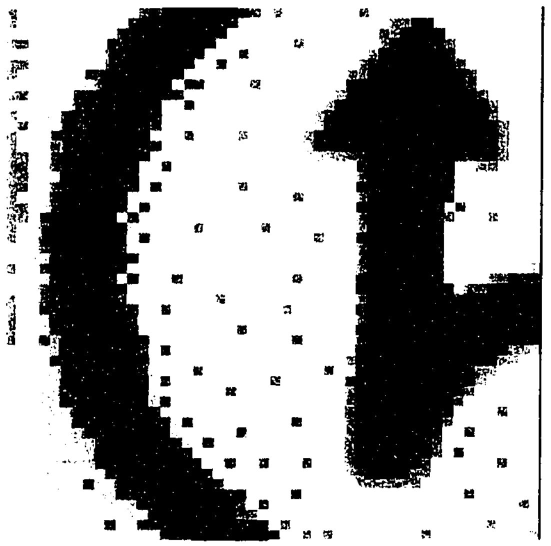


Figure 14: Segmentation of noisy road sign without filtering.

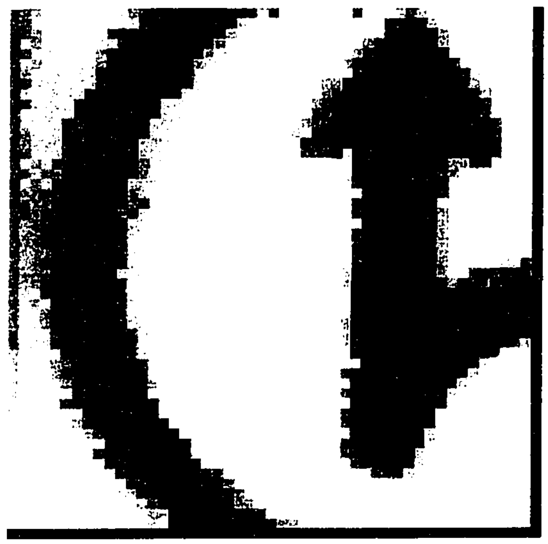


Figure 15: 1NN filtering result after two iterations.

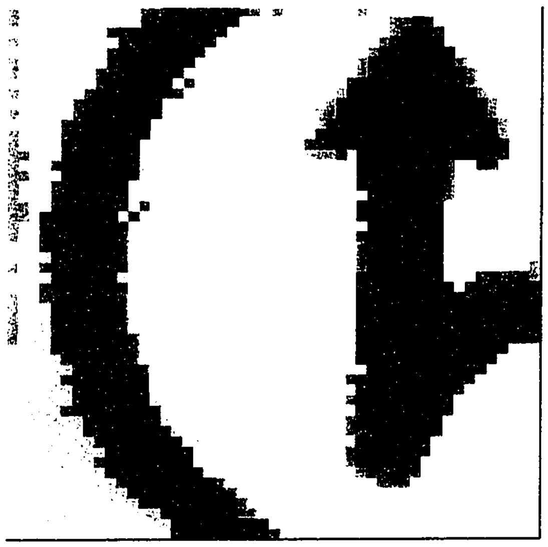


Figure 16: Region growing with 1NN filtering.

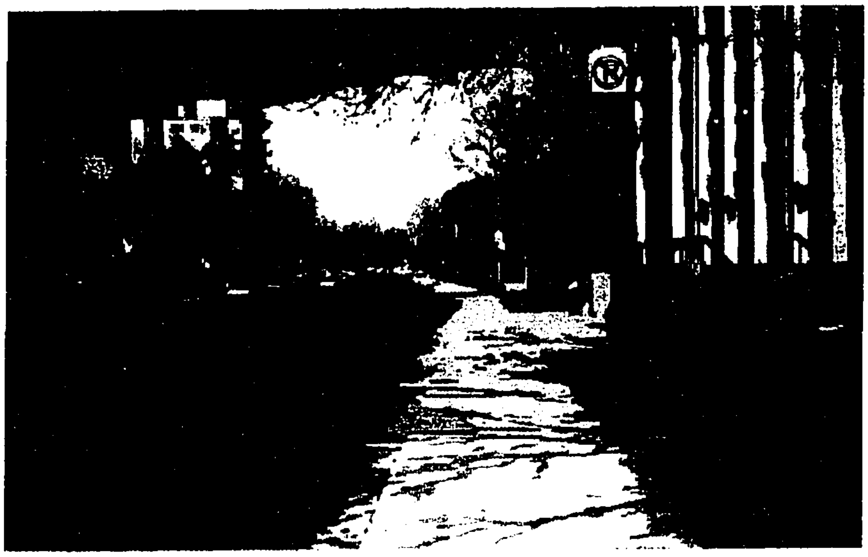


Figure 17: False edges are generated.

in both preserving finest structural detail and smoothing impulse noise. The image usually gets stabilized after a few iterations (Fig. 15). Segmentation improved a lot after smoothing (Fig. 16).

2.6 Further considerations

There are several possible aspects of enhancement in our segmentation method.

The first is the issue of false edges generated by region segmentation. Although region-growing method can generate boundaries quite naturally, it does generate spurious edges in some slow ramp areas when it tries to separate two regions apart. Fig. 6 shows an image with sky and road. The color of sky changes from white to blue starting from the horizon. However, edges are generated in the sky. Similar effect can be seen on the road. Edges are generated at places where there are no real edges (Fig. 17).

If accurate edge information is really needed from the segmentation, one way to solve this problem is to do region growing first. Region growing will generate natural boundaries along with some false boundaries. We can then trace along the boundaries of all regions and remove those boundary parts that are not strong enough in the original image. The result of the tracing is put into a separate edge map. This means that one can keep two segmentation results, one for region information, another for edge information. This is more natural than expecting one segmentation result to combine two different kinds of physical features into one.

Another issue is the selection of segmentation threshold. Little research has been done on the automatic selection of segmentation threshold. The reason might be it is very difficult. And the reason for it to be so difficult might be that a threshold that is suitable for any application purpose does not exist. Segmentation, as a low-level process, does not have any application level information to guide it in its process. However, we believe there should still be some physical properties that we can utilize to make the threshold self-adaptive to some degree. Before we can do this, it is better to keep the threshold relatively strict to avoid any major loss of object information during segmentation.

3 Model representation

Objects are identified by their features. Object recognition involves identifying a correspondence between certain features of an image and comparable features of a model. Both the object model and the image need to be represented in some way. We have discussed image representation in chapter 2, whereby images are represented by low-level physical features such as edges and regions. In this chapter, we focus on the issue of model representation for the purpose of object detection.

3.1 The criteria

There are some frequently cited criteria for a good model feature representation [51, 52, 53, 54]. They are listed as follows:

Scope and sensitivity: The representation should be able to describe a wide range of objects, while preserving their respective uniqueness.

Uniqueness: A representation should be as unique for one kind of object as possible. With good uniqueness, detection can be more effective and should be able to narrow down to the real objects quickly.

Stability: Small changes in the model produces small changes in representation. A representation should be robust to noise and occlusion.

Efficiency: It should be possible to compute a representation efficiently, and it should also be possible to decide a match between two sets of features efficiently.

To illustrate how these criteria are used in representation design, we consider a simple representation scheme. Suppose we want to detect red stop signs. We can define a representation of a stop sign as a cluster of red regions that fall within a rectangle area of certain size. Here we use a cluster of regions instead of single red region because we want the representation to be robust to noise and occlusion. With this representation, we can go out to search the image for red regions and find stop sign candidates for all red region clusters. Then we can send these candidates to final classification. This color representation scheme can be extended to other road signs with major background colors, and should be able to detect the candidates under good lighting condition. Its uniqueness is not very good because many other objects can also have the same color. The nice side of this scheme is its efficiency. Using this scheme, detection can be achieved to some extend at very low expense.

There is hardly any general feature representation scheme that can be applied to any problem. Rather, it depends on the task at hand and the conditions of external world. That means a representation has to capture any distinction needed to differentiate the object to be recognized, and also exploit invariant properties with respect to changing environment conditions. Representation also depends on the purpose it is to serve. For example, object recognition and object detection may have different needs for model representation; since recognition usually needs complete and strictly unique object representation to identify an object while detection does not.

3.2 Feature selection

To enable the search for a model object in an image, the representations of model and image should be closely related. We can use only low-level features, such as edge and patches, to represent an object. Many systems have been designed to used these simplest shape primitives, including HYPER, which uses polygonal approximations of 2D curves [56], and 3D-POLY, which uses quadric approximations of 3D surfaces [57]. Although these primitives are convenient, they produce unstable descriptions of objects under noise.

Parts are another important group of features that are used in many systems [58, 59, 60]. The basic idea is to use parameterized parts, such as *ribbons, symmetric axis transforms*, and *smoothed local symmetries*, to construct different objects. These parts are especially useful in line drawings and industrial parts where one pattern can appear repeatedly in different places with different orientation and scales. But this is not quite the case for our problem, where parts are not usually duplicated.

Another type of feature is called distinguished features. These features are not complete description of n objects, but they capture the important information that is useful to identify the objectss. This is especially true for road signs, which are designed specially with distinguished features for human's easy and quick recognition. Consequently, we can take advantage of these features in sign recognition. Examples of distinguished features are colors, color combinations, color histogram, area, compactness, etc. Most of these features are global features that can be computed automatically. However, these features are just partial description of signs; further recognition is usually needed to finally recognize a road sign candiadate.

In some cases, distinguished features are designed specially to represent a very restricted class of objects. Saund [55] demonstrated how knowledge can be used in

Figure 18: Semantic significance of a component is not in proportion to its physical significance.

designing the features to be used in recognition. He used a vocabulary of approximately thirty features to describe fish fin profiles. Each feature captures a fragment aspect of fin shape, and together they provide the representation that can be used to distinguish many fish fins. In another example, Gordon [61] represented faces with selected distinguished features such as eyes, nose, cheek and their geometric relations. These are quite interesting ideas in that they introduce higher level features as the distinctive factor in recognizing an object. In this work, we use this idea to organize features of a road sign model into hierarchy structure to achieve better recognition performance.

3.3 Hierarchical representation

A road sign is made up of different (color) components. The sum of these components constitutes the semantic meaning of the sign. The semantic significance of a component is not in proportion to its physical significance. For example, in the sign shown in Fig. 18, the arrow is more important to the white corner. It is better to label the three different color components as equally important, regardless of their actual size in the sign. This kind of information is very helpful to recognition. A recognition method is not reliable if it uses primitive features only.

This consideration leads us to building a hierarchical representation of object model, based on the concept that an object is composed of different components. This hierarchy can be multi-layered. An object model contains components, and these components can contain their own sub-components. An object can thus be represented in a hierarchical tree of component features. The tree root is the final representation of the object, while tree leaves are the primitive features that are most closely related to the features extracted from the image. The tree arcs represent the geometry constraints for lower-level components when they form higher-level components. Examples of

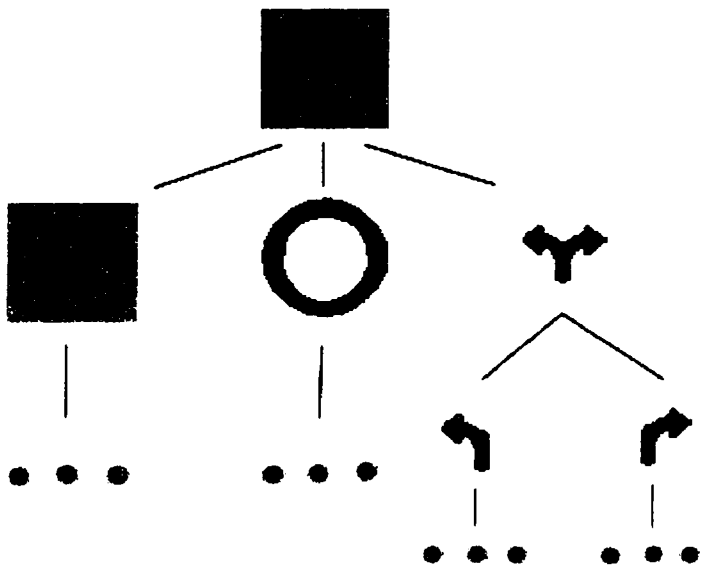


Figure 19: An example of representing a road sign features in hierarchy.

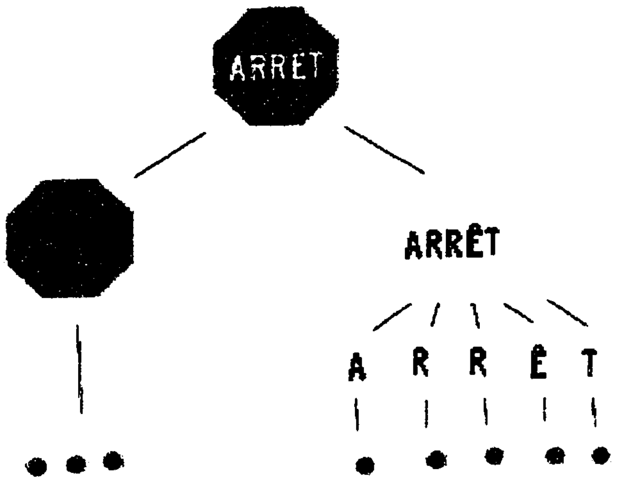


Figure 20: Another example of representing a road sign features in hierarchy.

such tree structure is shown in Fig. 19 and Fig. 20.

Each node in the tree has a number of features that describe its overall information. These features are called global features of the node. On the other hand, the composition of the child nodes is called the mode's local feature. For example, a road sign can have area, overall shape and possible location as its global feature, and have its different color components as its local feature. For each component, it can have global features like compactness, relative location to the whole sign, centroid etc, have edge and region segments as its local feature. If we go further, each edge segment has direction, length and location as its global feature, and particular pixels (maybe represented in chain code) as its local feature.

Using the tree structure has another major benefit. Since features of different scale are organized in different layer, object detection method can use this structure to model the search for object candidates. Higher level features are global features relative to lower level details. Before a method starts out finding lower level features, it can first check if the global features are present to decide whether further detail matching is worth doing.

This feature structure does not conflict with the ordinary flat one-level feature representation structure. Flat feature structure is just a special case of hierarchy structure, in which case the feature hierarchy tree is one-level deep. For the convenience of discussion, we will use hierarchy structure for the rest of the paper. Algorithms based on the hierarchy feature structure can be readily applied to ordinary one-level representation schemes.

3.4 Feature matching

To use a feature representation for object detection and recognition, a matching quality measure has to be defined to tell how well a model has been matched.

3.4.1 Some common matching models

Given an image and an object model, both represented in terms of their features, we want to locate the subsets of image features that match to the model features. The match decision is based on a matching quality measure, which is normally an error model based on how the image features differ from the location predicted by object model features. Some common error models are [2]:

- *Bounded error model:* matching quality is measured as the number of matching primitive features. A model feature is considered matched if there is an image feature within a close neighborhood of its predicted location.
- *Gaussian error model:* assumes that image features are distributed normally and independently about their predicted location. The matching quality considers both the number of matching features and the sum of the squares of their normalized errors.
- *Bayesian probability model:* Assume features are independent, given such factors as the prior probability of each object if present in the image, the prior distribution of its pose if present, the conditional probability that each model feature is matched if its object is present, and the conditional distribution of matching errors described by an error model, one can estimate the posterior probability that a particular object is present with a given pose and a given set of feature matches. This posterior probability serves as a match quality measure.

Bounded error model is straightforward and intuitive. Although Gaussian error model is supposedly to be more accurate, we find it somewhat unnecessary. It's hard to give a meaningful error measure to location difference, especially when the features are formed empirically rather than analytically. The meager contribution of measuring location errors can hardly justify the computation cost it incurs. Better matching result can be achieved more efficiently by other classification models. As far as Bayesian probability model is concerned, we think it is more of theoretical value.

3.4.2 Our matching model

We extend the bounded error model to suit our hierarchical feature representation. We call the extended model the *average error model.* The matching quality of a component is defined as the average matching quality of its child components. A model component is considered matched if there is such a component present in the image within a close range of its predicted location. If it's not matched, its matching quality contributed to its parent will be 0. Suppose for a component F, it contains child components $f_i, i = 1..N$, the matching quality of component f_i is P_i, then the matching quality of F is:

$$P_F = \sum_{i=1..N} P_{f_i}/N$$

The stableness of hierarchy structure is reflected in the averaging factor of a component. The effect of any change in lower-level features is contained inside one feature and thus will be restricted. In case of lowest level of features in model, its matching quality is usually either 1 or 0, i.e. matched or unmatched. Note that when the feature tree is one level, the matching model is exactly the same as bounded error model.

Normally this model should suffice. But sometimes it may be very convenient to address the different semantic importance of features in a model. To address this need, we introduce a model called *weighted error model.* In this model, every child component is assigned a weight value between 0 and 1. The sum of weights of all child components under one parent is 1. The matching quality of a parent component is the weighted sum of the matching quality of its child components. Using the previous notation, each child f_i has a weight w_i, then the matching quality for F is:

$$P_F = \sum_{i=1..N} w_i \times P_{f_i}$$

We can further extend the weighted error model to make it more expressive. For example, we may need to say that the presence of several components is vital to their parent component. The parent will not be present if any of its vital children is missing. This pivotal child concept can be quite handy to object detection. For example, we want to express that the arrow in the turning sign (Fig. 18) is a pivotal component. If the arrow in missing, even though we have detected perfect match for other components, the sign is still considered not present, and we will drop it as a sign candidate. To address the existence of vital feature, we modify the weighted error model as follows:

The matching quality of a component is measured only in 1 or 0, i.e. matched or unmatched. If the weight sum of its child components is higher than a matching threshold, the parent component is considered matched and its matching result is 1, otherwise, it's unmatched and the result is 0. Suppose the matching threshold for F is T, then the matching result of F can be defined as:

$$P_F = \begin{cases} 1 & if \ \sum_{i=1..N} w_i \times P_i > T \\ 0 & if \ \sum_{i=1..N} w_i \times P_i \leq T \end{cases}$$

By setting appropriate values for component's weight and matching threshold, we can use this model to represent the notion of important features. Suppose there are N components under a parent, not losing generality, suppose the first n components are indispensable to the parent. The weights can be assigned as follows:

$$w_i = \begin{cases} 1/(n+1), & if\ 1 \leq i \leq n+1 \\ 1/((n+1) \times (N-n)), & if\ n+1 \leq i \leq N \end{cases}$$

Then we set the matching threshold for the parent higher than $n/(n+1)$. Consequently, the parent will not be present if any of its pivotal children is missing, and detection can perform more efficiently by searching for the vital features first. The matching score will fall below the threshold if one of the vital features is not found.

3.5 Generating sign model representation

For many road signs, two-layer tree usually suffices to represent the features. The first layer is consisted of connected regions of one color. The second layer is composed of edge segments describing each of the connected regions in the first layer. If we really stretch it, there is a third layer, which contains the pixels that describe each line segment. Usually, these pixels are represented in chain code for each edge segment.

In this work, a road sign model is represented by the following feature structures:

- *global features*
  - *sign area mask* : a pixel mask depicting the pixel locations for the sign's visible area in the sign's bounding rectangle box. This is necessary for signs that are not rectangular.
  - *shape of bounding box* : the width and height ratio of bounding box
  - *color histogram* : the composition of its color elements.
- *local features*
  - *threshold* : the matching threshold of the model.
  - [component's weight] [1]: the weight for each constituting component. If not specially assigned, the weight of a line will in proportion to the area of the component.
  - [*component*]: components that form the road sign.

Each of the first layer features, namely, regions of the same color, contains the following elements:

[1][..] stands for a array of enclosed elements

- *global features*
  - *area* : the ratio of the component's area to the area of the whole sign.
  - *compactness* : the ratio of the component's area to the area of the bounding box of the component.
  - *location* : location of the bounding box of the component, measured relatively to the origin (upper-left) point of the sign. This also shows the bounding box shape information.
  - *centroid* : the center of the mass of the component relative to its bounding box.
  - *color* : the average color of its content. If color is not applicable to this component, use its intensity value.
- *local features*
  - *threshold*: the matching threshold of this component.
  - [*line segment weight*]: the weight for each constituting line segment. If not specially assigned, the weight of a line will in proportion to the length of the line.
  - [*line segments*]: line segments that compose the boundary of this component.

The second layer features are line segments which are represented as:

- *global features*
  - *direction* : the direction of the line. Boundaries of regions are traced clockwisely. The line direction is conformant to the tracing direction.
  - *location* : location of the bounding box of the component, measured relatively to the origin (upper-left) point of the sign. This field also represents length information since that the diagonal length the bounding rectangle stands for the length of the line.
  - *color* : the average color of its pixels. If color is not applicable to this component, use its intensity value.
- *local features*
  - *threshold*: the matching threshold of this component.

- [*edge pixels*]: edge pixels of this line segment.

Given a perfect road sign model image, all these features representations can be computed automatically. If the model image is noisy, we do color classification based on how many different colors should exist in the model, or the colors of interest can be pinpointed by human. Human interaction is required to assign different weights to different component and set the matching thresholds, if it becomes necessary. Otherwise, default values are assumed for weights and thresholds. For first layer features, their default weights are set to the same value. For second layer features, their default weights are set according to the their respective geometric significance (edge length, in our case).

In feature matching models, the matching quality measure is calculated based on matched image features. An image feature has to be in the correct location to be considered matched to a model feature. Usually an error tolerance is given to allow some variance.

4 Object detection

We divide the whole road sign recognition task into these two steps: road sign detection and recognition. Detection finds out road sign candidates in the image, while recognition uses whatever classification method to verify whether the detected candidates are instances of the wanted road sign. The assumption is detection can use partial information of a road sign to find out sign candidates at relatively low cost, and then more computationally expensive classification method can be used to reach final recognition decision. The main purpose of doing so is to optimize system performance.

There is a large free space for designing a detection strategy. It all depends on how much information about the object model is used in the process. For example, a road sign detector can be as simple as a rectangle generator, using no constraints from the object at all. It sequentially generates rectangles of certain size at different location of the image and feed these rectangles as bounding boxes of sign candidates to final recognition classifier. A detector can also be very sophisticated and powerful, using all the feature constraints of a sign to locate the real occurrences of a sign in an image, making a final classifier redundant and unnecessary. Again, the goal is to balance the workload between detector and classifier to achieve maximum system throughput.

4.1 The simplest case

As mentioned, the simplest case in road sign detection is a rectangle generator. Since road signs are rigid objects, we can imagine a bounding rectangle box around such a rigid object. Two rigid objects can be matched against each other using appropriate template matching method after their bounding boxes are normalized to the same size. Since any transformation to a rigid object can be defined by three parameters stand respectively for translation, scaling and rotation, the transformation can be determined by three corner points of the sign's bounding rectangle transformed version.

We can check all the possible transformed rectangles in the scene image to see if the content in any of them matches the object model. Assuming there are n points in the scene, t is the time to match two templates, the computation cost for testing all the coordinate possibilities is of $O(n^3 \times t)$. If rotation is not allowed for the sign occurrences in the scene, we only need two points to determine a bounding rectangle. In this case, the computation complexity is of $O(n^2 \times t)$. This is almost the case in our project, where road signs are usually posted vertically without much rotation.

Figure 21: A street scene.

How bad can this detection scheme be? For a typical image of 600 $\times$ 400 pixels, the computation for the whole recognition is in order of $5.76 \times 10^{10} \times t$. This computation cost is too expensive to be acceptable in real system (at least for now). By using constraints from the incoming image and the road sign model, we can reduce the computation cost significantly.

The first performance enhancement comes from segmentation. In segmentation, pixels are grouped into larger units based on their physical properties. This will reduce the basic units in an image by a large amount. For example, for a street scene image of 600 $\times$ 400 pixels (Fig. 21), segmentation generates 2643 regions (Fig. 22). It is reasonable to assume the boundary of a road sign instance in the image is reflected in the boundaries of a subset of these regions. Any two regions can form a bounding box for a sign candidate. As a result, the computation to test all bounding boxes is $2643^2 \times t \approx 7 \times 10^6 \times t$, which is much smaller than without segmentation.

Segmentation reduces searching space without using any information from the road sign models. However, real detection happens when it is guided by model features. For the rest of this chapter, we will discuss detection methods that use model features in pruning the search space.

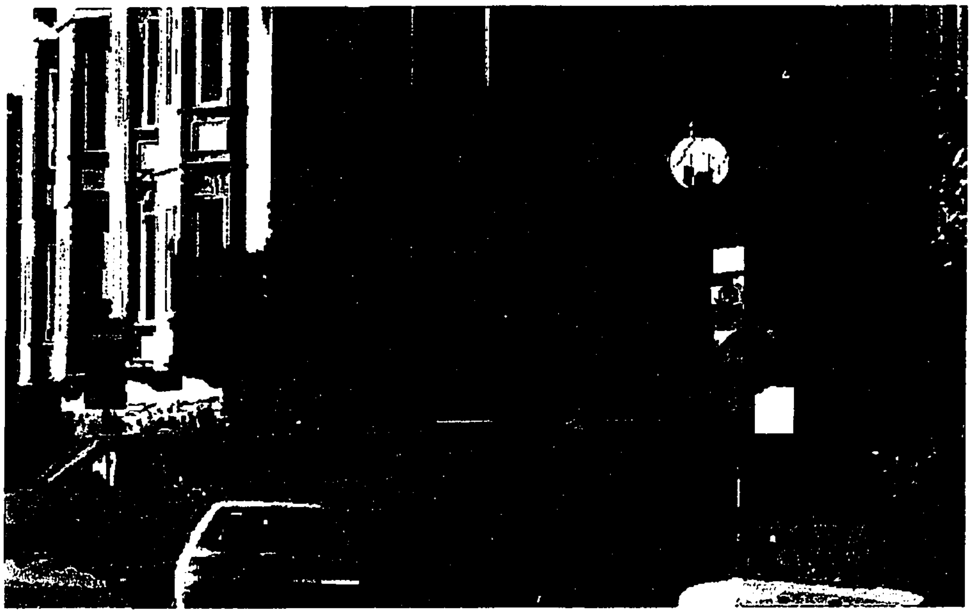


Figure 22: Segmentation of the scene.

4.2 Basic search techniques

Detection involves finding correspondence between a subset of image features and the model features. There are two problems to be solved: 1. What feature should be used? 2. Given the image and model features, how the correspondence can be found? We have discussed feature representation in Chapter 3. In this chapter, we will decide how these features are to be used in detection. But first of all, let's study the problem of how to find correspondence, from the perspective of a pure search problem.

A model is usually composed of a set of smaller entities, each with its own properties. Together, these entities form the object by obeying some kind of geometric and photometric constraints. Suppose we know how to find occurrences of the smaller entities in the image based on their properties, and we also know how to check the constraints among these entities, the question is: how can we find the object instances in the image? This is a pure search problem, in that we don't care what the features or constraints actually are.

There are several surveys concerning object detection and recognition in the literature [2, 62]. Detection methods can be classified in terms of the space in which a match is found [2], such as correspondence space, transformation space, or both. Correspondence space is the space of actual matches between image features and model features. Transformation space is the space of necessary transformation in order to match an image feature to a model feature.

4.2.1 Search in correspondence space

Two typical examples of correspondence space search are *tree search using constraints* and *Graph matching* .

In graph matching, searching for feature matching is the task to find a common sub-graph isomorphism between two attribute graphs [63, 64, 65, 66]. Graph nodes represent features, graph edges represent the constraints among them, and the nodes and edges are attributes to stand for properties of features and their relations. The search is to find optimal graph match that matches both topological structure and the attribute values. The exponential search is guided by some matching quality measurement.

In tree search [62], the system tests all possible correspondences between image and model features. It uses a search tree representing the choices of features in interpreting the image features. Starting from the root of the tree, the method checks an additional image feature at each layer of the tree. Branches at each level represent different choices of model features that can be matched to that image feature, plus the choice of matching nothing to it at all. A complete interpretation of the image is achieved by assigning some subset of image features to corresponding model features. All the interpretations are reflected in the tree's leaves. Not all combinations need to be tested because feature constraints can be used to prune the search tree.

Apart from constraints, *branch-and-bound* can also be used to prune the search tree. It uses a function to evaluate a score for a complete interpretation, and an estimator to get the maximum score of any complete interpretation from a partial interpretation. The search is pruned if the estimator shows no higher scores can be achieved than the current best score for a complete interpretation. *Heuristic search termination* [62] is another performance enhancing tactic. It terminates the search as soon as a match that meets certain minimum requirement has been found. Relaxation labeling is also a way to speed up the search while accepting a sub-optimal result [67, 68].

Even with the use of constraint and performance optimization, both methods require expensive computational costs, which is generally exponential in the number of features.

4.2.2 Search in transform space

The typical example of search in transform space is *Hough transform.* An array of bins is set to empty for parameters of possible matches. For each possible match between a model feature and an image feature, parameters with that match is determined and votes are cast for corresponding bin. When all votes have been cast, the array is scanned to find and verify and the parameters that receive the most votes are treated as matching candidates.

Lamdan and Wolfson [69] use a method called *geometric hashing* to achieve detection of object using pre-computed geometric hash tables. Assume a model of an object is defined by m features. For each ordered pair of model features the coordinates of all other $m - 2$ features are computed in the orthogonal coordinate frame defined by this basis pair. Each such coordinate is used as an entry to a hash-table, where the basis-pair at which the coordinate was obtained and the model are recorded. Assume an image has n features. Choose an arbitrary ordered pair in the scene and compute the coordinates of the scene features taking this pair as a basis. For each such coordinate check the appropriate entry in the hash-table, and for every record (model, basis-pair) appearing there, tally a vote for the model and the basis-pair as corresponding to the ones in the scene. If a certain record (model, basis-pair) scores a large number of votes, take it as a matching candidate. If the current basis-pair does not score high enough, pass to another basis-pair in the scene.

The major drawback of this kind of method is that they are not very accurate. In noisy images, it's hard to estimate transform parameters accurately. Besides, a bin may collect a lot of votes that not at all consistent, or votes from random clusters may overshadow a real solution [73]. Furthermore, the computation of these methods is still very expensive and memory demanding. All possible feature combination and parameters/coordinates have to be calculated to get the entire match vote.

4.2.3 Search in both space

Alignment method [70] was the first method that offers a solution in polynomial time complexity. It first aligns the model with an image using a small number of pairs of model and image features. Then the aligned model is compared directly against the image by mapping the object model into image coordinates. This requires three point features to determine an alignment transform for rigid object, only two point features if the object's orientation does not change. This method relies on at least one set of anchor features to produce a sufficiently accurate estimate of alignment transform. Efficient computing and verifying transformations is also vital to the success of this method.

An error in localizing the anchor feature produces error in alignment transform estimation, which leads to error in matching additional projected features. Methods [74, 75, 76] have been suggested to overcome the inaccuracy of the initial estimate by refining it iteratively with newly found additional features.

Another way of combining two spaces together is to first use transform space, such as a coarse Hough transform, to identify interesting regions of transformation. Then within each region, perform a correspondence search considering only matches that are consistent with the range of transformation [71].

4.2.4 Choose a method

With the above analysis of search methods, we can see the most robust way of detection is tree searching. It tests all matching choices exhaustively. The combination for matching test grows exponentially with the number of features and is computationally very expensive. For this reason, it is critical to use techniques that select subsets of the image likely to have come from a single object before establishing a correspondence between image and model features [72], namely, use it in places where the number of features is small; otherwise, the cost is not justifiable.

We also see that search efficiency can be greatly enhanced if we can make some assumptions about the object and some of its features. If we can assume that certain part of the object has been found and an alignment transform from the model to the image can be estimated, then the search problem becomes a mapping problem. It is polynomial in computation complexity. Suppose there are M features in the image. If rotation is not considered, we need at most two features to decide an alignment transformation. The computation cost is $O(M^2 \times t)$, where t is the time to verify a transformation, i.e., to match the transformed model to the corresponding part of image. However, the method fails if no anchor feature is present.

There are not many other choices for basic search methods. Based on the methods we discussed above, the choice is a tradeoff between efficiency and robustness. Generally speaking, road signs are objects of distinguished features (section 3.2). There are many assumptions one can make about a road sign candidate. It would be fairly unnecessary not to use any of these assumptions. Besides, road signs are rigid objects to which shape constraints and alignment projection can be applied. For this reason, we design our detection method based on feature alignment, with several enhancements aimed to make it more robust and efficient.

Apart from the tree search and alignment methods, there are transformation space detection methods such as geometric hashing. Geometric hashing is not very useful if only primitive features are used. Too many random casts will overshadow the real candidates. Its effectiveness can be enhanced if distinct features are used that can alienate the real object candidates from others (such as "super-segment" suggested in [81]). In the sense of finding out enough evidence (anchor features) to make location estimations, this is equivalent to alignment method.

4.3 Sign detection by alignment

To use feature alignment in detection, we need to find out anchor feature(s) of a sign model. An anchor feature is about two known points with fixed position in an object (assuming the object does not rotate). When such a feature is found in the image, an alignment transform can be estimated to map the object model to its projected region of the image. Feature alignment is really about finding anchor features and verifying the alignment transformations. In this section, we will discuss how to define anchor features in the model and how to find anchor feature instances in the image. We will discuss transform verification in section 4.4.

4.3.1 Defining anchor feature

The criteria in selecting anchor feature are basically the same as has been stated in section 3.1, namely, easy to find, robust, distinct, and general. The major requirement for an anchor feature is its reliability, since alignment transforms depend on the detection of anchor features and correct transform estimation. However, there can be multiple anchor features for one object. An object instance can be detected if at least one anchor feature is found. If multiple anchor features are used, the reliability requirement for each individual anchor feature can be relaxed, while the general reliability of the anchor features as a whole is still required to guarantee the detection.

Primitive features such as edge and regions are not stable enough to locate two fixed points in an image. Under the noise level of normal street environment, there are hardly any stable prominent points in a road sign. As has been pointed out in section 3.2, edges and regions can be easily perturbed by noise and become discontinued or fragmented. As a result, we look for anchor features in higher order features.

A natural choice of anchor feature for a road sign is the outer bounding limits of its color components. Since a component's location in a road sign is fixed, once we find out the bounding limits of a component, we can estimate the location of the sign. The bounding limit of a component is decided by the four edges of its bounding box. If

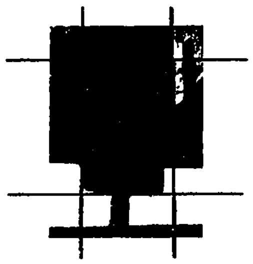


Figure 23: An anchor feature of a road sign.

the shape of component (ratio between width and height) does not change, then any three out of the four edges of the bounding box can decide the component's bounding limit.

Suppose there are M components in a sign. For a particular component $C_k, k = 1..M$, its bounding limit is decided by the four edges of its bounding box, say $E^k_{top}, E^k_{right}, E^k_{bottom}, E^k_{left}$. Now we group the edges from all components into four edge sets, namely, $E_{top}, E_{right}, E_{bottom}, E_{left}$. Since the locations of components in a road sign are fixed, we can determine a sign's location by any four edges coming out of the four edge sets, i.e., $E^k_{top}, E^l_{right}, E^m_{bottom}, E^n_{left}, 1 \le k, l, m, n \le M$. The number of edges required can be reduced to three if the shape of a sign does not change. An anchor feature example is shown in Fig. 23.

A group of anchor features can be defined using combinations of edges from the four edge sets. Name the anchor feature set as F:

$$F = \{F_{klmn} | F_{klmn} = (E^k_{top}, E^l_{right}, E^m_{bottom}, E^n_{left}), 1 \le k, l, m, n \le M\}$$

The size of F is M^4. If we use three edge combinations for F, the size of F is $4 \times M^3$. For a road sign with two components ($M = 2$), the size of F is 16 (four edges) or 32 (three edges). This set of anchor features is very reliable. The underlying assumption is that if a road sign is visible, then at least some of the outer limits of its components are visible. This assumption is true even in very difficult situations, e.g. if there is a lot of noise or the sign is partly occluded (see Fig. 24 and Fig. 25).

For the convenience of discussion, we call such anchor features as bounding features.

4.3.2 Candidate detection

An incoming image is decomposed into regions after segmentation. A road sign candidate in an image is composed of regions that contain the colors of the vital

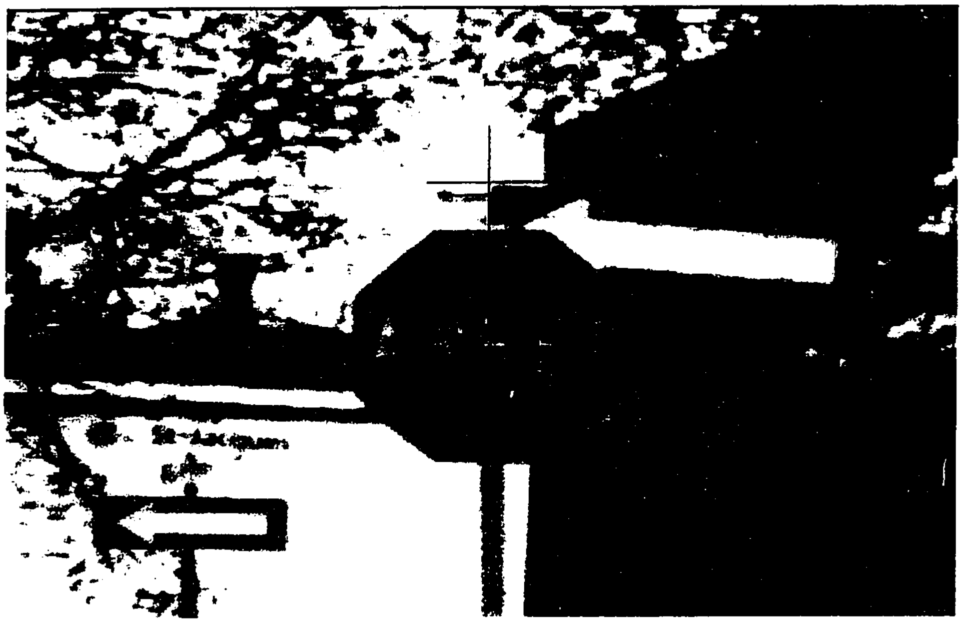


Figure 24: Our anchor feature works fine even in difficult situations.

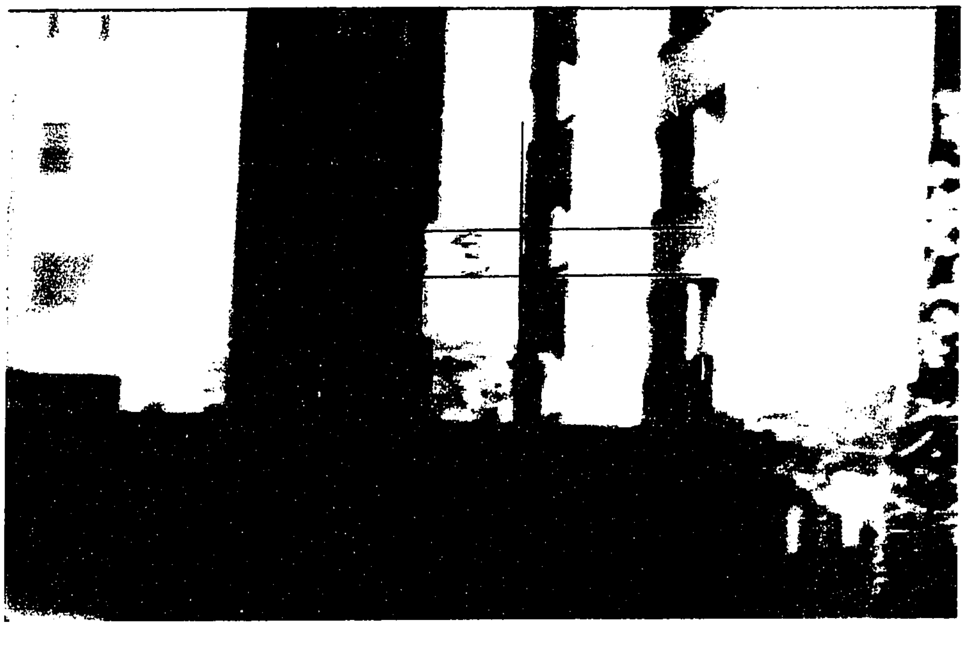


Figure 25: Another difficult situation.

components (section 3.4.2) of the sign. The sign's bounding features can be found among the edges of bounding boxes of its constituent regions.

The simplest situation

To find out the bounding features of sign candidates, we first pick out the image regions with the colors of a road sign. The bounding box edges of these regions can be the constituents that form the bounding features of the sign. Suppose there are M regions in the segmented image with the colors of a road sign, each region $R_k, k = 1..M$ has four outer limits, $E^k_{top}, E^k_{right}, E^k_{bottom}, E^k_{left}$. If we consider any combination of four outer limits as boundary features, the number of such features will be $O(M^4)$. For the image in Fig. 22, the number of regions after segmentation is 2643. Among them, the number of regions with red and white colors (the colors of no-enter sign) is 847. The total combination of their bounding limits is of order $847^4 \approx 5.1 \times 10^{11}$.

Normally a sign's shape does not change, which means the ratio between the width and height of it bounding box is constant. Using this assumption, we need only three edges to define a transform. Not only that, if sign's shape is constant, we don't use four edges for boundary features, because not all four edge combinations agree with the shape constraint. The computation is now of order $O(M^3)$, which is ($847^3 \approx 6.1 \times 10^8$ for the image. This amount of computation is still too big. Apparently, we need constraints to reduce the number of regions to be considered.

Applying constraints

The anchor feature used in above scheme is very general. It almost allows anything to happen to a road sign and ensures detection almost in any situations. The difficulty is that it's not distinctive enough to enable quick detection. However, in most cases, a road sign is fully visible in an image, presented with a clear boundary. It would be unnecessary to spend large amount of time just to catch very few sign candidates in difficult situations, which in most cases may turn out to be unrecognizable due to excessive occlusion or deformity. This conforms to the 80-20 rule, which says 80 percent of the goal can be achieved by 20 percent of the total effort. Adding constraints usually means making wise tradeoffs between efficiency and accuracy.

As have been said, the number of anchor features is $4 \times M^3$ if there are M components in a sign and if we use three bounding edges for an anchor feature. In most cases, we do not need to use so many anchor features in detection. There is often a lot of redundancy if many anchor features are used. Many alignment projections will be made on one object instance from different anchor features. We can reduce the number of anchor features by using just a subset of all possible combinations of bounding edges. For example, we can assume that at least one of the sign's components is fully

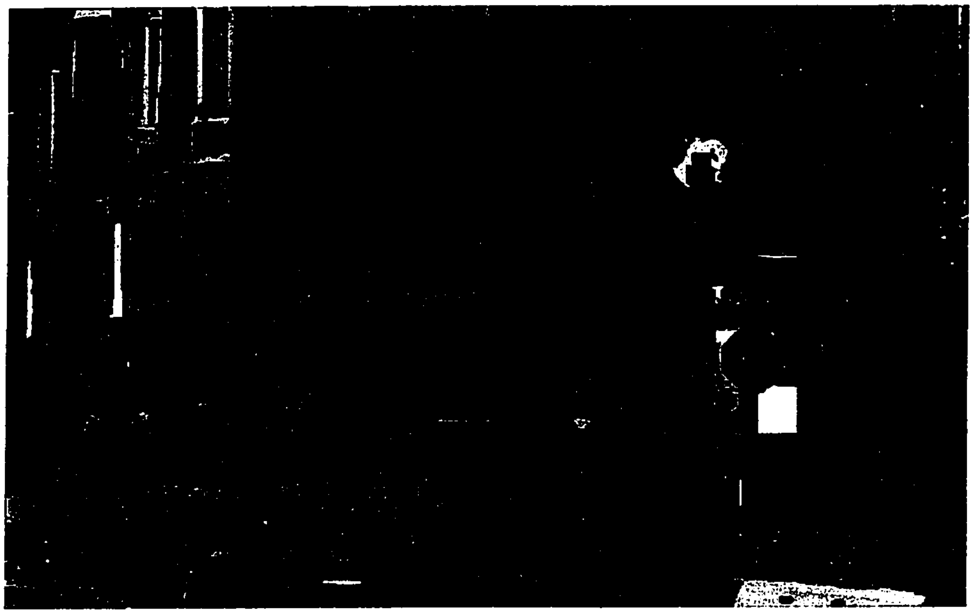


Figure 26: There are 847 regions with the colors of a road sign.

visible in the image. That means the bounding edges of at least one component are fully visible and can be used as a bounding feature to determine the sign's location. In this case, a road sign will have $4 \times M$ bounding features (each comes from one component) instead of $4 \times M^3$ bounding features.

We can put constraints on how big a road sign can appear in an image. Since an image is about a street scene, a sign normally occupies only a small portion of the image. Suppose the area of a sign candidate is limited by $1/K$ of the image, the number of regions is N, the regions distribute evenly throughout the image, then in average the number of anchor feature candidates is reduced to $N \times (N/K)^2$. For the image in Fig. 26, the combination would be less than 7.5×10^6. There can be also a limit for how small a sign can be. For example, if a sign candidate is smaller than 12×12 pixels, it usually becomes unrecognizable.

There are many fragmented small regions that don't make significant contributions to a road sign. If we discard these regions, the number of candidates will drop a lot. For example, in Fig. 27, the number of regions with the right colors and with an area larger than 4 pixels is 431. The anchor feature candidate number is then 9.9×10^4.

As a further step to reduce the number of anchor feature candidates from an image, we can use more bounding edges in an anchor feature. Under the assumption of no

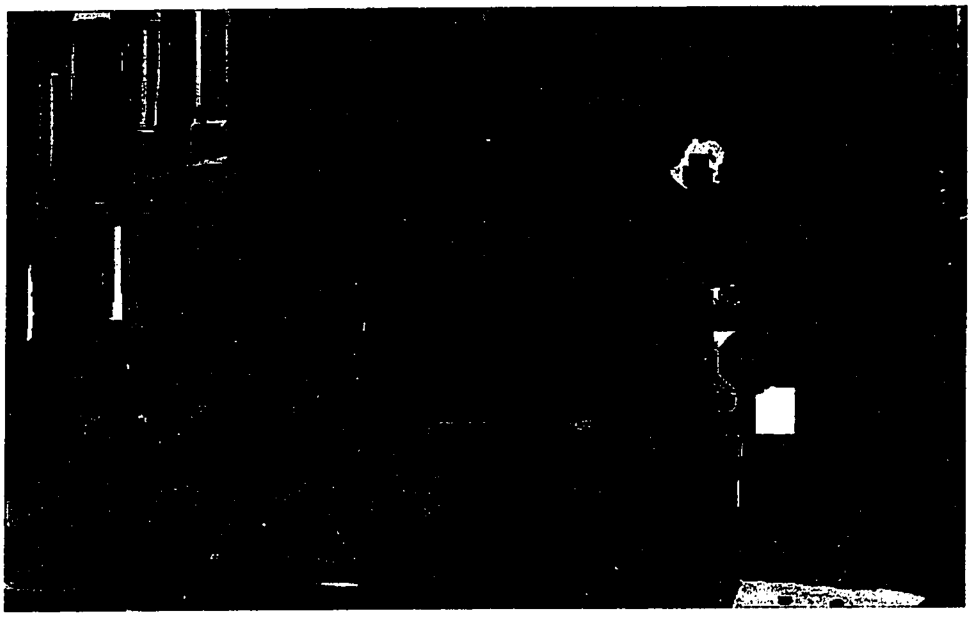


Figure 27: There are 431 qualified regions with area larger than 4.

shape change, once three (four if shape can change) bounding edges of a bounding feature have been found, an alignment transform is decided. Any extra bounding edges from that bounding feature will be used to validate or adjust the estimation of the transform. The more extra bounding edges used, the more shape constraints there will be to suppress false candidates. The side effect is the detection will be less robust to noise and occlusion if more bounding edges are used. We suggest using five or six bounding edges in a bounding feature.

These constraints greatly reduce the number of boundary feature detected in the image. The search starts at every E^k_{top}, $k = 1..M$. With the constraints, the effect of the search is like moving a number of windows of certain allowable shapes and sizes around the neighborhood below E^k_{top} to find out other bounding edges that satisfy the shape constraints among them. The result of detection using all the above constraints is shown in Fig. 28, where there are 200 detected candidates of the no-enter sign. This number is very small comparing to the situation where constraints about bounding features are not used.

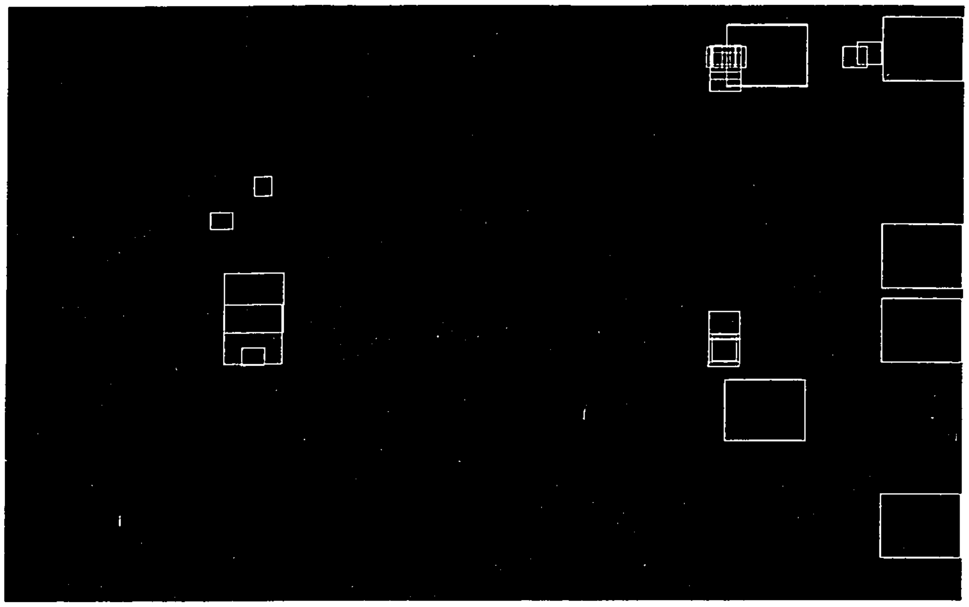


Figure 28: 200 projected road sign regions are detected.

4.4 Transform verification

For each detected anchor feature candidate, the alignment transform is performed and the image area is matched to the projected model image for verification. In this section we first discuss verification methods in general terms, and then apply the method to the verification of road sign candidates.

4.4.1 General matching strategy

For a given image, there may be far more road sign candidates detected than the real number of road sign instances. There are two reasons for the large amount of detected sign candidates. Either a lot of them are generated in places where bounding features appear but there is no real sign; or many bounding features are generated around a real road sign instance due to duplicated anchor features. We need a matching strategy to reduce the false candidates and verify real instances quickly. Using the hierarchy feature model defined in Chapter 3, we develop a unified matching method that integrates feature ordering and other matching performance enhancement tactics into one process.

In the feature model defined in Chapter 3, the i^{th} tree node at level j, namely N_i^j is composed of its own features F_i^j, its sub-nodes $N_{i,k}^{j+1}$, the weight $w_{i,k}^j$ for each sub-node, and matching threshold T_i^j. Suppose bounding features of a road sign have been found in the image. Apply alignment transform to estimate the locations $(A_i, i = 1..N)$ of the sign's candidates, using a perturbation to transform estimation to allow an error range. Mark every projected location as unmatched. For each unmatched projected location, perform the following recursive matching function using the root node N_0^0 as input.

MatchingVerification:
Input: Tree node N_i^j
Output: matching score for the node (between 0..1)

1. Project the area of node N_i^j of the model to the image. If the whole projected area in the image has been matched as part of a sign candidate, return 0.
2. Calculate node features F_i^j on the projected region in the image. If any of the calculated features from the image does not match the one in the model, go to step 6.
3. Initialize both the matching score $S_{i_{matched}}^j$ and the lost matching score $S_{i_{lost}}^j$ to 0.
4. For each sub-node $T_{i,k}^{j+1}$ of the current node,
   (a) Call function MatchingVerification($N_{i,k}^{j+1}$), get the matching score $S_{i,k}^{j+1}$.
   (b) $S_{i_{matched}}^j = S_{i_{matched}}^j + w_{i,k}^j \times S_{i,k}^{j+1}$. If $S_{i_{matched}}^j > T_i^j$, go to step 5.
   (c) $S_{i_{lost}}^j = S_{i_{lost}}^j + w_{i,k}^j \times (1 - S_{i,k}^{j+1})$. If $S_{i_{lost}}^j > 1 - T_i^j$, go to step 6.
5. T_l^j has been found, mark the projected area of this node as matched, return function with value 1.
6. T_i^j has not been found, return function with value 0.

This function works on each node recursively. It first checks if the projected image area for this node has been marked as part of any already matched sign candidate. If so, stop matching and return 0. This is based on the one-object-in-one-area assumption. This avoids repeated matching on one area that is projected by multiple anchor features.

It then checks that if the node's features are matched. If the one of the features is not matched, the node is declared not found. If features of current node are matched, the function goes on to match the node's sub-nodes. It starts from the sub-node with the biggest weight. The matching sequence uses vertical and horizontal feature ordering in matching each feature. Vertically, it matches higher layer features first, since they are easier to calculate than lower layer features. Horizontally, within each feature layer, it calculates large weight feature first to reach a decision more quickly.

In matching the sub-nodes, it accumulates the matching scores from the matching result of each sub-node. If the matching score passes the matching threshold, the matching process stops and the node is labeled as matched. This is equivalent to heuristic termination that the search stops when a good enough candidate has been found. The image area of the matched node is marked as matched so that future match won't happen in the same place again.

If the sub-node is not matched, lost score is increased by the weight of the sub-node. If the lost score is higher than $1 - T_I^j$, the matching stops and the node is labeled as unmatched. This is equivalent to the branch-and-bound technique that stops searching if there is no more hope of finding an acceptable candidate in the area.

4.4.2 Road sign candidate matching

Once a road sign candidate has been detected, there are a lot of possible ways to match the candidate to the road sign model. Due to the scope of the thesis, we will not investigate all the possible methods for object matching. Our main interest is to build up the framework of feature organization and candidate verification. Any feature can be used in such a framework as long as it provides the method to calculate it in the image, and the way to match two versions of such feature. To test our framework, we choose to use three simple features from three feature layers for candidate verification.

The root level global feature is the sign mask, bounding box shape, and color histogram. Sign mask is used in filtering out regions that is not in the area of a sign model (Fig. 27). This is necessary when the road sign is not rectangular. Matching won't happen in projected mask areas. Bounding box shape defines the shape of boundary features and has been used in alignment detection.

The first feature being used in matching process is color histogram. It is the area percentage distribution of road sign's color components. It is very powerful in distinguishing colored objects [77, 78, 79]. Since in alignment detection color is used to select candidate regions, we can assume that the projected candidate region contains the color components, say $C_k, k = 1..M$, of a sign model. The color histogram stored in the model contains the area percentage for every color component, say they are

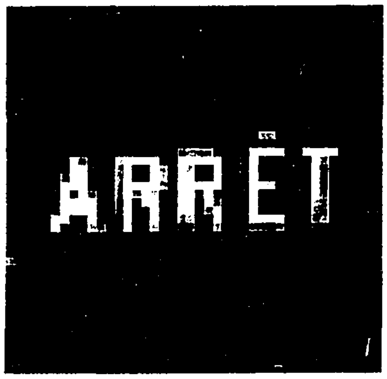


Figure 29: A road sign model with its mask.

$R_k, k = 1..M$. Calculate the area percentages of the image regions with these colors, R_k'. Then the color histogram matching score is

$$S = \sum_{k=1..M} min(R_k, R_k')$$

If the score is higher than a threshold, we go on to match the sub-nodes of the root node, which is the color component level.

For each component, we have following global features: location, color, compactness, centroid and area. Color and area have been used in calculating the color histogram feature in root node. Compactness and location are less reliable since a component's location may change due to noise and occlusion. So is the compactness, which depends on the value of location. As a result, we use only centroid as the feature in component level. The centroid is measured in terms of location of a component centroid (C_x, C_y) relative to the image's upper-left corner. That is, suppose the centroid is located at (x, y), the width and height of the image is w, h, then $C_x = x/w, C_y = y/h$. Supposed the component centroid for a sign candidate is (C_x', C_y'). The centroid matching score for the component is

$$S = max(C_x - C_x', C_y - C_y')$$

If there are more levels of sub-components, they are matched in the same fashion. Components are matched in descending order of their weights. However, components' weights are equal by default, unless assigned otherwise by human.

On the shape primitive level, we use the 8-directional chain-code histogram on the component edges as the feature to be matched. Suppose the percentages of chain

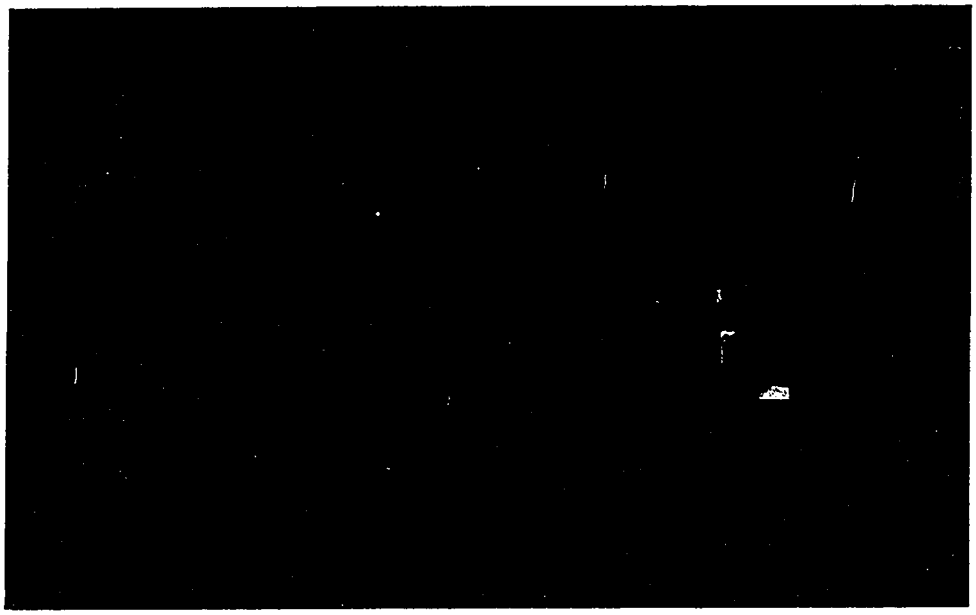


Figure 30: Verification result.

codes for the model are $C_i, i = 1..8$, with the chain code histogram in the candidate as $C'_i, i = 1..8$, the matching score is

$$S = \sum_{k=1..8} min(C_i, Ci')$$

The result after applying the features in function *MatchingVerification* in alignment detection is shown in Fig. 30. The intended no-enter sign and other two regions with some common characteristics are show in the image.

5 Sign recognition

Detected and verified road sign candidates are further matched against road sign model for final recognition. Theoretically we can use any classification method(s) to decide if a candidate belongs to certain type of road sign. Unlike in some applications where an incoming sample is for sure an instance of one of the models, candidates from the detection stage may not be a real road sign instance. On the other hand, it is possible that a detected candidate may be one of several similar road signs (such as parking and no parking) that the detection can not distinguish. As a result, in this application, a classifier needs to decide two things: whether a candidate is an instance of a road sign; and if it is, which road sign it belongs to.

A classifier usually estimates the likeliness between a candidate and a road sign model. It gives high response to real road sign instances, while suppresses its response to anything else. A similarity threshold is often used to decide whether a candidate can be an instance of the sign model, and if it's distances to multiple sign models are all under this threshold, the model which is nearest to the candidate is often chosen as the class for the candidate.

The classifier can work on either geometric structure, or the pixel template of the candidate. Shape and graph matching methods are usually expensive and complicated. Due to the extent of this work, we study the use of template matching as the way to verify a road sign candidate.

Compare to a perfect road sign model, a sign candidate coming out of detection is often perturbed by noise, slight translation and rotation. A matching method should deal with these variations and give correct answer under reasonable amount of noise and perturbation. Some most commonly used methods are cross-correlation [4, 5] and neural networks [84]. Here we will use Hausdorff distance as the base for measuring matching quality, because it is resilient to slight translation and rotation, and has strong discerning ability using just a few training samples.

5.1 Modified Hausdorff distance

Hausdorff distance is advocated by some researchers [82] as the measurement in comparing two point sets. The definition of Hausdorff distance is as follows:

Given two finite point sets $A = \{a_1, ..., a_p\}$ and $B = \{b_1, ..., b_q\}$, the Hausdorff distance is defined as

$$H(A, B) = \max(h(A, B), h(B, A))$$

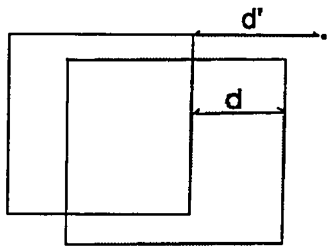


Figure 31: An example of Hausdorff distance.

where

$$h(A,B) = \max_{a\in A}\min_{b\in B} \parallel a - b \parallel$$

and $\parallel \cdot \parallel$ is some underlying norm on the points of A and B (e.g., the L_2 or Euclidean norm).

A simple example of Hausdorff distance is shown in Fig. 31. The distance between the two rectangles is measured as the maximum of minimum distances between the two point sets, which is d. It is argued that because the minimum Hausdorff distance (minimum is achieved by translations and rigid motions) obeys metric properties, that is, everywhere positive, identity, symmetry and triangle inequality, this distance correspond to our intuitive notions of shape resemblance. Since the distance is measured by maximum of minimum distance between two point sets, slight translation and distortion will only cause slight change in Hausdorff distance. This is a very nice feature, especially for our case.

It should be noted that Hausdorff distance is susceptible to noise. For example, adding into a point set just one pixel that is far away the previous points will drastically change the Hausdorff distance between the two point sets. As shown in Fig. 31, with the noise point, the distance is now d', which is much larger that d. A number of modified versions [82, 83] have been suggested to improve the resilience to noise. For example, a distance can be calculated on a kth (k is a portion of p and q) largest (instead of the maximum) distance of two images, assuming the number of noise is less than p-k (or q-k) pixels. The selection of k is somewhat arbitrary and hard to decide. Sometimes useful information will be cut off. Another choice is to use averaged minimum distance to alleviate the impact of noise, as defined below:

$$D(A,B) = \frac{\sum_{a\in A} h(a,B) + \sum_{b\in B} h(b,A)}{p+q}$$

where

$$h(a,B) = \min_{b\in B} \| a - b \|$$
$$h(b,A) = \min_{a\in A} \| b - a \|$$

h(a, B) is the minimum distance from a point in A to any point in B.

However, sometimes the averaging effect is very strong that it will scale down the small but significant changes in shape. To reduce this effect, we propose to use the maximum of unidirectional average minimum distances, we call it maximum averaged Hausdorff distance, as defined below:

$$D_h(A,B) = \max(\frac{\sum_{a\in A} h(a,B)}{p}, \frac{\sum_{b\in B} h(b,A)}{q})$$

After many tests, we found that this distance is much better than other options we mentioned above.

5.2 Computing the Hausdorff distance

Hausdorff distance can be calculated from Voronoi diagrams of two point sets. Given two edge maps of the same size, we want to compute the Hausdorff or averaged Hausdorff distance. To do this, we first calculate the Voronoi diagram of each edge map. The definition of Voronoi diagram is as follows:

> Consider a set of coplanar points P. For each point P_x in the set P draw a boundary enclosing all the intermediate points lying closer to P_x than to other points in the set P. Such a boundary is called a Voronoi polygon and the set of all Voronoi polygons for a given point set is called a Voronoi diagram.

Here we represent Voronoi diagram not in terms of boundaries, but in terms of the distance from each pointer to its nearest point in P. This representation is the basis for efficient calculation of Hausdroff distance.

We design an iterative growing process to calculate the Voronoi diagram for a point set image as shown in Fig. 32(a). In its corresponding Voronoi diagram, each pixel is assigned a distance value d, and a pair of values (dx, dy) as the horizontal and vertical distance to its nearest point in the point set. In initial state, the pixels for edge points are assigned distance value 0 (d=0, dx=0, dy=0), while other pixels of the image are undecided (labeled as "x") as shown in Fig 32 (a). In each iteration every undecided pixel in the image is assigned a distance value if one of its neighboring pixel has been assigned a distance value. For example, if $P_{dx,dy}$ has been assigned a distance value (d, dx, dy), and $d = \sqrt{dx^2 + dy^2}$, its neighboring pixel $P_{dx+d_1,dy+d_2}$, $d_1, d_2 \in -1, 0, 1$ has undecided distance value, then we update the distance value of this undecided point to $(d', dx + d_1, dy + d_2)$, and $d' = \sqrt{(dx + d_1)^2 + (dy + d_2)^2}$. If $P_{dx+d1,dy+d2}$ has more than one neighboring pixels, the distance value of $P_{dx+d1,dy+d2}$ is then the smallest of all possible updates. This update process will be iterated until all the pixels are assigned a distance value. Fig 32 (b), (c), (d) show the iterations for the sample image. The result is the Voronoi diagram Fig 32 (d), in which each pixel $P_{x,y}$ has a value $D^A_{x,y}$ which is the distance to its nearest point in the point set A.

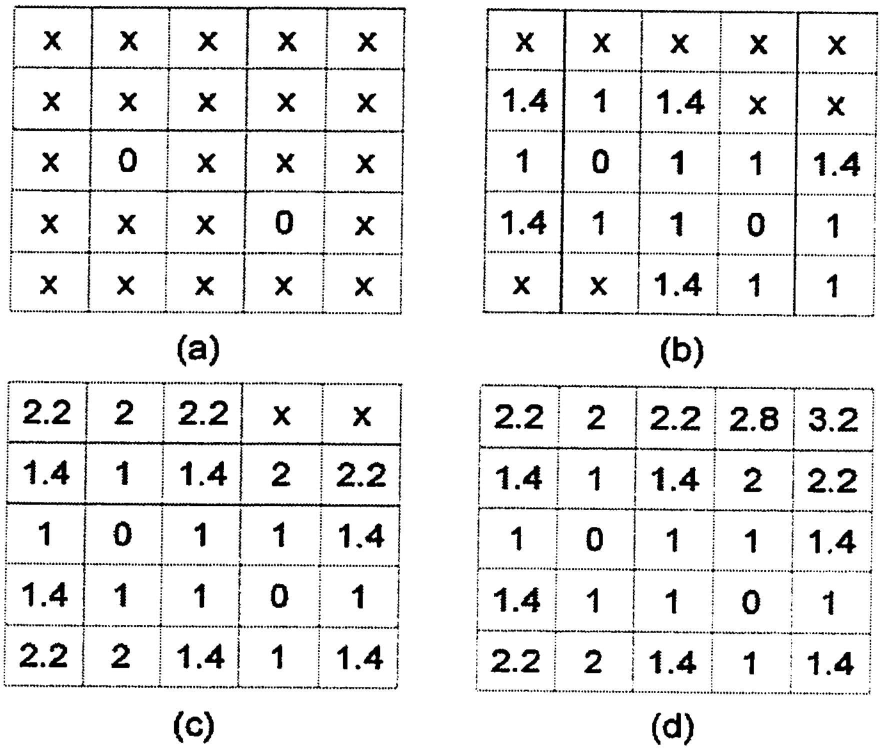
Now suppose another point set (Fig. 33(a)) comes in and we want to calculate the Hausdorff distance between the two point sets. We calculate the Voronoi diagram for this point set (Fig. 33(b)). For each point $P_{x,y}$ in set B, $h(P_{x,y}, A) = min_{a \in A} \| P_{x,y} - a \| = D^A_{x,y}$. The original Hausdorff distance is $H(A, B) = max(1, 1.4, 1) = 1.4$. The average minimum distance is $D(A, B) = (1 + 1.4 + 1)/3 = 1.13$. The maximum averaged distance is $D_h(A, B) = \max((1 + 1.4)/2, 1) = 1.2$.

We can see that with the help of Voronoi diagram, the calculation of the Hausdorff distance is just one scan over the Voronoi diagram to find out the distance value of certain points in one image whose corresponding points (points at the same places) in another image has a distance value of 0 (edge point). The computation is obviously very efficient.

5.3 Road sign classification

Hausdorff distance is defined on two point sets. There are many possible ways to extract point sets from an image for this purpose, such as points that come from one color component. For a road sign with several color components, there may be multiple point sets, and so there may be multiple Hausdorff distances that need to be combined into one final matching score. This is quite complicated. To keep it simple, we use edge points as the single point set for the distance calculation.

To generate the edge map for Hausdorff distance, we just need one-pixel width edges.



x	x	x	x	x
x	x	x	x	x
x	0	x	x	x
x	x	x	0	x
x	x	x	x	x

(a)

x	x	x	x	x
1.4	1	1.4	x	x
1	0	1	1	1.4
1.4	1	1	0	1
x	x	1.4	1	1

(b)

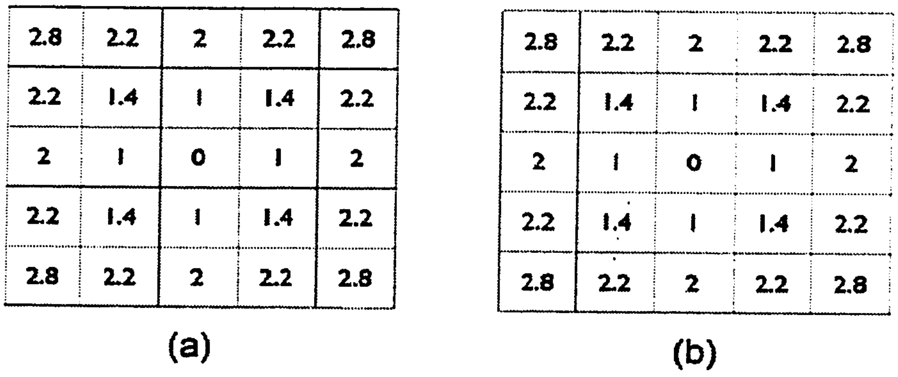
2.2	2	2.2	x	x
1.4	1	1.4	2	2.2
1	0	1	1	1.4
1.4	1	1	0	1
2.2	2	1.4	1	1.4

(c)

2.2	2	2.2	2.8	3.2
1.4	1	1.4	2	2.2
1	0	1	1	1.4
1.4	1	1	0	1
2.2	2	1.4	1	1.4

(d)

Figure 32: Point set A and its Voronoi diagram.



2.8	2.2	2	2.2	2.8
2.2	1.4	1	1.4	2.2
2	1	0	1	2
2.2	1.4	1	1.4	2.2
2.8	2.2	2	2.2	2.8

(a)

2.8	2.2	2	2.2	2.8
2.2	1.4	1	1.4	2.2
2	1	0	1	2
2.2	1.4	1	1.4	2.2
2.8	2.2	2	2.2	2.8

(b)

Figure 33: Point set B and its Voronoi diagram.

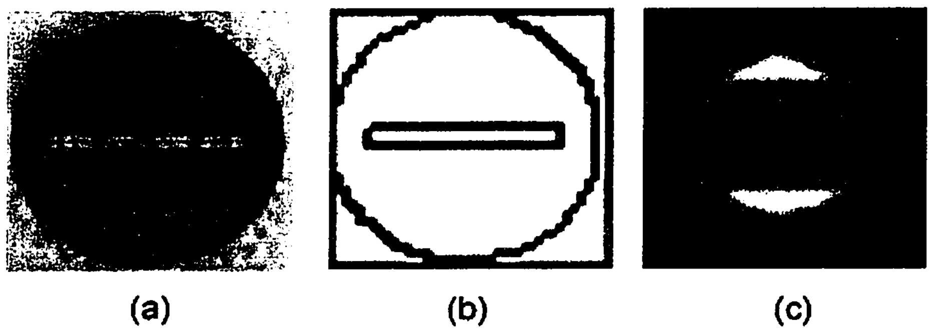


Figure 34: A road sign model (a), its edge map (b) and Voronoi diagram (c).

If we use region growing method to segment the two images, boundaries between regions are two pixels in width, with one boundary line contained in each region. We can take one side (say, upper-left side) of a boundary to form the edge map. The outside boundary of a road sign may or may not appear in a detected candidate or a model, while such a boundary should exist in the ideal case. To avoid matching inconsistencies, we put such a boundary into every candidate regardless of whether its outside boundary exists or not. For models with masks (see section 3.5), we also create a boundary between the model's mask and the model's real pixels.

A road sign, its edge map and the Voronoi diagram are shown in Fig. 34. The Voronoi diagram is depicted in intensity values, where higher intensity stands for larger distance value, and black area stands for points on the edge where distance value is zero.

To match two images, we first scale them into the same size, extract their edge maps, and then calculate the modified Hausdorff distance based on the Voronoi diagrams of the two edge maps. After that, we normalize the distance with the dimension of image. The final normalization step is necessary to get comparable results between models of different sizes. Suppose the width and height of the matching images (they have been scaled to the same size) are w, h, the normalized maximum averaged Hausdorff distance between A and B is:

$$D = D_h(A, B)/(w + h)$$

where $D_h(A, B)$ is the maximum average Hausdorff distance defined in section 5.1.

To test the classification ability of maximum averaged Hausdorff distance on edge maps, we use two groups of signs. We want to measure the distances between a road sign model and it's real candidates, as well as the distances between a sign's

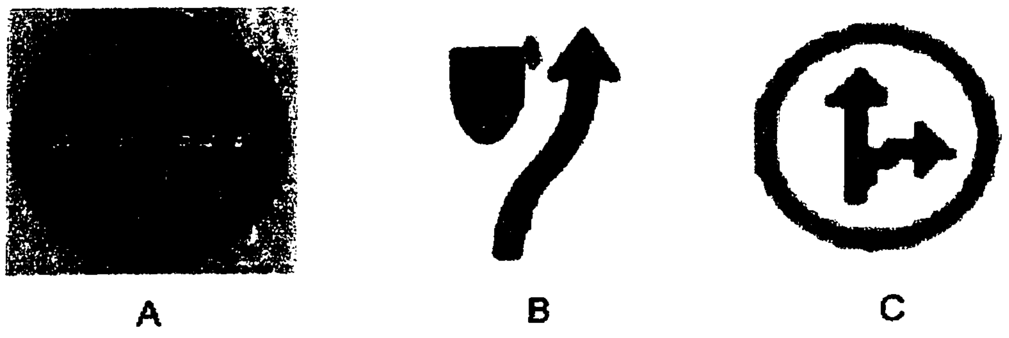


Figure 35: Three types of quite different road signs.

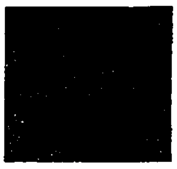
				
A	0.051	0.062	0.064	0.112
B	0.160	0.147	0.164	0.228
C	0.108	0.107	0.098	0.112

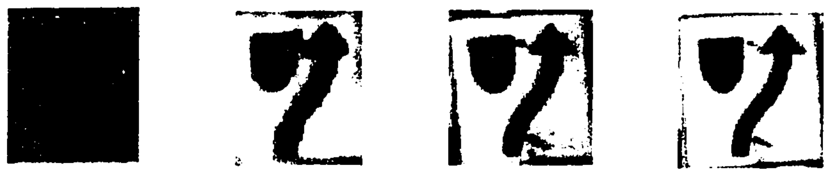
Figure 36: Classification of detected no-enter signs.

model to the instances of other type of road signs. For a good classifier, the inner-class distance should be small, while the inter-class distance should be big. If we use nearest neighborhood classification, then the basic requirement is that a candidate's true type should be its nearest neighbor in terms of Hausdorff distance among all the sign models.

The first group of signs we tested are three quite different types of signs, as shown in Fig 35.

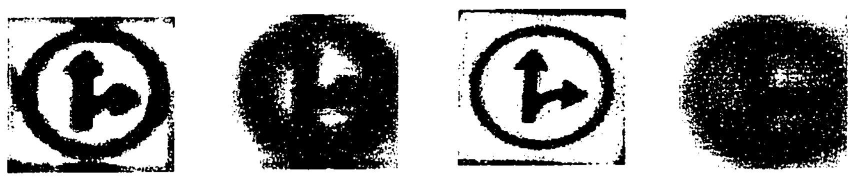
We have three set of sign candidates detected from the images for these three road sign models, and we match them to the three models (A,B,C) respectively. The results are shown in Fig. 36, Fig. 37, Fig. 38.

Then we test the classification on similar road signs. We choose two such signs as



A	0.092	0.177	0.209	0.367
B	0.043	0.065	0.142	0.240
C	0.113	0.125	0.183	0.308

Figure 37: classification of detected detour signs.



A	0.123	0.144	0.163	0.321
B	0.158	0.172	0.172	0.225
C	0.078	0.075	0.094	0.150

Figure 38: Classification of detected direction signs.

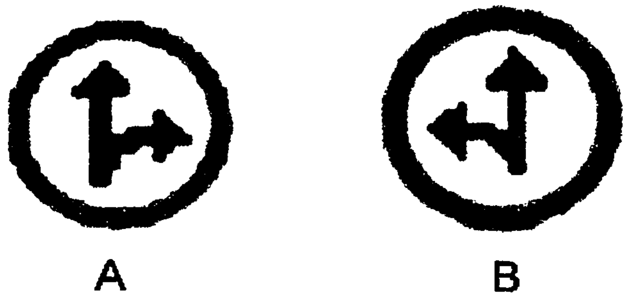
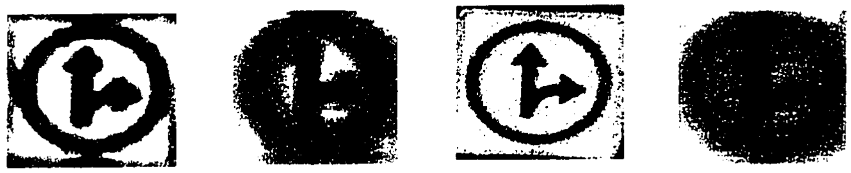


Figure 39: Two types of similar road signs.



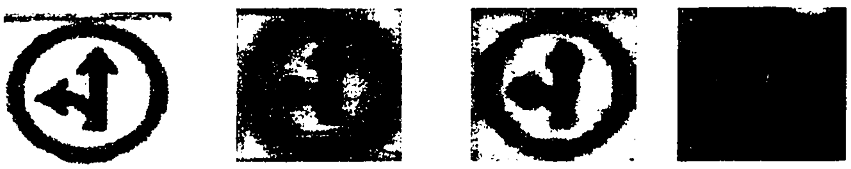
A	0.078	0.075	0.094	0.150
B	0.153	0.153	0.122	0.152

Figure 40: Classification of detected straight-right signs.

shown in Fig 39. The matching results with sign instances are shown in Fig 40 and Fig. 41.

As can be seen from the results, inner-class modified Hausorff distances are constantly lower than inter-class distances. This is very nice since we can use nearest neighbor classifier to achieve final recognition. In real situation, we also need a distance threshold to use the distance as a one-class classifier to exclude false positives. Since in many cases, the system detects candidates for just one type of road sign. The sign may be very different from other signs that there is no chance for the candidate to be the instance of other signs. Overall, the result is very encouraging since only one model for each type of sign is used, and the samples being tested are of different size, more or less distorted or translated, along with other noise.

We use modified Hausdorff distance to recognize the verified no-enter sign candidates in section 4.4. The three verified candidates are shown in Fig. 42. Note they all have white background, red circle and some white area inside the red circle. That's why they are detected and verified as candidates. Their Voronoi diagrams are in Fig. 43.



A	0.117	0.125	0.102	0.125
B	0.041	0.065	0.083	0.114

Figure 41: Classification of detected left-straight signs.

Figure 42: Verified candidates for no-enter sign from detection stage.

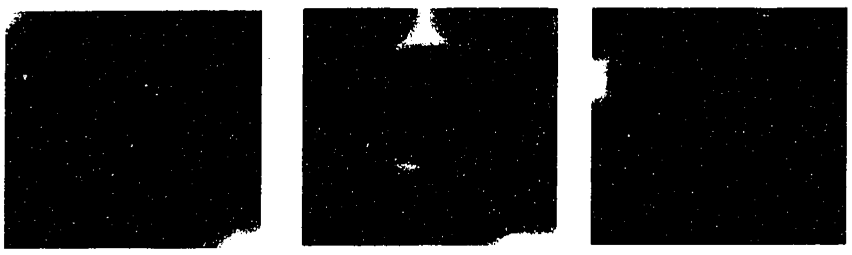


Figure 43: Voronois diagram of candidates. D=0.11, 0.20 and 0.18.

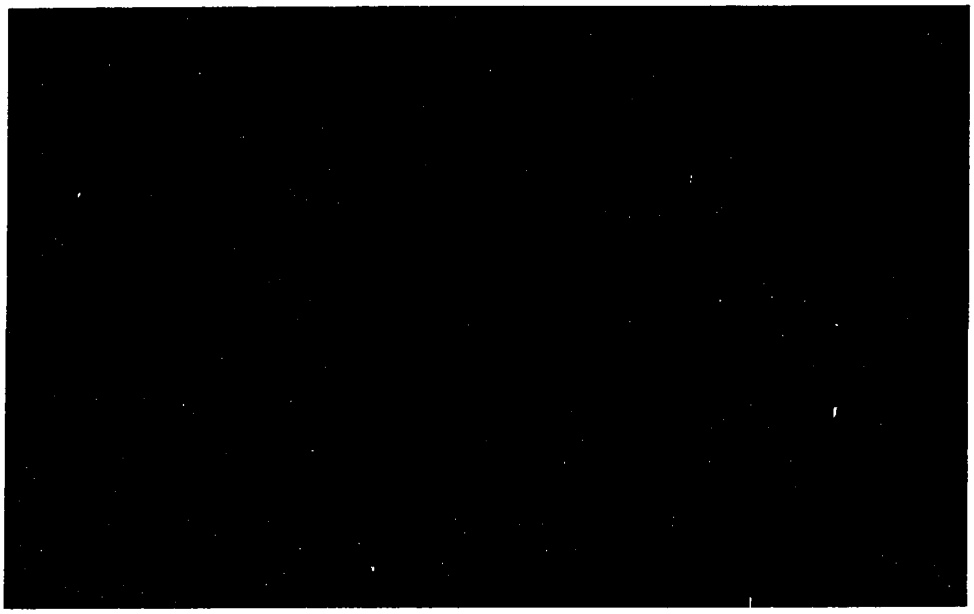


Figure 44: Final recognition result.

The no-enter sign is recognized since its modified Hausdorff distance is much lower than the other two.

The final recognition result is shown in Fig. 44.

6 Experiment results and Conclusions

A road sign recognition system has been developed to detect and recognize road signs from color images of street scene. It contains two major parts: model generation and sign recognition. In model generation, the user provides the system with a sample model, the definition of color components in the model, and the weights associated with the components. The system will then generate the feature representation of the model. This model is used in detection and recognition of road signs. In road sign recognition, the system reads in color images, performs segmentation, uses the feature from the model to detect sign candidate locations, verifies these locations with the features from the model, and finally recognizes the road sign, using matching algorithm.

6.1 Experiment result

Altogether we have found more than a dozen of road signs in our approximately 200 experimental images (Fig. 45). Due to the lack of sample images for each type of sign, and the sign size and quality vary from image to image, it is not very meaningful to produce a recognition rate for most of the signs. However, we have more than ten image samples for each of several major signs, such as no parking, no enter, stop, line direction etc. We conducted recognition tests using these samples.

For each type of these signs, we extract a medium sized sample as the model. We process these models and store their feature representations into different files. Then we use these files to test against different street scene images. In building a model from a sign image, we need to specify the number of color components of the sign, the RGB value of these components, and the color of the sign mask (section 3.5). We also need to set the thresholds for the system to make detection and recognition decisions. The thresholds include segmentation and detection color distance thresholds, possible sign size ranges, minimal region size for consideration in detection, anchor feature selection and feature matching thresholds. In detection, we use 6 boundary edges to form an anchor feature to reduce the number of candidates. In verification, we use color histogram, component centroid and 8-direction chain code histogram for component edges as three layer features. They are easy to compute and also effective to removing false positives. The tests are done on Pentium II 266.

Fig. 46 shows a model image. Fig. 47 shows a street scene. Fig. 48 shows the segmentation result. There are 6011 regions after segmentation, 1326 regions with the color of the model and 658 regions larger than 4 pixels. Fig. 49 shows 2 detected

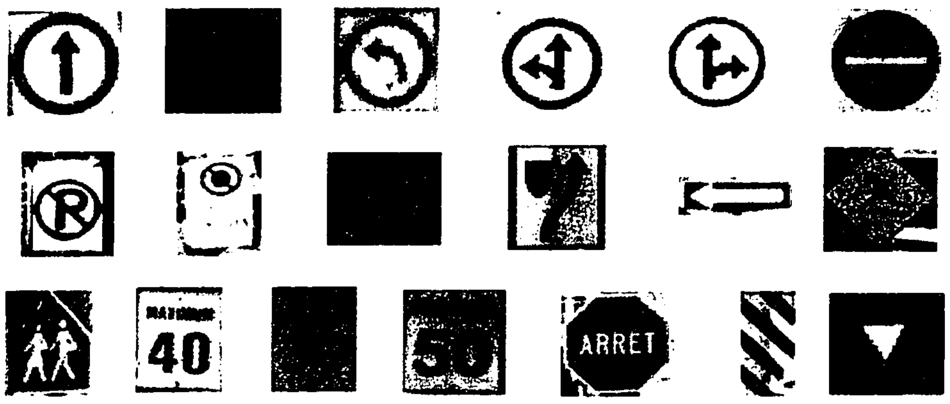


Figure 45: Road sign samples.

Figure 46: Road sign model.

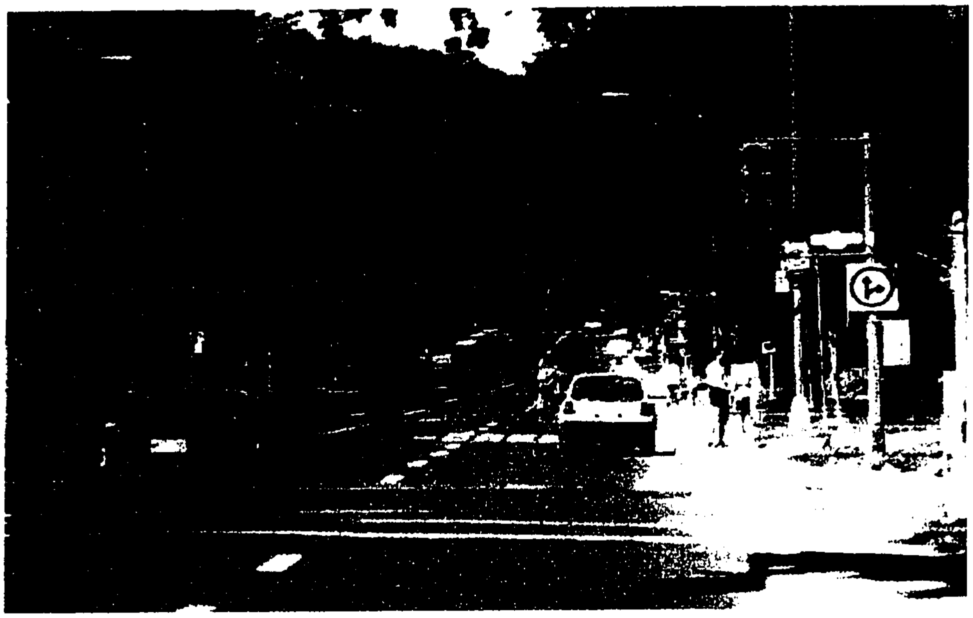


Figure 47: A street scene.

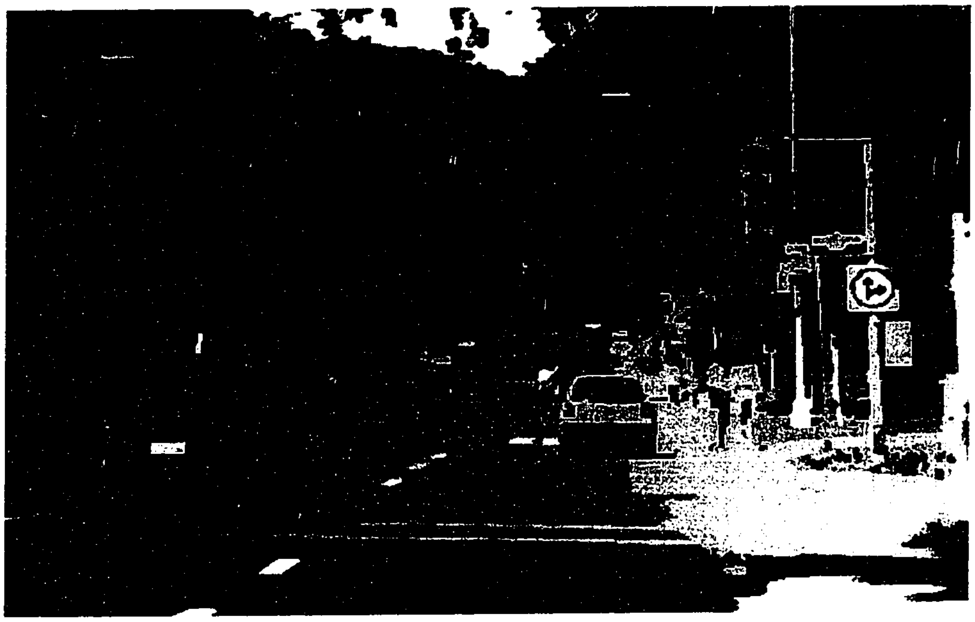


Figure 48: Segmentation result.

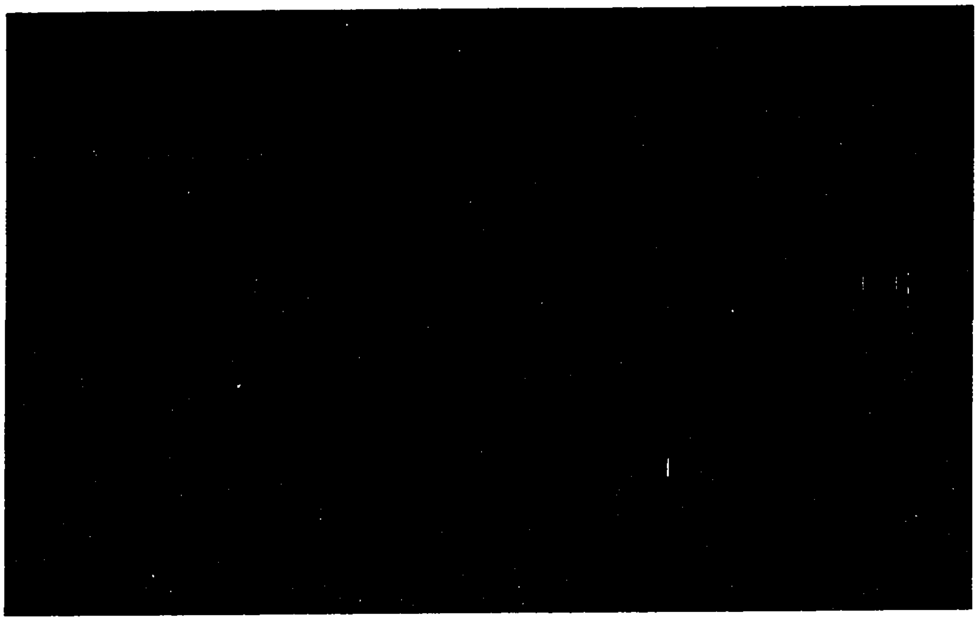


Figure 49: Detection result.

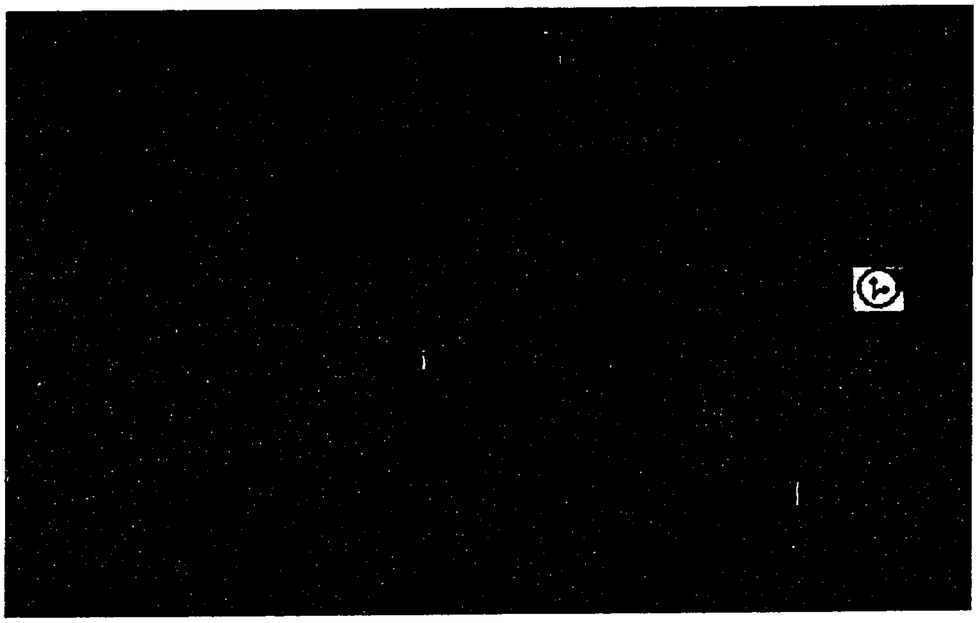


Figure 50: Recognition result.

Figure 51: A street scene containing another but similar road signs.

candidates, with the 2 regions at the right side verified. The final recognition result is shown in Fig. 50 where the sign instance is recognized with the modified Hausdorff distance to the model $D_h = 0.07$.

Then we use the same model in another image (Fig. 51) which contains similar but different road signs. There are 2798 regions after segmentation, 706 regions with the colors of the sign, and 314 regions larger than 4 pixels (Fig. 52). Four bounding boxes of detected candidates are shown in Fig. 53. Two objects are verified to be road sign candidates (Fig. 54) but we reject them as false positives because their modified Hausdorff distance to the model is quite high ($D_h > 0.13$).

To test our method's effectiveness in dealing with complicated scene, we run our program to recognize a black/white sign (Fig. 55) in a scene full of artifacts of similar color (Fig. 56). After segmentation, there are 2119 regions altogether, among them, 766 regions are with the colors of the sign, 417 regions with the size larger than 4 (Fig 57). Due to the similar color artifacts, a relatively larger number (67) of anchor features are detected (Fig. 58). However, only 1 of them are verified as sign candidate (Fig. 59) and it is finally accepted as an instance of the road sign ($D_h = 0.07$).

The region growing method is quite reliable and fast. A typical 600×400 image can be segmented in 2-3 seconds. Segmentation threshold is handpicked and is applicable to

Figure 52: Segmentation result.

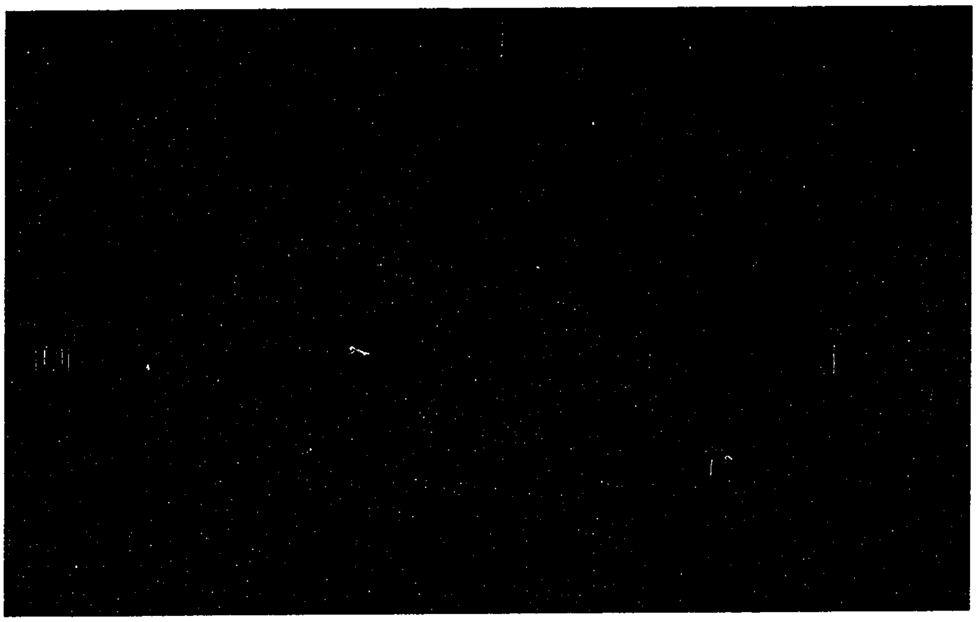


Figure 53: Detection result.

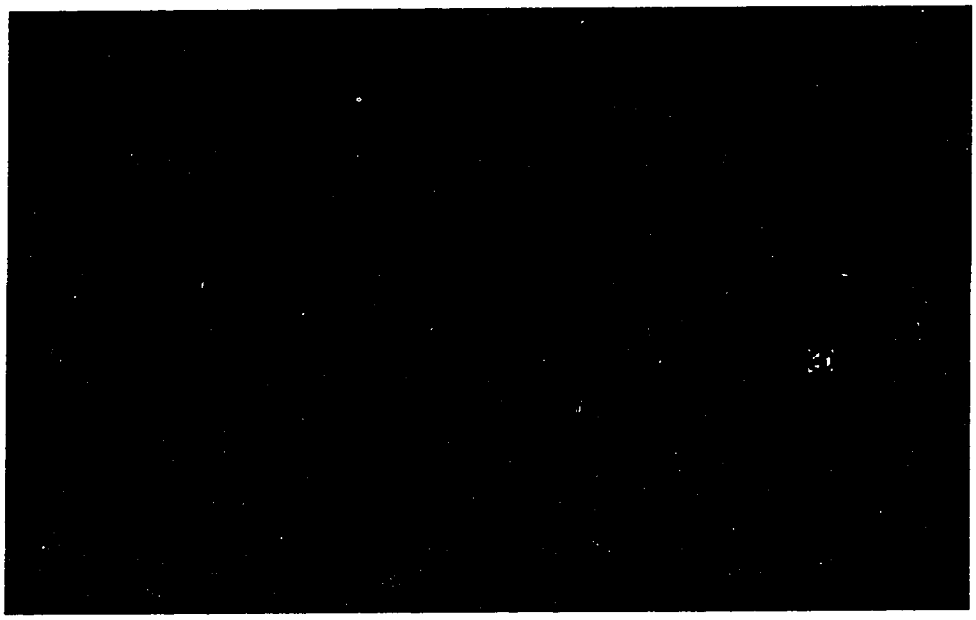


Figure 54: Verified candidates.

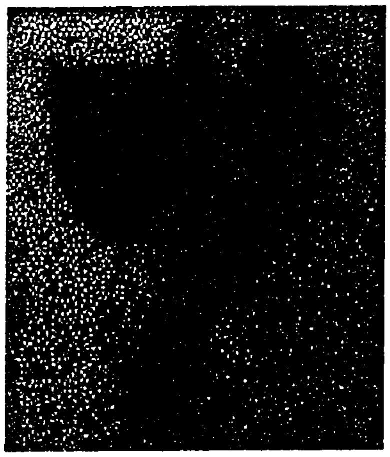


Figure 55: A black and white road sign.

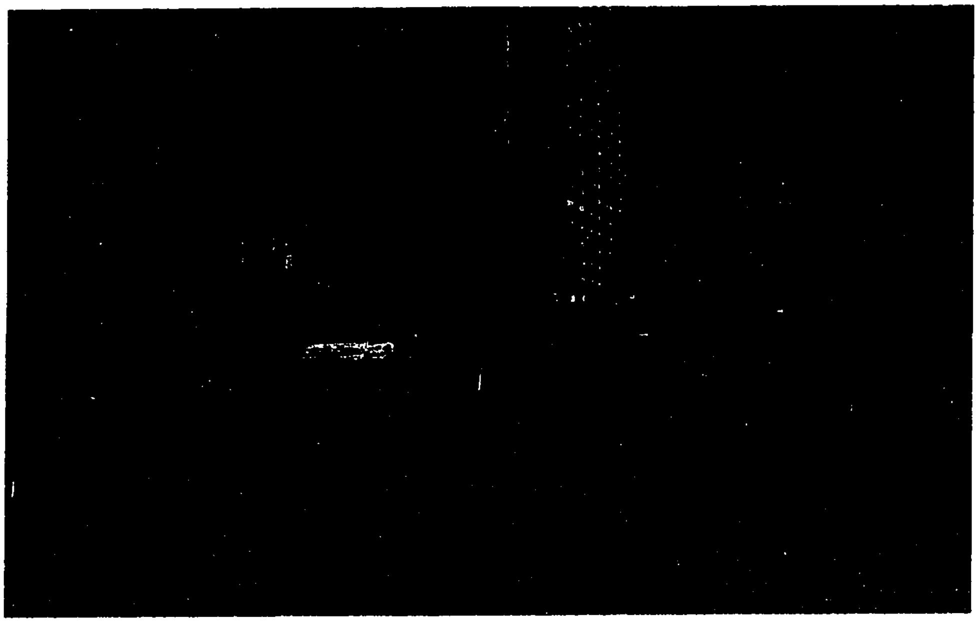


Figure 56: A complicated scene with similar color artifacts.

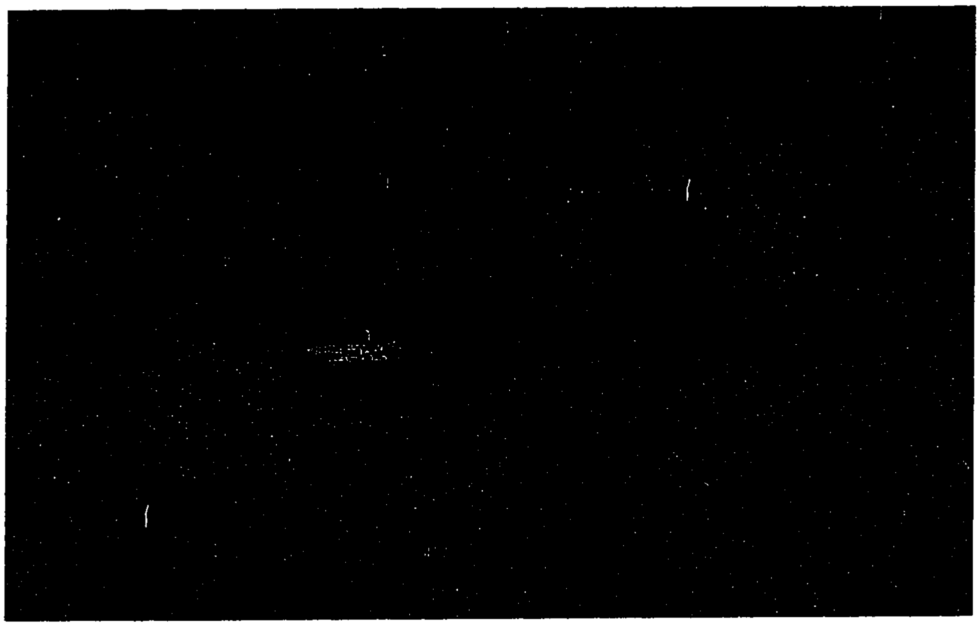


Figure 57: Segmentation result.

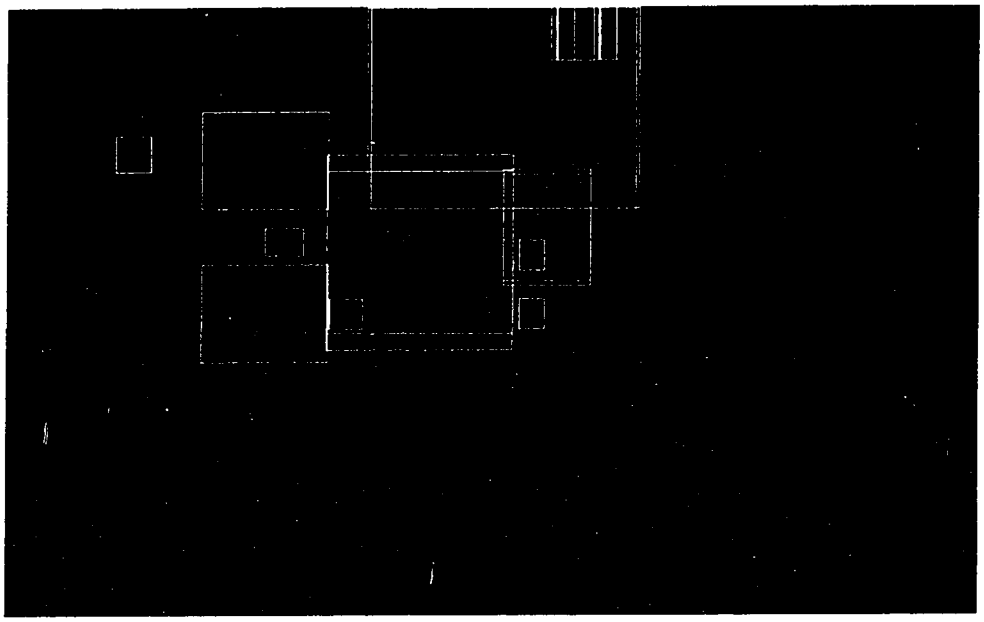


Figure 58: Detection result.

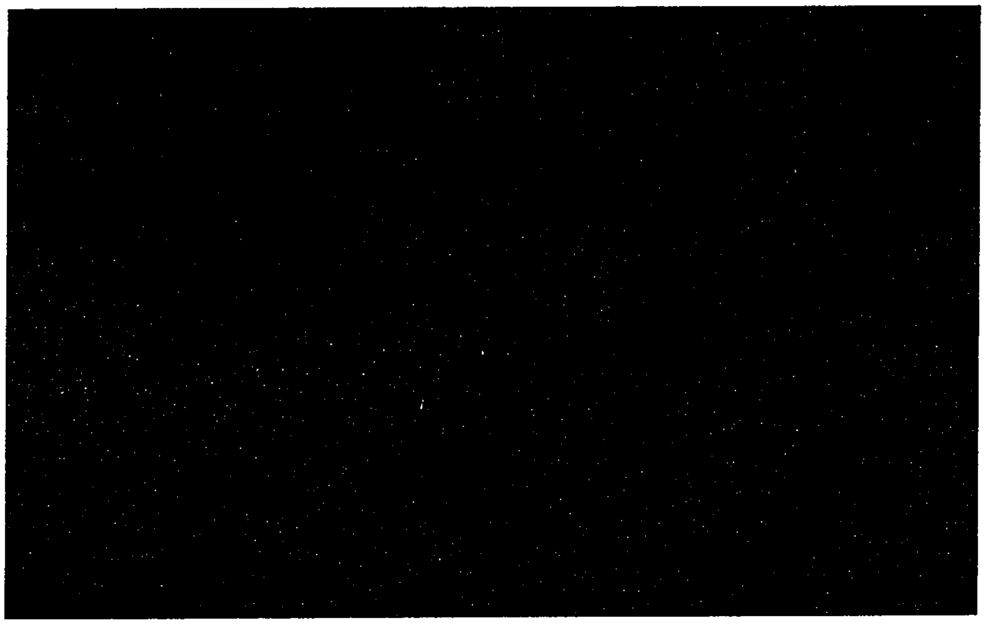


Figure 59: Recognition result.

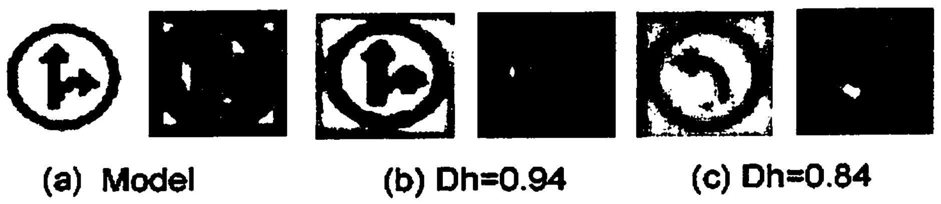


Figure 60: Hausdorff classification with false positive result

most cases. The principle is to use a lower threshold to keep the original information if we are not sure of the best value for certain images.

Anchor feature detection with constraints performs quite well. The time for detection is between 5 to 15 seconds, depending on the complexity of street scenes. Scene in Fig. 47 costs most processing time (15 seconds) because it contains the largest number of segmented regions. Six bounding edges form strong shape constraints for possible anchor features. It effectively reduces the number of anchor feature candidates to a manageable amount. The three layer features we used are also very effective in reducing false positives and keeping the real candidates.

Modified Hausdorff distance gives fairly good estimates on sign similarities. Some mistakes happen when the sign candidate is very small and very noisy. The candidate edge map becomes crowded with noise. When it is enlarged to the same size of the model, the error is enlarged. However, the distance between this kind of candidate and the model is usually much smaller than the distances between most none-sign candidates and the sign model. As mentioned in section 5.3, in order to use Hausdorff distance in classification, we need a distance threshold to exclude false positives, and we use nearest neighborhood to decide the candidate's class in case the candidate is detected for several similar sign models. In Fig. 60, if we still set the classification threshold higher to include the sign candidate, then the left-turn sign will in included as a false positive. This kind of error (false positive) won't be a problem if we consider multi-class classification. We can add a left-turn sign model to participate the classification using the candidate's nearest neighbor within certain distance threshold as the result of the final classification (Fig. 61).

6.2 Conclusion

In this work, we studied the problems involved in road sign recognition, such as segmentation, feature representation and matching, searching strategy, and template matching problem. A set of efficient and effective algorithms is proposed.

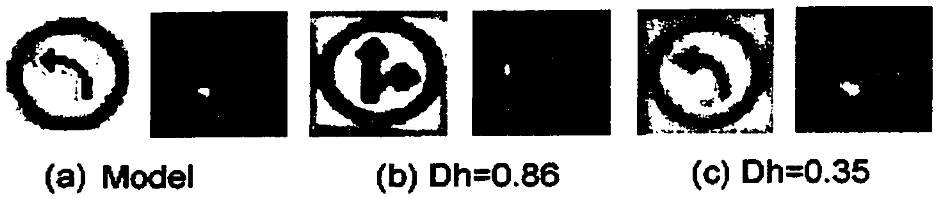


Figure 61: Using multi-class classification will get the right result.

Segmentation is the first step in our system. It provides the edge and region information to the later stages. The region growing method is quite reliable and fast. With our detection method, segmentation is important but not vital as it is used to be. Good segmentation can reduce the number of primitive features in an image and facilitate the detection, but more importantly we want to keep the key information to ensure the success of detection. The two major remaining issues are the choice of threshold and the false edge problem. Given the lack of analytic method to decide threshold, we suggest setting it relatively low so that vital information won't get lost. However, low threshold will increase the number of regions generated and slow down the processing speed. Fortunately, with given hardware device and certain environment, the lighting condition is more or less constant in most cases and the handpicked threshold works well. We also discussed the false edge problem that comes with region growing, and we suggest keeping edge and region as different segmentations in the same time. Fortunately, this problem is not very visible in road sign areas, since road signs usually don't have slow ramps where false edge is likely to appear.

We proposed a framework to organize features that represent a road sign model. Vertically, we represent road sign models in hierarchical structure, organize different levels of information involved in detection and recognition into different levels of the representation tree. Horizontally, the features in one level are ordered by their respective weight value. In verification stage, the most important feature is verified first, so that a verification decision can be reached more quickly. A weighted thresholded matching model is used to express the component importance of a sign. This framework is independent to specific features and is open to make use of any new features.

Our detection method using bounding edges as anchor features is quite robust and efficient. It works even when object is occluded by or cluttered with other objects. We can enhance the system's performance by adding more constraints to anchor features, such as fixed shape constraints, more participating bounding edges in one anchor feature. Using these constraints, the system may fail to detect some tricky sign candidates, but overall, it can achieve detection in most cases with little effort.

This is a tradeoff between robustness and efficiency, and we usually follow the 80-20 rule (80% result comes from 20% effort).

The method depends on the detection of regions of certain color. We can relax the color threshold to make it more tolerant to color/intensity variations. Higher color distance threshold for detection means more regions will participate in the process. Or we can add more sign models to the system; each represents the same model under different lighting conditions. Either way the system will be less efficient. The actual choice of method depends on the particular property of the sign in interest. If a sign's shape is peculiar, such as one-way sign, then we might broaden the intensity tolerance, hoping detection can still do a good job based on shape constraints.

Once sign candidates (anchor features) are located, a verification process takes place to validate these candidates. We designed our verification method using a number of performance enhancement techniques. Our method calculates model features from the candidate in image and matches these feature values to those in the model. Features are calculated and matched in such an order that higher level features are used before lower level ones, and among features of one level, more significant features (features with larger weights) are used first. By using features in such an order, false positives can be removed quickly and at very low cost. In this process, heuristic termination is used to terminate the matching once a good enough match has been reached. Branch-and-bound technique is used to terminate further matching if there is no hope to find a satisfactory match in the focused area. Areas where candidates have been validated are removed from further consideration. Repeated verification on one area is thus avoided.

For the final recognition, we proposed a maximum averaged Hausdorff distance calculated from edge maps. Our test shows that it is efficient, and tolerant to slight noise, translation and rotation. Its matching result is visually appealing. Correct classification can be reached using very small number of training samples.

Using the design criteria defined in section 1.3, we conclude that our method is very efficient. It is robust in the sense that its performance degrades gracefully in extreme conditions. Detection will be slower under larger color distance threshold but will not fail in general. Our method is designed without much assumption and precondition about the environment and road sign object. As a result, new kinds of road signs can be readily incorporated into the system. The system can be used in various environments as well. We still need larger scale of testing to demonstrate the correctness of the system, but initial tests show very good results.

6.3 Future work

First of all, to improve our system, we need to test it in a larger scale. We also need to incorporate it with hardware and tune it to suit practical situation.

Looking into the future, there is still much room for improvement in almost every aspect of road sign recognition. Of the segmentation method, a more self-adaptive threshold scheme can be developed to achieve better segmentation automatically. More expressive features can be established to enable better detection and recognition. As an example, how to develop good features to express the object's shape information is an interesting topic for future study. While feature alignment is a straightforward way to detect object instances, its success depends on the property of anchor features. Developing better anchor features will be one of the most important factors to enhance the performance. We established the framework for organizing road sign features, it is more important to develop features to capture the actual characteristics of the sign for the purpose of candidate verification and recognition. More and better features at different representation level can definitely help candidate verification. Moreover, the modified Hausdorff distance is very powerful but can not reflect all aspects of a road sign. As one can see, the key to the success of every processing step of the recognition system is feature extraction, which includes feature definition and calculation.

Hardware incorporation and enhancement is necessary and is an area that definitely needs more work.

References

[1] M.Lalonde, and Y.Li, Survey of the State of the Art for Sub-Project 2.4 Road Sign Recognition, Centre de recherche informatique de Montreal, 1995

[2] A.R.Pope, Model Based Object Recognition: A Survey of Recent Research, TR94-04, 1994

[3] P.Suetens, P.Fua and A.Hanson, Computational strategies for object recognition, Computing Surveys 24(1), pp.5-61, 1992

[4] G.Piccioli, E.D.Michelli, P.Parodi and M.Campani, Robust road sign detection and recognition from image sequence, In Proc. Intelligent Vehicles, pp. 278-283, 1994

[5] G.Piccioli, E.D.Michelli and M.Campani, A robust method for road sign detection and recognition, In Proc. European Conference on Computer Vision, pp. 495-500, 1994

[6] H.Akatsuka and S.Imai, Road signposts recognition system, Proc. SAE vehicle highway infrastructure: safety compatibility, pp. 189-196, 1987

[7] M.de Saint Blancard, Road sign recognition: A study of vision-based decision making for road environment recognition, Vision-based Vehicle Guidance, Springer-Series in Perception Engineering, Springer-Verlag, 1992

[8] Lutz Priese, Volker Rehrmann, A fast hybrid color segmentation method, Institut fur Informatik, Universitat Koblenz-Landau

[9] Y.J.Zheng et al, An adaptive system for traffic sign recognition, IEEE Proc. Intelligent Vejicles'94 Symposium, pp. 165-170, 1994

[10] B.Besserer, S.Estable, and B.Ulmer, Multiple knowledge sources and evidential reasoning for shape recognition, In Proc. IEEE 4th conference on Computer Vision, pp. 624-631, 1993

[11] L.Priese, J.Klieber, R.Lakmann, V.Rehrmann, and R.Schian, New results on traffic sign recognition, IEEE Proc. Intelligent Vehicles'94, pp. 249-153, 1994

[12] G.Nicchotti et al, Automatic road sign detection and classification from color image sequences, Proc. 7th Int. Conf. on Image Analysis and Processing, pp. 623-626, 1994

[13] S.L.Horowitz and T.Pavlidis, Picture segmentation by a directed split-and-merge procedure, Proc. Second Int. Joint Conf. Pattern Recognition, pp. 424-433, 1974

[14] N.R.Pal and S.K.Pal, A review on image segmentation techniques, Pattern Recognition, Vol. 26, No. 9, pp. 1277-1294, 1993

[15] R.M.Haralick and L.G.Shapiro, Survey: image segmentation techniques, Computer Vision, Graphics, and Image Processing, 29, pp. 100-132, 1985

[16] K.S.Fu and J.K.Mui, A survey on image segmentation, Pattern Recognition, Vol. 13, pp. 3-16, 1981

[17] N.Kehtarnavaz, N.C.Griswold and D.S.Kang, Stop-sign recognition based on color shape processing, Machine Vision and Applications, 6, pp. 206-208, 1993

[18] R.Ghica, S.W.Lu and X. Yuan, Rercognition of traffic signs using a multilayer neural network, Proc. Can Conf. on Electrical and Computer Engineering, 1994

[19] D.Kellmeyer and H.Zwahlen, Detection of highway warning signs in natural video images using color image processing and neural networks, IEEE Proc. Int. Conf. Neural Networks 1994, Vol 7, pp. 4226-4231, 1994

[20] N.Bartneck and W.Ritter, Color segmentation with polynomial classification, Proc. 11th Int. Conf. on Pattern Recognition, Vol.2, pp635-638, 1992

[21] Y.Nakagawa and A.Rosenfeld, Some experiments on variable thresholding, Pattern Recognition 11, pp. 191-204, 1979

[22] N.R.Pal and S.K.Pal, Image model, Poisson distribution and object extraction, Int. J. Pattern Recognition Artif. Intell. 5, pp. 459-483, 1991

[23] A.Sarabi and J.K.Aggarwal, Segmentation of chromatic images, Pattern Recognition, 13, pp. 417-427, 1981

[24] Ron Gerson, Aspects of perception and computation in color vision, Computer Vison, Graphics and Image Processing, 32, pp. 244-277, 1985

[25] A. Rosenfeld and A.C.Kak, Digital Image Processing, Academic Press, New York, 1982

[26] E.L.hall, Computer Image processing and Recognition, Academic Press, New York, 1979

[27] R.C.Gonzalez and P.Wintz, Digital Image processing, Addison-Wesley, Reading, Massachusetts, 1987

[28] R.Klette and P.Zamperoni, Handbook of Image Processing Operators, John Wiley & Sons, 1995

[29] J.Canny, A computational approach to edge detection, IEEE Trans. PAMI-8, pp. 679-698, 1986

[30] R.Deriche, Fast algorithms for low-level vision, IEEE Trans. PAMI-12, pp. 78-87, 1990

[31] M.Hueckel, An operator which locates edges in digital pictures, JACM, 18, 1, pp. 113-125, 1971

[32] Y.Ohta, T.Kanade, T,Sakai, Color information for region segmentation, Comput. Graphics and Image Process., 13, pp. 224-241, 1980

[33] A.R.Smith, Color gamut transform pairs, Computer Graphics, Vol. 12, No. 3, 8, pp. 12-19, 1978

[34] G.S.Robinson, Color edge detection, Proc. SPIE Symposium on Advances in Image Transmission Techniques, 87, San Diego, CA., August 1976.

[35] T.Carron and P.Lambert, Color edge detector using jointly hue, saturation and intensity, IEEE Int. Conf. On Image Processing, Vol. 3, pp. 977-981, 1994

[36] P.J. Besl and R.C. Jain, Segmentation through variable-order surface fitting, IEEE Trans. PAMI, Vol. 10, No. 2, pp. 167-192, 1988

[37] B.C. Li and J. Shen, Fast calculation of local moments and application to range image segmentation, , Proceedings 11th IAPR International Conference on Pattern Recognition, Volume III, Conference C. pp. 298-306, 1992

[38] P.C.Chen and T.Pavlidis, Image segmentation as an estimation problem, Comput. Graphics Image Process. 12, pp. 153-172, 1980

[39] J.D.Browning and S.L.Tanimoto, Segmentation of pictures into regions with a tile by tile method, Pattern Recognition 15, pp. 1-10, 1982

[40] J.L.Muerle and D.C.Allen, Experimental evaluation of techniques for automatic segmentation of objects in a complex scence, Pictorial Pattern Recognition, G.C.Cheng et al., Eds, pp 3-13, Thompsons, Washington, 1968

[41] T.Pavlidis, Segmentation of pictures and maps through functional approximation, Comp. Graphics Image Process. 1, pp. 360-372, 1972

[42] M.Kocher anc R.Leonardi, Adaptive region growing technique using polynomial functions for image approximation, Signal Processing 11, pp. 47-60, 1986

[43] J.P. Gambotto and O. Monga, A parallel and hierarchical algorithm for region growing, Proceedings, CVPR '85, pp. 649-652.

[44] C.Brice and C.Fennema, Scene analysis using regions, Artificial Intelligence 1, pp. 205-226, 1970

[45] T.Westman, D.Harwood, T.Laitinen, and M.Pietikainen, Color segmentation by hierarchical connected components analysis with image enhancement by symmetric neighborhood filters, Proc. 10th ICPR, 796-802

[46] T.Pavlidis and Y.T.Liow, Integrating region growing and edge detection, IEEE Trans. PAMI, 12 (3), pp. 225-233, 1990

[47] A.Rosenfeld and A.C.Kak, Digital Picture Processing, Vol. 1, Academic press, New York, 1982

[48] L.S.Davis and A.Rosenfeld, Noise cleaning by iterated local averaging, IEEE Trans. Syst. Man Cybern. 8. pp. 705-710

[49] D.harwood et al, A new class of edge-preserving smoothing filters, pattern Recognition Letters, 6, pp. 155-162, 1987

[50] T.Westman, D.Harwood, T.Laitinen, and M.Pietikainen, Color segmentation by hierarchical connected components analysis with image enhancement by symmetric neighborhood filters, Proc. 10th ICPR, 796-802.

[51] T.O.Binford, Survey of model-based image analysis systems, Int. J.Robotics Res. 1(1), pp. 18-64, 1982

[52] R.J.Woodham, Stable representation of shape, In Computational Processes in Human Vision, ed. By Z.Pylyshyn, Norwood, N.J. Ablex, 1987

[53] R.M.Haralick, A.K.Mackworth and S.L.Tanimoto, Computer vision update, In Handbook of Artificial Intelligence, ed. By A.Barr, P.R.Cohen and E.A.Feigenbaum, Reading, Mass. AddisonWesley, 1988

[54] F.Mokhtarian and A.K.Mackworth, A theory of multiscale, curvature-based shape representation for planar curves, IEEE Trans. PAMI. 14(8), pp. 789-805, 1992

[55] E.Saund, The role of knowledge in visual shape representation, Ph.D. dissertation, Cambridge, MIT. 1988

[56] N.Ayache and O.D.Faugeras, HYPER: A new approach for the recognition and positioning of two-dimensional objects, IEEE Trans. PAMI. 8(1), pp. 44-54, 1986

[57] C.H.Chen and A.C.Kak, A robot vision system for recognizing 3-D objects in low-order polynomial time, IEEE Trans. Syst. Man Cybernetics, 19(6), pp. 1535-1563, 1989

[58] R.A.Brooks, Symbolic reasoning among 3-D models and 2-D images, Artificial Intel. 17, pp. 285-348, 1981

[59] R.Bergevin and M.D.Levine, Generic object recognition: Building and matching coarse desciptions from line drawings, IEEE Trans. PAMI, 6(2), pp. 19-36, 1993

[60] S.J.Dickinson, A.P.Pentland and A.Rosenfeld, 3-D shape recovery using distributed aspect matching, IEEE Trans. PAMI, 14(2), pp. 174-198, 1993

[61] G.G.Gordon, Face recognition based on depth and curvature fratures, In Proc. IEEE Conf. Comput. Vis. Patt. Recogn., pp. 808-809, 1992

[62] W.E.L.Grimson, Object Recognition by Computer: The Role of Geometric Constraint. MIT Press, 1990

[63] J.Ben-Arie and A.Z.Meiri, 3D objects recognition by optimal matching search of multinary relations graphs, Comput. Vis. Graphics Image Processing, 37, pp. 345-361, 1990

[64] R.Bergevin and M.D.Levine, Extraction of line drawing features for object recognition, Patt. Recogn., 25(3), pp. 319-334, 1992

[65] E.K.Wong, Model matching in robot vision by subgraph isomorphism, Patt. Recogn., 25(3), pp. 287-304, 1992

[66] S.Zhang, G.Sullivan and K.Baker, The automatic construction of a view-independent relational model for 3-D object recognition, IEEE Trans. PAMI, 15(6), pp. 531-544, 1993

[67] B.Bhanu and O.D.Faugeras, Shape matching of two-dimensional objects, IEEE Trans. PAMI., 6(2), pp. 137-156, 1984

[68] L. Kitchen, Relaxation applied to matching quantitative relational structures, IEEE Trans. Syst. Man Cybernatics, 10(2), 96-101, 1980

[69] Y.Lamdan and H.J.Wolfson, Geometric hashing: A general and efficient model-based recognition scheme, Proc. IEEE Int. Conf. Comput. Vision, pp. 218-249, 1988

[70] D.P. Huttenlocher and S. Ullman, Object recognition using alignment, First International Conference on Computer Vision, London, England, June 8-11, 1987 (Proceedings published by IEEE Computer Society Press, Washington, DC.), 102-111

[71] Y.Kuno, Y.Okamoto and S.Okada, Robot vision using a feature search strategy generated from a 3-D object model, IEEE Trans. PAMI., 13(10), pp. 1085-1097, 1991

[72] W.E.L. Grimson and T. Lozano-Perez, Recognition and localization of overlapping parts from sparse data, In T. Kanade, ed., Three-Dimensional Machine Vision, Kluwer, Boston, MA, pp. 451-510, 1987

[73] W.E.L. Grimson and D.P. Huttenlocher, On the sensitivity of the Hough transform for object recognition, IEEE Trans. PAMI, 12(3), pp. 255-274, 1990

[74] D.G.Lowe, Three-dimensional object recognition from single two-dimensional images, Artificial Intell. 31, pp. 355-395, 1987

[75] D.G.Lowe, The viewpoint consistency constraint, Int. J. Comput. Vision, 1, pp. 57-72, 1987

[76] N.Ayache and O.D.Faugeras, HYPER: A new approach for the recognition and positioning of two-dimensional objects, IEEE Trans. PAMI., 8(1), pp. 44-54, 1986

[77] M.J.Swain and D.Ballard, Indexing via color histograms, In IEEE Proc. Third Conf. Computer Vision, IEEE, pp. 390-303, 1990

[78] M.J.Swain and D.Ballard, Color histograms, Int. J. Computer Vision, 7(1), pp. 11-32, 1991

[79] B.V.Funt and G.D.Finlayson, Color constant color indexing, IEEE Trans. PAMI., 17(5), pp. 522-529, 1995

[80] G.Healey and D.Slater, Global color constancy: recognition of objects by use of illumination-invariant properties of color distributions, I. Opt. Soc. Am. A., 11(11), pp. 3003-3010, 1994

[81] F. Stein and G. Medioni, Structural indexing: Efficient 3-D object recognition, IEEE Trans. PAMI., 14(2), pp. 125-145, 1992

[82] D.P. Huttenlocher, G.A. Klanderman, and W.J. Rucklidge, Comparing images using the Hausdorff distance, IEEE Trans. PAMI, 15(9), pp. 850-863, 1993

[83] M.P. Dubuisson and A.K. Jain, Modified Hausdorff distance for object matching, Proceedings 12th IAPR International Conference on Pattern Recognition, vol. 1, pp. 566-568, 1994

[84] S.Estable et al, A real time traffic sign recognition system, Proc. Intelligent Vihecles, pp. 213-218, 1994