Multimedia Technologies for Landmark-Based Vehicle Navigation

Wen Wu

CMU-LTI-09-014

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:

Jie Yang (Chair)
Daniel P. Siewiorek
Alexander G. Hauptmann
Thomas Seder

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

To my family and friends.

Abstract

Most existing Global Positioning System (GPS)-based vehicle navigation systems (also termed route guidance systems) utilize distance within their turn-by-turn navigation directions. For example, a system might give a voice instruction like ”turn left in 0.2 mile”. However, human drivers usually use landmarks to help their navigation. Some previous research has shown that there are performance-related benefits in using landmarks instead of distance for navigation. The goal of this dissertation is to develop multimedia techniques that can be used for achieving landmark-based navigation using computer vision and machine learning techniques.

This dissertation makes contributions in landmark labeling, detection, recognition and the human vehicle interfaces. Landmark labeling is essential for development of landmark recognition systems and using a landmark-based navigation system. The first contribution of this dissertation is a semi-supervised learning-based approach for labeling landmarks in images. The proposed approaches, SmartLabel and SmartLabel-2, minimize user input in labeling landmarks. Text on road signs carries much useful information for driving. The second contribution is an automatic system which detects text on road signs from driving videos. The third contribution is a novel approach for recognizing a given street landmark such as a store sign in a sequence of images. We develop a street landmark recognition system that combines salient region detection, segmentation, and object fingerprint extraction techniques. The fourth contribution is a landmark building recognition framework which is able to recognize and localize the target building in dynamic driving data. Navigation user interfaces have changed dramatically over the last decade due to available electronic maps and GPS devices. However, current navigation systems without landmarks have

not fully achieved user satisfaction. The fifth contribution of this dissertation is to demonstrate the concept of landmark-based vehicle navigation on a computer display and also show a prototype using a full-windshield head-up display system.

Acknowledgments

During my study at Carnegie Mellon University, I have met many people who have provided priceless advice, support and care to me and this work. Without them, I would not have been able to finish the dissertation and study.

First of all, I would like to thank my advisor, Jie Yang, for his great guidance and numerous support to my research and life. I have learned from Jie not only how to conduct research, how to solve real world problems, how to collaborate with other researchers, etc but also the way of balancing family life and work. I am especially thankful for his firm support when I stumbled in past years. I am really glad to have the opportunity to work with Jie Yang on my Ph.D. dissertation.

I would also like to thank my other thesis committee members. Thomas Seder have been supporting my research projects for years. I am thankful for his insightful questions and thoughtful comments on the dissertation. Daniel P. Siewiorek gave me not only valuable advices on the dissertation, but also timely feedback at various phrases of this dissertation. Alexander G. Hauptmann has greatly broadened my understanding of the problems in my thesis in the context of multimedia content analysis. I have enjoyed working with all of them, and benefited greatly from the interactions with them.

At Carnegie Mellon many research projects are sponsored by government agencies, companies and etc. General Motors Company has been the major sponsor to my research and study at Carnegie Mellon. Without the support, I would not have been able to start this dissertation. I would also like to thank other sponsors to my work: Microsoft Corporation, US National Science Foundation

(NSF), National Institutes of Health (NIH) and others.

I spent six and half years at Carnegie Mellon University. I want to thank the following collaborators, faculties, staffs, fellow students and friends, who made my life here a pleasant and memorable experience: Xilin Chen, Jing Zhang, Rong Yan, Yan Liu, Jiazhi Ou, Jian Zhang, Zhong Xiu, Mei Chen, Fabian Blaicher, Dehua Cui, Kapil Dev Dhingra, Rahul Sukthankar, Vitor R. Carvalho, Manuela Veloso, M. Bernardine Dias, Laura Forsyth, Zhonghuai Wang, Xiaojin Zhu, Datong Chen, Haiyan Wang, Rong Jin, Szu-Chen Jou, Jun Yang, Robert Chen, Yifen Huang, Brian Langner, Luo Si, Lingyun Gu, Wende Zhang, Hua Yu, Fan Li, Ying Zhang, Hugo Cheng, Terence Liu, Mickey Chou, Yik-Cheung Tam, Yansong Wang, Lin Nan, Jun Chen, Liying Ren, Jay Pan, Mingang Xu, Fei Yu, Fengling Song, Yanan Wang, Chih-Cheng Lin, Judy P. Xi, Sherry Xue, Gong Zhang, Zipei Tu, Weiting Zhang, Wang Liu, Ni Lao, Shijun Shen, Jessica Wang, Hui Yang, Jian Jiao, Shinjae Yoo, Jae Dong Kim, Richard Wang, Frank Lin, Simon Fung, Le Zhao, Xiangmin Jin, Jiangbo Miao, Liyang Zhang, Roger Hsiao, Antoine Raux, Shenyu Yan, Wenjie Fu, Qin Gao, Runxin Li, Weisi Duan, Yangbo Zhu, Yi Wu, Yutao He, Jiahai Zhou, Yuyu Huang, Ying Wang, Ningzhe Zhang, Ping Zhou, Handi Hu, Ting Liu, Ke Yang, Fei Huang, Kang Li, Yan Li, Lie Gu, Kuanquan Wang, Dejian Ye, Xiuyan Jiang, Hong Cheng, Xinghua Sun, Honggang Zhang, Lei Yang, Jinghao Fei, Qing Wang, Huajun Chen, Qin Jin, Yiheng Li, Yi Zhang, Yi Chang, Bing Zhao, Yanjun Qi, Qifa Ke, Zhenzhen Kou, Jimeng Sun, Gang Hua, Elaine Shi, Jie Lu, Eric Xing, Alan Black, Robert E. Frederking, Yiming Yang, Radha Rao, Stacey Young, Dana Houston, Linda Hager, Mary Jo Bensasi, Brooke Hyatt, Aaron Steinfeld, Matthew Turk, Shih-Fu Chang, Mubarak Shah. My apologies if I left your name out. I also want to thank you if you are reading this dissertation.

Finally, I sincerely thank my family. My parents raised me with their hearts and love. My brother always encouraged me. My dear wife always supports me no matter joy or hardship, success or failure. My relatives and other friends also deserve my thanks.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Navigation is the process of planning, recording, and controlling the movement of a craft or vehicle from one place to another [1]. Navigating a vehicle in a dynamic environment is one of the most demanding activities for drivers in their daily lives. Americans drive 12,000 miles per year on average. Studies have long identified the difficulties that drivers have in planning and following efficient routes [62, 108, 115].

A vehicle navigation system (also termed route guidance system) is usually a satellite navigation system designed for use in vehicles. Most systems typically use a combination of Global Positioning System (GPS) and digital map matching to calculate a variety of routes to a specified destination such as the shortest route. They then present a map overview and turn-by-turn instructions to drivers, using a combination of auditory and visual information. A typical turn-by-turn instruction is an auditory prompt such as "in 0.5 mile turn right", accompanied by a visual right turn arrow plus a distance-to-turn countdown that reduces to zero as the turn is approached. Vehicle navigation systems generally function well, although they are wholly dependent on the accuracy of the underlying map database and availability of GPS signals. However, from a human factor perspective, there are several potential limitations to the current design [79]: mainly presenting

[1] http://en.wikipedia.org/wiki/Navigation#_ref-bow799_0

procedural and paced navigation information to the driver, and relying on distance information to enable a driver to locate a turn.

1.1 Motivation

Human drivers often use landmarks for navigation. For example, we tell people to turn left after the second traffic light and to make a right at Starbucks. In our daily lives, a landmark can be anything that is easily recognizable and used for giving navigation directions, such as a sign or a building. It has been proposed that current navigation systems can be made more effective and safer by incorporating landmarks as key navigation cues [14]. Especially, landmarks support navigation in unfamiliar environments. By providing external reference points, which are easily remembered and recognized, landmarks can potentially reduce the need to refer to an information display in order to locate a navigation decision point.

The definition of landmark in navigation context has been studied from varying theoretical perspectives. Lynch described landmarks as external reference points that are easily observable from a distance [77]. Kaplan defined a landmark as ”a known place for which the individual has a well formed representation”, and described two theoretical factors that lead to a object or place acquiring landmark status: the frequency of contact with the object or place, and its distinctiveness [57]. Three types of distinctiveness were proposed: visual distinctiveness (a predominantly objective quality relating to the physical attributes that discriminate a landmark from the surrounding environment); inferred distinctiveness (knowledge concerning its structure or form that makes the landmark stand out from what is usual); functional distinctiveness (the salience in terms of the goals or sub-goals of the landmark). In addition to the visual characteristics of landmarks and their functional or social importance, the location of an object within the environment has also been shown to impact significantly on its effectiveness as a landmark [17, 2]. Burnett identified 4 out of 11 attributes that were most important characteristic of ’good’ landmarks for vehicle navigation systems [14]:

1. *Usefulness of location:* the ease with which the location of the landmark allows a navigational maneuver (e.g., a turning) to be identified.
2. *Visibility:* whether the landmark's size and shape can be clearly seen in all conditions.
3. *Uniqueness:* whether the appearance of the landmark is such that it is unlikely to be mistaken for anything else.
4. *Permanence:* the likelihood of the landmark being present.

Based on the human factor studies using the above attributes Burnett has further identified the top ten scoring landmark types in United Kingdom (UK) [14]:

1. Traffic lights
2. Pelican crossing
3. Bridge over road
4. Hump-backed bridge
5. Petrol station
6. Monument
7. Superstore
8. Street name signs
9. Railway station
10. Church

It is quite evident that several of these landmarks are UK specific. In the United States, we observe that common navigation-useful landmarks include 1) road signs, 2) other signs (including signs of gas stations, fast food restaurants, stores, subway stations, etc) and 3) buildings (including churches, stores, etc). This is our observation. Fig.1.1.1 depicts these three main classes of landmarks in USA, which are the focus of this dissertation. Sometimes, landmarks can also be areas where memorable event occurred. In our study, we only focus on landmarks from these three classes.

The potential benefits of landmarks are well established. A range of studies have empirically demonstrated how landmarks have the potential to enhance vehicle navigation systems in terms of: (1) effective navigation decisions [109], (2) reduced cognitive effort and distraction [14], and (3) increased confidence and satisfaction [45]. However, little research has been published that has developed technologies for detecting and recognizing landmarks for vehicle navigation systems.

Figure 1.1.1: Three main classes of street landmarks in US, which are the focus of this dissertation.

1.2 Dissertation Statement

In this dissertation, we aim to develop technologies for landmark based navigation for the following scenario. John is going to visit his friend Susan in another city. Susan sends John some street landmark images of her city on his route to her place. John will drive a car with a video camera that can capture the scene in front of the car. We would like to build such a system that could help John to automatically recognize landmarks from the video sequence. As we know, the training image and the test image sequence normally have different resolution and quality. An example of a fast food restaurant sign, Taco Bell, is shown in Fig.1.2.1.

To implement such a landmark-based navigation system, I organize my dissertation research around five main problems:

1. *Labeling street landmarks in images with minimal human effort* (Chapter 2). Manually labeling image data is labor intensive and time demanding. These difficulties motivate us to turn to a semi-supervised learning approach.
2. *Automatically detecting text on road signs from video* (Chapter 3). Correctly detecting text on road signs can be very useful and poses many challenges. Video images are relatively low

Figure 1.2.1: Example images of Taco Bell sign. Training image is 1280×960, while test image captured by a video camera is 320×240. Two images are re-scaled to be equal size. In this example, it is even hard for human to recognize the Taco Bell sign from the clutter on the right.

resolution and noisy. Both the background and foreground of a road sign can be complex and change frequently in video. Our proposed approach uses spatio-temporal information in video and fuses partial information for detecting text from frame to frame.

3. *Recognizing street landmarks (other signs)* (Chapter 4). General object recognition algorithms may not work well for this problem because of its unique challenges that include lack of training data, abrupt change of viewpoint and illumination, occlusion, etc. Our approach extracts salient features of the landmark as its representation and performs recognition using the salient representation.

4. *Recognizing landmark buildings* (Chapter 5). Landmark buildings are an important class of street landmarks which can be used as reference points for navigation. We have developed a robust landmark building recognition framework which is able to recognize and localize the target building in diverse views of driving data.
5. *Demonstrating the proposed technologies on a full-windshield head-up display system (FWD)* (Chapter 6). We have developed a preliminary concept system on a computer display and implemented a prototype of a landmark-based navigation system using a FWD.

The dissertation statement is, we aim to develop multimedia technologies for landmark-based vehicle navigation. We focus our attention on the following three main classes of landmarks (Fig.1.1.1): 1) road signs, 2) other signs and 3) buildings. We present technologies for labeling, detecting and recognizing these landmarks from images and videos. In addition, we implement a prototype of the resulting system using a full-windshield head-up display system.

Fig.1.2.2 illustrates the role of the dissertation in the field of landmark-based vehicle navigation. In the real world, a vehicle sees the three main classes of landmarks through an in-vehicle camera. The dissertation is to develop multimedia technologies for labeling, detecting and recognizing these landmarks from images and videos. We use a FWD to implement a prototype, but driver perception, or in other words, visualization is not a focus of this dissertation. Note the black arrows in Fig.1.2.2 are the processes which we do not study in this dissertation.

1.3 Expected Contributions

In order to learn a discriminative model of the landmark of interest for recognition, we need to first label the landmark versus its background in a given image. Manually labeling images is not only a labor intensive task, but also subject to human labeling and annotation errors. While efforts have been focused on online massive user labeling (e.g. MIT LabelMe [2], The ESP Game [3]),

[2] http://labelme.csail.mit.edu/

[3] http://www.gwap.com

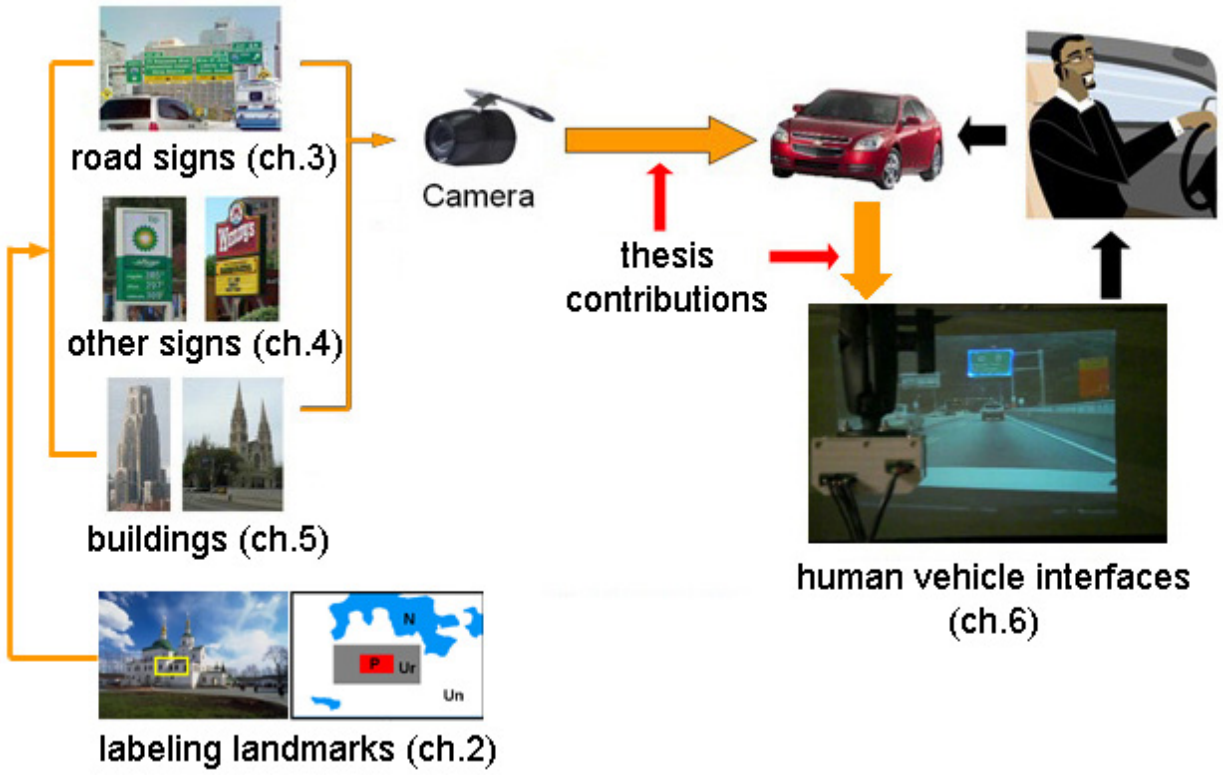

Figure 1.2.2: Thesis contributions and chapter layout.

limited attention has been paid to semi-automatically labeling objects in images or videos [4, 5]. Our proposed SmartLabel and SmartLabel-2 aim to let a user only mark a small region of interest inside the landmark (or object) on the image with simple input (e.g. dragging a rectangle), and our algorithms can then label the rest of the landmark (object) in the image [121, 123]. The evaluation of proposed SmartLabel-2 and comparison with other methods on a dataset of six object classes indicate that SmartLabel-2 not only works effectively with a small amount of user input (e.g., 1 .. 5% of image size) but also achieve very promising results (avg_F_1=0.84). In some cases, SmartLabel-2 even obtain nearly perfect performance.

We have introduced a novel framework that can incrementally detect text on road signs from video. The proposed framework takes advantage of spatio-temporal information in video and fuses partial information for detecting text from frame to frame. Text on road signs carries much useful

information for driving and describes the current traffic situation, defines right-of-way, provides warnings about potential risks, and permits or prohibits roadway access. Automatic detection of text on road signs can help to keep a driver aware of the traffic situation and surrounding environments by highlighting signs that are ahead and/or have been passed [119, 120, 118].The feasibility of the proposed framework has been evaluated using video sequences captured from a moving vehicle. This new framework gives an overall text detection rate of 88.9% and a false hit rate of 9.2%.

We further focus on recognition of other signs with limited training data. Street landmarks (other signs) are always characterized by unique characters. Each landmark has its own characters which we cannot find in other objects. We call such unique characters the landmark's object fingerprints. We define the problem of recognizing a particular street landmark as recognition of the landmark's object fingerprints. Instead of modeling and matching the landmark as a whole, our proposed approach extracts the landmark's object fingerprints in a given image and matches to a new image in order to recognize the landmark. We formulate recognition of the landmark's object fingerprints as a classification problem solved by a cascade of 1-nearest neighbor classifiers. We develop a street landmark recognition system that combines salient region detection, segmentation, and object fingerprint extraction techniques [122]. To evaluate, we have compiled a novel dataset which consists of 15 U.S. street landmarks' images and videos. Our experiments on this dataset show superior performance (avg_F_1=0.67) to state-of-the-art recognition algorithms. We believe with additional GPS information, our system can achieve better recognition performance.

Landmark buildings are an important class of street landmarks which can be used as reference points for driving navigation. We have developed a robust landmark building recognition framework which is able to recognize and localize the target building in diverse views. The recognition results can be further used for vehicle navigation. Evaluation on both a public dataset and our self-collected Pittsburgh Historic Landmark dataset has shown promising results (with accuracy equal to 0.94).

We have developed a preliminary landmark based navigation system on computer display. The

system gives driving instructions by overlaying navigational arrows onto live video and providing synthesized voice, in addition to highlighting key landmarks for coming maneuvers [124]. Furthermore, we have implemented a prototype of a landmark-based navigation system using a full-windshield head-up display system [117].

1.4 Related Works

In this section, we review some previous research related to landmark-based vehicle navigation in different research fields in the order of research problems that are presented in this dissertation.

Landmark Labeling

Labeling landmarks or objects in images can also be called image segmentation, foreground extraction, object extraction or image editing, although they are slightly different in terms of applications and domains. Image segmentation has been an active area of research for decades and its application has been widely adopted in many research fields including content-based image retrieval [106, 114, 32]. The goal is to create systems capable of segmenting foreground objects from the background accurately and to achieve good segments of the image. Recently, research has focused on the problem of *interactive* extraction of a foreground object from an image [90].

There are three key differences between our proposed SmartLabel framework and other state-of-the-art interactive segmentation tools. First, the goal of SmartLabel is to create labels for interesting object(s) in the image, not to segment the image into a number of blobs. Secondly, although SmartLabel is a semi-automatic labeling tool, it does not rely too much on user input; only initial specification and relevance feedback after the first iteration are required. Finally, SmartLabel can extract a foreground object at multiple locations in the image even though the user only specifies part of the object at one location, but most iterative segmentation tools extract objects within or around the user specified region of interest. In the chapter of landmark labeling, we will describe

several state of the art interactive object extraction (segmentation) tools: Magic Wand, GrabCut [90], ClickRemoval [82], and central object extraction [61].

Detection of Text on Road Signs

Based on its origin, text in video can be classified into two classes: graphic text and scene text [72]. Graphic text is text that is added to the video after the video is captured, such as captions added to news videos. Scene text exists as part of objects in a natural environment when it is directly captured by a camera, which includes billboards, and street names on road signs. A common assumption used by previous research in graphic text detection from video is that the text plane is perpendicular to the optical axis of the camera [54, 69]. This is suitable for some domains such as broadcast video where the camera is fixed or has relatively little motion. However, the assumption does not necessarily hold in the scene text detection task since road sign planes are often encountered at a non-perpendicular angle with respect to the camera optical axis.

More general techniques for detecting scene text from still images have been developed in pattern recognition and computer vision fields. Recently, some researchers were able to detect scene text from still images [21, 26] and reported that edge features can better handle lighting and scale variations in scene images than texture features [22], which are often used for detecting text in news video [54, 60]. Inspired by their work, we chose to use the edge-based features for text detection in this study. Myers et al. described a full perspective transformation model to detect 3D deformed text from still images [81].

Research on extracting scene text from video has informed our work. Fang et al. introduced a dynamic visual model for recognizing road signs in video but was limited to road sign symbols, such as those for ”stop” and ”do not enter” instead of text [33]. Haritaoglu and Haritaoglu used a combination of symmetric neighborhood filtering and hierarchical connected component analysis to extract written information on road signs in scene images [48]. Piccioli et al. used a priori knowledge on scene and color clues to search suitable regions for road signs in images [84]. This

approach works for images of cluttered urban streets as well as country roads and highways. Miura et al. designed a two-camera system consisting of a wide-angle-lens camera and a telephoto-lens. The wide-angle-lens is used to detect candidates for road signs using color, intensity, and shape features, and the telephoto-lens is directed to the road sign to capture a high-resolution image of the candidate [80]. Gandhi et al. applied a plane motion model to correct the perspective distortion of the text planes for robust detection [42]. Vitabile et al. proposed a method for focusing on detecting road sign symbols instead of text by using multi-layer perception neural network and image data to evaluate the system [112].

Recognition of Other Signs

Recognizing a street landmark (a sign) from a moving camera is a difficult task because of combined effects of camera motion, blur, and constantly changing illuminations. In addition, only one landmark image is provided in advance which invalidates many successful object recognition techniques such as algorithms requiring a certain amount of training data. While face recognition from a single image has been extensively studied [126] and detection of pedestrians and vehicles has been successfully demonstrated [67], little attention has been focused on recognition of street landmarks.

To achieve robust object recognition, many interesting approaches have been proposed [8, 16, 28, 36, 116]. Also object detection has made tremendous progress over past years in various directions: deformable object detection [85], application of geometric scene context for detection [50], use of object boundary for detection [83], and exploration of geometrical relationship between object components [96]. Location recognition has also been studied [98]. In the context of object recognition from one image, researchers have proposed different methods [75, 38, 91] to exploit object appearance and geometrical information. The previous work, however, requires modeling of the *whole* object, and recognition is performed by matching the learned object model to a new image. In contrast, our approach represents an object by a set of fingerprints and allows matching fingerprints instead of the *whole* object. Bigrams and triplets of feature descriptor have been shown

to help recognition which motivates our work [65, 131]. A bag of SIFT features for an image is used in a text retrieval approach for object matching in videos [105]. It is also worth mentioning very recent work by [86, 59, 94] which employ SIFT features to match landmark images or logos.

The closest work to ours is Ferencz et al. [37] which has only one image for training but ample data to train a category classifier. Although the two approaches agree in general spirit, our approach provides a different solution to recognize street landmarks from one training image. But [37] focuses on car and face instances. Furthermore, our approach applies multiple segmentations to extract patches, which has been shown to give better spatial support for recognition [78] than regular patches, which are used by [37].

Recognition of Landmark Buildings

The recognition of landmark buildings as considered in this dissertation comprises two phases: building representation and recognition of the target building. The problem of building recognition has attracted much attention in the past, mostly considering outdoors scenes. Some researchers formulate and tackle the problem in a content-based image retrieval manner [53, 71]. Other researchers proposed using vanishing direction for alignment of a building view in the query image to the canonical view in the database and proposed matching using interest regions' descriptors, followed by the relative pose recovery between the views from planar homographies [89]. As mentioned in the paper, the methods which employ solely geometric and local feature based matching techniques are often slow. In [100] authors proposed extracting invariant regions and used a set of color moment invariants to represent them. Recognition was performed based on the number of matched regions. In [110], an alternative approach was proposed to the context-based place recognition problem. The representation of individual locations was obtained by integrating responses of the bank of filters over coarse spatial regions and fitting a Gaussian mixture model to the responses. This approach enabled coarse classification of locations and also exploited spatial relationships between locations captured by a Hidden Markov Model. However, the location model did not allow for actual pose recovery of the camera with respect to the scene.

In the context of object recognition, both global and local image descriptors have been considered. Commonly used global descriptors, which provide some invariance to occlusions and clutter proposed in the past, include gist features [87] and multi-dimensional histograms [3]. The representatives of local image descriptors include scale invariant features [75] and their descriptors, which are invariant with respect to rotation, scale change and affine transformations. From the perspective of the application, the efficiency of the approach has to be considered. Therefore, when dealing with large databases, it is desirable to have some simple indexing vectors for all models, so that unlikely models can be eliminated in advance. More recent research on detection and recognition of buildings has been reported in [71, 64, 125, 98, 127].

In-Car Navigation Systems

Driving assistant systems are popular applications of multimedia technologies. A combination of GPS and electronic maps has led to revolutionary changes in car navigation systems. These systems provide drivers with an efficient route planning tool through a map database and allow convenient route guidance by GPS satellite signals. Users of such navigation systems can take advantage of such digital navigation services not only in their daily lives but also in unfamiliar areas or in regions with complex road-layouts and intersections.

However, state-of-the-art navigation systems have not yet reached their full potential and their use is often not intuitive and helpful to users with different abilities. There are two main kinds of navigation interfaces in existing systems, i.e., voice commands and abstract map images on a display, either integrated in the dashboard or being portable. Both interfaces have proven useful and effective. For example, current navigation systems can increase a driver's cognitive load because the driver has to map the information provided by the navigation system to the real environment outside the windshield. Another distracting effect of current navigation systems is that the driver has to move his/her attention away from the road to perceive navigation information.

The head-up display (HUD) is a type of display that presents data without blocking the user's

view. The technique was pioneered for military aviation and now used in commercial aviation, motor vehicle and other cases. Since 1988, General Motors has popularized HUDs on the Olds Cutlass, Pontiac Grand Prix, Bonneville, Buick LeSabre, Park Avenue and Rendezvous. In 1999, Automotive HUD technology made a big quality leap with the Chevrolet Corvette. In the new Corvette, which uses a HUD to display vehicle speed, engine RPM, navigation and more, the HUD has proven to be one of the most popular options. As of 2006, BMW featured the HUD as an option on their 5 and 7 series vehicles, with more HUDs anticipated from other European and Japanese OEMs. Despite the limited resolution and current generation HUDs' sizes, they provide information such as speed, turn by turn navigation and warnings via a virtual image projected onto the windshield by mirrors within the driver's normal field of vision. By adjusting the number of mirrors, display color and illumination elements, installation space requirements and costs can be adapted to respective vehicle models.

1.5 Chapters of the Dissertation

In the Chapter 2, we will discuss semi-automatically labeling landmarks and objects in images. In the Chapter 3, we will describe detection of text on road signs from video. In the Chapter 4 and 5, we present the methods for recognition of other signs and landmark buildings. In the Chapter 6, we will introduce our work on human vehicle interface including a prototype using a full-windshield head-up display system. Finally, we will conclude this dissertation and present some future works in the Chapter 7.

Chapter 2

Semi-Automatically Labeling Landmarks and Objects in Images

In this chapter we focus on the problem of labeling objects in images with application to landmark labeling. The fast growth of visual data on the Internet has created new challenges for the image processing (IP) community. Most IP tasks such as image annotation [113], content based image retrieval [92, 106, 114, 56, 30, 23] and object detection & recognition [111, 70], require training data. Manually labeling images is not only a labor-intensive and time-consuming task but also subject to human operation errors and variances. There has been much attention on attracting Internet users to label images manually such as MIT LabelMe [93] and ESPGame.org, but research on the semi-automatical manner has been limited [4, 5]. Our goal is to offer a semi-automatic framework to label objects in images effectively (focusing on *things* as opposed to *stuff* [18]). We have introduced a family of semi-automatic labeling methods in this study based on a graph-based semi-supervised learning algorithm [130].

To appreciate the difficulties in manually labeling objects, let us go through the labeling process of LabelMe [93]. The users need to trace object outlines of as many objects in the image and for as many images as they like. Often users click their mice greater than 30 times along the

outline to finish working on one object. Within 20 months since its inception, about 55 thousand polygons are manually labeled among which about 5 thousands are described. LabelMe's impact is influential, but the numbers of labeled objects and descriptions are small compared to the amount of contributed human efforts. These difficulties motivate us to seek semi-automatic methods to label objects in images. One solution is to let a user mark a small region of interest (ROI) on the object (e.g., dragging a rectangle) and the computer automatically extracts the object outline and its other instances in the image. Figure 2.0.2 shows examples of user inputs and results by SmartLabel [121] we have proposed and GrabCut [90].

Some research has been done in semi-automatic object extraction using various techniques [18, 90]. Semi-supervised learning (SSL), which leverages the availability of unlabeled data to

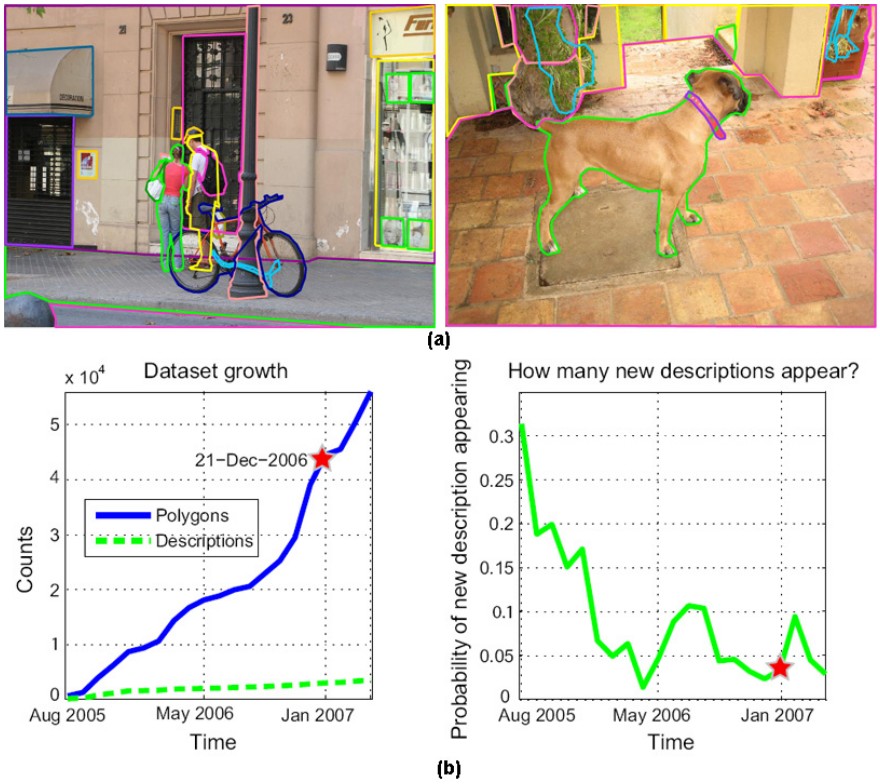

Figure 2.0.1: MIT LabelMe [93]. (a) Two example images. (b) Evolution of LabelMe database over time. Left: total number of polygons and descriptions over time. Right: the probability of a new description being entered into the database over time.

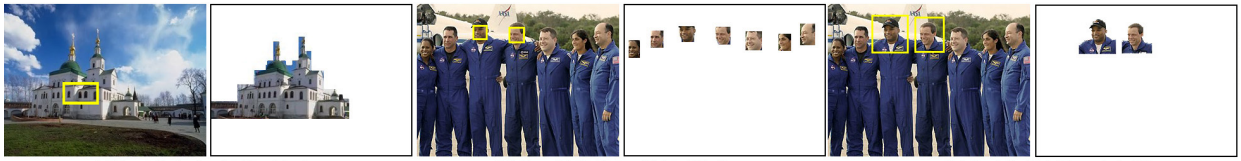

Figure 2.0.2: An illustration of SmartLabel [121] and a comparison with GrabCut [90]. SmartLabel allows users loosely drag a rectangle (or a loop stroke) inside an object. It can label both single and multiple objects. In each group, left are input images with user specified ROI (in yellow) and right are extracted object regions. Note GrabCut asks users to mark an ROI outside of the object in the image.

improve classification, has attracted a lot of attention in the past decade and been proven useful for many problems [101, 10, 11, 9, 130, 6, 128]. A survey on SSL can be found in [129]. We focus on addressing the problem within an SSL framework. We formulate the problem as follows. Given an image, an ROI is provided by the user. We divide the image into non-overlapping square patches; any patches overlapping the ROI are considered as labeled samples and put in L and the rest are considered as unlabeled samples and put in U. Formulating the problem in this way allows us to apply any SSL algorithm to classify patches in U. However, the formulation still presents two challenges. First, labeled patches contain only positive samples. Second, L may contain noise because patches which overlap ROI can include background. These two challenges become critical hurdles for many existing SSL algorithms.

We have come to a recently proposed graph-based SSL method [130], which we refer to as Zhu's SSL method in the following. Zhu's SSL method represents labeled and unlabeled samples as vertexes in a weighed graph. Edge weights represent the similarity between connected vertexes. It adopts Gaussian fields over a continuous state space rather than random fields over a discrete label set. The mean of the field is characterized in terms of harmonic functions and its solution can be efficiently obtained using matrix methods or belief propagation. This *relaxation* to a continuous space has some attractive properties. For instance, lack of negative samples and continuous label values can be naturally handled by Zhu's SSL method. However, Zhu's SSL method was

proposed for problems in other domains, and there are still difficulties in directly applying it to our problem. In [121], we have proposed SmartLabel based on Zhu's SSL method. SmartLabel has four novelties: 1) Real numbers from 0 to 1 are used as label values for positive patches. 2) The weighed graph is constructed from the input image using two spatial constraints. 3) Harmonic energy minimization is applied iteratively to estimate labels for patches in U and newly labeled patches are added to L for the next iteration. 4) It brings the human in the loop by plugging in relevance feedback (RF) to collect negative samples for learning.

Unfortunately, SmartLabel has some shortcomings such as demand of human supervision for RF and zigzag object boundaries in resulting object areas, shown in [121]. These issues limits SmartLabel's application to many tasks. To overcome these weaknesses, we have proposed SmartLabel-2 which improves in three aspects. In particular, we apply a quadtree structure to partition images instead of using regular gridding. Secondly, we introduce a novel saliency-based method to sample negative samples. Finally, we adopt image superpixel representation[88] to refine labeling results and extract smooth object boundaries. The evaluation of proposed SmartLabel-2 and comparison with SmartLabel and Zhu's SSL method in a dataset of six object classes indicate that SmartLabel-2 not only works effectively with a small amount of user input (e.g., $1-5\%$ of image size) but also achieve very promising results. In some cases, SmartLabel-2 even obtains nearly perfect performance.

2.1 Relations to Other Object Extraction Tools

Labeling objects in images is closely related to image segmentation, object extraction or image editing, but they are different in terms of applications. Image segmentation has been an active research area for decades. We describe some standard or state-of-the-art interactive object extraction tools: Magic Wand, GrabCut[90], ClickRemoval[82] and central object extraction[61].

Magic Wand as seen in Adobe Photoshop and many other photo-editing tools applies a basic

seeded region growing algorithm [1]. It is based on absolute color differences among adjacent pixels. It lets a user select a consistently colored area without having to trace its outline. While the user interface is straightforward, finding the appropriate tolerance level is often cumbersome and it cannot extract an object with multiple colored areas.

ClickRemoval[82] is an interactive tool to erase an object from an image. The user indicates the undesired object by pinpointing it with the mouse cursor. The object extraction step relies on a statistical region-growing segmentation technique using color information and the hole-filling step applies background texture based synthesis. This method does not support extracting objects at multiple locations.

Central Object Extraction[61] is developed for object-based image retrieval. It does not require user interactions and automatically extracts central objects. It relies on two underlying assumptions: interesting objects are located near the image center and contain regions with significant color distributions. It cannot extract objects at various places and its assumptions limit its applications.

GrabCut[90] and SmartLabel solve a similar problem but are different in three respects: 1) *User initial inputs are different* as shown in their original papers. GrabCut's common input is an ROI *outside* the object while SmartLabel needs an ROI *within* the object [1]. 2) *Inference models are different*. SmartLabel adopts Gaussian fields with harmonic functions. GrabCut extends the graph-cut approach [11]. 3) *Outcomes are different*. GrabCut achieves fine foreground and background segmentation while SmartLabel can extract multiple similar/same objects. Figure 2.0.2 shows an example of a comparison between SmartLabel and GrabCut.

To summarize, SmartLabel is different from other tools in terms of user inputs, underlying models, and outcomes.

[1] GrabCut and SmartLabel both can work by being given a few background or foreground pixels/patches

2.2 Labeling Objects in Images Using Zhu's SSL method

We first describe SSL notations and the application of Zhu's SSL method[130] to label objects.

SSL is about learning from labeled and unlabeled data. Given a data set, $X = \{x_1, ..., x_l, x_{l+1}, ..., x_n\}$, and a label set, $\mathcal{C} = \{1, ..., c\}$, the first l samples have labels $\{y_1, ..., y_l\} \in \mathcal{C}$ and remaining samples are unlabeled. We call $L = \{(x_1, y_1), ..., (x_l, y_l)\}$ the labeled set and $U = \{(x_{l+1},\ y_{l+1}), ..., (x_n, y_n)\}$ the unlabeled set. Graph-based SSL methods consider a connected graph, $G = (V, E)$, with n vertexes in V corresponding to n samples, where vertexes $L = \{1, ..., l\}$ are labeled samples and vertexes $U = \{l + 1, ..., n\}$ are unlabeled samples. The edges, E, are weighted by an $n \times n$ affinity matrix, W, computed by certain distance metrics.

The goal of Zhu's SSL method is to compute a real-valued labeling function, $g(\cdot) : V -> \mathcal{R}$, on G with certain nice properties, and then to assign labels for U based on $g(\cdot)$. The labeling function is constrained to assign labels such as $g(i) = g_l(i) \equiv y_i$, on L, $i = 1, ..., l$. Gaussian field configuration aims to make unlabeled samples that are nearby in the graph have similar labels. This motivates choosing a quadratic energy function, $E(g) = \frac{1}{2}\sum_{i,j} w_{ij}(g(i) - g(j))^2$. In the n-dim space, $x \in \mathbb{R}^n$, the weight matrix, W, can be defined as, $w_{ij} = exp\left(-\sum_{d=1}^{n} \frac{(x_{id}-x_{jd})^2}{2\sigma_d^2}\right)$, where x_{id} is d-th component of the feature vector $x_i \in \mathbb{R}^n$, and $\sigma_1, ..., \sigma_n$ are length scale hyperparameters for each dimension.

To assign a probability distribution on $g(\cdot)$, a Gaussian field is formed as $p_\beta(g) = \frac{e^{-\beta E(g)}}{Z_\beta}$, where β is the inverse temperature parameter and Z_β is the partition function $Z_\beta = \int_{g|g_l=L} exp(-\beta E(g))dg$, which normalizes over all functions constrained to g_l on L, where $(x_i, y_i) \in L, i = 1, ..., l$. To compute the harmonic solutions of $g(\cdot)$, we minimize $E(g)$ subject to the labeled data constraint.

$$g = \arg\min_{g|_L = g_l} E(g) = \arg\min_{g|_L = g_l} \frac{1}{2}\sum_{i,j} w_{ij}(g(i) - g(j))^2, \tag{2.2.1}$$

in other words, it satisfies $\Delta g = 0$ on all unlabeled samples in U and is subject to $g|_L = g_l$. Here Δ is called *combinatorial Laplacian*, and defined as $\Delta = D - W$, where$D = diag(d_i), d_i = \sum_j w_{ij}$.

The harmonic property indicates $g(\cdot)$ at each unlabeled sample is average of $g(\cdot)$ at its nearby samples,

$$g(j) = \frac{1}{d_j} \sum_{i\ j} w_{ij} g(i), \text{ for } j = l+1, ..., l+u, \tag{2.2.2}$$

which maintains the smoothness constraint of $g(\cdot)$ with respect to G. Expressed in the matrix form, Equation (1) can be rewritten as $g(\cdot) = Qg(\cdot)$, where $Q = D^{-1}W$. To compute the harmonic solution, W is split into 4 blocks based on the separation of L and U. $W_{n\times n} = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}_{n\times n}$.

Denote the target labeling function $g(\cdot) = \begin{bmatrix} g_l(\cdot) \\ g_u(\cdot) \end{bmatrix}_{n\times c}$, and $g_l(\cdot)$ denotes labels on L, $g_u(\cdot)$ denotes labels on U, and c is the number of classes. The unique harmonic solution $\Delta g(\cdot) = 0$ subject to $g(\cdot)|_L = g_l(\cdot) \equiv y_i, i = 1, ..., l$ is given as a $u \times c$ matrix $g_u(\cdot)$.

$$g_u(\cdot) = (D_{uu} - W_{uu})^{-1} W_{ul} g_l(\cdot). \tag{2.2.3}$$

Algorithm 1 shows Zhu's SSL method for labeling objects. SmartLabel improves it in several aspects.

2.3 SmartLabel

2.3.1 Soft Labeling

To formulate an SSL-based labeling problem, we construct L and U. Given an image and ROI(s), we partition the image into non-overlapping patches of size $B \times B$ (pixels). Each patch is treated as a sample. Patches that are in the ROI or overlap with it are treated as positive samples and added in L. Other patches are treated as unlabeled samples and put in U. The number of labeled samples

Input An image (I) and input ROI(s).

Initialize Divide I in patches of $B \times B$ pixels. patches in ROI are put in L and others in U.

Object Labeling using Zhu's SSL method

1. Construct the weight matrix on L and U. Form the weight matrix W using the weight factor defined in Section III and $W_{ii} = 1$.

2. Construct the combinatorial Laplacian. Compute the matrix $\Delta = D - W$, in which D is a diagonal matrix with D_{ii} equal to the sum of the i-th row of W.

3. Compute the harmonic solution. Compute the label prediction on U using the equation $g_u = (D_{uu} - W_{uu})^{-1} W_{ul} g_l$.

4. Assign labels to unlabeled data. For each x_i in U ($i > l$), assign the label as $y_i = g_u(x_i)$.

5. Output object regions. Produce object regions as combination f patches which are labeled positive.

Algorithm 1: Labeling Objects in Images using Zhu's SSL method.

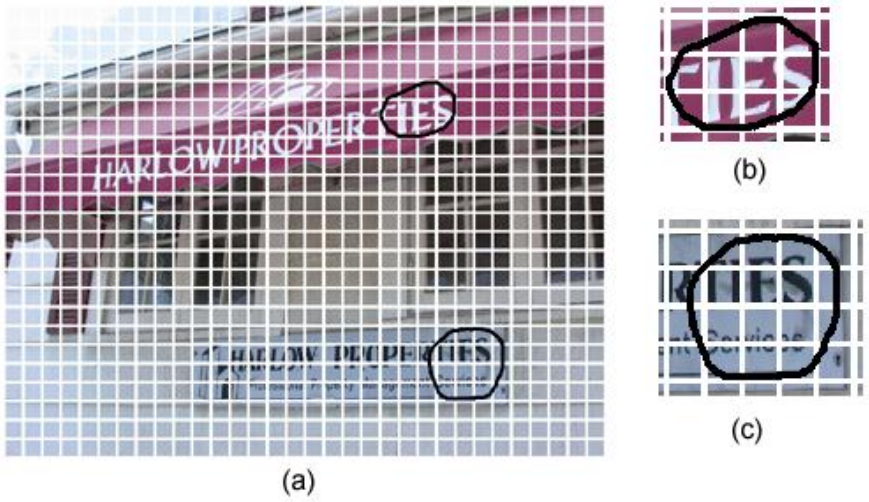

Figure 2.3.1: An illustration of necessity of soft labeling. After regular gridding, two ROIs (in black) include a few complete patches but overlaps with other patches on the borders. Similar problems happen too when the user drags a rectangle ROI.

is l and the number of unlabeled samples is u and $n = l + u$. Note L does not contain negative samples yet.

Most SSL methods use the 0 or 1 labeling strategy (0 means negative and 1 means positive). Problems rise with this strategy because samples in L are different: some patches are within ROI but others partially overlap with ROI. If we use uniform labeling $y_i = 1, i = 1, ..., l$ for all samples in L, the labeled data can be noisy because some patches may contain background. Figure 2.3.1 shows an example sign image with two ROIs. Some patches just partially overlap ROIs and some only contain canvas background.

One solution is to give up all patches which only partially overlap with an ROI and put them in U. But this is expensive by sacrificing already small L. By another solution, we relax the possible label values to real numbers between 0 and 1 as $y_i \in [0, 1]$. We call it soft labeling. The label value for each sample in L is computed as the ratio of its area within ROI to the patch size. Since the mapping function $g(\cdot)$ in Zhu's SSL method is real-value defined, this relaxation appears to be naturally handled by it.

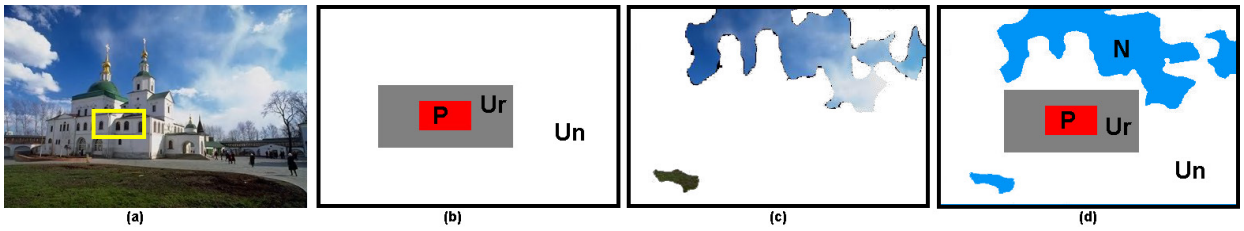

Figure 2.3.2: SSL settings in SmartLabel (b) and SmartLabel-2 (d). P: the positive set; U_r: the relevant unlabeled set; U_n: the irrelevant unlabeled set; N: the negative set. (a) An input image w. an ROI. (c) Detected less informative regions (Section V).

2.3.2 Graph Construction with Spatial Constraints

Labeling objects in an SSL manner possess some unique properties which other SSL problems do not have. One property is that location and context are important in images. Intuitively, pixels or patches nearby tend to belong to one object instance rather than those far away. Regions at nearby locations are more likely to contain similar color distributions than those far away. By observing this location-dependent property, we next describe two spatial constraints which we embed to build the weighed graph: a location-based similarity distance metric and separation of U into U_r and U_n.

We first define a new distance metric by introducing an additional spatial distance term.

$$w_{ij} = exp\left(-\sum_{d=1}^{m}\frac{(x_{id}-x_{jd})^2}{2\sigma_d^2}\cdot\frac{\sum_{k=1}^{2}(x_i^k-x_j^k)^2}{\epsilon}\right), \tag{2.3.1}$$

where the 1st term shows the feature distance as in [130] and the 2nd term measures the spatial distance between two samples. m is the number of dimensions of feature vector x_i, and $\sigma_1, ..., \sigma_m$ are length scale hyperparameters for each dimension. x_i^1 and x_i^2 are image coordinates of x_i and ϵ is a normalization term. The new distance metric approximates the relevancy between two samples by combining Mahalanobis distance in the feature space and Euclidean distance in $2D$.

In most SSL algorithms, samples in U are treated equally in learning, but this assumption may conflict with the location-dependent property. We treat unlabeled samples differently in terms of

their image locations. We separate U into two subsets, U_r and U_n, based on the location of L. U_r includes unlabeled patches that are close to L and used in learning. U_n includes the rest unlabeled patches. Figure 2.3.2 shows an illustration of this setting. With this scheme, we incrementally estimate labels on U_r without sacrificing the global configuration. SmartLabel is an iterative approach and T iterations guarantee all unlabeled patches will be included in the graph eventually. Furthermore, separation of U prevents the graph learning from adding many false positives in L, which can easily happen due to small number of samples in L. Also, searching globally instead of incrementally using L is problematic because the target object concept is under-represented by L without negative samples. By incrementally exploring neighborhood of L and adding high-confidence predicted positive patches in L iteratively, ur method can quickly converge and achieve robust results.

2.3.3 Iterated Harmonic Energy Minimization

SmartLabel introduces an iterated harmonic energy minimization in place of the one-shot minimization in Zhu's SSL method. This scheme allows an automatic increment of L after each iteration. Note in most SSL other algorithms L is fixed but in SmartLabel L is dynamically updated.

Algorithm 2 shows main elements of SmartLabel* (without relevance feedback). In each iteration, samples in U_r and L are studied and labels are predicted for U_r. SmartLabel* runs a fixed number of iterations or until no more unlabeled samples can be added into L. False positives are restrained during the label propagation by applying soft labeling and separation of U_r and U_n. In the following we analyze the time complexity of SmartLabel*. Step (a) takes $O(lu)$ time for update U_r & U_n computation. Sizes of l, u and u_r vary for applications. Step (b) takes $O((l+u_r)^2)$ time for computing the weight matrix of L and U_r. Step (c) takes $O(l+u_r)$ for Laplacian construction. Step (d) takes $O(u_r^2 + u_r l + l) = O(u_r(u_r + l))$ for computing a harmonic solution. And finally the augmenting of L takes $O(u_r)$ time. Thus, the time complexity, $C(T)$, of SmartLabel* in T iterations is $C(T) = O(T(lu + (l+u_r)^2 + l + u_r + u_r(u_r + l) + u_r)) \approx O(T \cdot lu$ when $l \sim u_r$.

Input An image (I) and input ROI(s).

Initialize Divide I in patches of $B \times B$ pixels. Create L and U. Set the max iteration as T and neighborhood size as H. $U_r = \emptyset, U_n = \emptyset$.

SmartLabel*

for $t = 1$ to T **do**

1. Update U_r & U_n. Extract neighboring patches from L in H pixels and put them into U_r, and the rest unlabeled patches are put in U_n.

2. Compute the matrix W' based on L and U_r. Form the weight matrix W' using certain weight measure and $W'_{ii} = 1$.

3. Construct the Laplacian. Compute the matrix $\Delta = D' - W'$, in which D' is a diagonal matrix with D'_{ii} equal to the sum of the i-th row of W'.

4. Compute the harmonic solution. Compute the label prediction on U_r using the following: $g_{u_r} = (D_{u_r u_r} - W_{u_r u_r})^{-1} W_{u_r l} g_l$.

5. Augment L. Add newly predicted unlabeled patches from U_r to L using the predictions g_{u_r}.

Output results. Produce extracted object regions from patches in L.

Algorithm 2: Smartlabel* (w/o. relevance feedback).

One way to speed up is to compute W for the whole image once and shuffle rows and columns to create W' based on updated L and U_r at every iteration.

To compute g_{u_r}, W is split in *9* blocks as $W_{n\times n} = \begin{bmatrix} W_{l\times l} & W_{l\times u_r} & W_{l\times u_n} \\ W_{u_r\times l} & W_{u_r\times u_r} & W_{u_r\times u_n} \\ W_{u_n\times l} & W_{u_n\times u_r} & W_{u_n\times u_n} \end{bmatrix}$ according to L, U_r and U_n. Similarly for the matrix D.

Since only U_r are considered, we denote W' as the first *four* blocks in upper left of W as $W'_{p\times p}$, where $p = l + u_r$. The harmonic solution of U_r is given as,

$$g_{u_r}(\cdot)_{u_r\times c} = (D_{u_r\times u_r} - W_{u_r\times u_r})^{-1} \cdot W_{u_r\times l} \cdot g_l(\cdot)_{l\times c}. \tag{2.3.2}$$

SmartLabel* supports marking of multiple ROIs, which either show many instances of one object or different objects. Then one-against-all classifiers compete using $g^c_{u_r}(j), c = 1, ..., K$ as posterior probability. In particular, an unlabeled sample j is labeled as follows. First, compute the harmonic solution $g_{u_r}(\cdot)$, a $u_r \times c$ matrix and obtain the j-th row which corresponds to the sample j. Second, identify the highest value in the row vector and label the sample j accordingly: $g_{u_r}(j) = \arg\max_{c\in\mathcal{C}} g^c_{u_r}(j)$.

2.3.4 Relevance Feedback

Note that initial ROI(s) provides only positive samples but lacks negative samples. This is problematic in SSL inference. The harmonic energy minimization in the first iteration has a difficulty in discriminating *non-object* patches that have similar appearance (color and/or texture) of the *object* from true *object* patches in U_r, since no negative samples are available initially in L, which is crucial to reduce false positives. To alleviate the problem of lacking negative samples, we bring the human in the loop by embedding relevance feedback (RF)[92] in SmartLabel*. Our aim is to improve performance by including RF data, but to reduce human supervision, we apply RF only in the first iteration. In other cases where the amount of human supervision is not a concern while the performance is crucial, the strategy will be to apply RF in all iterations to achieve the best results. Here, we only employ the first strategy.

Input & Initialization are the same as SmartLabel*.

SmartLabel

1. For $t = 1, 2, ..., T$, similar to Step 1 in **SmartLabel*** except the user gives feedback before augmenting L in sub-step (e) at Iteration 1.

2. *Output resulting extractions.* Produce extracted regions of the object from patches in L.

Algorithm 3: SmartLabel (w. relevance feedback).

We present patches that are labeled as the *object* by SmartLabel* to the user, ranked by $g_u(\cdot)$. When the user labels a patch as a negative example, its neighboring patches in the graph including itself are added to L as negative samples. The same process is done to *non-object* patches labeled by SmartLabel*. Other correctly labeled patches are also added in L. After RF, the graph learning in later iterations has more positive samples and additional negative samples to represent the target object concept discriminatively. The introduction of RF in SmartLabel* makes it possible to learn *object* and *non-object* regions in the image simultaneously and also utilize the underlying graph structure from the unlabeled samples to further improve labeling performance. Algorithm 3 shows the details of SmartLabel (w. RF).

2.4 SmartLabel-2

SmartLabel is not perfect. Its limitations include lack of negative samples, demanding human feedback, zigzag object boundaries and holes in extracted regions. To overcome these issues we have proposed SmartLabel-2 by extending SmartLabel in three aspects [123].

1. SmartLabel needs RF to collect negative samples. With the goal of minimizing human effort, SmartLabel-2 revise a spectral residual approach to sample negative samples from the input

image.

2. Furthermore, SmartLabel-2 apply quadtree to partition instead of regular gridding used in SmartLabel. This not only increases granularity of patches to reach object boundaries but also improves labeling accuracy by reducing mixture of object and background in one patch.
3. Finally, SmartLabel-2 refine labeled object patches via superpixels[88]. Each image's superpixels are computed offline and used with labeled patches to extract object boundaries.

2.4.1 Sampling Negative Samples from a Single Image

Negative samples are often used with positive samples to train a supervised classifier or a semi-supervised classifier. However, in our problem, ROI(s) inside the object only contains positive patches. Zhu's SSL method can learn a classifier using positive data and unlabeled data, but its performance is affected for missing negative samples. SmartLabel tackles the issue by requiring the user to give relevance feedback on newly predicted patches. This practice is expensive and inefficient because of the large number of images to label and limited manpower.

To overcome this challenge, we need an automatic robust scheme to sample negative samples from the input image. One solution is to sample along image borders, but this invalidates one advantage of SmartLabel-2 - *extracting objects anywhere in the image, and not only in the center*. Instead, we revise a spectral residual approach [51] to detect uninformative image regions such as sky and sample negative patches from them. Algorithm 4 shows the scheme and Figure 2.3.2 (c) and (d) show an example. In [51], the spectral residual approach is proposed to detect information rich regions. Using Inverse Fourier Transform, the method outputs the image's *saliency map* which contains the nontrivial image parts. We revise the method to detect less informative regions. We apply the inverse of saliency map to obtain a *background map* which mainly contains uninformative regions. Figure 2.3.2 shows detected uninformative regions by our scheme (top 20%) in (c) and the SSL setting in (d).

Our scheme suffers when the object(s) contains uninformative regions decided by the spectral

Input & Initialization: input image (I), user-specified ROI (P), relevant unlabeled set (U_r) and the threshold (T_s).

Saliency-based Sampling of Negative Samples from Single Image

1. Compute *saliency map* of I using the approach in [51].
2. Sort pixels in *saliency map* in *ascending* order, preserve top T_s portion to obtain negative set N.
3. Combine P, U_r and N to generate image label mask.

Algorithm 4: Saliency based sampling of negative samples from a single image.

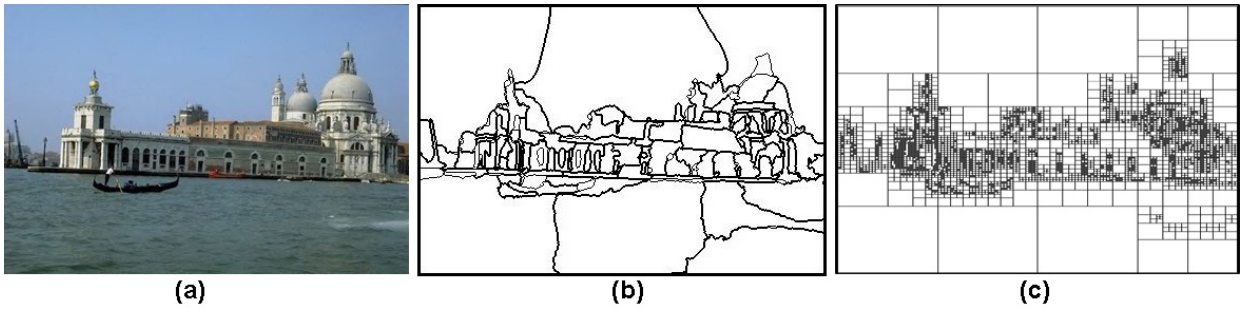

Figure 2.4.1: For SmartLabel-2, (a) an input image; (b) its superpixel map ($P = 100$); (c) its quadtree partition (min patch $= 2 \times 2$).

residual method. Fortunately, this happens only in few cases when labeling sky and ground among other uninformative objects. Even so, SmartLabel-2 works by skipping the step of sampling of negative samples step.

2.4.2 Quadtree Partitioning

A quadtree [40] is a tree data structure in which each internal node has up to four children. Quadtree is often used to partition a 2D space by recursively subdividing it into four quadrants

or regions. A similar partitioning is also known as a Q-tree. All forms of quadtrees share some common features: 1) They decompose space into adaptable cells. 2) Each cell has a maximum capacity. When maximum capacity is reached, the cell splits. 3) The tree directory follows the spatial decomposition of the quadtree.

SmartLabel-2 replaces regular gridding by quadtree based partitioning which changes the underlying patch structure. Figure 2.4.1 depicts quadtree partitioning of an example image. We adopt *qtdecomp* function in Matlab to perform quadtree decomposition, which needs a preprocessing step to resize the input image to a square image at resolution of 2^n. Resizing input images to be square changes the height-to-width, it does not affect patch-based feature extraction much. This resembles unaccessible selection of optimal size and ratio of sliding windows which are commonly used in the image processing and computer vision fields. Quadtree partitioning not only increases granularity of patches to reach arbitrary object boundaries but also improves accuracy by minimizing mixture of object and background in one patch. It is fast too and takes 0.2 second to process a 512×512 image on a 3.2GHz CPU.

2.4.3 Labeling Refinement Based on the Image's Superpixels

Many existing image processing algorithms use image pixel as the basic representation unit. However, a pixel is not a natural representation unit of real scenes [88]. It is only a representation unit of digital imaging. It would be more natural to work with perceptually meaningful units obtained from a low-level clustering process. One well-known way [88] to achieve this natural representation is to apply Normalized Cuts (NCut) [101] to partition an image into P segments, called superpixels. This method gives over-segmentation results, so most structures such as object boundaries and edges are conserved.

NCut is a classical segmentation algorithm which uses spectral clustering to exploit pairwise brightness, color and texture affinities between pixels. As in [88], we apply NCut to over-segment the image to obtain superpixels. This is done *offline* because it normally takes 3 minutes to process

a 320×240 image. Figure 2.4.1 shows an image's superpixel map. The labeled regions are compared with superpixels to generate refined object boundaries. Algorithm 5 shows the process.

Init: input image (I), labeling mask (M), the image's superpixels (S) and result map (R).

Refine quadtree-based labeling results using superpixels.

1. For i=1,2,...,P; (P: number of superpixels) Compute the overlap between $S(i)$ and M, if it is greater than 50%, add $S(i)$ in R.

Use R to cut out predicted object regions from I.

Algorithm 5: Saliency based sampling of negative samples from single image.

2.5 Experiments and Discussion

2.5.1 Experimental Setting

To evaluate SmartLabel-2 and compare it with SmartLabel, we have collected images from public datasets [34, 104, 64] which are the same data used in [121]. Two sets of experiments are conducted, *quantitative analysis* with simulated ROI and RF, and *qualitative analysis* with user input ROIs. Six classes of objects are studied: *airplane, animal, building, car, flower and text*. These classes cover a range of man-made and natural objects.

Two kinds of low-level features are used: color information and Gabor texture. Both features are extracted for each patch. The color info consists of the 1st and 2nd moments for each RGB channel and color histograms with 32 bins for each channel. This results in a 102-dim color vector for each patch. The texture features are obtained from convoluting each patch with various Gabor filters. Here 4 scales and 8 angles are used. The center and 2nd-order moments are computed from each filter output. This results in a 64-dim texture vector. After normalization, we concatenate

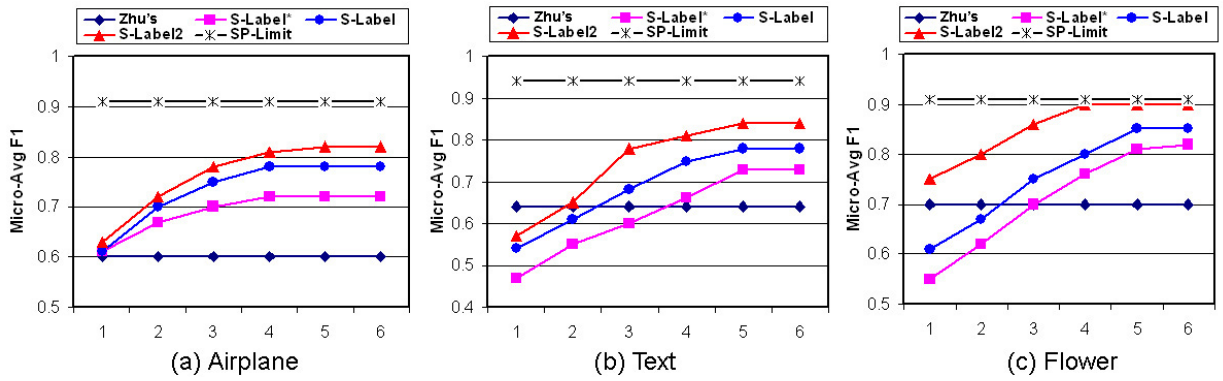

Figure 2.5.1: Comparing four algorithms: Zhu's SSL method[130] (baseline), S-Label* (SmartLabel w/o. relevance feedback), S-Label (SmartLabel) and S-Label2 (SmartLabel-2). Each curve plots F_1^M against iterations. SP-Limit: superpixel-based upper bound.

color and texture vectors into a 166-dim vector. Hyperparameters, $\sigma_1, ..., \sigma_m$, in Equation (5) are set as variance for each dimension.

To quantitatively compare SmartLabel and SmartLabel-2, we simulate input ROIs by randomly selecting a 10% of object regions. In SmartLabel, after the 1st iteration, we simulate RF by labeling top 5 false positives as negative based on ground truth and adding them to L. For every new negative sample, we extract its 3 nearest neighbors in the graph and add them to L. Duplication is not allowed. F_1^M value is used as overall performance metric and defined as $F_1^M = \frac{2\sum_{i=1}^{K} R_i \sum_{i=1}^{K} P_i}{K(\sum_{i=1}^{K} P_i + \sum_{i=1}^{K} R_i)}$, where P_i and R_i are precision and recall computed for i-th image and K is the number of images in the class. Except SmartLabel-2, other methods in our study use regular gridding as in [121]. We report results based on two patch sizes, $B = 8$ or $B = 16$ and for various sizes of U_r such as $H = 32$ and $H = 64$.

2.5.2 Performance Evaluation

We compare four semi-supervised labeling methods as follows. Zhu's SSL method (**Zhu's**) takes ROI(s) as positive samples and makes one-shot labeling prediction on all unlabeled patches. **Smart-**

Label* utilizes the iterative labeling scheme without RF. **SmartLabel** includes RF. **SmartLabel-2** requires no human supervision during learning and incorporates quadtree partitioning and sampling of negative samples.

Figure 2.5.1 depicts performance curves on three object categories, (a) airplane, (b) text and (c) flower. Each subfigure contains the curves of Zhu's, SmartLabel*, SmartLabel and SmartLabel-2 and the superpixel-based upper bound (SP-Limit), which is reachable performance by using superpixel representation. Since Zhu's is an one-shot approach, its performance does not change as iterations. In the 1st loop, Zhu's does better than or is comparable to other methods. This is because Zhu's predicts labels on U while other methods predict only on U_r. We observe that after 3 or 4 iterations, the other methods pick up and do better than Zhu's. Comparing SmartLabel*, SmartLabel and SmartLabel-2, we see that their performances stand in the order of SmartLabel* $\ll$ SmartLabel $\ll$ SmartLabel-2. This observation corroborates the benefits of new improvements proposed in SmartLabel-2. We also notice SmartLabel-2 achieves near SP-Limit F_1^M on the *flower* class. With the number of loops increasing, all three SmartLabel methods monotonically improve and normally converge at the 5th iteration. In summary, SmartLabel (w. RF) performs better than SmartLabel* but underperforms SmartLabel-2.

Figure 2.5.2 shows results by SmartLabel-2 on several classes. The 1st column lists input images with input ROIs. The 2nd column shows corresponding superpixel maps. The 3rd column shows labeling results produced by SmartLabel-2. The last column depicts ground truth. We have several interesting observations from the results. First, SmartLabel-2 generalizes very across man-made objects such as airplanes, buildings and text signs, and natural objects such as flowers and animals. Second, superpixels are proven to be effective in SmartLabel-2 to extract most of object boundaries despite its heavy computation. Third, the results suggest that SmartLabel-2 can assist humans to generate ground truth of object locations in images. Top two building examples in Figure 2.5.2 show SmartLabel-2 even does a better job than a human labeler. For complete results of all six classes, please see our submitted supplementary package.

Figure 2.5.3 shows labeling curves in terms of the size of U_r on three classes: (a) airplane,

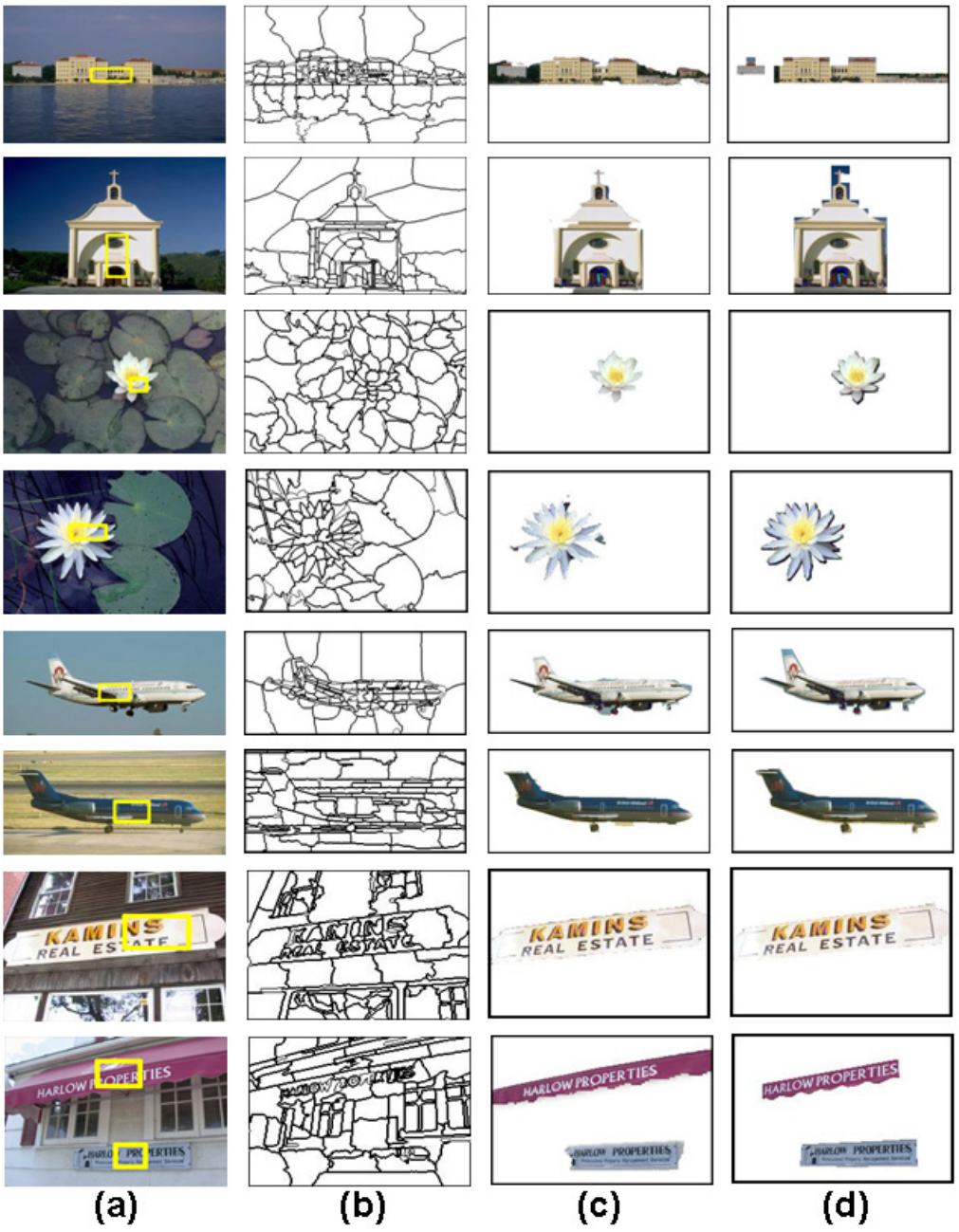

Figure 2.5.2: Results by SmartLabel-2. (a) Input images with ROIs. (b) Superpixel maps ($P = 100$). (c) SmartLabel-2's labeling results. (d) Ground truth. For complete results of all six classes, see our submitted supplementary package.

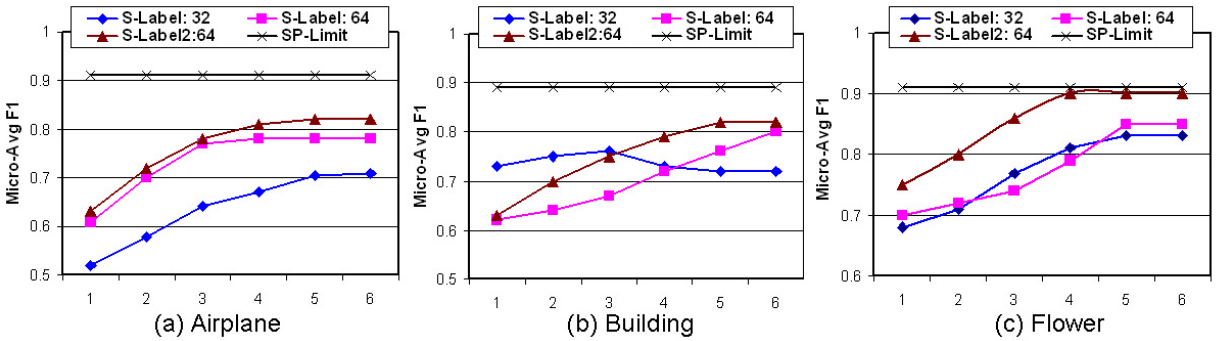

Figure 2.5.3: The impact of U_r's size on performance. Comparison on three classes. $B = 16$. Two settings for SmartLabel: $H = 32$ and $H = 64$. For SmartLabel-2, $H = 64$. Each curve plots F_1^M against iterations. SP-Limit: superpixel-based upper bound.

(b) building and (c) flower. $B = 16$ is used for both SmartLabel and SmartLabel-2. We compare them in two cases: $H = 32$ and $H = 64$. Each subplot shows F_1^M curves. Figure 2.5.3 shows several interesting points. First, a large U_r ($H = 64$) results in higher performance than $H = 32$ along iterations for the *airplane* class. However, the observation is opposite for the *building* class in first few iterations. To investigate the reasons, we examine images in both classes and find that backgrounds in *airplane* images are simpler than those in *building* images. This explains why a large U_r boosts performance on the *airplane* class but degrades performance in complex *building* images. Also, performance trends seem interesting in *building*. We examine images and intermediate results at every iteration, and we find that *building* regions across the class have different sizes and shapes. $H = 64$ gives better performance on some *building* images than others and $H = 32$ can also favor some images not others. This is why the blue curve slightly decreases after the 3rd iteration. In contrast SmartLabel-2 consistently outperforms two other SmartLabel variants.

To make a qualitative comparison between SmartLabel-2 and SmartLabel [121], Figure 2.5.4 shows various labeling results by two methods. The 1st row shows input images with ROIs. The 2nd row shows SmartLabel's results and the third row is SmartLabel-2's results. SmartLabel obtains reasonable results but with zigzag boundaries, holes and some false positives. SmartLabel-2

	Zhu's		S-Label*		S-Label		S-Label2		SP-Limit
	B=8	B=16	B=8	B=16	B=8	B=16	w/o SP	SP	C=0.5
Airplane (128)	0.56	0.60	0.71	0.72	0.78	0.77	0.78	0.82	0.91
Animal (59)	0.57	0.53	0.72	0.75	0.74	0.76	0.79	0.85	0.92
Building (42)	0.60	0.63	0.75	0.70	0.80	0.78	0.77	0.82	0.89
Car (123)	0.54	0.59	0.72	0.69	0.76	0.73	0.74	0.78	0.91
Flower (37)	0.66	0.70	0.82	0.81	0.85	0.83	0.85	0.90	0.91
Text (57)	0.64	0.61	0.73	0.71	0.78	0.78	0.80	0.84	0.94
Avg-F_1^M	0.60	0.61	0.74	0.73	0.79	0.78	0.79	0.84	0.91

Table 2.5.1: Comparing 3 methods proposed in [121] with SmartLabel-2 (S-Label2). For S-Label* and S-Label, $H = 64$ and $B = 8$ or $B = 16$. S-Label2 uses $B = 16$ and $H = 64$. SP: refinement by superpixels, SP-Limit: superpixel-based upper bound, $C = 0.5$: 50% overlap criterion for selecting a superpixel.

gives results with smooth object boundaries and no holes. The *quantitative* improvement by F_1^M may not be significant on these results, but the *qualitative* improvement in terms of user satisfaction is distinctive.

Table 2.5.1 lists a comprehensive comparison on six classes. Numbers in the 1st column are numbers of images in each class. The average of 10 runs' F_1^M is shown for each method. Input images are resized to 512×512. Results of published methods are slightly different from those in [121] due to different experiment settings. We observe that large patches ($B = 16$) improve Zhu's because segmenting images into small patches would create more patches which merely contain background and bring noise to L. We see that SmartLabel-2 achieves near SP-Limit F_1^M on the *flower* class. For other classes, SmartLabel-2 performs very well and better than SmartLabel. The *car* class is harder than others because its images are all gray so color info is not explored. SmartLabel runs for a few seconds processing an image of 384×256 using 16×16 patches on a 3.2GHz CPU. SmartLabel-2 requires a few seconds more. Computing superpixels takes several

	Zhu's	SmartLabel*	SmartLabel	SmartLabel-2
User input	✓	✓	✓	✓
Regular partitioning	✓	✓	✓	
Quadtree partitioning				✓
No human supervision	✓	✓		✓
Negative sampling				✓
Iterative labeling		✓	✓	✓
Boundary refinement				✓

Table 2.5.2: Summary of four SSL labeling methods described here.

	Quadtree	Superpixels
Computation Time (s)	0.4	836.5
Feature Extraction	✓	N/A

Table 2.5.3: Comparison of quadtree and superpixel ($P = 100$) on computation time and FE using 512×512 images.

minutes per 320×240 image, so it can be done offline or on a distributed cluster.

Figure 2.5.5 shows some failure cases by SmartLabel-2. After monitoring the intermediate results at each iteration, we found that most of these failures resulted from two reasons: 1) likeness between background and the object appearance and 2) background regions are added in the labeled set (L) as positive data in initial iterations. These are challenging issues which cannot be dealt with by any unified scheme.

Table 2.5.2 summarizes properties of four semi-supervised labeling methods proposed here.

Fig.2.5.6 shows more results of SmartLabel-2 on all six categories. Table 2.5.3 shows comparison of quadtree and superpixels based on computation and feature extraction (FR).

Figure 2.5.4: Comparison between SmartLabel [121] and SmartLabel-2. In two groups of three rows. The 1st row shows input images with ROIs. The 2nd row shows results by SmartLabel. The 3rd row shows results by SmartLabel-2. Two merits of SmartLabel-2 compared to SmartLabel: 1) smooth object boundaries instead of zigzag lines and 2) less misclassified patches.

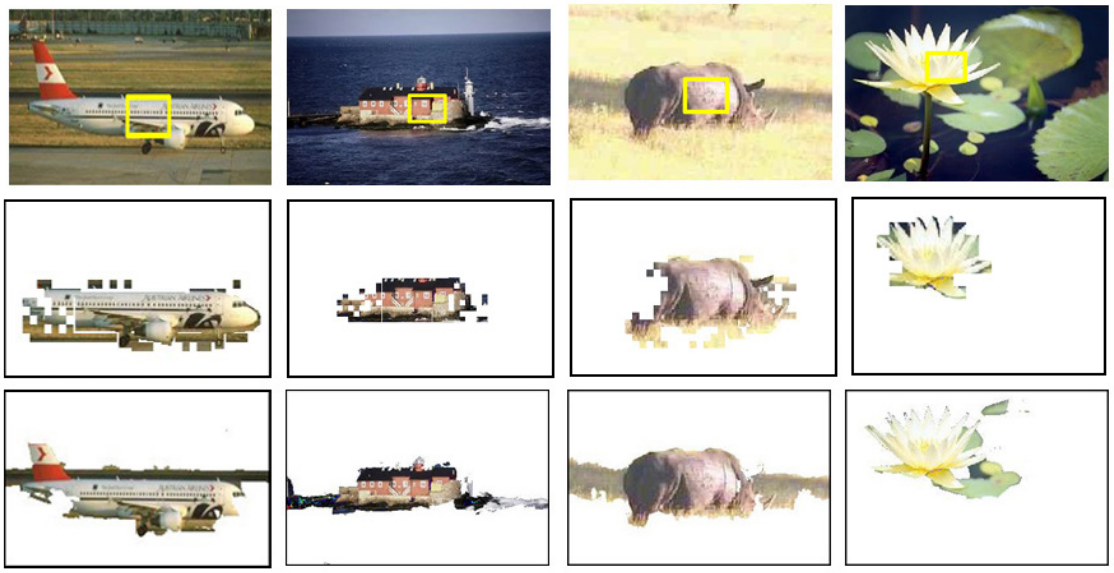

Figure 2.5.5: Some failure examples by SmartLabel-2 (the bottom row). Results by SmartLabel are shown in the middle row.

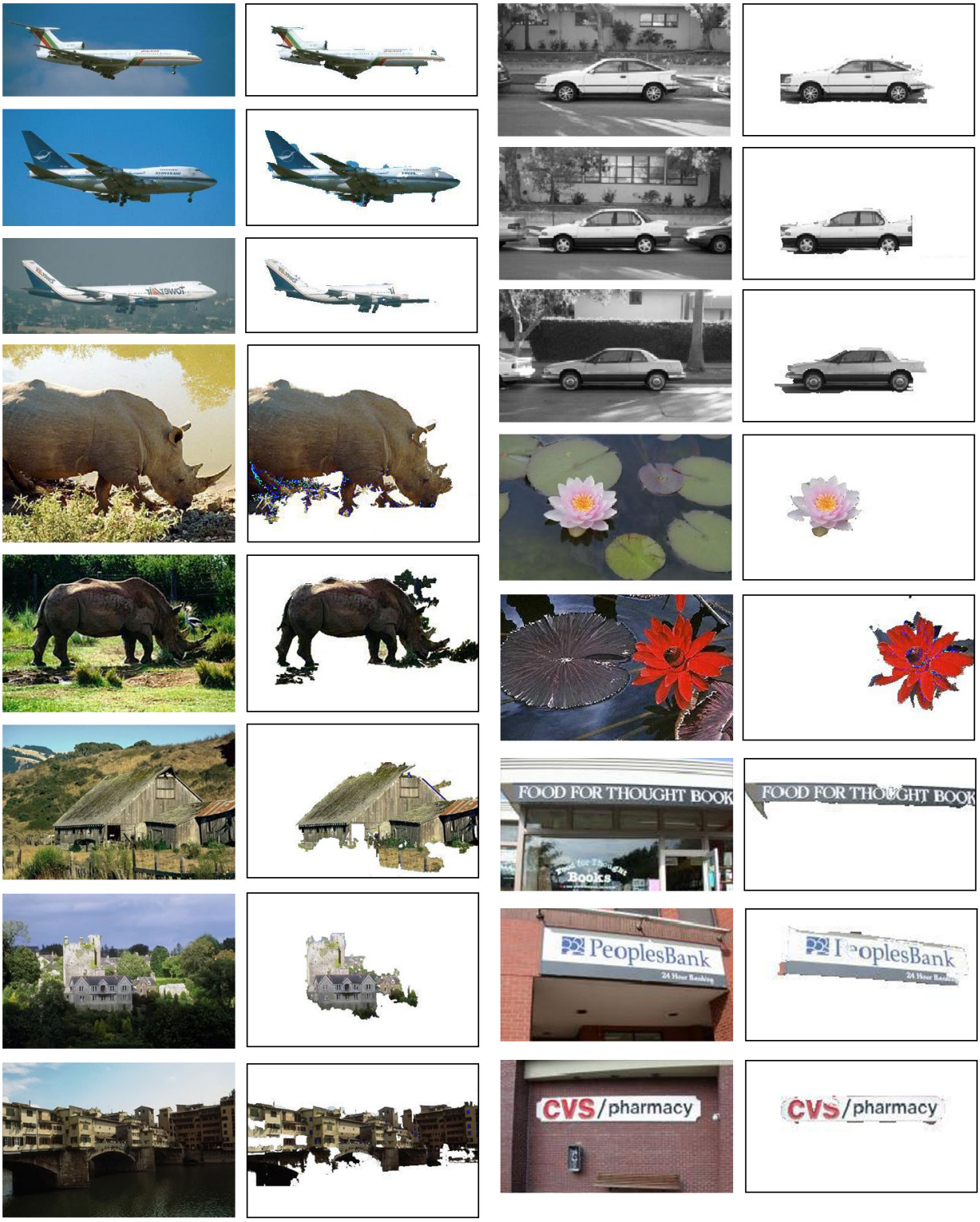

Figure 2.5.6: More results of SmartLabel-2 of all six classes.

Chapter 3

Detection of Text on Road Signs from Video

In this chapter we will focus on one sub problem of detecting text on road signs from video. Automatic detection of text from video is an essential task for autonomous or intelligent transportation systems [58]. There have been extensive research efforts in the detection, segmentation, and recognition of text from still images and video [20, 54, 69, 72, 76]. In addition, research on road sign detection and recognition has recently become an active topic [21, 22, 31, 33, 84]. Related research on license plate recognition and vision-based navigation can be found in [19, 43]. Here we focus on the task of automatically detecting text on road signs from video and using that information in a driver assistance system. Text on road signs carries much useful information for driving; it describes the current traffic situation, defines right-of-way, provides warnings about potential risks, and permits or prohibits roadway access. Automatic detection of text on road signs can help to keep a driver aware of the traffic situation and surrounding environments by seeing and highlighting signs that are ahead and/or have been passed.

The application scenario begins with a video camera mounted on a moving vehicle capturing the scene in the front of the vehicle. The system attempts to detect text on road signs from video input in order to help the driver maneuver in traffic. Fig.3.0.1 shows four examples of road signs. We can see that correctly detecting text on road signs poses many challenges. First, video images

Figure 3.0.1: Examples of road signs in different situations including different lighting conditions, weather and highlights.

are relatively low resolution and noisy. Both the background and foreground of a road sign can be very complex and can change frequently in video. Lighting conditions are uncontrollable due to time of the day and the current weather. A sign reading system must be able to read signs in a variety of conditions such as broad daylight, shaded areas, cloudy days, dusk, rain, and snowing. Second, the typography of the sign text can be rendered in a multitude of fonts, sizes, and colors. Third, text moving quickly in video can be blurred by motion or occluded by other objects. Finally, text can be distorted by the slant, tilt, and shape of signs. In addition to the horizontal left-to-right orientation, other orientations include vertical, circularly wrapped around another object, and even mixed orientations within the same text area. We will only address the horizontal case here and leave other situations to future research.

In order to address the above difficulties, we have proposed a novel framework that can incrementally detect text on road signs from video. The proposed framework takes advantage of spatio-temporal information in video and fuses partial information for detecting text from frame

to frame. The framework employs a two step strategy: 1) locate road signs before detecting text via a plane classification model by using features like discriminative points and color; and 2) detect text within the candidate road sign areas and then fuse the detection results with the help of a feature-based tracker.

3.1 Overview of the Proposed Approach

Some previous research work has paid particular attention to detecting and recognizing symbols on road signs, particularly warning signs such as ”STOP”, ”YIELD”, and ”DO NOT ENTER”. Since only a finite number of shapes and colors can be applied on these warning signs, color and edge-based shape features are normally used to train the detector [33]. In this thesis however, we are interested in detecting not only symbols, but also text on road signs. Text appearing on road signs can have a variety of appearances. Color and shape features are not enough to train a robust detector. Without knowing text on the signs, drivers cannot obtain correct information about current traffic situation and appropriate driving instructions.

To improve the efficiency of the detection process while maintaining a low false hit rate in this task, we naturally employ a divide and conquer strategy to decompose the original task into the two subtasks: localizing road signs and detecting text. The key idea for realizing such an incremental framework is to exploit the temporal information available in video. This idea has been shown to be effective in other text detection tasks such as caption detection in broadcast video [69, 73]. Moreover, because of government requirements on the design and placement of road signs [39], this task also has some auspicious properties for the new framework: 1) text on road signs is highly luminant compared to most sign background colors; 2) text on the same road sign always has similar foreground and background patterns; 3) most road signs exist on vertical planes; and 4) there are only a limited number colors used as background colors.

Fig.3.1.1 shows the architecture of the framework of which four main steps are summarized as

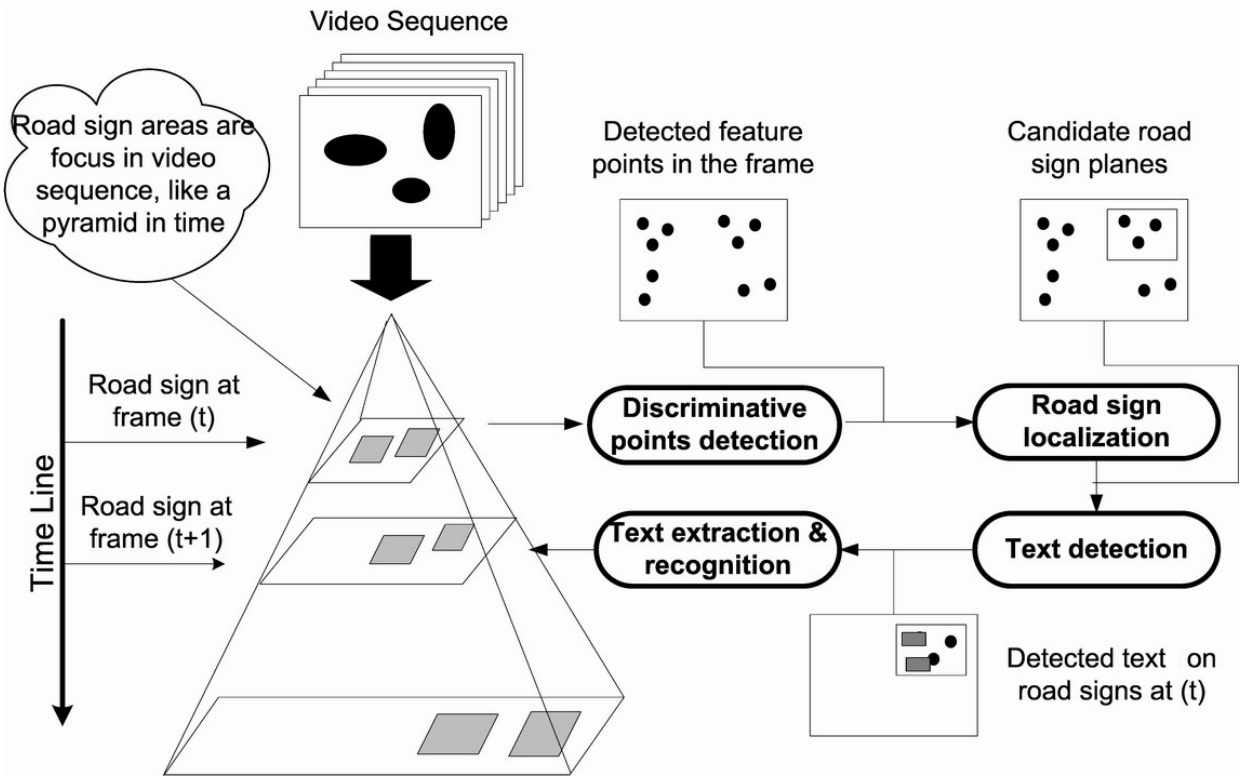

Figure 3.1.1: The architecture of the proposed framework.

follows:

1. Discriminative points detection and clustering - detect discriminative feature points in every video frame using the algorithm proposed in [102] and partition them into clusters.

2. Road sign localization - select candidate road sign regions corresponding to clusters of feature points using a vertical plane criterion.

3. Text detection - detect text on candidate road sign areas and track them.

4. Text extraction and recognition - extract text in candidate sign plane for recognition given a satisfactory size.

There are some interesting properties of the new framework. First, the number of selected points in step 1, N, balances the sign localization speed and system process rate because more

feature point's likelihood that the sign is located early. A large number of feature points also mean intensive computation. Second, spatio-temporal information is extracted and used by the framework to recover the orientation of potential planes in the 3D space. Once a point cluster is classified as a vertical plane, the text detection algorithm will be run on it. Third, the framework applies a feature-based tracker which can track a feature point in a subpixel level. The corners of detected road sign areas and MBRs are tracked to the next frame by averaging the motions of the nearest points of each corner. There are two reasons for tracking discriminative points instead of the boundary corners directly: 1) boundary corners may not be a good feature to track compared to those selected points; and 2) tracking the selected points on the road sign area can relieve the problem of partial occlusion when it happens in video. This property is illustrated and discussed more detailed later.

The new framework possesses two unique merits:

- By applying the divide and conquer strategy, the first two steps of the algorithm can significantly narrow down the search space for the later text detection step and thus reduce the majority of false hits which occur in the case of the whole-image text detection;

- It takes advantage of both temporal and spatial information in video for detecting text on road signs over the timeline.

3.2 Road Sign Localization

In order to differentiate road signs from other objects, we have to use properties of road signs such as color distribution and geometric constraints. The following subsections, we show how to detect discriminative points and use the vertical plane criterion for finding road signs from video.

3.2.1 Discriminative Point Detection

To recover the orientations rigid planes in videos, the system finds a number of discriminative feature points in the current video frame at any given frame. Features are found using the detector of Shi and Tomasi [102]. This method finds features which are good and easy to track. Compute the Laplacian matrix for each pixel in the image and also its minimum eigenvalue λ_m. Select, λ_{max}, the maximum value of λ_m over the whole image. Retain the image pixels that have a λ_m value larger than 10% of λ_{max}. From these selected pixels, retain the local maximum pixels whose value is larger than that of any other pixel in its 3×3 neighborhood. In addition, keep a subset of those pixels so that the minimum distance between any pair of pixels is larger than a given threshold distance. After the feature selection step, we model the neighborhood of each detected feature point using a Gaussian Mixture Model (GMM) since points on road signs share common color properties.

$$g(c) = \beta G_f(\mu_f, \theta_f) + (1 - \beta) G_b(\mu_b + \theta_b), 0 \leq \beta \leq 1, \tag{3.2.1}$$

where G_f, G_b are the color distributions of the foreground and background respectively. Therefore, each feature point can be represented as a vector such as $(\beta, \mu_f, \mu_b, \theta_f, \theta_b)$. The GMM parameters are used as features for the clustering of selected points. Each color space has its own characteristics. Previous research shows that the HSI color space can better handle the lighting variations than others when saturation is not too low [21, 22], which often happen in the natural scene environment. We use the H component in color analysis here.

In this stage, we then use the K-means algorithm to obtain a set of feature point clusters using color analysis, $C_1^t, C_2^t, ..., C_K^t$at time t. K is the number of clusters and is set at 10 in our experiments. $C_i^t = [p_j^t, ..., p_k^t]$ is a cluster including from the j-th feature point to the k-th point. Points in the same cluster share the similar color patterns in their local regions. Thus, a cluster can be naturally considered as a candidate object plane for the later verification. Minimum bounding rectangles are computed for each cluster. The visual illustration of the outcome after this step is

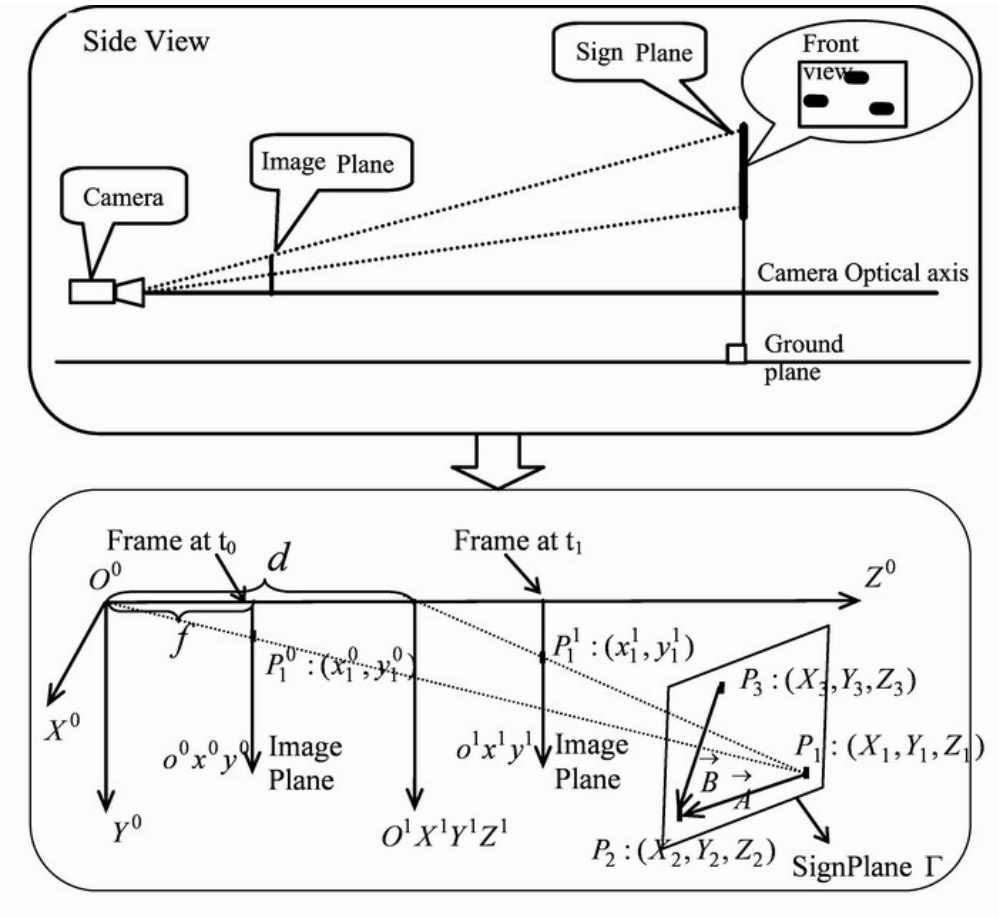

Figure 3.2.1: The basic spatial relationship between two frames.

shown in Fig.3.2.1, where different colors show different clusters of points, e.g., red points in one cluster, blue points in another cluster, so on and so forth.

3.2.2 A Vertical Plane Assumption

We are estimating the orientations of the candidate planes (signs) given three or more points in two successive frames. Here, we make two assumptions: 1) the optical axis of the camera is roughly horizontal and the motion of camera is also going along its optical axis; and 2) scene text lies on planar surfaces. These two assumptions are often true in the real world setting. Particularly, camera is mounted on the vehicle in our task and its optical axis is calibrated to parallel to the

horizontal plane of the vehicle. Fig.3.2.1 upper figure shows the side view of the scenario and the lower one shows the spatial constraints among the road sign plane, the image plane and the camera between two successive frames. Here we have several coordinate systems:

1. The camera coordinate system $O^tX^tY^tZ^t$ and the imaging coordinate system $o^tx^ty^t$ at time t. The optical axis is the Z axis of $O^tX^tY^tZ^t$, and the X axis is parallel to the horizon.

2. We take the camera coordinate system at time t_0 as the basic world coordinate system, $OXYZ$.

$P_1(X_1, Y_1, Z_1), P_2(X_2, Y_2, Z_2)$ and $P_3(X_3, Y_3, Z_3)$ are assumed to be 3 no collinear points on a road sign plane Γ in Fig.3.2.1. Here, a pinhole camera model is used and let the camera's focal length be f, and the camera moves forward a distance d from time t_0 to t_1. As $(t_1 - t_0)$ is normally very small for a real-time video stream, the assumption that the motion of vehicle is always small enough from t_0 to t_1, often holds. The projections of the points $P_i, (i = 1, 2, 3)$ onto two image planes are $p_i^{t_0} : (x_i^{t_0}, y_i^{t_0})$ and $p_i^{t_1} : (x_i^{t_1}, y_i^{t_1}), i = 1, 2, 3$ respectively.

Here, a feature-based tracker is used to find the correspondence of points between t_0 and t_1 [73]. Equation (3.2.2) defines the projection between two coordinate systems. The left sides of the equations are points' coordinates in $o^tx^ty^t$ and the right sides are coordinates in $OXYZ$.

$$\begin{pmatrix} x_i^{t_0} \\ y_i^{t_0} \end{pmatrix} = \frac{f}{Z_i} \begin{pmatrix} X_i \\ Y_i \end{pmatrix}, i = 1, 2, 3 \tag{3.2.2}$$

$$\begin{pmatrix} x_i^{t_1} \\ y_i^{t_1} \end{pmatrix} = \frac{f}{Z_i - d} \begin{pmatrix} X_i \\ Y_i \end{pmatrix}, i = 1, 2, 3 \tag{3.2.3}$$

We further write down the expressions for P_i in Equation (3.2.4).

$$P_i : \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \frac{d}{f} \cdot \left(\frac{x_i^{t_0} \cdot x_i^{t_1}}{x_i^{t_1} - x_i^{t_0}}, \frac{y_i^{t_0} \cdot y_i^{t_1}}{y_i^{t_1} - y_i^{t_0}}, \frac{f \cdot x_i^{t_1}}{x_i^{t_1} - x_i^{t_0}} \right)^T, i = 1, 2, 3. \tag{3.2.4}$$

Although f and d are unknown, we will soon see that their values are not necessary to be specified in the later algorithm. We can find that

$$\frac{x_k^{t_1}}{x_k^{t_1} - x_k^{t_0}} = \frac{y_k^{t_1}}{y_k^{t_1} - y_k^{t_0}}, k = 1, 2, 3. \tag{3.2.5}$$

For simplification in the following derivation, we define the following ratios M_k as:

$$M_k = \frac{x_k^{t_1}}{x_k^{t_1} - x_k^{t_0}} = \frac{y_k^{t_1}}{y_k^{t_1} - y_k^{t_0}}, k = 1, 2, 3. \tag{3.2.6}$$

Fig.3.2.1 lower figure depicts that $\overrightarrow{A}$ is a vector from P_1 to P_2, and $\overrightarrow{B}$ is a vector from P_3 to P_2. Using the estimated coordinates of P_i in Equation (3.2.4), we further obtain the estimations of $\overrightarrow{A}$ and $\overrightarrow{B}$ as:

$$\overrightarrow{A} : \begin{pmatrix} X_1 - X_2 \\ Y_1 - Y_2 \\ Z_1 - Z_2 \end{pmatrix} = \begin{pmatrix} d \cdot \frac{x_1^{t_0}}{f} \cdot M_1 - d \cdot \frac{x_2^{t_0}}{f} \cdot M_2 \\ d \cdot \frac{y_1^{t_0}}{f} \cdot M_1 - d \cdot \frac{y_2^{t_0}}{f} \cdot M_2 \\ d \cdot (M_1 - M_2) \end{pmatrix}, \tag{3.2.7}$$

$$\overrightarrow{B} : \begin{pmatrix} X_3 - X_2 \\ Y_3 - Y_2 \\ Z_3 - Z_2 \end{pmatrix} = \begin{pmatrix} d \cdot \frac{x_3^{t_0}}{f} \cdot M_3 - d \cdot \frac{x_2^{t_0}}{f} \cdot M_2 \\ d \cdot \frac{y_3^{t_0}}{f} \cdot M_3 - d \cdot \frac{y_2^{t_0}}{f} \cdot M_2 \\ d \cdot (M_3 - M_2) \end{pmatrix}. \tag{3.2.8}$$

In order to recover the orientation of the sign plane Γ, we need to further know the normal vector of Γ , noted as N, which can be obtained by the cross product of $\overrightarrow{A}$ and $\overrightarrow{B}$.

$$N = \overrightarrow{A} \times \overrightarrow{B} = (X_\Gamma Y_\Gamma Z_\Gamma)^T, \tag{3.2.9}$$

where the expression for each component of N, i.e., $X_\Gamma, Y_\Gamma, Z_\Gamma$, is as follows:

$$X_\Gamma = \frac{d^2}{f} \cdot [(y_1^{t_0} M_1 - y_2^{t_0} M_2)(M_3 - M_2) - (y_3^{t_0} M_3 - y_2^{t_0} M_2)(M_1 - M_2) = \frac{d^2}{f} \cdot C_X,$$

$$Y_\Gamma = \frac{d^2}{f} \cdot [(x_3^{t_0} M_3 - x_2^{t_0} M_2)(M_1 - M_2) - (x_1^{t_0} M_1 - x_2^{t_0} M_2)(M_3 - M_2) = \frac{d^2}{f} \cdot C_Y,$$

$$Z_\Gamma = \frac{d^2}{f^2} \cdot [(x_1^{t_0} M_1 - x_2^{t_0} M_2)(y_3^{t_0} M_3 - y_2^{t_0} M_2) - (x_3^{t_0} M_3 - x_2^{t_0} M_2)(y_1^{t_0} M_1 - y_2^{t_0} M_2) = \frac{d^2}{f^2} \cdot C_Z,$$

where C_X, C_Y and C are used to represent the long terms in equations and to simplify the expressions. Equation (3.2.9) gives a nice way to estimate the orientation of the candidate plane by using the three points' image coordinates given the spatial constraints. By taking advantage of the approximations, we can further define a model to classify planes into positive and negative categories using different criteria such as vertical vs. non-vertical planes or rigid vs. nonrigid planes. Here, we are interested in the vertical plane criterion.

Based on the property that most road signs are on vertical planes, the ratio of the component to the length of N is supposed to be smaller than a certain threshold. Thus, we can estimate the ratio and use it to locate vertical planes. The ratio is defined as follows:

$$R = \frac{|Y|}{\|N\|} = \frac{|C_Y|}{\sqrt{C_X^2 + C_Y^2 + \frac{1}{f^2} \cdot C_Z^2}}. \tag{3.2.10}$$

3.2.3 Sign Localization Algorithm Description

We have obtained a number of clusters of feature points from the feature detection step. Now we can apply the vertical plane assumption to verify if a cluster represents a candidate road sign plane. In order to reduce the error caused by outliers, we use the median as the normal vector of candidate plane and also calculate its corresponding variance.

$$R_i = median(R_{ij}), j = 1, 2, ..., m_i, \tag{3.2.11}$$

$$Var(R_i) = E(R_{ij} - R_i)^2, j = 1, 2, ..., m_i, \tag{3.2.12}$$

where R_i is the median estimation for the ith cluster, m_i is the number of recovered orientation vectors from one cluster, and R_{ij} is the jth vector in the ith cluster. We use the median to approximate the orientation of the candidate plane associated with each cluster C_i^t. A cluster will be classified as a rigid plane if $Var(R_i)$ is smaller than a threshold. Moreover, it will be classified as a vertical plane if R_i is smaller than a threshold.

Specifically, when the sign plane is perpendicular to the optical axis of the camera, we can use a simplified criterion to verify candidate planes. In this case, points on the sign plane have almost

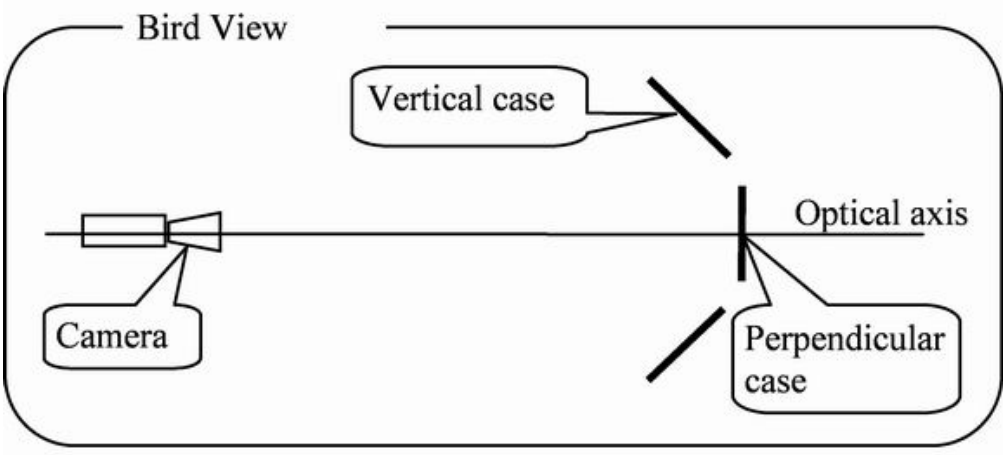

Figure 3.2.2: Bird view of the perpendicular case.

the same distance to the camera along the Z axis in $OXYZ$. The constraint for P_1, P_2, P_3 can be written as:

$$Z_1 \approx Z_2 \approx Z_3. \tag{3.2.13}$$

Given the above condition, we obtain the equality among the ratios M_k from Equations (3.2.5) and (3.2.6):

$$M_1 \approx M_2 \approx M_3. \tag{3.2.14}$$

From Equation (3.2.6) we see that the ratios of both components X and Y to the length of N should be smaller than a threshold if Γ is a vertical plane. This is quite an intuitive observation since we can imagine that N is almost parallel to the optical axis Z given the Equation (3.2.13) holds. Fig.3.2.2 depicts the perpendicular situation as a special case of the vertical case.

Thus, the simplified verification criterion is shown as follows,

$$\overline{\gamma} = \frac{1}{J}\sum_{j=1}^{J} \frac{\sqrt{|X_j|^2 + |Y_j|^2}}{\|N_j\|}, \tag{3.2.15}$$

$$Var(\overline{\gamma}) = E_\gamma(\gamma_j - \overline{\gamma})^2. \tag{3.2.16}$$

The criterion indicates the triangle of P_1, P_2, P_3 is about to maintain the consistent spatial structure from t_0 to t_1. More generally, such consistency of the spatial structure should hold whenever

Input: Feature point clusters $C_1^t, C_2^t, ..., C_K^t$.

Output: $L^t = l_1^t, l_2^t, ..., l_K^t, l_i^t \in P, V, R, NV$ label for C_i^t. P is the perpendicular planes, V for vertical planes, R for rigid planes and NR for non-rigid planes.

Algorithms:

1.*R-plane identification step:* compute R_i and $Var(R_i)$. If $Var(R_i) < \varepsilon$, go to Step 2; otherwise, label $l_j^t = NR$. Get the next cluster. If all clusters have been examined, exit.

2. *V-plane identification step:* If $R_i < \delta$, which means it meets the vertical plane criterion, and then go to Step 3 to further verify it is a perpendicular plane; otherwise, label $l_j^t = R$; it is a rigid but not vertical plane. Go back to Step 1.

3. *P-plane identification step:* Compute the mean $\overline{\gamma}$ and variance $Var(\overline{\gamma})$. If $\overline{\gamma} < \xi$ and $Var(\overline{\gamma}) < \psi$, label $l_i^t = P$; otherwise, label $l_i^t = V$. Go back to Step 1.

Note: $\varepsilon, \delta, \xi, \psi$ are thresholds in the algorithm.

Algorithm 6: The algorithm for the road sign localization.

the condition in Equation (3.2.13) is satisfied. We use the Maximum Likelihood criterion to estimate the mean and variance of . Algorithm I summarizes the process of sign plane classification. In this algorithm, the perpendicular plane is named as P, the vertical plane as V, the non-vertical plane as NV, the rigid plane as R and the nonrigid plane as NR. The relationship among them is: $P \subset V \subset R, V \cap NV = \varnothing$and $R \cap NR = \varnothing$. Once we identify candidate road signs from feature point clusters, we can focus on text detection within these candidates only, which can substantially reduce the search space for the text detection.

3.2.4 A More General Case

Previously, we assume that the camera motion will only be traversal along the optical axis as in Fig.3.2.2. However, this assumption is likely violated under some conditions such as when the vehicle makes a turn. Calibrating the camera parallel to the horizontal plane of vehicle does not help obviate the issue. From Equation (3.2.10), we can see that variations of the ratio may come from two sources: a change in the focal length, and violation of the assumption motion occurs only along the optical axis. The first issue can be negated after f is calibrated prior the experiment and fixed during experiments. The second issue is relatively complicated because it could cause the variations of more than one term in Equation (3.2.10). Next, we will devote the analysis for the turning case in particular.

The key question to ask here is, can we recover the normal vector N of Γ when the vehicle makes a turn? Our conclusion is that we can recover N if only if we know the turning angle, such as θ. Assume the relative movement from t_0 to t_1 is $(\Delta X, 0, \Delta Z)$ caused by turning and the turning angle relative to the Y axis is θ. Fig.3.2.3 depicts such a turning scenario.

The normal vector to the plane $O^0P_1P_2$ in the $O^0X^0Y^0Z^0$ coordinate system is:

$$N_0 = \begin{pmatrix} x_1^0 \\ y_1^0 \\ f \end{pmatrix} \times \begin{pmatrix} x_2^0 \\ y_2^0 \\ f \end{pmatrix} = \begin{pmatrix} (y_1^0 - y_2^0)f \\ (x_2^0 - x_1^0)f \\ x_1^0 y_2^0 - x_2^0 y_1^0 \end{pmatrix}. \tag{3.2.17}$$

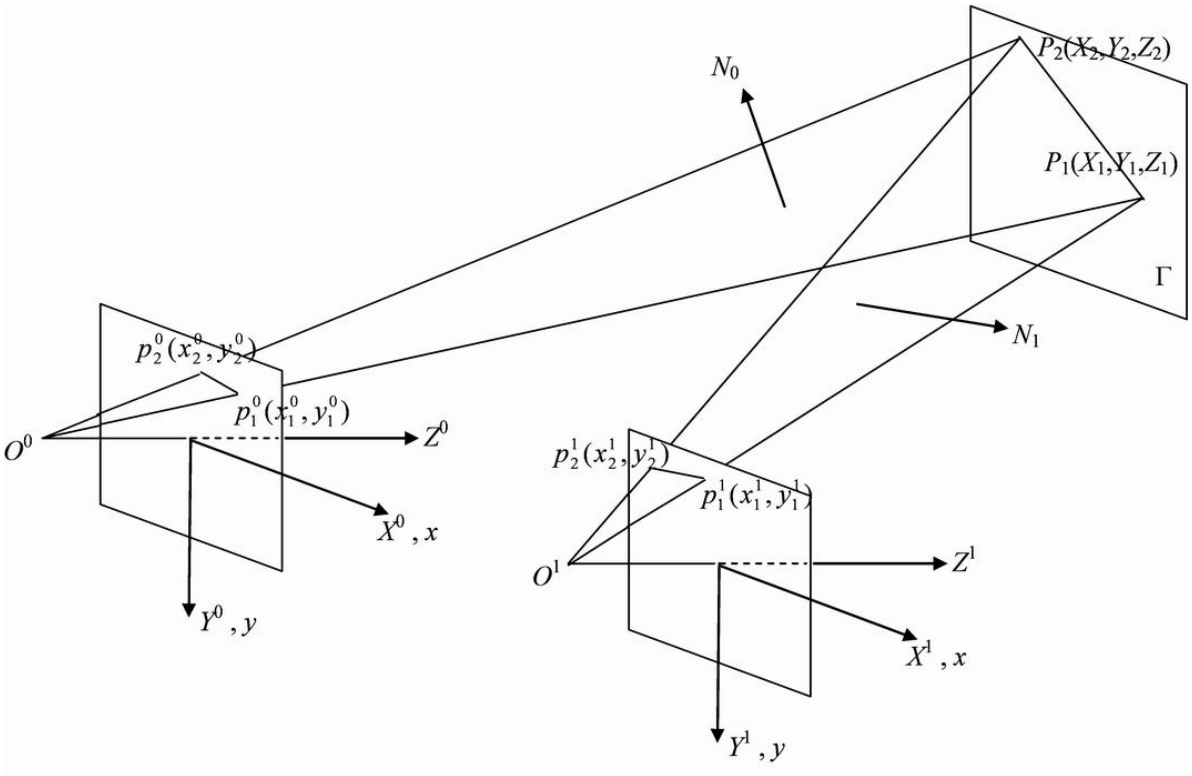

Figure 3.2.3: Model sensitivity analysis for the vehicle turning case, when the assumption of the translation along the optical axis is violated.

Similarly, the normal vector to the plane $O^1P_1P_2$ in the $O^1X^1Y^1Z^1$ coordinate system is:

$$N_0 = \begin{pmatrix} x_1^1 \\ y_1^1 \\ f \end{pmatrix} \times \begin{pmatrix} x_2^1 \\ y_2^1 \\ f \end{pmatrix} = \begin{pmatrix} (y_1^1 - y_2^1)f \\ (x_2^1 - x_1^1)f \\ x_1^1 y_2^1 - x_2^1 y_1^1 \end{pmatrix}. \tag{3.2.18}$$

Therefore, the normal vector to the plane $O^1P_1P_2$ in the $O^0X^0Y^0Z^0$ coordinate system can be computed as:

$$N_1' = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \begin{pmatrix} (y_1^1 - y_2^1)f \\ (x_2^1 - x_1^1)f \\ x_1^1 y_2^1 - x_2^1 y_1^1) \end{pmatrix} = \begin{pmatrix} (y_1^1 - y_2^1)f\cos\theta + (x_1^1 y_2^1 - x_2^1 y_1^1)\sin\theta \\ (x_2^1 - x_1^1)f \\ (y_2^1 - y_1^1)f\cos\theta + (x_1^1 y_2^1 - x_2^1 y_1^1)\cos\theta \end{pmatrix} \tag{3.2.19}$$

Thus, the normal vector of $\rightarrow P_1P_2$ is:

$$N_{12} = N_0 \times N_1'. \tag{3.2.20}$$

Similarly, given a third point, P_3, we obtain N_{23} and N_{13}. Therefore, the normal vector of Γ will be

$$N = N_{12} \times N_{13}. \tag{3.2.21}$$

As we can see from the above equation, the answer to the question at the beginning of this section is that we can recover the normal vector of the plane Γ if we know or can approximate the turning angle θ.

3.3 Detection of Text on Road Signs

Even when sign locations are known in images, correctly detecting text on road signs is still not easy because of deformations, highlights, shadows and other factors. To work around these changes in an image, we use an edge-based cascade text detection method that integrates edge detection, adaptive searching, color analysis and geometry alignment analysis. This method was first proposed in [22]. Fig.3.3.1 shows the basic flow of this schema.

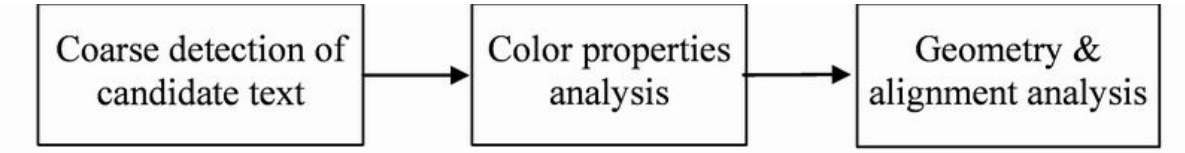

Figure 3.3.1: An edge-based text detection method.

3.3.1 Coarse Detection of Candidate Text

The intensity of an image is a major source of information for text detection; however, it is well known that the intensity is sensitive to lighting variations. In contrast, the gradient of the intensity

(edge) is less sensitive to lighting changes. Therefore, we use edge-based features in the first coarse detection phase. The main idea of the detection algorithm for coarse detection is as follows: A multi-scale Laplacian of Gaussian (LOG) edge detector is used to obtain an edge set for each candidate text area. The properties of the edge set associated with each edge patch, such as size, intensity, mean, and variance, are then calculated. Some edge patches will be excluded from further consideration based on certain criteria applied to the properties, and the rest will be passed to a recursive procedure. The procedure attempts to merge adjoining edge patches with similar properties and re-calculate the properties recursively until no update can be made. With LOG, we can obtain enhanced correspondences on different edge scales by using a suitable deviation. Since English letters in the same context share some common patterns, we can use them to analyze the alignment and rectification parameters and refine detection results. Color distribution of the foreground and background is one such important property.

3.3.2 Color Analysis

Text on signs is designed for drivers to view at a distance, so they have highly distinguishable colors on their foregrounds and backgrounds, and also have a high intensity contrast in their grayscale images. This property helps make it easy to segment text and describe letters using marginal distributions in a color space. However, it is almost impossible to obtain uniform color distributions of the foreground and background because of lighting sources, shadows, dust, etc. Here again, we use the Gaussian Mixture Model to characterize color distributions of the foreground and background of road signs, and more specifically, for each words on signs. Recall that β provides a cue on the complexity of the letter, $\|\mu_f - \mu_b\|$ indicates the contrast for a color space invariant to the lighting condition, and θ_b, θ_f yields the font style.

Since there could be multiple lighting sources and shadows in natural scene video, contrasts of foreground and background might change significantly across the entire sign. Therefore, we model the distribution of each letter separately rather than the entire sign as a whole. We normalize HSI

components within the range of [0, 255] in computation. These GMM parameters are used for text alignment analysis and estimation of affine parameters. An expectation maximization (EM) algorithm is applied to estimate the parameters. To differentiate the background and foreground, we enlarge the boundary of the letter by 2 pixels on each side and calculate the color distribution of the region between the original boundary and the enlarged boundary. This distribution should be the same or similar to the background distribution. We then can determine the background distribution in the GMM by comparing distributions in the GMM to this distribution.

3.3.3 Text Alignment Analysis

The objective of text alignment analysis is to align characters in an optimal way so that letters that belong to the same context will be grouped together. Text alignment analysis includes two cluster features: intrinsic and extrinsic features. The intrinsic features are those which do not change with the camera position and the extrinsic features are those ones which change with the camera position. The intrinsic features include font style, color, etc; the extrinsic features include letter size, text orientation, etc. Both the intrinsic and extrinsic features can provide cues for the alignment analysis. The algorithm first clusters text regions using intrinsic and extrinsic features, including the center, height, width, and GMM parameters of candidate text regions. Then, it uses the Hough transform to find all possible line segments. These line segments form several compatible sets. The two smallest sets are selected as candidates. One winner from the compatible sets is selected by picking the line segment which has a larger mean length than all line segments excluding the shortest. Then, it removes all small candidate regions and finally outputs the corner positions of MBR for each candidate region.

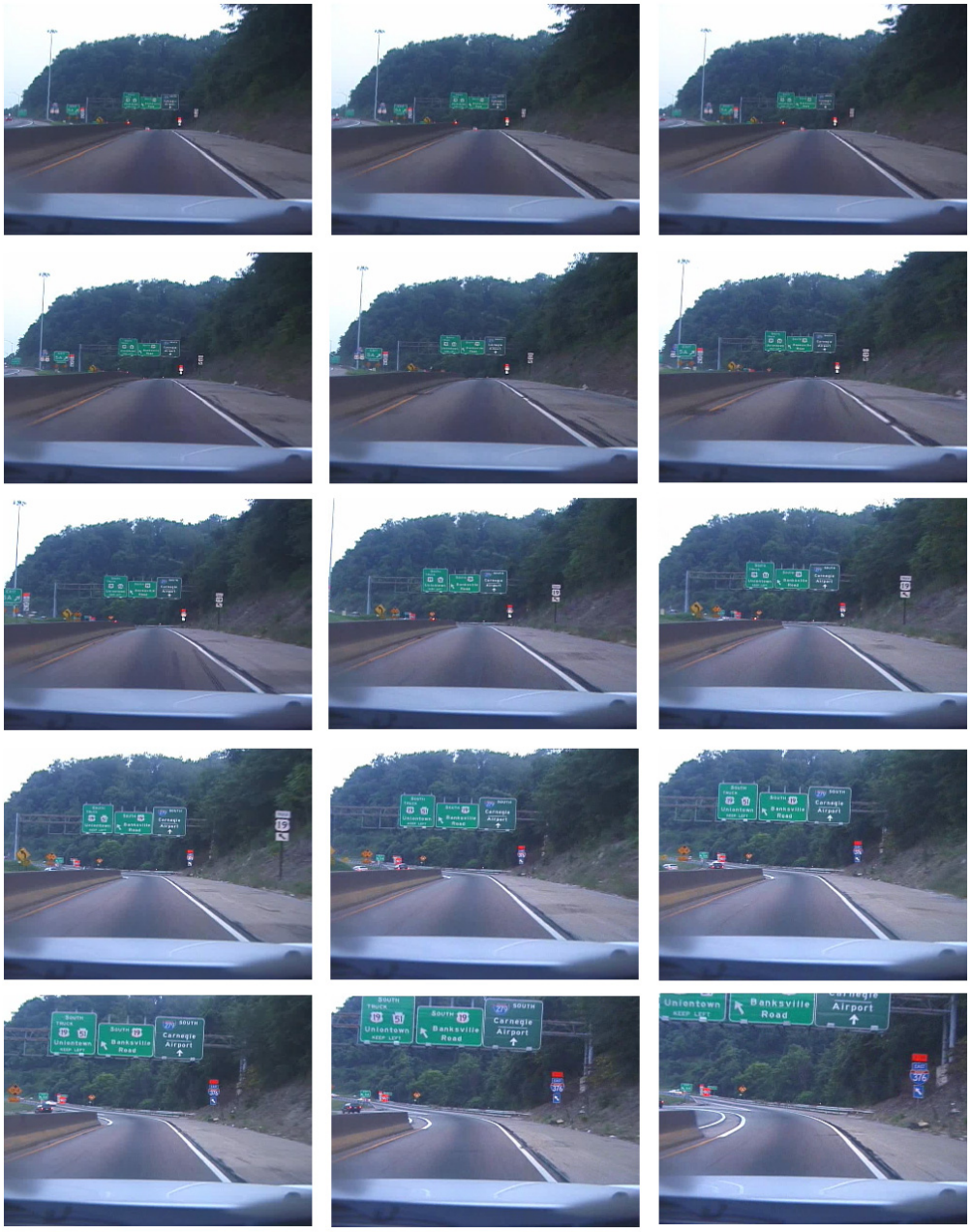

Figure 3.3.2: An example sequence from our dataset. From left to right and top to down.

3.4 Experimental Results and Discussion

The proposed framework has been evaluated through experiments with a large and diverse set of road sign video sequences. From a video database of 3-hour natural scene videos captured by a

DV camera mounted in a moving mini van, we selected 22 video sequences with different driving situations including different road conditions (straight, curve), vehicle speed (low, high), weather conditions (sunny, cloudy), and daylight variations. Fig.3.3.2 shows a typical example of video frames from our dataset. The objective of the selection was to be as diverse as possible and cover the range of difficulty as well as the generality of the task. Thus, we did not include the extreme cases, such as crooked lateral signs. Each video sequence is about 30 seconds, contains an average of 92 road signs and 359 words (including numbers such as a speed limit), and has a frame size of 640×480. Our system was implemented in C++ and tested on a 1.8 GHz Pentium IV PC. The number of selected points was set at $N = 150$ and the number of clusters was set at $K = 10$. The parameters in the Algorithm I were $\varepsilon = 0.1, \delta = 0.15, \xi = 0.2, \psi = 0.1$.

3.4.1 Incremental Text Detection Process

Fig.3.4.1 illustrates the process of incremental text detection. During a few initial video frames, no discriminative points are found on the road sign plane as shown in (a). In the frame (b), some feature points are detected in the frame, shown as blue points. The system then verifies the sign area by using the plane classification algorithm. Once the area is confirmed as a vertical plane it will be bounded with a yellow rectangle as in (c). The following frames, such as (d) - (h), show that more and more texts are detected on the road sign over the time. Newly detected text regions are merged with previously partial detection results. In the meantime, all detected text regions are tracked by averaging the optical flows of the feature points within the detected areas. Finally, all texts on the road sign are correctly detected as shown in (i). Note that there was a sign on the right at the beginning of the sequence (a)-(c), however, our system did not detect it. The reason is that no feature points were found in that sign area.

Fig.3.4.2 illustrates three more examples one in each row. The video sequence in (a) contains three adjacent green road signs and other small road signs. The video was captured during daylight under cloudy weather. Selected feature points mainly appear in the texture-rich areas, such as the

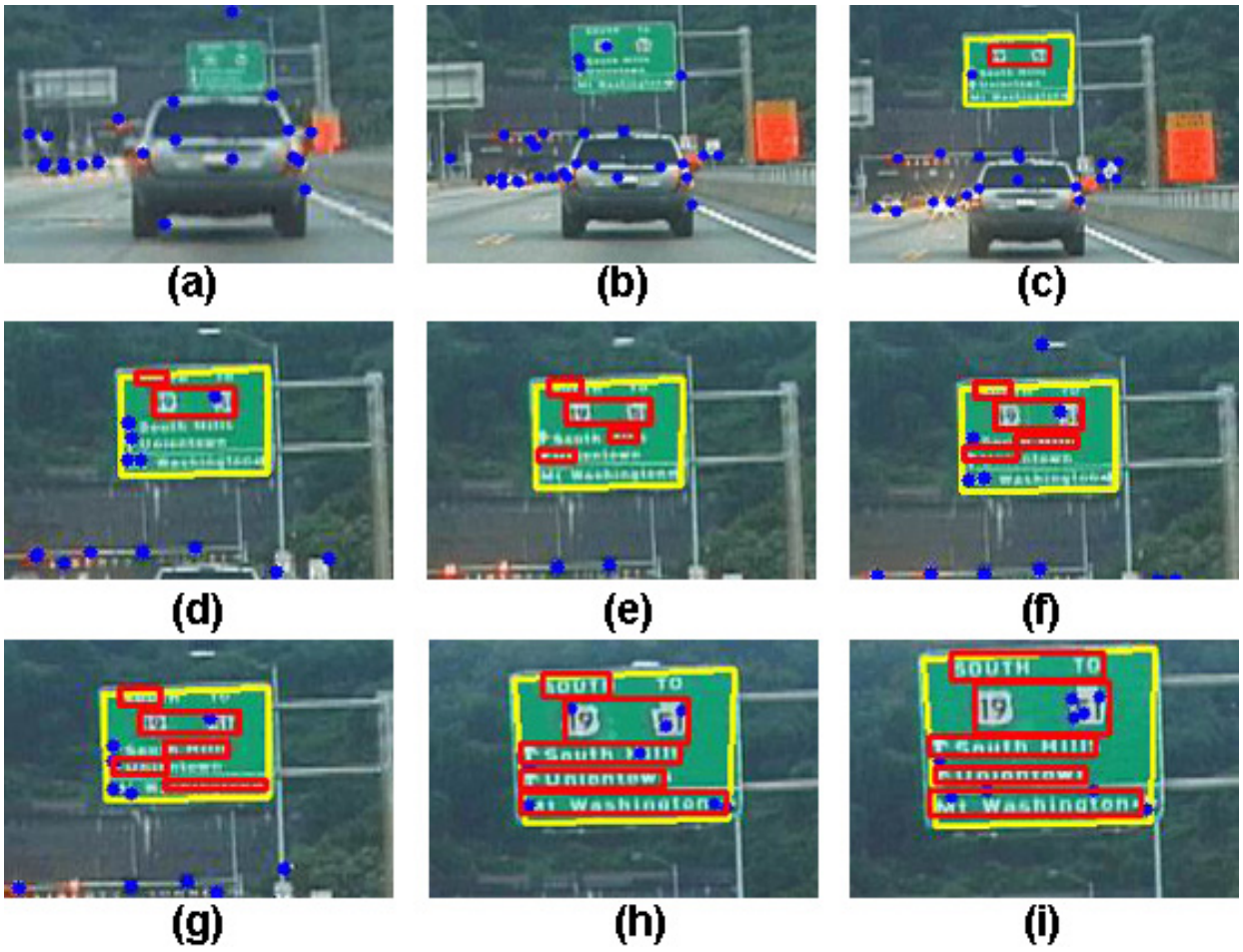

Figure 3.4.1: An illustration of incremental text detection on a video sequence. Blue points are selected feature points, the yellow box bounds the localized road sign area and red boxes indicate the detected text lines.

edge of forest in background, road signs and side roads. Very few feature points appear in less texture areas, such as sky and front ground. Obviously, the feature selection step not only provides candidate clues for road sign localization but also reduces the search space for later text detection. Images in column 3 show results of road sign localization: the labeled candidate road signs within white rectangle. This step further refines the selected features by sign plane classification algorithm and avoids feeding some texture-rich areas, such as forests and side roads, into the text detection module, which could cause false hits. In the next few frames, partial texts are detected on the road signs from frame to frame, as shown with yellow boxes. Detected text regions on road signs

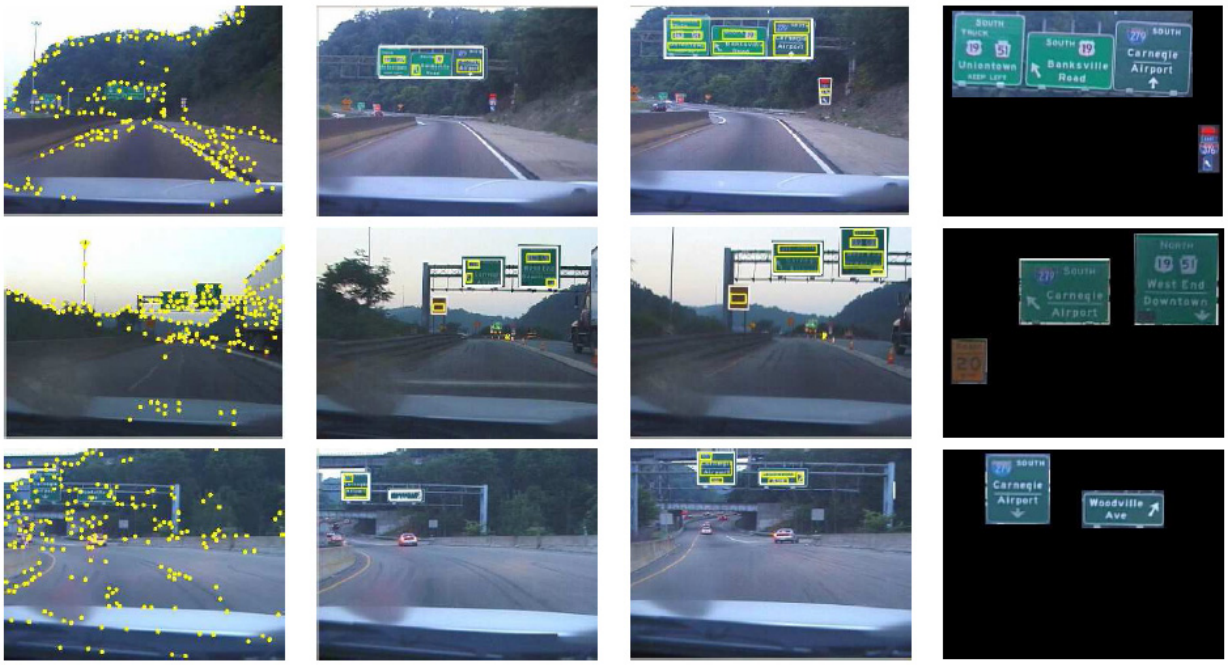

Figure 3.4.2: Three examples of incremental text detection process. One in each row. First column: selected discriminative points; second column: intermediate sign localization and text detection results; third column: final detection results before the road sign(s) disappear; fourth column: extracted road sign segments from video.

are merged and tracked over the time until all texts are detected or the road signs disappear from the frame. Finally, the image segments of extracted road sign are stored in the database and their appearance times and durations are recorded for other services such as indexing or retrieval. Row (b) shows the process of detecting another example of road signs from another sequence under dusk environment. Sequences in both (a) and (b) were captured when the vehicle was driving straight. However, the row (c) shows the case when the vehicle makes turning. As we analyze in the Section 3.4, when we assume a very small angle for the turn case, our algorithm can still handle the sign localization step well.

3.4.2 Overall Evaluation

Table I summarizes the sign localization performance. The system correctly detected 85 of 92 signs. For each frame, the ground truth for text bounding boxes was initially created by our system and further verified and adjusted manually. We apply the definitions of "hit rate" and "false hits" as defined in [72], and shown in Equation (3.4.1).

$$\begin{aligned} hit\ rate_{box-based} &= \tfrac{1}{M} \textstyle\sum_{g \in G, a \in A} \delta(a, g), \\ miss\ rate_{box-based} &= 1 - hit\ rate_{box-based}, \\ false\ hits_{box-based} &= N - \textstyle\sum_{g \in G, a \in A} \delta(a, g). \end{aligned} \tag{3.4.1}$$

where $A = a_1, ..., a_N$ and $G = g_1, ..., g_M$ are the sets of box sets representing the system assigned road sign boxes and the ground truth sign boxes. $N = |A|$ and $M = |G|$. Also, $\delta(a, g)$ is defined as:

$$\delta(a, g) = \begin{cases} 1, if\ min(|a \cap g|/|a|, |a \cap g|/|g|) \geq 0.8 \\ 0, otherwise \end{cases}$$

We used a slight variation on the evaluation methodology used by Lienhart. We directly counted the number of false hits instead of computing the ratio to avoid the bigger-than-one case. The text box-based hit rate, false hit rate, and miss rate refer to the number of detected boxes that match with the ground truth. A system assigned text bounding box was regarded as matching a ground truth text bounding box if only if only if these two boxes overlapped with each other by at least 80%.

TABLE I
RESULTS OF ROAD SIGN LOCALIZATION.

	Box-based	
Total # of road signs	Hit rate	False hits
92	92.4%	17 (17.9%)

Table II summarize the overall text detection performance. We compare two methods in the experiment: the baseline algorithm (which analyzes the whole image for every video frame) and

the proposed algorithm. There are a total of 359 words in the testing videos, including numbers and symbols such as arrows on all road signs. The new framework significantly reduces the false hit rate and achieves a higher hit rate than the baseline algorithm. The low false hit rate is mainly because of the two-step strategy of the new framework and a higher hit rate is due to the sign localization step before detection. The total false hit rate of text detection is 9.2% as shown in the Table II. On average, there are about two false positives per minute in areas without traffic signs.

TABLE II
RESULTS OF TEXT DETECTION.

Text Detection	Box-based		
	hit rate	false hits	miss rate
Refined baseline algorithm with text tracking	80.2%	307 (85.6%)	19.8%
Incremental algorithm with vertical-plane model	88.9%	33 (9.2%)	11.1%

Another merit of the new framework to mention is the improvement of processing bandwidth of the working system. Currently, the working system can process the video at the rate of 8 - 16 fps, which is about twice to three times faster than the baseline algorithm. The detection rate of 88.9% is still not good enough for real-world applications. Some signs are missed in the sign plane localization step as shown in Table I. On the other hand, sign tracking improves the text detection performance by reducing the number of false positives. At the beginning of stage, signs are localized but the characters on them are too small to be detected because the signs are far away from the vehicle. The feature-based tracker in our system tracks the localized signs in video, so it is possible to merge the newly detected text with previously detected text over the time.

3.5 System Prototype and Interface

Fig.3.5.1 shows an early prototype of the proposed framework for testing on indoor road sign printouts. Implemented functions include near real-time road sign detection, text recognition and

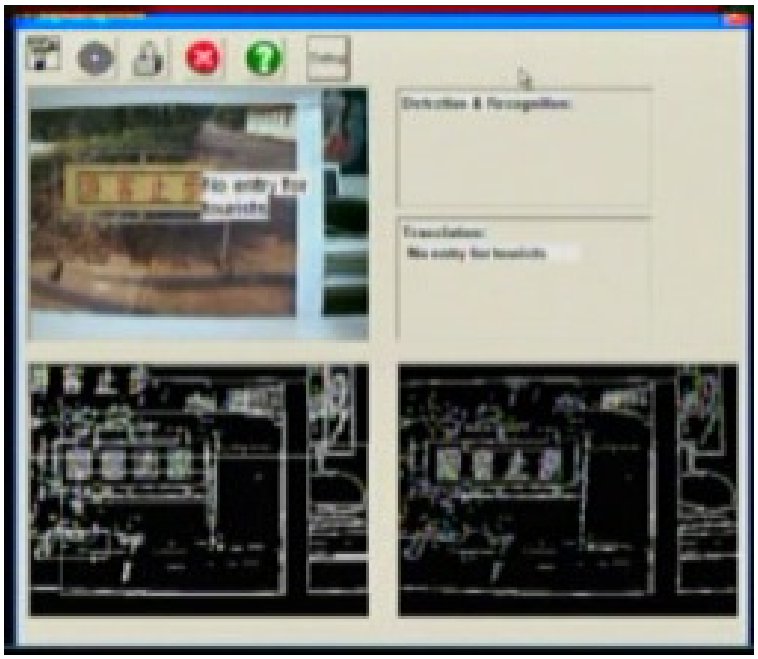

Figure 3.5.1: An early prototype for indoor road sign (printout) recognition. Functions include near real-time road sign detection, text recognition and Chinese-English translation. Bottom windows show intermediate results.

Chinese-English translation. Bottom windows show intermediate results. The sign show here was written in Chinese meaning ”No entry for tourists.”

Fig.3.5.2 shows the more advanced prototype (current version). Dynamic windows show multiple information and results: source video sequence, road sign detection results, road sign tracking results, text detection and segmentation, and text detection binarization. Our system can read in video sequences, or directly accept video stream from cameras, or read input images. Video-playing mode interface allows users quickly learn how to operate our system and process data.

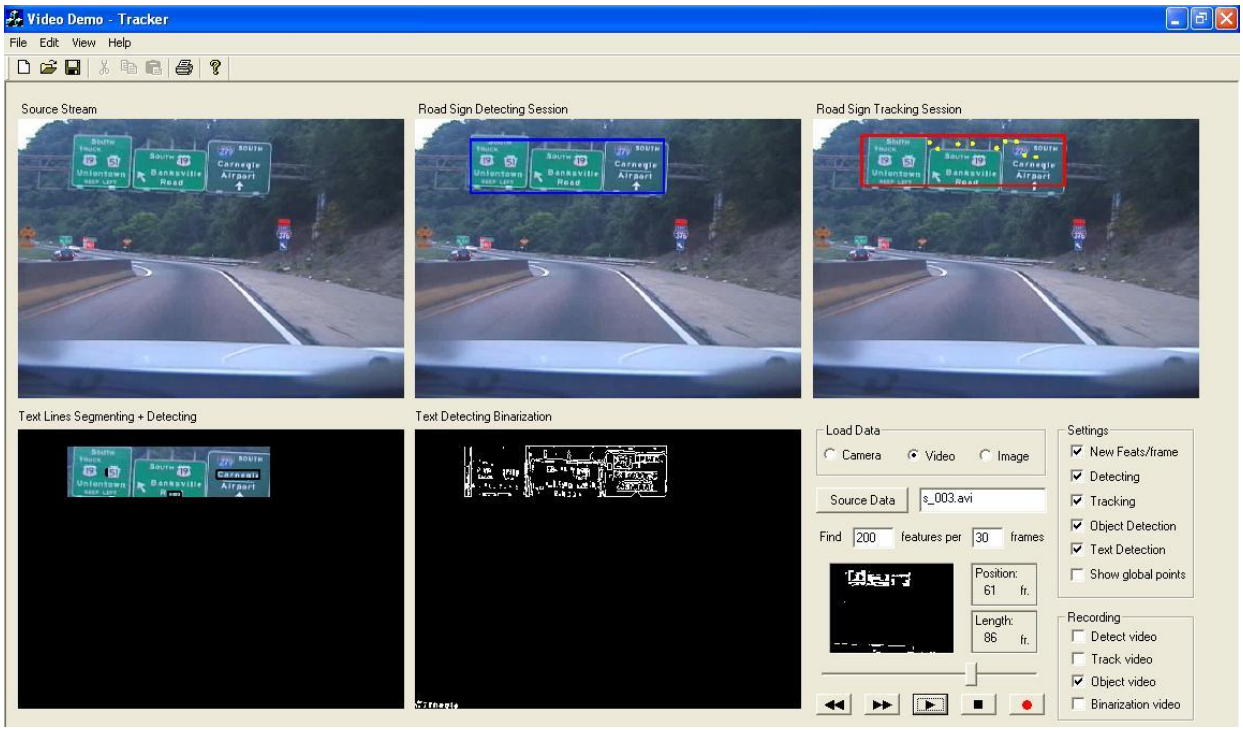

Figure 3.5.2: The more advanced prototype. Dynamic windows show, source video sequence, road sign detection results, road sign tracking results, text detection and segmentation, and text detection binarization

.

Chapter 4

Recognition of Other Signs

In the previous chapter we have shown how to detect text on road sings from video and its benefits to the landmark-based car navigation. In this chapter we will extend the work to recognition of other signs such as store signs.

4.1 Introduction

Unlike a text document, we do not have ”key words” in understanding a multimedia document. In order to analyze video data, we need some explicit structures. An object is something material that may be perceived by the senses and it can be a basic unit at the lowest level of such a structure. In video analysis, many algorithms can be largely categorized in *detectors*, such as face detector [126], text detector [119], and person detector [63], etc. Simple detectors can form a more complex detector, e.g., a face detector and a shot classifier can be combined as an anchor detector. Therefore object detection plays an important role in multimedia content analysis. Object detection determines whether or not the object is present, and, if present, determines the location and scale of each object in an given image. The challenge of this task is largely because of the infinite combinations of scale, orientation, viewpoint, lighting, and many other factors. To model such

variations, a machine learning algorithm usually requires extracting a number of features at every pixel based on the statistics of the surrounding pixel values. For example, we could train a good face detector using a large number of data [126]. For some content analysis applications, however, we might not have enough training data. For example, we frequently must deal with only one query image. In this research, we are interested in object recognition and localization with very limited training data (e.g., one image per object for training). An application of such a problem is street landmark localization which is important to location-aware multimedia applications and in-car navigation tasks.

In daily life, a street landmark means anything that is easily recognizable, such as a monument, a building, a store sign, or other structure [14]. In this research, we refer to street landmarks as objects that can be used for giving directions for driving navigation. In an urban environment, commonly used street landmarks include gas stations, fast food restaurants, and churches, etc. The research is a part of our efforts in developing landmark-based navigation technologies for drivers. Current GPS based navigation systems provide turn-by-turn instructions, e.g., turn left in 50 feet. However, human drivers often use landmarks for helping navigation. For example, we tell people to turn left after a BP gas station and then make a right at Starbucks. The application scenario is as follows: John is going to visit his sister Linda in another city, to which John is new. To help John, Linda sends him some street landmark images of her city on his route to her place. John will drive a car with a video camera that can capture the scene in front of his car. We would like to build a system that could help John to automatically recognize these landmarks from the video sequence. We assume that only one image per landmark is available for training in this task. In addition, the training images and the test video sequences always have different quality and resolution. An example from such a scenario is shown in Fig.4.1.1.

Although street landmark recognition is an object recognition problem, many general object recognition algorithms may not work well for this application because of the special characteristics of the problem. A fundamental challenge is lack of training data. Other challenges include rigid object recognition with different viewpoints, scale and illumination changes, partial occlusion,

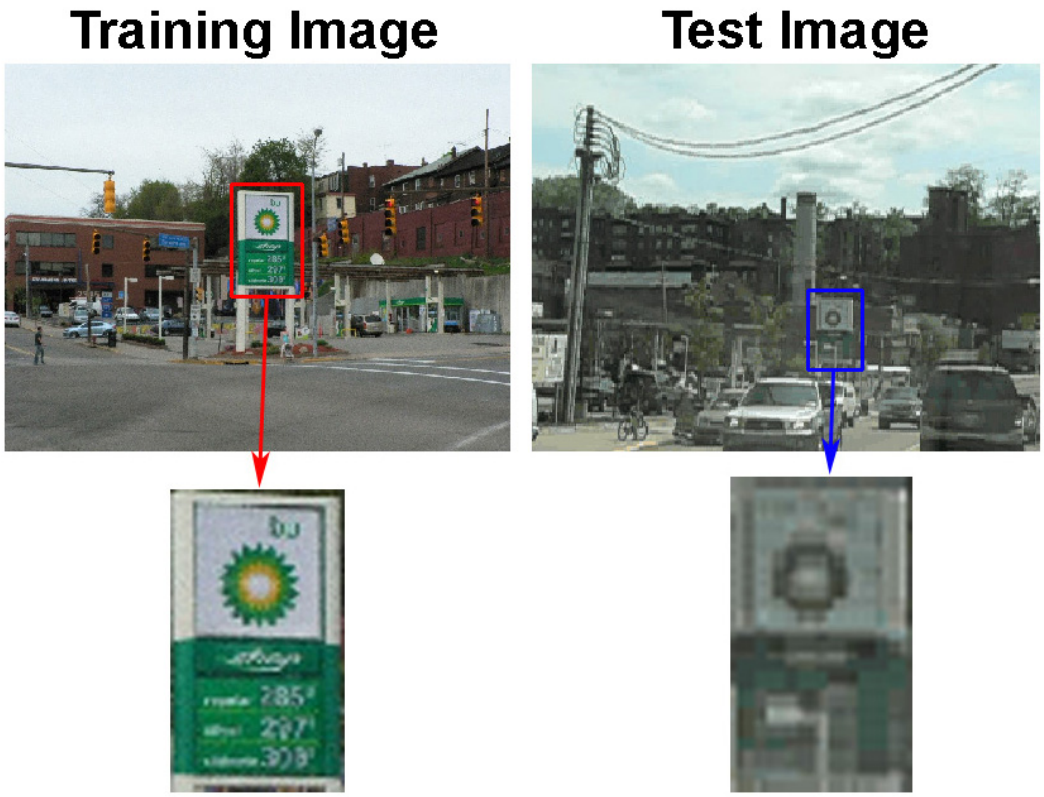

Figure 4.1.1: A BP sign example from our dataset. The training image, captured by a digital camera, has 1280×960 resolution, but the test image (a video frame) is 320×240. It is even hard for human eyes to recognize the BP sign from the right given the left training image.

and mismatching resolutions between the training and testing images. In order to address these challenges, we have proposed a unified approach for learning a model based on a given landmark image such that the system can recognize the landmark from a new street scene image. Our main insight is to identify a set of features that are unique to the landmark which we call fingerprints, and consider finding any fingerprints in a new image as recognition of the landmark.

We believe both appearance and context information of the landmark are crucial for landmark recognition. However we hypothesize that we do not need all the information to model the entire landmark in order to recognize it. Rather, we believe that the recognition of the landmark's fingerprints such as the star shape in the center of BP sign (Fig.4.2.1(a)), suffices to provide a robust

recognition of the landmark. The contributions of this work are:

- We have proposed a new landmark recognition approach by mining and matching landmark's fingerprints. Appearance and bag of features are surprisingly useful in general object recognition [105]. We combine them to extract unique fingerprints of a given landmark. Given a new image containing the landmark, our approach not only recognizes the landmark, but also estimates its scale and pose.
- Sliding window scanning is commonly used in object recognition, however, more researchers have advocated the use of image segmentation to get a better spatial support for recognition. Inspired by [78], we use multiple segmentations to extract landmark patches.

4.2 Sign Recognition as Fingerprint Matching

We highlight in this section key ideas of our fingerprint based landmark recognition approach. Fig.4.2.1 provides an overview of core steps of our approach. Let us first define how recognizing a seen landmark from an input image can be formulated as a fingerprint matching problem. During training, we construct a set $\Delta = \delta_1, ..., \delta_M$ of M fingerprints lying on the landmark. Fingerprints of a landmark are defined as a combination of informative features or parts such as edges, patches and keypoints. At runtime, given an input image Fig.4.2.1(c), we first detect salient regions to remove backgrounds Fig.4.2.1(d) and then decompose the image into a set of features Fig.4.2.1(e)-(g) from which we want to decide whether or not they contain the landmark's fingerprints. In other words, by representing the object through M fingerprints we transform the landmark recognition problem to a fingerprint searching problem Fig.4.2.1(h).

Variations of object scale and viewpoint are challenging issues in our problem. Keypoints have been shown to be effective for matching object within certain scale changes, while shape and appearance features can tolerate more scale variations. By combining two we may achieve robust feature combinations. In other recognition tasks, there are usually a large set of training

data available. However, for street landmark recognition, it is impractical to require the user to provide many landmark images. Most likely, we have only one image per landmark. In order to achieve robustness against viewpoint, scale and illumination changes, we synthesize many images of the landmark using a simple rendering technique to extract robust features.

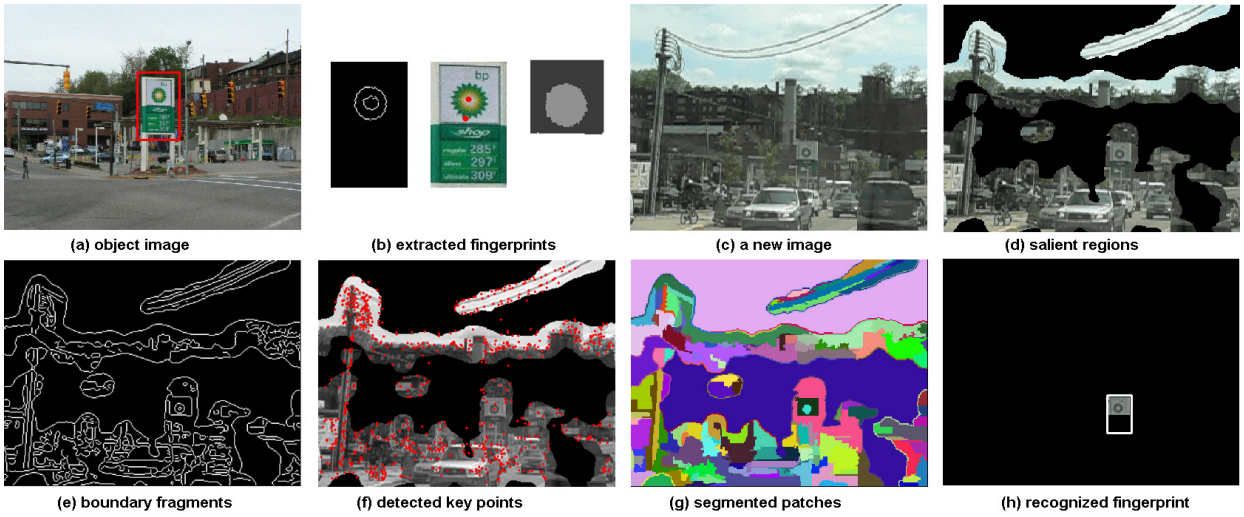

Figure 4.2.1: Given a training image with the landmark in red box (a), our method first extracts salient features to form a set of fingerprints. Here 3 bigram fingerprints of the BP sign are shown (b). For a new image (c), our method first detects salient regions (d) and extracts low and mid-level features via Canny edge detector, keypoint detector [75], graph-base segmentation [35] to obtain (e)-(g) in which it searches the landmark's fingerprints. (h) shows the matched landmark part and inferred boundary.

4.3 Extracting Fingerprints of a Sign

We describe here how to extract a set of fingerprints from a single landmark image. We assume that only one landmark exists in the image. The landmark boundary is provided automatically by another algorithm or manually by a user. The task of learning is to start with this single image, synthesize different views of the landmark, extract features, identify salient features to form a set of fingerprints.

Figure 4.3.1: An example of synthesized views for Taco Bell sign, and the extracted keypoints and edge features for these views.

4.3.1 Building Synthetic Views

Inspired by [68], we generate new views of the landmark and then extract features in these views as depicted in Fig.4.3.1. This method allows us to easily determine stable features from noises and perspective distortions, which in turn helps to make the matching in low quality images robust to the changes of viewpoint, illumination and clutter.

Since most street landmarks are rigid bodies and planar, a new view can be generated by warping the landmark image using an affine transformation which approximates homography. The affine transformation can be decomposed as: $H = R_\theta R_\phi^{-1} S R_\phi$, where R_θ and R_ϕ are two rotation matrices respectively parameterized by the angles θ and ϕ, and $S = diag[\lambda_1, \lambda_2]$ is a scaling matrix. Similar to [68], we use a random sampling of the affine transformation space, the angles θ and ϕ changing in the range $[-\frac{1}{4}\pi, +\frac{1}{4}\pi]$, and the scales λ_1 and λ_2 changing in the range $[0.15, 2.0]$. We use a much large scale range because of two reasons: 1) to select the most stable scale-invariant features and 2) to later recognize the landmark in different distances. Therefore, we obtain a set of landmark images, K, including the original landmark view for the landmark, O.

4.3.2 Feature Extraction

Local image features are basic building blocks of an landmark in the image. The spirit of our approach is to identify the landmark's unique fingerprints, and works independently of any particular

choice of feature detectors and descriptors. In this work, we extract edges, keypoints and patches. We choose Canny edge detector to extract edges for its simplicity. Long and close loop edges which are longer than 50 pixels are selected. For keypoints, we choose the SIFT detector for its robustness and rapidness [75]. Number of scales per octave is set to be 3 and $DoG\_thresh >= 0.01$.

It has been demonstrated that using multiple different segmentations of the same image can improve recognition accuracy [78]. Meanwhile, computational complexity and speed are not concerns during training. We choose three most popular segmentation algorithms, Normalized Cuts [101], Mean-Shift [27] and the FH algorithm [35], to generate the "soup of segments" for the given landmark image. For Normalized Cuts, we generate 7 different segmentations per landmark by varying the number of segments $k = 2, 3, 4, 5, 6, 7, 8$. For the Mean-Shift segmentation, we get 9 segmentations by setting $min\_region = size(landmark)/15$ and varying $spatial\_band = 5, 7, 9$ and $color\_band = 7, 14, 21$. For the FH algorithm, we get 12 segmentations by setting $\sigma = 0.5, 1, 1.5, 2$, $k = 200, 500, 1000$, and $min\_region = size(landmark)/15$. From these 28 segmentations, we let the user select the most satisfactory one as the landmark patch representation.

We apply Canny detector, SIFT and the chosen segmentation algorithm on the synthesized set K. For each feature, f_i, which is detected in the original view we define a missing score to measure how much percent the feature is not detected in K.

$$\Gamma(f_i) = 1 - \frac{C_j}{C},$$

where C_j is the number of synthesized images in which f_i is detected and C is total number of synthesized images. $\Gamma(f_i)$ smaller is better.

4.3.3 Finding Informative Features

The task of this section is to select the most informative features (appeared in the original view) about the existence of the landmark. This becomes extremely important when conditions of the

new image differ greatly from the original landmark image and only signature information of the landmark is preserved. We select 2000 natural scene images from Google Images and Flickr.com as SG. And for a particular landmark, O, we select 1000 images from SG, which do not contain O, to build a database S. For every image in S, we apply the same procedure to extract edge, keypoint and patch features, from which we select the informative features about O. Intuitively, we want to find features which occur on the landmark, but rarely or never occur anywhere else. This intuition can be formally implemented by the information gain criterion and commonly used for feature selection [98].

Information gain $I(A|B)$ is a measure how much uncertainty is removed from a distribution by adding some additional information. It is defined based on the entropy $H(A)$ and conditional entropy $I(A|B) = H(A) - H(A|B)$. In our problem, information gain $I(O|f_i)$ is defined with respect to the existence of the landmark O and a particular feature f_i. O is a binary variable that is true when the landmark is present and f_i is a binary variable that is true when the feature f_i is present. Therefore, the information gain of feature f_i at presence of the landmark O is:

$$I(O|f_i) = H(O) - H(O|f_i). \tag{4.3.1}$$

Recall our goal is to find those features on the landmark O which maximize this information gain value. Since the entropy $H(O)$ is constant across all features on the landmark, then maximizing $I(O|f_i)$ leads to minimize the conditional entropy $H(O|f_i)$. We can compute $H(O|f_i)$ from four terms: N,N_O, N_{f_iO} and $N_{f_i\overline{O}}$. The first two terms are constant: N is the total number of images; N_O is the number of landmark instances. The last two terms vary with feature: N_{f_iO} is the number of times feature f_i occurs on O. $N_{f_i\overline{O}}$ is the number times f_i occurs on other instances. Based on the definition of the conditional entropy, $H(O|f_i)$ depends on six probabilities and essentially is a function of N,N_O, N_{f_iO} and $N_{f_i\overline{O}}$. For simplicity, we substitute x and y for N_{f_iO} and

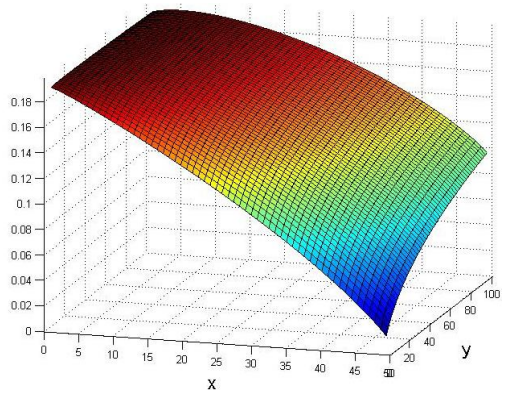

Figure 4.3.2: The minimal of conditional entropy $H(O|f_i)$ indicates the most informative feature with respect to the existence of O. It is a function of x and y, the number of times f_i occurs on O and at other locations. The figure shows $H(O|f_i)$ is minimized when f_i occurs on every instance of O and does not at any other location.

$N_{f_i\overline{O}}$ and we have

$$P(O|f_i) = \frac{x}{x+y} \quad P(O|\overline{f_i}) = \frac{N_O - x}{N - x - y}$$
$$P(\overline{O}|f_i) = \frac{y}{x+y} \quad P(\overline{O}|\overline{f_i}) = \frac{N - N_O - y}{N - x - y}$$
$$P(f_i) = \frac{x+y}{N} \quad P(\overline{f_i}) = \frac{N - x - y}{N}$$

From the definitions $H(A) = -\sum_x P(X = x) log P(X = x)$ and $H(A|B) = \sum_y P(B = b) H(A|B = b)$ we obtain

$$\begin{aligned} &H(O|f_i) = \\ &-\frac{x+y}{N}\left(\frac{x}{x+y} log\left(\frac{x}{x+y}\right) + \frac{y}{x+y} log\left(\frac{y}{x+y}\right)\right) \\ &-\frac{N-x-y}{N}\left(\frac{N_O - x}{N-x-y} log\left(\frac{N_O - x}{N-x-y}\right)\right. \\ &\left.+\frac{N - N_O - y}{N-x-y} log\left(\frac{N - N_O - y}{N-x-y}\right)\right). \end{aligned}$$

This equation shows that the information gain of a feature $H(O|f_i)$ is determined by a function of x and y as shown in Fig.4.3.2.

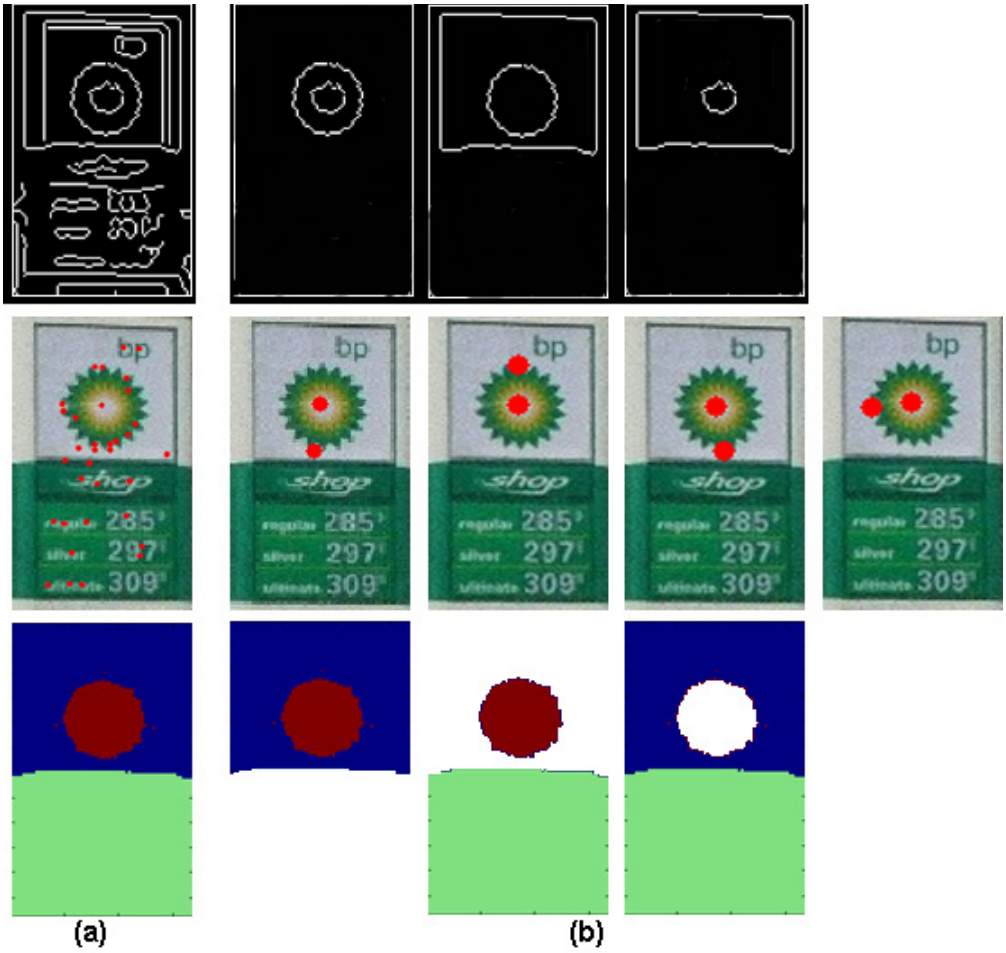

Figure 4.3.3: Extracted local and region features (a) and selected bigram fingerprints for each feature type (b).

4.3.4 Selecting Fingerprints

Motivated by [65], we use bigrams and triplets, which can be formed by different types of features, as our matching primitives. By combining two or three features into a set, we increase their discriminative power over single features. Similarly, we compute the conditional entropy for each bigram and triplet which appear on the landmark. However, the above definition of the conditional entropy only considers saliency of a bigram or triplet while ignoring the robustness of them to noises. To achieve the trade-off between selecting informative and stable features, we rank

extracted bigrams and triplets by using the following value.

$$Q(\delta_j) = (1 - \mu) \cdot H(O|\delta_j) + \mu \cdot \Gamma(\delta_j), \tag{4.3.2}$$

where $\Gamma(\delta_j) = \sum_{i=1}^{n} \Gamma(f_i)$, $f_i \in \delta_j$, is the sum of missing scores of each feature in δ_j, $n = 2$ for bigram and $n = 3$ for triplet. μ is a weighting factor. By computing $Q(\delta_j)$ for extracted bigram and triplet features on the landmark, we rank them in ascending order and keep top M to form the set of fingerprints Δ for O. Fig.4.3.3 depicts the selected bigram fingerprints from each feature type for BP sign. These M fingerprints are used to build the acceptance cascade to recognize O at runtime as shown in Fig.4.4.1.

4.4 Recognizing a Sign

Given a new image, $I1$, our task is to detect whether the landmark exists, localize where the landmark is and estimate its boundary. We go through the following steps: removing less-information regions (e.g. sky and ground), extracting image features, matching the landmark's fingerprints to the image features and finally estimating pose and boundary of the landmark if it exists.

4.4.1 Saliency Detection and Feature Extraction

During training, the user indicates the boundary of the landmark while for $I1$ we have no prior knowledge whether the landmark exists and where it is. Instead, to avoid greedy feature extraction on the whole image, we rely on a simple method for visual saliency detection [51]. By analyzing the log-spectrum of the input image, the method extracts the spectral residual of an image in spectral domain and constructs the corresponding saliency map in spatial domain (Fig.4.2.1(d)).

We follow the same procedure to extract image features as we do during training. We apply SIFT detector to extract keypoints which are represented by 128-dimension vectors. Canny edge detector ($\sigma = 1$) is applied to extract edge fragments. For patch features, we do not apply multiple

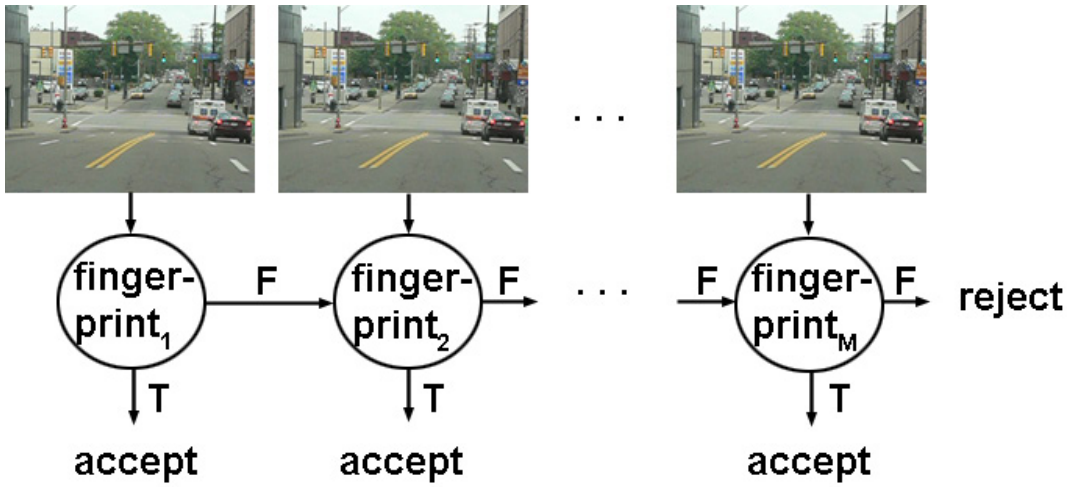

Figure 4.4.1: Cascade of fingerprints using 1-Nearest Neighbor classifiers. The most informative and robust fingerprints are listed in the beginning of the cascade. During recognition, once a fingerprint is matched, the landmark will be recognized.

segmentations due to computation concern while only apply the FH algorithm [35] by letting $\sigma = 0.5, k = 250$ and $min\_region = 50$. Thus, we obtain three feature maps for $I1$, as shown in Fig.4.2.1(e)-(g).

4.4.2 Matching a Fingerprint to the Image

For landmark recognition, we combine M 1-Nearest Neighbor classifiers with acceptance cascade, as shown in Fig.4.4.1. Different from face or human detection using rejection cascade which combines a number of weak classifiers, our fingerprint-based cascade method recognizes the landmark once verification occurs at any cascade level. At every cascade level, we match the fingerprint to the image via a pictorial model [36].

$$L^* = \arg\min_{L}(\sum_{i=1}^{n} m_i(l_i) + \sum_{(f_i,f_j)\in E} d_{ij}(l_i, l_j)), \tag{4.4.1}$$

where an candidate matching fingerprint is given by a configuration $L = (l_1, .., l_n)$, where l_i specifies the location of feature f_i. $n = 2$ for bigram fingerprint or $n = 3$ for triplet. $(f_i, f_j) \in E$

indicates f_i and f_j are connected. $m_i(l_i)$ measures the degree of mismatch when feature f_i is at location l_i, and $d_{ij}(l_i, l_j)$ measures the deformation of the model when f_i is at l_i and f_j is at l_j. For $m_i(l_i)$, we define it as Shape Context cost [7] for edge and patch features; for keypoints we use $1 - cos(a, b)$ distance metric. For $d_{ij}(l_i, l_j)$, we choose either feature in bigram, the mid-feature in triplet, as a reference point, say f_i and its matched position in $I1$, l_i. We then define $d_{ij}(l_i, l_j)$ as the L_2 distance from l_j to f_j. For triplets, the second term in Eq.(3) is sum of deformation values on two edges. Since the pictorial model in our case is a two-node or three-node tree, an efficient search algorithm [36] has been applied to obtain the optimal match.

The mismatch distance upper bounds for edge (ψ_e), keypoint (ψ_k) and patch (ψ_p) features are learned from the synthesized training set K. If the mismatch distance (the first term in Eq.(3)) from the fingerprint to the nearest bigram or triplet found in $I1$ is smaller than the sum of features' upper bounds, the detected bigram or triplet will be matched to the fingerprint and the landmark be recognized.

The pose of the landmark can be estimated by projecting back to the synthesized landmark views which give rise to the minimal residual on the matched fingerprints. In order to estimate the landmark boundary, we can either re-scale and overlay the boundary of synthesized view obtained from the last step, or compute the landmark image height from its image position, 3D height (obtained from the training image) and the viewpoint using the method introduced in [50]. We use the first method in this work.

4.5 A Dataset of Other Signs

Due to the lack of established research in street landmark recognition, it is difficult to obtain a standard dataset to compare our approach with. Thus, we compiled a dataset of 15 different US store signs including 4 categories (Fig.4.5.1): 1) gas station, 2) fast food restaurant, 3) pharmacy and 4) store. They are:

1. Shell
2. BP
3. Sunoco
4. Exxon
5. Gulf
6. KFC
7. Wendy's
8. Taco Bell
9. McDonald's
10. Pizza Hut
11. Arby's
12. CVS
13. Eckerd
14. Target
15. Lowe's

For each landmark, we captured one or a few frontal images for training, and recorded several test videos in the city of Pittsburgh.

4.6 Experiments and Discussion

Precision, recall, F_1 are computed for each landmark for evaluation. We consider a correct recognition if the matched fingerprint is within the landmark region. Parameters such as $\mu, \psi_e, \psi_k, \psi_p$ are varied to obtain precision-recall curves and $M = 20$.

Fig.4.6.1 depicts the detection performance using precision recall curves on four landmark examples, i.e., 1) BP, 2) KFC, 3) CVS and 4) Lowe's, which in general represents the detection patterns of general street landmarks. Each subfigure contains the precision recall curves of four detection algorithms, i.e., SIFT [75], bigram only fingerprints, triplets only and full model that uses both bigrams and triplets. We vary DoG_thresh and $match\_thresh$ in SIFT to generate curves. The figure shows several interesting observations. First, we observe that four methods consistently perform better on some landmarks than others. Best performance appears for the BP sign which can be well explained by its salient star logo with high contrast pattern. However, for landmarks which mainly contain text such as CVS, four methods all perform poorly due to lack of appropriate low-level features to capture saliency of text in our model. Second, we can see that three algorithms proposed in this work all perform better than SIFT on this dataset. Third, for most examples, bigram model achieves higher recall while lower precision than the triplet model since the triplet model requires matching one more feature for verification. Finally, by combining

bigram and triplet features, the full model obtain the best performance in F_1 across all examples.

To further examine the proposed algorithms, Table 4.6.1 lists a more comprehensive quantitative comparison for all 15 street landmarks. There are about 7000 labeled video frames (15fps).

Figure 4.5.1: Example images from our street landmark dataset, which consists of 15 different US street landmarks. (a) training images; (b)-(e) test video images.

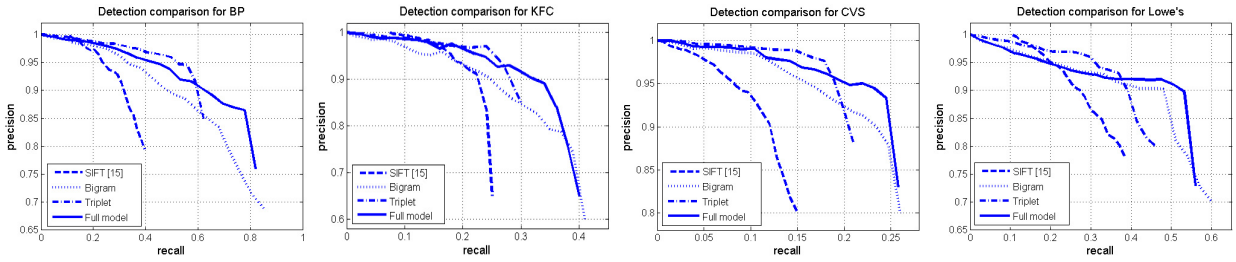

Figure 4.6.1: Precision recall curves using SIFT [75], bigram only fingerprints, triplets and full model. One street landmark is randomly selected from each landmark category to show the curve. They are BP, KFC, CVS and Lowe's in this figure.

F_1 is reported for each landmark using four algorithms, SIFT, Bigram, Triplet and Full model. All three proposed algorithms consistently perform better than SIFT for all landmarks. Overall, the average F_1 increases 32.02%, from 0.506 to 0.668. By combining both bigram and triplet fingerprints, the full model achieves the best performance for each landmark. As we can see, our approach performs very well on gas station landmarks which generally are big with salient appearance, performs well on fast food and store categories which also often contain signature logos or shapes. In contrast, pharmacy landmarks, such as CVS and Eckerd, only have texts in their signs which are hard to be recognized by our approach. In addition, they are much smaller than other landmarks, sometimes even hard to distinguish from the videos for humans. Another advantage of our approach is to combine low level features, i.e., edges and keypoints, with mid level features, i.e., patches, to detect landmarks in various scales which is essential for landmark recognition in driving videos. The full model usually runs for several seconds processing a video frame of 320×240 on a PC with 3.2GHz CPU and 2GB RAM. Our system combined with GPS information and more training data (more than one model image) can surely achieve better performance. We have simplified the problem setting and only focused on the recognition part in our work.

Organization of web image search results have been recently studied in the multimedia community [55]. To demonstrate that our object fingerprint-based approach can be generalized to other content analysis tasks, we apply our method to refine web image search results by re-ranking returned images. We used Google image search in this experiment. We assume that the No.1 returned

image is correct and use it as the model image to re-rank the rest of the top 100 images returned by Google. Instead of quitting after acceptance, we run matching through all object fingerprints in the cascade and count the number of matched fingerprints for each image. Use the count as the ranking score and images with same counts are ranked by their sizes. Images with at least one matched fingerprint are included in the re-ranking list. Fig.4.6.2 depicts the refined top results by

	SIFT	**Bigram**	**Triplet**	**FP**	**↑(%)**
Shell (404)	0.512	0.654	0.638	0.689	34.57
BP (448)	0.605	0.756	0.774	0.827	36.69
Sunoco (660)	0.619	0.721	0.718	0.761	22.94
Exxon (540)	0.633	0.719	0.685	0.756	19.43
Gulf (411)	0.625	0.727	0.659	0.776	24.16
KFC (488)	0.430	0.551	0.536	0.584	35.81
Wendy's (408)	0.448	0.579	0.523	0.608	35.71
Taco Bell (490)	0.667	0.780	0.729	0.815	22.19
McDonald's (519)	0.501	0.624	0.575	0.651	29.94
Pizza Hut (492)	0.485	0.692	0.687	0.708	45.98
Arby's (475)	0.472	0.611	0.586	0.637	34.96
CVS (415)	0.316	0.383	0.370	0.419	32.59
Eckerd (484)	0.293	0.368	0.354	0.386	31.74
Target (498)	0.469	0.732	0.681	0.740	55.22
Lowe's (427)	0.512	0.637	0.603	0.669	27.73
Avg_F_1	0.506	0.634	0.608	0.668	32.02

Table 4.6.1: Comparison of F_1 among 15 street landmarks. SIFT uses the implementation in [75]; Bigram: use only bi-gram fingerprints; Triplet: use only triplet fingerprints; FP: the full model with mixed bigram and triplet fingerprints, and ↑(%): F_1 improvement of the full model over SIFT. The number in the first column is the # of test images for each landmark.

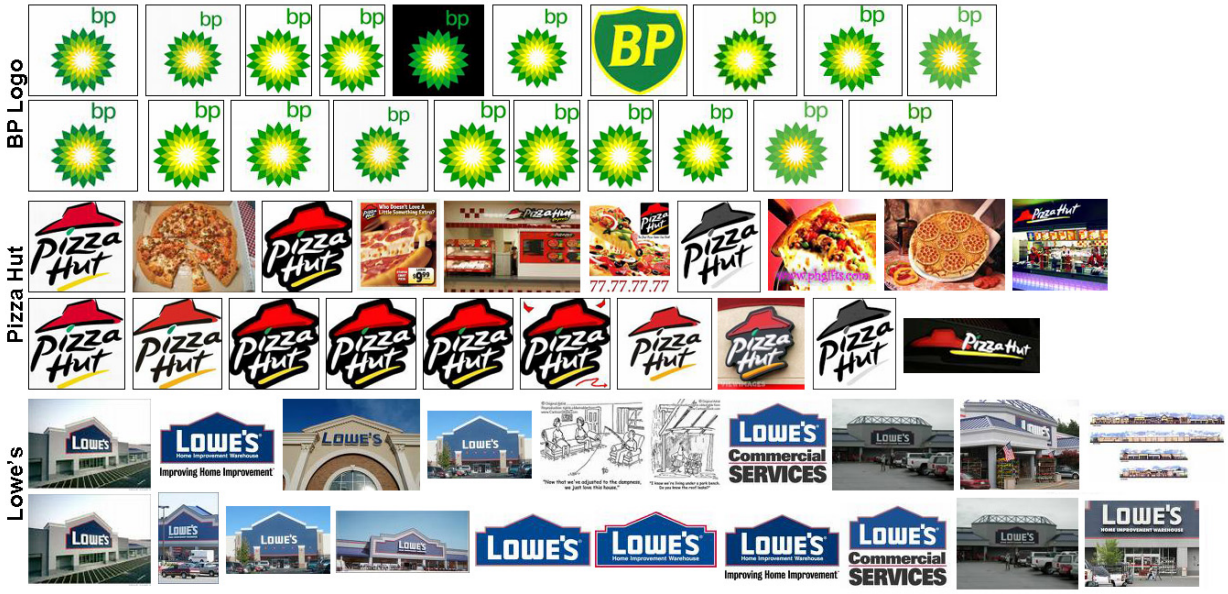

Figure 4.6.2: Refine web image search results by our approach. Google search results by 3 queries, i.e. *BP logo, Pizza Hut* and *Lowe's*, are shown. For each query, the first row shows top 10 images returned by Google and the second row shows refined top 10 images by our method using the first returned image as model image.

our approach. On the other hand, to achieve diversity of search results, our approach can also be applied to eliminate near-duplicates from search results.

Chapter 5

Recognition of Landmark Buildings

Landmark buildings are an important class of street landmarks which can be used as reference points for driving navigation. This chapter describes and discusses the process of recognizing landmark buildings.

5.1 Introduction

Commonly used landmark buildings include churches, hotels, office buildings, etc. Accurately recognizing landmark buildings poses some new challenges which are different from those of recognizing road signs or other signs.

- Unlike road signs or other signs, landmark buildings exist there not for reference. In other words, buildings do not provide navigation and geographic information. Basically, buildings are not salient compared to signs.

- Most buildings are occluded by nearby objects such as trees. Therefore, different view angles may result in different occlusion appearance.

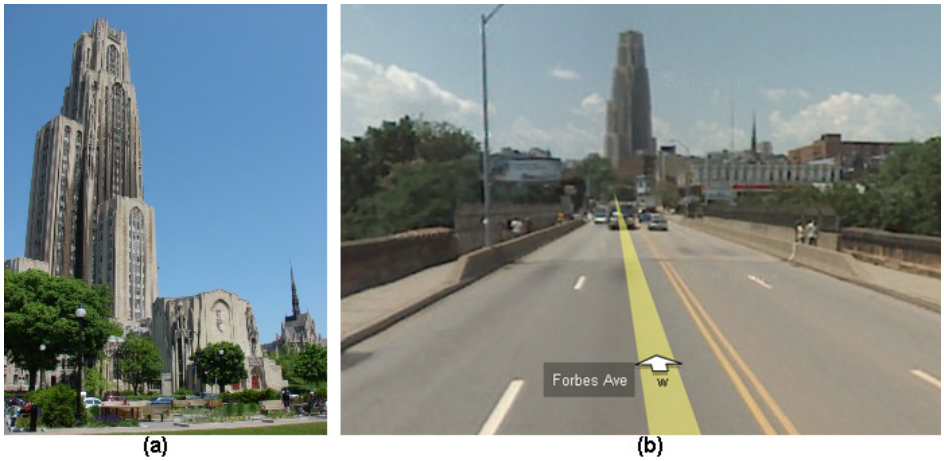

Figure 5.1.1: Two images of Cathedral of Learning (a landmark building in Pittsburgh): (a) from Flickr.com and (b) from Google Map Street View. Our application goal is to recognize the building in (b) using (a).

- Appearance changes due to lighting, environmental changes, weather or season changes.

These challenges prevent us from applying object fingerprint-based approach to recognize buildings. The recognition problem as considered in this application can consist of three phases: building recognition, building segmentation and pose recovery of the building with respect to the camera view. In this work, we only focus on the recognition aspect within the driving context. Iterate our application context as mentioned in the dissertation statement. John is going to visit his friend Susan in another city. Susan sends John some street landmark images of her city on his route to her place including some images of landmark buildings. John will drive a car with a video camera that can capture the scene in front of the car. In this chapter, we would like to build a system that could help John to automatically identify the target landmark buildings from the video sequence. We assume that only one landmark image is available for training in our task. Different from other related works, we are not tackling a general image retrieval task or an object recognition problem. Instead, we deal with a location-based building recognition problem in a driving context. Fig.5.1.1 gives a glimpse of problem challenges.

The problem of location and building recognition has been addressed in a number of publications in the past, mostly considering outdoors scenes. In [71], authors introduce a new mid-level feature, the consistent line cluster for building recognition in the context of content-based image retrieval. The color, orientation, and spatial features of line segments are used to group them into line clusters. The interrelationships among different clusters and the intra-relationships within single clusters are used to recognize and roughly locate buildings in images. Experiments on a database of color images of outdoor scenes show promising results. On the other hand, authors in [64] focuses the detection aspect and presents a generative model based approach to man-made structure (mainly buildings) detection in natural images. The proposed approach uses a causal multi-scale random field as a prior model on the class labels on the image sites. The distribution of the multi-scale feature vectors is modeled as mixture of Gaussians. A set of robust multi-scale features is presented that captures the general statistical properties of man-made structures at multiple scales. The proposed approach is evaluated on images from the Corel dataset. One most related work to our work is presented in [125]. Authors present a hierarchical approach for recognition of buildings. At the first stage, they use a novel and efficient representation named localized color histograms. This representation enables efficient retrieval of a small number of candidate images from the database. At the second stage, recognition is refined by matching local image descriptors associated with image regions. Once the correct building is identified, the relative camera pose with respect to the building is recovered. Two most recent works in the context of large-scale image retrieval are presented in [98, 127]. The first work [98] focuses at the problem of location recognition in a large image dataset using a vocabulary tree. The proposed method can find the location of a query image in a large dataset of more than 10^4 street-side images of a city. Authors introduce a generalization of the traditional vocabulary tree search algorithm which improves performance by effectively increasing the branching factor of a fixed vocabulary tree. The other work [127] utilizes the vast amount of multimedia data on the web, the availability of an Internet image search engine, and object recognition and clustering techniques, to build web-scale landmark recognition engine. First, a comprehensive list of landmarks is mined from two sources: (1) millions of GPS-tagged photos and (2) online tour guide web pages. Candidate images for each

landmark are then obtained from photo sharing websites or by querying an image search engine. Second, landmark visual models are built by pruning candidate images using image matching and clustering techniques. Finally, the landmarks and their visual models are evaluated by checking authorship of their member images. The resulting landmark recognition engine incorporates 5312 landmarks from 1259 cities in 144 countries. Although above related works are closely relevant to the problem in our application scenario, our problem focus and problem characteristics are still different from theirs.

Inspired by the work [125], we tackle the landmark building recognition problem by a three-stage approach. The first stage is comprised of an efficient coarse recognition scheme based on matching local descriptors between a model image and candidate driving images. A ranking list of candidate images is generated for the second re-ranking stage comprised of matching top candidate images to the rest of images. This step generates a number of re-ranking lists. The final stage is to merge re-ranking lists to a final list. SIFT descriptor are used in our proposed method and other descriptors can be used to replace SIFT ones.

For evaluation, we have conducted experiments on two datasets: 1) ZuBuD (201 buildings, 5 training images for each building and 115 test images) is a published dataset of Zurich buildings [99]. 2) Pittsburgh Historic Landmarks, a dataset which we have collected 65 Pittsburgh historic landmarks mainly including buildings. Training images for each building are collected from a online photograph collection provided by a photographer and test images are collected from the Google Maps Street View imagery. In the following, we present the three-stage building recognition approach followed by description of two datasets and experiments.

5.2 Recognition Framework

We have proposed to conquer recognition of landmark buildings in the driving context by a three-stage approach. The first stage is comprised of an efficient landmark building recognition scheme

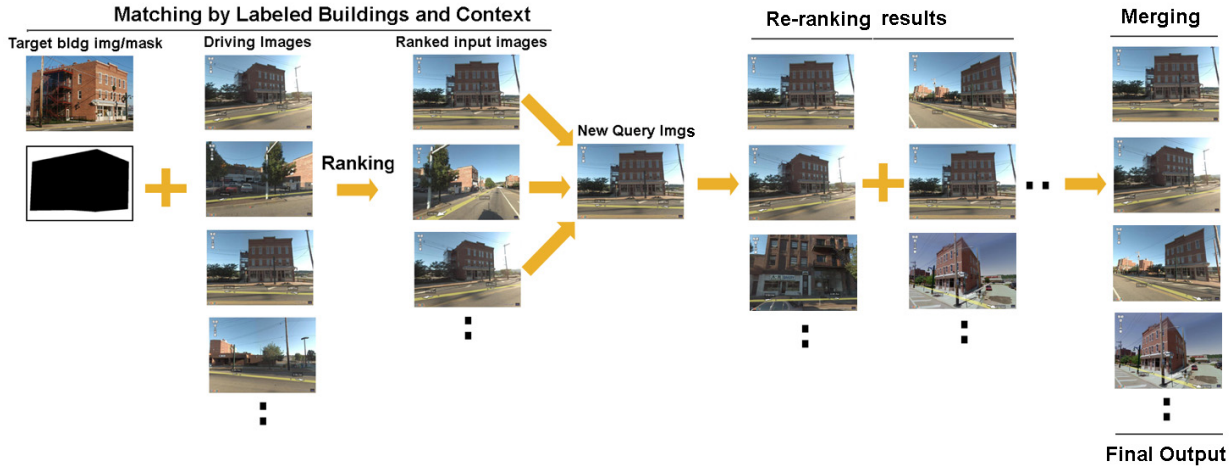

Figure 5.1.2: Our proposed approach to recognize landmark buildings to help car navigation.

based on matching local descriptors between a building model image M_i with its mask and candidate images $I_{ij}, j = 1...k$ captured approaching the location of the target landmark building. An initial ranking list of candidate images, named R_{iF} (short for the first), is generated for the second stage comprised of re-matching each of top candidate images in R_{iF} to the rest of candidate images in $I_{ij}, j = 1...k$. This is called re-ranking stage which generates a number of re-ranking lists $R_{ip}, p = 1...P$. The third also the final stage is to merge re-ranking lists $R_{ip}, p = 1...P$ to generate a final ranking list R_{iL} (short for the last) of $I_{ij}, j = 1...k$. SIFT-based keypoint detector and descriptors are used in our proposed method but other descriptors can be used to replace SIFT ones. Fig.5.1.2 shows an overview of our proposed approach to recognize landmark buildings for navigation while driving. In the following, we will first describe a handful of improvements over the SIFT matching method [75] and then our recognition method in details.

Note that our proposed re-ranking and merging is similar to pseudo relevance feedback (PRF), also known as blind relevance feedback. PRF works by doing normal retrieval to find an initial set of most relevant documents, to then assuming that the top k ranked documents are relevant, and doing relevance feedback under this assumption. Our approach is still different from PRF for two reasons. First, we do not assume the top k returned images are correct. Second, we are doing

image matching instead of document retrieval.

Some analysis of the following presentation is based on two datasets that we have used in our experiments: 1) Pittsburgh Historic Landmarks Dataset (PHL) dataset that contains color images of 65 historic landmark buildings/structures in Pittsburgh, USA, which we have built ourselves; 2) ZuBuD [99] dataset that is a public dataset including color images of 201 buildings in Zurich. Details of these two datasets are presented in the section before experiments.

5.2.1 Symmetrical Match of Local Features

Accurately matching extracted local features is crucial to ensure the quality of high-level applications such as object detection and recognition. The key issue is to define a reliable matching criterion so that correctly matched candidates are not missed while reducing mismatchings caused by background clutter or noisy data. Many state-of-the-art descriptor matching algorithms choose to use the criterion proposed by Lowe [75], which is defined as a threshold on the ratio of distance from the closest neighbor to that of the second-closest neighbor. The criterion has been proved to be reliable and robust in many applications [75]. However, there is an asymmetric phenomenon which often happens when we apply Lowe's matching algorithm on two images twice with different matching directions, one from I_1 to I_2, and the other from I_2 to I_1. The images in Figure 5.2.1 shows the two matching results by Lowe's method. As we can see, the matched features from (a) and (b) are not identical for the same two images. The reason actually roots from the definition of the matching ratio the Lowe's method uses. Assume a feature f_i in I_1, we find its best match f_j from I_2. But, in the other direction, for the feature f_j in I_2, f_i is not guaranteed to be f_j's best match in I_1 by the Lowe's matching criterion. This observation motivates us to put a symmetrical constraint on Lowe's matching criterion, which means, only if f_i and f_j are the best match to each other from both matching directions using Lowe's method, the pairs will be selected.

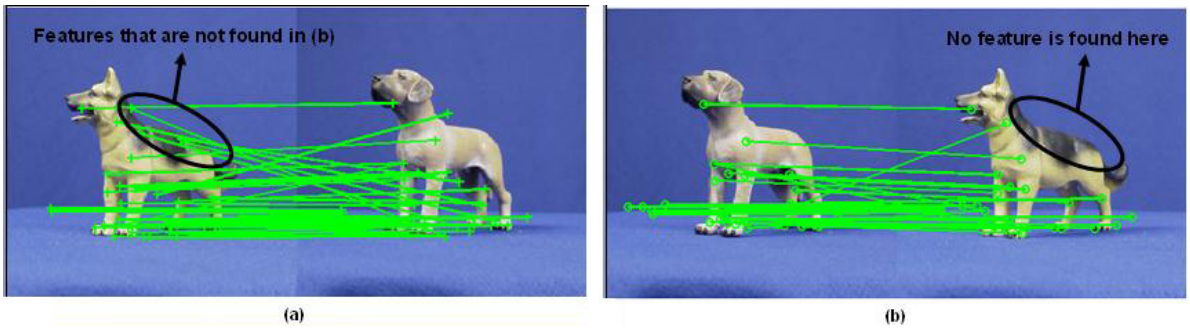

Figure 5.2.1: Asymmetric matchings obtained by Lowe's matching criterion [75]. Matching two identical images in two directions. Notice features selected on one dog's back in (a) are not selected in (b).

5.2.2 Geometric Matching Using RANSAC

In order to reduce mismatchings and refine correct matches, we adopt RANSAC (abbreviation for "RANdom SAmple Consensus") process to refine the standard SIFT matching. The RANSAC algorithm was first published by Fischler and Bolles in 1981 [41]. RANSAC is an iterative algorithm to estimate parameters of a mathematical model from a set of observed data which contain outliers. A basic assumption is that the data consists of "inliers", i.e., data whose distribution can be fitted by the mathematical model, and "outliers" which are data that do not fit the model. It is a non-deterministic algorithm since it produces a reasonable result (estimation of parameters of the underlying mathematical model) only with a certain probability, with this probability increasing as more iterations are allowed. Because the data can be subject to noise, the outliers can come from different sources: extreme values of the noise, or from erroneous measurements, or incorrect hypotheses about the interpretation of data. Fischler and Bolles assume that, given a small set of inliers, the RANSAC algorithm gives a procedure which can estimate the model parameters that can optimally explains or fits this data.

We first apply RANSAC algorithm to refine found matches based on homography constraint and then grow matches between two views from obtained seeds (matches). It is motivated by the

observation that buildings appearing in the images usually contain one or two dominant facades. The input to the RANSAC algorithm is a set of found matched pairs with coordinates, a parameterized homography matrix (model) which can explain the observations, and a error threshold parameter. The RANSAC algorithm finds the optimal (good) homography matrix by iteratively processing a number of randomly permuted correspondences (pairs of matched keypoints). It is assumed that among these data, some are hypothetical inliers. This assumption is then tested as follows:

- Do the following K times:

    1. From a randomly permuted set of correspondences, the first four correspondences are selected.
    2. A homography matrix is fitted to the four correspondences using equations in [49].
    3. All other correspondences are then tested against the fitted homography matrix and, if computed error to the estimated matrix is smaller than the threshold, the correspondence is considered as a hypothetical inlier.
    4. If the number of inliers is greater than the max_inlier_number (which is initially set to 0) the max_inlier_number is set to the number of inliers.

- Return the best fit matrix found (the one corresponding to the highest max_inlier_number).

The procedure is repeated K times ($K = 500$ if the number of correspondences is greater than 15; otherwise $K = 250$ in our study), each time the algorithm finds a model which either is accepted because more inliers are fitted or a rejected model with less number of inliers. Once the best fit matrix is found, all inliers are selected as grown correspondences. Fig.5.2.2 shows the results after RANSAC and the growing step between two views. Similar technique has also been applied in previous works such as in [97].

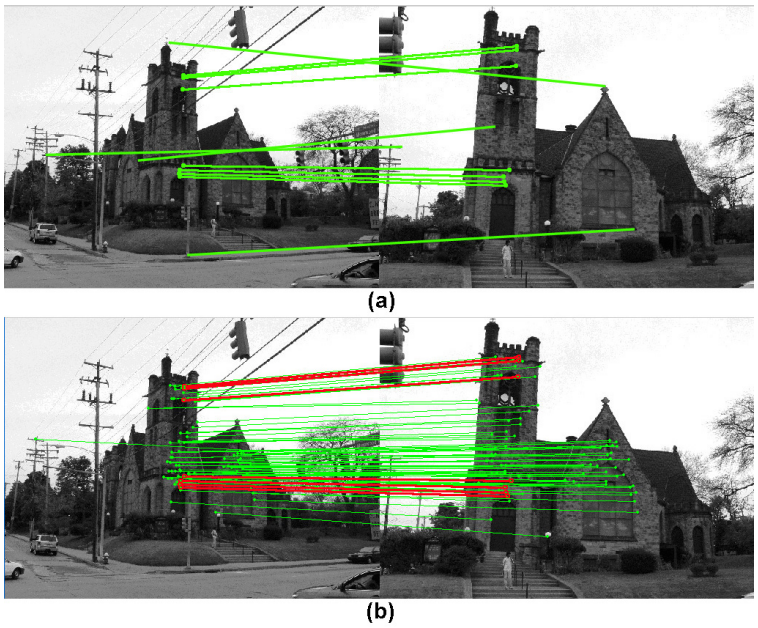

Figure 5.2.2: Sift matching between two views of a same building. (a) results after the standard SIFT matching [75] including mismatches. (b) Grown matches (in green) based on homography constraint after RANSAC. Red lines indicate correct correspondences after RANSAC.

5.2.3 Matching by Labeled Buildings and Context

Imagine our application scenario and problem: *The car is driving toward the location of the landmark L_i, the camera captures some scene images $I_{i1}, ..., I_{ik}$ (called driving images in the following) and has a model image (M_i) of L_i, the goal is to recognize L_i in $I_{i1}, ..., I_{ik}$ to help localize current position of the car with additional information such as GPS estimation.*

In our study, one online photograph of each landmark captured by one photographer is used as the model image. These model images have resolutions around 640×480. Images captured by the professional photographer, which have controlled view angle, optimal exposure and reduced occlusion, are good choices for model images. The above characteristics are also reasons that we choose online photographes instead of self-capture images. This setting also make it possible for our application scenario as close to the real world scenario as possible since a large amount of building and landmark imagery are available online and free to download. For driving images of

each landmark, which cannot be replaced by sidewalk-view images, we choose images from the Google Maps Street View. By doing this, we collect data to match the application scenario as much as possible.

To match a *model image* to each *driving image*, we first extract scale-invariant feature transform (SIFT)[75] keypoints on both images. We choose SIFT over other image descriptors such as color and edges because SIFT is less sensitive to scale change, rotation and occlusion which often happen in the driving context. This method is called *model*-based method in our study although it is different from the original algorithm in [75]. Matching is performed based on keypoints' SIFT descriptors. Numbers of correspondences are ranked across *driving images*. The following equation shows above the revised matching schemes of *Lowe's*,

$$\frac{d(f_1, g_2)}{d(f_1, g_1)} > \rho_d, \quad \frac{d(g_1, f_2)}{d(g_1, f_1)} > \rho_d \tag{5.2.1}$$

where f_1, f_2 are g_1's nearest and second nearest descriptors in the *model image* and g_1, g_2 are f_1's nearest and second nearest descriptors in the *driving image*. This matching scheme represents the previously proposed symmetrical match of local features.

We use a UCLA SIFT package [1]. We set the number of scales per octave as 3, keypoint selection threshold from $sTh = 0.001$ to $sTh = 0.03$ and keypoint matching criterion as $\rho_d = 1.4$ to $\rho_d = 3.2$. Fig.5.2.3 shows two studies of keypoint selection and match thresholds on the PHL dataset. Evaluation is performed by calculating average accuracy on top 10 returned *driving images* over all 65 landmarks. As we can see, $sTh = 0.01$ and $\rho_d = 2.0$ result in the best performance on the PHL dataset. In the following we will use this parameter setting as the baseline.

In previous works [75], keypoint selection and matching are usually performed over the whole image. However, this scheme may decrease recognition performance because analysis of background objects introduce noises and increase computation burden. On the other hand, background objects play a critical role in recognizing foreground objects in some cases such as foreground

[1] http://vision.ucla.edu/ vedaldi/code/sift/

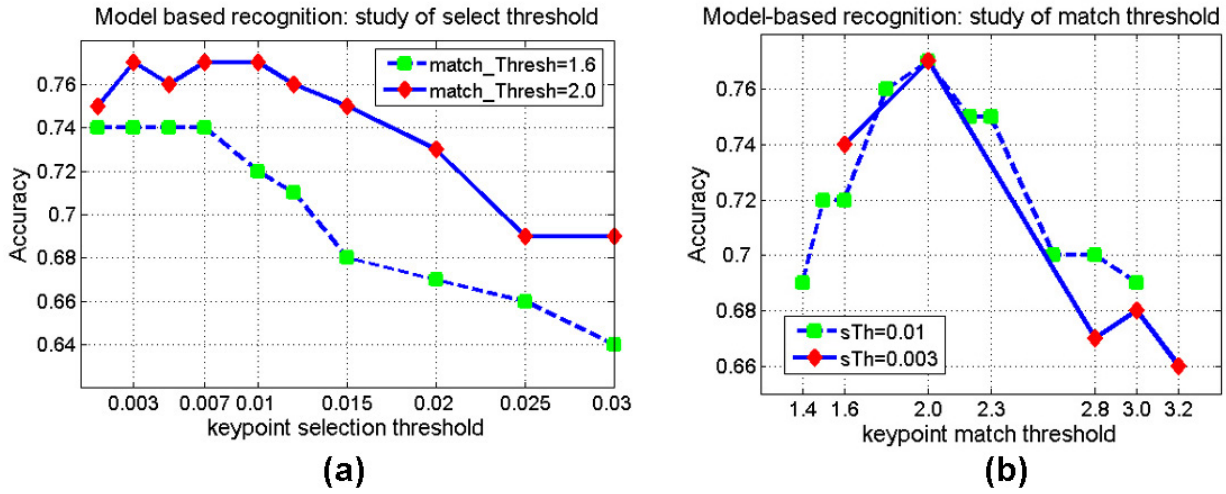

Figure 5.2.3: Study of keypoint selection threshold in (a) and matching threshold in (b) on the PHL dataset.

occlusion. Therefore, completely discarding background objects when selecting and matching keypoints is not a good choice. We introduce a parameter R called landmark_to_context ratio to balance contribution of foreground/background objects in the matching step. The PHL dataset provides landmark masks (object/non-object map label) and enables our proposal.

$$num(I_{ij}, M_i) = num_l(I_{ij}, M_i) + \frac{num_c(I_{ij}, M_i)}{R}, \tag{5.2.2}$$

where $num(I_{ij}, M_i)$ are the number of correspondences between the j-th driving image of the landmark L_i and the model image M_i; $num_l(a, b)$ is the number of correspondences which are within the landmark mask and $num_c(a, b)$ is the number of rest correspondences. Our experimental analysis confirms our expectation and the proposed idea of the new parameter R. Fig.5.2.4 shows the experimental analysis. From the figure, we have two observations. First, the new approach (called model-based in the graphes) generally performs than the matching over the whole image. Secondly, $R = 2$ and $R = 50$ lead to the same recognition performance for the model-based method and RANSAC-based method. In the following analysis, we set $R = 50$.

Except the *Lowe's* method and the *RANSAC* based method (called *ransac*), we have also devel-

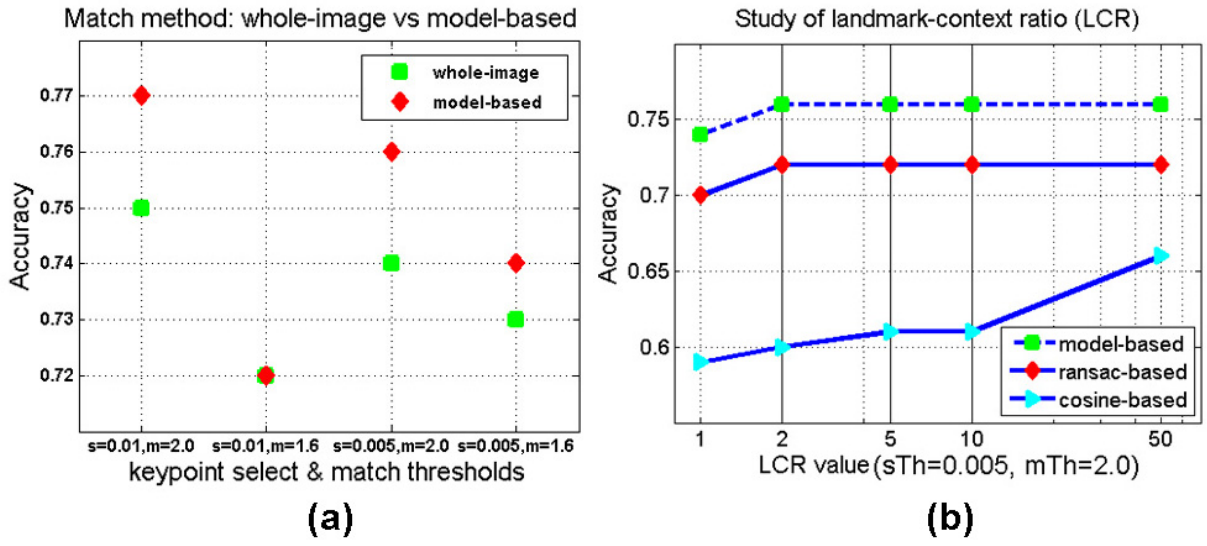

Figure 5.2.4: Analysis on the PHL dataset: (a) comparing matching by whole image vs. landmark part; (b) study of the landmark-to-context ratio (R).

oped another method (called *cosine*-based method) and it is inspired the work in [125]) by adding $cosine$ matching criterion (performing union operation over the correspondences found by *Lowe's* method). This approach is inspired by the promising performance of adding $cosine$ criterion reported in [125]. The observation is that the *model* method is effective but not in all cases, in some of which the *cosine* method performs better. The matching scheme is as follows:

$$cos(f, g) = \frac{f^T g}{\|f\|_2 \|g\|_2} > \rho_c. \tag{5.2.3}$$

where f, g are descriptors in two images. Different parameters lead to different performance for the *ransac* method and the *cosine* method. Fig.5.2.5 shows the experimental analysis. As we can see from the figure, the higher $cosine$ threshold lead the better performance for all three sets of keypoint selection and matching parameters. On the other hand, for the better-performance parameter set $sTh = 0.005$ and $\rho_d = 2.0$, setting *RANSAC* error threshold from 3 to 9 results in the similar performance. In the following experiments, we set the $cosine$ threshold as 0.995 and the *RANSAC* error threshold as 9.

Fig.5.2.6 shows the keypoint match number ratio between the first correct and the first wrong

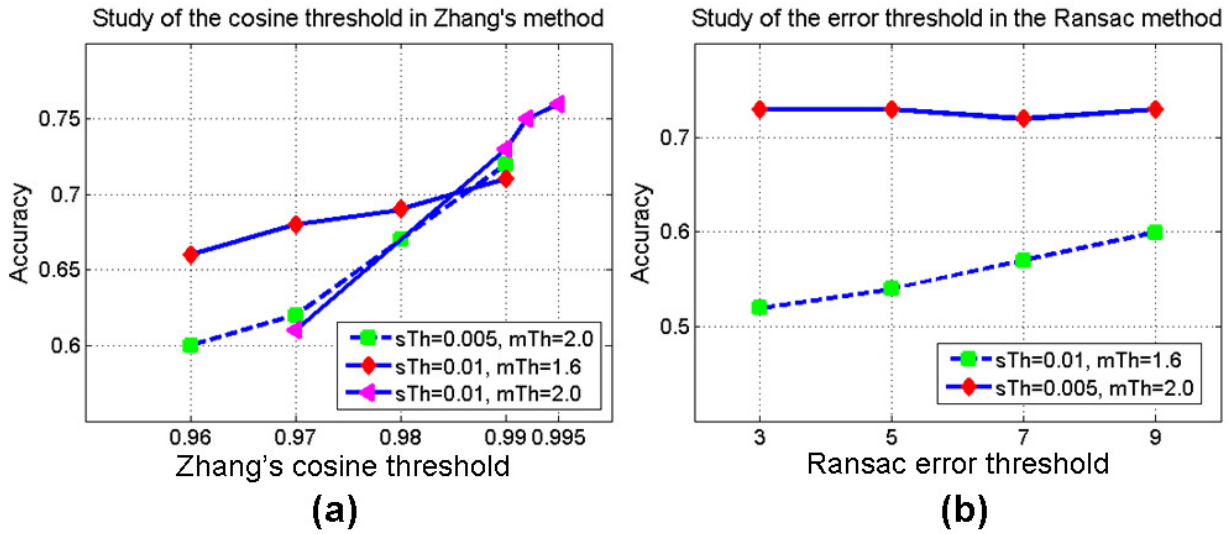

Figure 5.2.5: Analysis on the PHL dataset: (a) study of cosine threshold for Zhang's method; (b) study of error threshold for the Ransac method.

on the PHL dataset. Comparison is shown among all three methods: *model*, *ransac* and *cosine*. The higher ratio means the larger distance between the first correct and the first wrong which shows more robust recognition behavior. From the figure, we can see the *model* method (symmetrical matching criterion) is more robust than the other two methods. Except for a few landmarks, the

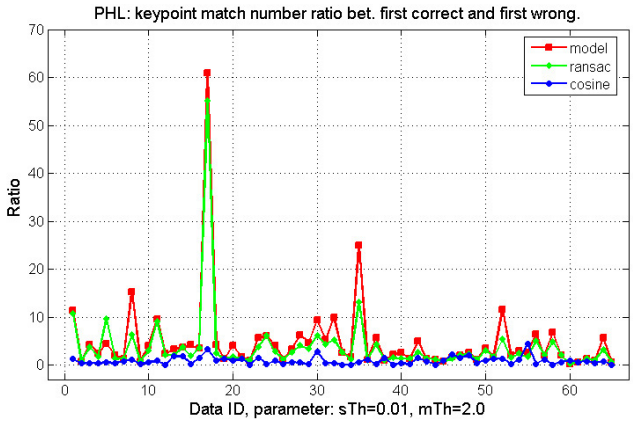

Figure 5.2.6: The match number ratio (between the first correct pair and first wrong pair) for 3 methods on the PHL dataset. The higher the better.

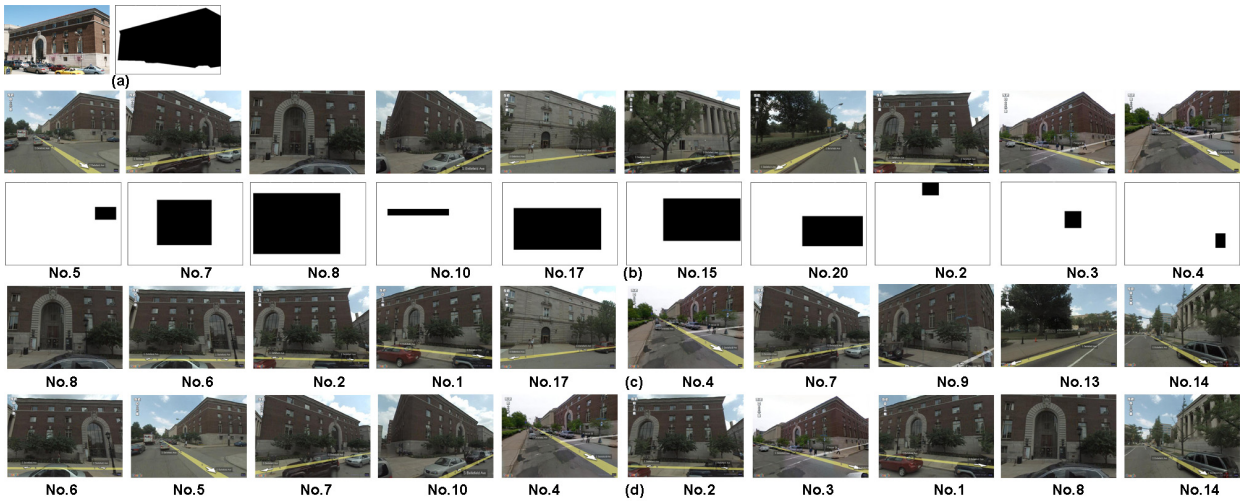

Figure 5.2.7: Illustration of our re-ranking and merging algorithms. (a) a model image (No.22 landmark [Bellefield Hall] in the PHL dataset) and its mask; (b) top 10 images returned by the Lowe's method using the model image and its mask as query; areas of repeating patterns (correspondences) and image number are also shown; (c) results after re-ranking using No.8 image (the third in (a)) and its mask as query; (d) results after merging results of re-ranking all top 10 images in (a).

ratio is generally below 10 for most landmarks.

5.2.4 Re-Ranking by Associating Repeating Patterns and Merging

Having improved the matching scheme the next step is to improve recognition performance using re-ranking and merging algorithms. Re-ranking and merging ranked lists returned by different search engines is a challenging information retrieval (IR) task because different retrieval algorithms may be used to search the databases, and because different databases have different corpus statistics. Both problems have attracted much attention in the IR community. Many solutions have been proposed in the IR literature and are quite effective. Here, we formulate recognition of a landmark building from driving imagery as an image matching and ranking task. Inspired by the rich literature of merging search engines' results in the IR area, we seek effective algorithms from

the literature since we are solving a similar but simplified task.

Fig.5.2.7 shows illustration of steps of our re-ranking and merging algorithms. Fig.5.2.7(a) shows a model image (No.22 landmark [Bellefield Hall] in the PHL dataset) and its mask. Fig.5.2.7(b) shows top 10 images returned by the modified *model* method using the model image and its mask as matching query. Below each result image are the area of repeating patterns (correspondence region) and image number. Correspondence region is computed as the minimal bounding box which covers all correspondence points. Note the size of correspondence region does accurately reflect the number of correspondence points. Fig.5.2.7(c) show results after re-ranking using No.8 image (the third in Fig.5.2.7(a)) and its mask as matching query. Fig.5.2.7(d) show results after merging results after re-ranking using all top 10 images in Fig.5.2.7(a) as matching query. Images with numbers that are greater than 10 are the ones that do not contain the target landmark, thus are falsely recognized images.

A novelty here is to use the result merging algorithm associated with the CORI resource selection algorithm (called CORI in the following) [15]. CORI algorithm uses a simple heuristic to normalize database-specific document scores. Here database-specific document scores mean re-ranking trial specific image scores. The higher score means the higher rank. The normalized image score suitable for merging is calculated as follows. Again, notations are as follows. *The car is driving toward the location of the landmark L_i, the camera captures some scene images $I_{i1}, ..., I_{ik}$ (called driving images in the following) and has a model image (M_i) of L_i, the goal is to recognize L_i in $I_{i1}, ..., I_{ik}$ to help localize current position of the car with additional information such as GPS estimation.*

To simplify, we just consider one landmark here, L_i. Firstly, we transform image ranks of each trial (re-ranking lists in Fig.5.1.2, up to 10 trials in our study)to scores ranging $[0, 1]$. Illustration of different trials is shown in Fig.5.1.2. The higher rank gets the greater score.

$$\bar{X}_q = 1 - \frac{X_q}{d}, \tag{5.2.4}$$

where X_q is indices of images of the q-th trial (the one using q-th image in R_{iF}) and d is the maximal rank and equal to 20 in our experiments (the minimal rank equal to 1 indicates the top 1 result, the highest rank).

Secondly, we normalize rank scores to the range $[0, 1]$ based on importance of trials. In other words, we normalize the scored calculated from the above equation by multiplying a coefficient associated with q.

$$X'_q = \frac{\bar{X}_q + \alpha \times \bar{X}_q \times \delta_q}{1 + \alpha},\ where \quad \delta_q = \frac{r + 1 - q}{r} \tag{5.2.5}$$

where r is the total number of trials and equal to 10 in our experiments. α is a coefficient to balance the importance of individual trial scores and ranks of trials (aka q). We tried different values of α and found similar resulting performance. In the following we set $\alpha = 0.4$.

Lastly, we sum up X'_q scores from the first trial to the cutoff (called c), and sort the merged array and generate the final ranking list R_{iF} for the landmark building L_i. Once the R_{iF} is produced, recognition results are evaluated by verifying the top x images to the ground truth.

$$R_{ic} = \sum_{i=1}^{c} X'_q; \tag{5.2.6}$$

$$R_{iF} = sort(R_c, ' descend'); \tag{5.2.7}$$

5.3 Two Datasets

5.3.1 ZUBUD Dataset

ZuBuD [99] is a database of color images of 201 buildings in Zurich. Each building is represented by five snapshot taken from five different viewpoints which result in 1005 color images not

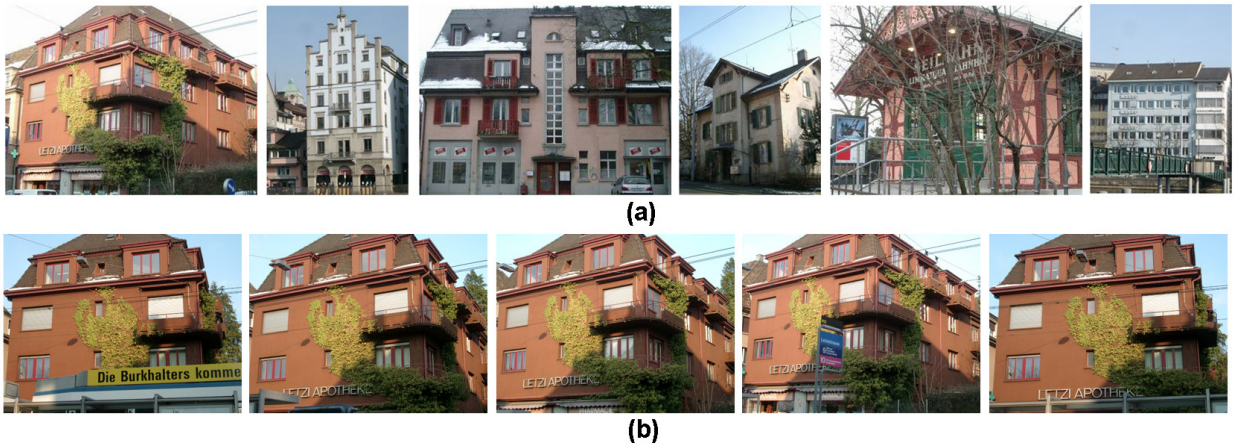

Figure 5.3.1: Examples from Zubud dataset: (a) query images; (b) training images of one of buildings shown in (a).

including test images. Illumination conditions may vary for different buildings/scenes. All the buildings/scenes are random selected buildings in Zurich which is the biggest city of Switzerland. The database, called ZuBuD, has been used to test image based recognition algorithms

All the images' resolutions are fixed at 640×480, without flash. In the database, for each building or scene, five images are acquired at random arbitrary view points. The database includes a wide variety of photometric conditions since all the images are taken under different seasons, weather conditions and by two different cameras. For some images purposely some occlusions by tree and other objects are included. Fig.5.3.1 show some examples of query images and 5 training examples of one of buildings.

Altogether 115 query (test) images have been collected to test the recognition performance. All these images contain the buildings included in the database, however the imaging conditions between query and training images do not match exactly. These query images are captured at different viewpoints and under varying illumination.

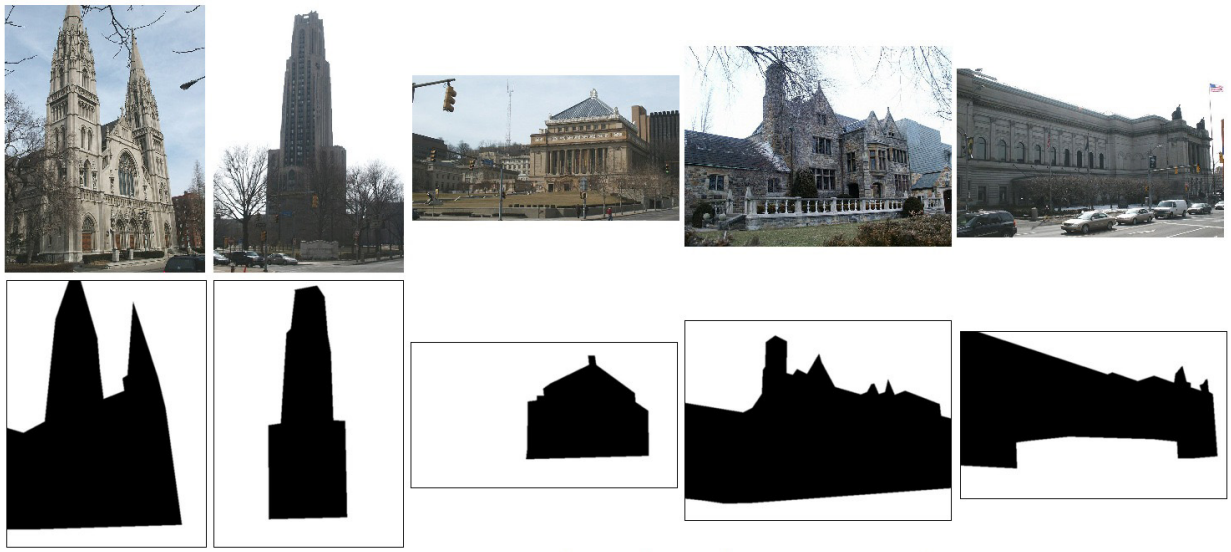

Figure 5.3.2: Some PHL training examples and their masks.

5.3.2 Pittsburgh Historic Landmarks Dataset

Pittsburgh Historic Landmarks Dataset (PHL) is a dataset of color images of 65 historic landmark buildings/structures in Pittsburgh, USA, which we have built ourselves. These are representative landmark buildings/structures in Pittsburgh. We first download a collection of 79 landmarks' photographes with their location GPS coordinates from http://www.geosnapper.com by the photographer Kordite. These images' resolutions are about 640×480. From these we select 65 landmarks (mainly buildings) into the dataset as landmark building model images. We manually label the object mask for each image. Fig.5.3.2 show some PHL instances and their masks.

To collect real world driving imagery, we choose to collect images from Google Maps Street View. Google Maps Street View is a newly launched feature. With Street View, we can virtually explore city neighborhoods by viewing and navigating within 360-degree scenes of street-level imagery. Currently Google Maps Street View provides street-level imagery for many cities, towns, parks and remote parts of the world including Pittsburgh. Street View imagery is gathered by vehicles equipped with advanced imaging devices and technology, driving on public roads. At this time, Google is not accepting photo submissions for inclusion in Street View. Because he

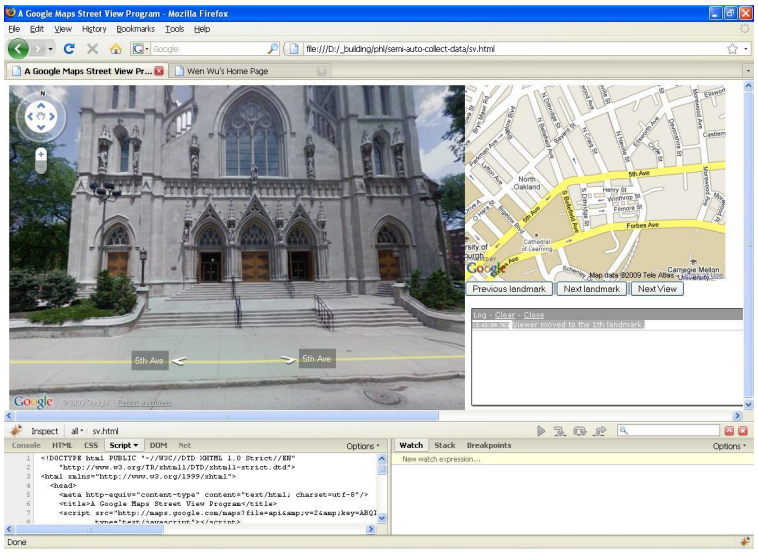

Figure 5.3.3: PHL data collection interface based on Google Map API.

images were taken in the past year by specialized vehicles driving around the cities, Street View imagery is not real-time and does not reflect weather, season and other changes. Street View mainly provides three functions. First, users can view street level photographs; secondly, users can take virtual walks in 360-degree scenes; they can pan, rotate and zoom through cities around the world; thirdly, users can find shops, restaurants, parks, hotels and other points of interest.

There are no available tools to automatically collect images from Street View. However, the Google Maps API allows users embed Google Maps in their own web pages with JavaScript. The API provides a number of utilities for manipulating maps (just like on the http://maps.google.com web page) and adding content to the map through a variety of services, allowing users to create robust maps applications on their websites. Therefore, we build a data collection interface in Javascript using the Google Maps API. The interface allows users to walk through all 65 Pittsburgh landmarks (whose GPS coordinates are stored in advance) and view/save images from the Street View imagery which are at or near each landmark location. Fig.5.3.3 shows the interface. Users can use buttons on the right to navigate through 65 landmarks and images at each location. The bottom windows are for JavaScript debug purpose.

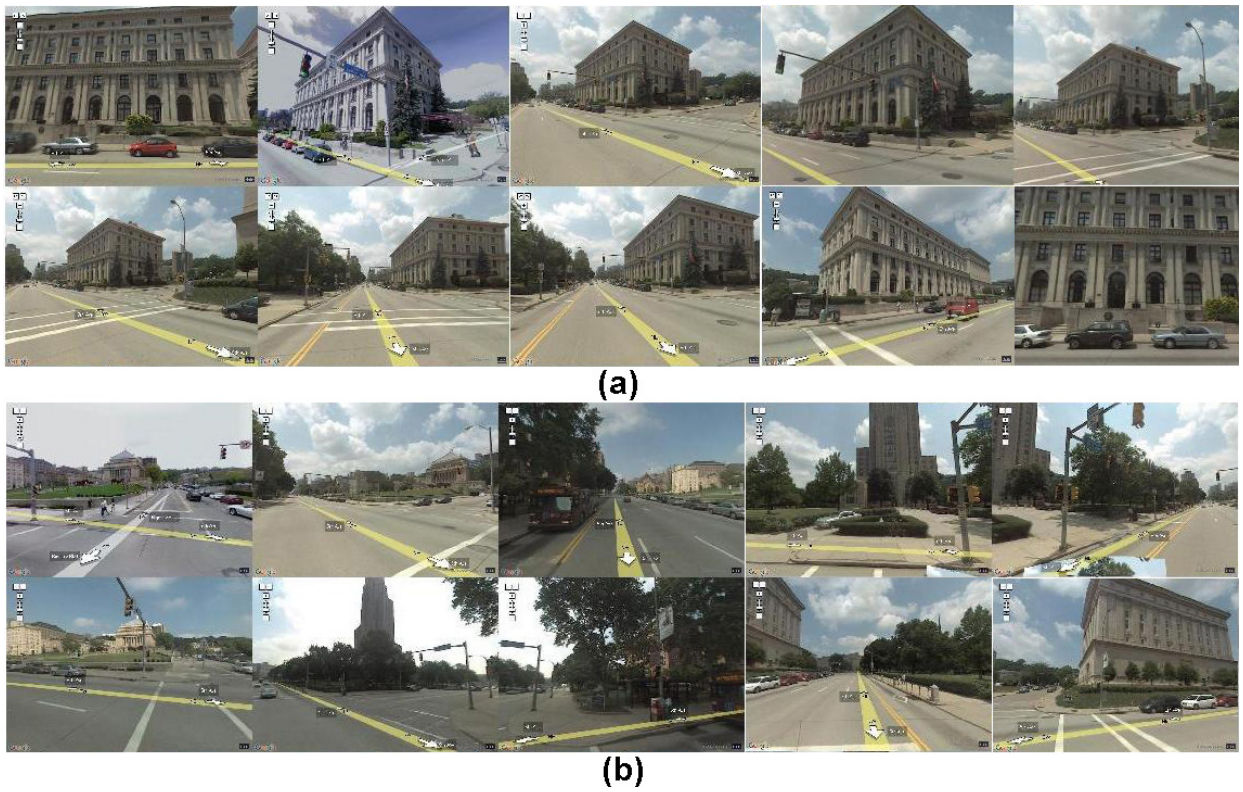

Figure 5.3.4: PHL test examples: (a) 10 Street View images that contain No.11 landmark building (Pittsburgh Athletic Association); (b) 10 Street View images around the location that do not contain No.11 landmark building.

To simulate the real-world driving context, we collect 10 images at each landmark's location that contain the landmark by human judgement and 10 images within 50 meters from the landmark's location that do not contain the landmark. The problem is formulated as follows: given each landmark's model image and its mask, identify 10 Street View images from the 20 images that contain the landmark. Fig.5.3.4 shows 10 images that contain the No.11 landmark building (Pittsburgh Athletic Association) in the PHL dataset and 10 images around the landmark location that do not contain the building. Images collected from Street View have resolution at 700×500. Note that lighting and weather conditions remain relatively constant across the Google Street View data. To protect privacy, Google blurs faces in the imagery. Fig.5.3.5 shows two common Street View errors we encountered during the data collection.

5.4 Experiments and Discussion

To evaluate our proposed recognition framework on our target task - recognizing landmark buildings in the driving context, we have performed experiments on two datasets: ZuBuD dataset and PHL dataset. Some analysis and much discussion have been presented in the previous sections. In our experiments, only one kind of low-level image feature is used: SIFT keypoints and descriptors. We have described the parameter settings of the SIFT algorithm and our proposed matching schemes in previous sections. Average accuracy computed over the top x images of the final ranking list R_{iF} for all landmarks is the chosen evaluation metric in our experiments with $x = 10$. In the following we will show results on the ZuBuD dataset and the PHL dataset respectively.

5.4.1 Results on ZuBuD dataset

Table.5.4.1 summarizes the results on the ZuBuD dataset. Experimental setting is the same as the first experiment in [125]. The first views of the 201 buildings are chosen as models and the second views are chosen as test images. The table lists the results given two different parameter sets whose descriptions are given previously. err and cos are the error and cosine thresholds used in the *ransac* and *cosine* methods. Evaluation (accuracy) is performed on the first ($1st$), top 5

Figure 5.3.5: Errors of Google Street View images: (a) mismatch between actual address and shown address; (b) corrupted image.

	1st	top5	top10	avg_rank	match_num_ratio
parameters	sTh=0.01,mTh=2.0,R=50,err=7,cos=0.99				
model	1.000	1.000	1.000	1.000	66.27
ransac	1.000	1.000	1.000	1.000	52.76
cosine	0.995	1.000	1.000	1.005	47.38
parameters	sTh=0.005,mTh=1.6,R=50,err=7,cos=0.99				
model	1.000	1.000	1.000	1.000	22.07
ransac	0.990	1.000	1.000	1.015	23.46
cosine	0.995	1.000	1.000	1.000	20.57

Table 5.4.1: Experimental results on the ZuBuD dataset. Evaluation (accuracy) is performed on the first ($1st$), top 5 images and top 10 images of the ranking lists produced by each method. avg_rank means the average rank of correct hits. $match\_num\_ratio$ is the ratio of number of correspondences of the first correct to that of the first wrong match.

images and top 10 images of the ranking lists produced by each method. avg_rank means the average rank of correct hits. If $avg\_rank = 1.015$, the algorithm finds the correct match at the 1.015-th rank. $match\_num\_ratio$ is the ratio of number of correspondences of the first correct to that of the first wrong match. The higher $match\_num\_ratio$ is, the more robust is the matching algorithm. As we can see from the table, all three methods produce very good results. Among three, *model* gives the best performance with $avg\_rank = 1.000$ and $match\_num\_ratio = 66.27$ which shows the algorithm very robust in distinguishing the correct match from others. Again $sTh = 0.01, mTh = 2.0$ give the better performance across all three methods. Since all three one-step matching algorithms perform well on this dataset, it is not necessary to evaluate our proposed re-ranking and merging algorithms. Fig.5.4.1 shows two detailed figures of match number ratio which confirm our above observations.

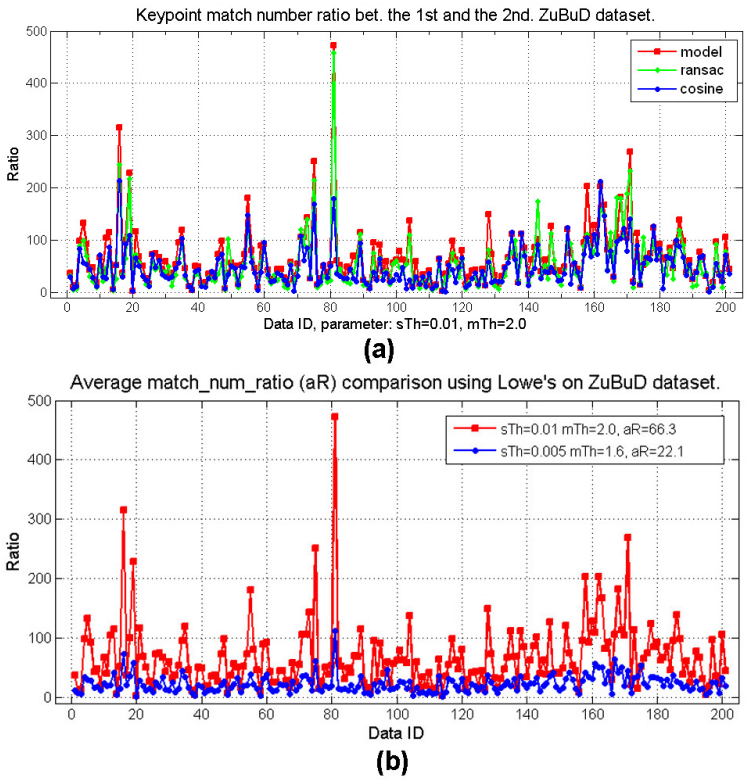

Figure 5.4.1: Comparing match number ratio (between the first correct pair and first wrong pair) on the ZuBuD dataset (the higher the better): (a) comparing three methods; (b) comparing two different parameter sets using the model method.

5.4.2 Results on PHL dataset

Some analysis and discussion have been presented in the previous sections. Here we present the results of re-ranking and merging algorithms on the PHL dataset. Fig.5.4.2 shows comparison among re-ranking algorithms. Fig.5.4.2(a) shows matching by the whole images vs. finding repeating patterns. We can see that finding repeating patterns lead to much performance improvement over the matching-over-the-whole-image scheme. Fig.5.4.2(b) depicts performance of three matching algorithms. It shows that the *ransac* method does the best and this confirms our motivation of using the *ransac* method - buildings generally have one or two dominant planes. In general, the trends in Fig.5.4.2(a) and (b) is going down and it intuitive since the higher rank images more likely contain

$c =$	1	2	3	4	5	6	7	8	9	10
$\alpha = 0.2$	0.79	0.84	0.88	0.90	0.90	0.92	0.93	0.94	0.93	0.94
$\alpha = 0.4$	0.79	0.84	0.88	0.90	0.90	0.92	0.93	0.94	0.93	0.94
$\alpha = 0.8$	0.79	0.84	0.88	0.90	0.90	0.92	0.93	0.94	0.94	0.93

Table 5.4.2: Merging re-ranking results by the *ransac* method using CORI, $\alpha = 0.2, 0.4, 0.8$.

the target building and lead higher re-ranking performance.

Table.5.4.2 shows the results of merging re-ranking results by the Ransac method using CORI. c is the cutoff parameter. As we can see, the greater c is higher performance the algorithm obtain. But performance generally converges around $c = 8$. Another observation we can see from the table is that values of α do not change the trend of performance. Thus we choose $\alpha = 0.4$.

Fig.5.4.3 shows merging re-ranking results. Fig.5.4.3(a) depicts the comparison between the count-based and the CORI-based merging methods. The count-based method generates the final ranking list by sorting the accumulated numbers of correspondences of re-ranking trials. As we can see from the figure, the CORI-based method does much better the count-based method. This endorses our selection of CORI as the merging scheme in our framework. Fig.5.4.3(b) shows

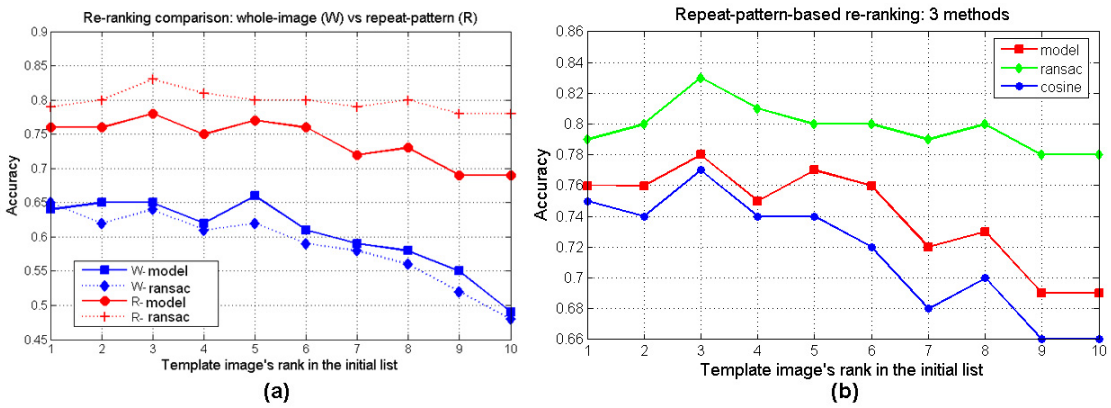

Figure 5.4.2: Re-ranking comparison: (a) matching by the whole images vs. finding repeating patterns; (b) comparing 3 methods by repeat-pattern-based re-ranking.

three algorithms' behavior of merging re-ranking results. Again, we can see the *ransac* method performs better than the other two algorithms. In addition, performance usually converges at $c = 8$ with accuracy equal to 0.94. The high accuracy shows that our proposed method is sufficient for recognizing the target landmark buildings.

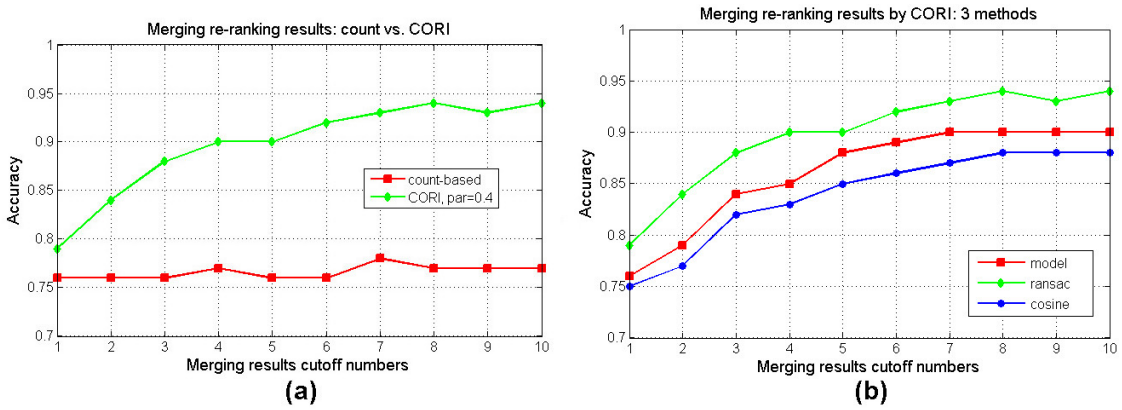

Figure 5.4.3: Merging re-ranking results: (a) count-based vs. CORI; (b) merging re-ranking results of 3 methods by CORI.

Fig.5.4.4 shows the summary of recognition errors the *ransac* method (where there are more than one falsely recognized image) on the PHL dataset. The *ransac* method performs better than the other methods on this dataset. As we can see, among the shown cases, there are at most two falsely recognized images among ten predictions. Fig.5.4.4(b) shows correctly recognized images with $rank = 1$. Although the landmarks appear dramatically different from the ones in model images due to cropping, distortion, view point change, weather change, artifacts, lighting change, our proposed recognition frame still correctly recognizes the target building. Another observation is that falsely recognized images only get low ranks such as $rank = 9$ or $rank = 10$ which shows the robustness of our proposed framework. Fig.5.4.5 shows some example of challenging cases where even humans can hardly recognize the landmark buildings but our methods correctly recognize: (a) landmark buildings are occluded by trees or other objects; (b) building appearance is changed due to renovation.

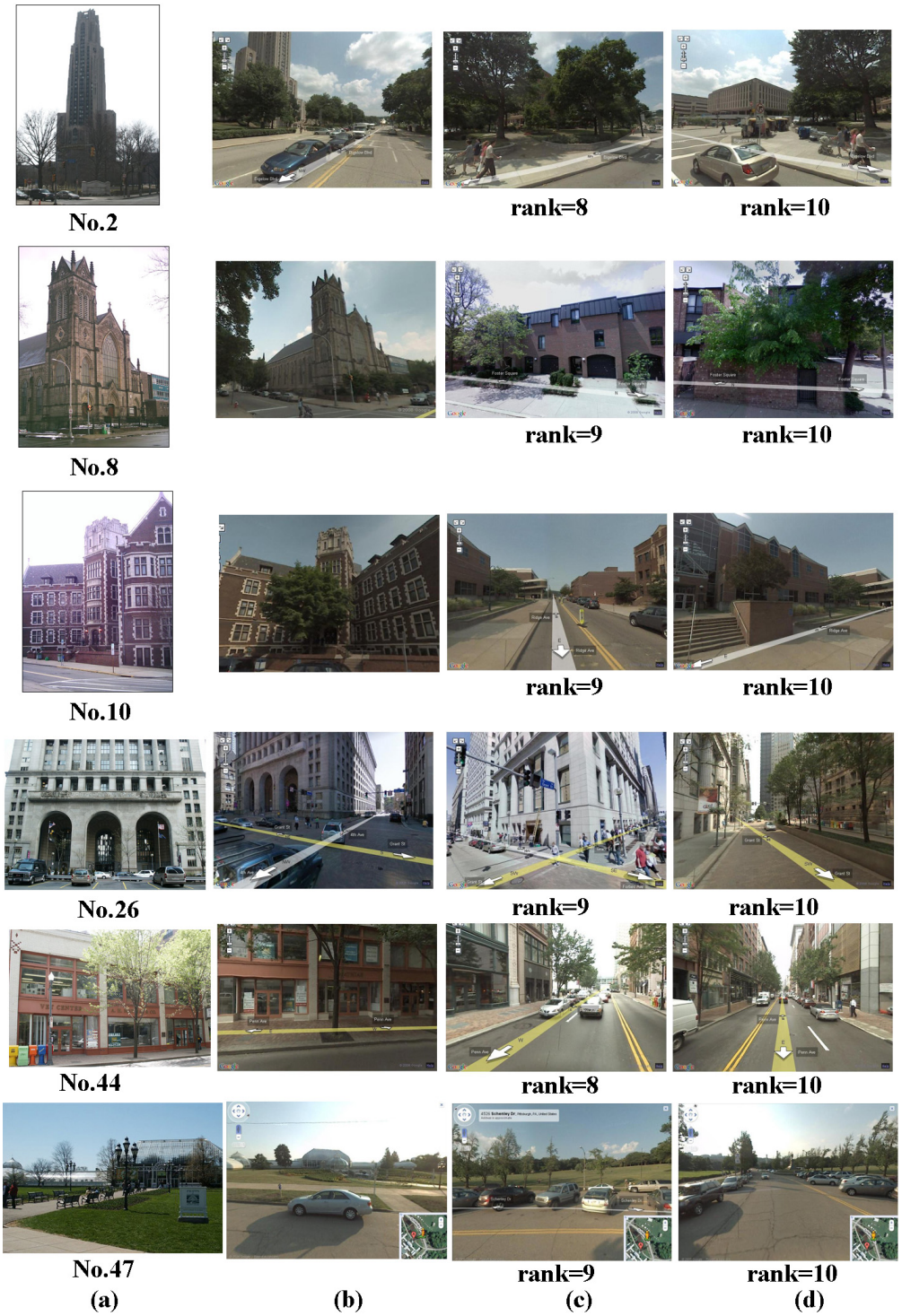

Figure 5.4.4: Errors of the *Ransac* method on PHL: (a) landmark images with their numbers; (b) correctly recognized images with $rank = 1$; (c) and (d) falsely recognized with ranks.

Figure 5.4.5: Some example of challenging cases where our methods correctly recognize landmark buildings: (a) landmark buildings are occluded by trees or other objects (top row: testing data; the 2nd row: training data); (b) building appearance is changed due to renovation (top row: testing data; the 2nd row: training data).

Chapter 6

Human Vehicle Interface

In previous chapters we have discussed several computer vision-based multimedia technologies for landmark-based car navigation. In this chapter, we will focus on human vehicle interface and prototyping using a full-windshield head-up display system.

6.1 Introduction

Navigation user interfaces have changed dramatically over the last few years due to the availability of electronic maps and GPS devices. With increasing popularity of GPS hardware and the Internet, travel by driving has become much more convenient in terms of trip planning and navigation. Most drivers rely on map services on the Internet for trip planning, and simple turn-by-turn guidance (turn instruction symbols and voices) for navigation. During driving, a driver has to map abstract driving instructions, e.g., an arrow indicator on a map, to real world coordinates, which adds extra cognitive load to the cognitively intensive driving task. Therefore, new navigation technologies are not necessarily effective.

Driving is a focus-attention multi-task process. The driver needs to distribute attention among

different aspects of the process. First of all, the driver needs to pay attention to issues directly related to driving, including the surrounding traffic, dashboard displays, and other influx of information on the road such as traffic lights and road signs. In addition, the driver may choose to talk to the passenger(s), listen to the radio, and talk on the cell phone. The limiting factor for information flow during driving is the driver's cognitive load. Although there are many ways to potentially reduce the cognitive load of a driver, in this chapter we focus on the human vehicle interface for landmark based navigation.

There has been much research and applications on in-vehicle navigation. Early research focused on human factor tests in navigation displays [47]. Green et al. surveyed human factors for driver information systems [46]. They described objectives, principles and guidelines for the design of in-vehicle devices. Dale et al. studied the problem of generating natural route descriptions for navigational assistance [29]. Nevertheless, landmarks have great potential in both route sharing and driving contexts. Burnett et al. studied which landmarks are valued for driving navigation and their salient characteristics [13]. They found the significance of so called 'road furniture' landmarks, such as traffic lights and petrol stations. Landmarks can also support pedestrian navigation [44]. Combining visual cues with voice instructions was discussed in [12]. Lee et al. presents a contextually optimized map system for in-vehicle navigation [66].

6.2 Prototype Development on Computer Display

Most current in-vehicle navigation systems give driving instructions in the form of map and synthesized voice. However, the lack-visual-cue instruction is indirect and sometimes attention demanding to the driver. In particular, it can cause problems when driving to an unfamiliar area or the driver's attention gets disturbed by other tasks, like receiving a phone call and talking. Sometimes drivers just do not have enough time to map from the driving instructions to real world situations. Then they get lost. Since the voice and turn-by-turn instructions are not good enough and even not helpful in some cases, can we add additional visual cues for in-vehicle navigation?

We develop a multimedia system to demonstrate landmark based navigation technologies on computer display. We use captured video to simulate the driving course. The system automatically gives driving instructions in three ways: a) overlaying navigational arrows on the top of the road, b) highlighting relevant landmarks for key maneuvers and c) playing synchronized voice directions.

Studies have shown that that users' performance can be significantly improved using a combination of directional arrows and photographs in navigation tasks, and also that this combination was highly preferred by users [24]. Following the same rational, we introduce a concept demo of *landmark-based navigation* which hope to reduce the driver's cognitive load and further enhance driving safety. Our system combines visual cues with voice instructions. It overlays navigational arrows on the road and gives landmark images at each turning intersection. In our system we use driving videos to simulate the real scenes in front of the car. The driver can still navigate the route by looking at the display even after missing the voice instructions.

Fig.6.2.1 shows the user interface for a concept demonstration of landmark-based navigation. The system gives voice instructions based on landmark information. For example, a system such as in Fig.6.2.1 tells drivers "go straight", "stay right", "stay right and continue on route 51", "go straight and pass the library", "turn left and pass the gymnasium", "turn left after passing the BP gas station sign", "arriving Donner house on your left", etc. Meanwhile, the electronic map shows current position of the car and positions of referred landmarks such as buildings, store signs and road signs. Our landmark-based navigation system with GPS aims to offer technologies necessary for the John visiting Susan scenario that is introduced in the Chapter 1.

The system knows the vehicle's current location from GPS and the distance, K, from the next turning intersection from the map system. If $K \geq \lambda$, the system shows flashing road boundaries to indicate going straight as shown in Fig.6.2.1; otherwise, the system shows the turning arrow (left or right) based on the direction. $\lambda = 30m$ for out-door and $\lambda = 6m$ for in-door. The task involves two steps: a) measuring the distances from the vehicle to its surrounding objects; and b) overlaying on the road in video the turn arrow with proper perspective. We adopt the stereo vision technique for measuring the distance because of its cost advantages over other devices, like radar

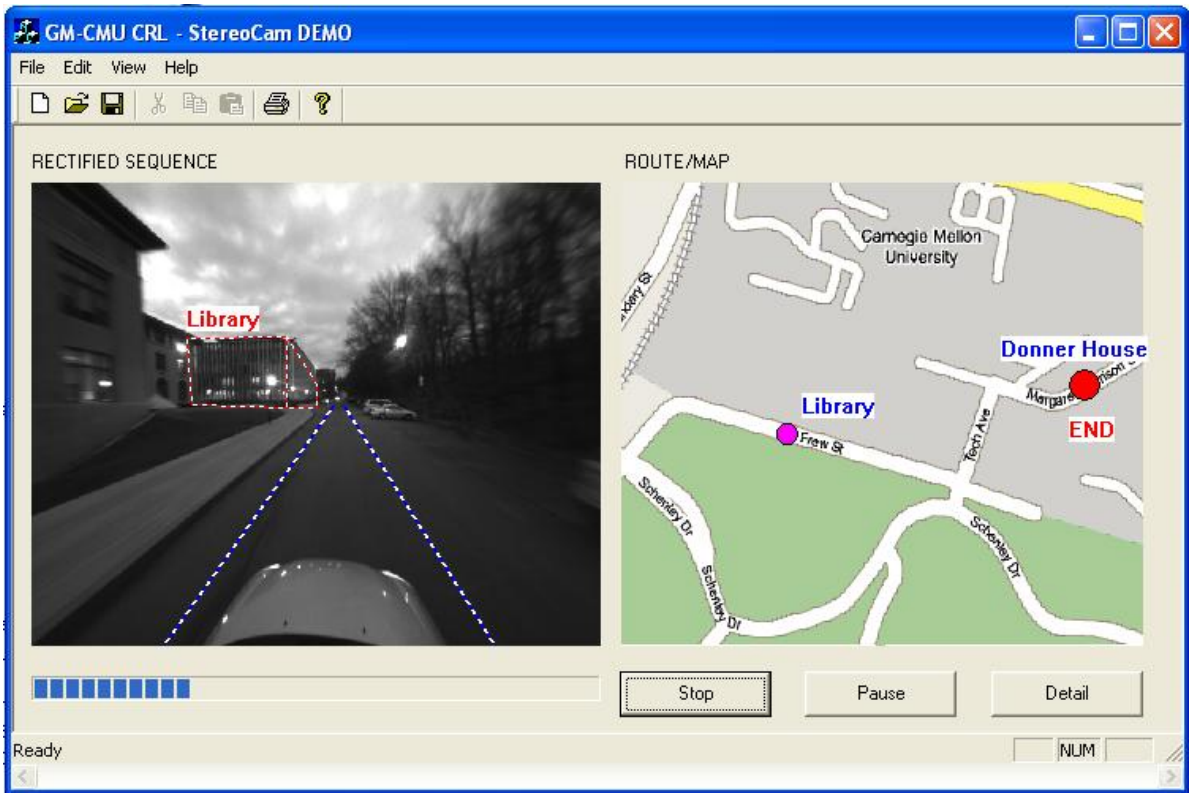

Figure 6.2.1: User interface for *on-road navigation*. The left window shows overlaying the navigational instructions (e.g., flashing lines) and highlighting the landmark (e.g., the library). The right window shows the dynamic map.

and lidar. We use the Point Grey's Bumblebee, a two-lens stereo camera, in this project. Stereo processing is a three-part procedure. First, it establishes correspondence between image features in two views of the scene. Second, it calculates the relative displacement between feature coordinates in each image. The obtained displacements for every pixel constitute the disparity image. Third, it determines the 3D location of each feature point relative to the camera by knowing the camera geometry. Since we only need the depth for each point,x_i, and it can be computed as, $Z = \frac{Bf}{d}$, where B is the distance between the optical centers of two stereo cameras, and called the baseline; f is the focal length; d is the disparity at x_i and Z is the distance (depth) between x_i to the stereo camera center. $B = 12cm$ and $f = 2mm$.

Once we know where to mark, the next step is how to mark the navigational arrow *perspectively*

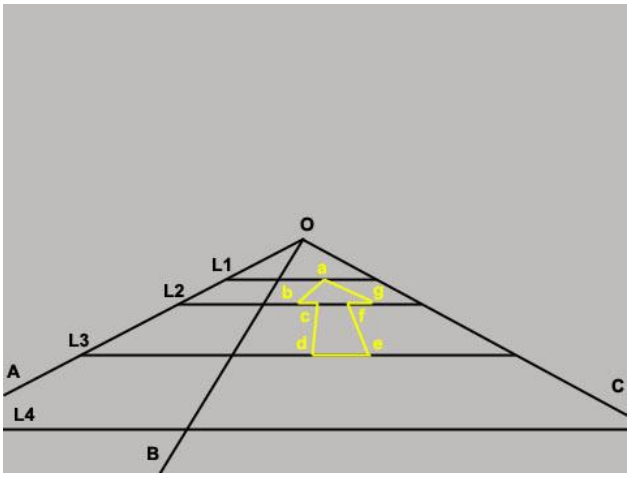

Figure 6.2.2: Geometry template to depict the perspective arrow.

on the road. Take the go-ahead case as an example. Fig.6.2.2 illustrates the geometry template we design to draw the perspective go-ahead arrow. OA and OC represent the left and right road borders, O is the vanishing point and OB shows the dividing line. $L1, L2, L3$ show the three horizontal lines controlling the vertical positions of the overlay arrow, and $L4$ represents the upper boundary of the in-vehicle portion. Furthermore, $a - g$ specify the seven key points of the arrow sign. We estimate the coordinates of O, A, B, C from the data empirically by assuming the known car and camera geometry. Other vanishing point and lane border detection algorithms can be applied if high accuracy is desired. Coordinates of $a - g$ are further determined according to the template based on the estimations of O, A, B, C. The same technique also applies to depict turning arrows and we cannot elaborate it here due to limited space.

Performing a rigorous and comprehensive evaluation of the system of this kind is an extremely costly work. Following the evaluation methodology of Dale et al. in [29], we have performed a small-scale expert evaluation in a task-based context. Our user study group includes five users, one vehicle manufactory designer, two experienced drivers, and another two junior drivers. The goal of the evaluation is to obtain user feedback on the functions and user interfaces of the system. The experiment is carefully designed to minimize the many factors which can influence the feedback of the users. For the concept demo of *landmark-based navigation*, we test the system in both

indoor and outdoor contexts. For the indoor case, we run the system on a laptop, put it on a trolley and move in hallways. For outdoor case, we choose LCD as the in-vehicle display due to the limited amount of vehicles which include reconfigurable FWDs. Originally, we design the icon-based navigational arrows and later refine the system with the current perspective arrows. Fig.6.2.3 shows four examples. All five users prefer the perspective arrows over icon ones. For the *route sharing*, we compare user feedback on two types of input modality, i.e., drawing a route on the map or selecting way points. Among five users, three prefer selecting way points while other two choose drawing the route.

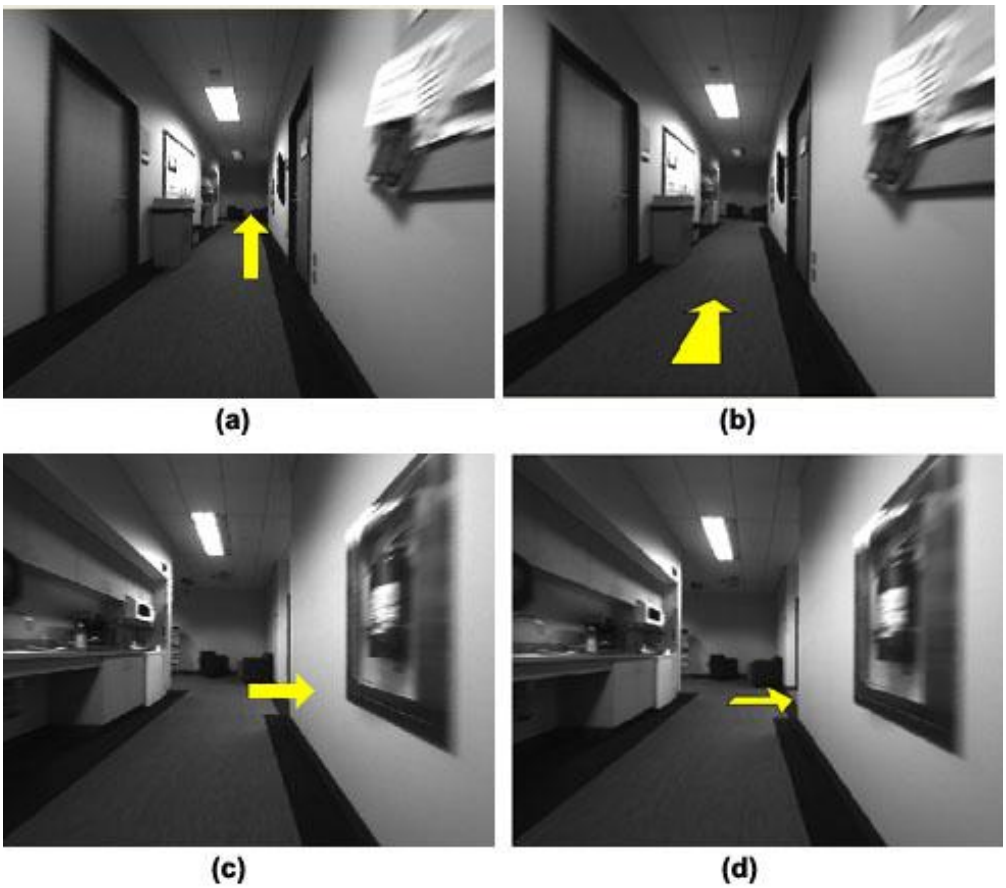

Figure 6.2.3: Navigational arrows for *on-road navigation* in indoor context. (a)&(c): icon based go-ahead and turn-right arrows; (b)&(d): perspective go-ahead and turn right arrows.

6.3 Demonstration on a Full-Windshield Display System

A head-up display, or HUD, is any transparent display that presents data without obstructing the user's view [1]. Although they were initially developed for military aviation, HUDs are now used in commercial aircraft, automobiles, and other applications. Full-Windshield Display (FWD) is a particular term for automobile application.

Figure 6.3.1: GMC Acadia Full-Windshield Display showing speed and other information.

FWDs have been available as factory equipment in some American and Japanese cars (primarily to present speed information and warnings) since the 1988 model year [14]. The advantages and disadvantages of FWD have been long researched since then. FWD may provide benefits beyond enhanced navigation performance. By placing navigation information in the driver's field of view, the amount of time the driver is looking away from the road should be reduced. Logically, this should reduce accidents [107].

[1] http://en.wikipedia.org/wiki/Head-up_display

A typical FWD Development Kit includes a windshield, a laser scanner and a controller. OpenVG based Scan Ray Vector Graphics library is usually developed so that the vector graphics path can be rendered on the windshield. Fig.6.3.1 depicts a Acadia FWD displaying unobtrusively projects speed and other information (circled in picture for clarity) right onto the windshield. The purpose of this part of dissertation is to demonstrate our approach for street landmark recognition on the FWD provided by our project sponsor General Motors Company [117]. Here is some information about the FWD which we have used. It is a 2-color full-windshield display system. It has high speed laser scanner, 30 kilo-points per second. 16 bit resolution (X & Y scan) , 256 levels of laser intensity. [2]

6.3.1 Motivation and Overview

Driving assistant systems are popular applications of multimedia technologies. A combination of GPS (global positioning system) and electronic maps has led to revolutionary changes in car navigation systems. These systems provide drivers with an efficient route planning tool through a map database and allow convenient route guidance by GPS satellite signals. Users of such navigation systems can take advantage of such digital navigation services not only in their daily lifes but also in unfamiliar areas or in regions with complex road-layouts and intersections.

However, state-of-the-art navigation systems have not yet reached their full potential and their use is often not intuitive and helpful to users with different abilities. There are two main kinds of navigation interfaces in existing systems, i.e. voice commands and abstract map images on a display, either integrated in the dashboard or being portable. Both interfaces have proven useful and effective but not best. For example, current navigation systems can increase a driver's cognitive load because the driver has to map the information provided by the navigation system to the real environment outside the windshield. Another distracting effect of current navigation systems is that the driver has to move his/her attention away from the road to perceive navigation information.

[2] http://www.superimaging.com

The head-up display (HUD) is a type of display that presents data without blocking the user's view. The technique was pioneered for military aviation and now used in commercial aviation, motor vehicle and other cases. Since 1988, General Motors has popularized HUDs on the Olds Cutlass, Pontiac Grand Prix, Bonneville, Buick LeSabre, Park Avenue and Rendezvous. In 1999, Automotive HUD technology made a big quality leap with the Chevrolet Corvette. The new Corvette, which uses a HUD to display vehicle speed, engine RPM, navigation and more, has proven the HUD to be one most popular option. As of 2006, BMW featured the HUD as an option on their 5 and 7 series vehicles, with more HUDs anticipated from other European and Japanese OEMs. Despite the limited resolution and current generation HUDs' sizes, they provide information such as speed, turn by turn navigation and warnings via a virtual image projected onto the windshield by mirrors within the driver's normal field of vision. By adjusting the number of mirrors, display color and illumination elements, installation space requirements and costs can be adapted to respective vehicle models.

In this research, we are developing technologies for next generation navigation systems using a full windshield head-up display (FWD) system manufactured by SuperImaging, Inc. [52]. FWD can project information everywhere on the windshield while HUD projects information in a limited windshield region. In driving applications, a FWD system provides a flexible way for delivering information. A FWD can highlight important road signs such as speed limit signs. It can be used to enhance the vision of the driver in dangerous situations. Infrared cameras or laser scanner can gather information from the car surroundings which is not easily visible to the driver in poor lighting conditions. Deer, pedestrians or bicyclists can be detected and highlighted. So common dangerous situations can be avoided by showing threats on the FWD. In situations such as bad weathers, FWD can also be used to highlight street boundaries which can especially help old people with weak vision capabilities.

Directly projecting text, images and graphics on the FWD's windshield causes distortion because of the windshield's non-planer surface. We assume that the viewing angle of the driver is fixed. Conceptually, our method can be extended to deal with changing viewing angles. Distortion

correction is the prerequisite for a FWD. We have proposed to use a computer vision system to automatically correct distortions at different locations of the windshield. Our system consists of a FWD, a camera, and an LCD projector. In the calibration phase, the camera will capture the patterns generated by the projector and the system will automatically model the distortions. In the working phase, the LCD video projector projects a video sequence onto a wall to simulate driving situation. We demonstrate the feasibility of the concept of landmark-based navigation using the road sign detection results from our previously developed road sign detection system [120, 119].

There has been much research directed to driving assistance systems, car navigation and HUD. Liu et al. [74] have compared driving performance and psychological workload of drivers using a head-down display and an HUD. The results showed that the response time to urgent events is faster produces less mental stress with an HUD. Wu et al. [124] presented a multimedia system for route sharing among users and navigation by overlaying turning arrows and highlighting landmarks in videos. Chu et al. [25] have recently proposed to use a full-windshield HUD to show various information to a driver. However, they have not presented a working system in the paper. Sato et al. [95] constructed a similar setup to ours with a full-windshield HUD but use a different projection technology. Our system includes a new FWD model, a camera, a LCD projector and a road sign detector.

6.3.2 Automatic FWD Windshield Distortion Correction and Prototyping

Our driving assistance system for highlighting landmarks on a FWD can be used in real-time applications and is robust to different lighting and road conditions. The main components of the system are a laptop, a video camera, a laser projector, a dedicated controller, and a MediaGlass™ windshield. The glass is covered with transparent phosphors to reflect blue or red light emitted by the laser. The system provides easier navigation with less distraction for safer driving. Fig.6.3.2 shows the setup of our experimental system. We tested our system inside a laboratory and simulated different test driving scenarios with an LCD projector.

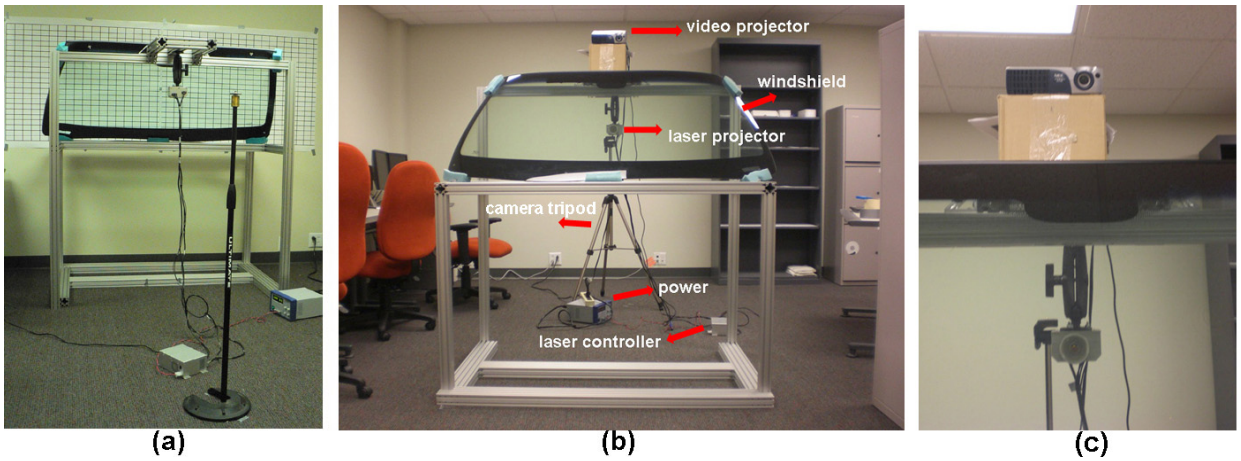

Figure 6.3.2: The FWD setup (a) a back-view of the set-up; (b) an overview with annotation and (c) a front-view of the video projector and the FWD laser projector.

Before using the FWD we need solve one key problem - correcting the to-project data to avoid windshield distortion. All images projected to the windshield are distorted. We solve this problem, by applying two pre-warp functions $f(x, y)$ and $g(x, y)$ to all images before projecting them to the windshield. Assume projected laser points's coordinates in the camera view as $(x_i, y_i) : i = 1, ..., N$ and in the laser projector plane as $(u_i, v_i) : i = 1, ..., N$. We want to find the functions $f(x, y)$ and $g(x, y)$ such that $f(x_i, y_i) \approx u_i,\ g(x_i, y_i) \approx v_i,\ i = 1, ..., N$. The process of generating the pre-warp information for the distortion correction is as follows: the first step is to project a pattern on the windshield and capture the result. This pattern is used to compute the correct pre-warped points. Any arbitrary point can be correctly projected on the windshield with interpolation based on the pre-warped points. Creating the pre-warping information poses different challenges, which are influenced by different aspects. The lighting conditions of the experimental setup or partly occlusion of the windshield play a role here. The non-planar surface correction is sensitive to movements of the FWD projector and the windshield which are fixed in our setups and expected commercial uses.

The process to correct distortion on a windshield is as follows. The first step for constructing the pre-warp mapping is measuring the windshield. Therefore we project a special pattern on the

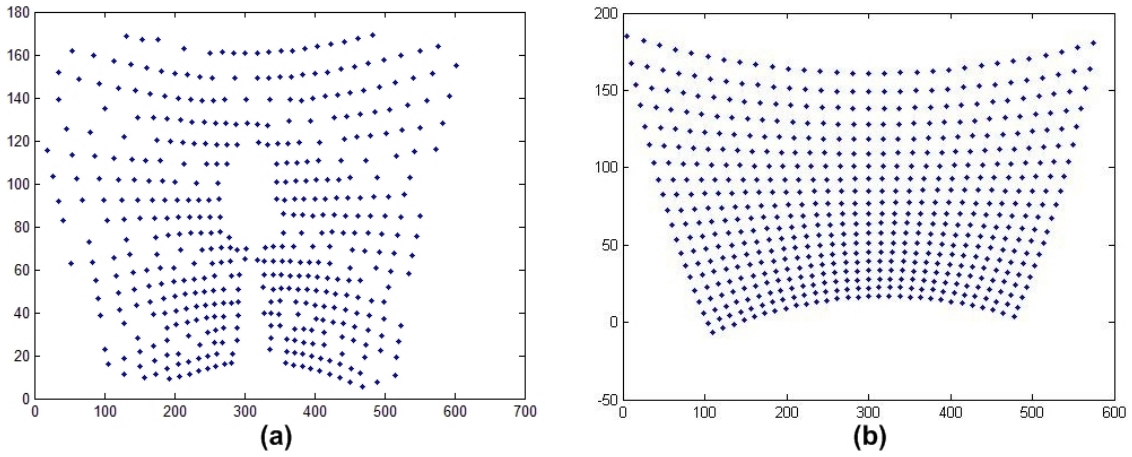

Figure 6.3.3: Distortion correction for FWD. (a) Initially corrected projected laser points from the video. (b) Corrected projection grid. Upside down axis due to laser projection scheme. Note that both images are scaled along the x-axis.

windshield and record it with a camera. The next step is to compute a pre-warp function by using the correspondence from the projected points to the recorded points. With this function we can adjust images so they do not look distorted to the driver. This image will then be projected to the windshield. The calibration process for the pre-warp function must only be done once. After registering the surface of the windshield one can use the pre-warp function independently of the position of the driver just by small adjustments.

For the first step we project a pattern to the windshield. Our use pattern is a grid of blue points which forms the shape of a rectangle. We choose a grid of 30 points per row and 20 points per column. All points are projected for 250 ms. Shorter time intervals lead to disturbing artifact from the laser moving between the points. A pattern of grid points is chosen, because other patterns commonly used by a calibration process like check boards, cannot be projected by FWD because FWD cannot project filled areas properly. The projected pattern is recorded to video by an ordinary digital camera on a tripod. This camera represents the view of the driver.

Before extracting the points from the recorded video we have to compensate for different lighting conditions, because the recognition of the laser grid points is sensitive to bright spots. All

possibly disturbing light sources should be removed, for example reflections on the windshield frame. Therefore we mask the image from which we extract the coordinates and use solely the part which contains the windshield. Additionally we limit the color information in the extracted images to the blue color channel. The reasoning behind is that the calibration process only uses blue laser rays. A different part of the complexity of the extraction process is due to small parts of the windshield which are occluded from the laser projector. Additionally are several other points which cannot be extracted from gathered data. The projected pattern is larger than the windshield and therefore are some points not projected to the reflecting windshield. We must compensate for them as we need to find the relations between all projected and captured points. Another problem that occurs is that some extracted frames from the video contain artifacts from the previous projected grid points. Our approach to correct bad or missing points is creating an interpolating spline curves for each row of points. These splines compensate for missing points or smooth out existing points. We used piecewise polynomial functions of degree two because they fitted the best to the curved windshield surface.

Based on the extracted points we create the pre-warped points. We are using the correspondence of the coordinates of the distorted points extracted from the camera (x, y) to the coordinates used to control the laser (u, v) to generate a mapping. Therefore we compute a new orthogonal grid of points in the coordinates of the camera (x, y). For all these new points we use interpolation to compute the warped points in the coordinate system of the laser (u, v). We use a 2D linear hyper-surface fitting function for the interpolation which computes the pre-warped points from the new grid points, the extracted laser coordinates and the corresponding coordinates (u, v). The new interpolated points are used as input for pre-warp functions to project images on the FWD. The pre-warped points can be used to project a grid of undistorted points to the windshield. To reach the goal of projecting any image to an arbitrary position we use interpolation to correct points between the pre-warped grid points. With the pre-warp function FWD can display arbitrary images on arbitrary positions. Fig.6.3.4a shows an image before pre-warping is applied. Fig.6.3.4b shows the pre-warped image. One can see that not only is the image distorted, but the size changed significantly.

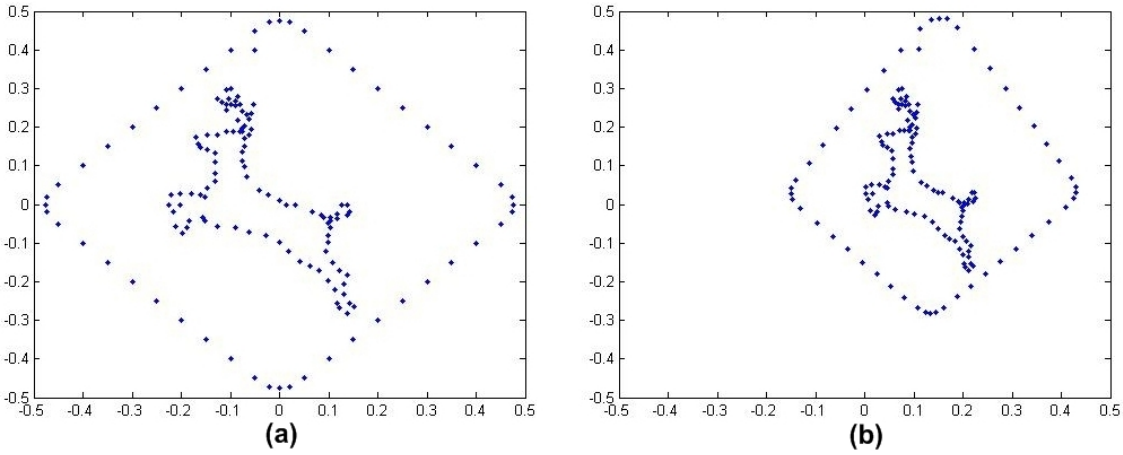

Figure 6.3.4: (a) shows a typical icon used as input for the FWD. (b) shows the same icon after the pre-warp function was applied.

6.3.3 Results and Demos

Projecting a Moving Pattern

Fig.6.3.5 shows three testing examples of different icons that are projected at different locations without any distortion in the camera view. Using a pre-warp function together with a look-up table of pre-warped points allows for fast correction in real-time projections. Due to the use of interpolation between the pre-warped points does the projection program need only little memory. This is especially valuable in cars with limited computer systems. As the installation of the MediaGlass™ and the laser projector was not in a real car, we used video projection of different self-recorded drives for the simulation. Qualitative evaluation of the resulting projection on the windshield shows that the approach was successful. The pre-warp function completely compensates distortion of the windshield. The FWD projection on the windshield is bright and clearly visible. Thus our windshield distortion correction method is beneficial in other applications.

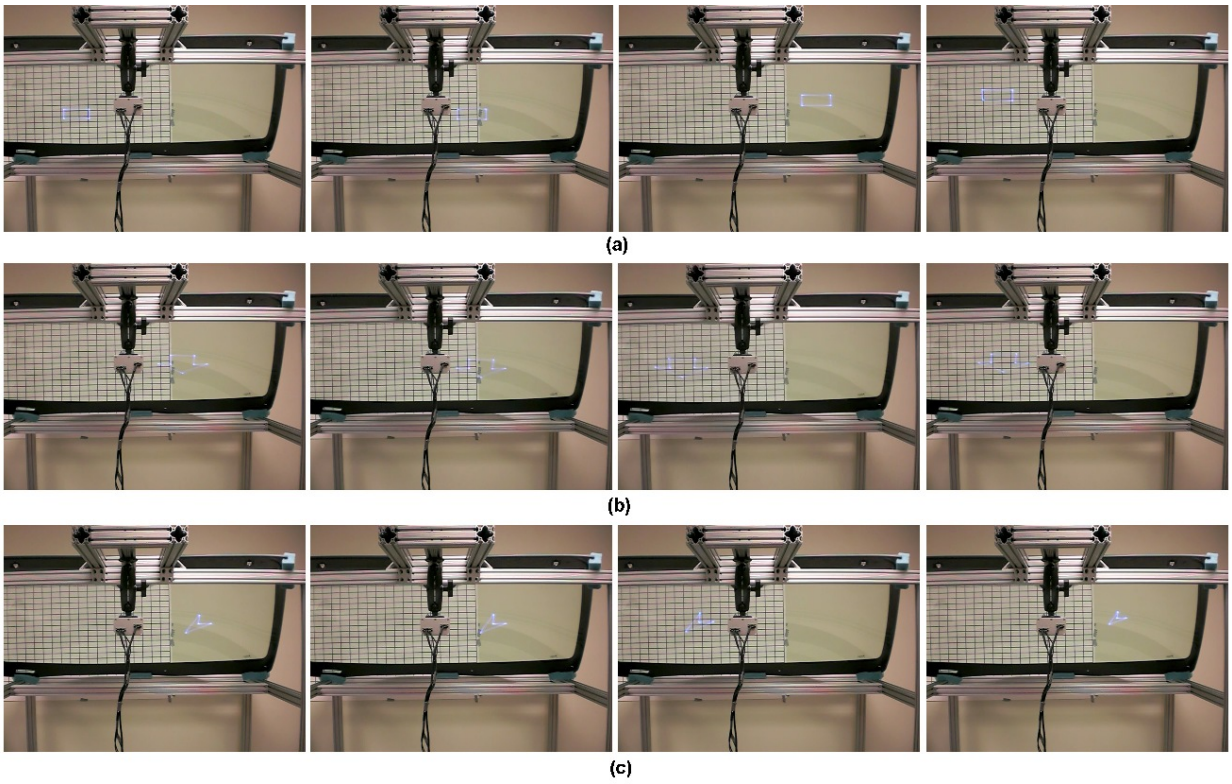

Figure 6.3.5: Distortion correction test examples: a rectangle in (a), an arrow in (b) and another arrow in (c). Regions in the blue boundary, which are projected by the FWD laser projector at different windshield locations, show no distortion.

Highlighting Moving Road Signs in the Video

Another demo with highlighted road signs in a video on the FWD also shows promising results. The demo use outputs from our previously developed road sign detector [120] that gives locations of detected signs in the video. The detector naturally employs a divide-and-conquer strategy to decompose the original task into the two subtasks, namely: 1) localizing road signs and 2) detecting text. Because of government requirements on the design and placement of road signs, our observation shows that 1) text on road signs has higher contrast compared to most sign background colors; 2) text on the same road sign always has similar foreground and background patterns; 3) most road signs exist on vertical planes; and 4) there are only a limited number of colors used as

background colors. Our system has four steps: 1) Discriminative points detection and clustering that detects discriminative feature points in every video frame and partition them into clusters. 2) Road sign localization that selects candidate road sign regions corresponding to clusters of feature points using a vertical plane criterion. 3) Text detection that detects text on candidate road sign areas and track them. 4) Text extraction and recognition that extract text in candidate sign plane for recognition given a satisfactory size. The sign is highlighted by FWD. Fig.6.3.6 shows two demo sequences in which the highlighted (by FWD) road sign is presented in the simulated driver view. Highlighting landmarks like roads signs can help the driver while navigating. It is not clear if distraction of FWD projection revokes the advantages of the provided navigation info. Our results shows FWD can give the driver helpful guidance with help of other vision sensors and systems and offer a good basis for future research.

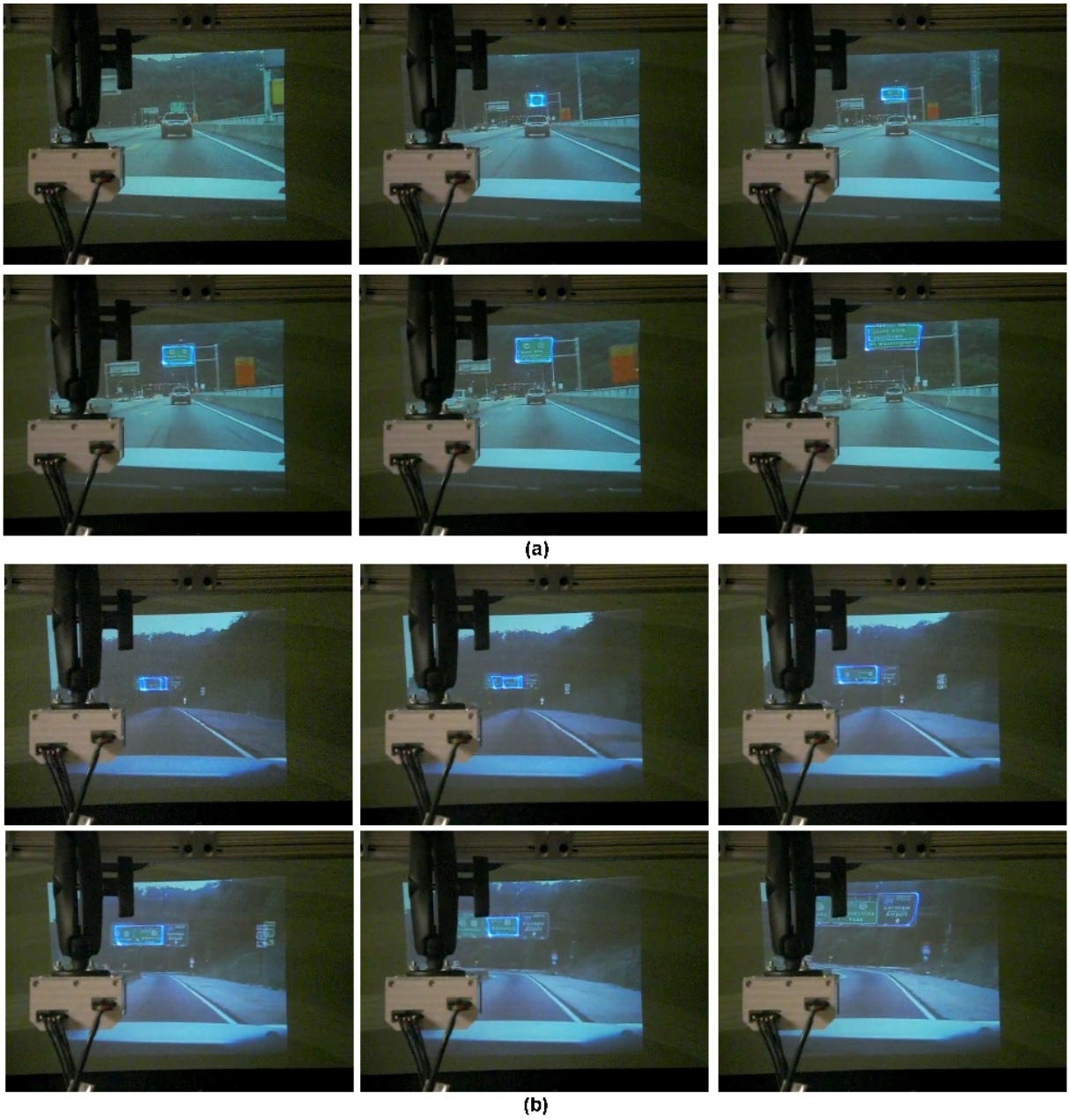

Figure 6.3.6: A road sign is highlighted by FWD projection (in blue) over time that overlays the sign on the wall projected by a video projector. Two demo sequences are in (a) and (b).

Chapter 7

Conclusions and Future Work

There are several contributions that have been made in the course of my exploration of this dissertation. Some are summarized in the following.

7.1 Summary of Contributions

Labeling Landmarks and Objects in Image

In the chapter of *Semi-automatically Labeling Landmarks and Objects in Images*, we have presented a family of semi-automatic object labeling methods based on Zhu's SSL method [130] in a semi-supervised learning framework. Given an image with a small-size ROI, our methods can extract the contour of the object and also the same or similar objects at other locations in the image [121]. In particular, we have proposed SmartLabel-2 to enhance SmartLabel by overcoming three of its limitations [123]. Our experiments on various object classes have demonstrated that SmartLabel-2 not only outperforms SmartLabel but also achieves close-to-fine extraction of object contours in many classes. One future direction is to reinvent SmartLabel-2 for labeling objects in videos.

One question rises when using over-segmentation in SmartLabel-2 - *why not use over-segmentation patches (superpixels) directly as the labeling units?* We offer three thoughts for discussion: 1) *Computation*. Generating 100 superpixels on a 512×512 image can take $>$10 minutes but *quadtree* normally takes $<$1 second, both using Matlab. We do not seek real time computing in this work. 2) *Feature representation (FR)*. FR is not well addressed for superpixels yet. Many current FR methods are for representing features on regular regions not arbitrary ones. Apart from the FR issue, normalization and similarity measurement can be problematic too. 3) *Effectiveness*. Applying quadtree partitioning and refinement by superpixels has led to promising results in our experiments. We would like to leave this interesting direction for future audience.

Detection of Text on Road Signs from Video

Large amounts of information are embedded in natural scenes. Road signs are good examples of objects in natural environments that have rich information content. In the chapter of *Detection of Text on Road Signs from Video*, we have presented a new framework for incrementally detecting text on road signs from video. The proposed framework efficiently embeds road sign plane localization and text detection mechanisms with feature-based tracking into an incremental detection framework. The framework can significantly improve the robustness and efficiency of text detection. The new framework has also provides a novel way to detect road sign text from video by integrating image features and the vertical plane properties of road signs. Experimental results have demonstrated the feasibility of the new incremental detection framework under real-world settings.

Recognition of Other Signs

In the chapter of *Recognition of Other Signs*, we have proposed a novel approach for recognizing other signs such as store signs from a sequence of images. Objects are basic units for multimedia content analysis. Much research has focused on exploring similarity among objects and contexts

for content analysis. In this chapter, we have proposed a novel approach for multimedia content analysis based on objects' unique characteristics which we call object fingerprints. In particular, we have demonstrated its performance on the problem of street landmark localization. We introduce the concept of object fingerprints to represent salient appearances of the object as well as the geometric relationships among local features. Another contribution of this work is a way to deal with object recognition and localization with very limited training data. We have focused on single landmark recognition from images. However, our approach can be well generalized to tackle other kinds of content analysis tasks. Our approach employs some state-of-the-art techniques for feature extraction, and has achieved better results than existing methods. In addition, occlusion can be handled by matching object fingerprints at various locations of the object as long as at least one object fingerprint is matched.

Recognition of Landmark Buildings

In the chapter of *Recognition of Landmark Buildings*, we have proposed a three-stage approach to recognize the landmark buildings for helping localize the vehicle's current position while driving. We have collected images from Internet resources and built the Pittsburgh Historic Landmark (PHL) dataset. To the best of our knowledge, the PHL dataset is one of the first driving-based building image datasets. Evaluation done on the PHL dataset and the ZuBuD dataset (a public image dataset of Zurich buildings) have both shown the robustness and effectiveness of our proposed approach. We believe the proposed approach can be surely applied to larger datasets and generalized to other recognition tasks in the driving application domain.

Human Vehicle Interface

In the chapter of *Human Vehicle Interface*, we have showed that highlighting landmarks on a Full-Windshield Head-Up Display system (FWD) is possible and may help the driver to navigate. In order of calibrate the FWD, we have proposed projecting a point grid using the FWD, recording

the pattern and creating a interpolation-based function to correct for distortion due to non-planar windshield surface. Our experimental setup was used to project route guidance information on a prototype FWD. Highlighting landmarks such as road signs is demonstrated as support for navigation. Although the current cost for the prototype is relatively expensive (more than $1000 for a FWD), the prospect of proposed system is promising when the cost decreases in the future. In our experiments we assume that the viewing angle of the driver is fixed (driver position and head orientation), which is essential for our proposed automated distortion correction method to work. The assumption is limited and always violated in the real driving situation. Detection of road signs could be extended to detect pedestrians, bicyclists or crossing animals on the street, so the driver could be warned in advance of dangerous situations.

7.2 Future Work

- For labeling landmarks and objects in images, future directions might include:
  - Directly use over-segmentation patches (superpixels) as the labeling units.
  - Reinvent the SmartLabel-2 algorithm for labeling objects in videos.
- For detection of text on road signs from video, future directions might include:
  - Generalize the proposed framework to detect scene text on other surfaces.
  - Apply the proposed road sign detection algorithm to classify and analyze other objects in images and videos.
- For recognition of street landmarks, future directions might include:
  - Study the importance of various features in selected object fingerprints for different street landmarks.

- Combine *context* and *content* for more robust object recognition and localization. Some pioneer work in this direction has been published, using image context [110], geometric scene context [50] and image location tags [59].

- For recognition of landmark buildings, future directions might include:

  - Evaluate our proposed algorithm on the large-scale datasets by considering computation and accuracy at the same time.
  - Study the problem of recognizing general buildings in both views of pedestrians and moving vehicles.

- For human vehicle interface, future directions might include:

  - Introduce a head-tracking system with the goal that the driver sees the correct projection on the windshield matching the environment independent of his/her perspective.
  - Analyze the psychological effects of using an FWD. Further research on this topic is necessary to fully understand psychological workload of the driver before installing FWDs in mass produced cars. The possible future design and implementation based on FWD would be guided by the analysis of human psychological workload in a human-centered design manner.

7.3 Conclusion

In the last decade, car navigation systems and tools have evolved considerably as briefly shown in Fig.7.3.1. Many car navigation systems have become available. These systems offer the promise of easily-accessible and friendly multimodal user interface, but the existence of such diverse navigation tools raises the question of *"what is the better way to provide route guidance and navigation information to the drivers?"*

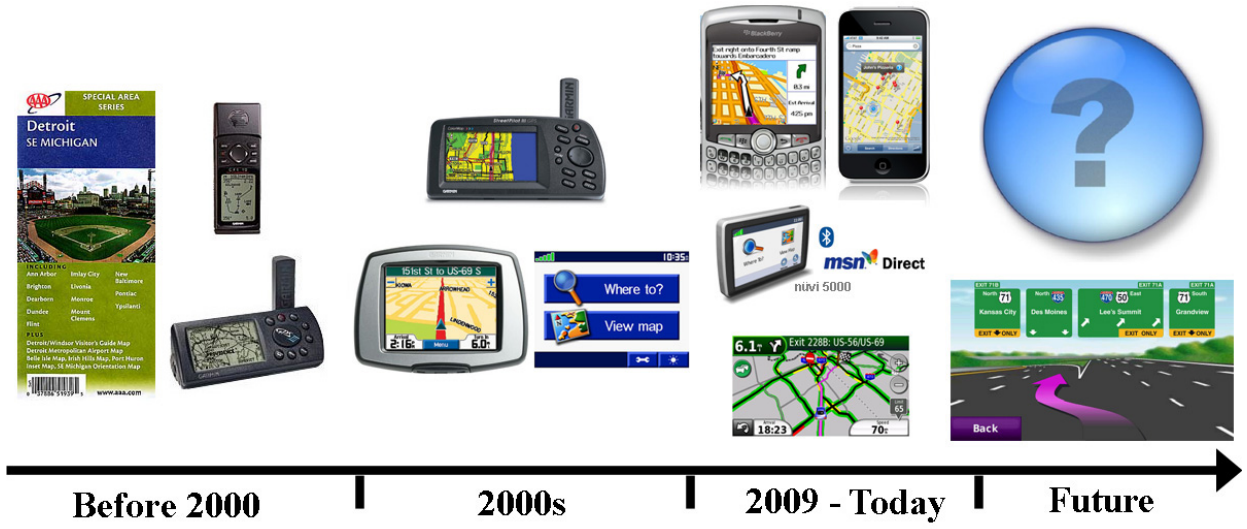

Figure 7.3.1: A brief overview of evolvement of in-car navigation tools. Some images are from www.Garmin.com and rest from the Internet.

Simple portable navigation devices are capturing the world car navigation market, accounting for about 80% of the 45 million devices worldwide. This market is currently dominated by three firms, which hold about 80% of the market among them: TomTom NV of the Netherlands, Garmin International Inc of the US, and MiTAC International Corp of Taiwan [103]. Their massive market shares are mainly made possible by low prices - about $200 per device. This situation is however changing. As the 3G mobile telephony network expands, smart-phones such as BlackBerry, iPhone and other similar mobile terminals are beginning to offer the same - or sometimes better - navigation functions as these simple portable navigation devices. The change in the car navigation market is affecting not only portable devices, but installed models as well. As a result, the whole car navigation systems market is becoming to a tri-polar structure: mobile phones, portable devices, and factory-installed navigation systems. The change allows more companies such as Google and Apple enter and transform the car navigation market.

In this dissertation we offer an array of multimedia technologies that aim to achieve landmark-based navigation in the next generation of vehicle navigation systems. We start by presenting work done in *how to label landmarks in images* that provides labeled data for training detection

or recognition algorithms. We note that our proposed semi-supervised learning-based SmartLabel framework is but one of the possible methods for object labeling, a field that is currently active with several method showing promise.

We then describe an automatic system which detects text on road signs from video input and helps the driver maneuver in traffic. Our research and evaluation suggests that by decomposing the original task into two subtasks via a divide-and-conquer strategy, we are able to naturally incorporate two sub-solutions into an unified framework through a feature-based tracking algorithm. The proposed framework provides a novel way to detect text from video by integrating two-dimensional 2D image features in each video frame with the 3D geometric structure information of objects extracted from video sequence. In addition, we have demonstrated the road sign detection and recognition on a full-windshield head-up display system, which shows promising results and potentials.

In coping with other street landmarks, we have focused on recognition of store signs and landmark buildings. Our research suggests that by mining and recognizing street landmarks' object fingerprints, we are able to bypass some of the more challenging recognition issues that typically hinders object recognition algorithms. By developing a landmark recognition system that combines salient region detection, segmentation, and object fingerprint extraction techniques, we are able to recognize and localize street landmark from images and videos. We have demonstrated improved results by using the object fingerprint-based recognition method. We have also developed a robust re-ranking-based recognition framework which is able to recognize the buildings of interest in diverse driving views. As an artifact of this dissertation, we have created two datasets for testing above recognition algorithms that will be made available as a by-product of this dissertation.

Some of the issues are not within the scope of this dissertation, namely combining all proposed technologies into an unified system including user interface, which certainly have an impact in evaluating the proposed techniques systematically. We have chosen not to perform integration of the proposed technologies due to the workload and also availability of evaluation results of individual approaches. We believe evaluation based on an integrated system is valuable and a plus,

however, current evaluation results are convincing and have shown the benefits and advantages of each individual technology. Therefore, we suggest in the future work that by integrating and merging proposed technologies in this dissertation, it is possible to find a set of research directions that is of tremendous use to improve the overall system performance and user satisfaction.

Within the course of this dissertation research, some obstacles have been found in the evaluation of proposed methods. The lack of standard datasets for evaluation in the field of this dissertation hinders development of the field. The author of this dissertation with collaborators has worked hard to collect real-world data that allow other researchers to continue exploration in the field. We believe that in the end, the most important part of the work done for this dissertation is to offer some technical possibilities of our dissertation question that is not fully answered yet, *"what is the better way to provide route guidance and navigation information to the drivers?"* Trying answering this question has played a tremendous role in shaping the structure of this dissertation. However, fully answering the question is beyond the scope of one single dissertation and requires more than technology advances. We hope that we have succeeded in presenting some useful multimedia technologies for landmark-based vehicle navigation and some interesting new directions in the field. We look forward to contributing further to improve driving safety, efficiency and overall experience.

Bibliography

[1] R. Adams and L. Bischof. Seeded region growing. *IEEE Transactions on Patten Analysis & Machine Intelligence (PAMI)*, 1994.

[2] G. L. Allen, A. W. Siegel, and R. R. Rosinski. The role of perceptual context in structuring spatial knowledge. *Journal of Experimental Psychology*, 1978.

[3] B. S. H. Aoki and A. Pentland. Recognizing places using image sequences. In *Conference on Perceptual User Interfaces*, 1998.

[4] S. Ayache and G. Quenot. Evaluation of active learning strategies for video indexing. *Signal Processing: Image Communication*, 2007.

[5] S. Ayache and G. Quenot. Video corpus annotation using active learning. In *30th European Conference on Information Retrieval (ECIR'08)*, 2008.

[6] M. Belkin and P. Niyogi. Semi-supervised learning on riemannian manifolds. In *Machine Learning*, volume 56, pages 209–239. Kluwer Academic Publishers, 2004.

[7] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, 2002.

[8] A. C. Berg, T. L. Berg, and J. Malik. Shape matching and object recognition using low distortion correspondences. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2005.

[9] A. Blum and S. Chawla. Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the Intl. Conference on Machine Learning (ICML)*, 2001.

[10] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. In *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, 2001.

[11] Y. Y. Boykov and M.-P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *Proceedings of the IEEE Intl. Conference on Computer Vision (ICCV)*, 2001.

[12] G. E. Burnett. Usable vehicle navigation systems: Are we there yet? In *Vehicle Electronic Systems 2000 - European conference and exhibition*, June 2000.

[13] G. E. Burnett and D. Smith. Supporting the navigation task: Characteristics of 'good' landmarks. In M.A. Hanson, editor, *Proceedings of the Annual Conference of the Ergonomics Society*, 2001.

[14] G.E. Burnett. "turn right at the King's Head": drivers' requirements for route guidance information. *PhD thesis, Loughborough University, UK*, 1998.

[15] J. Callan. Distributed information retrieval. *Advances in Information Retrieval*, 2000.

[16] G. Carneiro and D. Lowe. Sparse flexible models of local features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006.

[17] S. Carr and D. Schissler. Perceptual selection and memory in the view from the road. *Environment and Behavior*, 1969.

[18] C. Carson, S. Belongie, H. Greenspan, and J. Malik. Blobworld: Image segmentation using expectation-maximization and its application to image querying. *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, 2002.

[19] S.-L. Chang, L.-S. Chen, Y.-C. Chung, and S.-W. Chen. Automatic license plate recognition. *IEEE Transactions on Intelligent Transportation Systems*, 5, 2004.

[20] D. Chen, J.M. Odobez, and H. Bourlard. Text detection and recognition in images and video frames. *Pattern Recognition*, 2004.

[21] X. Chen, J. Yang, J. Zhang, and A. Waibel. Automatic detection of signs with affine transformation. In *Proceedings of the IEEE Workshop on Applications of Computer Vision (WACV)*, pages 32–36, 2002.

[22] X. Chen, J. Yang, J. Zhang, and A. Waibel. Automatic detection and recognition of signs from natural scenes. *IEEE Transactions on Image Processing*, 13(1), January 2004.

[23] Y. Chen, J. Z. Wang, and R. Krovetz. CLUE: Cluster-based retrieval of images by unsupervised learning. In *IEEE Transactions on Image Processing*, volume 14 of *8*, pages 1187–1201, 2005.

[24] L. Chittaro and S. Burigat. Augmenting audio messages with visual directions in mobile guides: an evaluation of three applications. In *MobileHCI*, September 2005.

[25] K.-H. Chu, R. Brewer, and S. Joseph. Traffic and navigation support through an automobile heads up display (A-HUD). *ICS2008-05-02, Tech Report, Univ. of Hawaii at Manoa*.

[26] P. Clark and M. Mirmehdi. Estimating the orientation and recovery of text planes in a single image. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2001.

[27] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, 2002.

[28] D. J. Crandall, P. Felzenszwalb, and D. P. Huttenlocher. Spatial priors for part-based recognition using statistical models. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[29] R. Dale, S. Geldof, and J.-P. Prost. Using natural language generation in automatic route description. In *Journal of Research and Practice in Information Technology*, volume 37 of *1*, February 2005.

[30] T. Deselaers, D. Keysers, and H. Ney. FIRE - flexible image retrieval engine: ImageCLEF 2004 evaluation. In *CLEF 2004, LNCS 3491*, 2004.

[31] A. de la Escalera, J. M. Armingol, J. M. Pastor, and F. J. Rodriguez. Visual sign information extraction and identification by deformable models for intelligent vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 2004.

[32] J. Fan, D. K. Y. Yau, A. K. Elmagarmid, and W. G. Aref. Automatic image segmentation by integrating color-edge extraction and seeded region growing. In *IEEE Transactions on Image Processing*, 2001.

[33] C.-Y. Fang, S.-W. Chen, and C.-S. Fuh. Road-sign detection and tracking. *IEEE Transactions on Vehicular Technology*, 52:1329–1341, 2003.

[34] L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. In *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, 2006.

[35] P. Felzenszwalb and D. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 2004.

[36] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision*, 2005.

[37] A. Ferencz, E. G. Learned-Miller, and J. Malik. Learning to locate informative features for visual identification. *International Journal of Computer Vision: Special Issue on Learning and Vision*, 2007.

[38] V. Ferrari, T. Tuytelaars, and L. Van Gool. Simultaneous object recognition by unsupervised scale-invariant learning. *International Journal of Computer Vision*, 2006.

[39] FHWA. Manual on uniform traffic control devices (MUTCD) for streets and highways. In *The Federal Highway Administration (FHWA), U.S. Dept. Transp.*, page Available: http://www.fhwa.dot.gov/, 2003.

[40] R. Finkel and J.L. Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 1974.

[41] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Of the ACM*, 1981.

[42] T. Gandhi, R. Kasturi, and S. Antani. Application of planar motion segmentation for scene text extraction. In *Proceedings of Intl. Conference on Pattern Recognition (ICPR)*, 2000.

[43] D. M. Gavrila, U. Franke, C. Wohler, and S. Gorzig. Real-time vision for intelligent vehicles. *IEEE Instrumentation and Measurement Magazine*, 2:22–27, 2001.

[44] J. Goodman, S. A. Brewster, and P. Gray. How can we best use landmarks to support older people in navigation? In *Behaviour & Information Technology*, volume 24 of *1*, pages 3–20, January-February 2005.

[45] P. Green, E. Hoekstra, M. Williams, C. Wen, and K. George. Examination of a videotape-based method to evaluate the usability of route guidance and traffic information systems. *(Tech. Rep. No. UMTRI-93-31). Ann Arbor, MI: University of Michigan Transportation Research Institute.*, 1993.

[46] P. Green, W. Levison, G. Paelke, and C. Serafin. Preliminary human factors design guidelines for driver information systems. In *UMTRI-93-21*, 1995.

[47] P. Green and M. Williams. Perspective in orientation/navigation displays: A human factors test. In *The Third International Conference on Vehicle Navigation and Information Systems (VNIS)*, 1992.

[48] E. D. Haritaoglu and I. Haritaoglu. Real time image enhancement and segmentation for sign/text detection. In *Proceedings of the IEEE Intl. Conference on Image Processing (ICIP)*, pages 993–996, 2003.

[49] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[50] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[51] X. Hou and L. Zhang. Saliency detection: a spectral residual approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[52] Superimaging Inc. 2-color full-windshield display system. *www.superimaging.com*.

[53] Q. Iqbal and J. K. Aggarwal. Applying perceptual grouping to content-based image retrieval: Building images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1999.

[54] A. K. Jain and B. Yu. Automatic text location in images and video frames. *Pattern Recognition*, 31:2055–2076, December 1998.

[55] F. Jing, C. Wang, Y. Yao, K. Deng, L. Zhang, and W.Y. Ma. IGroup: web image search results clustering. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, 2006.

[56] F. Jing, B. Zhang, F. Lin, W.-Y. Ma, and H.-J. Zhang. A novel region-based image retrieval method using relevance feedback. In *ACM Workshops on Multimedia: Multimedia Information Retrieval*, 2001.

[57] S. Kaplan. Adaption, structure and knowledge. *G. Moore & R. Golledge (Eds.) Environmental knowing: Theories, research and methods*, 1976.

[58] V. Kastrinaki, M. Zervakis, and K. Kalaitzakis. A survey of video processing techniques for traffic applications. *Image and Vision Computing*, 21:359–381, 2003.

[59] L. Kennedy, M. Naaman, S. Ahern, R. Nair, and T. Rattenbury. How flickr helps us make sense of the world: context and content in community-contributed media collections. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, 2007.

[60] K. I. Kim, K. Jung, and J. H. Kim. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. In *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, volume 25 of *12*, pages 1631–1639, December 2003.

[61] S. Kim, S. Park, and M. Kim. Central object extraction for object-based image retrieval. In *Proceedings of the ACM Intl. Conference on Image and Video Retrieval (CIVR)*, 2003.

[62] G. F. King. Driver performance in highway navigation tasks. *Transportation research record*, 1986.

[63] B. Kroon, S. Boughorbel, and A. Hanjalic. Person-based search in videos. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM), Demo*, 2007.

[64] S. Kumar and M. Hebert. Man-made structure detection in natural images using a causal multiscale random field. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[65] X. Lan, C. L. Zitnick, and R. Szeliski. Local bi-gram model for object recognition. In *Microsoft Research Technical Report*, 2007.

[66] J. Lee, J. Forlizzi, and S. E. Hudson. Studying the effectiveness of MOVE: A contextually optimized in-vehicle navigation system. In *Conference on Human Factors in Computing Systems (CHI)*, 2005.

[67] B. Leibe, N. Cornelis, K. Cornelis, and L. V. Gool. Dynamic 3d scene analysis from a moving vehicle. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[68] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[69] H. Li, D. Doermann, and O. Kia. Automatic text detection and tracking in digital video. In *IEEE Transactions on Image Processing*, volume 1 of *9*, pages 147–156, 2000.

[70] L. Li, W. Huang, I. Y. H. G, and Q. Tian. Foreground object detection from videos containing complex background. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, 2003.

[71] Y. Li and L. G. Shapiro. Consistent line clusters for building recognition in CBIR. In *Proceedings of Intl. Conference on Pattern Recognition (ICPR)*, 2002.

[72] R. Lienhart. Automatic text recognition for video indexing. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, pages 11–20, Nov 1996.

[73] R. Lienhart and A. Wernicke. Localizing and segmenting text in images and videos. In *IEEE Transactions on Circuits and Systems for Video Technology*, volume 4 of *12*, pages 256–268, 2002.

[74] Y.-C. Liu and M.-H. Wen. Comparison of head-up display (HUD) vs. head-down display (HDD): driving performance of commercial vehicle operators in taiwan. *International Journal of Human-Computer Studies*.

[75] D.G. Lowe. Distinctive image features from scale-invariant keypoints. In *International Journal of Computer Vision*, 2004.

[76] B. Luo, X. Tang, J. Liu, and H.-J. Zhang. Video caption detection and extraction using temporal information. In *Proceedings of the IEEE Intl. Conference on Image Processing (ICIP)*, pages 297–300, 2003.

[77] K. Lynch. *The Image of the City*. MIT Press, 1960.

[78] T. Malisiewicz and A. A. Efros. Improving spatial support for objects via multiple segmentations. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2007.

[79] A. J. May and T. Ross. Presence and quality of navigational landmarks: Effect on driver performance and implications for design. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 2006.

[80] J. Miura, T. Kanda, and Y. Shirai. An active vision system for real-time traffic sign recognition. In *Proceedings of the IEEE Intelligent Transportation Systems Conference*, pages 52–57, 2000.

[81] G. Myers, R. Bolles, Q.-T. Luong, and J. Herson. Recognition of text in 3-D scenes. In *4th Symposium on Document Image Understanding Technology*, pages 23–25, April 2001.

[82] F. Nielsen and R. Nock. Clickremoval: Interactive pinpoint image object removal. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, 2005.

[83] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2006.

[84] G. Piccioli, E. De Micheli, P. Parodi, and M. Campani. Robust method for road sign detection and recognition. In *Image and Vision Computing*, volume 14, pages 109–223, 1996.

[85] J. Pilet, V. Lepetit, and P. Fua. Real-time non-rigid surface detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[86] A. Qamra and E. Y. Chang. Scalable landmark recognition using extent. In *Journal of Multimedia Tools and Applications*, 2007.

[87] A. Quattoni and A. Torralba. Recognizing indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[88] X. Ren and J. Malik. Learning a classification model for segmentation. In *Proceedings of the IEEE Intl. Conference on Computer Vision (ICCV)*, 2003.

[89] D. Robertson and R. Cipolla. An image-based system for urban navigation. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2004.

[90] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *Proceedings of the ACM SIGGRAPH*, 2004.

[91] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce. 3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints. In *International Journal of Computer Vision*, 2006.

[92] Y. Rui and T. S. Huang. A novel relevance feedback technique in image retrieval. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, 1999.

[93] B.C. Russell, A. Torralba, K.P. Murphy, and W.T. Freeman. LabelMe: a database and web-based tool for image annotation. *Intl. Journal of Computer Vision*, 2008.

[94] S. Sanyal and S. H. Srinivasan. Logoseeker: a system for detecting and matching logos in natural images. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, 2007.

[95] A. Sato, I. Kitahara, Y. Kameda, and Y. Ohta. Visual navigation system on windshield head-up display. In *13th World Congress on Intelligent Transportation Systems*.

[96] S. Savarese and L. Fei-Fei. 3d generic object categorization, localization and pose estimation. In *Proceedings of the IEEE Intl. Conference on Computer Vision (ICCV)*, 2007.

[97] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "How do i organize my holiday snaps?". In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2002.

[98] G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

[99] H. Shao, T. Svoboda, and L. Van Gool. ZuBuD - Zurich buildings database for image based recognition. *Technical Report No.260, Swiss Federal Institute of Technology*, 2003.

[100] T. T. H. Shao, T. Svoboda, and L. V. Gool. Hpat indexing for fast object/scene recognition based on local appearance. In *Computer lecture notes on Image and video retrieval*, 2003.

[101] J. Shi and J. Malik. Normalized cuts and image segmentation. In *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, volume 22 of *8*, pages 888–905, 2000.

[102] J. Shi and C. Tomasi. Good features to track. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.

[103] N. Shimizu. Google, Nokia transform car navigation market. *Nikkei Electronics Asia*, 2009.

[104] P. Silapachote, J. Weinman, A. Hanson, R. Weissy, and M. A Mattar. Automatic sign detection and recognition in natural scenes. In *IEEE Workshop on Computer Vision Applications for the Visually Impaired*, June 2005.

[105] J. Sivic and A. Zisserman. Video google: a text retrieval approach to object matching in videos. In *Proceedings of the IEEE Intl. Conference on Computer Vision (ICCV)*, 2003.

[106] A. W. M. Smeulders, M. Worring, A. Gupta S. Santin and, and R. Jain. Content-based image retrieval at the end of the early years. In *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, volume 22 of *12*, 2000.

[107] A. Steinfeld and P. Green. Driver responses to navigation information on full-windshield, head-up displays. *International Journal of Vehicle Design*, 1998.

[108] L. A. Streeter and D. Vitello. A profile of drivers' map-reading abilities. *Human factors*, 1986.

[109] A. Tom and M. Denis. Referring to landmark or street information in route directions: What difference does it make? *COSIT 2003 Lecture Notes in Computer Science 2825*, 2003.

[110] A. Torralba, K.P. Murphy, W.T. Freeman, and M. Rubin. Context-based vision system for place and object recognition. In *Proceedings of the IEEE Intl. Conference on Computer Vision (ICCV)*, 2003.

[111] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.

[112] S. Vitabile, A. Gentile, and F. Sorbello. A neural network based automatic road signs recognizer. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 2315–2320, 2002.

[113] T. Volkmer, J. R. Smith, and A. Natsev. A web-based system for collaborative annotation of large image and video collections: an evaluation and user study. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, 2005.

[114] J. Z. Wang, J. Li, and G. Wiederhold. SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. In *IEEE Transactions on Pattern Analysis & Machine Intelligence (PAMI)*, volume 23 of *9*, 2001.

[115] W. W. Wierwille, J. F. Antin, T. A. Dingus, and M. C. Hulse. Visual attentional demand of an in-car navigation display system. *Vision in vehicles II*, 1989.

[116] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[117] W. Wu, F. Blaicher, J. Yang, T. Seder, and D. Cui. A prototype of landmark-based car navigation using a full-windshield head-up display system. In *ACM Intl. Conference on Multimedia - Workshop on Ambient Media Computing*, 2009.

[118] W. Wu, D. Chen, and J. Yang. Integrating co-training and recognition for text detection. In *Proceedings of the IEEE Intl. Conference on Multimedia & Expo (ICME)*, 2005.

[119] W. Wu, X. Chen, and J. Yang. Incremental detection of text on road signs from video with application to a driving assistant system. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, 2004.

[120] W. Wu, X. Chen, and J. Yang. Detection of text on road signs from video. *IEEE Transactions on Intelligent Transportation Systems*, 6(4):378–390, 2005.

[121] W. Wu and J. Yang. SmartLabel: An object labeling tool using iterated harmonic energy minimization. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, 2006.

[122] W. Wu and J. Yang. Object fingerprints for content analysis with applications to street landmark localization. In *Proceedings of the ACM Intl. Conference on Multimedia (ACM MM)*, 2008.

[123] W. Wu and J. Yang. Semi-automatically labeling objects in images. *IEEE Transactions on Image Processing*, 2009.

[124] W. Wu, J. Yang, and J. Zhang. A multimedia system for route sharing and video-based navigation. In *Proceedings of the IEEE Intl. Conference on Multimedia & Expo (ICME)*, 2006.

[125] W. Zhang and J. Kosecka. Localization based on building recognition. In *1st IEEE Workshop on Computer Vision Applications for the Visually Impaired, Satellite Workshop of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[126] W. Zhao, R. Chellappa, A. Rosenfeld, and P. J. Phillips. Face recognition: A literature survey. In *ACM Computing Surveys*, 2003.

[127] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, A. Bissacco, F. Brucher, T.-S. Chua, and H. Neven. Tour the World: building a web-scale landmark recognition engine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[128] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Scholkopf. Learning with local and global consistency. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, 2004.

[129] X. Zhu. Semi-supervised learning literature survey. *Computer Sciences TR 1530, University of Wisconsin - Madison*, 2008.

[130] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proceedings of the Intl. Conference on Machine Learning (ICML)*, 2003.

[131] C. L. Zitnick, J. Sun, R. Szeliski, and S. Winder. Object instance recognition using triplets of feature symbols. In *Microsoft Research Technical Report*, 2007.