Augmenting Cartographic Resources and Assessing Roadway State for Vehicle Navigation

Young-Woo Seo

CMU-RI-TR-12-13

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics.

The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

April 2012

Thesis Committee:
Chris Urmson, Co-chair
David Wettergreen, Co-chair
Martial Hebert
John Krumm, Microsoft Research

This thesis work was supported through the General Motors/Carnegie Mellon University Autonomous Driving Collaborative Research Laboratory.

Keywords: Orthoimagery Analysis for Building Highway Maps and Parking lot Maps, Aerial Image Analysis, Perspective Image Analysis for Assessing Roadway State, Computer Vision, Machine Learning, Geographic Information System

To Won-Sun

Abstract

Maps are important for both human and robot navigation. Given a route, driving-assistance systems consult maps to guide human drivers to their destinations. Similarly, topological maps of a road network provide a robotic vehicle with information about where it can drive and what driving behaviors it should use. By providing the necessary information about the driving environment, maps simplify both manual and autonomous driving.

The majority of existing cartographic databases are built, using manual surveys and operator interactions, to primarily assist human navigation. Hence, the resolution of existing maps is insufficient for use in robotics applications. Also, the coverage of these maps fails to extend to places where robotics applications require detailed geometric information.

To augment the resolution and coverage of existing maps, this thesis investigates computer vision algorithms to automatically build lane-level detailed maps of highways and parking lots by analyzing publicly available cartographic resources, such as orthoimagery.

Our map-building methods recognize image patterns and objects that are tightly coupled with the structure of the underlying road network by 1) identifying, without human intervention, locally consistent image cues and 2) linking them based on the obtained local evidence and prior information about roadways. We demonstrate the accuracy of our bootstrapping approach in building lane-level detailed roadway maps through experiments.

Due to expected abnormal events on highways such as roadwork, the geometry and traffic rules of highways that appear on maps can occasionally change. This thesis also addresses the problem of updating the resulting maps with temporary changes by analyzing perspective imagery acquired from a vision sensor installed on a vehicle.

To robustly recognize highway work zones, our sign recognizer focuses on handling variations of signs' colors and shapes. Sign recognition errors, which are inevitable, can cause our system to misread temporary highway changes. To handle potential errors, our method utilizes the temporal redundancy of sign occurrences and their corresponding classification decisions. We demonstrate the effectiveness and robustness of our approach highway workzone recognition through testing with video data recorded under various weather conditions.

Two major results of this thesis work are 1) algorithms that analyze orthoimages to produce lane-level detailed maps of highways and parking lots and 2) on-vehicle computer vision algorithms that are able to recognize temporary changes on highways. Our maps can provide detailed information about a route, in advance, to either a human driver or a self-driving vehicle. While driving on highways, our roadway-assessing algorithms enable the vehicle to update the resulting maps with temporary changes to the route.

Acknowledgments

This thesis was possible only through the guidance and support of many others.

First and foremost, I would like to thank my wife, Won-Sun, for her unaffected support and endless love. She has always believed in me, convinced me I could do whatever I wanted to, and encouraged me to pursue my personal dreams. Without her sacrifices, the completion of this dissertation would have been impossible. Thank you and I love you. I would like to thank, Jun-Young (Allen) and Jun-Hyeok (Matthew) for being my sons and for understanding about my busy schedule. Their smiles give my life meaning and put everything in perspective. Won-Sun, Jun-Young, and Jun-Hyeok have all contributed to this dissertation more than one could ever imagine. They are the best things that have happened to my life.

I would like to thank to my two great advisers, Chris Urmson and David Wettergreen for their continued encouragement, support, guidance and patience. Chris provided me with an opportunity to work for the Urban Challenge and continued to provide me feedback and advice for my dissertation work, even though for the past three years he was at Mountain View. After Chris left for Google, David generously agreed to continue advising me as his student. David guided me through many difficult times with unwavering integrity and wisdom.

I would also like to thank my committee members, Martial Hebert and John Krumm, for the insightful discussions and suggestions for this work. Despite their busy schedules, they always found time to talk about my work, guide the direction, or offer needed advice. I would like to thank General Motors Inc., for supporting my PhD thesis work.

What now feels like a very long time ago, Professor Byoung-Tak Zhang at Seoul National University took chance in choosing me as one of his first graduate students. He taught me the basics of and best way of conducting research. He always encouraged us to go abroad to see what and how others research, which led me to meet Professor Katia Sycara on the Autonomous Agent Conference at Barcelona, Spain. She generously offered me an opportunity to come here to work with her. She helped me my research career mature. I appreciate the guidance and support of both these individuals.

Since then, I have spent more than a decade at the Robotics Institute. This place, I firmly believe, is the best and most unique place in the world to conduct robotics research. During my time at the Robotics Institute, I was privileged to interact with many exceptional people who inspired and helped me come this far. Particularly, while pursuing my doctorate, I have met a lot of bright and hard-working students (aka Robograds) who enlightened and hung out with me. I would like to thank my fellow Robograds for making my graduate study an exciting experience. Particularly, I would like to give my special thanks to Amir Degani and Mike Stilman for being buddies and willing to chat about anything from how to build a robot to everything about life; Nathan Ratliff, Kevin Peterson, Sebastian Scherer, Maxim Makatchev, Prasanna Velagapudi, Uland Wong for being collaborators and friends. I am also grateful, for sharing my hard and happy times with me in my mother tongue, to my Korean friends, Myung Hwangbo, Joonhwan Lee, Jaedong Kim, Jingu Heo, Kiho Kwak, and other Koreans, who dwelt and dwell Newell-Simon Hall. It was a pleasant and memorable experience to work and spend time with you.

Under different contexts and research projects, I was also extremely lucky to work with some of the RI faculty members who were willing to share their busy time with me. I would like to thank Paul Scerri, Drew Bagnell, Rahul Sukthankar, and John Dolan. While pursuing my doctoral degree, I was fortunate to be a part of the awesome Field Robotics Center where there is always something moving and cool around. While working on the Urban Challenge, I have had fun working with the Tartan Racing team members and have learned a lot from each of the team members. While working at Intelligent Software Agents group, I was extremely lucky to work with many generous and great lab mates who taught me how we do research here and helped me get accustomed to a completely different culture and research environment. I would like to thank Sean Owens, Robin Glinton, Paul Scerri, Martin van Nelson, Joseph Giampapa, and many others who have moved elsewhere, but have not been forgotten. I am also indebted to the staff members at the Robotics Institute who kept the institute running smoothly: Marliese Bonk, Rachel Burcin, Alan Guisewite, Jean Harpley, Sanae Minick, Suzanne Muth, Brian Staszel, and Debra Tobin.

I would like to thank for their support and love my grandmother, my family-in-laws and friends overseas. My special thanks go to my two life-long friends, JongChul Hwang and HanWook Kang.

Last but not least, I am deeply grateful to Jesus Christ for all that I have been given. Thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Mundane and trivial, driving on roads is still an essential and complex task of modern life. While driving, a person intentionally or subconsciously performs various behaviors in parallel: recognizing geometric shapes, perceiving the posted rules of the roads, observing the driving of other vehicles, steering the vehicle, preparing for the unexpected, and so on.

We humans drive so effortlessly well because we have an exceptional perception capability that enables us to recognize road geometry, to understand traffic rules by reading signs, and to comprehend traffic situations based on what we perceive and what we have experienced.[1] In addition, we have our own model of driving that is customized to our behavior through countless repetition and which plays an important role in handling something unexpected. Given this capacity, we can even drive flawlessly through entirely foreign terrain. Even so, cartographic resources can aid us, facilitating our arrival at the destination. In particular, cartographic information on hand-held devices can inform us of where we can drive (e.g., take a left turn on Forbes avenue) and of how we can drive (e.g., the speed limit is 30 miles per hour). Such information regarding roads is essential for driving, even in familiar places; it enables us to focus our attention on the regions that require detailed analysis. For example, while driving downtown, where pedestrians might jaywalk or heedlessly cross the road, we could give our attention to such potential hazards, instead of peering around for our destination, because cartographic information would keep us informed of the destination.

In a similar way, but even on a larger scale, cartographic information about road geometry and traffic rules plays a critical role for a robotic vehicle performing autonomous driving maneuvers. The value of such information was demonstrated during the 2007 DARPA Urban Challenge.[2] Figure 1.1 shows a sample of the route map used during the competition. As an example, the road-map in Figure 1.1 enabled a robotic vehicle to anticipate upcoming intersections. In particular, it informed a vehicle that the speed limit of a certain segment of road was 30 miles per hour and that the intersection (labeled “I14135”) was a yield-intersection. As a result, vehicles should

[1] Our perception of the scene is based not only on the immediate sensory readings, but on our long history of visual experiences and interactions with the world [Warren and Warren, 1968].

[2] The Urban Challenge (or the 2007 DARPA Urban Challenge) was a robot car competition in which competitors had to build vehicles capable of autonomously driving 60 miles amongst moving traffic in an urban environment. A good overview of the Urban Challenge can be found [Urmson et al., 2009]. Visit the following for more information about the Urban Challenge, `http://www.darpa.mil/grandchallenge/index.asp`

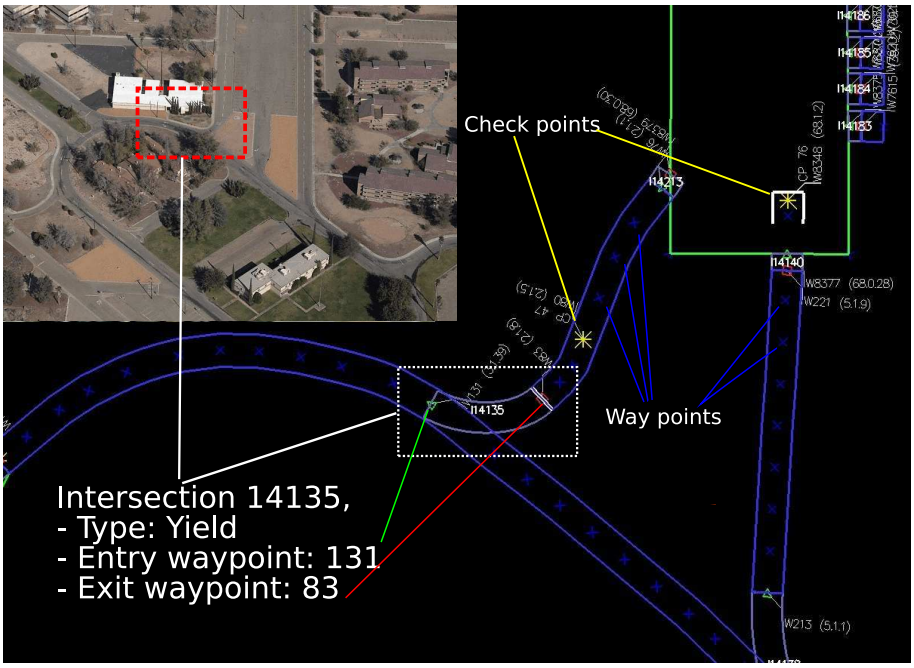


Figure 1.1: A part of our route map shows the starting chute of the Urban Challenge birds-eye aerial image. The route map is an internal representation of a robotic vehicle's driving environment. In its topological representation, there are a set of vertices (e.g., waypoints marked by blue "x" and checkpoints marked by yellow "*") and their connections. An intersection is a region that includes a subset of waypoints between entry and exit points [Seo and Urmson, 2008].

have been prepared to execute a "yielding" behavior, waiting for the road to clear before merging. Without this model, a vehicle would have difficulty understanding if this was a controlled intersection or a yield-type intersection.

Let us take another generic example of a self-driving vehicle's maneuver to clearly understand the role of cartographic information in robotic applications. The control loop of Boss, the winner of the Urban Challenge [Urmson et al., 2008, Urmson et al., 2009], can be abstracted into three parts: First, behaviors are initialized based on available map data (e.g., *handle_intersection or drive_down_lane*). Second, on-board sensor outputs are analyzed to interpret the surroundings (e.g., estimating the current pose or perceiving static and dynamic obstacles in drivable regions). Lastly, vehicle motions are executed to achieve current behavior goals (e.g., arriving at a particular waypoint by driving on a road-lane). In this loop, road-map information simplifies autonomous driving in that it dictates when each driving behavior should be implemented; allows a robotic vehicle to focus its attention on drivable regions that require detailed analysis, neglecting less important regions [Hebert, 1989],[Seo and Urmson, 2009a]; and provides guidelines on the execution of micro-level motions to achieve intermediate goals. Without prior knowledge of road geometry and associated behaviors, achieving the level of performance demonstrated during the Urban Challenge would be even more difficult.

However, the level of detail and the coverage of existing cartographic databases are far from being useful for such robotic applications because they are primarily built for human consump-

tions – for most of the cases, human drivers may not need lane-level detailed road-maps or parking lot maps for their driving. In this thesis we describe our approach to augment the contents of existing cartographic databases. We increase existing road-maps' level of detail by building lane-level highway maps and extend the coverage of existing cartographic databases by building maps of parking lots. Instead of taking conventional "going-out" map building approaches, we analyze publicly available aerial-images to accomplish our goal of building lane-level detail maps. In addition, we develop a bootstrapping image processing that exploits the prior information and analyzes readily-collectible image features to automatically harvest task-relevant and task-specific information to accomplish our goals.

1.1 Motivation

In the robotics community, the most common way of building a map is to drive through the environment to collect sensor measurements and then either manually or automatically fit a model to the collected data [Hebert, 1989, Thorpe et al., 1991, Thrun et al., 2005, Montemerlo et al., 2008, Urmson et al., 2008]. Such a model mostly describes parts of the environment that are not moving and are traversable. Such a navigational map simplifies autonomous navigation by providing an a priori model of the driving environments [Hebert, 1989, Seo et al., 2009a].

An alternative way of building such maps is to use overhead imagery. Building maps by pre-driving is expensive while at the same time high-quality overhead imagery is publicly available[3], the outdoor robot navigation community has begun taking an interest in building maps using overhead imagery analysis. On the other side, the GIS (Geographic Information System) researchers have long been working on building maps, mainly for human consumption, by analyzing overhead imagery. In the GIS community, the majority of map-building techniques demand human interactions and manual surveys of the regions under investigation. Thus building maps using overhead imagery analysis requires substantial effort. In addition, because their applications primarily target manual driving, they pay less attention to certain details of resulting maps and non-road drivable regions. For example, instead of representing a road-segment as a multiple of road-lanes, a polyline is considered to be enough for guiding manual driving. A point, instead of detailed geometry, is regarded as sufficient to represent a parking lot.

In this thesis, we suggest that we should analyze publicly available high-resolution orthoimagery to build maps of driving environments. Our first main contribution is

> **Automatic Building of Lane-level Maps:** *We propose algorithms that automatically generate lane-level maps of 1) inter-city highways and 2) parking lots by analyzing orthoimages.*

Such lane-level detail highway maps with information including traffic rules and accurate coordinates can be prepared in advance to facilitate the guiding of autonomous and manual highway driving. However, it is impossible to describe unexpected occurrences a priori, such as traffic

[3] Aerial imagery with approximately one foot resolution throughout the U.S. and its territories are publicly available from United States Geological Survey (USGS), `http://www.usgs.org`

accidents or road work. A self-driving vehicle must be able to effectively handle such events as they can lead to temporary changes in driving conditions. It would be disastrous if the vehicle's braking distance is longer than its sensing horizon when the road lane a vehicle is driving on shifts laterally due to a road work ahead, while the road-lane depicted on the map as following a straight path. Similarly a human driver must be on alert while driving through such unexpected events.

Hence our second main contribution is

> **Recognizing Temporary Changes in Driving Conditions on Highways***: We propose algorithms that automatically recognize 1) the bounds of workzones and 2) temporary changes of driving conditions on highways by analyzing perspective images.*

1.2 Problem Statement

Most existing cartographic databases are primarily built, through manual surveys, to assist human navigation. The resolution of maps are insufficient for use in robotics applications and their coverage fails to reach any places where robotics applications require detailed geometric information. This thesis addresses the problem of automatically generating lane-level maps of highways and parking lots by analyzing publicly available orthoimages.

Due to the expected aberrations that appear on highways, such as road-work, the geometry and traffic rules of highways that appear on maps can occasionally change. This thesis also addresses the problem of updating the resulting map with temporary changes of driving conditions by analyzing perspective imagery acquired from a vision sensor installed on a ground vehicle.

1.3 Thesis Statement

This thesis demonstrates that:

> *Overhead* and *perspective* imagery can be combined to generate *sub-meter accurate* cartographic information.

In particular, by *overhead* imagery, we mean, publicly available orthoimage. An orthoimage is an aerial image where terrain relief and camera tilt are removed through a rectification process. In this dissertation, the ground resolution of orthoimages is 15 centimeter per pixel.

By *perspective*, we mean, an image that is acquired from a front-looking vision sensor installed on a ground vehicle. The scenes appearing on the image are distorted by perspective transformation.

By *sub-meter accurate*, we mean, the accuracy of the resulting maps' geometry. The accuracy will be defined and measured at a pixel-level that is readily converted into real-world distances based on the ground resolution of an image. The accuracy will be further analyzed by such standard metrics as precision and recall.

1.4 Our Approach

To build maps by analyzing orthoimages, first we must be able to find photometric or geometric patterns regarding the underlying true maps. Such patterns may include image intensity contrasts along road boundaries, color of road-markings, texture of road-surface image regions, periodic rectangular shapes, and spatial relations among these patterns. Being able to distinguish the boundaries of road-lanes on highways may seem obvious just as it might for distinguishing parking spots in parking lots. These seem obvious on account of the background and its seemingly typical and salient geometric patterns, such as parallel line whitish lane-markings along road-direction and rectangular road-markings grids. Without considering the variations of their salient appearances, it seems to be fairly straightforward to extract the geometric structures of road-lanes' boundaries and parking lots first by detecting these lane-markings and then by connecting them based on their regularity.

However, these assumptions are not always valid. These salient and regular patterns are not always available for image processing; the actual values of lane-marking pixels vary based on image acquisition conditions. The image acquisition conditions are determined by illumination conditions, the intrinsic and extrinsic parameters of a camera, and the line of the sight between an acquisition-vehicle and the ground with respect to the location of the sun. Such variations in object appearance are the most serious challenges in analyzing imagery to extract meaningful patterns.

To effectively handle variation in an object's photometric and geometric appearance, one could learn appearance models from data that consist of image patches and their class assignments. But how much data would be sufficient to produce output with accuracies which are acceptable for a given task? The machine learning community has been actively searching for a generic answer to this question, the most commonly used solution is the rule-of-thumb that the desirable amount of data is determined by the complexity of the data and the problem. For our cases, it would require a huge amount of data to train a lane-marking detector that produces a reasonable performance on every image of the area of interest. Indeed, the data should have at least one sample for every possible appearance variation which is hard to quantitatively measure.

In this thesis, instead of taking such a conventional way of obtaining patterns of interest from input images, we exploit the prior information in a given image to extract task-relevant patterns. The prior information is the information that is already available when a problem is formulated and is relevant in solving the problem. For example, we utilize the regularities of our target objects, such as the parallelism of road-lane boundaries to extract road-width cues and image sub-regions encompassed by evenly-spaced rectangular road-markings to extract a set of self-labeled parking spot examples. These task-relevant patterns obtained locally from each test image provide us with a sufficient amount of cues about the local geometry of the underlying true highway road-lanes and true geometric structure of parking lots. Such local-specific patterns are useful to our map building application. This thesis, after all, primarily concerns the extracting of the true geometry of road-structures, which are partially observed in a given image. Our approach of harvesting task-relevant local patterns through bootstrapping will also reduce the frequency of human intervention.

Our approach of exploiting prior information can be applied to the task of workzone sign detection as well. For the purpose of evaluating a sign detector's outputs, it is necessary to

annotate each video by drawing bounding boxes of true workzone signs. By using such ground truth annotations, we can define an image sub-region within which workzone signs are most likely to appear by projecting all of the ground truth bounding boxes onto a Cartesian coordinate. The prior knowledge of sign locations is used to facilitate a search of sign candidate image blobs and to filter out false positive sign detection outputs.

1.5 Document Outline

The rest of this document is organized as follows. Chapter 2 highlights research work significantly related to our thesis work, particularly in the areas of map-building, aerial-imagery analysis, and perspective image analysis. Chapter 3 details our approach to generate lane-level detail maps from highway orthoimages. Chapter 4 explains our approach to recognizing the bounds of highway workzones and temporary changes to highways. Chapter 5 describes our approach to produce maps of parking lots that specify the locations of parking spots and the geometry of drivable regions. Finally, in Chapter 6, we conclude and discuss future directions of work.

Chapter 2

Related Work

This thesis work focuses on developing computer vision algorithms for analyzing publicly available aerial images with application goal of building highway maps and parking lot maps. Therefore, it is useful to investigate available cartographic resources and existing map building techniques. Section 2.1 investigates some of the work on building road-map information used for robotic vehicles to drive autonomously. Section 2.2 surveys four of the existing cartographic resources in terms of the available data to populate road-map structures for robotic applications. The robotics community has been developing map building techniques for robot navigation while the GIS community has primarily focused on maps for human consumption. Section 2.3 compares our approach with existing raster and orthoimage analysis in the GIS community. In section 2.3.2, we investigate how overhead data such as orthoimagery and digital elevation maps are used in the robotics community. While analyzing orthoimage, our algorithms employ techniques from image processing, computer vision, and machine learning. Section 2.3.3 investigates relevant research works in computer vision and image processing. Unlike most of the existing techniques in computer vision and robotics community, this thesis work explores a method that acquires task-relevant patterns through bootstrapping to minimize human intervention. Section 2.3.4 investigates two machine learning approaches that minimize human involvement in machine learning tasks.

To address temporary changes of driving conditions on the resulting highway maps, a part of this thesis work aims at developing algorithms for analyzing perspective images with application goal of recognizing temporary changes to a highway. Section 2.4 reviews computer vision and machine learning techniques related to the task of recognizing temporary changes of driving conditions. In particular, section 2.4.1 surveys existing techniques that detect and classify traffic sign by analyzing perspective videos. Most of existing work in this field focuses primarily on improving the accuracy of their sign recognitions whereas ours has error handling methods to accurately infer the properties of temporary events. Section 2.4.2 reviews some of the object recognition works that explicitly model how to handle recognition errors.

2.1 Use of Cartographic Information for Robotic Applications

We are concerned with the usefulness of existing cartographic resources for automatic roadmap generation, thus we first review prior work in building road network information using onboard sensor measurements and then investigate the usage of this information by teams that participated in the Urban Challenge.

2.1.1 Building Roadmaps for Robotic Applications

A common feature of these approaches is the use of onboard sensor measurements, which are collected by pre-driving a target area. In fact building maps is practically its own field: SLAM (simultaneous localization and mapping) which addresses the problem of building maps by analyzing sensor measurements while also localizing robots' poses [Thrun et al., 2005],[Thrun and Montemerlo, 2005],[Durrant-Whyte and Bailey, 2006]. Because we are mainly investigating techniques for building, by analyzing aerial images, maps in urban environments, work related to SLAM and off-road environments will not be reviewed here.

Hebert devises a roadmap building algorithm that analyzes range measurements to generate a roadmap [Hebert, 1989]. The algorithm estimates the statistics of reflectance data around road regions from manually labeled data; finds road regions in images by thresholding reflectance values; and then fits two parallel lines onto the left and right edges of the estimated road regions. Road mapping is done by aligning two points in a sequence of consecutive images.

Thorpe and his colleagues [Thorpe et al., 1991] build a roadmap for their mail delivery vehicle in a semi-automatic way. They drove a robotic vehicle to annotate global coordinates of landmarks in the robot's operating environment by using a laser range finder. The resulting roadmap is primarily used for vehicle localization.

There are two similar work to ours in terms of building maps for autonomous parking lot driving. Dolgov and Thrun devise algorithms that build a lane-network graph of a parking lot from sensor readings from their robotic vehicle [Dolgov and Thrun, 2009]. They first build a grid map of static obstacles from range measurements and then use Markov Random Fields to infer a topological graph that most likely fits the grid map. They define a series of potentials to incorporate their prior on a road network. However, instead of directly minimizing these potentials imposed on road segments, a generalized Voronoi diagram is used as a subset of the topological road network.

Kummerle and his colleagues build a multilevel (or multilevel surface) map of a parking building [Kummerle et al., 2009]. A multilevel-map is a 2D grid map that each of cells maintains a stack of patches. As individual patches in a cell correspond to different height estimates, this multilevel structure is used to represent drivable regions and vertical objects. To fill in individual cells, they first formulate a mapping as a graph construction problem that a node represents a vehicle pose and an edge represents a relative motion between poses; and then find optimal nodes based on constraints imposed on edges. A new node is continuously added to the graph until a loop closure is found.

2.1.2 Use of Cartographic Information for the Urban Challenge

For the Urban Challenge, we also manually created models of road networks using a combination of GPS survey and overhead imagery [Urmson et al., 2008]. Performing a GPS survey requires manually driving a vehicle, which is capable of estimating its pose, through an environment that an autonomous vehicle is intending to driving through. The set of surveyed GPS coordinates should be aligned with the orthoimagery of the region to enable annotating characteristics of the route.

Manually built road networks were extensively utilized in the Urban Challenge [Bacha et al., 2008, Bohren et al., 2008, Miller et al., 2008, Montemerlo et al., 2008, Urmson et al., 2008]. Like many mobile robot navigation problems, autonomous driving in urban environments is in essence way-point[1] navigation from a starting location to a goal location. Each waypoint in a road network is represented as a node with directed edges, coming in and out from the node, connecting logically reachable waypoints. These edges are associated with various navigation costs such as expected time to traverse, length and other information related to autonomous driving [Ferguson et al., 2008a, Urmson et al., 2008]. Such a detailed information about a road network was saved in a Route Network Definition File (RNDF) format defined by the DARPA for the Urban Challenge [DARPA, 2007].

Some example uses of road network information in the Urban Challenge include:

- *Geometric Information about Drivable Regions* This includes a representation of the lane centers and widths and parking lot boundaries [Montemerlo et al., 2008, Urmson et al., 2008].
- *Mission (Route) Planning*: The goal of mission (or route) planning is to choose a globally optimal path between two geographic locations using the road network [Bohren et al., 2008, Miller et al., 2008, Montemerlo et al., 2008, Urmson et al., 2008]. The optimality of a path may be determined by considering static (e.g., speed limits) and dynamic factors (e.g., temporary blockages).
- *Macro-level Motion Planning (or behavioral system)*: Given a path to a goal, a behavioral system executes macro-level motions: lane-changing, precedence-handling, on-road driving, intersection-handling, yielding, and so on [Urmson et al., 2008].
- *Micro-level Motion Planning (or local motion planner*): In on-road driving, the local motion planner generates a set of trajectories along the centerline of the road lane and choose one of them that satisfies the optimality condition [Ferguson et al., 2008b].
- *Perception* The road geometry provided from road network information is used to localize vehicles' locations with respect to road-boundaries [Leonard et al., 2008, Urmson et al., 2008] and validate detected vehicles and other obstacles [Bacha et al., 2008, Miller et al., 2008, Urmson et al., 2008].

During the Urban Challenge, road network information was extensively used to provide robotic vehicles with information about driving environments that is hard to acquire from on-board sensor measurements. This information was crucial for robotic vehicles to perform reliable and intelligent maneuvers in an urban environment.

[1] A waypoint is a reference point that identifies a geographic location.

Following the successful demonstration of autonomous driving in the Urban Challenge, research efforts on the development of autonomous vehicles has surged. We review two such studies as they relate to our thesis work. The relationship is in terms of augmenting a road-vector database with additional information that can be later used for autonomous driving and of utilizing the information of road networks.

Fairfield and Urmson develop a perception algorithm for detecting traffic lights and mapping their locations [Fairfield and Urmson, 2011]. The work aimed at obtaining precise geo-spatial information about traffic lights to facilitate an autonomous driving task of handling intersections controlled by traffic lights. To obtain such information, they drove a mapping vehicle, which is capable of localizing its pose in sub-meter accuracy, and used a camera to collect traffic light images. Using the geometric information from a pair of Google maps and vehicle position information, and traffic sign image classification, they choose some of the traffic light images to estimate traffic light positions – more than two images associated with a traffic light are used to estimate traffic light position through a linear triangulation. This information is used to augment a roadmap database with detailed information about traffic lights, e.g., their positions and types. This is then used to inform a self-driving vehicle of where to look to infer precedence about when to drive through traffic-light controlled intersections.

Frankel and his colleagues developed a system that aids human drivers to safely change lanes on highways [Frankel et al., 2010]. To change lanes on a highway, it is necessary to know precisely a vehicle's position with respect to its road-lane. Their vehicle localized its pose with a GPS-based Inertial Navigation System (INS) system that periodically reports a sub-meter accurate pose on average. But, this position estimate combined with inaccurate RNDF would result in the vehicle's occasional crossing of the centerline. To correct this, they used local sensor measurements that allowed them to identify boundaries of road-lanes [Montemerlo et al., 2008]. They also augmented the content of RNDF with other information necessary for performing highway lane-changes, such as lane heading and the current point's geometric relation to other points of interest, e.g., the closest highway or waypoints in the same or other lanes. Additionally, road network information was used to direct perception modules to pay attention to the region of interest, e.g., clearance of lanes to merge and moving vehicles of interest to track.

These works differ from ours in that they need to manually drive a robot to collect range measurements.

2.2 Existing Cartographic Resources

There are a number of cartographic resources publicly and commercially available. Although they are developed and maintained primarily for human navigation, it is useful to review their properties to contrast human navigation with autonomous driving.

In this section, we review three publicly available resources such as orthomaps (or orthophotos) from the United States Geological Survey (USGS), the Google Maps API, and the Topologically Integrated Geographic Encoding and Referencing (TIGER) from the U.S. Census Bureau. We also review one commercially available resource: Navteq's roadmap database.

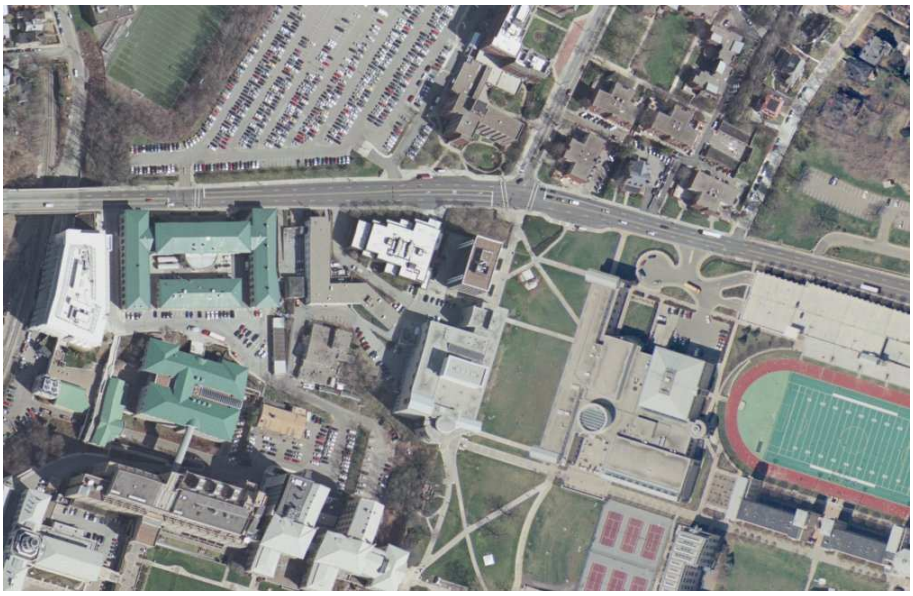


Figure 2.1: This orthoimage was obtained in 2006 and its resolution is 10000 $\times$ 10000 (1 feet/pixel), approximately covering 3,040 $\times$ 3,040 meters. The original size of this image is 286 Mega bytes.

2.2.1 USGS' Orthomaps

U.S. Geological Survey (USGS) [2] is a US governmental organization that provides geological services. Some of the commercial map service providers use USGS' orthoimages or satellite images. There is a significant amount of data publicly available such as LIDAR (Light Detection And Ranging), digital raster graphic, non-rectified satellite images, etc. Among these, this section only reviews the orthophotos (aka orthomaps, orthoimagery).

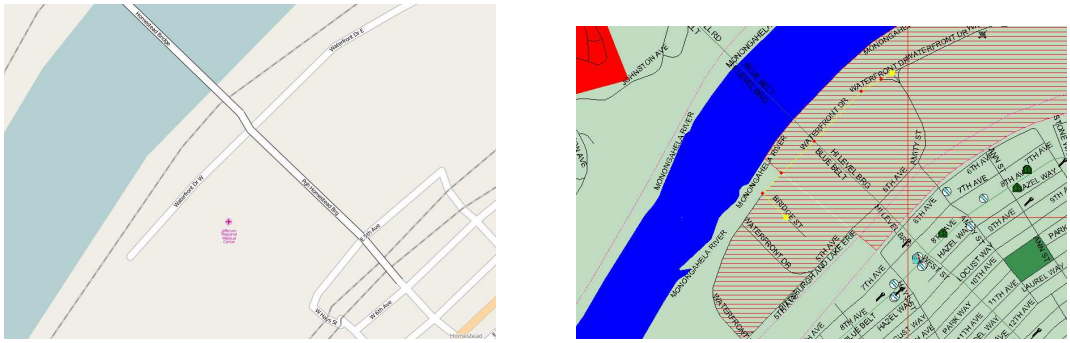
An image without the effects of topography (or relief displacement) and camera tilt is called an orthoimage (or orthophoto) [3] and has a uniform scale. Since an orthophoto has a uniform scale, it is possible to directly measure distance on it like other maps. An orthophoto may also serve as a base map onto which other map information can be overlain. When an orthomap is combined with other digital products, such as digital raster graphics (DRG) or digital elevation models (DEM), the resulting image provides additional visual information for the extraction and revision of base cartographic information. Figure 2.1 shows a high-resolution orthoimage of the Carnegie Mellon University campus.

2.2.2 Google's Maps API

Google Inc. provides registered users access, through their Application Programming Interface (API), to their cartographic databases. Since there is no publication about Google's cartographic

[2] `http://www.usgs.gov`

[3] This is because orthoimagery has orthographic properties rather those of the central perspective of the original aerial photograph.



(a) A part of the TIGER roadmap database.

(b) A part of Navteq's roadmap database.

Figure 2.2: The "Waterfront" region near Pittsburgh is depicted by two roadmap databases.

database, it is not possible to review their structures in detail.

Google had been using roadmaps from a number of the map providers, including Tele Atlas,[4] Europa technologies,[5] USGS,[6] NASA, GeoEye,[7], Sanborn,[8] DigitalGlobe,[9] etc, to provide users with various interesting visualizations through their Maps APIs.[10] Since October 2009, Google has switched to using their own maps. Table 2.1 compares some of the Google Maps API's features with three other cartographic resources.

2.2.3 TIGER

The TIGER (Topologically Integrated Geographic Encoding and Referencing system or TIGER/Line) is a cartographic database used by the U.S. Census Bureau to describe land attributes such as roads, building, rivers, and lakes.[11]

TIGER data includes complete coverage of the United States territories and includes both land attributes such as roads, buildings, rivers, and lakes, as well as areas such as counties, census tracts, and census blocks.

Figure 2.2(a) shows a screen capture of a web-based map service provider that uses TIGER as their basemap. As seen in this figure, TIGER has been utilized as a base map that provides the connectivity information about the region of interest.

[4] `http://www.teleatlas.com/index.htm`
[5] `http://www.europa.uk.com/`
[6] `http://www.usgs.gov`
[7] `http://www.geoeye.com/CorpSite/`
[8] `http://www.sanborn.com/`
[9] `http://www.digitalglobe.com/`
[10] `http://code.google.com/apis/maps/documentation/reference.html`
[11] The TIGER is publicly available at, `http://www.census.gov/geo/www/tiger/`. For the technical descriptions, refer to "TIGER/Line shapefiles: Technical documentation."

2.2.4 Navteq's Roadmap Database

Navteq[12] is one of the major GIS data providers. Their roadmap database seems to be built based on first hand observation of geographic features.

Due to a limited access to the latest version of Navteq's roadmap database, this section focuses a brief overview of Navteq's road map database structure.

In Navteq's roadmap database, there are three primary cartographic types: point, polygon, and polyline. We describe these primary data types in detail and review their usages.[13]

- *Point*: Is used to represent points of interests (POIs) such as restaurants, hotels, etc.
- *Polygon*: Is used to represent a cartographic polygon, such as a lake or a park.
- *Polyline*: Is used to represent a cartographic polyline, such as a road, walkway, railroad, etc.
    1. Relative speed category: Specifies the relative speed category
    2. Number of lanes: Describes how many lanes this polyline represents
    3. Pavement: Describes the status of the pavement on a road segment (e.g., paved, private, frontage, bridge, etc.)
    4. Accessibility: Describes the types of transportation allowed on that segment (e.g., automobiles, buses, taxis, carpools, pedestrians, bicycles, trucks, through traffic, deliveries, emergency vehicles)
    5. Driving direction: Explains the driving direction of a road segment.

Figure 2.2(b) shows screen captures of the "Waterfront" region where the corresponding part of the vectorized roadmap is depicted. This map shows the same area depicted in figure 2.2(a), but it contains more detailed information about the area.

We review four existing cartographic resources that are built primarily for human consumption. Because of their primary usage, these resources are not directly applicable to use as roadmaps for autonomous driving where precise information about road structures is required. For example, to drive reliably and safely, a robotic vehicle needs to know the width, curvature, and speed limit of a road segment that a human drive can easily obtain while driving on the road segment. Reviewing their properties elucidates what kind of information is necessary to build roadmaps for autonomous driving; which of cartographic resources are useful as a complement resource, and; how the proposed framework might be helpful in building roadmaps for human consumption. Table 2.1 compares the properties provided by four cartographic resources in the perspective of the information relevant to autonomous driving.

[12] `http://www.navteq.com`

[13] Refer to "NAVTECH SDAL version 1.7 Programmer's Reference" for more information.

	USGS' Orthomaps	Google Map APIs	TIGER/Line	Navteq SDAL
Geocoding	No	Yes	Yes	Yes
Reverse geocoding	No	Yes	Unknown	Unknown
Waypoints along path	No	Possible	Yes	Possible
Curvature	No	Possible	Possible	Possible
Traffic rules encoding	No	No	No	Partial
Parking lot boundary	No	No	No	No
Speed limit	No	No	No	Yes
Number of lanes	No	No	No	Partial
Pavement status	No	No	No	Yes
Road accessibility	No	No	No	Yes
Driving direction	No	Possible	Unknown	Yes
Availability	Public	Public/Commercial	Public	Commercial

Table 2.1: Comparison of properties provided by existing cartographic databases. "Unknown" means the feature is not known at the time of survey, "partial" means "there is something under the category, but not enough to implement the feature," and "possible" means that "there is a way to implement the idea."

2.3 Overhead Data Analysis

2.3.1 Aerial Image Analysis in the GIS Community

The GIS community has a broad focus including building maps for human consumption in various applications and contexts; theoretical studies of road network structure for developing faster algorithms for handling geospatial data [Eppstein and Goodrich, 2008], extracting connectivity of roads from raster maps [Chiang and Knoblock, 2008], localizing moving objects on known road networks [Wang and Zimmermann, 2008], and many other topics. Among these, research on raster map analysis shares the most commonality with our approach in that it involves extracting interesting features from a raster images of maps [Chen et al., 2006b, Chiang and Knoblock, 2008, Khotanzad and Zink, 2003] and satellite images (e.g., IKONOS, SPOT, etc) [Geman and Jedynak, 1996],[Haverkamp, 2002]. Some of this work use specialized aerial images, such as color-infrared [Grote et al., 2007], panchromatic [Gruen and Li, 1995], multispectral images [Doucette et al., 2004], [Zhang, 2006] to emphasize regions of interest by utilizing invisible parts of electromagnetic radiation. Some also make use of data in a different modality such as airborne Light Detection and Ranging (LIDAR) measurements [Zhu and Mordohai, 2009, Qian et al., 2010] or surface elevation information [Schpok, 2011] to analyze regions of interest from different point of views. Only some of

these works will be reviewed here because their work is significantly related to this thesis work in terms of analyzing overhead data for extracting or updating road network.

Aerial Imagery Analysis for Cartographic Databases Maintenance In [Chiang and Knoblock, 2008], the authors present image processing algorithms that obtain the location and orientation of road intersections from raster images and estimate the connectivity of the region under investigation. They utilize combinations of morphology operators (e.g., thinning and thickening) and heuristics. The distinction between their raster image analysis and ours lies in the characteristics of the images. Extracting data from rasterized maps has different challenges than analyzing overhead aerial imagery (e.g., aliasing and distorted color vs. variations in illumination and appearance, occlusions).

In orthophoto analysis, Chen and his colleagues propose a conflation algorithm that integrates two different geospatial data sets such as vectorized road maps and orthoimages [Chen et al., 2006a]. Their approach is similar to ours in that they use a classification algorithm (a naive Bayes classifier) to estimate boundaries of roads and generate and filter out hypotheses on interesting points for conflation. But their filtering is guided by information in vectorized road maps whereas ours is based on learning distributions directly from imagery.

A common problem is to extract geospatial features from aerial imagery and use them to refine and update records in geospatial database [Baltsavias and Zhang, 2005, flavie Auclair Fortier et al., 2000]. In [flavie Auclair Fortier et al., 2000] the authors present an aerial image processing algorithm that utilizes up-to-date aerial images to modify the content of road maps. Through multiple steps of image processing, a set of line junction candidates is identified. The database of vectorized roadmaps is then used to filter out line junctions for which image coordinates are distant from those of actual intersections. The remaining line junctions are used to match the closest intersections and are used to adjust the coordinates of the matched intersections. Geman and Jedynak use a machine learning technique for tracking roads in satellite images. A decision tree is used to test individual pixels' image features, such as the presence of arcs and local filter responses, to determine whether they belong to roads [Geman and Jedynak, 1996]. Their algorithms are intended to be a part of an interactive aerial image analysis system where an operator provides the system with starting points and directions for tracking road networks. Similarly, Hu and his colleagues developed a heuristic-based road tracking algorithm for extracting road networks appearing on low-resolution aerial images [Hu et al., 2007]. They first approximated the local geometry of roads centered at sampled points by investigating intensity changes along line segments coming out from the points. They connected some of these local road polygons based on heuristics and refined the resulting road network based on statistics of ratios of areas to perimeters.

Overhead Data Analysis We detect interesting road structures, such as intersections and overpasses, for identifying potentially complex road geometry. For example, knowing the boundary of an overpass is useful in correctly extracting the boundaries of road-lanes around it. We approximate the geometry of the underlying road network by using a screenshot of a road-vector. For depicting 3D structures in (birds-eye view) aerial images, Schpok proposed an overpass detection method using road-vector databases with surface elevation information [Schpok, 2011]. They first searched for a list of potential overpass locations by investigating the surface elevations of locations sampled along a road-vector. These overpass candidate locations are grouped together based on the similarity of their geometric properties, i.e., road span. To define the

boundaries of overpasses, they expanded outgoing edges of each cluster to include neighboring road spans until terminating conditions were satisfied. An overpass is then finally reconstructed using road spans and surface elevation information.

Some work in the GIS community use 3-dimensional laser scan measurements to detect road-structures and to extract a road network. Qian and his colleagues used 3D point clouds to detect the boundaries of an overpass [Qian et al., 2010]. They collected 3D lidar scans by driving on highways and analyzed point clouds based on heuristic of basically identifying and grouping "jumps" to detect overpasses. We use road-vector sketches depicted on map images to detect overpasses and analyze extracted lines to identify the boundaries of the detected overpasses. This process is what distinguishes ours from these approaches using 3D information.

Zhu and Mordohai proposed a road network extraction algorithm from aerial LIDAR range measurements [Zhu and Mordohai, 2009]. Similar to our approach for generating lane-level detail highway maps, they formulated a road network extraction problem using a min-cover scheme. Individual range scans were grouped together based on three-dimensional information and some of the scan groups are classified as ground planes based on the groups' geometric shapes. All the scan points on the ground planes were projected onto a 2D plane to produce an image of LIDAR intensity values. They used responses of polarized rectangular filters and textures to extract features of the boundaries and interiors of roads. These boundary and road features were used to generate road hypotheses. The likelihoods of these being true road regions are computed based on convolution. In spite of the different overhead data to analyze and methods to generate hypotheses about true road regions, the model of their road hypothesis bears a strong resemblance to ours in that it is represented as a polyline in which each edge (or control point) is associated with geometric properties such as width and direction. They used a minimum cover idea to search for a set of road hypotheses that maximally covers a likelihood map while minimizing the cost of generating hypotheses. Similar to our approach, they found a greedy solution, which look for a locally optimal solution, to approximate the optimal solution of a NP-hard minimum cover problem.

Significant research has been done extracting road network structures from overhead aerial imagery. However, to the best of our knowledge, our work is the first attempt to, by analyzing publicly available aerial-images, build roadmaps sufficient for autonomous driving.

2.3.2 Overhead Data Analysis in Robotics

Overhead imagery data can be utilized to provide prior information about environments for outdoor robot navigation. Despite being potentially out of date, aerial image analysis can provide an important structural overview of operational environments that can enable robots to plan globally to achieve their goals. In combination with other onboard sensors such as vision sensors and range finders, overhead imagery offers an avenue for generating a complete view of an operating environment.

Schematic Overview of Operating Environments Silver and his colleagues utilize overhead data to produce cost maps for long-range traversals [Silver et al., 2006]. They fuse multiple overhead data sources to compute relative measures of mobility risk in regions of interest. Similarly, Sofman and his colleagues use overhead aerial images to generate long-range traversability maps [Sofman et al., 2006]. They use local range estimations as self-labeled examples to learn

relations between the characteristics of local terrain and corresponding regions in aerial images. These learned relations are used to map aerial images to long range estimates of traversability over areas that the robot may explore. Vandapel and his colleagues use aerial LIDAR survey data to obtain a long-distance traversability map for an environment [Vandapel et al., 2003]. A manned helicopter is flown over regions of interest multiple times to collect high point density data (4 to 10 points per square meters). Raw range measurements are divided into 1m$\times$1m cells and then categorized into either ground or vegetation. The areas classified as ground are used to populate a traversability map.

Persson and his colleagues use aerial images to disambiguate local sensor measurements [Persson et al., 2008]. An occupancy grid-like map is built using range measurements. Edges extracted from geo-referenced aerial images of a region are used to determine the class of cells (building or not). This matching process extends the robot's myopic local perception.

Global Localization Overhead orthoimages are also used to solve the global localization problem [Dogruer et al., 2008]. Dogruer and his colleagues propose a localization algorithm that utilizes Monte Carlo Localization to identify the location of a mobile robot in an urban environment. This algorithm is essentially about matching features from local sensor measurements to features extracted from overhead imagery (e.g., buildings detected in laser scans and identified in the overhead images). Overhead images are segmented beforehand into several regions: building, vegetation, ground, and (asphalt) road. Because the robot is always driving on roads in their experimental settings, if the world model is accurate (i.e., if the segmentation result accurately depicts where roads are), the distribution of particles about the robot location converges to the correct location over time.

Similarly, Carle and Barfoot utilized overhead data for a long-range localization of their vehicle [Carle and Barfoot, 2010]. Local sensor measurements were used to match features appearing on a 3D orbital elevation map through feature constellations. Particularly they used, for the feature matching, multiple-steps of hypothesis refinement and, for the pose refinement, a combination of RANSAC (RANdom SAmple Consensus)[Fischler and Bolles, 1981] and parts of SLAM techniques.

Urban scenes appearing on aerial imagery have also been utilized to localize the position of an unmanned aerial vehicle (UAV) [Soleimani et al., 2010]. The authors claimed that a geometric structure of a road network is useful for search-and-rescue robotic applications because road structures are disaster invariant. A geometric structure of a road network was obtained by detecting roads appearing on aerial images and then was represented as a grid map, in which the value of each grid cell was a real value of being occupied by roads ranging between 0 and 1. This representation is used to match simulated local sensor measurements of a UAV.

World Modeling Scrapper and his colleagues utilize aerial images to build a world model of a robot's operating environment [Scrapper et al., 2003]. To provide prior information about an operating environment, they manually annotate topographic data and features (e.g., woods, rivers, roads) by aerial survey and manual driving of the robot. Using the current pose of the robot (measured by a combination of GPS and IMU), the robot searches for the list of objects possibly observed from its current location and interprets its local sensor (e.g., color camera, an imaging ladar) measurements by its confidence about the measurements of its surroundings. Such a world model database can simplify autonomous navigation in regions covered by the database. The approach of building a world model is similar to that of [Urmson et al., 2008].

In this work, the road world model is built by a combination of aerial imagery and GPS survey. The GPS survey is carried out by manually driving a robotic vehicle on the region of interest. A set of the surveyed GPS coordinates is associated with aerial images of the corresponding area. However, as we pointed out earlier, although such a world model simplifies autonomous navigation in regions covered by the database, manually building the model is labor-intensive and error-prone.

Overhead imagery has been used as a complement to onboard sensor data to guide outdoor robot navigation. Regions in an overhead image are labeled based on corresponding measurements from onboard sensors. To the best of our knowledge, no work in overhead imagery analysis attempts to utilize patterns in images as training examples for extracting road structures in the image. Because low-level image patterns share common image characteristics of local objects, they are useful to detect road structures in the images. In this regard, this thesis work is the first attempt to automatically build road maps by recovering road structures while minimizing the use of manually-labeled data.

2.3.3 Overhead Imagery Analysis in Computer Vision

In this section, we will consider aerial image analysis for object boundary detection, image road cue tracking and spatial structure recognition. We will then investigate relevant object detection, structure recovery, and optimization methods in computer vision.

Object Boundary Detection Object boundary detection has been considered as one of the most important problems in the computer vision community [Martin et al., 2004, Zheng et al., 2007]. This is because knowing the boundaries of objects appearing in images helps researchers meet the objectives of several computer vision tasks, such as object recognition [Dollar et al., 2006, Zheng et al., 2007] and segmentation [Malik et al., 2001, Martin et al., 2004], to name only two. A great deal of high-quality research work on this field has been published. However, we review that which is significantly related to this thesis work. Martin and his colleagues proposed a learning-based natural object boundary detection method [Martin et al., 2004]. Using image data where object boundaries are manually annotated, they trained models of boundaries' image characteristics, such as discontinuities in brightness, as well as the texture and color values of neighboring pixels. In our study, the boundaries of road-lanes are, at least by definition, clearly specified as lane-markings. But the variations of their appearances are high. We learned a classifier to tackle such variations. Similar to the learning method in [Martin et al., 2004], to learn contrast between lane-marking pixels and neighboring background pixels, we use a set of features, such as Local Binary Pattern (LBP) [Ojala et al., 2002] and some other textural statistics. Their seminal work also provided the computer vision community with a bench-mark testbed. We used their probabilistic boundary outputs to compare the performance of our road-lane boundary detection. Dollar and his colleagues' approach to detect object boundaries is similar to ours and methods described in [Martin et al., 2004] to identify objects' boundaries, they trained a classifier, probabilistic boosting tree, from manually labeled image data to identify objects' boundaries [Dollar et al., 2006]. What sets theirs apart is how they represented their features. They used a set of generic features, such as Haar wavelets and image gradients at multiple scales, to represent a patch around a boundary pixel, instead of just using a pixel. Their feature representations are in fact effective for identifying objects and for even picking up roads

appearing on low-resolution aerial images. Similar to Dollar and his colleagues, Mnih and Hinton used a set of generic features to train a road-detector, using Restricted Boltzmann Machines (RBMs). This detector was to learn a binary model that could classify whether a given image patch was a part of the road [Mnih and Hinton, 2010]. To obtain the training data, they utilized road-vector database to label aerial images. This idea also bears a resemblance to our method of obtaining labeled superpixel images for road-region segmentation. Instead of just using a set of local image patches as a training data, they compiled a pair of a pixel and its neighboring pixels and used them as a training example to model regional image characteristics. To smooth out classification outputs focused on local image patches, they repeatedly applied a local filter at fixed intervals over the entire image region.

Tracking of Road Image Cues For our methods of generating highway maps, we connect hypotheses about true road-lanes based on photometric and geometric image cues. While linking road-lane hypotheses, we develop a heuristic for tracking road-boundary images cues. There are two aerial imagery analysis works that employ a Bayesian filter to track road cues. Zhou et al use two non-linear Bayesian filters, such as the extended Kalman filter and particle filter, to track roads appearing on digital orthophotos and to extract parts of the underlying road network [Zhou et al., 2006]. Given initial locations by human operators, the filters move forward according to their motion models and choose the best path based on the observation models, which match intensity values of neighboring pixels around the current pixel location with the next possible locations. Similarly, the authors use unscented Kalman filter for tracking roads appearing on satellite images [Movaghati and Moghaddamjoo, 2008]. They employed different filters based on their assumptions on the non-linearity of road boundaries. The idea of using non-linear Bayesian filters to track roads is similar to our approach of tracking road-boundary cues in that the tracking direction is adjusted by photometric road-cues. The main difference between ours and these approaches is the ground resolution of test images. Variations of object appearances in a low-resolution aerial imagery, i.e., greater than 1 meter per pixel, are not as significant as those of high-resolution imagery (e.g., 15 centimeter per pixel).

Perceptual Grouping To develop a function for linking road-lane hypotheses based on geometric image constraints, we employ Gestalt laws of grouping image features in a non-accidentalness, considering proximity, smooth continuation, parallelism and compactness [Palmer, 1999]. In particular, a possible linking between two road-lane hypotheses is assigned a higher value if its Euclidean distance is shorter while the linking angle is smaller. Three other studies specifically implement such a perceptual grouping of image features based on Gestalt laws. These other studies used it as follows: 1) to complete object boundary contours [Estrada and Jepson, 2006], 2) to close contour of an object boundary represented by linear lines [Elder and Zucker, 1996], and 3) to group edges based on their symmetries [Stahl and Wang, 2006]. These approaches resemble to ours in terms of modeling Gestalt laws for boundary completion, but differ in terms of implementing the linking function. Particularly we formulated a linking as a cost minimization in the min-cover framework.

Parking Lot Structure Analysis There are a similar work in the realm of spatial structure recovery. Wang and Hanson [Wang and Hanson, 1998] propose an algorithm that uses multiple aerial images to extract the structure of a parking lot for simulation and visualization of parking lot activities. Multiple images from different angles are used to build a 2.5 dimensional elevation map of the parking lot. This usage of multiple images makes it difficult to generalize

their method because it is not easy to obtain such images of the same geographic location from publicly available imagery.

Most prior work in parking lot image analysis focuses primarily on detecting empty parking spots in surveillance footage when the overall geometrical structure of the parking lot is known [Fabian, 2008, Huang et al., 2008, Wu et al., 2007]. Our work on parking lot geometry analysis addresses an alternative problem: extracting the entire parking lot structure from overhead satellite imagery. Similarity between our work and these works on empty parking spot detection lies in the fact that we utilize coherent structural patterns over an image region to infer the availability of certain parking spots.

With a recent increase of web-based cartographic services, aerial image analysis has attracted increased attention from the computer vision community. This thesis proposes a combination of detection and optimization for recovering relevant spatial structures. This approach implicitly assumes contextual associations among objects in aerial imagery [Oliva and Torralba, 2007].

Exploitation of Spatial Coherence Three computer vision works that effectively exploit contextual relations among spatial objects are [Porway et al., 2008], [Heitz and Koller, 2008], and [Kluckner et al., 2009]. Porway and his colleagues propose a statistical framework that hierarchically interprets objects in a given aerial image [Porway et al., 2008]. A scene in an aerial image is modeled by a coherent spatial relationship among cars, roads, and parking lots. By using a set of labeled aerial images, they first obtain appearance models of relevant objects such as trees, parking lots, roofs, and roads. False positives of these object detections are filtered by their "top-down" contextual models that are estimated in a minimax entropy framework. While this work is similar to ours, they rely on manually labeled examples and do not attempt to generate fine-grained geometry for the detected spatial structures. Heitz and Koller present a graphical model for detecting objects in aerial imagery [Heitz and Koller, 2008]. Their models learn a conditional distribution for the presence of an object given particular image features. In other words, the presence of particular objects is inferred from image regions in which individual pixels share similar image features such as color or texture. Kluckner and his colleagues present an aerial image analysis algorithm that classifies a given aerial image into one of the several predefined classes such as "tree", "streetlayer", "waterbody", "tree," and "building" [Kluckner et al., 2009]. They use sigma-points to represent several generic features, such as color in Lab, texture, height information, and train a randomized forest [Breiman, 2001]. To smooth out classification outputs to individual pixels, they use a conditional random field that makes the assigned class labels of neighboring pixels similar to one another. Geraud and Mouret proposed a bottom-up approach to extract a road network appearing on low-resolution aerial imagery [Geraud and Mouret, 2004]. As we collect low-level image features, they use the watershed transform to produce a superpixel image. Two intersection points between superpixel boundaries define a curve that potentially represents a part of road. They built a curve adjacency graph about the connectivity of these curves. They formulate the task of extracting road network from the curve adjacency graph as graph labeling problem and use a pairwise MRF to model optimal interactions between curves. A simulated annealing is used to find the most probable state of the graph, equivalent to the most probable road network given an input image.

Feature Representation For supervised object recognition from images, the development of techniques and theories has been largely focused on two parts: methods that effectively represent images as feature vectors and methods that assign image features with correct labels. Since it is

very hard to develop a technique or theory at either extreme, most work leverages both.

Feature representation refers to the projection of data into another space where the original data is represented in a more compact form. One of the ways to achieve this is to reduce the original dimension with minimal loss of information. Turk and Pentland utilize principal component analysis to effectively represent high-dimensional human face images in a low-dimensional space without significant loss of data [Turk and Pentland, 1991]. They choose the k most significant eigenvalues of the covariance matrix of face images and use the corresponding eigenvectors to form a new low-dimensional face space. They showed a significant reduction in feature dimensions while maintaining the performance of the full feature space for their face detection task.

Another purpose of feature representation is to inject invariance into the original images so that features are less sensitive to variation of illumination and appearance. As an example, the Histogram of Oriented Gradients (HOG) feature is used to describe local object appearance and shape in an image by the distribution of intensity gradients (or edge directions) and has been used in pedestrian detection [Dalal and Triggs, 2005]. An image is divided into either evenly-spaced or connected region cells. For each cell, the HOG features are obtained by counting occurrence of gradient (discretized) orientations. These local histograms can be illumination or shadow-invariant by combining several of them into a bigger cell (called a block) and normalizing their histograms.

Object Recognition Once we have a good representation, the next major step for detecting objects in imagery is to learn the models of the target objects from the data. A tremendous amount of work has been done in this area. In general, there are two major theoretical approaches: discriminative and generative. A discriminative approach solves the object detection problem by finding a decision boundary from labeled data whereas a generative approach solves the same problem by first modeling the data generation process and then using an estimated model to assign labels probabilistically [Duda et al., 2001].

Successful discriminative methods used in computer vision include support vector machines [Dalal and Triggs, 2005, Felzenszwalb et al., 2008], and ensemble machines such as AdaBoost [Viola and Jones, 2004], composition boosting [Porway et al., 2008] or randomized forests [Breiman, 2001].

Probabilistic graphical models are canonical examples of generative approach. Markov networks and their variants have been extensively studied in computer vision community under the topic of modeling such spatial relationships. These techniques include Markov Random Fields [Freeman et al., 2000], [Li, 2000], [Singhal et al., 2003] or its variants such as Conditional Random Fields [Carbonetto et al., 2004], [Weinman et al., 2004] and Discriminative Random Fields [Kumar and Hebert, 2005]. The fundamental idea of Markov networks is that the value of an image pixel cannot be independent of its neighbors. To support this idea, they offer numerous ways of massaging the joint probability distribution of random variables. In particular, these techniques are flexible in modeling real worlds problems because they offer compatibility functions, which are used to model interactions among variables of interest, and provide well-defined techniques for solving learning and inference problems in undirected graphical models. These techniques have been mainly applied to computer vision and their application areas which include augmentation of range measurements by intensity images [Diebel and Thrun, 2005], generation of the optimal navigable path [Nabbe et al., 2004], 3D reconstruction of environments from a sin-

gle image [Saxena et al., 2007], image segmentation [Freeman et al., 2000], and object recognition [Verbeek and Triggs, 2007], [Gallagher and Chen, 2007], [Weinman et al., 2004]. Although application areas differ, the most frequent problem to solve is an inference problem: to find the most probable state of the world given an image [Koller et al., 2007].

In our parking lot structure recognition, we utilize a MRF to find the most probable layout of parking spots as a cue for the underlying parking lot structure in a given aerial image. Specifically, we hypothesize the locations of the true parking spots by choosing a number of image locations. In our MRF implementation, each of these hypotheses is modeled as a random variable and their joint probability distribution is factorized by undirected graph. We choose an undirected graph to estimate the joint density because it is hard to identify causal directionality among random variables. Since it is also difficult to learn the optimal structure of an undirected graph primarily due to absence of directionality among the variables, we assume that there is a lattice structure of the undirected graph upon the detected parking spots. The given structure allows only pairwise interactions among the variables. Given a particular structure and values of random variables, one of the most interesting problems to solve is to know the most likely parking lot structure. For solving this inference problem, there are three different types of inference techniques: exact inference, sampling-based approximate inference, and variational/belief propagation approximate inference [Jordan, 2004], [Koller et al., 2007]. We choose loopy belief propagation [Freeman et al., 2000], [Yedidia et al., 2003] for solving the most likely labeling on parking spot hypotheses for its simple implementation.

Road-marker detection is a very flexible method for autonomous driving and aerial image analysis. The detected road-markers may be used for driver-assistance system and urban structure analysis from overhead aerial images. A major problem in detecting road-markers from images is that the appearance of road-markings are not consistent because of occlusions by other objects, quality and age of markings, illumination differences. For handling inconsistent appearance in road-markers, researchers have used different color spaces instead of directly using RGB values. Sun and his colleagues use the HSI (Hue-Saturation-Intensity) color space and devised a heuristic to determine when saturation values of images should be used [Sun et al., 2006]. Li and his colleagues devise a heuristic that converts RGB values into another space where lane-markings are more salient [Li et al., 2003]. In particular, a clustering algorithm is used to identify the probable image regions, which contain road-markers, from the transformed image and a connected-component algorithm is used to detect road-markers. Lipski and his colleagues also convert multiple road images in the HSI color space into an overhead image and analyze local histograms of color distributions [Lipski et al., 2008]. They combine these color distributions with inputs from other sensors such as lidar and radar to identify lane-markings.

Our road-marker detector is a classification method that assigns individual pixels to binary labels [Seo et al., 2009b]. Self-labeled parking spots are used as the samples for learning the color distribution of road-markers. The learned Gaussian distribution is used to execute a pixel-wise classification – assign each of pixels with a binary label, either “road-marker” or “non-road marker.”

Addressing Variations in Objects’ Appearances An alternative way of handling the inconsistent appearance problem is to use geometric primitives such as straight lines or curves. The Hough transform and its variants have been used to extract straight lines to detect road-markers [Saudi et al., 2008], [Voisin et al., 2005], [Yu and Jain, 1997]. Spline

functions and their variants have also been used to detect road-markers and their shapes [Wang et al., 2000], [Wang et al., 2004] by linearly connecting the detected straight lines. Wang and Hanson sample the image characteristics of road-markers by manually identifying road-markers around parking spaces and use them to identify the rest of road-markers in a parking lot [Wang and Hanson, 1998]. Lacoste and her colleagues [Lacoste et al., 2005] utilize a set of lines extracted along roads appearing on low-resolution aerial images and use a stochastic process, point processes, to link these lines, in order to extract the underlying road network.

Part-based detection approaches [Felzenszwalb and Huttenlocher, 2005] have been extensively studied. The underlying idea is conceptually appealing in that structures or complete objects can be modeled by part-objects in a deformable configuration. This is similar to our approach in that the detection of parts assists in the localization of road structures. This approach requires learning appearance models of part-objects and correlations between parts from supervised examples. The estimated models are applied probabilistically or discriminatively to search the probable image regions of part-objects or target objects. Felzenszwalb and his colleagues devise a hybrid approach for part-based object recognition that is called a latent SVM. In their pedestrian detection application, a discrete SVM is used to detect part-objects (e.g., arms or legs) and a target object (e.g., pedestrians). A probabilistic method is used to infer the configuration that optimally associates the detected part-objects with the detected target object [Felzenszwalb et al., 2008]. Similarly, Saragih and his colleagues present a part-based face alignment where the same face in two different images is aligned by first building a response map of face parts (e.g., eyes, nose) and then optimizing the difference between two faces in different images [Saragih et al., 2009]. These methods implementing part-based object recognition are different from ours in that it is not always obvious to define parts of the relevant road structures. Although, in our parking lot structure recognition, it may be straightforward to define parking spots as the parts of a parking lot structure, parts of a road segment vary based on the geometry of the road segment.

2.3.4 Machine Learning Methods for Reducing Human Interventions

As a part of an onboard system in an autonomous vehicle, it is desirable for automatic roadmap building algorithms to have minimal human intervention. Because of this, self-supervised learning is attracting attention from the robot learning community since it requires no (or substantially less) human involvement for carrying out learning tasks. This framework is highly desirable for robot learning because it is usually hard to collect large quantities of high-quality human-labeled data from any real world robotic application domain. In this section we investigate two machine learning frameworks: self-supervised learning and incremental learning.

Self-supervised learning frameworks typically utilize the most precise data source available to label other data sources that are complementary, but unlabeled. This approach has been used to extend a mobile robot's sensing coverage by combining local range measurements with other measurements such as overhead imagery, inertial measurements, and camera imagery.

Stavens and Thrun utilize laser range measurements to predict terrain roughness [Stavens and Thrun, 2006]. They first analyze the associations between inertial data and laser readings on the same terrain and use the learned rules to predict possible high shock areas in upcoming terrains. Similarly, Sofman and his colleagues use local range estimates as self-labeled

examples to learn relations between the characteristics of local terrain and corresponding regions in aerial images [Sofman et al., 2006]. The learned relations are used to map aerial images to long range estimates of traversability over regions that a robot is exploring. Lieb et al devised a self-supervised approach to road following that analyzes image characteristics of previously traversed roads and extracts templates for detecting boundaries of upcoming roads [Lieb et al., 2005].

Local range measurements can also be used to label scenes in images. Vision sensors usually see wider and farther than range finders, but labeling images from vision sensors is difficult mainly due to variation in appearance and illumination. In [Dahlkamp et al., 2006], local range measurements less sensitive to those variations are used to identify local drivable regions around a vehicle by analyzing the characteristics of image parts corresponding to those range measurements. These learned features are then used to predict other parts of images that are not covered by range finders. Katz and his colleagues use vehicle images labeled by range measurements to train a moving obstacle classifier [Katz et al., 2008]. In related work, Brooks and Iagnemma use vibration data to train a classifier to identify the roughness of upcoming terrains [Brooks and Iagnemma, 2007]. During navigation on different terrains, the vibration data is collected by recoding power spectral density using a microphone attached to a front wheel. This vibration data is aligned to video data that is taken from a forward-looking camera. Since the vibration data has higher frequency when wheels are climbing on rocks, image data captured at the same time is used to train a terrain classifier that identifies rock-regions from images taken by the forward-looking camera. Unlike other self-supervised learning examples, they trained a classifier for vibration data and then used the classifier for generating inputs for a vision-based terrain classifier. Kim and his colleagues utilize a mobile robot's previous experience navigating to determine if the upcoming terrain is traversable [Kim et al., 2006]. While interacting with operating environments, a mobile robot accumulates its experience such as slippage and collisions from its internal sensors and uses them as positive (or negative) data to label images taken at time of the event. Nair and Clark exploit motion information in video to automatically collect training examples [Nair and Clark, 2004] for the task of detecting people from the video of an office corridor scene. Their self-labeler is obtained by implementing careful observations of the task. They first manually obtain a background model by averaging several consecutive frames that do not contain any significant motions; then acquire the model of foreground by analyzing the differences of pixels in two consecutive frames; and finally collect training examples of people by grouping the connected foreground pixels. An example is classified as a person if the dimension of a connected region satisfies a heuristically defined threshold.

In our parking lot geometry analysis task, the self-labeler analyzes the spatial arrangements between extracted lines, which are aligned with road-markings in a parking lot, and produces a set of nominal parking spot images that are used as training examples for existing machine learning techniques [Seo et al., 2009a]. Furthermore these self-labeled examples are used to guide a random selection of negative examples; to provide a parking lot boundary segmentation with a cue of drivable regions' image characteristics and a road-marker classification with samples of road-markers' image characteristics [Seo et al., 2009b].

A self-supervised learning approach also works well in domains where it is hard to define what to learn in advance. For example, detecting geologic features in images is challenging because their appearances are significantly affected by illumination conditions. Unless replicating

all of the possible illuminating conditions, it is not possible to clearly define the appearances of interesting geologic features a priori. [Thompson et al., 2005].

Incremental learning is a machine learning approach where the initial model of the target function is estimated by using a small number of examples and is continuously updated as new examples arrive. In this framework, the learner needs to know only what is actually necessary for the specific task or data. This framework fits well our aerial image analysis work where only a small number of pertinent examples are available at the beginning of the learning and need to learn the local appearance model of road structure parts.

At the outset, our algorithms for parking spot detection are exposed to a small number of nominal parking spot examples and learn the initial model of parking spot appearances. Later the algorithms are given several aerial images containing parking spots in unusual appearances (e.g., trapezoidal geometric shapes or different illumination conditions). The initial appearance model should be generalized to accommodate the changes of feature values.

Incremental learning is closely related to lifelong learning (or transfer learning). Thrun and Mitchell address the lifelong robot learning problem in terms of transfer learning [Thrun and Mitchell, 1995], [Thrun, 1996]. They are interested in learning a collection of control policies for a robot with multiple tasks in both known and unknown environments. These environments are previously unknown to the robot. To this end, their algorithms first identify invariants about a robot's sensors, effectors and environments between tasks under both the same environment and the different environments. The learned invariants are transferred to a new task to expedite the learning process. For example, a explanation-based neural network is used to figure out invariants of action models, which is transferred to inject a bias to another policy learning task.

Rosenberg and his colleagues proposed an object detection algorithm that incrementally learns intra-class variation on human faces by utilizing unlabeled images [Rosenberg et al., 2005]. The unlabeled images are exploited to minimize efforts of human labeling and to generalize the learned knowledge of human faces. In particular, the detector is initially trained with labeled data and then assigns unlabeled data with classification confidences. Unlabeled data with high-confidence values are used as training data with the supervised ones to expand the classifier's knowledge of varying appearances of faces. They choose high-confidence unlabeled data to increase the number of training data whereas our incremental learning selects the k most uncertain unlabeled examples because uncertain unlabeled examples are the ones to learn in order to optimally move the decision boundary.

2.4 Perspective Image Analysis for Recognizing Highway Workzones

2.4.1 Traffic Sign Detection and Classification

In this section, we compare our approach to highway workzone recognition with previous work in the area of traffic sign recognition. For any traffic sign recognition method that focuses on vision sensors, an initial requirement is to locate potential sign image regions from an input image. Some systems, including ours, use color information to localize signs. In addition, for any sign

recognition system which utilizes color, it is necessary to find an optimal range of target color values because the actual values of the target color vary based on image acquisition processes. These threshold values are often obtained empirically by repeating manual surveys of pixel color values from sample sign images [Eichner and Breckon, 2008, de la Escalera et al., 2004, Yuille et al., 1998]. Because of its simple implementation, such a manual process is attractive, yet tends to be error-prone and expensive. By contrast, our approach automatically obtains the limits of optimal color-values through binary pixel-classifier training.

Another dominant approach for traffic sign detection is to use of sign shapes. Some researchers use the geometric property of sign shapes, such as equiangularity, in order to locate the centroids of traffic signs [Barnes et al., 2008, Loy and Barnes, 2004]. This approach is intrinsically error-prone because it relies on a geometric property, which is not preserved under perspective imaging, and also because it assumes high contrast in image intensity, which is hard to acquire from real-world image acquisition. An alternative approach for utilizing the geometric properties of signs is to locate parts (e.g., corners or edges) of a traffic sign and to combine the results of these partial detections. For example, in order to identify potential sign image locations, some researchers have used Haar-like features [Schlosser et al., 2010, Timofte et al., 2009, Viola and Jones, 2004] and their variants, such as a set of rectangular features in particular color channels [Bahlmann et al., 2005] and non-symmetric dissociated dipoles [Baro et al., 2009] or a variant of the histograms of oriented gradients (HOG) [Overett and Petersson, 2011]. This learning approach demonstrated successful performance only when a large number of manually labeled data was available to train the detector on multiple-scales for a long period of time.

2.4.2 Traffic Sign Recognition Error Handling

Some of the existing methods [Overett and Petersson, 2011, Timofte et al., 2009] have demonstrated very impressive recognition results in their experimental setups, e.g., a detection rate of more than 98.8%. However, in general, it is unrealistic to expect perfect performance in sign recognition. Most traffic sign recognition methods may miss a workzone sign or may also incorrectly classify a sign image in a stream of perspective images. Such inevitable errors would cause any sign recognition method to misunderstand the traffic rules and road geometry. To cope with such potential object recognition errors, Viola and Jones proposed a hierarchical image processing structure that trains classifiers at particular levels within a hierarchy until their performance, e.g., false negative rates, reaches a desirable level [Viola and Jones, 2004]. However, because such an approach requires a large amount of manually labeled data, it is inapplicable for our case, as it is very hard to collect a large number of workzone sign images. We instead utilize our sign classification output, which is highly accurate per image, in a twofold manner: first, we propagate classification confidence values toward future in order to reduce the impact of false negatives; second, we investigate previous classification decisions for the same sign in order to reduce the number of false positives.To the best of our knowledge, we have not seen such methods, particularly in handling potential sign recognition errors for recognizing temporary highway changes, in the field of traffic sign classification.

We developed such methods for handling potential sign recognition errors to minimize the frequency of misreading traffic rules and road geometry temporarily altered by highway road works. In the Intelligent Transportation Systems (ITS) community, there are three similar works

to ours in terms of assessing roadway state. But their methods are different from ours because, instead of analyzing traffic sign images, they used vehicle- or transportation infrastructure-generated data. For example, the vehicle-generated data include measurements of wheel-turning-speed, lane-change, and location over time [Ma et al., 2009],[Tabibiazar and Basir, 2011] whereas infrastructure-generated data include traffic flow, occupancy, and vehicle counts over unit time [Niu and Liu, 2011]. The collected data were used to train a classifier to infer roadway state. For example, Ma and his colleagues compiled vehicle driving information from simulated roadside units and used a SVM to classify the state of a route into three different states: normal, passed by incident site, and stopped in queue [Ma et al., 2009]; Niu and Liu trained a neural network to identify whether the traffic condition of a highway is normal [Niu and Liu, 2011]; Tabibiazar and Basir used a Gaussian mixture to model congestion sites based on GPS data [Tabibiazar and Basir, 2011].

2.5 Summary

In this section we reviewed several research fields related to this work.

The maps of robots' operating environments are primarily static. These models simplify autonomous navigation by providing a strong prior on environments. Most of these maps are created by analyzing sensor measurements obtained from pre-driving in target areas. Because pre-driving based map building is expensive and high-quality overhead imagery is publicly available, overhead imagery based map building has been intensively studied.

The review of existing cartographic resources, which are built primarily for human consumption, clarifies the desirable properties of roadmaps for autonomous driving.

In the robotics community, overhead data has been utilized as a complement to onboard sensor measurements. Low-level image patterns, which are useful to detect local objects in the images, have not been extensively utilized yet.

Most of the work in recovering relevant objects in overhead imagery relies on manually labeled data to learn an object detector (or classifier). Because manual labeling is expensive and error-prone, there have been several efforts to minimize the frequency of supervised examples while maintaining good performance. Self-supervised and semi-supervised learning in the machine learning community have been intensively studied [Chapelle et al., 2006]. Particularly, the self-supervised learning approach is attractive in that it needs no (or substantially less) human involvement. Despite its benefits, self-supervised learning approach or acquiring task-specific patterns via bootstrapping has not been extensively studied in aerial image analysis.

This thesis will provide use cases of recovering road structures in orthoimagery through bootstrapping. Our bootstrapping approach is unique in that it minimizes human involvement while effectively recovering road structures. Several heuristics analyze low-level image patterns, such as spatial arrangements of lines, and collect a set of task-specific examples based on analyzed patterns. The task-specific mid-level image features are used to train conventional machine learning algorithms as part-detectors that identify the probable regions of road structures and are used to generate highly probably hypotheses about unknown true road-lanes. Our approach exploits these self-obtained task-specific image features to seek a spatial configuration that optimally satisfies geometric and image constraints of part detection results.

	Geometric Structure Recovery	Parkinglot Geometry Analysis	Roadway Geometry Analysis	Lane Network Generation	Use of self-labeled Data	Incremental Learning	Part-based Object Recognition	Optimization of Detected Parts	Traffic Sign Recognition	Highway State Assessment	Application Domains
[Hebert, 1989]	√	–	√	–	–	–	–	–	–	–	Robotic vehicle navigation
[Thorpe et al., 1991]	√	–	√	√	–	–	–	–	–	–	
[Sofman et al., 2006]	√	–	–	–	√	–	–	–	–	–	
[Dolgov and Thrun, 2009]	√	√	–	√	–	–	–	–	–	–	
[Kummerle et al., 2009]	√	√	–	–	–	–	√	–	–	–	
[Nair and Clark, 2004]	–	–	–	–	√	√	–	–	–	–	Pedestrian detection
[Felzenszwalb et al., 2008]	–	–	–	–	–	–	√	√	–	–	
[Rosenberg et al., 2005]	–	–	–	–	–	√	√	–	–	–	Face alignment or detection
[Saragih et al., 2009]	–	–	–	–	–	–	√	√	–	–	
[Wang and Hanson, 1998]	√	√	–	–	–	–	–	–	–	–	Aerial image analysis
[Porway et al., 2008]	√	–	√	–	–	–	√	–	–	–	
[Heitz and Koller, 2008]	√	–	–	–	–	–	–	–	–	–	
[Timofte et al., 2009]	–	–	–	–	–	–	–	–	√	–	Traffic sign recognition
[Overett and Petersson, 2011]	–	–	–	–	–	–	–	–	√	–	
[Ma et al., 2009]	–	–	–	–	–	–	–	–	–	√	ITS
Goal of this thesis	√	√	√	√	√	√	√	√	√	√	**All**

Table 2.2: Comparison of related work. In the "application domain" column, ITS stands for Intelligent Transportation Systems.

Table 2.2 summarizes the related work and the goal of this thesis.

Chapter 3

Lane-Level Highway Map Generation

This chapter describes our approach to building a map of road-lanes that appear on a given highway image. In this thesis, a road-lane (or lane) refers to the part of a road built for controlling and guiding a single line of vehicles. The output of this procedure is cartographic information about road-lanes in a set of pixel coordinates of road-lanes' centerlines and lateral road-widths, and orientations at those pixel coordinates. Such lane-level detailed highway maps with traffic rules and accurate coordinates can be prepared in advance to facilitate the guiding of autonomous and manual highway driving.

To extract such lane-level detailed information from a given image, pixels along road-lane boundaries must be visually and computationally accessible. To meet this requirement, we choose orthoimages with 15-centimeter ground resolution in which lane boundaries can be observed by the naked eye and can be potentially processed computationally.[1] Highways appearing in these images are inter-city (or arterial) highways built for facilitating transportation between cities [U.S. Department of Transportation, 2000]. Another reason we choose this particular pair of image data is that arterial highways on such high resolution aerial imagery pose sufficient challenges for extracting road-lane boundaries. The challenges concern the variation of objects' appearances and the complexity of the road geometry. In contrast, the road geometry of interstate highway images would be too monotonic with only slight photometric variation, and roads in a city pose too many complexities unrelated to the task of extracting road boundary lines (e.g., occlusions by frequent appearance of cars, pedestrians, other urban structures).

Since our target images are depicted in high-resolution, such image objects as lane-markings and road image-regions contain significant variations in their appearances, such that an object appears differently based on the condition of an image acquisition process.[2] This complicates identifying boundaries of road-lanes. For example, even in a given arterial highway image, road surfaces may be covered with different materials, such as asphalt or concrete. Such variation in road surfaces cause an inconsistency in color and texture of lane-markings and road-regions.

[1]Because the normal longitudinal pavement markings on highways are 4 - 12 inches wide (10 - 30.48 centimeters) [U.S. Department of Transportation, 2009], there are at least two pixels for laterally delineating a part of lane-markings.

[2]These factors include illumination conditions, the resolution, intrinsic and extrinsic parameters of the camera, the spectral sensitivity [Tonjes and Growe, 1998], the resolution of and the line of the sight between an acquisition-vehicle and the ground with respect to the location of sun.

Another example of appearance variation is occlusions caused by man-made structures such as buildings, over-hanging traffic signs, as well as overpasses and their shadows. These structures make parts of roads partially or completely unobservable. Discontinued road-lane boundaries make it hard to discern the true geometric shape of a road-lane. The geometry of arterial highways also makes it difficult to follow a lane's boundary. Ramps with circular paths have high curvatures that require a boundary following process that tracks non-linear paths. Road-lane junctions near an overpass require extra care due to the complex traffic directions. Road-boundary tracking must also be carefully done at a bifurcation point, where one splits into two, because one of the multiple tracking lanes might disappear.

To effectively tackle these challenges, we develop a hierarchical approach to three tasks: to gathering road boundary image cues, generating road-lane hypotheses, and linking the hypotheses. Our approach builds a map of road-lanes by linking road-lane hypotheses while satisfying the constraints of the underlying roads imposed by the collected image cues and prior information about the U.S. highways. To this end, first we scrutinize a given input image to harvest two types of image cues about the underlying roads: road image-regions and geometry. Knowledge of road image-regions are useful in specifying where to look for road-lane boundaries. To obtain the information of road image-regions, we formulate this image segmentation problem as a binary classification. In this classification, our method exploits a road-vector screenshot of the given image to obtain weakly-labeled examples of road-region superpixels and to train a probabilistic road-region detector. The binary classification outputs are then theoretically averaged out through a Markovian framework to produce a globally-coherent segmentation result.

Another important image cue we collect is the geometry of the underlying roads. To obtain this information, we extract lines and analyze the screenshot image of the road-vector to estimate the legitimate driving direction and to identify relevant road structures, such as overpasses. These collected image cues about road surface and geometry will provide strong evidences of the true road-lanes. In particular, these cues facilitate a road-lane hypothesis generation and guide a linking of these hypotheses to build a correct map of road-lanes. We call these collected image cues mid-level, task-specific features[3] because they are directly used to simplify, in three ways, our problem of building a map of road-lanes on a highway image: 1) the identified image road-regions are a good approximation of the unknown true road-lanes; 2) the hypotheses about true road-lanes are generated only from the identified image road-regions; and finally; 3) these cues are used to dictate how these hypotheses should be linked to one another. Figure 3.1 illustrates our approach of building up low-level image features to mid-level, task-specific features. At the lowest level, we analyze the inputs and prior information to extract task-relevant image features, such as lines and superpixels. And then we refine these low-level features to produce task-specific mid-level image features, including results of driving-direction estimation, results of road-region segmentation, and results of interesting road-structure detection. These mid-level

[3]From the theoretical perspective of computer vision, our low-level image features are a mixture of canonical low-level features (or primal sketch) and mid-level features (or $2\frac{1}{2}$-dimensional sketch) [Poggio, 1981],[Marr, 1982],[Palmer, 1999]. In this thesis, we divide into three categories any intermediate results from image processing tasks: 1) low-level features: image processing results that contain task-relevant information, 2) mid-level features: results of image processing that contain task-specific information, and 3) high-level features: image processing results that contain information directly related to a given task or can be directly used to achieve the goal of a given computer vision task.

features in turn are used to generate hypotheses about unknown true road-lanes and also guide the task of linking these hypotheses to recover individual road-lanes. For the problem of linking road-lane hypotheses, we formulate it as the min-cover problem [Vazirani, 2004]. We look for a set of hypotheses about the unknown true road-lanes to maximally cover the estimated road image-regions with a minimum sum of costs.

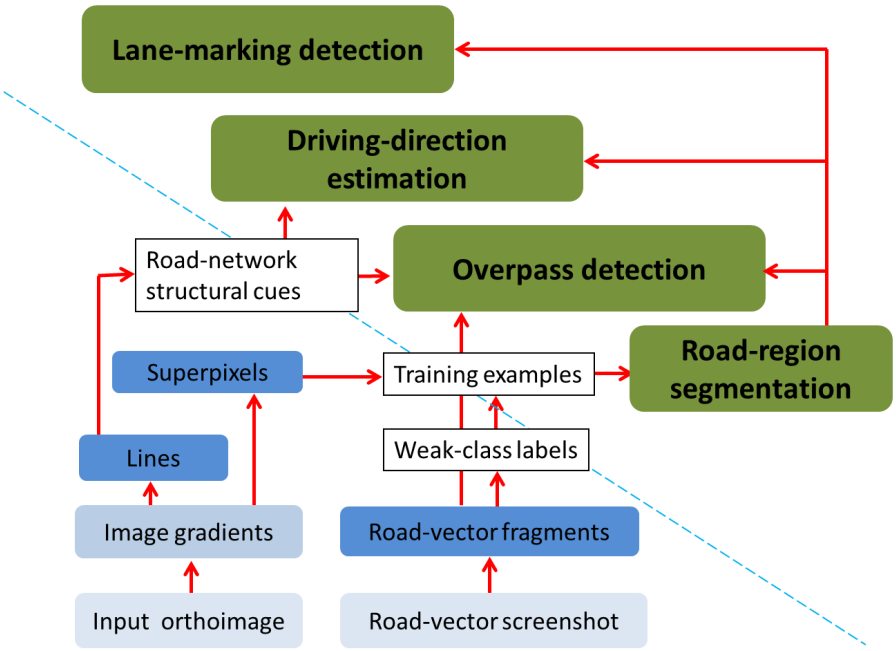
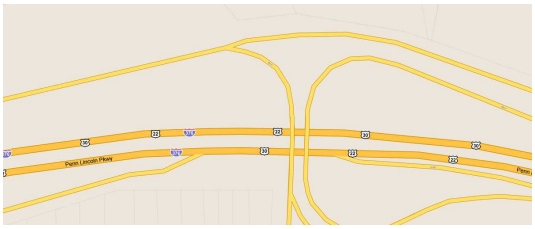


Figure 3.1: This diagram illustrates information flow between low-level and mid-level image features.

In what follows, we detail the methods of harvesting low-level features, the methods of converting these low-level features into meaningful mid-level features, and the methods of generating road-lane hypotheses and of linking them, so as to generate a map of road-lanes in a given image.

(a) The ground resolution of an input orthoimage is 15 centimeters per pixel. In such a high-resolution highway image, pixels along road-lane boundaries are perceptually and computationally accessible.



(b) A screenshot of the road-vector of the input image. This image depicts the underlying road-network with other cartographic information such as names of highways.

Figure 3.2: Our approach extracts image cues such that their photometric and geometric patterns are local, but sufficient to build a map of highways from two input images.

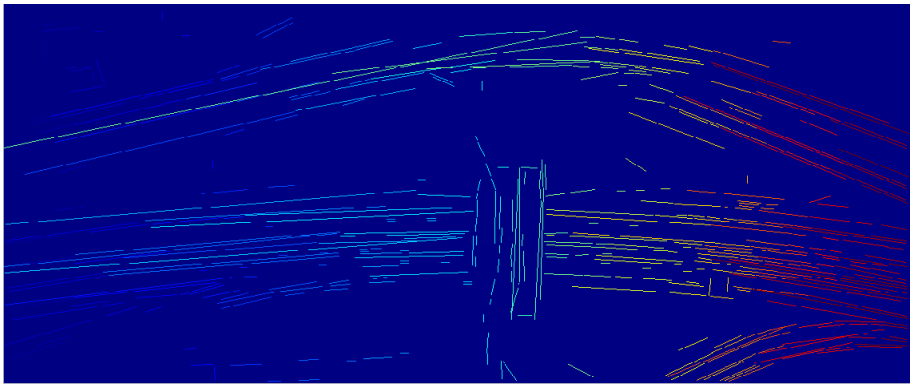
3.1 Harvesting Road-Boundary Image Cues via Bootstrapping

This section explains how to analyze input images to extract low-level image features and how to refine these features to produce mid-level image features that can be useful to execute other tasks. Our algorithms take two images as the input: A highway orthoimage and the input image's road-vector screenshot. Figure 3.2(a) shows an example of highway orthoimage and Figure 3.2(b) shows an example of road-vector screenshot image. A road-vector screenshot is a screencapture image that, with distinctive colors, depicts the underlying road-network of the highway scene. When a road-vector image is overlaid with an orthoimage, road-regions in the orthoimage are labeled with real-world cartographic information. One might think that the road-vector screenshot image would trivialize the extraction of boundaries of road-lanes appearing on an orthoimage. Yet such is not the case. First, the sketches (or drawings) of road-vectors are just parts of images, meaning that they do not possess any information about road-vectors, which are directly accessible in a computational form. To make these image sub-regions useful, they must be processed extensively and properly. Secondly, the road-vector sketches are not entirely overlapped with images of road-regions, resulting in cases where some road-regions remain uncovered. From image processing or pattern recognition perspectives, this is a very confusing signal. Some regions of a true road-lane image are marked as "road," while some other image regions very next to those regions labeled as "road" are indicated as "non-road." This holds for the opposite case as well – indicating non-road regions as road regions, e.g., a road-vector painting over trees or buildings. Thus, when a screenshot of a road-vector is used, an extra care should be taken.

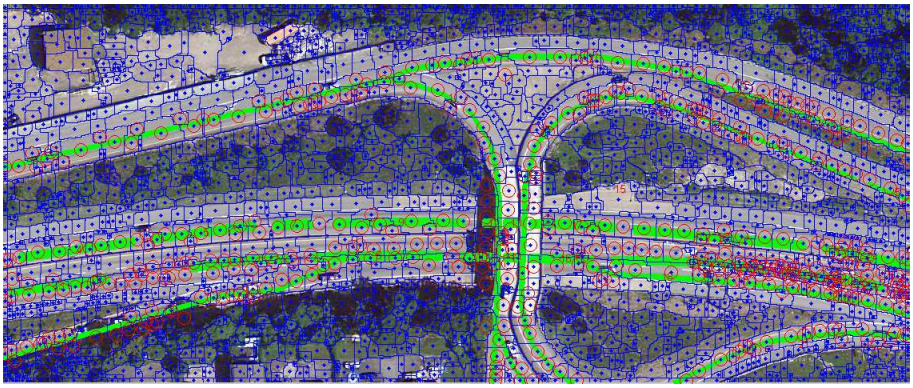
3.1.1 Extraction of Low-Level Image Features

We first parse two input images, to extract low-level features, such as image gradients, lines, and superpixels. For a line extraction, we first compute the image gradients and use the quantized

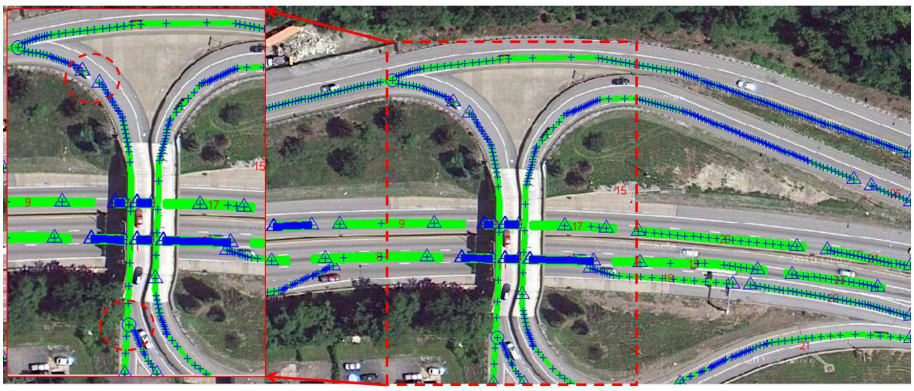
orientations of individual pixels to group them. For each of the pixel groups, we compute the eigen values and vectors about pixel coordinates in the group. This analysis provides us with two useful pieces of information about the pixel group's geometric shape: orientation and magnitude, which are used to produce a line for the group. In particular, the ratio of the largest eigen vectors' two components is used to approximate the orientation, $(\theta = [0, \pi])$, of a line. The Euclidean distance between the two farthest pixels in the group is used as the length of the resulting line [Kahn et al., 1990]. Figure 3.3(a) shows an example of extracted lines.



(a) An image of lines. Each of the extracted lines is depicted in a different color. Notice that these lines are mostly extracted from road image-regions.



(b) A superpixel image is shown. The elongated green polygons (or blobs) are fragments of a road-vector screenshot.



(c) An analysis of road-vector fragments is performed to obtain their geometric properties.

Figure 3.3: An intensive image analysis results in three task-relevant, low-level image features.

To obtain a superpixel image, we first smooth the input image using bilateral filter [Tomasi and Manduchi, 1998], compute gradient magnitudes on the filtered images, and then apply the watershed segmentation algorithm to the image of gradient magnitudes to obtain a coarse segmentation. We then reiterate this process until the dimensions of individual superpixels are large enough [Lalonde et al., 2010]. Particularly, we terminate this iteration when the current number of superpixels is smaller or equal to the predefined proportion (e.g., 15%) of the initial number of segments produced by the watershed algorithm. Figure 3.3(b) shows an example of a superpixel image.

To extract the useful geometric information of the underlying roads from a road-vector screenshot, we first identify image regions of road-vector sketches and produce a binary image. This contains only these fragments of road-vector without any map-symbols. We then further analyze each of the road-vector fragments, to obtain their geometric properties, such as extremity and bifurcation points. Because a road-vector fragment is a polygon bounded by closed path, the skeleton of a fragment is useful in acquiring these pieces of information. A skeleton of a polygon is a series of linear lines linking ridge points which are local extrema sitting in the middle of a polygonal shape. We apply a distance transform to each of the road-vector fragments and identify these ridges points. Section 5.3 details this process. Figure 3.3(c) shows a result of such analysis. Each (green) polygon represents road-vector fragments where "+" indicates a ridge point, "+" with a triangle is an extremity point, and "+" with a circle is a bifurcation point.

Since these low-level features contain only basic information about road-lanes appearing on the input image, we need to refine these features into features more relevant and useful in executing our task of analyzing highway geometry analysis task. These new features, we call task-specific mid-level image features, include an estimation of some legitimate driving directions of roads appearing on the input image, locations of interesting road-structures, such as intersections and overpasses; and segmentation of road image-regions.

3.1.2 Extraction of Mid-Level Image Features

Road-Region Segmentation Knowledge of road image regions would be very useful in that such knowledge could specify where to look for road-lane boundaries. Acquiring such knowledge is the task of image segmentation that divides an input image into more than one sub-regions. In our case, it is to divide a highway image into two sub-regions: road and non-road regions. We tackle this problem as a binary classification problem that takes superpixels as input and assigns each superpixel with one of two class labels: road or non-road. In a common classification task, a person assigns class labels to superpixels and prepares a set of superpixels and their class label pairs, so as to train a classifier. The number of training examples may vary, but can be roughly determined based on the dimensionality of data and the complexity of the problem.

In this thesis, we take a different approach to executing a binary classification. Instead of relying on numerous human-labeled examples, we utilize one of our inputs, a road-vector screenshot image, to prepare a self-labeled training data. In particular, we treat a superpixel as a positive example if its area is significantly overlapped (i.e., more than 90%) with road-vector paintings; otherwise we treat it as a negative example. Notice that the sketches (or drawings) of road-vectors are not entirely overlapped with image road-regions, resulting in some of road-region superpixels are missing. This means that some of superpixels, which should be treated as parts of positive

examples, are indicated as non road-regions. In addition, by such weak labellings, some non-road regions to be regarded as positive examples due to misalignment between an orthoimage and the underlying road network (e.g., road-vector painting over trees or buildings.)

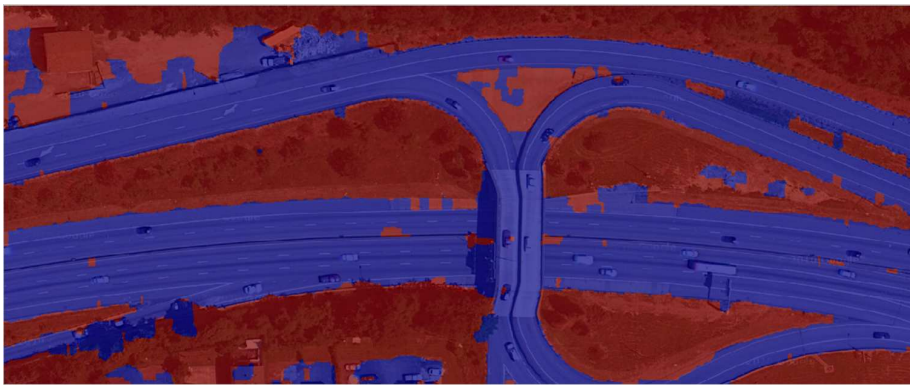


Figure 3.4: Results of road image region segmentation. The blue regions represent identified road image-regions and the red regions represent non-road image-regions. Although some non-road image-regions are labeled as road, most of the segmentation results correctly depict road image-regions.

To execute the superpixel classification, we first represent each of the superpixels as a feature vector. A feature vector consists of color and texture information. We use a histogram to represent color values in a superpixel and a texton [Martin et al., 2004] to represent texture value. To minimize the effect of superpixels' incorrect class assignments, we learn a probabilistic classifier, a Gaussian Mixture Model (GMM), and assign individual superpixels with class labels. To smooth out the outputs of the GMM, we run pairwise Markov Random Fields (MRF) and infer the most probable segmentation of the input image using loopy belief propagation. Figure 3.4 shows a result of image road-region segmentation. Results of the road-region segmentation define image regions of interest where all of the remaining tasks for building lane-level highway map have been executed.

Driving-Direction Estimation The goal of our task is to extract boundaries of individual roadlanes in the given image. This requires tracking boundary pixels of road-lanes that appear on the given image. Notice that the directions of these road-lane boundaries always align with the driving directions of the road-lanes. Thus knowing the driving direction at any given image locations is very useful for tracking road-lanes boundaries.

To approximate the driving direction from a given image, we use line extraction results that each of the extracted lines partially explain as the contour of roads in a given image. It is undesirable to approximate the driving direction at a pixel level because of all the noise that must be tackled. Instead we partition the input image into a number of grid cells. For each grid cell, we identify extracted lines which pass by it and use them to approximate the driving direction of the grid cell. Suppose there are n number of lines identified as passing the ith grid cell, $j \in [1, n]$. We compute the direction of a grid cell, i, by using the vector sum method, $\hat{\theta}_i = \arctan(y, x)$, where $x = \sum_j \cos(\theta_j)$ and $y = \sum_j \sin(\theta_j)$, where θ_j is the orientation of jth line. The orientation of a grid cell is mostly homogeneous to its neighboring cells, particularly in road image regions. To

enforce such a constraint, we run a MRF to infer the most probable driving direction of the input image as a whole. Our method of approximating driving direction is motivated by the method proposed in [Dolgov and Thrun, 2008] where the authors extract lines from laser-scan data and run an MRF to infer the driving direction homogeneous to a given parking lot image. They use a combination of Canny edge detection and Hough transform to extract lines from a laser scan image. We also tried this combination and found the extracted lines were too short to use. Figure 3.5 shows a result of driving-direction estimation.

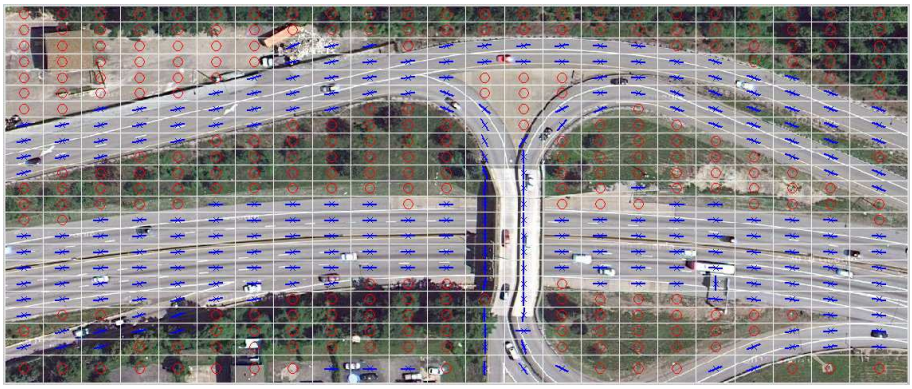


Figure 3.5: Results of driving direction estimation. The blue lines indicate the estimated driving direction of a grid cell and the non-drivable regions are depicted by red circles.

Lane-Marking Detection Lane-markings are one of the most important photometric cues for extracting road-lane boundaries. In fact, a perfect lane-marking classification on a given orthoimage would make it easier to accomplish our goal. Indeed, the results of such classification would provide us with crucial information about road-lane boundaries such as their exact locations in a given image.

Lane-markings are a type of road-marking that depicts boundaries of roadlanes. On an orthoimage, we can, readily with the naked eye, distinguish lane-markings because they have whitish colors, relatively higher intensity than their neighboring pixels, and occupy approximately known locations. However, these salient features are not always available for image processing because the actual values of lane-marking pixels vary based on image acquisition conditions.

To effectively meet the challenge of appearance variation of lane-marking pixels', we formulate the lane-marking detection task as a binary classification problem of discriminating non lane-marking pixels from true lane-marking pixels. Such formulation enables us to utilize a combination of well-established feature representation and classification method that effectively handles the appearance variation. To this end, we first label some highway images, for marking positive and negative examples, convert these example pixels into features, and then learn a binary classification model of lane-marking pixels' photometric variations. Note that this is the only place we use manually labeled data for training a part of our system.

For the feature representation, we want to convert the original color intensity pixels, $\mathbf{x} \in R^p$, into something different, $\mathbf{x} \in R^q, p \neq q$, in which we can better discern computationally the characteristic of lane-marking pixels. To this end, instead of directly changing the intensity value, we

look into the contrast of intensity values between a lane-marking pixel and its neighboring pixels. In fact, we use the local binary pattern (LBP) [Ojala et al., 2002] and four different statistics about texture to generate a feature vector of the pixel.

The LBP is proposed to express a spatial pattern of a pixel's neighboring pixel values as a binary bit vector. It looks into the difference of pixel values (e.g., gray-scale or a color channel value) and mark down the difference as $I(g_j - g_c)2^j$, where $I(\cdot)$ is an indicator function, g_j is the jth neighbor's pixel value, $j \in [1, k]$, and g_c is the target pixel value. There are two parameters that control the range of the neighbors (or the size of the local image patch): the number of neighboring pixels, P, equally spaced on a circle of radius R. For example, when $R = 1$ and $P = 8$, the neighbor of a pixel coordinate, (x, y), is equivalent to the second order Markovian, starting from $(x+1, y), (x+1, y-1), (x, y-1), ...$ to $(x+1, y+1)$. The contrast value between the target pixel and each of the neighboring pixels is saved to a binary bit vector where the least significant bit is assigned to the contrast value of the right neighbor, $(x + 1, y)$. The LBP of a lane-marking pixel is then computed by

$$LBP_{P,R} = \sum_{p=0}^{P-1} I(g_j - g_c)2^p$$

Because a single step rotation of the final binary bit vector could generate a completely different pattern, a circular bit-wise (right) shift needs to be applied before finalizing the feature conversion [Ojala et al., 2002]. In addition, we also compute four different statistics of the pixel and its neighboring pixels, such as smoothness (t_1), $1 - \frac{1}{1+\sigma^2}$, entropy (t_2), $-\sum_{l=1}^{L} P(z_i) \log_2 P(z_i)$, uniformity ($t_3$), $\sum_{l=1}^{L-1} P^2(z_i)$, and the variance ($t_4$) of image gradients' magnitudes on the neighboring pixels where σ is variance of pixel values, L is the maximum value, z_i is the ith value. For example, L will be 255 if we measure the intensity value, $z_i \in [0, 2^8 - 1]$. Our feature representation converts a lane-marking color pixel, $\mathbf{x} \in R^3$ into a feature vector, $\hat{\mathbf{x}} =< LBP_{P,R}, t_1, ..., t_4 > \in R^6$.

We tried several different combinations of these features and classification methods to find a best one for our lane-marking detection task. We downloaded 20 orthoimages separated from the images for generating lane-level highway map and collected 47,640 pixels. These consisted of 15,204 lane-marking (positive) pixels and 32,436 non lane-marking (negative) pixels.

We set aside a portion (about 30%, 14,292) of the labeled pixel data as testing data and use the rest of them to train a classifier. We tried six different classification setups and found that the AdaBoost outperformed all others – AdaBoost with a feature representation without color information produced 0.98 precision and 0.97 recall rates on average. Table 3.1 shows detailed information about a performance comparison between different classification setups. We ran five different tests that required random sampling of positive and negative training examples. Due to this randomness, we averaged our results over five separate runs. Each cell in the table displays the mean and standard deviation. Figure 3.6 shows a result of lane-marking detection.

Interesting Road-Structure Detection To completely determine a road-lane's boundaries, it is necessary to recognize road-structures that may indicate complex road geometries and may also occlude boundary lines. These road-structures include overpasses, over-hanging traffic signs, and trees. Because of their relative sizes in a given image, shadows of traffic signs and trees cause little occlusion on roads. However, shadows of overpasses impose serious occlusions

	Precision	Recall	False Positive	False Negative	Accuracy
SVM w/o color	0.9769 (0.0020)	0.9733 (0.0033)	0.0702 (0.0966)	0.0116 (0.0010)	0.9833 (0.0014)
AdaBoost w/o color	**0.9888 (0.0018)**	0.9781 (0.0025)	**0.0025 (0.0218)**	**0.0055 (0.0009)**	0.9889 (0.0010)
AdaBoost w/ RGB	0.9786 (0.0309)	0.9850 (0.0155)	0.0149 (0.0155)	0.0123 (0.0164)	0.9867 (0.0098)
AdaBoost w/ HSV	0.9854 (0.0140)	**0.9944 (0.0180)**	0.0155 (0.0180)	0.0074 (0.00724)	0.9898 (0.0075)
AdaBoost w/ YCbCr	0.9661 (0.0343)	0.9942 (0.0039)	0.0057 (0.0039)	0.0180 (0.0186)	0.9868 (0.0118)
AdaBoost w/ Lab	0.9841 (0.0125)	0.9927 (0.0072)	0.0052 (0.0028)	0.0081 (0.0065)	**0.9928 (0.0049)**

Table 3.1: Performance comparison of different lane-marking classification methods. The numbers in bold faces are the best for the corresponding column.

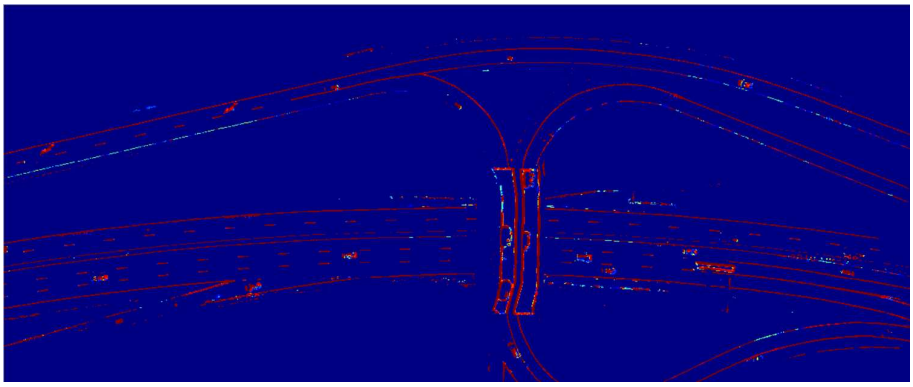
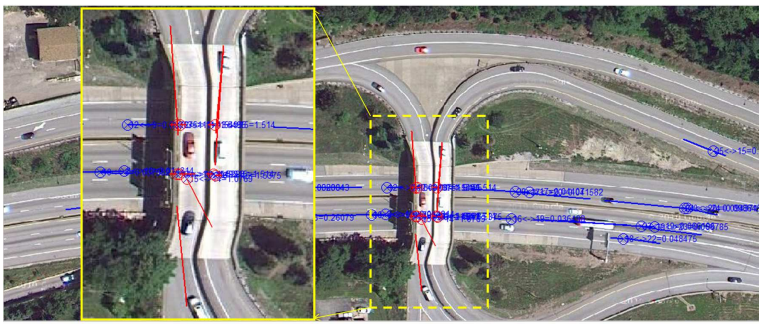


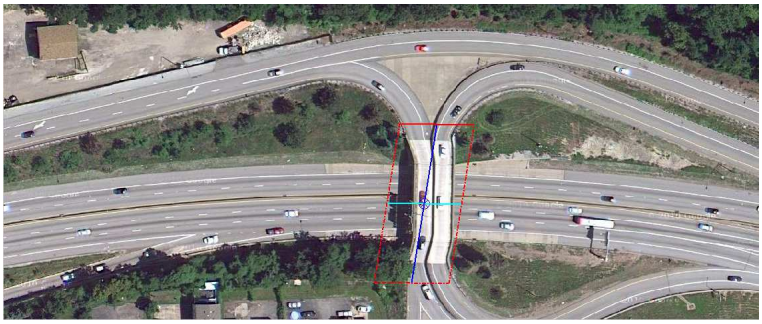
Figure 3.6: Because the outputs of our lane-marking classification are probabilistic, the results are shown in a heat-image where the color closest to red represents the highest probability.

over road-regions in that unobserved road-regions are not insignificant for completely delineating road boundary contours. In addition, the appearance of overpasses also increases the complexity of road geometry because multiple roads pass each other orthogonally in the same image region. In this thesis, as overpass structures are frequently observed on arterial highways, we want to detect them, so we can properly handle the geometry and occlusions around any detected overpass.

The input of the overpass detection algorithm is the road-vector screenshot. As described earlier, the road-vector screenshot image is analyzed and converted into a set of road-vector fragments. Each of the road-vector fragments contains the geometric characteristic of parts of the underlying roads. For each of the road-vector fragments, we extend each of the extremity points in the direction of the fragment and identify any intersection with other fragments if their intersection angle is greater than a given threshold (e.g., $\pi/3$). Figure 3.7(a) shows a result of overpass localization where a multiple of two (red) intersection lines indicate potential overpass regions. The final process of detecting overpasses is to identify the boundary of a detected overpass. To this end, we search for any of the closest extracted lines that intersect with any of the two lines from the overpass localization and are greater than the same threshold used



(a) Results of overpass localization. Each two intersected red lines indicates a potential location of an overpass. A blue line is a line extended from a road-vector fragment.



(b) Results of overpass detection. A red trapezoidal polygon represents the boundary of the detected overpass, and two (blue and cyan) lines inside the polygon depict two principal axes.

Figure 3.7: These figures show the sequence of overpass detection.

earlier. Figure 3.7(b) shows the final result of overpass detection. The bounding box of a detected overpass lets other tasks of extracting lane-level highway information know of the existence of an overpass and that the road geometry around this bounding box has more than one direction.

We describe how we obtain four mid-level image features. Aside from the lane-marking detection in which we used some human labeled data to train a lane-marking classifier, we obtain, without human intervention, three other important cues – road-region segmentation, driving direction estimation, and overpass detection. These are obtained by extensively analyzing what is available on the input image. Although we obtained these cues from analysis of an image's local photometric characteristic, their properties are global and strong enough to coherently guide the following high-level vision tasks such as generating of hypotheses about true road-lanes and connecting hypotheses in that:

- Results of road-region segmentation specify image sub-regions where the true road-lanes most probably appear.
- Lane-marking detection results narrow down further the image sub-regions defined by

road-region segmentation and enable the process of road-segment hypothesis generation to pick up relevant image location more precisely.

- Results of driving direction estimation inform the hypothesis-linking process of how the generated road-segment hypotheses should be linked. In contrast, the detected overpasses define an image sub-region where individual hypotheses near this region should be carefully linked to one another.

In the next section we detail how these four mid-level features are used to generate road-segment hypotheses and how to link them to build lane-level detail highway maps.

3.2 Generating Hypotheses about Road Lanes

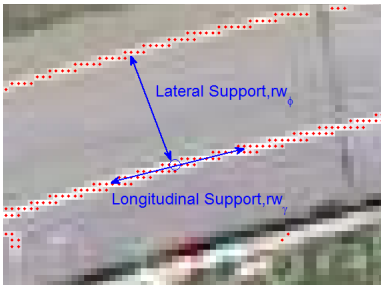
From the previous steps of extracting low-level and mid-level image features, we now have a better understanding of the input image. In particular, we know which image sub-regions are most probably road-regions, which pixels within the road-regions are likely parts of lane-markings, how the roads are laid out, and where overpass structures occur. Based on this understanding, we are generating road-lane hypotheses and linking them, in order to delineate road-lane boundaries.

3.2.1 Road-Width Hypotheses as Cues for Road-Lane Hypotheses

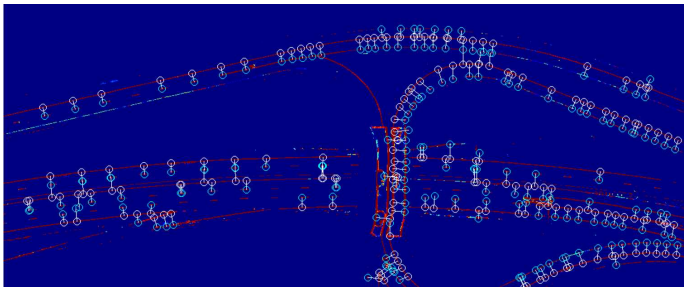
As mentioned earlier, a road-lane is modeled by a piecewise linear curve that comprises multiple control points and their properties, such as lateral width and orientation. Thus, generating a hypothesis about a true road-lane would be equivalent to identifying these (control) points' locations. However, given that the boundary location of road-lanes are unknown, it is difficult to localize the centerlines of road-lanes. Instead we investigate pixels of lane-marking detection results (or lane-marking pixels). No lane-marking pixels along the true centerlines of road-lanes are available, but one can interpolate the centerline locations from a set of regularly-spaced lane-marking pixels.

As discussed earlier while describing lane-marking detection, a true lane-marking pixel has many neighboring lane-marking pixels regularly-spaced longitudinally and laterally (or orthogonal to the longitudinal direction). Because two true lane-markings located laterally at each other's side can be used to accurately measure the width of the road at a location, we are looking for lane-marking pixels that have strong supportive (or neighboring) patterns in longitudinal and lateral directions of the roads. The likelihood of a lane-marking pixel being a good road-width cue is measured by two scores capturing these neighboring supports. Figure 3.8(a) illustrates an example of lateral and longitudinal supports for a lane-marking pixel.

While searching for lane-marking pixel candidates, we can also utilize our prior knowledge of the actual road-width of a normal highway. For example, according to a governmental guideline [U.S. Department of Transportation, 2007], the minimum width of a highway lane is 12 feet (3.7 meters). Because the ground resolution of our test image is known, we can remove any pairs of lane-marking pixels that have lateral support (i.e., distance measured orthogonally to the estimated driving direction) shorter than 24 pixels (24 pixels $\times$ 15 cm/pixel = 3.75 meters) or longer than any maximum values. However, care must be taken before incorporating such prior knowledge because road-widths vary – on arterial highway images, some of the road-lanes may



(a) A lane-marking pixel has its neighboring lane-marking pixels along the longitudinal direction and across the lateral direction.



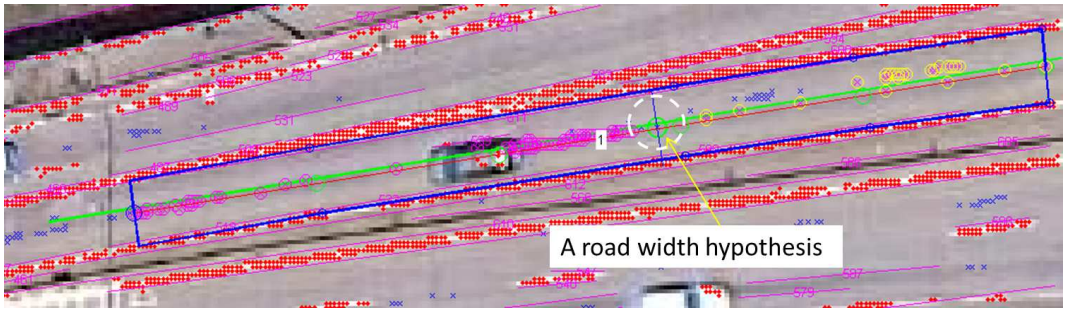
(b) Resulting Road Width Cues. A dumbbell-like symbol is a road-width cue where the two circles at the ends of a line indicate lane-marking pixel locations with strong longitudinal and lateral neighboring lane-markings.

Figure 3.8: These figures show the process of searching for road-width image cues.

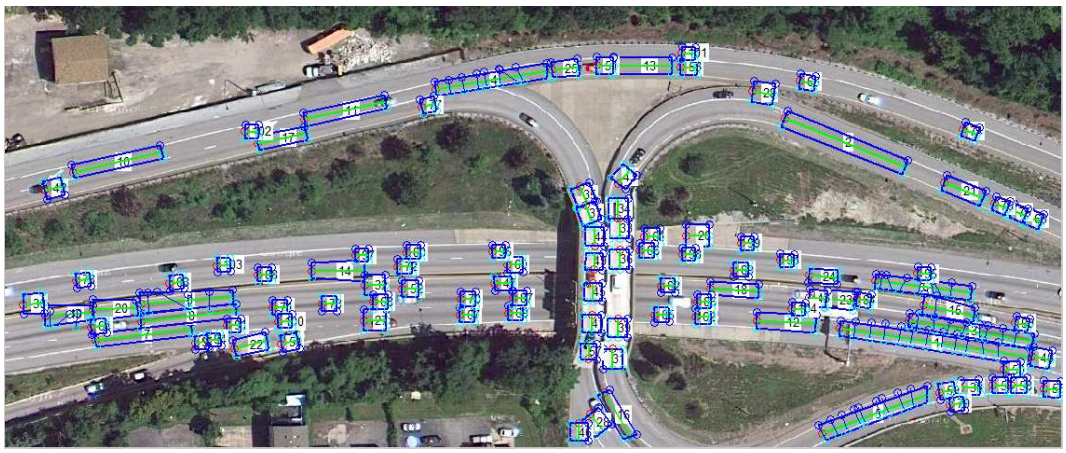
have wider or shorter lateral lengths. For example some ramps have relatively shorter widths where they merge and there they enter. We empirically found that 22 to 35 pixels worked best for the variation in road-widths.

From road-region segmentation results and lane-marking detection results, we already have a good sense of which image sub-regions are likely to be parts of roads and which part of estimated road-regions are probably lane-markings. To make the search of these road-width cues efficient, we begin with superpixels that belong to the segmented road-regions. For each superpixel, we investigate whether each of the lane-marking pixels has a sufficient number of neighboring lane-marking pixels in longitudinal and lateral directions on the roads. Any lane-markings with more than the predefined threshold remain in the candidate list for generating road-lane hypotheses. Figure 3.8(b) shows the results of a road-width cue search. A road-width hypothesis is represented by a pair of a numeral width value and two lane-marking pixels.

After we find a set of road-width cues, the next step is to generate a set of road-lane hypotheses. This process is executed in a similar manner to that of the road-width cue search. For each road-width cue (or road-width hypothesis), we draw two lines from the center of the two lane-marking locations in the longitudinal direction and group together any road-width cues within extending line segments. This forms a road-lane hypothesis. The longitudinal direction corresponds to the driving direction estimated earlier from extracted lines. Figure 3.9(a) shows an example of a road-lane hypothesis. A green circle with a white dashed circle represents an input road-width hypothesis and the green line defines the range of a road-lane hypothesis search. This search results in grouping the neighboring road-width cues, depicted by magenta and yellow circles, around the input road-width hypothesis. A blue rectangle depicts the boundary of a road-lane hypothesis which is obtained by connecting two lane-marking pixels of all road-width hypotheses. Figure 3.9(b) shows a set of resulting road-lane hypotheses.



(a) A search around an input road-width hypothesis is executed to generate a road-lane hypothesis. The resulting road-lane hypothesis covers a part of a true road-lane.



(b) The road-lane hypothesis generation process produces 99 hypotheses about the 10 true road-lanes.

Figure 3.9: These figures show the results of the road-lane hypothesis generation process. Most of the resulting hypotheses are found from true road-lanes.

3.2.2 Connecting Road Segments for Delineating Road Boundary

By searching for road-width image cues and linking the identified cues, we generate a set of road-lane hypotheses. To extract boundary lines of true road-lanes, we need to link road-lane hypotheses together. We formulate the problem of linking hypotheses as the min-cover problem in which we search for a set of road-lane hypotheses to maximally cover the estimated road regions with a minimum sum of costs.

Suppose that there is n number of the true road-lanes, $R_1, ..., R_n$, in an input image. Let H_r denote a hypothesis about a true road-lane, $r \in R$. The hypothesis, H_r, is a noisy estimate, based on collected image cues, of a part of a true road-lane in the input image.

A road-lane hypothesis, H_r, is represented as a piecewise linear curve (or a polyline) that consists of m number of vertices (or control points), $V = (v_{r,1}, ..., v_{r,m})$ and $|E| = m - 1$ edges linking two adjacent vertices. Each of the vertices has three properties: $v_{r,j} =< l_{r,j}, w_{r,j}, \theta_{r,j} >^T$,

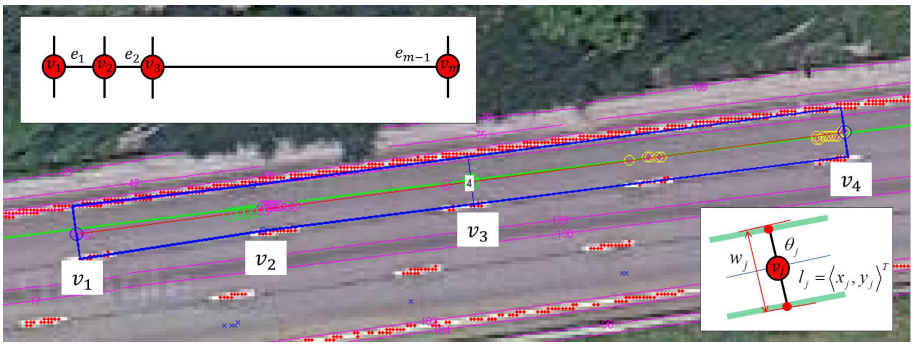


Figure 3.10: This figure illustrates a road-lane hypothesis that consists of four vertices. The blue rectangle represents the boundary of the hypothesis. In this example, to obtain the boundary of a road-lane hypothesis, for each of four vertices, we first draw a line segment with a length of the lateral width in an orthogonal direction to the driving direction of the road-lane; we then collect two extremity points from individual line segments; we finally connect them either in a clockwise (or counterclockwise) fashion.

where $l_{r,j} = (x_{r,j}, y_{r,j})$ is a 2-dimensional location (or pixel coordinates), $w_{r,j}$ is the (lateral) width at the given location, and $\theta_{r,j} \in [0, \pi]$, is the orientation. In fact a vertex corresponds to a road-width hypothesis described in the previous section. Figure 3.10 shows an example of a road-lane hypothesis consisting of four vertices and three edges. Figure 3.9(b) shows a set of the generated road-lane hypotheses.

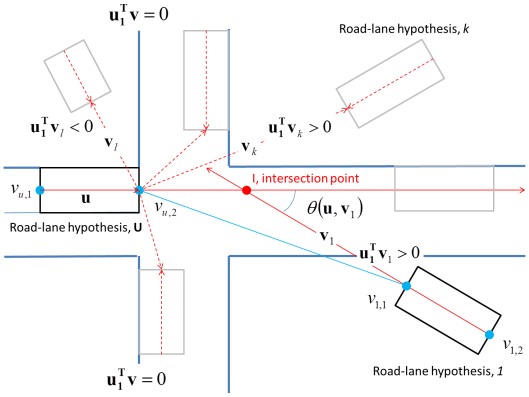
We are looking for a new set of road-lane hypotheses, $X = \{L_1, ..., L_k\}$, which link the generated road-lane hypotheses based on the previously obtained local evidences of the unknown true road-lanes with a minimum sum of linking costs. While linking road-lane hypotheses, the new set of road-lane hypotheses should maximally cover the estimated road image-regions.

$$\begin{aligned} X^* &= \arg\min_X Cost(X) \\ Cost(X) &= \sum_{L_i \in X} C(L_i) \end{aligned}$$

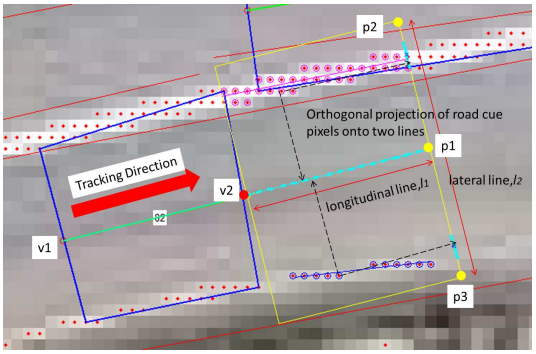
where $C(L_i)$ is a cost of linking between any two road-lane hypotheses, H_r and H_s. Our formulation is motivated by two previous studies [Felzenszwalb and McAllester, 2006], [Zhu and Mordohai, 2009]. For our cases, we generate a set of hypotheses about unknown true road-lanes to cover approximated true road image regions, whereas, for their cases, they generate hypotheses to delineate object contours [Felzenszwalb and McAllester, 2006] and to cover road regions in a LIDAR intensity image [Zhu and Mordohai, 2009]. Ours differs from their approaches in that we search for potential links based on the collected geometric cues and we identify the most probable link between two hypotheses by tracking photometric image cues. In contrast, they seek a series of sequential hypotheses.

To find approximate solutions to these cost functions, we devise two different linking functions. While linking these hypotheses, we must be careful not to link any that are not on the same true road-lane or if a potential link between two hypotheses fails to comply with any local geometric and photometric constraints. The first linking function considers a potential connection

between any two hypotheses purely following geometric constraints. And the second function investigates any photometric constraints of a potential link.



(a) The collected image cues and prior information about arterial highways imposes geometric constraints on finding a potential linking of any two road-lane hypotheses.



(b) The obtained photometric image cues provide strong evidence of potential links among the generated road-lane hypotheses. A tracking of photometric cues among any potential, geometrically plausible, links is conducted before actual linkings occur. Red dots represent lane-marking pixels and a blue rectangle represents a road-lane hypothesis.

Figure 3.11: These figures illustrate how two linking functions find the best potential links among road-lane hypotheses.

While implementing the first linking function, we refer to the geometry of actual arterial highways where the geometric shape of the road is highly correlated with highways' speed limits. In other words, it is easy to observe a low curvature road-shape on highways due to its higher speed limits. Another piece of useful knowledge for linking hypotheses based on geometric constraints is to observe driving direction between two road-lane hypotheses. It is highly unlikely for any two hypotheses to be linked to each other when a path of homogeneous driving direction is absent. Figure 3.11(a) illustrates an example of geometry-based hypotheses linking, where a road-lane hypothesis, $\mathbf{u}$, is searching for a good candidate hypothesis with which to link. Due to the fact that our target roads are arterial highways, any hypotheses located behind an input hypothesis should be discarded. We compute one-to-many dot-products between an input hypothesis and all remaining hypotheses. We do this to filter out any hypotheses located behind the input hypothesis. In the example shown in Figure 3.11(a), the hypothesis, $\mathbf{v}_l$, is removed from the candidate list, $\mathbf{v}_1$ and $\mathbf{v}_k$, remain in the candidate list for further consideration. For each of the hypotheses in the candidate list, we compute the value of geometric linking potential:

$$\mathbf{v}^* = \arg\min_{\mathbf{v}} G(\mathbf{u},\mathbf{v}), \quad \text{where,}$$

$$G(\mathbf{u},\mathbf{v}) = \sum_{j=1}^{k} g_j(\mathbf{u},\mathbf{v})$$

where $g_j(\mathbf{u}, \mathbf{v})$ is a function computing the jth geometric property between the two hypotheses, $\mathbf{u}$ and $\mathbf{v}$. The geometric properties of a link are three: curvature, $g_1(\mathbf{u}, \mathbf{v})$, intersection angle, $g_2(\mathbf{u}, \mathbf{v})$, and Euclidean distance, $g_3(\mathbf{u}, \mathbf{v})$. The curvature at the intersection point I can be computed by using two linked points the vertex v_2 of $\mathbf{u}_1$ and v_1 of $\mathbf{v}_1$ are computed by

$$\kappa_I = \frac{\begin{vmatrix} v_2 - I \\ v_1 - v_2 \end{vmatrix}}{|v_2 - I||v_1 - v_2||v_1 - I|}$$

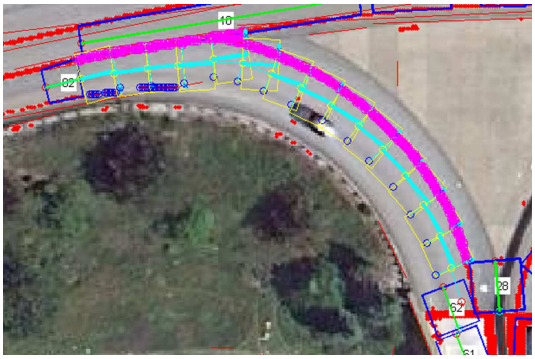
The second linking function investigates whether sufficient image cues are present (i.e., lane-marking pixels) on a potential path linking any two hypotheses. A potential path can be thought of as a piecewise linear curve along the centerline of a true road-lane just like our road-lane model where each line segment of the curve meets another line segment at a control point (or a vertex between edges). We assume that an optimal link always exists between two consecutive control points that maximizes photometric constraints around the link. The second linking function thus searches for the locally optimal link between two vertices along a potential path between two road-lane hypotheses. A connection between all these locally optimal links would result in an optimal approximation of a true road-lane. The incremental examination of consecutive links will be terminated when the next potential move intersects with either another road-lane hypothesis or one of the image bounds. While tracking the locally optimal path, the direction of tracking is initially set to the direction of the hypothesis, but after the initial step, the direction is adjusted by looking at the estimated driving direction.

Figure 3.11(b) illustrates such tracking of road-lane boundary cues. In this example, the tracking is about to begin at the vertex ($v2$) of a road-lane hypothesis and search for a locally optimal link for the next point. Currently, it examines one of the possible links to a point, $p1$, within the yellow rectangle where lane-marking pixels on the left side of the tracking direction are marked with magenta circles and lane-marking pixels on the right side are marked with blue circles. We use two line segments to collect road boundary cues: a longitudinal line, $l_1 = p1 - v2$ and a lateral line, $l_2 = p2 - p3$. We first project all lane-marking (magenta and blue) pixels onto these two lines. Let us denote $p(b)$ as a projected point of a lane-marking pixel p. The projected point $p(b)$ on line l_1, for example, can be expressed as $p(b) = v2 + b(p1 - v2)$, where $b = \frac{(P-v2)^T(p1-v2)}{(p1-v2)^T(p1-v2)}$. $p(b)$ is projected on the line segment l_1 if it satisfies $b \in [0, 1]$. Using these projected points, the second linking function evaluates the quality of a potential link to the next control point (e.g., a line segment between $v2$ and $p1$).

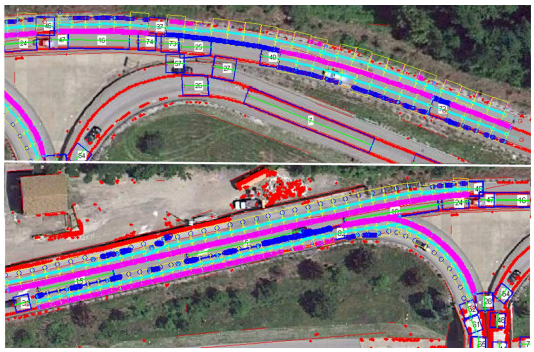
$$\begin{aligned} Q(P) &= \alpha q_1(P) + (1 - \alpha) q_2(P), \{p_i = (x_i, y_i) \in P\}_{i=1,\ldots,m} \\ q_1(P) &= \frac{\sum_{i=1} I\left(p_i \in \mathrm{Bin}_j\right)}{K}, \mathrm{Bin}_{j=1,\ldots,K} \\ q_2(P) &= 1 - \frac{|\max(b) - \min(b)|}{\mathrm{roadwidth}/2} \end{aligned}$$

where P is a set of the projected points, α is a variable that controls the contributions of two quality function values, "Bin" is a quantized histogram between 0 and 1 about b and "roadwidth" is an estimated quantity that is initially set to the average of lateral lengths of vertices in a road-lane hypothesis (e.g., the mean value of lateral lengths of $v1$ and $v2$). The function q_1 measures how

widely dispersed the points projected onto the longitudinal line l_1 are. By contrast, the function q_2 measures how closely dispersed the points projected on lateral line l_2 are. In general, the optimal link has a wide spread projection on the longitudinal line and a narrow spread projection on the lateral line. Figure 3.12(a) shows the result of a photometric road-lane boundary cue tracking and Figure 3.12(b) shows two other results of tracking long paths among road-lane hypotheses. This linking function based on tracking is similar to work that traces road image cues to extract road-networks from low-resolution aerial images. In particular, Zhou et al. use for their road cue tracking an extended Kalman filter [Zhou et al., 2006] and Movaghati and his colleagues utilize an unscented Kalman filter [Movaghati and Moghaddamjoo, 2008]. The primary difference is the ground resolution of testing images. Most of the variations in object appearances, imperative to analyzing high-resolution orthoimages, fail to appear in low-resolution aerial images.



(a) The result of photometric cue tracking illustrated in Figure 3.11(b). The road-lane hypothesis labeled 82 was successfully, through a high-curvature path, linked to another road-lane hypothesis labeled 62 .



(b) The obtained photometric image cues provide strong evidence of potential links among the generated road-lane hypotheses. A tracking of photometric cues among any potential, geometrically plausible links is conducted before actual linkings occur. Red dots represent lane-marking pixels and blue rectangles represent the generated road-lane hypotheses.

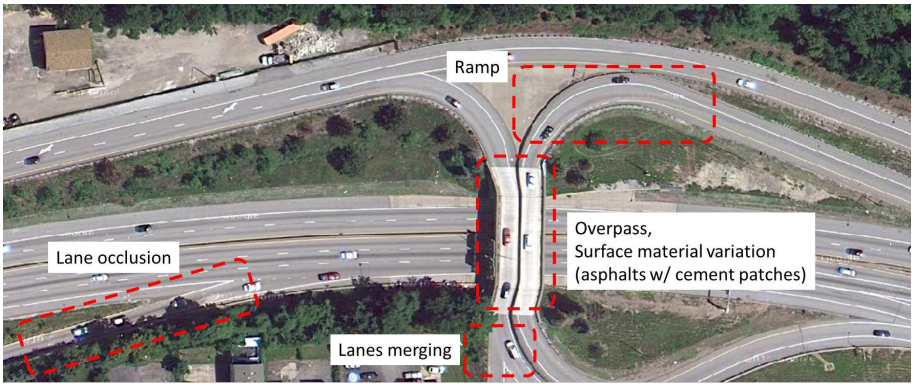
Figure 3.12: These figures show examples of photometric road-lane boundary cue tracking. These figures also demonstrate that our tracker is able to follow a high-curvature paths and long paths between two road-lane hypotheses.

In summary, the linking function based on local geometric constraints searches for the potential links that maximally satisfy geometric cues. These geometric cues are obtained from the mid-level image cues and prior information about the U.S. arterial highways. The link function based on photometric constraints searches for a potential link that maximally complies with the spatial patterns of the detected lane-marking pixels. The best link between two road-lane hypotheses would be one that locally minimizes these two constraint functions. Unlike previous work of the min-cover algorithm applications [Felzenszwalb and McAllester, 2006],[Zhu and Mordohai, 2009], where their solutions are explicitly searching for a sequence of hypotheses, we look for a set of hypothesis pairs such that

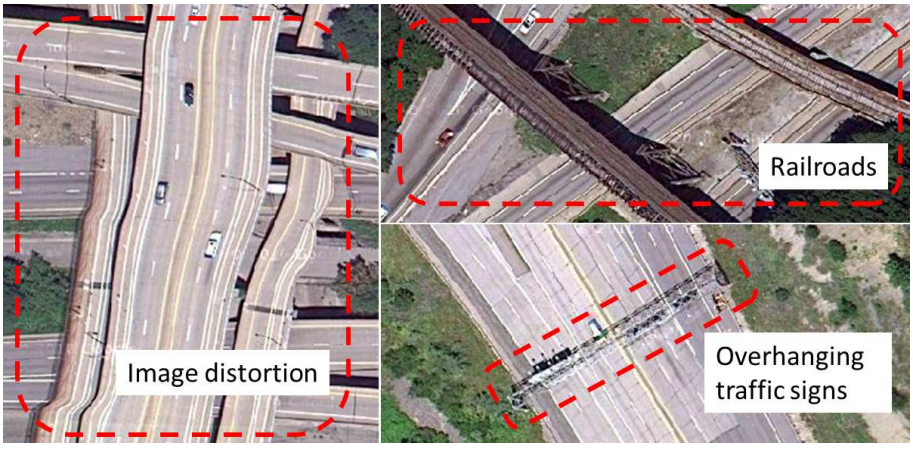
their potential, geometrically plausible, links are sequentially traced by photometric image cues to cover road image-regions.

3.3 Experiments

This section details experiments conducted to investigate the robustness of our approach to extracting a lane-level highway map and the accuracy of the resulting maps. In what follows, we first explain the experimental setup and evaluation methods, then show experimental results, and finally discuss the findings.



(a) Examples of “ramp,” “lanes merging,” “overpass,” “road surface material variation,” are shown.



(b) Examples of “image distortion,” “railroads,” “overhanging traffic signs” are shown.

Figure 3.13: Examples of complexity factors for measuring test image characteristics.

3.3.1 Experimental Settings

From Google's map service[4], we collected 50 orthoimages that are sampled from the route between the Squirrel Hill Tunnel to the Pittsburgh International Airport. We also saved road-vector screenshots of the orthoimages and manually drew boundary lines of individual roadlanes in each of the collected images for the ground truth.

The collected images are of arterial highway scenes. We believe this level of highway image contains a sufficient level of difficulty, which we would have to overcome if we increased the number of test images. To validate our assumption about the complexity of our image collection, we scrutinized each of our highway images. Figure 3.13 shows complexity factors used for this analysis. Table 3.2 summarizes the result of this analysis in terms of the geometric and photometric characteristics of our test images. We surveyed our images considering eight factors, each of which indicated the complexity of an orthoimage from the perspective of extracting boundary lines of road-lanes. For example, the presence of a "ramp" and "overpass" may cause the process of linking road-lane hypotheses to track a complex (or non-linear) road geometry. Note that we are concerned with only ramps with high curvatures. On 18 out of 50 images 23 ramps appear. In addition, when two lanes merge, one of the tracked lanes must, to produce a correct road geometry, disappear. From 27 images, we observed 39 lane-mergings. Unusual photometric variations on highway images would prevent our approach from extracting, at a desirable level of quality, a sufficient amount of low- or mid-level image features, resulting in incomplete road boundary lines. "Material variations" indicate a variation in road-surface materials (e.g., asphalt with concrete patches). Such material variation was observed in more than half the test images. "Urban Structures" refer to any man-made structures, such as over-hanging signs or railroads crossing, orthogonal to the highway driving direction. "Image distortion" indicates whether an orthoimage has any ortho-rectification errors by a map company, such as uncanny surface warping or unrealistic 3-dimensional surface reconstructions. From the statistics found in Table 3.2, we can rest assured that our testing image collection poses sufficient difficulty to our highway image analysis algorithms regarding photometric image variations and the complexity of road geometry.

We have a list of methods that require optimal parameters for producing desirable results. While extracting lines, we remove any lines the lengths of which are greater than half of the input image width (e.g., 600 pixels). We remove them because such long lines usually fail to align with any highway contours, resulting in incorrect driving direction estimation. In executing road-region segmentation, we apply a Leung-Malik filter bank [Leung and Malik, 2001], a multi-scale and multi-orientation filter bank consisting of 48 filters, to hand-labeled highway images, which are used for training a lane-marking classifier, and obtain 64 different textons to represent each superpixel. We found our road-region segmentation method produced the best results when β is set to 0.2. β is a parameter of the MRF that controls interactions between neighboring superpixels. For driving direction estimation, the size of each grid cell was determined by dividing the input image width and height by the diagonal length (i.e., 23.3 pixels) of a normal vehicle dimension (e.g., width $\times$ length = 12×20 pixels) estimated from 15 centimeter per pixel ground resolution. This allows a grid cell to contain a normal sized vehicle, resulting in reasonable estimation of driving direction. For lane-marking detection, we use AdaBoost and found that 50

[4] `http://maps.google.com`

Geometric				Photometric			
Ramp	Overpass	Lane merging	Number of Lanes	Material variation	Lane occlusion	Urban Structures	Image distortion
18 (23)	18 (33)	27 (39)	437	33	28	24	8

Table 3.2: Summary of testing highway images' characteristics.

decision stumps (i.e., weak learners) produced the best lane-marking detection results. We also used a logistic regression to convert the discrete output of AdaBoost into a probabilistic output. For overpass detection, we set the angle threshold at $\pi/3$ so as to detect greater intersection angles between road-vector fragments. To execute the linking function based on local geometric constraints, we used $\pi/8$ as a cutoff that removed any potential link whose intersection angle was greater than this.

3.3.2 Experimental Results

In this section we discuss the findings from testing our algorithms. To the best of our knowledge, no prior work or image data is available on extracting road-lane boundaries that we could use for comparison. Hence, we had to come up with reasonable ways of evaluating our results.

We evaluate resulting road-lane boundary delineation in two-folds: accuracy of matching between output and ground truth pixels and counting the number of correctly recovered road-lanes in the final outputs. Matching pixel-to-pixel aims at investigating the performance of our approach at a micro-level; counting the number of road-lanes aims at revealing the accuracy of the resulting geometries. To evaluate our results at a pixel-to-pixel level, we utilized the method from evaluating performance of object boundary detection [Martin et al., 2004]. Similar to [Martin et al., 2004], we regard the extraction of road-lane boundaries as a classification problem of identifying boundary pixels and of applying the precision-recall curves using manually labeled road-lane boundaries as ground truth. Precision is manifested in the fraction of outputs that are true positives; recall is the fraction of correct outputs over true positives. The precision-recall depicts these two values together as the threshold varies, capturing the trade-off between accuracy and noise. In a precision-recall curve, each of the output pixels is evaluated by whether it detects true positive pixels. Once we obtain such correspondence between output pixels and ground truth pixels, computing the precision and recall is straightforward. While resolving this correspondence problem, we must carefully consider a localization error that accounts for the (Euclidean) distance between an output pixel and a ground truth pixel. Indeed, localization errors are present even in the ground truth images as well. For resolving the corre-

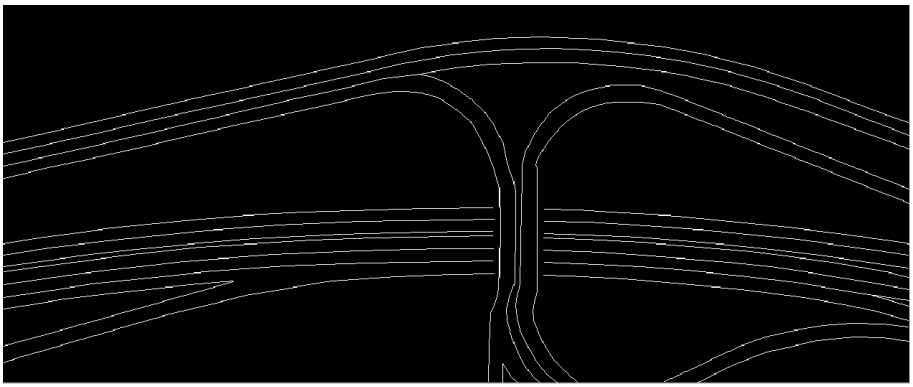
spondence between output pixels and ground truth pixels, we utilized the Berkeley Segmentation Engine's [5] performance evaluation scripts. These scripts solve, using Goldberg's CSA package, the correspondence problem as a minimum cost bipartite assignment problem. We also used, as a baseline method, BSE's probabilistic boundary detection outputs. BSE was developed to detect object boundaries, not road-lane boundaries. In addition, since training BSE with our image data is impossible, it may fall short of being a fair comparison. But since anyone can think of such probabilistic boundary outputs as a starting point of delineating road-lane boundary lines, we compared it with our output.

Figure 3.14(a) shows the ground truth image of the input image's road-lane boundaries. Figure 3.14(b) shows the probabilistic output of boundary in which the color closest to red represents the highest probability of being a boundary pixel. Figure 3.14(c) shows the binary output of our algorithms. Qualitatively speaking, our results outperform those of BSE in that most of the road-lane boundary lines are recovered. The BSE outputs, in contrast, produced a great deal of non-road boundaries, such as those from vegetation or houses. Figure 3.15 quantitatively confirms such qualitative differences between the two outputs in a quantitative way with a precision-recall curve. Table 3.3 presents an averaged performance difference between the two outputs over fifty test images.

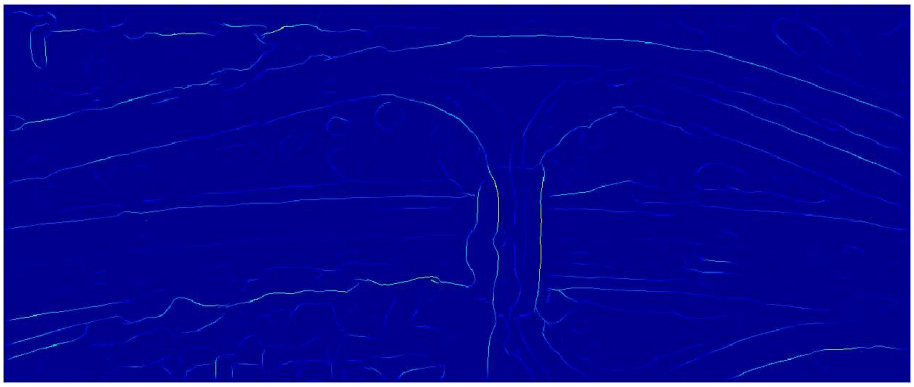
	F-measure	Precision	Recall
Ours	0.82	0.77	0.89
BSE's	0.44	0.38	0.54

Table 3.3: An averaged precision-recall measure of micro-level performance between the two outputs.

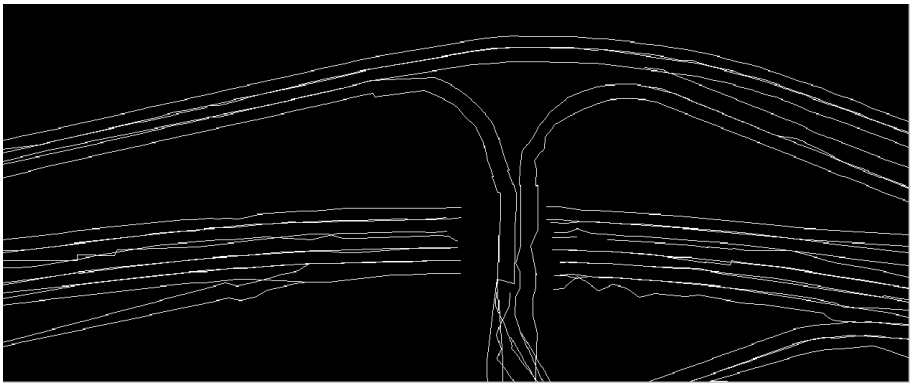
[5]The BSE and related information are available at `http://www.cs.berkeley.edu/~fowlkes/BSE/`



(a) Ground truth binary image.

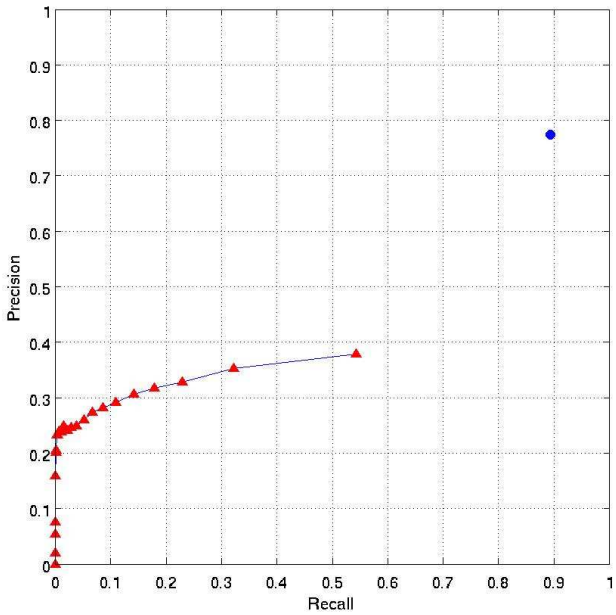


(b) Probabilistic boundary output by BSE.

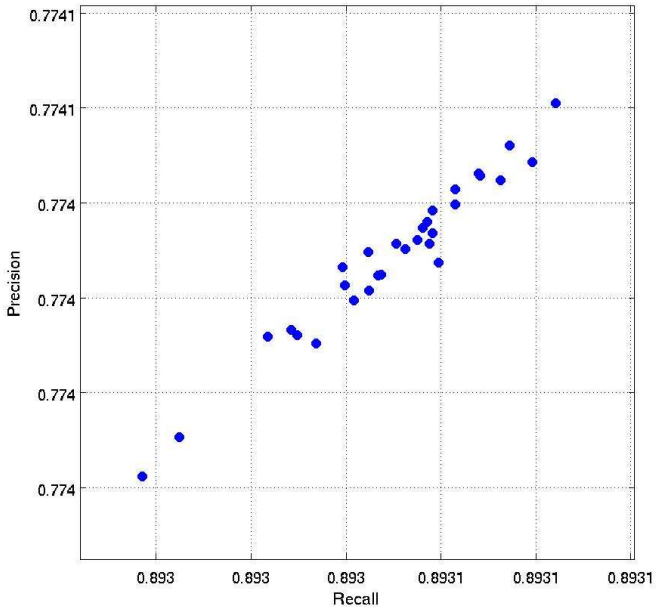


(c) Output binary image.

Figure 3.14: These figures show the ground truth binary image and two output images.



(a) This figure shows an averaged precision-recall curve. A curve with red triangles shows BSE's performance whereas a blue dot cluster shows our performance.



(b) This sub-figure magnifies our performance in the precision-recall curve.

Figure 3.15: A precision-recall curve about micro-level pixel-to-pixel matching is shown.

The performance evaluation by a pixel-to-pixel matching for road-lane boundary extraction outputs might be insufficient in terms of achieving our goal because the pixel-to-pixel measure counted a match when an output boundary pixel is located to a true boundary pixel within a predefined distance threshold (e.g., 100 pixels). Therefore a collection of boundary pixels does not necessarily correspond to a road-lane boundary. To be useful, these detected boundary pixels must be interpreted as parts of a road-lane. In other words, the desirable output for our purpose, is one that treats a road-lane as a polygon, bounded by a closed path, where we can estimate lateral road widths, curvature, and other interesting geometric properties along the centerline of a road-lane polygon. Such an output would also present a clear difference between ours and those of BSE's probabilistic boundary output. To measure such macro-level performance, we first visually inspected our outputs and the input image to resolve the correspondence between the resulting road-lanes and true road-lanes appearing on the input image. We then counted the number of correct and incorrect output road-lanes and missed true road-lanes. If the area of overlap between a road-lane output and a true road-lane is roughly greater than 80%, then we count it a correct match. This counting results in a two-contingency table for the performance of each test image. Table 3.4 shows a macro-level performance that is obtained by merging individual contingency tables over fifty test images. An averaged performance is then computed by using this table, precision = $0.792 = \frac{337}{337+88}$, and recall = $0.771 = \frac{337}{337+100}$, meaning that 79% of the resulting road-lanes are correct and 77% of true road-lanes appearing on the test images are correctly recovered.

		Ground Truth	
		Road-lane	Not road-lane
Output	Road-lane	337	88
	Not road-lane	100	$\times$

Table 3.4: A contingency table is used to measure the macro-level performance of our highway map generation methods.

Examples of resulting maps are shown in Figures 3.16 through 3.18. Appendix C includes complete results of 50 test images.

Figure 3.16 shows some of the most accurate results with all of the true road-lanes appearing on test images recovered correctly. While processing these images, our approach successfully tracked high-curvature ramps, correctly connected road-lane boundaries around overpasses, effectively handled variations in road-surface materials and partial image distortions.

Figure 3.17 shows some reasonable results where most of the true road-lanes are recovered correctly. However, not all the true road-lanes have been recovered and some of the geometry of the resulting road-lanes is incorrect. Our approach was unable to correctly produce road-lane maps from the testing images in Figure 3.17 because these images contain more challenging image characteristics. For example, the overpass in the first row was successfully detected. But, the underestimated boundary of the detected overpass resulted in inaccurate linkages of road-lanes at the edge of the overpass. For the examples in the second row, there was a false positive around the ramp. This happened because our method identified the road-shoulder image-regions as a road-lane. In the testing image in the third row, the shadow of the overpass covers most of

road-lanes located at the left of the overpass. Even with a successful detection of the overpass, due to a relatively high curvature, our approach failed to correctly identify the direction of road-lanes. The railroad appearing in the testing image in the fourth row imposed an occlusion around the lane-merging image-region, resulting in an incorrect linkage between the recovered road-lanes. For the last example of Figure 3.17, the road image-regions on the overpass are distorted, resulting in one of the three road-lanes being completely undiscovered.

Figure 3.18 shows near-failure cases where some of the true road-lanes are not recovered and where some of the true road-lanes are incorrect. The test image shown in the first row posed the most significant challenge in our test image collection. The road-lanes appearing on the left of the image are significantly distorted and a cascade of overpasses makes it even harder to analyze. Although our approach recovered some parts of the true road-lanes, most of them were inaccurate and the linkages among them were incorrectly determined. In the second example, our approach failed to link road-lane hypotheses due to the presence of the bridge's suspension span and was unable to complete the linkage of road-lanes near the overpass at the bridge-entering region. Testing images shown in the third and fourth rows showed complicated road geometries. Image distortions appearing on overpasses made it even harder to track road-boundary image cues. Our lane-marking detector failed to detect road-boundary cues from the road surface of the overpass in the last example and was unable to correctly delineate road-lane boundaries, resulting in incorrect linkages of road-lane hypotheses around the overpass.

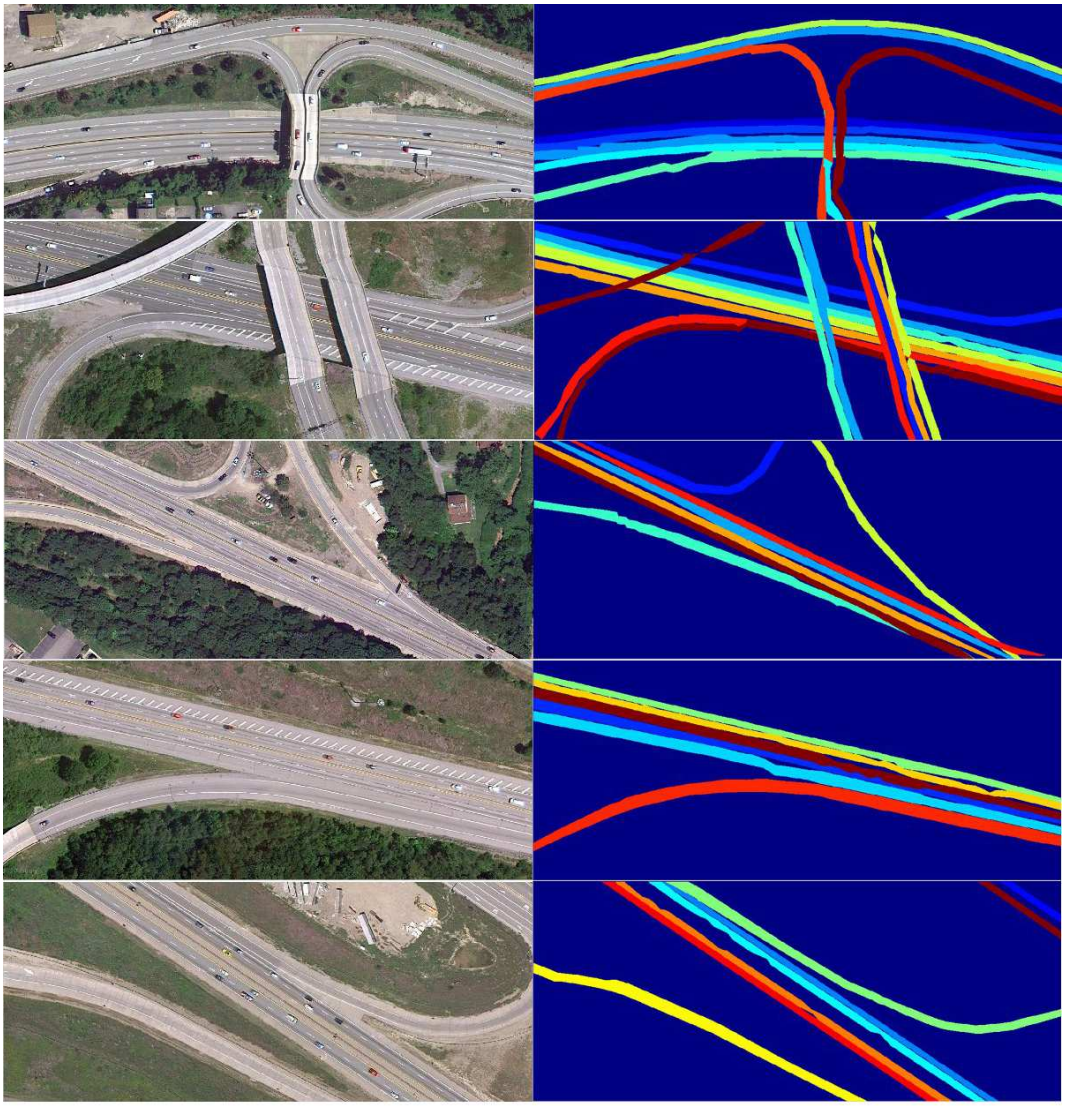


Figure 3.16: There are two sub-figures in each row. The figure on the left is a test image and the figure on the right is our output, where each road-lane output is depicted in a different color and the background is depicted in blue.

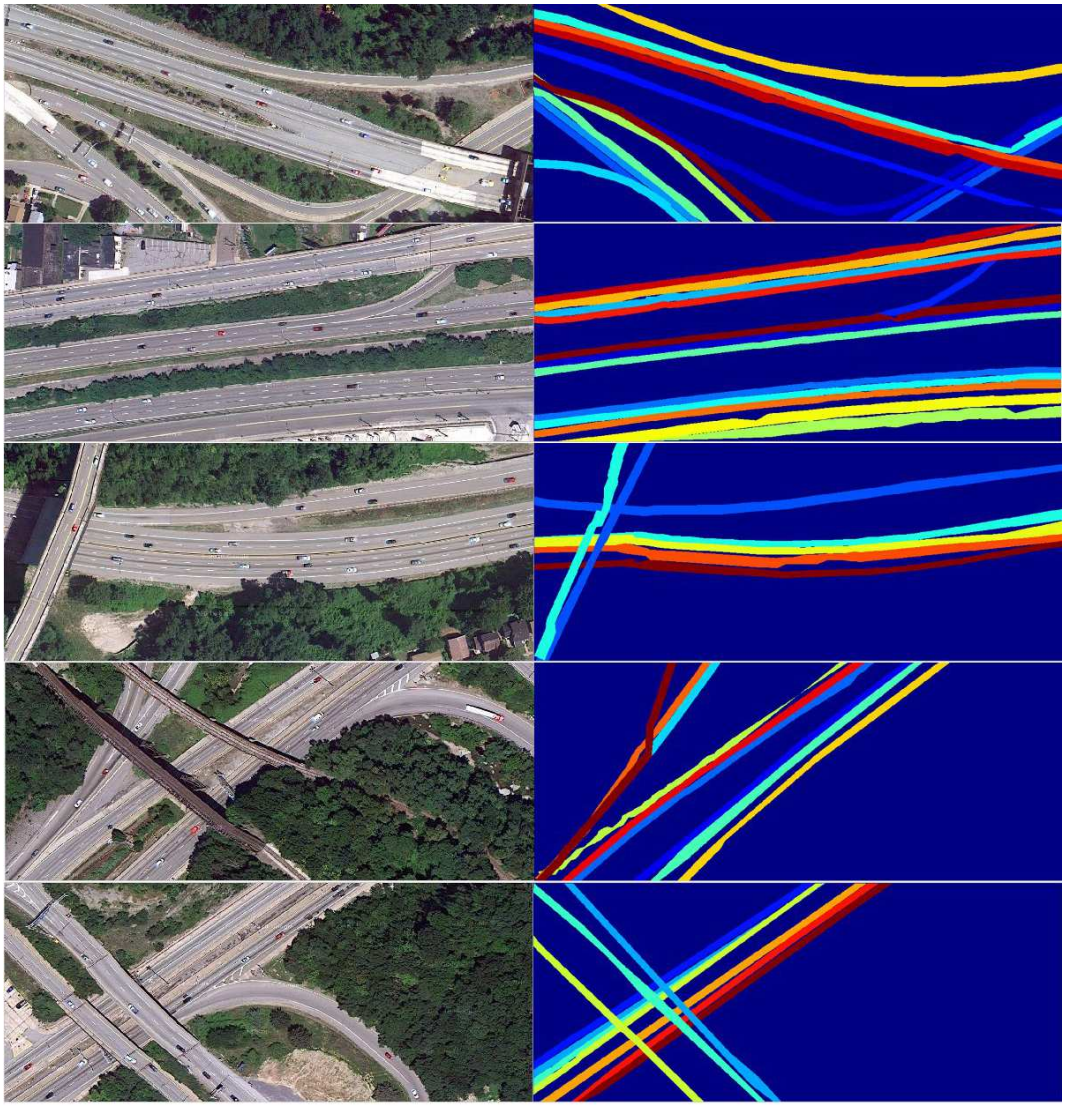


Figure 3.17: These figures show reasonable results where some of the true road-lanes are not recovered or some of the recovered road-lanes have incorrect connections.

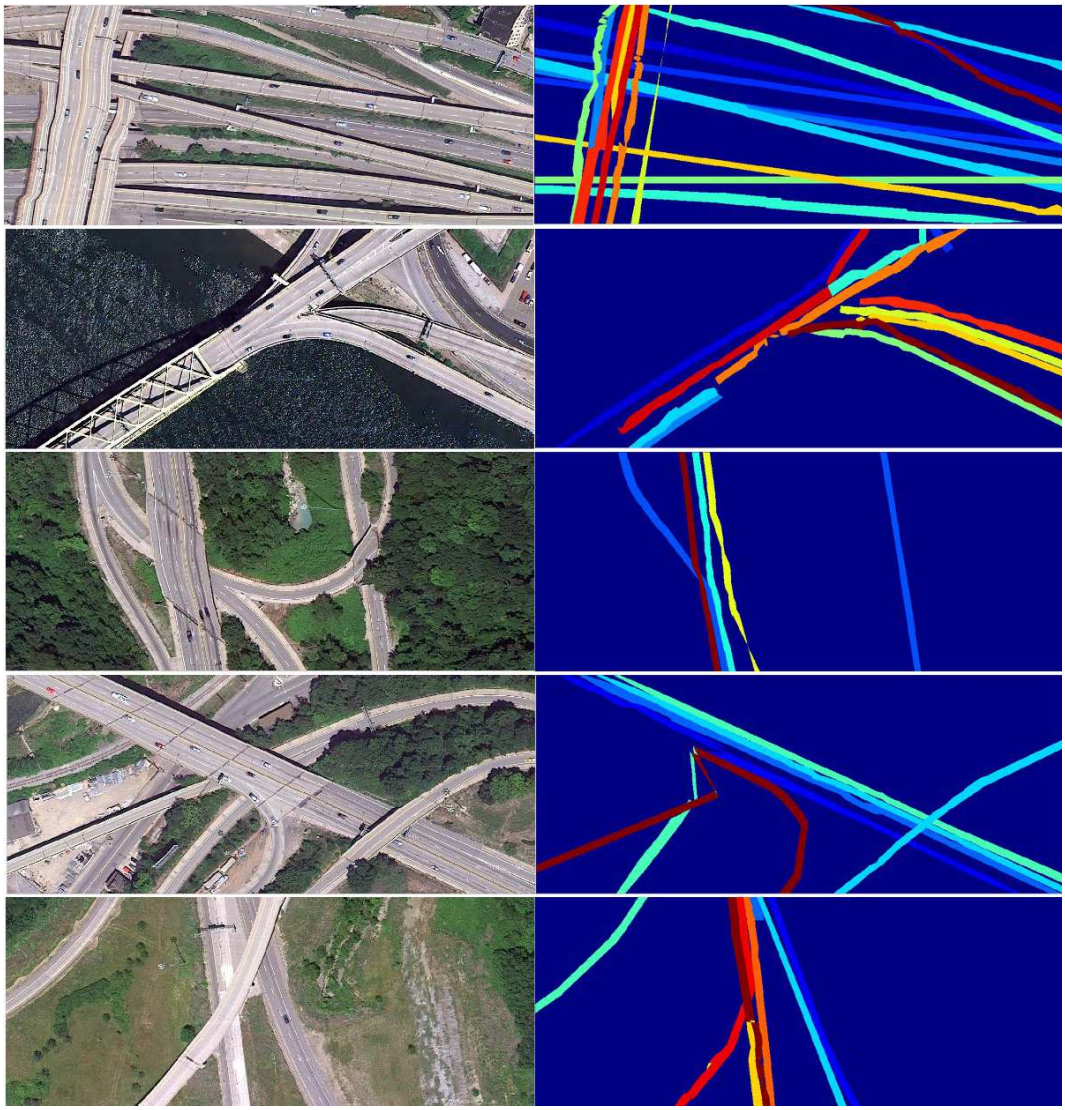


Figure 3.18: These figures show some of near-failure cases where some of the true road-lanes appearing on test images are completely missed: some of the recovered road-lanes do not match to any of the true road-lanes: and finally some of the recovered road-lanes have incorrect connections.

3.4 Summary

This chapter described our approach to extracting lane-level highway maps from a given orthoimage. To make road-lane boundaries computationally accessible and impose reasonable challenges on our task, we chose inter-city highway orthoimages with 15-centimeter ground resolution. Such high-resolution orthoimages pose significant challenges, such as variations in object appearances and complex geometry, to the task of extracting road-lane boundaries. To effectively address these challenges, we developed a hierarchical approach to three tasks: to collecting road boundary image cues via bootstrapping, to generating hypotheses about the unknown true road-lanes, and to linking hypotheses with respect to the photometric and geometric constraints imposed by the collected image cues and prior information. In particular, through bootstrapping, we collected low-level image features from extensive image processing. We refined them to produce task-specific, mid-level image features, such as lane-marking detection, driving-direction estimation, overpass detection, and road-region segmentation. Results of road-region segmentation defined the image sub-region of interest where lane-marking detection results were directly used to generate road-lane hypotheses. We formulated the task of linking road-lane hypotheses as a min-cover problem and found an approximate solution by implementing two linking functions. The first function searched for potential links between two road-lane hypotheses based on the gathered geometric cues. The second function verified these potential linkages by tracking potential paths between two hypotheses. Such a tracking of road boundary image cues enabled us to link two hypotheses along distant, high-curvature, and partially occluded paths. We tested our algorithms with 50 challenging arterial highway images. The results were evaluated according to two aspects: pixel-to-pixel matching and counting correct and incorrect outputs. Our approach demonstrated promising results in that, overall, 79% of the resulting road-lanes were correct and 77% of true road-lanes appearing on the test images were correctly recovered.

Chapter 4

Recognizing Temporary Changes to a Highway

In the previous chapter, we demonstrated that we could generate a highway map of road-lanes from an orthoimage. Such a lane-level detailed highway map with information of traffic rules and real-world coordinates can be prepared in advance for guiding autonomous and assisting manual highway driving. However, describing unexpected occurrences a priori, such as traffic accidents or road work, is of course impossible. A self-driving vehicle must be able to effectively handle such events, as they can lead to temporary changes in road conditions. For example, suppose that the road-lane a vehicle is driving on is unexpectedly shifted laterally due to a road work, whereas that lane is depicted on the map as following a straight path. What if the vehicle's braking distance is longer than its sensing horizon?

To effectively handle unexpected events on a highway, an autonomous vehicle should first be able to recognize them. To provide a vehicle with such perception capability, this chapter presents a collection of computer vision methods that identifies the bounds of a workzone, e.g., the beginning/end of a workzone, and recognizes temporary changes to highway driving conditions, e.g., a decrease in speed or blockage of a lane, through recognition of workzone traffic signs in perspective images [Seo et al., 2011a, Seo et al., 2011b]. Such detailed information about a highway workzone would help a robotic vehicle properly respond to unexpected events on a highway and in turn lead to safe and reliable autonomous highway driving. This functionality would also help a human driver be on alert while driving by such unexpected events.

Regarding their location and appearance, workzone signs are highly constrained by governmental regulations [U.S. Department of Transportation, 2009]. However, such constraints do not make it easy to recognize signs in images because of the high variation in each sign's image appearance. Under perspective imaging, the projection of a 3-dimensional traffic sign onto a 2-dimensional image plane distorts most of the sign's geometric properties, such as its angles, distance, and ratios of angles [Hartley and Zisserman, 2003]. As an example, consider a canonical workzone warning sign in the U.S. [U.S. Department of Transportation, 2009]. It has a diamond shape, orange color, equal corner angles, and equal-length edges. When such a sign is projected onto an image, the sign's equiangularity and equilaterality are not preserved. In addition, the line of sight between a sign and a camera perceptually and computationally changes the color of a workzone sign from that of the sign template. This leads to the problem of intra-class appearance

variation. This type of variation occurs when the appearance of the same workzone sign varies based on the conditions of the image acquisition process.

To cope with such challenges in recognizing workzone signs, our approach learns variations in color in workzone sign images to perform a pixel-wise binary color classification, identifies blobs to localize sign image regions, then represents a cropped image in a homogeneous feature space to reduce the variation of geometric shapes for more accurate sign classification. Realistically, any sign recognition system is going to make errors – incorrectly classifying signs or even missing some signs. To address these potential recognition errors, we devise two algorithms. The first makes use of temporal redundancy of sign occurrences and their corresponding classification decisions, in order to reduce false positives. The second estimates the likelihood of driving in a workzone based on confidence values of the previous classifications, in order to avoid the impact of false negatives.

4.1 Recognizing Highway Workzone Signs

A highway workzone is an exceptional event that briefly changes driving conditions in terms of road geometry and traffic rules. Although a workzone plan is usually advertised in advance, such warnings are not precise enough that a robotic vehicle would know the bounds and types of work. Individual configurations of workzones vary, but, for safe driving, a highway workzone in the United States is required to consist of four sections (or areas): advance warning, transition, activity, and termination [U.S. Department of Transportation, 2009]. Upon entering an advance warning section, human drivers are informed of what to expect ahead. While driving through a transition area, drivers may be forced to deviate from their normal paths. The activity area is where the work actually takes place and the termination area is where the traffic resumes normal activity.

Figure 4.1 shows 10 workzone signs that are typically observed while driving thru these four different workzone sections. Specifically, the first image is observed in an advance warning area, and the next three images appear in a transition area, indicating the bounds of the workzone. The remaining signs are about temporary changes to the highway's traffic rules and geometry.

Our task in this chapter is to reliably detect and accurately classify relevant workzone signs from a perspective video, in order to acquire detailed information about a highway workzone, such as where a workzone begins/ends and how the work changes driving conditions.

To this end, we develop computer vision methods capable of detecting and classifying a set of relevant highway workzone signs as well as reducing potential sign recognition errors based on the confidence values of previous sign classifications. Section 4.1.1 and section 4.1.2 detail our approach for detection and for classification of workzone signs. Section 4.2 explains our approach for dealing with possible sign recognition errors.

4.1.1 Workzone Sign Detection

Although it is obvious that the color of a workzone sign is orange, it is challenging to correctly identify orange pixels in a given image because of possible variation of the color orange. To effectively deal with such variation, we formulate the learning of the orange color variation as a

Figure 4.1: A montage of ground truth annotation examples. The (cyan) rectangular lines outlining the signs represent the contours of the true signs, and the (green) boxes represent the signs' bounding boxes. In the top row, from the left, the images represent examples of W20-1, R22-1, G20-2, W21-19, and R2-2-2. The bottom row includes examples of W1-4, W1-4L, W1-4R, W4-2R, and W4-2L. We include these designations (or identifiers) of workzone signs for completeness and also to later represent the sign target classes.

binary color classification using the Bayesian inference framework. [Bishop, 2006].

$$P(\text{sign}|\mathbf{X}) = \eta P(\mathbf{X}|\text{sign})P(\text{sign})$$

where $\mathbf{X}$ is an image comprised of $|\text{width}\times\text{height}|$ number of m-dimensional pixels, $\mathbf{x}_j \in \mathbf{X}$ and η is a normalizer for the posterior distribution. In particular, $\mathbf{x}_j$ is a 2-dimensional color vector of which components include hue and saturation values. The posterior probability, $P(\text{sign}|\mathbf{X})$, which assigns a value to the probability that individual pixels are part of workzone signs, is computed by multiplying the likelihood function, $P(\mathbf{X}|\text{sign})$, and the prior probability distribution, $P(\text{sign})$, of traffic sign locations found in image frames. We obtain the prior probability density of workzone sign locations from our ground truth data, which is comprised of several hours of highway workzone video footage and manual annotations. Figure 4.2 shows the density of workzone traffic sign locations, which is obtained by projection all of the ground truth bounding boxes onto an image, and is used as the prior for workzone sign locations.

We use AdaBoost [Freund and Schapire, 1996] to learn the likelihood function, $P(\mathbf{X}|\text{sign})$, of a given pixel as a part of a workzone sign. The training data is comprised of a set of workzone images, some of which were downloaded from the web while the rest were obtained from our

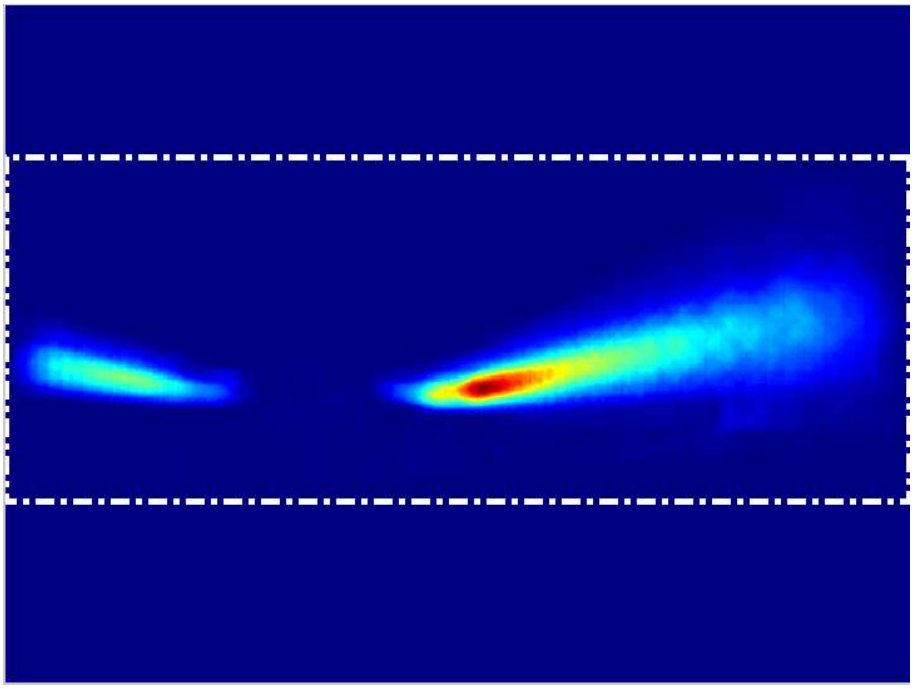


Figure 4.2: A heat-image (640×480) showing workzone sign locations where the color closest to red represents the highest density. The (white) rectangular dashed-line defines the region of interest (ROI) for our workzone sign detection.

workzone video images. Using this data, we train a set of weak-learners and their weights.

$$\begin{aligned} P(\mathbf{x}|\text{sign}) &= \text{mode}\left(\cup_j g(f(\mathbf{x}_j|sign))\right) \\ \text{where,} & \\ f(\mathbf{x}_j|\text{sign}) &= \sum_{i=1}^{H} \alpha_i h_i(\mathbf{x}_j) \end{aligned}$$

where H is the number of weak learners, h_i represents a weak learner implemented by a decision stump and α_i is its weight. We use logistic regression to implement the function g to convert the binary output of AdaBoost into a probabilistic output [Friedman et al., 2000], $g(f(\cdot)) = \frac{\exp(f(\cdot))}{\exp(f(\cdot))+\exp(-f(\cdot))}$.

For a given image, our color classifier evaluates pixels within the ROI, as presented in the Figure 4.2, and assigns a probability for whether individual pixels are part of an orange workzone sign. Our sign detector runs a connected-component grouping algorithm to identify orange blobs and generates up to k bounding boxes as candidates for a workzone sign. The detector then removes any of bounding boxes with radii[1] smaller or larger than the predefined thresholds and uses non-maximal suppression to select the largest bounding box. The confidence value of the selected bounding box is computed using the mode of the confidence values assigned to all

[1]The radius of a polygon is measured by computing the Euclidean distance between a point on a side (or edge) of a polygon and the centroid of the polygon.

pixels within the bounding box. To detect a regulatory workzone sign which includes two colors (orange at the top and white at the bottom), we implement a heuristic for investigating the aspect ratio of a bounding box, in order to extend the height of the bounding box.

To evaluate the performance of our sign detector, we use the performance metrics used for PASCAL object detection challenges [Ponce et al., 2006]. An output bounding box, o_i, is considered a potential match to the ground truth bounding box, g_i, in a given image frame, i, if their area of overlap is greater than a predefined value, $\tau < \frac{Area(o_i \cap g_i)}{Area(o_i \cup g_i)}$. When a potential match is found in a given image, sign detection performance can be further analyzed by measuring the following performance metrics: *precision*=$\frac{Area(o_i \cap g_i)}{Area(o_i)}$ and *recall*=$\frac{Area(o_i \cap g_i)}{Area(g_i)}$.

	Color-based		Shaped-based [Barnes et al., 2008]	
	Warning	*Regulatory*	*Warning*	*Regulatory*
Precision	0.951	0.954	0.487	0.535
Recall	0.928	0.903	0.497	0.662

Table 4.1: For this test, we used 103 workzone images, including 55 warning (or diamond-shaped) signs and 41 regulatory (rectangular-shaped) workzone signs. We set τ 0.5 as the value for a potential match.

Table 4.1 presents macro-averages of precision and recall where a macro-average is computed by averaging individual measurements over testing images. We compared the performance of our color-based sign detection approach to Loy and Barnes' method ("Shaped-based" method in Table 4.1) which utilizes geometric shapes of signs to achieve sign detection [Barnes et al., 2008]. Loy and Barnes' method did not perform well for our data because most of our testing sign images have low contrast in image intensity. Figure 4.3 shows some examples of the sign detection output obtained by our method.

4.1.2 Workzone Sign Classification

An image sub-region localized as a potential workzone sign is given as input to our sign classification module. Our task in this thesis is to recognize the bounds of a workzone and temporary changes to highways by classifying workzone signs. In this regard we chose 9 workzone signs as reliable indicators of workzone bounds and driving condition changes and assigned all remaining workzone signs to another class. Table 4.2 shows the number of sign image examples used for each target class.

W20-1	R22-1	G20-2	R2-2-2	W1-4	W1-4L	W1-4R	W4-2L	W20-5L	*all other*
86	55	35	75	36	26	43	19	17	92

Table 4.2: The number of sign images for each target class. The *all other* class represents one that includes all other workzone signs.

Figure 4.3: Some sign detection output images are shown. The thick (yellow) rectangles outlining the signs represent the overlap between the ground truth rectangle (green) and the detection output (red).

It is challenging to correctly classify sign images because of the variation between each sign's appearance in an image. To reduce such variation, we use a log-polar transform, which is a method used for transforming an image from Cartesian coordinates into an image in log-polar coordinates [Belongie et al., 2002]. This transform is effective in reducing the variation of sign shape and text because it densely samples image intensity values near the center of a sign image where the difference between signs images is relatively small, and then sparsely collects values from sign image boundaries, where the geometric distortions are large. The log-polar transformation of a point in Cartesian coordinates, $I(x, y)$ is mathematically defined as

$$\begin{aligned} T(I(x,y)) &= [c_x + \rho(x,y)\cos(\theta), c_y + \rho(x,y)\sin(\theta)], \\ \rho(x,y) &= \log_{10}\left(\sqrt{(c_x - x)^2 + (c_y - y)^2}\right) \end{aligned}$$

where c_x and c_y are coordinates of a sign sub-image's centroid. The coordinate-transformed values are then quantized into predefined bins.

Figure 4.4 shows three different images of the same workzone sign and their log-polar images. The first image in the top row is the canonical template of a workzone sign while the other two images are workzone sign images cropped from our video data. Although their actual appearances are quite different, their log-polar images are similar to each other. The dimension of the canonical template image (32,000) is determined by its width (200) and height (160) whereas that of the log-polar image (1,024) is determined by the combination of the number of bins indicating the distance to the center, ρ, (32), and the number of bins indicating orientations

in counterclockwise, θ, (32).

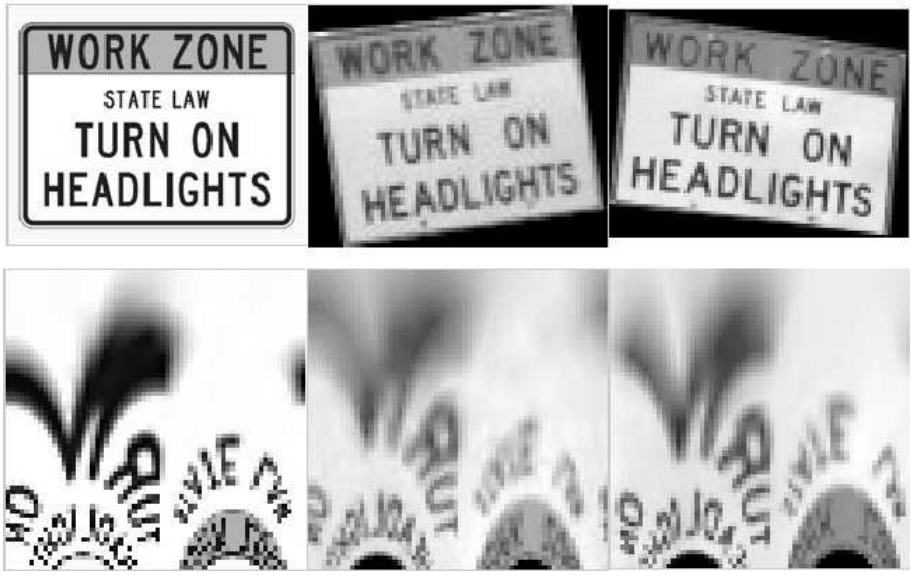


Figure 4.4: Examples of log-polar transformation. Three different images of the same workzone sign compared in a Cartesian coordinate and a log-polar coordinate.

For a given image sub-region localized by our sign detector, our sign classifier first normalizes the image to reduce intensity variation, then converts the cropped image into a log-polar image based on the parameters, ρ and θ, and finally produces a column vector, $|\rho \times \theta| \times 1$.

Even with such an effective feature representation method, any conventional supervised classifier might still fail to generalize the target function in a high-dimensionality space (e.g., 1024) because of the small number of examples (e.g., less than a hundred) for each of the target sign classes. To handle the curse of this dimensionality problem, we further reduce the original dimension of the log-polar image using principal component analysis (PCA). We then build an eigen-space from the labeled training data and project a testing sign image in the log-polar coordinate space onto this eigen-space. The eigen-space is comprised of k eigen bases, all of which represent more than 95% of the total variance in the log-polar image data matrix. Empirically we found that 10 eigen bases achieved the best performance.

Table 4.3 shows the results of our workzone sign classification. The hyper-parameters of these classification methods were chosen through cross-validation.[2] Due to the random selection of our training data, we averaged our results over 5 separate runs for each method. To measure the effectiveness of our sign image representation, we compared the results with another representation method, which scales raw-intensity sign images into an image of the same size (e.g.,100$\times$100), converts it into a multi-dimensional vector (e.g., 10000$\times$1) and then reduces the

[2]For SVM, we used the LIBSVM `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`. We found that a SVM with an RBF kernel ($\sigma = 0.125$) worked best and used an one-against-one scheme for multi-class classification. LDA used the weight vector that best performed against the validation set for testing. Our kNN implementation worked best when it used the top 10 closest neighbors in terms of Euclidean distance in the eigenspace.

dimensions using the precomputed eigen bases. The last two rows of Table 4.3 show the performance of three classifiers that use a raw-intensity image representation. The table demonstrates that log-polar sign image representation helps classifiers achieve better classification results. All evaluation metrics indicate that SVM outperforms the other two methods.

	SVM	*LDA*	*kNN*
Precision	**0.965/0.012**	0.856/0.035	0.285/0.027
Recall	**0.957/0.016**	0.854/0.040	0.387/0.007
Precision	0.896/0.035	0.756/0.030	0.252/0.015
Recall	0.841/0.028	0.742/0.030	0.377/0.015

Table 4.3: Performance of three different sign classification methods measured by standard metrics. Each cell in the table shows the mean and standard deviation.

4.2 Handling Workzone Sign Recognition Errors

Although the previous two sections demonstrated promising results for our sign detector and sign classifier, it is realistic to conjecture that our approach makes mistakes in recognizing some workzone signs. When either a miss or incorrect classification occurs, our methods might fail to acquire detailed information about a highway workzone.

To handle such potential sign recognition errors, we devise two algorithms that utilize the sequence of previous sign classifications. These methods rely on the accuracy of our sign recognition method, which is able to accurately recognize the majority of the target class signs.

Our sign classifier produces a sign classification decision and its confidence value in cases where the sign detector produces a bounding box as a potential sign image. These confidence values represent the level of confidence in our approach's representation in terms of determining whether the cropped images are instances of target workzone signs. We can thus use the magnitude of the confidence value to infer whether our vehicle is driving in a workzone. However, a problem with using these values directly is that a sparsity of confidence values exists, as we cannot obtain such evidence from workzone regions where no workzone signs are posted or from true workzone signs that are misclassified as other objects. The underlying aim of our algorithm is to propagate confidence values over time in order to hold non-zero values while driving in a workzone, even when the system does not have direct observation of a workzone sign or misses any workzone signs. While spreading these values, the effect of propagation should decay over time, in order to prevent an over-estimation of the true state.

To implement our idea for driving region inference, we use Gaussian smoothing of the confidence values over a specific time domain. Classification confidence at the ith time step, δ_i, is propagated to adjacent time periods as far as the value of σ.

$$[\delta_i * w_j]_{j=-\sigma,\ldots,-1,1,..,\sigma}, w_j = \exp\left(-\frac{i-j}{2\sigma^2}\right)$$

Assuming that the driving speed is 50 mph and the frame rate is 15 per second, an image frame in a video represents a distance of 1.4 meters of driving. In this case, if we set σ to 150 (or 150

image frames), the confidence value will be propagated over 210 meters, both toward future and past time steps. Note that, although it is unnecessary to propagate confidence values toward the past, we propagated them in both directions for convenient implementation, without paying any extra computational cost. The choice of σ is critical for production of a smooth inference curve. If σ is not optimal, either discontinuity or inflation will appear in the resulting curve. We define the value of σ based on the rough estimation of distance between workzone signs. The likelihood function for driving in a workzone is then computed by adding the current classification confidence value, δ_i, and the propagated confidence values accumulated at time step, i, obtained from the neighboring time frames, $score_i = \delta_i + \hat{\delta}_i$, where $\hat{\delta}_i$ represents the confidence values propagated to the time step, i. For example, suppose that our sign recognizer misses a true sign at time step, j, where $i < j \leq i + \sigma$. Because time step j is in the propagated confidence interval, σ_i, our vehicle knows that it is driving on a highway workzone, even with a sign recognition miss.

In a workzone video, a workzone sign appears multiple times before it disappears from the camera's field of view. Our approach utilizes such temporal redundancy of sign occurrences to improve classification accuracy, particularly reducing the rate of false positive. Specifically, when the system makes a classification decision, it refers to previous classification outputs.

$$y_t(\mathbf{o}_t) = \arg\max_c \left\{ h_t(\mathbf{o}_t, c) + \sum_{l=1}^{T} \gamma^l h_{t-l}(\mathbf{o}_{t-l}, c) \right\}$$

where $h_t(\mathbf{o}, c)$ and $h_{t-l}(\mathbf{o}, c)$ represent the classification outputs for an image sub-region, $\mathbf{o}$, at time steps, t and $t-l$, for the class, c, and γ is a discounting factor that determines contributions of previous decisions to the current classification decision. Note the sign detector applies non-maximum suppression in order to ensure that image sub-regions, $\mathbf{o}_t$ and $\mathbf{o}_{t-l}$, represent the same object in different scales. By investigating previous classification decisions on the same sign in different scales, our approach offers the opportunity to alter its current classification decision, which has $1-0.965$ chance of producing a false positive based on Table 4.3. When a classification decision is made, the system propagates its classification confidence over adjacent time frames based on σ.

4.3 Experiments

This section details experiments conducted to investigate the robustness of our sign detection and classification method with images acquired under various illumination conditions. This section also discusses the reliability of our potential sign recognition error handling method under the use of a series of sign classification outputs.

4.3.1 Experimental Settings

We collected several hours of video footage of various highway driving experiences and prepared 5 videos out of these as testing data, where each of the five videos showed a vehicle's perspective when driving on a normal highway, passing a workzone, and driving on another normal highway.

Figure 4.5 shows the setup for our video acquisition. Each of these videos was decompressed into a set of images. The number of images in the videos varied from several hundred to thousands. Table 4.4 gives detailed information about the video data. For example, in the video data, A, there are 447 out of 3,305 images containing workzone signs and 36 images containing other traffic signs.

	A	B	C	D	E
Sum of images	3,305	4,232	874	3,148	3,280
Workzone signs	447	603	234	451	477
Other traffic signs	36	89	21	68	62

Table 4.4: These video data were acquired under various weather conditions. The first two videos (A&B) were recorded in winter with snow accumulation in the background, while the following two videos, (C&D) were obtained in spring, under fairly gentle illumination conditions (i.e., sunny and clear skies), and the last was recorded on a rainy day in spring.

Figure 4.5: A setup of workzone video recording.

For each of the video data, a stream of images was given to our system, which was required to localize signs, if any, and classify them, if necessary. For the sign detector and classifier, we used the best-performing learners described in Section 4.1.1 and Section 4.1.2. We empirically found that the temporal smoothing worked best when γ was 0.9. We set σ to a value in the range of 350 (i.e., 500 meters) to 800 for Gaussian smoothing based on the scale of highways and the maximum inter-distance between signs, as described in [U.S. Department of Transportation, 2009].

	A	B	C	D	E
Sign Detection	0.782/0.474	0.314/0.796	0.825/0.426	0.721/0.548	0.928/0.471
Sign Localization	0.940/0.474	0.841/0.909	0.915/0.775	0.948/0.790	0.918/0.757
Sign Classification	0.823/0.678	0.696/0.539	0.598/0.579	0.719/0.704	0.580/0.608

Table 4.5: Results of performance tests for individual modules.

4.3.2 Experimental Results

Table 4.5 shows experimental results of workzone sign recognition tests. The first row shows the performance of sign detection, which measured the number of signs detected in video data. This is done on a per sign instance basis. The second row shows the accuracy of sign localization in terms of overlap with true signs. The third row shows the accuracy of the output of our sign classification with a given localized sign image. The two numbers in each cell correspond to precision and recall. In comparison with individual unit tests described in two previous sections, Section 4.1.1 and Section 4.1.2, the overall performances are slightly degraded, particularly, for the recall rates of detection in the first row. This is the result of the dense manual labeling that we used when annotating the bounding boxes, in which the labeling began when a true workzone sign was about 20×20 pixels. By contrast, our sign detector was tuned to filter out any orange regions with dimensions smaller than 40×40. Performances of sign localization showed that most of the true sign image regions were recovered once they were detected. For example, for video data, A, even though the sign detector detected only 47% of the ground truth signs, 94% of the localized sign images were correct. These cropped images are then forwarded to the sign classifier that produced highly accurate classifications, 82%. Some parts of the true signs are cropped away, but the parts of the true signs that are important for classification were passed to the sign classifier. This enabled our approach to perfectly identify the bounds of the workzone and robustly detect most of the driving condition changes, as shown in Table 4.6.

Table 4.6 summarizes the experimental results in terms of recognition accuracy of temporary changes in driving conditions. The first two rows show the accuracy of workzone bound recognition and the remaining five rows show that of driving condition change recognition. Our approach demonstrated excellent performance in identification of workzone bounds. For example, for the test video data E, there are 3 images that contain "workzone-begin" signs (i.e., *R22-1* in Figure 4.3). Although the performance of workzone sign classification on these sign occurrences was not impressive (i.e., 0.083 as precision and 0.333 as recall), our sign detector and classifier successfully recognized one of the three signs with high confidence. The correct classification in fact, happened at the middle of the three sign appearances and the first and the last (or latest, in terms of time elapse) classification decision on the same sign were incorrect. If our system only considered the latest classification decision without looking into previous classification decisions, the system would miss an important workzone sign and eventually, at best, underestimate the bounds of the workzone. But one of our recognition error handling methods utilized these consecutive classification decisions as explained in equation 4.1, enabling the system to turn on the flag to indicate whether our vehicle was driving in a workzone. Without these methods, we might see inconsistent sign classification decisions on the same sign in different time frames (or scales) and miss some of the workzone signs which are important for determining the bounds of

a workzone. Thus, while there were fluctuations in workzone sign classification performance, the overall trend was similar to this example, resulting in our system's recognition of all of the highway workzone bounds in the test video.

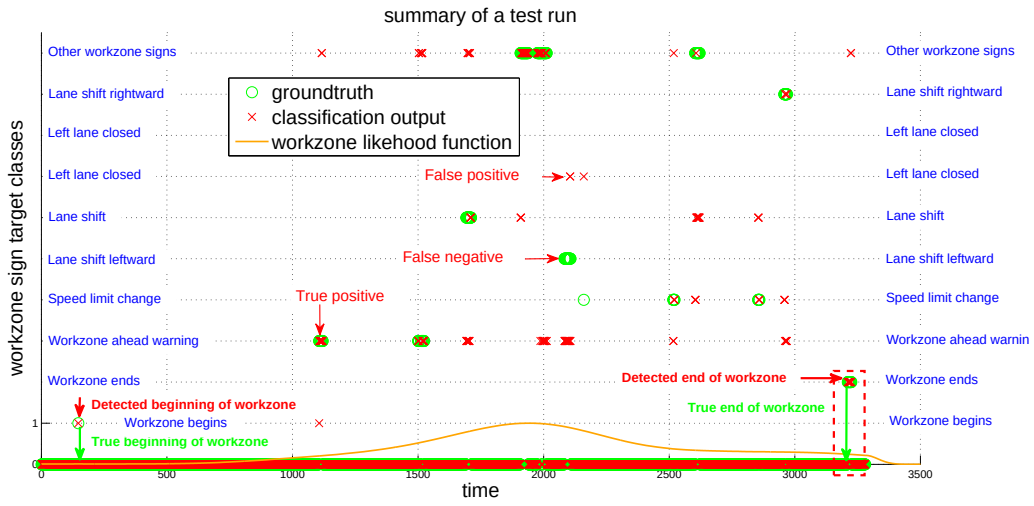
Our methods did, however, make some mistakes in detecting temporary changes in highways conditions. There were 14 occurrences of temporary changes to highway driving environments in our test videos, and four of them were not recognized. This was primarily a result of our sign detector being unable to localize signs in the under- or over-exposed images of the test video data. This resulted in zeroes of the sign classifier's performance because our workzone sign classifier did not receive potential sign images from the detector for classification and incorrectly classified some of the potential sign images.

	A	B	C	D	E
R22-1	○/9/0.727/0.889	○/8/0.5/0.875	○/1/0.5/1.0	○/5/0.625/1.000	○/3/0.083/0.333
G20-2	○/12/1.0/0.333	○/31/1.0/0.290	○/15/1.0/0.867	○/12/1.0/0.833	○/15/1.0/0.933
R2-2-2	N/A	○/12/1.0/0.083	○/13/0.5/0.461	○/9/0.857/0.666	×/7/0.0/0.0
W1-4	N/A	N/A	○/20/0.166/0.200	○/52/0.444/0.077	○/53/0.333/0.170
W1-4L	N/A	N/A	×/25/0.0/0.0	○/24/0.876/0.876	○/31/0.875/0.903
W1-4R	N/A	○/30/0.882/0.5	○/12/1.0/0.250	N/A	×/9/0.0/0.0
W4-2L	N/A	×/22/0.0/0.0	N/A	N/A	N/A

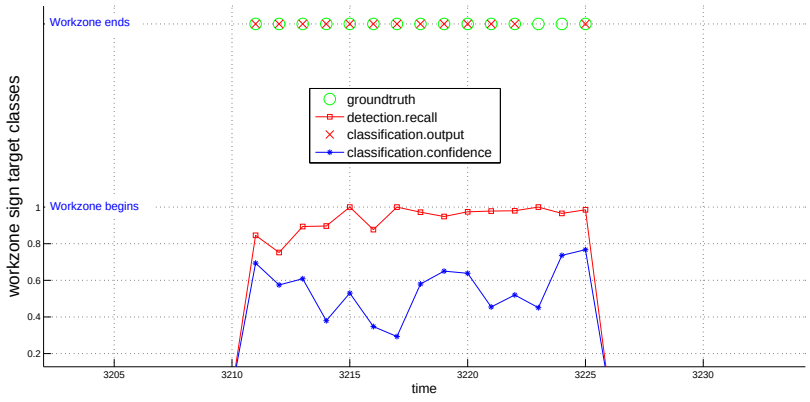
Table 4.6: Results of performance tests on detection of driving condition changes. The four symbols in each cell correspond to success (○) or fail (×)/number of corresponding images/precision/recall respectively.

Figure 4.6 details one of the experimental results, i.e., video data D, where the x-axis represents the number of image frames organized by time and the y-axis represents the target class labels. An instance of sign recognition was counted as correct whenever a (green) circle, representing a ground truth, overlapped with a (red) "x," representing the output of the sign classifier. Figure 4.6(b) magnifies the dashed rectangle in Figure 4.6(a) where the "end-of-workzone" signs appeared 12 times before they disappeared from the camera's field of view. Two additional pieces of information about the recall of sign detection are depicted (in red), which are not available to our system during the testing phase, and the confidence values of sign classification (in blue). The dimensions of a sign in an image enlarge as it approaches the bounds of a camera's viewpoint. Nevertheless, because of unavoidable recognition errors, the larger sign dimensions are no guarantee of performance improvement. In our case, the values of detection recall and classification confidence increased as the sign grew in size. However the fluctuation of these numbers were observed to be a result of recognition error. Two of the last five classification decisions were incorrect. In spite of this, the discounted sum of the confidence values concluded that the system recognized the "end-of-workzone" sign and turned the flag off, indicating that our vehicle was leaving a highway workzone. The (orange) curve represents the estimated function value of the likelihood of driving in a workzone. As shown, the values of this function are greater than zero within a workzone. Although the estimated curve slightly overestimated the actual workzone bounds, this function can be used to inform our vehicle of the likelihood of driving

in a workzone, even when our approach misses signs that indicate the beginning or end of a workzone.



(a) A graph showing the summary of the test results of video data, D. For this test we set σ as 800 (about 1,120 meters).



(b) This sub-figure magnifies the dashed rectangle in Figure 4.6(a) and shows two additional pieces of information.

Figure 4.6: Results of a highway workzone recognition test.

4.4 Summary

This chapter presented a set of computer vision methods that localize, detect, and classify workzone signs in video data. This is done to obtain detailed information about highway workzones, such as the bounds of a workzone and temporary highway changes caused by road work. We developed such a perception capability to bridge the gap between what appears on our lane-level detail highway map and what is actually happening on the highway at the time a vehicle is driving on it.

Through a bottom-up structure, our system hierarchically processes an image by first performing a pixel-wise orange color classification step to filter image regions not containing workzone signs; second, by selecting a bounding-box as a potential sign image; and lastly, by classifying the bounding box into one of the predefined target classes. It is, of course, unrealistic to expect error-free sign recognition. Thus we devised a likelihood function to represent driving in a workzone based on the confidence values of previous sign classifications.

We found that our approach is capable of identifying workzone bounds and of recognizing most driving condition changes. We believe that a successful demonstration of our approach is contingent on taking into account three factors. The first is that instead of manually tuning the optimal ranges of color values, we learned variations of orange color through a machine learning technique to localize sign image regions. Some researchers in this field may be skeptical of using color information for sign recognition. We showed, however, that a color-based sign recognizer works successfully as long as the test data is composed of the same color variations as those of the training data. In fact, we believe that this approach is further useful in that it makes it easy for one to produce a sign detector for localizing particular traffic signs (e.g., red stop signs or yellow yield signs) if the relevant sign images are manually prepared. However, we expect cases where in practice a color-based sign detector fails (e.g., a variation of color has not been seen during the training phase). Thus, for future work, we would like to investigate an approach that fuses color information with shape information. Returning to the three factors on which the success of our approach is based, the second one is that we used the log-polar transform to represent localized sign images and PCA to reduce the dimensionality of sign image vectors. This approach was effective in reducing variation of geometric distortion in sign images. This was important in our case where an insufficient amount of sign image data was available to learn individual target sign classes. The last factor is that our methods for handling potential sign recognition errors worked effectively. Without these methods we might have seen inconsistent sign classification decisions on the same sign in different time frames. Such inconsistent decisions would have led to our approach missing some of the workzone signs important to determining the bounds of a workzone.

Chapter 5

Parking Lot Map Generation

In the previous two chapters, we demonstrated that we can provide self-driving vehicles, and human drivers alike, with a lane-level detail highway map for assisting their highway driving. To address possible transient changes in driving conditions on highways appearing on the resulting map, we developed computer vision algorithms for recognizing temporary changes, and demonstrated that the developed algorithms are capable of identifying workzone boundaries and recognizing a functional majority of temporary changes on highways.

Suppose that a self-driving vehicle is about to arrive at a parking lot. It would be useful to provide that vehicle with a map of the parking lot before it enters. In particular, information about parking spots' locations and the geometry of drivable regions would ease its parking maneuvers. Without such information, autonomous parking would be quite challenging because it would require simultaneously acquiring this geometric information with on-board sensors, using it to plan and execute motions in real-time.

In looking for an appropriate parking lot model, one might consider drawing from existing cartographic databases. However, as pointed out earlier, this would not be realistic at least for a while, considering that existing road-maps do not contain the required information. At best, a parking lot in a road-map database is depicted as a point in a two-dimensional map space. Alternatively, one can build the needed model of a particular parking lot by fitting a geometric model to sensor measurements [Dolgov and Thrun, 2009, Kummerle et al., 2009]. This approach requires an additional, labor-intensive step requiring that a robot be driven manually to collect sensor measurements.

In this chapter we will instead analyze high-resolution aerial imagery to build our parking lot map. The generated map will specify the location of parking spots and the geometry of drivable regions. Our approach begins with parking spot detection because parking lot structure can be easily determined if the image coordinates of the visible parking spots are identified. Section 5.1 describes how parking spots in a lot are detected using self-labeled examples.

Next, we explain our approach to recognizing drivable regions within a parking lot. These regions can be determined by superimposing the detected parking spots on an estimated parking lot boundary. Section 5.2 describes how the skeleton of drivable regions in a parking lot is automatically extracted from an orthoimage.

In order to be useful, detected drivable regions must be represented in a concise form. Section 5.3 describes an algorithm that generates a graph structure representing a parking lot's drivable

regions.

A parking spot detector might not be able to correctly classify a parking spot if it is of unusual appearance. Unusual-looking parking spots are hard to obtain through our self-labeling process, so they are obtained manually. The number of these manually-labeled examples is small, but their differing appearance enables our parking spot classifier to improve its performance, in terms of false negative rate. Since manual labeling is expensive, the frequency of its usage must be kept at a minimum. Section 5.4 explains how confidence classifiers incrementally utilize the manually-labeled occurrences to handle intra-class variation, reducing the false negative rate. Uncertainty sampling is exploited to minimize the use of manually-labeled data.

5.1 Parking Spot Detection

Figure 5.1 illustrates how we represent parking lots in this work. Our approach parameterizes each individual *parking spot* by its height, width, orientation, and centroid in image coordinates. We define a *parking block* as a row of parking spots for which open-end directions are the same. Each parking block is characterized by the distance between neighboring parking spots in the block (i.e., "D1" in figure 5.1). Parking blocks are related to each other by two distance measures: the distance between conjugate parking spots (i.e., "D2") and the distance between blocks (i.e., "D3" in figure 5.1).

If the image coordinates of all visible parking spots are known, it would be trivial to estimate parameters shown in the figure 5.1. However, in practice we must estimate these parameters from an image. In this section, we detail our hierarchical approach to detecting parking spots. We first presents the image processing steps involved in the low-level image analysis layer. This layer accurately extracts a set of easily found parking spots from the image. We then explain the high-level processing layer which extrapolates and interpolates the spots found by the low-level analysis to hypothesize the locations of remaining parking spots. We then discuss our self-supervised hypothesis filtering approach, which filters bad parking spot hypotheses.

5.1.1 Collecting Self-Labeled Parking Spot Examples

Geometrical and image characteristics differ between parking lots. Most overhead aerial parking lot images contain a number of well-illuminated empty parking spots. Our low-level analysis extracts these easy-to-find spots to be used by the high-level analysis as "seeds" for additional hypothesis generation and by the final filtering stage as canonical self-labeled training examples to adapt the filter to this particular image. The low-level layer carries out multiple image processing steps: line extraction, line clustering, and (parking) block prediction.

Straight lines are important to understanding the shape of a parking lot. We extract most of the available straight lines using the approach proposed in [Kahn et al., 1990]. The approach computes image derivatives to obtain intensity gradients at each pixel and quantizes the gradient directions using predefined ranges. A connected component algorithm is then used to group pixels assigned the same direction to form line supporting regions. The first principal eigenvector of a line supporting region determines the direction of the line.

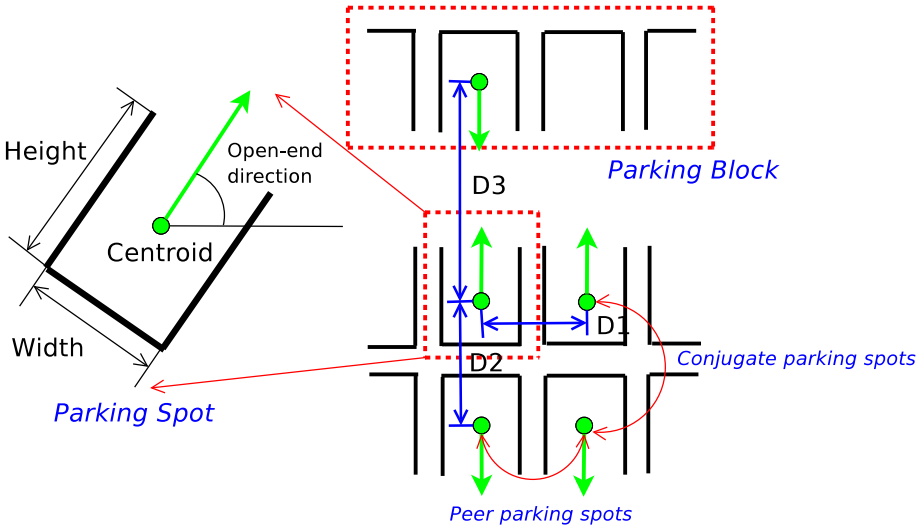


Figure 5.1: A model of parking lot is illustrated.

Although a majority of the extracted lines may align with lane markings in the parking lot, some of them come from other image regions such as road lanes or contours of adjacent buildings. Since we only need the lines aligned with the line-markings of the parking lot, it is necessary to remove lines that do not belong to parking lot structure. To this end, we first group the extracted lines into clusters based on their orientations and then remove lines that are either too short or too long from each of the line clusters. The remaining lines are used for estimating parameters of a parking block. A line cluster corresponds to (at least) one of the parking blocks. [1] We repeat this process (the removal of some of the extracted lines and estimation of parameters of a parking block) with each line cluster.

For parameter estimation, we first estimate the nominal height of a parking spot by computing the mode of lines in the selected cluster. We next build a Euclidean distance matrix across all possible line pairs, quantize the distances and compute the mode to obtain the nominal width of parking spots within a lot. Finally, we quantize the orientation of lines and compute the mode again to estimate the orientation of each parking spots' open-end.

The completion of these image processing steps results in generating few, but highly accurate initial estimates of the true parking spots. Figure 5.2 shows rectangular patches around the image locations of detected parking spots. Although most of these self-labeled parking spot templates are in fact true parking spots, some of them are not since the line analysis algorithm is imperfect. To filter out these incorrect self-supervised parking spot templates, we train a SVM with parking spot examples, which are previously obtained, and conduct a binary classification.

To detect parking blocks, we project the centroids of all the initial parking spots onto a virtual line whose orientation is the mean of the initial parking spots' orientation. This projection returns the distances of centroids from the origin, $\rho_i = c_{i,x}\cos(\theta_i) + c_{i,y}\sin(\theta_i)$, where $c_{i,x}$ and

[1] For most of the testing images used in this thesis, this is true because individual images have a large portion or a whole part of a parking lot. However, when a non-parking lot image is given, our parameter estimation based on line detection might not work.

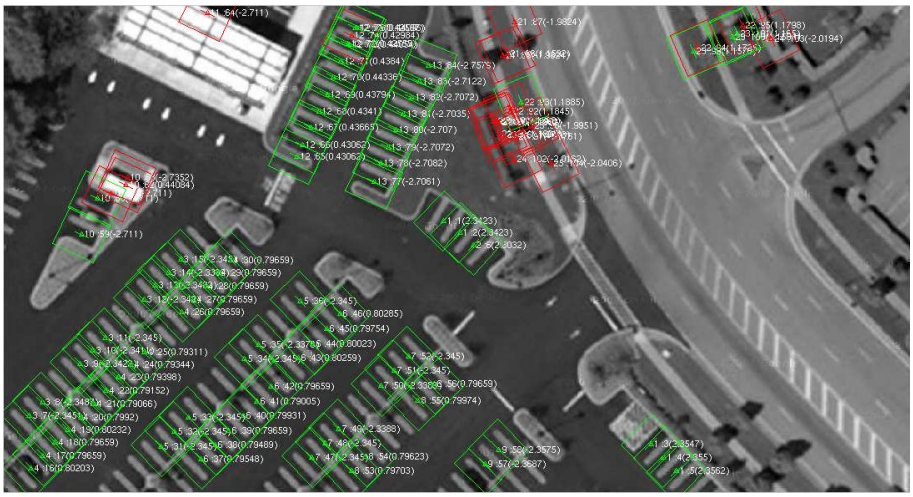


Figure 5.2: An illustrative example image is shown. The low-level analysis produces a set of self-labeled parking spots that are depicted by rectangular patches around their centroids. After filtering out some of the patches (i.e., red patches), the remaining patches (i.e., green patches) are used as positive example to train our hypothesis filters.

$c_{i,y}$ are image coordinates of a parking spot centroid and θ_i is the open-end orientation of the ith initial parking spot. After projection, boundaries between parking blocks are clearly visible and the distance between peer parking spots (i.e. $D1$ in the Figure 5.1) is used to determine boundaries between parking blocks. We finish the parameter estimation process by computing three distances between parking blocks (i.e. $D1$, $D2$, and $D3$ in the Figure 5.1).

5.1.2 High-Level Structure Analysis

The high-level layer is intended to detect all the visible parking spots in an image. It first hypothesizes the parking spot locations based on the parameters estimated by the low-level layer. It then filters these hypotheses by classifying the rectangular image patches around these hypotheses using self-supervised classifiers.

Parking Spot Interpolation and Extrapolation

A parking spot hypothesis represents an image coordinate that indicates the centroid of a potential parking spot. A rectangular image patch around the hypothesis is evaluated to determine if a local characteristic of the image is similar to that of a true parking spot. To cover the image regions that possibly contain true parking spots, we use the image coordinates of the centroids of each self-supervised parking spot as the starting point for each of the discovered parking blocks. We then generate parking spot hypotheses by selecting image locations through three processes: interpolation, extrapolation, and block prediction. The hypothesis generation step aims to intelligently sample image regions, where the low-level did not find spots, The interpolation procedure chooses image coordinates between two end parking spots in a parking block, whereas the extrapolation procedure extends hypotheses beyond the ends of a parking block. The estimated parking

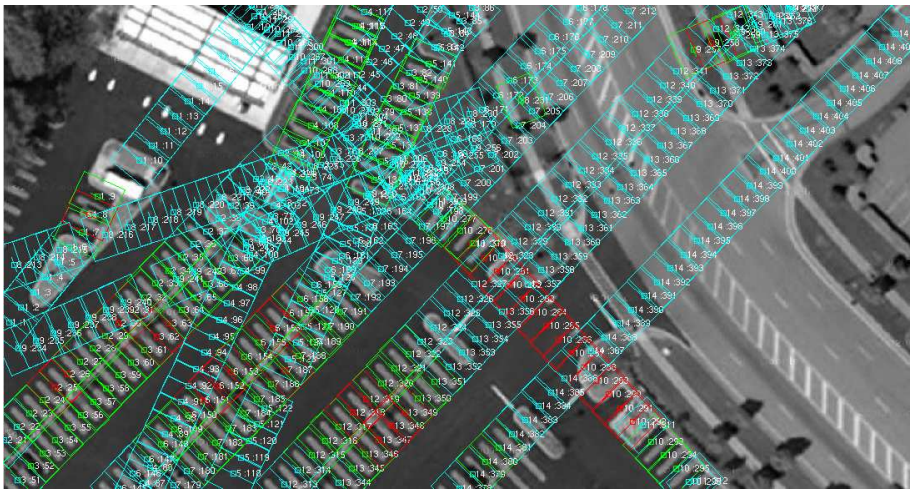


Figure 5.3: A set of the generated parking spot hypotheses is shown. Parking spot hypotheses are rectangular image patches. Different rectangle colors indicate results of different hypothesis generation processes (red patches by the interpolation, cyan ones by extrapolation, and green ones by the low-level analysis). In this example image, there are 114 true parking spots and 411 parking spot hypotheses.

spot width is used as the spatial interval between parking spot hypotheses. Block prediction aims to discover any missing parking blocks.

Self-supervised Hypothesis Filtering

The hypothesis generation process produces n parking spot hypotheses represented by the corresponding number of rectangular image patches, $\mathbf{g}_1, ..., \mathbf{g}_n$. Figure 5.3 shows a representative set of generated parking spot hypotheses where individual parking spot hypotheses are represented as rectangles. Each parking spot hypothesis is evaluated to determine if it is a true parking spot. We formulate this decision problem as binary classification for assigning a label, $y_i \in \{-1, +1\}$, to a given patch vector, $\mathbf{g}_i$, where $\mathbf{g}_i$ is an m $(= \text{height} \times \text{width})$-dimensional column vector. Because raw intensity values of a gray scale image patch might not be consistent even in the same class, we use three different pieces of information to inject invariance into our parking spot patch representation: intensity statistics (such as mean, variance, smoothness, skewness, uniformity, and entropy), responses of the Radon transform, and local histograms of oriented gradients (HOG) [Dalal and Triggs, 2005]. In the next section, we compare the performance of hypothesis filters trained using these features versus using the raw pixel-intensity patched directly.

Our experiments compare four machine learning techniques as hypothesis filters for this binary classification task: Support Vector Machines (SVMs), Eigenspots, Markov Random Fields (MRFs), and Bayesian Linear Regression (BLR).

Support Vector Machines SVMs are a common supervised learning algorithm for binary classification. They seek to find the hyperplane that maximizes a notion of margin between each class [Vapnik, 1995]. Linear SVMs are fast, have publicly available implementations, and handle high-dimensional feature spaces well. This algorithm and its variants have been extensively used

as *de factor* object detection algorithms.

Eigenspots Since processing high-dimensional image patches is computationally expensive, we reduce the dimensionality of our vector space by using principal component analysis (PCA) [Bishop, 2006] to find the principal subspace of the self-supervised parking spots obtained by the low-level analysis; we retain the top k dimensions of the original vector space, where $k \ll m$. In homage to Turk and Pentland [Turk and Pentland, 1991], we call the eigenvectors of the parking spot space extracted by this method the "Eigenspots" of the space.

We use this new space in two ways. Our first technique simply measures the distance from a candidate patch to the center of the space (i.e. the mean canonical parking spot, $\mathbf{\Psi}$). Given a new image patch $\mathbf{g}$, we compute, $T(\mathbf{g}) = \|\mathbf{D}^{-1/2}\mathbf{E}^T(\mathbf{g} - \mathbf{\Psi})\|^2$ where $\mathbf{\Psi} = \frac{1}{\text{number of positives}} \sum_i \mathbf{g_i}$, $\mathbf{D}$ is a diagonal matrix containing eigenvalues $\lambda_1, ...\lambda_k$, and $\mathbf{E}$ is a matrix whose columns are the eigenvectors of the covariance matrix used in the PCA computation. $T(\mathbf{g})$ is also known as the Mahalanobis distance [Bishop, 2006] from the origin of the Eigenspot space. If this distance is less than a threshold, we classify the new image patch as a parking spot. Our second usage simply pushes the examples through the PCA transformation before training a SVM classifier and learning a mixture of multivariate Gaussian distributions. Specifically, we transform each example as $\widetilde{\mathbf{g}} = \mathbf{D}^{-1/2}\mathbf{E}^T(\mathbf{g} - \mathbf{\Psi})$.

Pairwise Markov Random Fields. Because SVMs and Eigenspots only consider the local characteristics of an image patch to perform binary classification, their performances are limited by the distribution of the training data. Thus it is useful to consider neighboring image patches around the patch of interest as well as looking at characteristics of the image patch. An image patch is highly likely a parking spot when the majority of neighboring patches are parking spots, even if the local characteristics of the patch would classify it otherwise.

To implement this idea, we use a pairwise Markov Random Fields (MRFs) [Li, 2000]. A pairwise MRF, $\mathcal{H}$, is an undirected graphical model that factorizes the underlying joint probability distribution $P(Y, G)$ by a set of pairwise cliques. [2] $\mathcal{H}$ is comprised of a set of nodes and their edges where a node models a random variable and the edge between nodes represents dependence between them.

In this work, there are two different types of nodes: observed and unobserved nodes. An observed node corresponds to an image patch whereas an unobserved node is the true label of the observed node. Although we observe the value of a node ($G_k = \mathbf{g}_k$), the true label of the node ($Y_k = y_k \in \{-1, +1\}$) is not observed. The task is then to compute the most likely values of Y (i.e. whether a hypothesis ($\mathbf{g}_i$) is a parking spot ($y_i = 1$) or not) given the structure of the undirected graph, $\mathcal{H}$, and characteristics of image patches, G. The joint probability distribution is factorized as

$$P(Y, G) = \frac{1}{Z} \prod_{i=1}^{N} \Phi(G_i, Y_i) \prod_{j \in N(i)} \Psi(Y_i, Y_j)$$

where $\Phi(G_i, Y_i)$ is a node potential, $\Psi(Y_i, Y_j)$ is an edge potential, Z is the partition function that ensures a probability density of this model, $N(i)$ is the set of nodes in the neighborhood of the ith node. Our implementation of MRFs considers first-order neighbors.

[2] There may be bigger cliques in the graph, but the pairwise MRF only consider pairwise cliques.

Since we assume that candidate parking spots are generated from a mixture of multivariate Gaussian distributions, we estimate the node potentials using a Gaussian Mixture model (GMM) [Bishop, 2006]. Due to the possibility of two class labels, each node has two potentials: a potential of being a parking spot, $\Phi(G_i, Y_{j=+1})$ and the potential of not being not a parking spot, $\Phi(G_i, Y_{j=-1})$. The edge potential is computed by Potts model [Li, 2000].

$$\Psi(Y_i, Y_j) = \psi(Y_i, Y_j) = \exp\left\{-\beta(Y_i - Y_j)^2\right\}$$

where β is a penalty factor for label disagreement between nodes. In particular, if $\beta = 0$, edge potentials are identical regardless of the label disagreement and only node potentials are used. On the contrary, if $\beta = \infty$, only the edge potentials are meaningful and the node potentials are ignored. For inferencing the most likely labels of individual parking spot hypotheses in a given aerial image, we use loopy belief propagation because it is easy to implement [Yedidia et al., 2002].

Bayesian Linear Regression Our self-supervised canonical parking spots are highly accurate, but their number is often too few to generalize. To remedy this insufficient number of positive examples, we use canonical parking spots previously obtained from other aerial images. As will be shown in the experimental results, this approach helps our hypothesis filters improve their performances. However, naively consuming all the available data might result in a solution that is overfit. Thus to effectively utilize data, we employ Bayesian linear regression (BLR). BLR provides a theoretical way of incorporating previously obtained parking spot templates as a prior information for the optimal weight vector learning. The optimal weight vector, $\mathbf{w}^*$, is obtained by

$$\begin{aligned}
p(\mathbf{w}^*|\mathbf{G}) &\propto \arg\max_{\mathbf{w}} p(\mathbf{G}|\mathbf{w})p(\mathbf{w}) \\
p(\mathbf{G}|\mathbf{w}) &= \prod_{i=1}^{N} p((\mathbf{g}_i, y_i)|\mathbf{w}) \propto \exp\left\{\frac{1}{2\sigma^2}\sum_i (y_i - \mathbf{w}^T\mathbf{g}_i)^2\right\} \\
p(\mathbf{w}) &\propto \exp\left\{-\frac{1}{2}\mathbf{w}\boldsymbol{\Sigma}^{-1}\mathbf{w} + \mu^T\boldsymbol{\Sigma}^{-1}\mathbf{w}\right\}
\end{aligned}$$

where $p(\mathbf{G}|\mathbf{w})$ is the likelihood function and $p(\mathbf{w})$ is the prior distribution that is a zero-mean Gaussian. The final form of BLR is a regularized linear regression where the parameters of the resulting conditional Gaussian distribution of $\mathbf{w}^*$ given data D is

$$\begin{aligned}
\boldsymbol{\Sigma}_{\mathbf{w}|\mathbf{D}} &= \left(\mathbf{G}\mathbf{G}^T + \lambda\mathbf{I}\right)^{-1} \\
\mu_{\mathbf{w}|\mathbf{D}} &= \left(\mathbf{G}\mathbf{G}^T + (\sigma^2\lambda)\mathbf{I}\right)^{-1}\mathbf{Y}\mathbf{G}
\end{aligned}$$

where λ is a regularizing term that controls contributions of the weight prior. We classify an image patch as positive if the regression value is greater than the predefined threshold,

$$h(\mathbf{g}_i) = 2I\left[y(\mathbf{g}_i) \geq \delta\right] - 1, \delta \in \mathbb{R}.$$

where $y(\mathbf{g}_i) = \mathbf{g}_i^T\mathbf{w}^*$ is the output of BLR and $I\left[y(\mathbf{g}_i) \geq \delta\right]$ is an indicator function that returns 1 if $y(\mathbf{g}_i)$ is greater than δ, otherwise 0.

Experimental Results

The knowledge of the image coordinates of parking spots facilitates estimation of parameters that describe the structure of a parking lot. Thus the purpose of our experiments is to measure how well our filtering methods perform in detecting all the visible parking spots in an aerial image.

We use twenty aerial images collected from the *Google* [3] map service. There are on average 116 visible parking spots in each individual image in different shapes and under different illumination conditions and a total of 2,324 parking spots across all aerial images.

	false negative	*false positive*	*accuracy*
Self-supervised Parking Spots	0.5512	0.0471	0.7008
Generated Hypotheses	0.3719	0.9382	0.2311

Table 5.1: Performance comparison of parking spot hypotheses generated by the low-level and high-level analysis layers is measured by three different performance metrics such as "false negative," "false positive," and "accuracy."

Table 5.1 shows the micro-averaged performance of the generated hypotheses by the low-level and the high-level analysis. The accuracy is defined as a ratio of the number of correctly classified parking spots to the total number of parking spots used in evaluation. This micro-averaged performance is computed by merging contingency tables across the twenty different images and then using the merged table to compute performance measures. Since the self-supervised examples are highly accurate (a low false positive rate (4.71%)), their parking spot templates can be used as positive examples for training all filtering methods. An equal number of negative examples are randomly generated.

In this work we are particularly concerned about false positives since, in the worst case, a false positive output might make a robotic vehicle drive somewhere that the robot should not drive. While generating few false positives, the self-supervised parking spot detector recover only 43.55% of the true parking spots (1,012 out of 2,324 true parking spots over 20 images.) This high false negative rate [4] may cause problems for autonomous driving: for example, an autonomous robotic vehicle might not be able to park itself even if there are plenty of parking spots available. By using information provided by the low-level analysis, the high-level hypothesis generation analysis reduces the false negative rate from 55.12% to 37.19%. However, it increases the false positive rate to 93.82% as well (i.e., it predicts many spots which are not true spots). The filtering stage then corrects this shift in false positive rate by removing erroneous hypotheses. Importantly, as we will see in the results, this technique cannot recover from false negatives in the hypothesis generation.

Table 5.2 compares the performance of self-trained filtering methods. The parking spot

[3] http://map.google.com

[4] A false negative is a parking-spot example that is classified as a non-parking-spot example.

	false negative	*false positive*	*accuracy*
SVMs	0.3880 ± 0.0012 (0.0188)	0.3136 ± 0.0106 (-0.0230)	0.6627 ± 0.0012 (0.0073)
Eigenspots	0.3074 (0.0090)	0.8004 (0.1085)	0.3013 (-0.0880)
SVMs w/ Eigenspots	0.3826 ± 0.0221 (-0.0116)	0.3227 ± 0.0201 (-0.0256)	0.6603 ± 0.0109 (0.0200)
MRFs w/ GMM	0.3929 ± 0.0301 (-0.0147)	0.3644 ± 0.0041 (-0.0098)	0.6280 ± 0.0074 (0.0101)
BLR	0.3270 ± 0.0009 (0.0184)	0.6611 ± 0.0129 (-0.1007)	0.4091 ± 0.0070 (0.1238)
SVMs	0.4271 ± 0.0350 (-0.0186)	**0.0429 ± 0.0112 (-0.0086)**	0.9189 ± 0.0012 (0.0095)
Eigenspots	**0.2765 (0.0000)**	0.3969 (0.0196)	0.6151 (-0.0176)
SVMs w/ Eigenspots	0.4320 ± 0.0111 (-0.1142)	0.0450 ± 0.0276 (-0.0143)	0.9165 ± 0.0012 (0.0242)
MRFs w/ GMM	0.3466 ± 0.0786 (-0.1846)	0.0798 ± 0.0145 (0.0110)	0.8937 ± 0.0243 (0.0085)
BLR	0.4136 ± 0.0313 (-0.0099)	0.2827 ± 0.0241 (0.0151)	0.7043 ± 0.0232 (-0.0121)
SVMs	0.3951 ± 0.0345 (0.0113)	0.0457 ± 0.0012 (-0.0105)	**0.9213 ± 0.0111 (0.0085)**
Eigenspots	0.2765 (0.0000)	0.3759 (0.0144)	0.6335 (-0.0130)
SVM w/ Eigenspots	0.3880 ± 0.0011 (-0.0567)	0.0486 ± 0.0012 (-0.0165)	0.9194 ± 0.0042 (0.0204)
MRFs w/ GMM	0.3342 ± 0.0188 (-0.1318)	0.0817 ± 0.0012 (0.0126)	0.8945 ± 0.0174 (0.0011)
BLR	0.3970 ± 0.0114 (-0.0105)	0.2712 ± 0.0005 (0.0098)	0.7169 ± 0.0011 (-0.0079)

Table 5.2: Results comparing different filtering methods. The numbers in parentheses indiciate the performance difference between different parking spot patch representations. Positive values in the accuracy indicate improvements of our feature representation over raw-pixel intensity whereas negative values in false positive and negative columns indicate improvements. Overally, the performance difference is negligible, but our feature representation method enables our filtering algorithms to reduce the dimension (m) of parking spot patches' from 240 to 93, resulting in computationally more efficient solution (i.e., faster training with less memory).

hypotheses generated by the high-level layer were labeled by hand for evaluation. Hyper-parameters of SVMs were determined by 10-fold cross validation.[5] Eigenspots are computed only using positive examples. For the MRF inference, we build a mesh from the estimated layout of parking spot hypotheses where a node in the grid corresponds to an image patch. We use positive and negative examples to obtain GMM and use the obtained GMM to estimate node potentials. We observe the results by varying β in the range 0 to 10 with steps of size 2.[6] We empirically set 2 as β for the MRFs, 5 as λ and .5 as a threshold for binary classification for the BLR implementations.

In the table 5.2, there are three blocks of rows describing three different experimental scenarios. In the first scenario, we trained the filtering methods using parking spot templates from the image under analysis consisting of the self-supervised parking templates as positive examples and randomly generated negative examples. In the second scenario, we trained these methods using self-supervised examples from all other images not including the target image. Finally, in the last scenario we trained the methods using self-supervised examples from all images. The randomly generated negative examples were sampled while running each of these scenarios. Due to this randomness in negative examples, we averaged our results over 5 separate runs for each scenario. Each cell in the table displays the mean and standard deviation.

[5]For SVM implementation, we use libsvm which is publicly available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm/

[6]We fit our Gaussian Mixture model using the publicly available GMMBayes from http://www.it.lut.fi/project/gmmbayes/

In addition, we wanted to measure the usefulness of our feature representation over raw-intensity parking spot patches. We re-ran the above experiments using the same parking patches in raw-intensity values. The numbers in parentheses indicates the performance difference between different parking spot patch representations. Positive values in the accuracy indicate improvements of our feature representation over raw-intensity whereas negative values in false positive and false negative columns indicate improvements. Overall, the performance difference is negligible, but our feature representation method enables our algorithms to reduce the dimension (m) of parking spot patches' from 240 to 93, resulting in computationally more efficient solution (i.e., faster training with less memory).

Ideally, the method with the lowest false positive and negative rates would be the best, but in practice it is hard to achieve both of them simultaneously. For our autonomous driving application, we prefer the method with the lowest false positive to one with lowest false negative because a false positive is more risky than a false negative. In general, the performances of hypothesis filters are improved as the amount of training data is increased. Linear SVMs performed surprisingly well, particularly in terms of false positives and accuracy. Additionally, training an SVM using the subspace generated by the Eigenspots analysis performs only marginally better than simply using the Eigenspot distance measure computation. This performance difference can potentially be decreased by statistically fitting the threshold value used during distance measure classification. As discussed earlier, MRFs utilize higher-level interactions to improve prediction accuracy. However, estimating the GMM requires a substantial amount of data; the performance degradation in the first row of the table indicates that the canonical parking spots extracted by the low-level analysis alone were too few to accurately fit this model.

5.2 Recognizing Parking Lot Drivable Region

In this thesis, *skeletonization* refers to a process of extracting the skeleton of drivable regions in a parking lot image. To accurately build a skeleton, we need to know the structure of a parking lot. This is done by estimating boundaries of parking blocks that are obtained from parking spot detection. In parallel, we segment a given aerial image into two regions: "parking lot" and "non-parking lot" regions. Then the drivable regions in a parking lot are recovered by superimposing the constructed structure over the segmented parking lot image. In this step, we use self-supervised examples to find cues for parking lot for the boundary segmentation and road-marking classification.

The following sections detail how self-supervised examples are used in segmenting the parking lot boundary and in detecting road-markings.

5.2.1 Parking Lot Boundary Segmentation

The flood-fill algorithm is a technique to fill connected regions within an image with a constant value. We assume that the magnitude of image gradient in drivable regions is similar to those of parking spots. After computing magnitudes of the image gradient, we randomly select some of the self-supervised parking spots and use them to obtain a threshold value. The centroids of those selected parking spots are used as starting points. Despite its simplicity, our modified flood-fill

algorithm works reasonably well in that it detects all the visible parking lot regions in our test images. Figure 5.4(a) shows a binary image of the segmented drivable region that is indicated in white.

5.2.2 Road-Markings Detection

Road-markings are important parts of drivable regions and differentiated from other drivable regions by their intensity and color histograms. To detect road-markings in a parking lot image, we train a binary road-marking classifier that assigns a pixel as either a road-marking ($+1$) or non-road-marking (-1). To obtain a training set, we utilize road-markings that are parts of the self-supervised parking spot templates. A set of the randomly selected road-markings are used to learn characteristics of road-marking in a particular parking lot. We use Bresenham's line algorithm to select pixels along the selected lines and learn a multivariate Gaussian distribution of two different color spaces: Hue-Saturation-Intensity (HSI) and RGB, in which individual pixels are represented by six-dimensional vectors, $\mathbf{x}_i \in \mathbb{R}^6$. [7]

$$p(\mathbf{x}_i|C_k) = \frac{1}{(2\pi)^{d/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}_i - \mu_k)^T \mathbf{\Sigma}_k^{-1}(\mathbf{x}_i - \mu_k)\right\}$$

where $p(\mathbf{x}_i|C_k)$ is a conditional probability of $\mathbf{x}$ given $C_k, k \in \{-1, 1\}$, d is the dimension of a pixel vector, $\mathbf{\Sigma} = d \times d$ is kth class' covariance matrix, and $\mu = d \times 1$ is kth class' mean vector. We learn another Gaussian distribution for non-road-marking class. The road-marking detection is done by investigating the likelihood ratio between two classes:

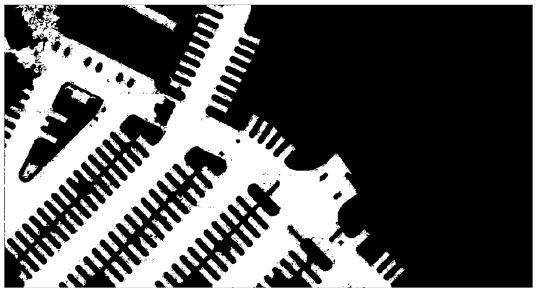
$$y(\mathbf{x}_i) = \begin{cases} 1 & \text{if } \log\left(\frac{p(\mathbf{x}_i|C_1)}{p(\mathbf{x}_i|C_{-1})}\right) > 0 \\ -1 & \text{otherwise} \end{cases}$$

A result of the road-marking classification is shown in Figure 5.4(b). Note that there are a number of false positives along road lanes outside of the parking lot. These errors occur because the magnitudes of road lane are similar to those of parking spots. However, since the detection result is used in conjunction with other results (i.e., parking lot boundary segmentation and parking spot detection) to build the skeleton of drivable regions, it is acceptable to include some of the non-parking lot regions.

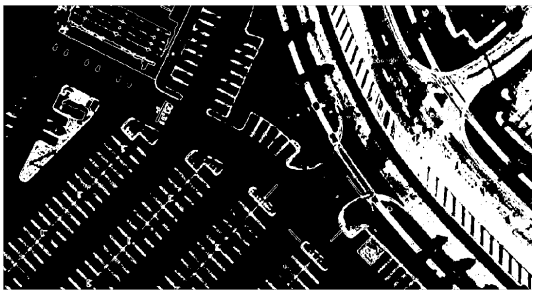
5.2.3 Drivable Region Identification

There are three inputs for identifying drivable regions: results of parking spot detection, results of road-marking detection, and results of parking lot boundary segmentation. Although none of these inputs is perfect, a combination of these imperfect inputs works reasonably because they are complementary to each other. For example, our road-marking detection method produces a number of false positives on road lanes (See Figure 5.4(a)), but during the drivable region identification phase, these high false-positive regions are disregarded because they are located

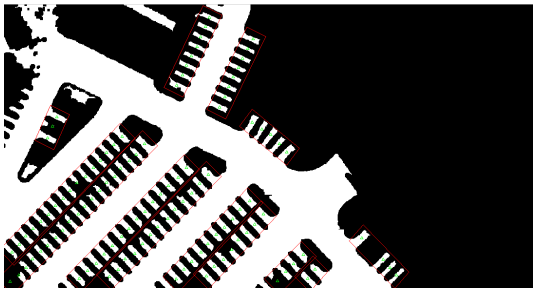
[7] We utilized other color spaces such as Lab and YCbCr and found that a combination of HSI and RGB works best.



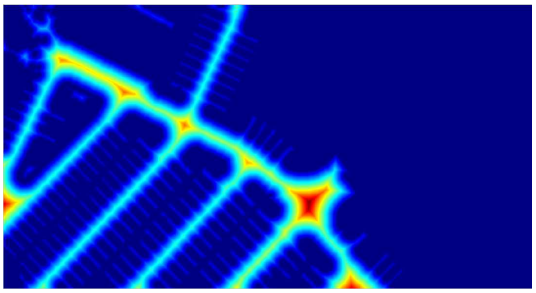
(a) The detected boundary of a parking lot.



(b) The results of the road-marking detection.



(c) The structure of a parking lot is superimposed over parking lot boundary segmentation. The red rectangles represents boundaries of parking blocks whereas the green triangles are detected parking spots.



(d) The final results of skeletonization. Safety of traversability is color-scaled for visualization purpose. Red corresponds to highest safety of traversability whereas blue is lowest safety of traversability.

Figure 5.4: The figure at bottom right shows the final result of the skeletonization and all other figures are inputs for the skeletonization process.

outside of the parking lot based on the result of parking lot boundary segmentation result (See Figure 5.4(b)).

Based on the best result of parking spot detection, the structure of parking lot is uncovered by computing boundaries of parking blocks. This structure can roughly tell us what the geometric shape of a parking lot looks like, but cannot tell where exactly an autonomous vehicle should drive. To define drivable regions of a parking lot, the parking lot structure is superimposed over the segmented parking block boundaries. Then drivable regions of a parking lot become clearly visible to the vehicle. However, the binary image of drivable regions shown in 5.4(a) still has some errors. Although these black speckles do not look significant in the image, they may cause serious problems when used for autonomous driving as they may be regarded as obstacles. To remove these errors, we apply a morphological operation ("close") to smooth the segmentation binary image. Since the smoothing can only remove small-size speckles, we implement heuristics to remove islands in the drivable regions. These islands are in fact road-markings (e.g., stop-lines, driving direction marking) on the drivable region in the original image. While these features represent important contextual information, we want to remove them from

the description of the drivable regions. To remove these islands, we utilize the results of road-marking detection. That is, for each of the islands in drivable regions, it can be removed if an island does not belong to road-markings that are parts of parking blocks. Figure 5.4(c) shows the binary image of drivable regions after removing speckles and islands. Finally to accurately depict boundaries of drivable regions, we apply a modified "brushfire" algorithm that incrementally propagates distance values from non-drivable regions (e.g., parking blocks). Figure 5.4(d) shows the result of the skeletonization that depicts drivable regions in a parking lot image.

5.3 Lane-Graphs for Parking Lot Map Generation

This section details an algorithm that produces a lane-graph of drivable regions by analyzing a single parking lot image. From the previous section, drivable regions in a parking lot are determined by a combination of results of parking spot detection and results of parking lot boundary segmentation. A distance transform is used to reveal the skeleton of drivable regions' geometry. Our lane-graph generation algorithm iteratively searches for a lane-graph in the distance transform map that concisely represents drivable regions.

5.3.1 Topological Map of Drivable Regions in Parking lot

In this thesis, a lane-graph of a parking lot refers to a topological representation of drivable regions. Our lane-graph generation algorithm requires a map of non-drivable (or an obstacle map) as an input. We built this map by using two aerial image analysis results from the previous sections. In particular, we developed several different orthoimage analysis algorithms to detect all of the visible parking spots in a parking lot orthoimage. Our self-labeling method analyzes the spatial layout of extracted lines and automatically obtains some of the easy-to-detect true parking spots. These self-labeled parking spot image patches are used for several purposes. First, the geometric properties of self-labeled examples, such as average length, width and distance between them, are used to generate hypotheses that are predictions of the true parking spot locations. Second, these self-labeled parking spot images are used to train a binary classifier to filter out incorrect hypotheses. Lastly, the image characteristics of self-labeled examples are used to learn a road-marking classifier and a parking lot boundary segmentor[Seo et al., 2009b]. The image regions of the estimated parking lot boundary are overlapped with the detected parking spots to produce the map of non-drivable regions in a parking lot orthoimage.

5.3.2 Connecting Maximal Circles for Discovering Topology of Lane-Graph

Algorithm 1 describes the procedure of our lane-graph generation in detail.

The algorithm requires an obstacle map of a parking lot image. This map of non-drivable regions, $I_{non-drivable}$, is obtained by combining parking spot detection results and parking lot boundary segmentation results. Figure 5.5(a) shows examples of parking spot detection results and parking lot boundary segmentation results.

Algorithm 1 Lane-graph generation algorithm.

Require: - I, a parking lot orthoimage,
- $I_{non-drivable}$, a binary image of non-drivable regions

Ensure: - $\mathbb{G}$, a lane-graph that maximally covers drivable regions in the parking lot

1: $\mathbb{G} = \{\mathbf{V}, \mathbf{E}\}, \mathbf{V} = \{\phi\}, \mathbf{E} = \{\phi\}$
2: I_{dt} = *generate_distance_map*($I_{non-drivable}$)
3: $\mathbb{M}$ = *find_local_maxima*(I_{dt}), $\mathbb{M} = \{m_1, ..., m_{|\mathbb{M}|}\}$
4: $\mathbb{M}_1$ = *prune_local_maxima*($\mathbb{M}$), $|\mathbb{M}_1| \ll |\mathbb{M}|$
5: $\mathbb{M}_1$ = *sort*($\mathbb{M}_1$), $m_1 > m_2 >, ..., > m_{|\mathbb{M}_1|}$
6: $\mathbb{M}_2$ = *define_rendezvous_point*($\mathbb{M}_1$), $|\mathbb{M}_2| \ll |\mathbb{M}_1|$
7: **repeat**
8: mc_i = *find_maximal_circle*$(m_i), m_i \in \mathbb{M}_1$
9: Remove all of the local maxima within the circle, mc_i
10: Create a vertex, $v \leftarrow m_i$,
11: $\mathbf{V} = \{\mathbf{V} \cup v\}$
12: **until** all of the rendezvous points, $\mathbb{M}_2$, are visited
13: **for all** $v_i \in \mathbf{V}$ **do**
14: Identify neighboring vertices, $N(i)$, of v_i
15: Create an edge, e_{ij}, if the jth neighboring vertex,$v_j \in N(i), (i \neq j)$, is visible from the $i$the vertex, v_i.
16: $\mathbf{E} = \{\mathbf{E} \cup e\}$
17: **end for**
18: **Return** $\mathbb{G} = \{\mathbf{V}, \mathbf{E}\}$

A distance transform is often applied to a robot's operational environment for identifying obstacle-free regions. In our case, the function, *generate_distance_map*($I_{non-drivable}$), implements the brush-fire algorithm to propagate distance values from non-drivable regions. Figure 5.5(b) depicts the resulting distance map where farthest points from local obstacles have highest values. Ridge points on the distance map, which are local maxima of the map, are good candidates for building a lane-graph because they are skeleton points of drivable regions.

To locate these local maxima, we use the discrete analog of derivative because the second derivative of a distance map function is zero when the magnitude of the derivative is extremal. To implement this idea, our search is carried out by investigating individual columns and rows in the distance map, I_{dt}. The distance map, I_{dt}, is a m-by-n real-valued matrix where $I_{dt}(i, j)$ is the distance transform value of the ith row and the jth column. The function, *find_local_maxima*(I_{dt}), computes numerical derivatives of individual columns (and rows) in the distance map using the forward difference, $dx_i(k) = I_{dt}(i, k+1) - I_{dt}(i, k), \quad (dy_j(k) = I_{dt}(k+1, j) - I_{dt}(k, j))$, where $k = 1, ..., n-1$ [8]. This is the first derivative of the distance map that locates the changes of distance values in a column (or a row). We then compute the second derivative: Compute the forward difference again only for elements which all of the $dx_i(k)$ are greater than zero. The value of the second derivative is zero when the distance value is a local extremum. A local

[8]For example, computing the changes of distance values at the ith row, $dx_i(2) = I_{dt}(i, 2) - I_{dt}(i, 1), dx_i(3) = I_{dt}(i, 3) - I_{dt}(i, 2), ..., dx_i(n-1) = I_{dt}(i, n) - I_{dt}(i, n-1)$

extremum is a maximum if the slope of the second derivatives at its neighbor points is changed from negative to positive.[9] These points correspond to peaks in a column (or row) of the distance map. A cross-check of these points with other rows and diagonal elements in the matrix results in local maxima. Because of the discrete nature of an image, this method does not always guarantee to find all of the true extrema, but provide a sufficient number of local maxima for our lane-graph generation.

A simple connection of all of the detected local maxima might produce a lane-graph that has an unnecessarily detailed structure due to an imperfect obstacle map. For example, there is a small creek in the upper left corner of the figure 5.5(b) that depicts cracks of the obstacle map and causes the distance transform to produce a lot of small-valued local maxima. Therefore we should handpick some of the local maxima for constructing the topology of a lane-graph.

To facilitate the lane-graph building process, there are three initialization steps: elimination of irrelevant local maxima; sorting of the selected maxima; and generation of rendezvous points. The function, *prune_local_maxima*($\mathbb{M}$), remove any local maxima that the radius of its maximal circle is smaller than the average width of the detected parking spots because their surrounding regions are not wide enough for the navigation of a common-size vehicle. To define the maximal circle of a local maximum, the initial radius of an inscribed circle is set to the average width of the detected parking spots. The radius is increased until the circle touches any of neighboring obstacles. An inscribed circle is maximal if no other inscribed circle, without touching neighboring obstacles, contains it properly. The idea of maximal circle has been studied for shape recognition and abstraction [Kimmel et al., 2003]. Figure 5.5(c) depicts a set of the selected local maxima. A sorting of the selected local maxima in descending order of their distance map values is necessary because the surrounding region of a local maximum with higher value contains more important geometric structure in a parking lot and it should be considered before any other local maxima with smaller values. Lastly we need a criterion to determine when to stop our topology building step. One might think this iteration can be stopped when it connects all of the selected local maxima. However because of incomplete boundary segmentation result, a connection of all of the selected local maxima will result in a lane-graph that is inconsistent to the actual shape of drivable regions. To properly stop the iteration while ensuring the consistency of a resulting graph, we utilize the locations of some local maxima. We call them rendezvous points because their locations must be visited for building a consistent lane-graph. A rendezvous point is a local maximum point that represents more than one detected parking spot. Given the fact that our parking spot detection algorithm recovers the open-end orientation of a parking spot, in the function, *define_rendezvous_point*($\mathbb{M}_1$), for each of the detected parking spots, we find the local maximum point that is orthogonally closest to that parking spot. These local maxima points are orthogonal projections of the detected parking spots onto the center-line of drivable regions.

Although the collection of rendezvous points does not always cover all of the area of the drivable regions, the ordering of the selected local maxima based on their values ensures that the topology visiting all of the rendezvous points completely aligns with the area of drivable regions. Thus the topology of a lane-graph is consistent to the shape of drivable regions if it includes all

[9]One can also find a local minimum by looking at the point where its second derivative is zero and the slope is changed from positive to negative. In practice, this extrema search can easily be done by convolving individual columns (or rows) with a Laplacian operator, $[1, -4, 1]^T$ and then looking for the changes of slopes.

of the rendezvous points. Figure 5.5(c) depicts the identified rendezvous points.

Once these initialization steps are completed, our algorithm examines each of the selected local maxima by investigating its surrounding region. In particular, for each of local maxima, the function, *find_maximal_circle*(m_i), defines a maximal circle centered at the local maxima under investigation. Any local maxima within the maximal circle can be removed from further consideration. These steps are repeated until all of the rendezvous points are visited. Figure 5.5(d) depicts the selected local maxima. In this example, there are 402 local maxima initially identified and 55 of them are selected as vertices for possible lane-graphs.

The last step of our algorithm is to connect each of the identified vertices to neighboring vertices if the vertex is visible from its neighbors. The visibility test checks whether a line segment between two vertices passes through any obstacles and any neighboring vertices. In particular, we use the Bresenham algorithm to examine image coordinates along the line linking two vertices whether they are overlapped with any non-drivable regions.

5.3.3 Results of Lane-Graph Generation

Figure 5.6 shows some results of our lane-graph generation algorithm. Testing orthoimages are downloaded from the *Google* map service.[10]

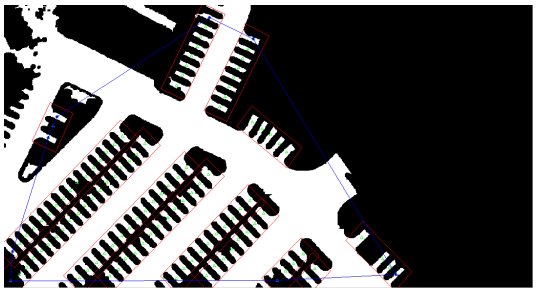
For each of the testing images, we first execute our parking spot detection and parking lot boundary segmentation algorithms to produce the map of non-drivable regions in the image, and then run our lane-graph generation algorithm.

For most test images, our algorithm works well in that the topologies of resulting lane-graphs concisely represent drivable regions in parking lot images: edges align with the center lines of road segments and vertices at intersection points connect merging road segments. Results shown in figure 5.5(f), 5.6(a), 5.6(b), 5.6(c), and 5.6(d) are example images of successful cases. These successful results rely on the map of non-drivable regions: highly accurate results of parking spot detection and parking lot boundary segmentation. Any false positive result by either of these tasks overestimates the true area of drivable regions in a parking lot image. For example, in the figure 5.6(a), the shadows of trees are segmented as non-drivable regions and the resulting edge passing through that region is bended to avoid false non-drivable regions. A drivable region with different visual appearances such as shadows and occlusions is one of the common causes for erroneous segmentation and parking spot detection results. Because we developed a very accurate parking spot detector that produces a very small false positive errors (less than 0.06%), most of the overestimated drivable region is caused by parking lot boundary segmentation.

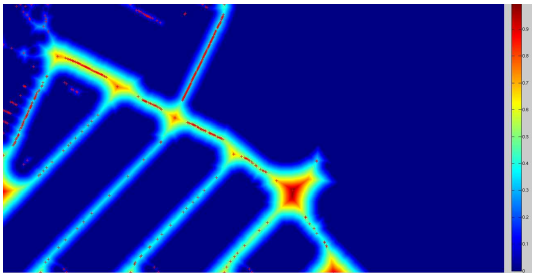
Figure 5.7 shows an example that our algorithm did not work well. The segmentation result in the figure 5.7(a) produces the overestimated boundary, resulting in a lane-graph inconsistent to the actual drivable regions. Owing to the relatively accurate parking spot detector, the right side of drivable regions in the figure 5.7(b) is partially covered by the resulting lane-graph. The lane-graph is inconsistently generated primarily due to the simplicity of our segmentation algorithm in that it connects two neighboring pixels if their image characteristics (i.e., magnitudes of image gradients and color) are similar. Thus it fails to correctly segment regions when the appearance of pixels greatly vary.

[10] http://map.google.com

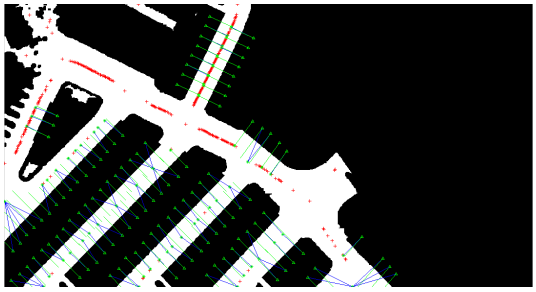
By contrast, a false negative one underestimates the true area of drivable regions in a parking lot image. For example, in the figure 5.6(c), the edge of the intersection at the bottom left is passing through a part of the non-drivable regions because the parking block is completely missed by our parking spot detection algorithm.



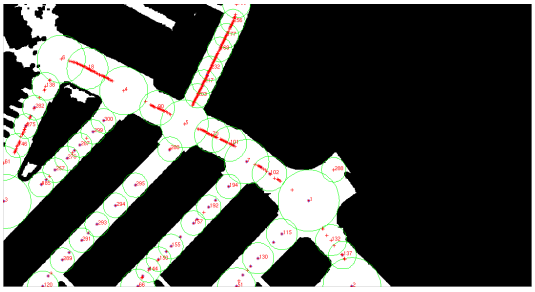
(a) Two inputs for lane-graph generation: Parking block polygons depicted in (red) rectangles and parking lot boundary segmentation depicted as a binary image. Individual parking spots are depicted (green) triangles at their centroids. A blue line is a convex hull of all of the detected parking spots.



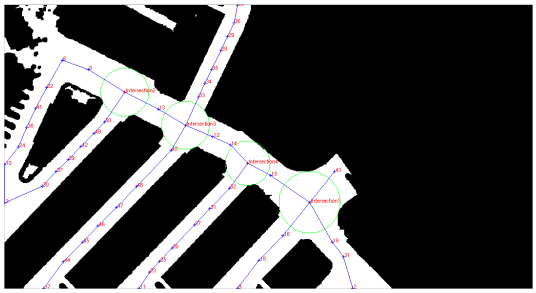
(b) A distance map is computed from the obstacle map of a parking lot. Red regions are farthest ones from local obstacles depicted in blue. Red "x" marks represent local maxima in the distance map.



(c) Some of the local maxima are selected as rendezvous points that are used to determine when the search of a lane-graph topology stops. Green "*" marks rendezvous points.



(d) The radius of a circle centered on a local maximum is increased to find its maximal circle. Any local maxima within the maximal circle will be considered redundant and removed.

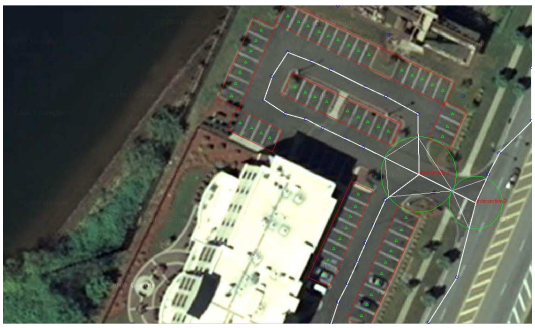


(e) Connection of visible neighboring vertices reveals the topology of a lane-graph.

(f) The output of our lane-graph generation algorithm. A vertex is represented as an intersection if its edges are more than 2.

Figure 5.5: These figures show the sequence of our lane-graph generation algorithm.

(a) The parking lot area is underestimated due to the shadows of trees. As a result, some edges of the resulting lane-graph do not align with the center-lines of drivable regions. There are only 37 (depicted as blue circle) out of 472 local maxima used as the vertices of the resulting lane-graph.

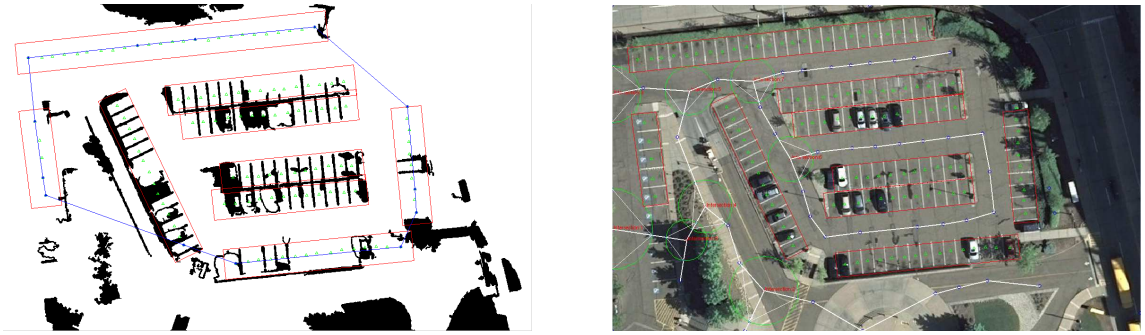


(b) Inaccurate segmentation results in extending the resulting lane-graph to the outside of the parking lot. There are 31 out of 430 local maxima used as vertices.

(c) Some of the parking spots at the bottom left are missed by the parking spot detection. This cause our algorithm to overestimate the actual parking lot area. There are 43 out of 428 local maxima used as vertices.

(d) There is an isolated vertex at the top left because of inaccurate segmentation. There are 35 out of 298 local maxima used.

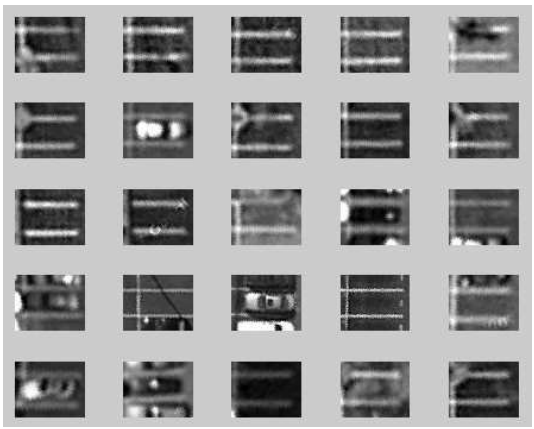
Figure 5.6: Four additional examples of lane-graph generation.



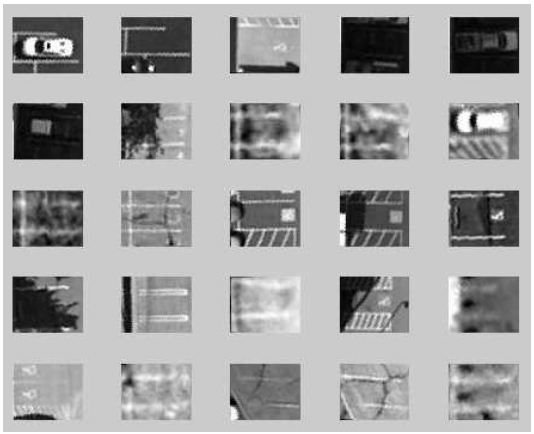
(a) A fail case. An overestimated parking lot area.

(b) Due to the inaccurately estimated parking lot boundary, the resulting lane-graph fails to capture the topology consistent to drivable regions.

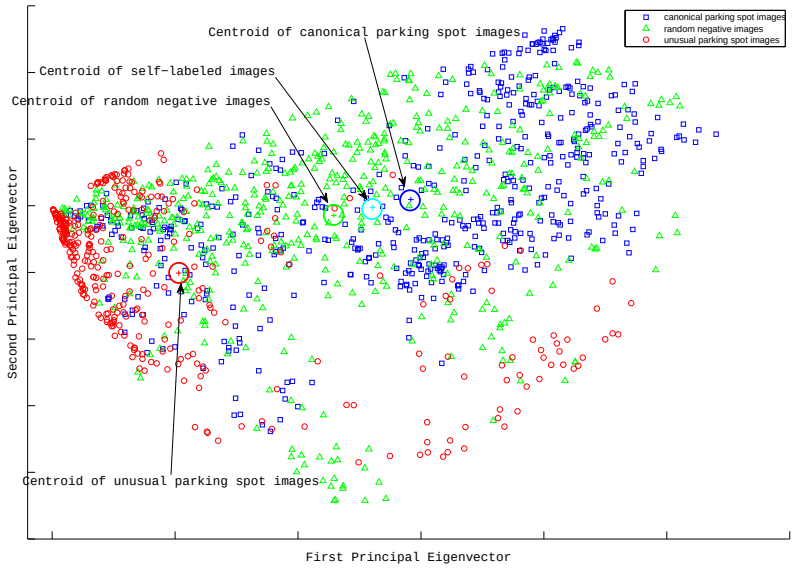
Figure 5.7: The topology of the resulting graph is sensitive to the accuracy of non-drivable map.



(a) Examples of parking spots with canonical appearances.



(b) Examples of parking spot images with unusual appearances.



(c) A scatter plot of (canonical and unusual) parking spot images and randomly generated negative images' 2-dimensional representation. All (blue) squares represent a canonical parking spots, (green) triangles represent negative images, and (red) circles represent unusual parking spots. There are 532 canonical parking spots, 532 negative examples, and 365 unusual parking spots.

Figure 5.8: A two-dimensional representation of parking spots with varying appearances.

5.4 Incremental Learning for Handling Intra-Class Variation

In automatically building a roadmap for autonomous driving from orthoimagery, it is important to reliably detect relevant objects in imagery because knowledge of objects' (e.g., parking spots) image coordinates facilitates the recognition of underlying structures (e.g., the geometric structure of a parking lot). Learning a reliable object detector is challenging because appearance of objects in aerial imagery varies primarily due to the illumination conditions and the object properties.

There are two ways to deal with this intra-class variation problem. One is to collect a tremendous amount of human-labeled orthoimage data that is assumed to cover all (or at least a large portion of) of the possible appearances; train a detector by using this data; and use the learned detector as needed. However it is very difficult and expensive (maybe even impossible) to prepare such data in advance. Another way to handle this appearance inconsistency problem is to collect orthoimage data around target areas, learn a detector by using this small, but targeted data and apply a detector on the fly. This is more practical and tractable. When a robotic vehicle needs to build a roadmap for a route, it collects orthoimages along the route and use them to learn an object detector. Although these local orthoimages might not be helpful to learn a generic object detector, these images might be useful to learn a detector that works efficiently on the target region because these images share common image characteristics with the target image regions. However, this approach introduces another problem that requires a human operator to continuously assign labels to newly collected images.

We have developed a self-supervised learning approach that automatically collects a small set of training examples from the orthoimagery about the target region and learns the object model by using these self-labeled examples. Figure 5.8 shows parking spot image examples with varying appearances and their projections on a two-dimensional appearance space. Figure 5.8(a) shows some of the canonical parking spot images that are represented as (blue) rectangles in the figure 5.8(c). To project m-dimensional parking spot image into two-dimension, we first compute the eigenvectors of an affinity matrix about parking spots and then use the k most significant eigenvectors to represent the high-dimensional parking spot images in a k-dimensional space [Weiss, 1999]. The affinity matrix, $\mathbf{W}$, is computed by

$$\mathbf{W}_{i,j} = \exp\left\{-\frac{d(\mathbf{g}_i - \mathbf{g}_j)}{2\sigma^2}\right\}$$

where $d(\mathbf{g}_i - \mathbf{g}_j)$ is a function that measures Euclidean distance between two parking spot images, $\mathbf{g}_i$ and $\mathbf{g}_j$; and σ is the width of a kernel that controls the range of neighbors.

Suppose that a Cartesian coordinates in the figure 5.8(c) represents the appearance space, which our parking spot detector needs to estimate for reliable parking spot detection. Because of our imperfect self-labeler, the only available training data are parking spot images with canonical appearances shown in 5.8(a) and parking spot images with unusual appearances shown in 5.8(b) are initially not obtained. Any learning algorithm might produce the solution for parking spot classification around the centroid of the self-labeled examples in the figure 5.8(c) if it is learned by using only canonical parking spots. This solution might be the optimal in that it minimizes the error of canonical parking spot classification. However, this solution, which is

biased to the centroid of canonical parking spots, might not be the optimal when the whole appearance space is considered, resulting in a high error rate on unusual parking spot classification [Seo et al., 2009a]. To remedy this problem, it is necessary to uniformly sample parking spots from the appearance space. This uniform sampling can be done by either manually or automatically. Manual sampling requires human involvement whereas automatic sampling needs an improvement of our self-labeler's capability.

In this section, we describe our approach that manually samples some of the unusual parking spots and use them to improve the performance of our parking spot detector. Since manual labeling is expensive, we use an uncertainty sampling to minimize the use of the manually labeled data. Additionally, we convert our discrete classifiers into probabilistic ones so that classifiers can represent their confidences on classification decisions.

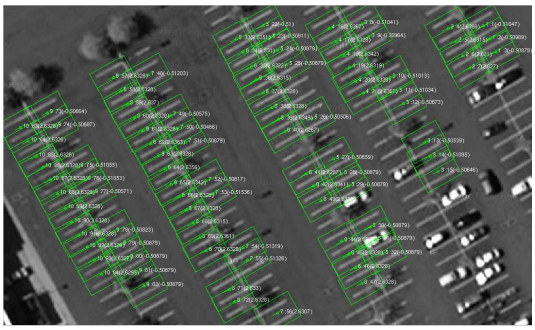
5.4.1 Uncertainty Sampling for Minimizing the Use of Manually Labeled Data

To reliably detect parking spots with varying appearance, it would be best to prepare a set of all possible parking spot images in advance and then learn an appearance model. Unfortunately, it is impractical to collect training data with all of the possible appearances a priori. In our case, because we are concerned with an analysis of orthoimagery along a specific route, it might be unnecessary to learn a generic appearance model. Rather, it may be adequate to collect local image data around the target region to learn a local appearance model. Although it is a reasonable approach for our task, it is undesirable to ask a human operator to assign labels to image data collected from the target region.

To minimize human intervention in automatic road map building, we developed a self-labeling method that analyzes extracted straight lines and collects small but high-quality training parking spots. An analysis of the spatial relationships between the collected parking spots results in localizing undiscovered parking spots. Several machine learning methods are then trained with the self-labeled examples to filter out erroneous parking spot hypotheses. The results are promising in that there are negligible performance differences relative to a detector trained on human-labeled examples [Seo et al., 2009a, Seo and Urmson, 2009b]. Our self-supervised approach fits well with autonomous vehicle applications where it is necessary to generate a road map for drivers on demand. However our self-labeling method is not perfect. It works based on a combination of computationally inexpensive low-level image procedures. We observed that the limited types and amount of self-collected parking spot data results in parking spot detectors producing around 17% false negative rate, meaning that detectors missed 17% of parking spots with abnormal appearances on average.

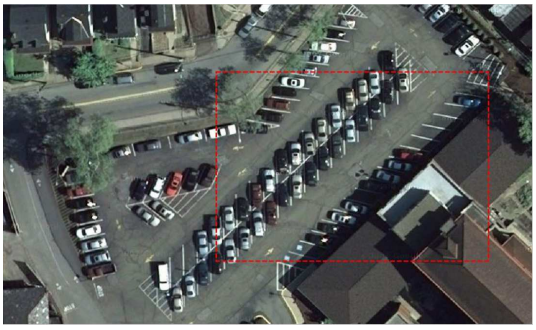
For example, the parking spot detector trained on self-labeled images (shown in the Figure 5.9(a) and images shown in the first row in the figure 5.10) fails to detect fishbone-shape parking spots under shade in the Figure 5.9(b) (images at the third row in the Figure 5.10) and in faded lane-markings in the Figure 5.9(c) (images at the last row in the Figure 5.10).

Figure 5.10 shows examples of parking spots with usual and with unusual appearances. Regardless of observation counts in aerial imagery, we consider the appearance of a parking spot as *unusual* if



(a) A parking lot image used for self-labeling. Each of the (green) rectangles is a self-labeled training parking spot example.

(b) A parking lot image where road-markings are obsolete.



(c) A parking lot image contains parking spots with unusual appearances: parking spots under shade and parking spots with fish-bone shapes.

Figure 5.9: Three examples of the aerial parking lot images used in our study are shown. The image at the left is one of the images that is used for self-labeling and the remaining two images contain parking spots with abnormal appearances.

- Its geometric shape is not rectangular (e.g., fishbones and trapezoidal) or its dimensions are larger than nominal (e.g., spots for the handicap and special-purpose vehicles such as trailer). [11] (See the second row in the Figure 5.10)
- Its intensity is distorted by aerial image acquisition process or under different illumination conditions (e.g., spots under shade or color aberration) (See the third row in the Figure 5.10)
- Its road-markings (or lane-markings) are faded. (See the last row in the Figure 5.10)

To improve classification performance sensitive to appearance variation, we consider two different approaches: classification decisions with confidence and sampling to optimally use

[11]The magnitude of a parking spot is regarded nominal if it afford a normal size vehicle that its dimensions are roughly assumed to be 3×5 meters. When a nominal parking spot is projected on a publicly available orthoimage, it occupies a subimage in 10×17 pixels. The finest resolution of publicly available orthoimage is 1 feet per pixel.

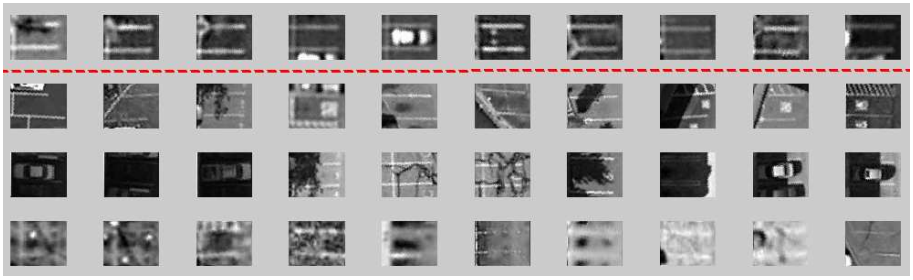


Figure 5.10: Parking spot image patches in different appearances are shown. Parking spots shown above the (red) dashed line are canonical parking spots and all the remaining ones are parking spots in unusual appearances. Particularly, unusual parking spots on the second row are ones in unusual shapes such as in fishbone, for the handicapped, in trapezoidal. Ones at the third row are in varying illumination conditions. Ones at the last row are with obsolete lane-markings.

human-labeled data. Instead of producing binary decisions, a confidence classifier produce produces a probability of a testing image being a parking spot. Sampling helps reduce human-labeled data usage by selecting a small set of the most informative samples for re-training.

When dealing with unusual parking spots, a purely binary classification is undesirable because it does not fit well into common detection framework where there is always insufficient data to estimate the true model. For example, when classifying a fishbone-shaped parking spot with a class label, it is more reasonable for a classifier to assign a low-confidence than rejecting as non-parking spot. The confidence represents a degree of discriminative ability based on a detector's current understanding about the appearance of parking spots. In other words, a low-confidence assigned to a fish-bone shape parking spot is interpreted as lack of experience with such type but the image has common features such as geometry (or maybe color histogram). To implement parking spot classification with confidence, we consider two methods: Ada.Boost [Freund and Shapire, 1997] and probabilistic support vector machines (SVMs) [Platt, 2000].

Ada.Boost Our implementation of Ada.Boost is motivated by Viola and Jones' approach for face detection [Viola and Jones, 2004] where decision stumps of individual features are linearly combined with their learned confidences to predict the class membership of a test image. In our case, a weak learner is a decision stump of one of the six parts in our feature representation. A decision stump is a single-layer decision tree that computes the similarity between a parking spot and the centroid of each of two classes (parking-spot and non-parking-spot) and assigns the label of the closest centroid to the spot. For the similarity calculation, we use the histogram intersection [Swain and Ballard, 1991]. Given two histograms, $\mathbf{h}_i$ and $\mathbf{h}_j$, the histogram intersection is computed by

$$HI(\mathbf{h}_i, \mathbf{h}_j) = \frac{\sum_{k=1}^{K} \min(\mathbf{h}_{i,k}, \mathbf{h}_{j,k})}{\sum_{k=1}^{K} \mathbf{h}_{j,k}}$$

where $\mathbf{h}_i$ is one of the six part in our feature representation (e.g., color histogram) and $\mathbf{h}_j$ is the centroid of the j class.

At the boosting iteration t, decision stumps, which are restricted to use only one part of our feature vector, are trained. The decision stump with the lowest error is selected as the tth

classifier, h_t. Its confidence α_t is computed based on its error, ϵ_t.

$$\begin{aligned} \alpha_t &= \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \\ \epsilon_t &= \frac{1}{n}\sum_{i=1}^{n} \mathrm{I}(y_i \neq h_t(\mathbf{x}_i)) \end{aligned}$$

where n is the number of test examples and $\mathrm{I}(\cdot)$ is an indicator function that returns 1 if the condition is satisfied. When classifying a new image, the Ada.Boost linearly combines the T classifiers with respect to their confidences

$$H(\mathbf{x}) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

Probabilistic SVM The idea of converting a SVM's discrete outputs into probabilistic ones is to fit a logistic sigmoid to the outputs of a previously trained SVM [Platt, 2000].

$$p(y_i = 1|\mathbf{x}_i) = \sigma(\mathbf{A}h(\mathbf{x}_i) + \mathbf{B})$$

where $\mathbf{x}_i$ is the ith image patch, $h(\mathbf{x}_i)$ is the SVM's output on $\mathbf{x}_i$, A and B are parameters of fitting a logistic function.

Uncertainty Sampling Manual labeling can be very informative but expensive and thus the amount needs to be minimized. Uncertainty sampling is a pool-based active learning framework that helps a learner choose a set of informative samples for manual labeling, avoiding a request of complete labeling of data in a pool [Lewis and Catlett, 1994]. In our case we have a pool of human collected parking spot images of which appearances are not similar to those of self-labeled images. Uncertainty sampling is utilized to minimize the usage of these human-labeled data. Since confidences associated with classification decisions reveal the classifier's current understanding of parking spot appearances, incorrect classified examples with low-confidence are likely near at the decision boundary and very informative for relocating the current decision boundary. In other words, because the classifier is uncertain about the class membership of low-confident examples, performance can be improved if the classifier is retrained with those examples.

5.4.2 Experimental Results

Several experiments were performed to understand how the performance of our parking spot classifiers changes as human-labeled examples are incrementally added to the training data.

There are 1,429 images used in our experiments. Our self-labeler analyzes 8 different parking lot orthoimages and automatically collects 532 parking spot images that 42 contain vehicles and 32 are not actual parking spots. The same number of negative images are also collected by randomly generating image patches around actual parking spot patches. We also manually collect 365 unusual parking spot images. Figure 5.10 shows example parking spot images with normal (or canonical) and unusual appearances. Although parking spot images used in this thesis

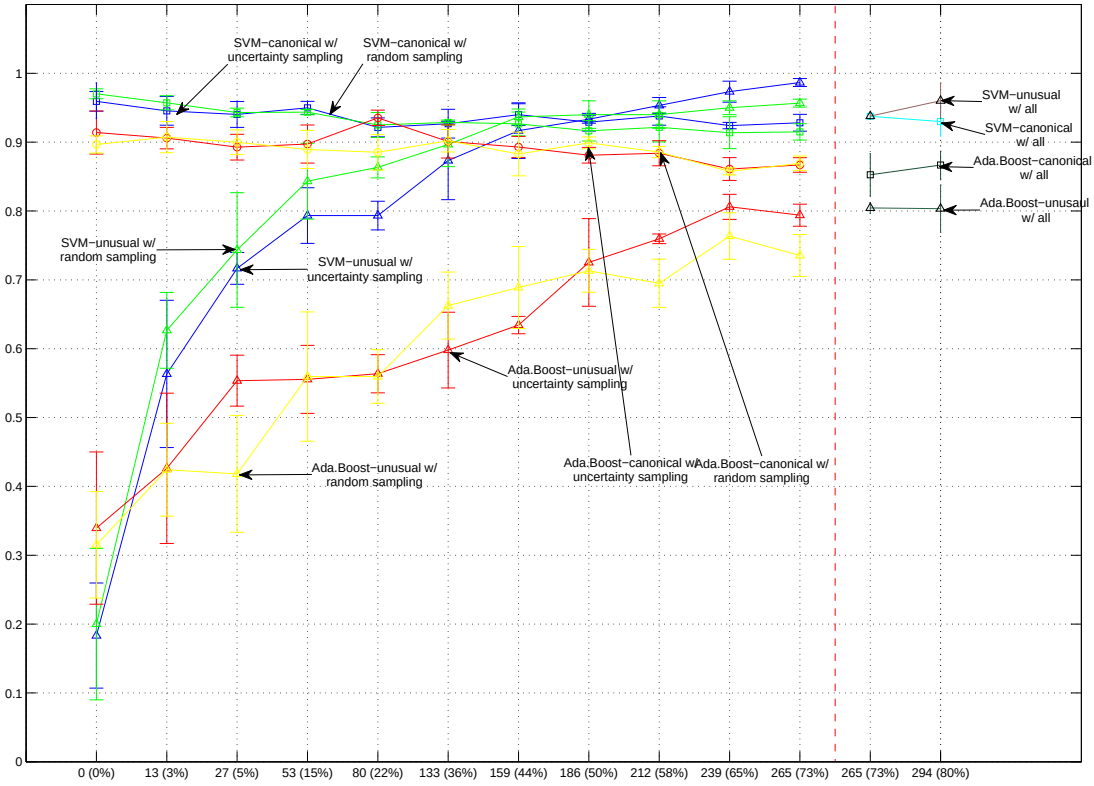


Figure 5.11: Experimental results of incremental retraining.

are scale- and rotation-free, their appearances vary greatly in intensity, geometric shape, color, texture, and quality of road-markings.

We test the performance of two machine learning algorithms with different sampling methods in an incremental re-training setting. They are initially trained with a set of self-labeled (canonical) parking spot images and their classification accuracies are measured with respect to two different types of parking spots. During several retraining processes after the initial training, a small amount of human-labeled parking spot data is added to the training data. Since the amount of the training data is fixed, the increase of unusual data in the training data decreases the amount of normal data. The classifiers are then re-trained with a composite of normal and unusual parking spot images.

This incremental re-training scenario is not uncommon in automatic building of road maps for autonomous driving in that a classifier is initially trained with a small amount of the training data (i.e., for our case, a small number of the self-labeled examples) and is retrained again when the performance drops below the desired level due to examples with different appearances.

Figure 5.11 shows experimental results. The x-axis represents the percentage of unusual

parking spot images as a part of the training data and the y-axis is classification accuracy that is defined as a ratio of the number of correctly classified test examples to the total number of test examples. There are two numbers at each x: numbers of unusual parking spots in the training data and the percentage of unusual parking spots used as a part of the training data. For example, the values at the end of the x-axis (265 (73%)) represents that 73% of the unusual data (i.e., 265 images) are used for the training data. Since we fix the total amount of training data at 851 images (80% of the self-labeled examples), the number of canonical images is 586 in this iteration. The vertical (red) dashed line in the figure divides graphs into two parts: results of incremental learning and results of batch learning. In a batch learning setting, all of the available parking spot images are split into two different sets: training and test and a general learning process is conducted.

There are two different classification methods to compare. Each of the classification methods is combined with two different sampling methods and individual combinations are tested with respect to two different parking spot image types, such as canonical and unusual, resulting in eight different settings to compare. We repeated each experiment five times to account for randomness in the data-split and sampling process.

Although the initial performance of Ada.Boost is slightly better than that of SVM, their performances are not acceptable. As the amount of unusual data is increased, their performances increases drastically whereas the performance on canonical spot remains relatively stable. In particular, the probabilistic SVM produces a consistent accuracy of over 90%. It is interesting to observe that the changes in training data composite do not affect performance on canonical parking spot classification. When 28% of the training data (239 out of 851) are unusual parking spots, SVM with uncertainty sampling outperforms all of the other approaches. Random sampling did well on improving the performance up to the setting where 50% of unusual parking spots are sampled.

We conducted another experiment that trained two classifiers with all of the available parking spot images and compared their performances with those of incremental learning settings. The performance of these batch learning approaches are depicted on the right side of the vertical dashed line at the end of the x-axis. Table 5.3 shows experimental results around the vertical dashed line in figure 5.11 and details a comparison of accuracies between incremental learning and batch learning. The results shows clearly that uncertainty sampling helped improve the performance of the probabilistic SVM with less data. For example, a combination of uncertainty sampling and a probabilistic SVM produced 97.33% accuracy by using 65% of the manually labeled data whereas the probabilistic SVM trained using 80% of the manually labeled data produced 96.02% accuracy. Particularly when the probabilistic SVM with uncertainty sampling utilized 58% of the manually labeled data (212), it used 82 fewer manually labeled examples to achieve 6% (95.33% vs. 96.02%) less accurate result than that of the batch learner trained by using 80% (294) of the manually labeled data.

In this section, we present a new method to handle intra-class variation that is caused by varying appearances of parking spots. Because our self-labeler is only able to automatically collect easy-to-detect examples, a parking spot detector trained by using self-labeled examples is incapable of correctly classifying parking spots with unusual appearances. To improve performance, we manually collected unusual parking spots and use them as a part of training data. We use uncertainty sampling to minimize the use of the manually labeled data because human labeling is

expensive. Experimental results show that uncertainty sampling helps improve the performance of a probabilistic SVM with less data than that of a batch learning approach.

		186 (50%)	212 (58%)	239 (65%)	265 (73%)	265 (73%)	294 (80%)
Uncertainty	Canonical	.9290 (±.0128)	.9380 (±.0133)	.9241 (±.0046)	.9282 (±.0123)	.9376 (±.0046)	.9299 (±.0212)
	Unusual	.9333 (±.0058)	.9533 (±.0115)	.9733 (±.0153)	**.9867** (±.0058)	.9379 (±.0047)	.9602 (±.0260)
Random	Canonical	.9164 (±.0147)	.9216 (±.0212)	.9137 (±.0231)	.9149 (±.0119)	–	–
	Unusual	.9400 (±.0200)	.9400 (±.0200)	.9500 (±.0100)	.9567 (±.0058)	–	–
Uncertainty	Canonical	.8810 (±.0114)	.8838 (±.0182)	.8609 (±.0166)	.8668 (±.0105)	.8526 (±.0307)	.8667 (±.0203)
	Unusual	.7253 (±.0637)	.7596 (±.0070)	.8061 (±.0182)	.7940 (±.0160)	.8044 (±.0078)	.8034 (±.0345)
Random	Canonical	.8989 (±.0087)	.8856 (±.0082)	.8575 (±.0051)	.8692 (±.0106)	–	–
	Unusual	.7131 (±.0311)	.6950 (±.0350)	.7636 (±.0338)	.7353 (±.0305)	–	–

Table 5.3: Comparison of accuracy between incremental learnings and batch learnings.

5.5 Summary

This chapter details orthoimage analysis methods that automatically build parking-lot maps. In a hierarchical scheme, our algorithms analyze structures and build skeletons representing drivable regions of a parking lot from orthoimage. For parking spot detection, a low-level analysis layer extracts a set of easily-detected canonical parking spots and estimates parking blocks using line detection and clustering techniques. A high-level analysis then extends those spots using geometrical characteristics of typical parking lot structures to interpolate and extrapolate new hypotheses and uses self-supervised machine learning techniques to filter out false positives in the proposed hypotheses. Experiments show that training the classifiers using a self-supervised set of canonical parking spots extracted by low-level analysis successfully adapts the filter stage to the particular characteristics of the image under analysis. Self-supervised examples are also effectively utilized to train a road-marking detector and a parking lot boundary segment. A composite of this information is then used to estimate the structure of the parking lot.

A lane-graph is used to concisely represent the obtained information. Our lane-graph generation algorithm first produces a distance transform map from the boundaries of drivable regions and the locations of parking spots, to reveal the skeletal of drivable regions; second, it identifies the locations of peaks in the distance transform map; and finally it connects some of these peaks to produce a lane-graph consistent with drivable regions.

Since our approach to building a parking lot map begins with parking spot detection, it is important for the parking spot detector to robustly identify parking spots in a given image. Since our detector is trained using self-labeled canonical parking spot images, it often fails to detect parking spots of unusual appearance. To cope with such appearance variations in parking spot images, we devised an uncertainty sampling method that incrementally improves our detector's capability of handling parking spot images with unusual appearances by using a small number of manually labeled parking spot images. Empirically we found that an incremental update of the parking spot model through uncertainty sampling increased the performance of our parking spot detector.

Chapter 6

Conclusions

In this thesis we demonstrated that we could provide lane-level detailed maps of highways and parking lots by analyzing publicly available orthoimages. Instead of relying on human labeled data, our algorithms acquire, through bootstrapping, task-relevant image features by extensively exploiting prior information and readily-collectible low-level image features. For generating highway maps, our algorithm, without human intervention, is capable of identifying image road sub-regions; of detecting overpass boundaries; and of estimating legitimate driving directions, by extensively analyzing what is already available such as lines and screenshot images of road-vectors. These mid-level image features are used to generate hypotheses about true road-lanes and guide the process of linking those hypotheses to extract the boundaries of highway road-lanes. For generating parking lot maps, our algorithm is able to acquire a set of parking spot image examples in canonical appearances by analyzing lines derived from image intensity gradient analysis. These self-obtained parking spot image examples are used to train a model of parking spot appearance, which is in turn utilized to detect all of the visible parking spots appearing in the input images.

While we demonstrated the capability of our perspective image analysis algorithm, we also utilized the scheme of extensively using prior information. Prior information about traffic sign location was obtained from ground truth data and was used to improve the performance of our sign detector – removing any false positive and negative sign detections.

Through testing our algorithms with real-world data, highway and parking lot orthoimages and highway workzone videos, we showed a promising result. Our orthoimage analysis algorithms produce lane-level detailed maps of highways and parking lots. Our perspective-image analysis algorithm recognizes the bounds of workzones and most of the temporary changes.

6.1 Contributions

6.1.1 Orthoimage Analysis for Building Lane-level Roadmaps

Our first main contribution is to demonstrate the usefulness of bootstrapping to achieve the goal of a given computer vision task. We demonstrated that we can collect task-relevant and task-specific information only from a given image by extensively utilizing prior information and inten-

sively analyzing readily collectible low-level image features. For example, to generate highway maps, our algorithms obtain task-specific mid-level image cues, such as results of road-region segmentation, results of driving direction identification, and results of overpass detection, to guide the process of linking hypotheses about true road-lanes. This enables us to extract the boundaries of highway road-lanes. Similarly, to generate parking lot maps, our algorithms acquire task-specific information, e.g., a set of parking spot images, by analyzing extracted lines and their spatial dependencies. These self-labeled examples are used to train a model of parking spot appearances. This model is in turn used to detect all of the visible parking spots in a given image. Knowledge of parking spot locations makes it easier for our algorithm to extract the geometric structures of parking lots.

A System for Producing Lane-level Detail Highway Maps: We developed a computer vision system that can produce a lane-level detail highway map by analyzing a given orthoimage. For collecting task-specific image patterns, we developed algorithms that extensively analyze prior information and readily collectible low-level image features and produce three task-specific pieces of information: road-region segmentation, driving-direction estimation, and overpass detection. We developed a hypothesis-linking function that connects hypotheses about true road-lanes based on the geometric and photometric constraints.

A System for Producing Parking Lot Maps: We developed a computer vision system that can produce a map of a parking lot by analyzing a given orthoimage without human intervention. The resulting map contains locations of parking spots and the geometry of drivable regions. For detecting parking spots appearing on a given image, we developed an algorithm that automatically acquires easily collectible parking spot images in canonical appearance and uses these examples to learn a model of parking spot appearance. We developed a segmentation algorithm that is capable of distinguishing parking lot image regions from the background. We developed an algorithm that extracts a lane-graph of a parking lot by analyzing a polygonal shape about the spatial relations between detected parking spots and identified drivable regions.

Demonstration of Usefulness of Bootstrapping Image Processing: This thesis provides a good use case of bootstrapping image processing that exploits the salient characteristic of a given problem; it extensively analyzes a given image to obtain task-relevant and task-specific information while minimizing human intervention.

6.1.2 Perspective Imagery Analysis for Assessing Roadway State

Our second main contribution is to demonstrate that our system can recognize 1) the bounds of workzones and 2) temporary changes in driving conditions on highways by analyzing perspective images. In addition, to address potential sign recognition errors, we developed two algorithms. The first, in minimizing the impact of false negative sign detections, propagates sign classification confidences over time. The second, to reduce the number of false positive detections, utilizes temporarily redundant appearances of the same signs.

A System for Workzone Sign Detection and Classification: We developed a computer vision system that analyzes perspective videos to detect workzone signs based on pixel-wise color classification. The system then classifies the detected sign image regions into one of the predefined workzone sign classes. We demonstrated a promising result through testing this system with five real-world workzone videos.

Development of Sign Recognition Error Handling Methods: We developed algorithms that handle potential workzone sign recognition errors. To address false negative detections, we devised a confidence propagation algorithm that forwards sign classification confidences over time to sustain, for a while, our belief in driving on a highway workzone. To minimize the number of false positive detections, we developed an algorithm that makes use of temporal redundancy of sign occurrences and their corresponding classification confidences, to augment the current sign classification.

6.2 Future Work

In this thesis, we proposed a set of methods to produce maps of highways and parking lots and update potential temporary changes on the resulting highway maps. We hope that others will build on our ideas and other related work in the community to advance the study of the related fields. We now propose several ideas for future work that would extend our findings.

Better Understanding of Hierarchical Approaches for Executing Computer Vision Tasks We explicitly or implicitly employed a hierarchical bottom-up approach for analyzing images. Such a hierarchical approach is prevalent in the computer vision community. However, we have not yet investigated when and how such a system fails to accomplish a given task.

Annotating Global Coordinates to Resulting Maps The resulting maps of highways and parking lots are coordinate-free. To be useful, it is necessary to assign global coordinates, such as Universal Transverse Mercator, to pixel coordinates. One possible way is to utilize pairs of latitude and longitude from an Internet map service, such Google maps, to sparsely assign global coordinates and later interpolate these sparse coordinates while driving in the regions appearing on the map.

Parsing Images about In-City Roads to Complete a Route Map Our initial goal was to generate a route map between two locations, e.g., a route from a person's garage to a shopping mall. To completely generate a map of such a route, it is necessary to parse images about in-city roads, from ones house to a highway. We excluded this part from the thesis work because we believe analyzing in-city images involves too much work of handling objects, such as vehicles, pedestrians, occlusions by urban structures. The high frequency of these objects' occurrences may be irrelevant to the task of map generation but important to detecting them properly for other vision tasks.

Making Color-Based Sign Detection Robust Although we demonstrate the effectiveness of utilizing color in detecting signs in perspective images, a color-based sign detector may, in practice, occasionally fail (e.g., where a variation of the target color has not been observed during the training phase). For future work, it would be useful to investigate an approach that combines color information with shape information. Our color-based sign detector failed to detect some workzone signs when their images were under- and over-exposed. In order to handle such images, we would like to investigate a method that estimates the illumination response function of our vision sensor. In addition, our tests that evaluated the acquisition of highway workzone information may not be exhaustive. In this regard, in future work, we would like to collect more video data to include other events and to extensively evaluate our approach's workzone recognition capability.

Improving Performance of Part-Detectors In the course of this thesis work we developed a couple of detectors that identify parts of objects of interest. These could include such things as a lane-marking detector for identifying image cues of highway road-lanes, a parking spot detector for parking lot structure analysis, and a workzone sign detector for recognizing workzone bounds. The performance of each part-detector affects that of the final output. For example, a high false positive rate of lane-marking detector for highway map generation may be acceptable in terms of adding noise to the image cues we need to handle. However, it might cause a serious problem when our detector misses a majority of true lane-markings.

Incremental Learning for Handling Variations in Object Appearances In general, variations in object appearances are one of the most challenging factors in executing any computer vision tasks. In our cases, we have observed variations in color and texture of highway surface materials and lane-markings, parking spots' geometric and image appearances, and color of workzone signs. We have developed a learning method that utilizes a small number of manually labeled examples to incrementally update the appearance models of parking spots. It would be useful if a generic learning method could handle such variations as new data about unusual appearances arrive.

Appendix A

Examples of Hand-Labeled Data

In Chapter 3, we use two types of hand-labeled data: ground truth annotations of test highway orthoimages and highway orthoimages with lane-marking annotations.

The ground truth annotations of test highway orthoimages are created to evaluate the accuracies of the resulting lane-level detailed highway maps. Figure A.1 shows a ground truth annotation of an orthoimage. For each test orthoimage, we saved a screenshot of the geographic region appearing on the Google maps service. We then manually delineated, with a distinctive color, the boundaries of road-lanes appearing on each image. The width of the ground truth annotation is one pixel. The evaluation procesure is detailed in Section 3.3. The next section details how to access these ground truth annotations.

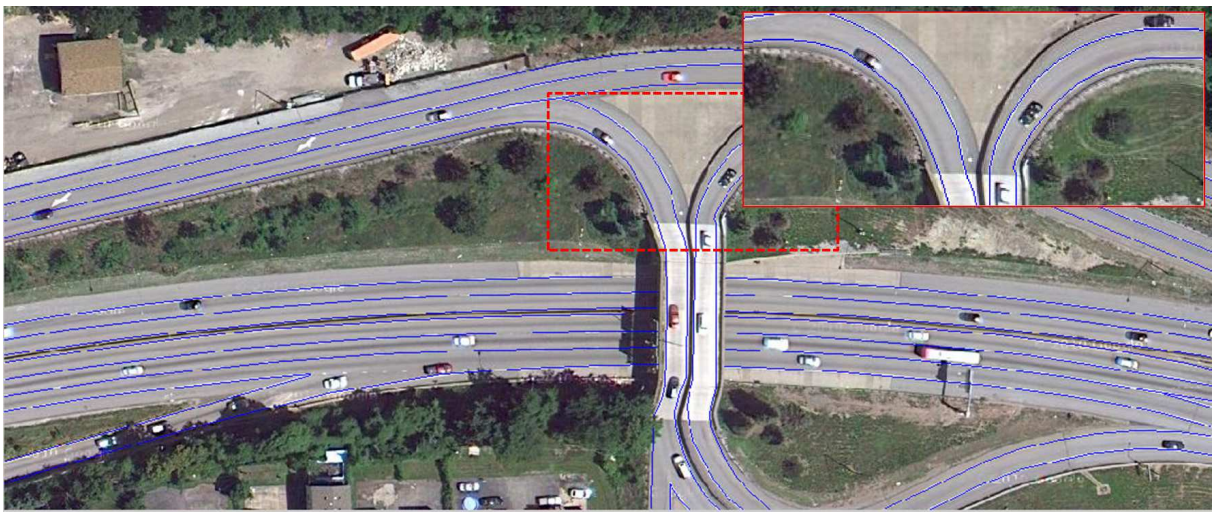


Figure A.1: A ground truth annotation of a highway orthoimage. The blue lines are manually drawn to indicate the boundaries of the road-lanes appearing on the image.

The other hand-labeled data are highway orthoimages with manual annotations of lane-markings. Figure A.2 shows five examples of orthoimages with lane-marking annotations. We have 22 such images and used them to train a lane-marking detector. The geographic regions appearing on these images are inter-state highways near Pittsburgh. For marking some pixel

locations, we drew blue lines for some of the true lane-marking image regions and red lines for non lane-marking regions. For training a lane-marking detector, we investigated each of the marked pixels to identify its image characteristics and to produce a feature vector. These features were used to train a road-marking detector. The procedure of training a lane-marking detector is detailed in Section 3.1.2.

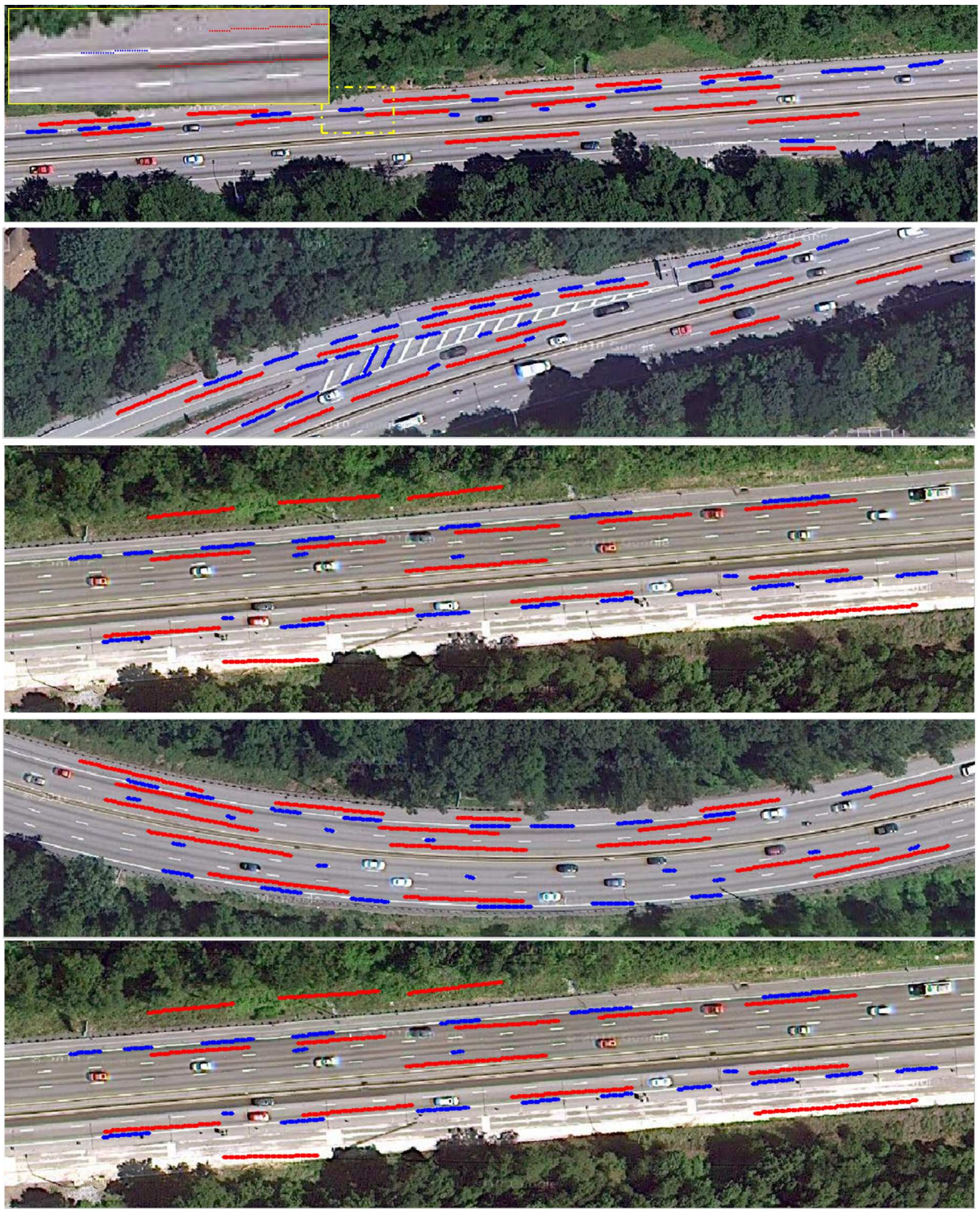


Figure A.2: Examples of highway orthoimages with lane-marking annotations. Some of the lane-markings are manually marked to indicate positive (blue) and negative (red) examples of lane-marking pixels.

Appendix B

Test Orthoimages for Highway Map Generation

We, as a reference, attach all 50 orthoimages used for testing our highway map generation algorithms and provide the information about how to access to the ground truth annotations.

Table B.1 and Table B.2 show lists of the GPS coordinates. The coordinates at each row define two corners of the rectangular viewpoint of a test image. Each of the ground truth annotation is available as a Portable Network Graphic (PNG) image from the following:

http://www.frc.ri.cmu.edu/∼ywseo/projects/highway-map-generation/groundtruth/route-376-?-20m-groundtruth.png

where the symbol ? at the end of the address should be replaced with the image number, 1-50, to browse a ground truth image. For example, the url,*http://www.frc.ri.cmu.edu/∼ywseo/projects/highway-map-generation/groundtruth/route-376-1-20m-groundtruth.png*, will let you browse the ground truth annotation of the first test image.

Image Number	Top Left	Bottom Right	Link to Ground truth
1	40.428891, -79.929531	40.427861, -79.926311	link[1]
2	40.435677,-79.968965	40.434654,-79.965719	link[2]
3	40.429607, -79.933474	40.428586,-79.930290	link[3]
4	40.429103,-79.938079	40.428023,-79.934909	link[4]
5	40.436802,-79.973425	40.435765,-79.970169	link[5]
6	40.436679,-79.977355	40.435632,-79.974107	link[6]
7	40.435085,-79.989229	40.434084,-79.986332	link[7]
8	40.434876,-79.992611	40.433694,-79.989213	link[8]
9	40.434899,-79.996892	40.433805,-79.993445	link[9]
10	40.435765,-79.999990	40.434672,-79.997136	link[10]
11	40.437228,-80.002645	40.436097,-79.999311	link[11]
12	40.439543,-80.008959	40.438525,-80.005775	link[12]
13	40.439986,-80.011507	40.438996,-80.008291	link[13]
14	40.438521,-80.015160	40.437555,-80.011982	link[14]
15	40.431534,-80.027043	40.430463,-80.024382	link[15]
16	40.430575,-80.029188	40.429550,-80.026031	link[16]
17	40.429107,-80.031176	40.428041,-80.028183	link[17]
18	40.427061,-80.032241	40.426020,-80.028899	link[18]
19	40.425491,-80.031970	40.424382,-80.028711	link[19]
20	40.423655,-80.031374	40.422600,-80.028185	link[20]
21	40.420954,-80.034207	40.419918,-80.030951	link[21]
22	40.420649,-80.037069	40.419643,-80.033751	link[22]
23	40.421117,-80.040548	40.420008,-80.037039	link[23]
24	40.421652,-80.045979	40.420641,-80.042822	link[24]
25	40.420529,-80.052226	40.419539,-80.049029	link[25]

Table B.1: This table lists two pairs of GPS coordinates of the first 25 test images numbered 1 to 25. These images are sampled on December, 2011.

Image Number	Top Left	Bottom Right	Link to Ground truth
26	40.419194, -80.054916	40.418175, -80.051738	link[26]
27	40.411246,-80.072232	40.410251,-80.069030	link[27]
28	40.412271, -80.077994	40.411258,-80.074751	link[28]
29	40.413139,-80.080499	40.412146,-80.077417	link[29]
30	40.413312,-80.086207	40.415265,-80.083117	link[30]
31	40.418834,-80.089889	40.417795,-80.086523	link[31]
32	40.421803,-80.096745	40.420770,-80.093446	link[32]
33	40.422840,-80.102000	40.421781,-80.098497	link[33]
34	40.423923,-80.108094	40.422873,-80.104827	link[34]
35	40.424758,-80.112661	40.423712,-80.109432	link[35]
36	40.425487,-80.121207	40.424413,-80.118037	link[36]
37	40.429977,-80.134417	40.428909,-80.131279	link[37]
38	40.430722,-80.137247	40.429685,-80.133964	link[38]
39	40.438521,-80.015160	40.437555,-80.011982	link[39]
40	40.445045,-80.165040	40.444089,-80.161856	link[40]
41	40.451644,-80.170388	40.450644,-80.167210	link[41]
42	40.449776,-80.169264	40.448792,-80.165917	link[42]
43	40.454828,-80.183230	40.453787,-80.179937	link[43]
44	40.460046,-80.191824	40.459056,-80.188517	link[44]
45	40.451903,-80.171362	40.450878,-80.168274	link[45]
46	40.453117,-80.177563	40.452101,-80.174392	link[46]
47	40.455470,-80.184979	40.454515,-80.181750	link[47]
48	40.466152,-80.197218	40.465178,-80.194007	link[48]
49	40.469366,-80.200233	40.468288,-80.196781	link[49]
50	40.465193,-80.196456	40.464056,-80.193109	link[50]

Table B.2: This table enumerates two pairs of GPS coordinates of the remaining 25 test images numbered 26 to 50.

(a) Test image 1.

(b) Test image 2.

(c) Test image 3.

(d) Test image 4.

Figure B.1: Test highway orthoimages 1-4.

(a) Test image 5.

(b) Test image 6.

(c) Test image 7.

(d) Test image 8.

Figure B.2: Test highway orthoimages 5-8.

(a) Test image 9.

(b) Test image 10.

(c) Test image 11.

(d) Test image 12.

Figure B.3: Test highway orthoimages 9-12.

(a) Test image 13.

(b) Test image 14.

(c) Test image 15.

(d) Test image 16.

Figure B.4: Test highway orthoimages 13-16.

(a) Test image 17.

(b) Test image 18.

(c) Test image 19.

(d) Test image 20.

Figure B.5: Test highway orthoimages 17-20.

(a) Test image 21.

(b) Test image 22.

(c) Test image 23.

(d) Test image 24.

Figure B.6: Test highway orthoimages 21-24.

(a) Test image 25.

(b) Test image 26.

(c) Test image 27.

(d) Test image 28.

Figure B.7: Test highway orthoimages 25-28.

(a) Test image 29.

(b) Test image 30.

(c) Test image 31.

(d) Test image 32.

Figure B.8: Test highway orthoimages 29-32.

(a) Test image 33.

(b) Test image 34.

(c) Test image 35.

(d) Test image 36.

Figure B.9: Test highway orthoimages 33-36.

(a) Test image 37.

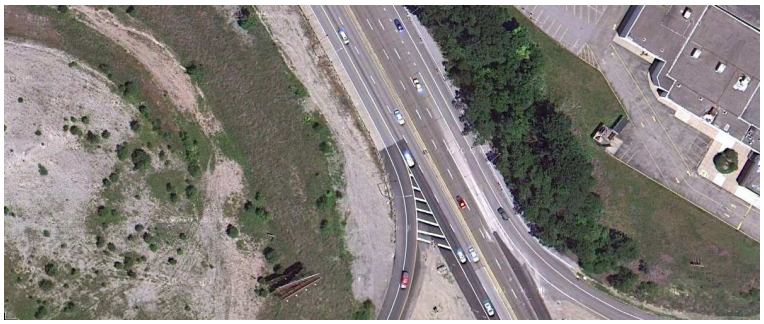
(b) Test image 38.

(c) Test image 39.

(d) Test image 40.

Figure B.10: Test highway orthoimages 37-40.

(a) Test image 41.



(b) Test image 42.

(c) Test image 43.

(d) Test image 44.

Figure B.11: Test highway orthoimages 41-44.

(a) Test image 45.

(b) Test image 46.

(c) Test image 47.

(d) Test image 48.

Figure B.12: Test highway orthoimages 45-48.

(a) Test image 49.

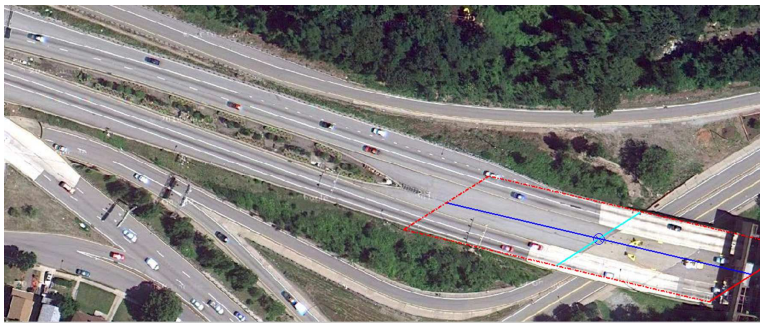
(b) Test image 50.

Figure B.13: Test highway orthoimages 49-50.

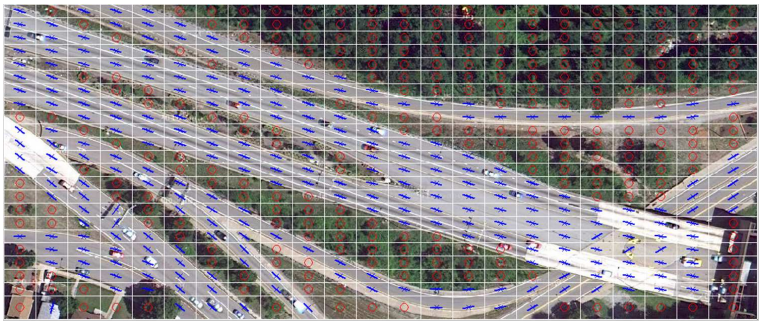
Appendix C

Some Additional Results of Highway Map Generation

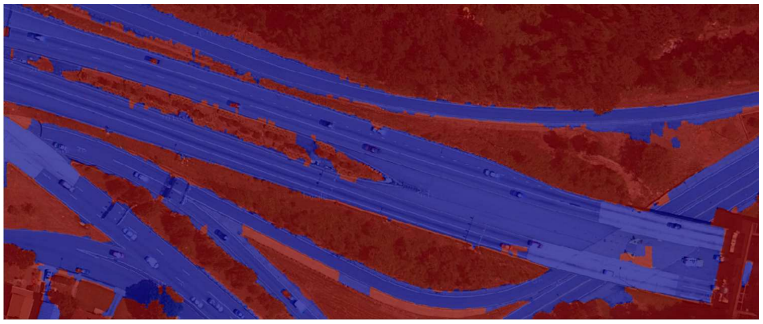
In the previous chapter, we showed all 50 test images used for evaluating the accuracies of the resulting highway maps. For each of the test images, we ran several algorithms to produce intermediate results and refined these results to produce a lane-level detailed highway map. This chapter shows some important results of our highway map generation. These results include three outputs from mid-level bootstrapping tasks such as overpass detection results, driving-direction estimation results, image road-region segmentation results, and the resulting lane-level highway map.



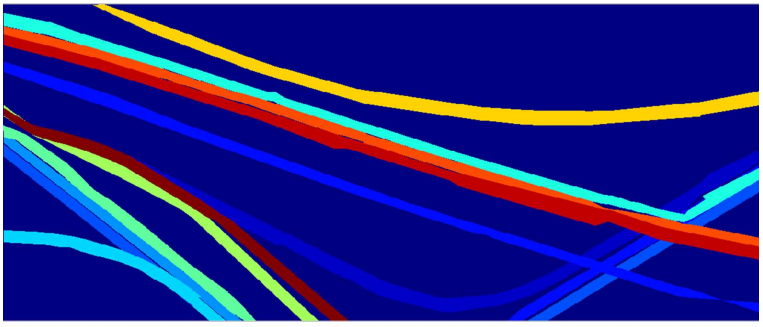
(a) Results of overpass detection.



(b) Results of driving direction estimation.

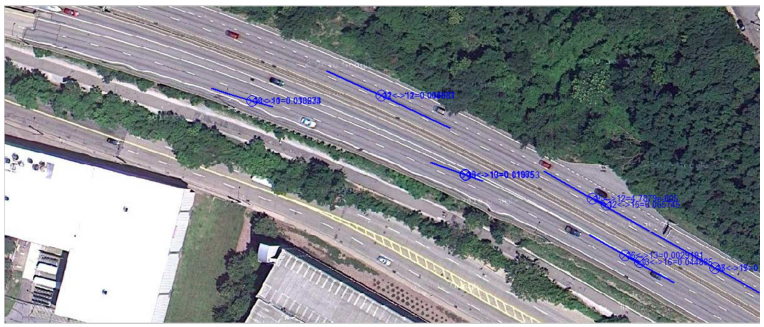


(c) Results of road-region segmentation.

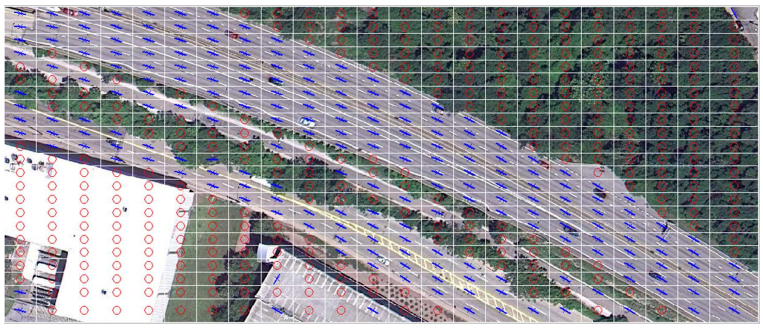


(d) Results of highway map generation.

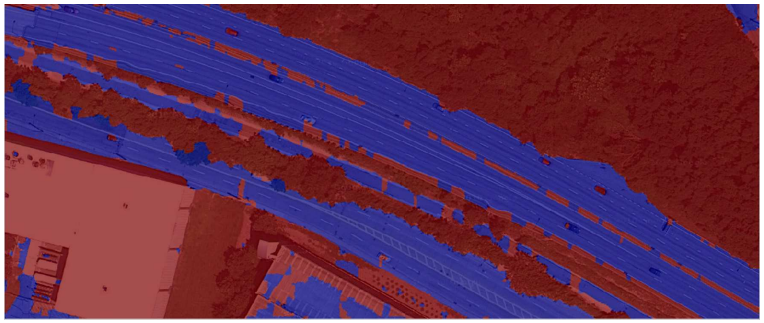
Figure C.1: Test highway orthoimage 1.



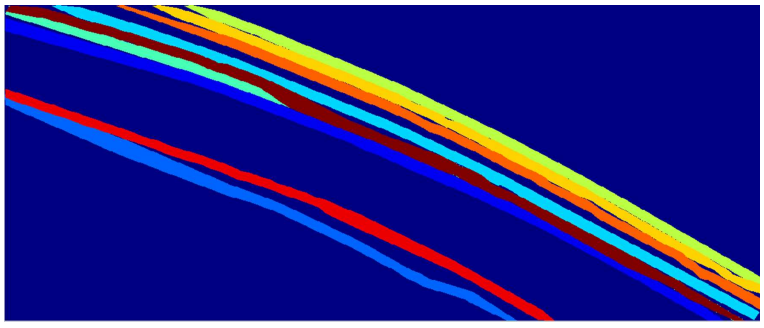
(a) Results of overpass detection.



(b) Results of driving direction estimation.

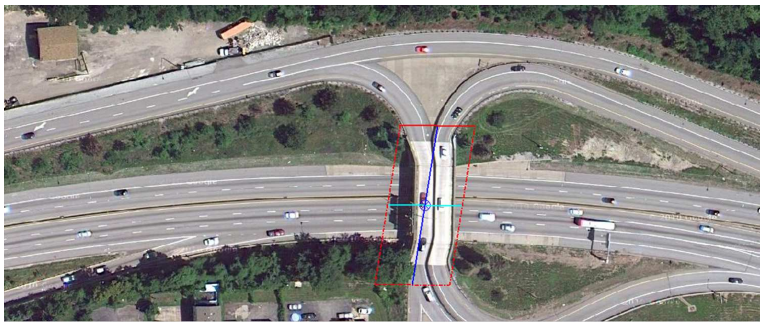


(c) Results of road-region segmentation.

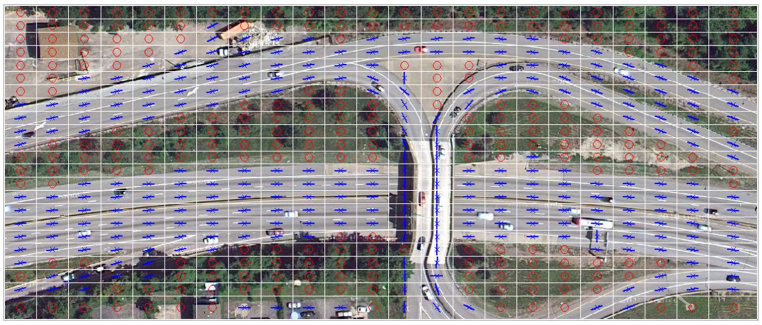


(d) Results of highway map generation.

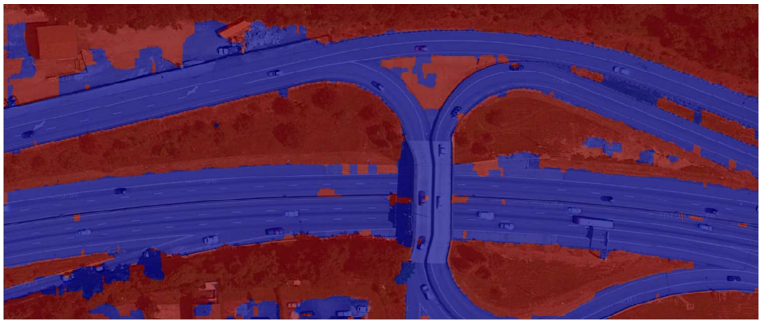
Figure C.2: Test highway orthoimage 2.



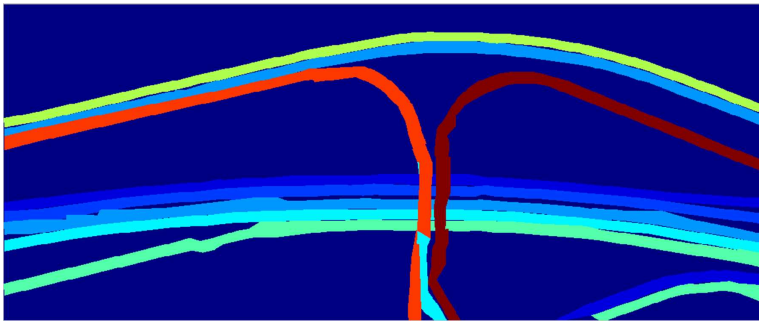
(a) Results of overpass detection.



(b) Results of driving direction estimation.



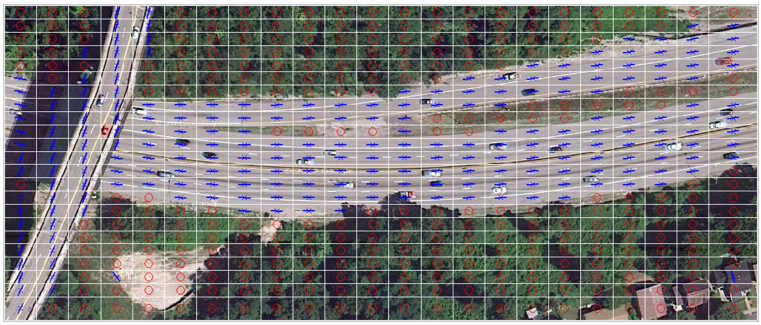
(c) Results of road-region segmentation.



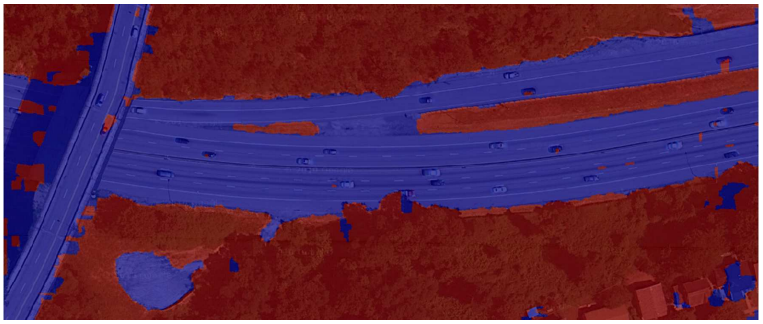
(d) Results of highway map generation.

Figure C.3: Test highway orthoimage 3.

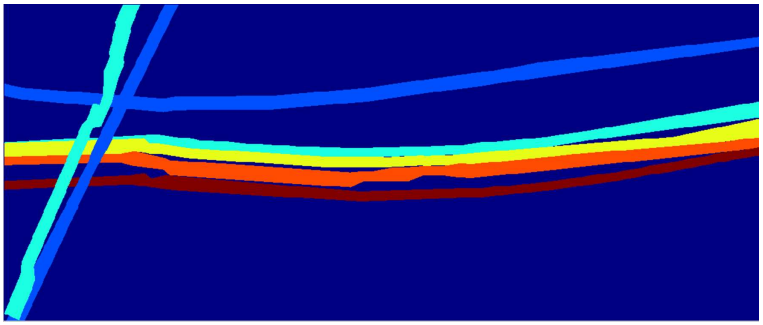
(a) Results of overpass detection.



(b) Results of driving direction estimation.

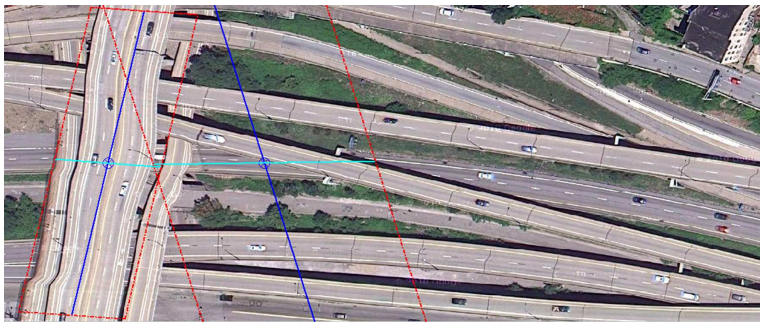


(c) Results of road-region segmentation.

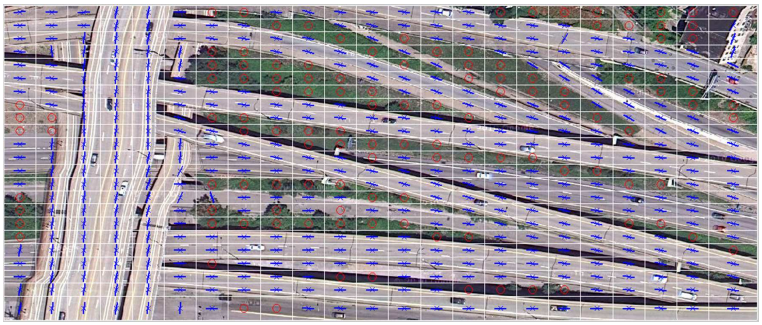


(d) Results of highway map generation.

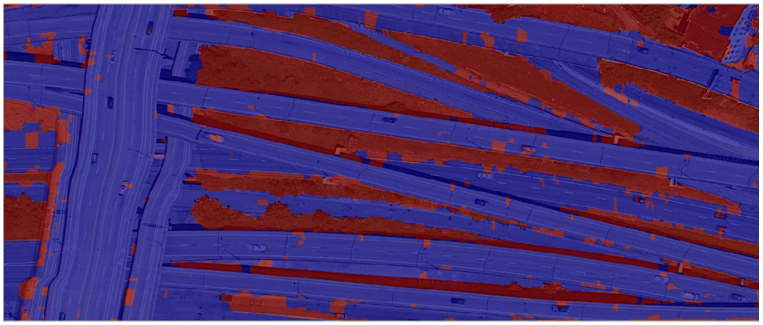
Figure C.4: Test highway orthoimage 4.



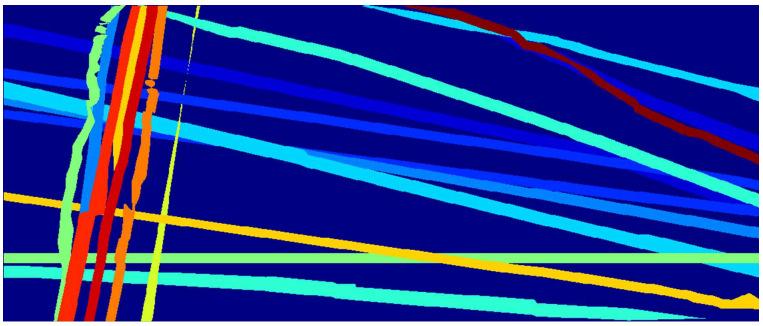
(a) Results of overpass detection.



(b) Results of driving direction estimation.



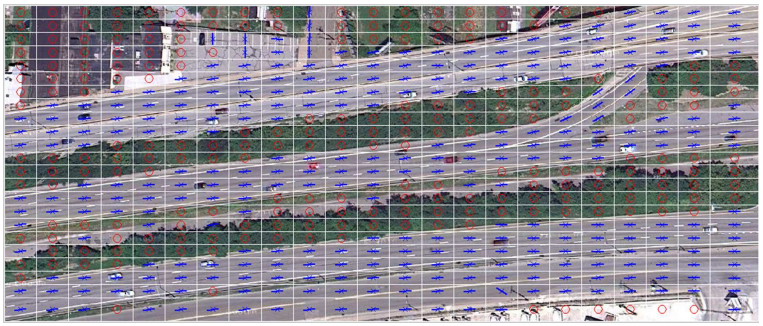
(c) Results of road-region segmentation.



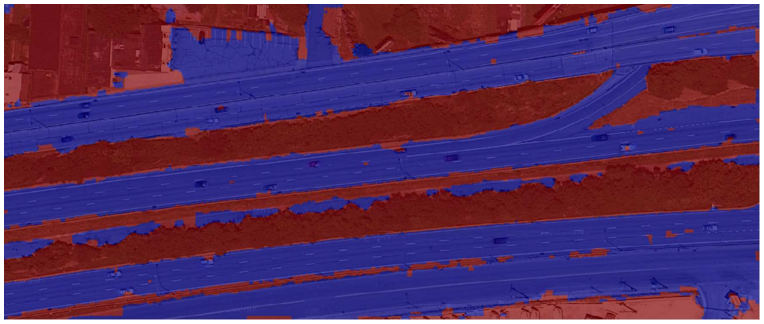
(d) Results of highway map generation.

Figure C.5: Test highway orthoimage 5.

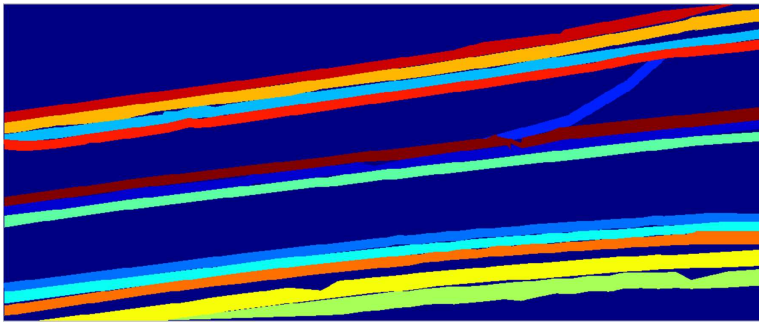
(a) Results of overpass detection.



(b) Results of driving direction estimation.



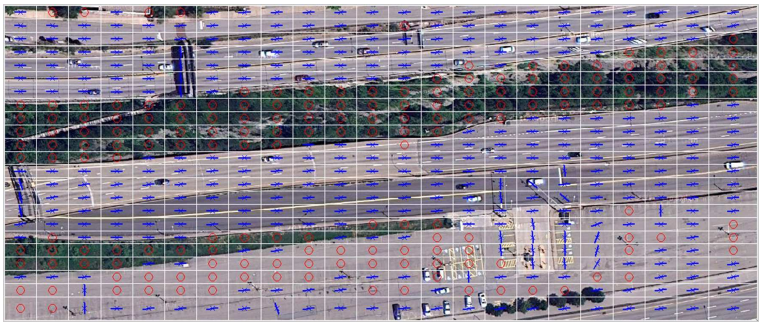
(c) Results of road-region segmentation.



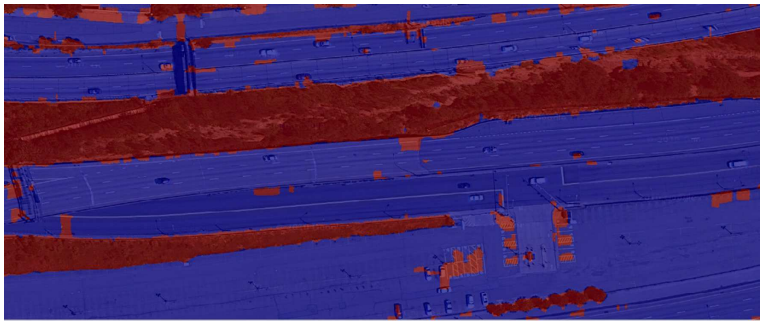
(d) Results of highway map generation.

Figure C.6: Test highway orthoimage 6.

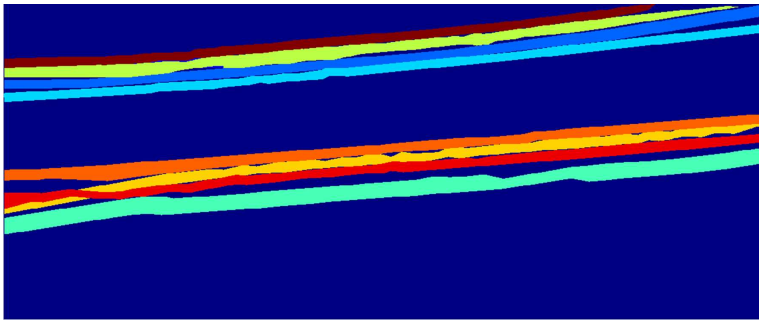
(a) Results of overpass detection.



(b) Results of driving direction estimation.

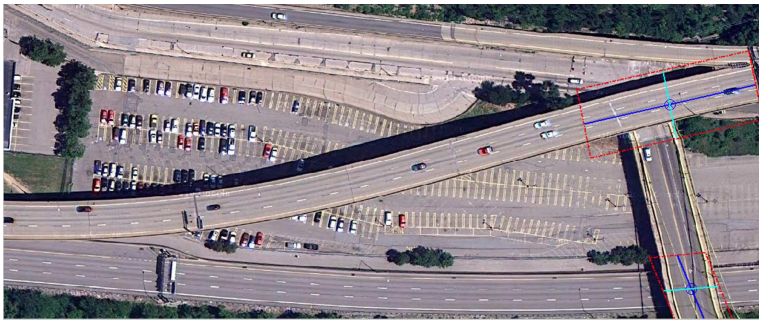


(c) Results of road-region segmentation.

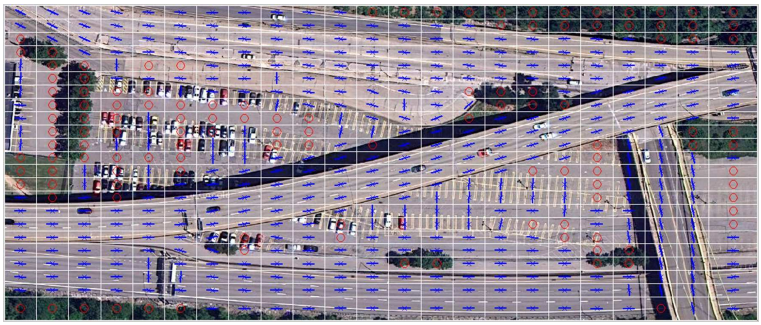


(d) Results of highway map generation.

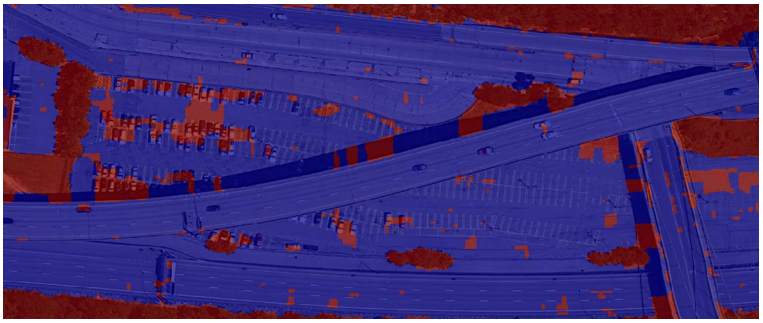
Figure C.7: Test highway orthoimage 7.



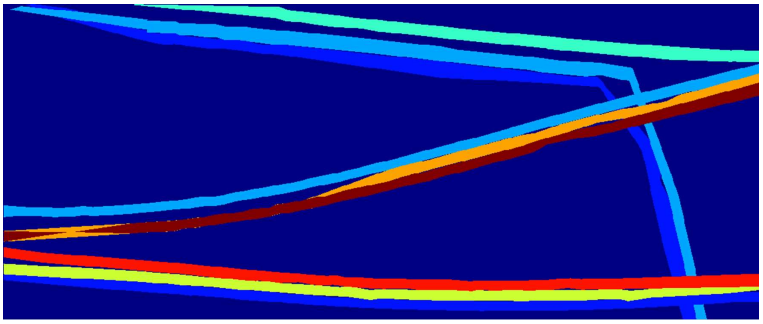
(a) Results of overpass detection.



(b) Results of driving direction estimation.

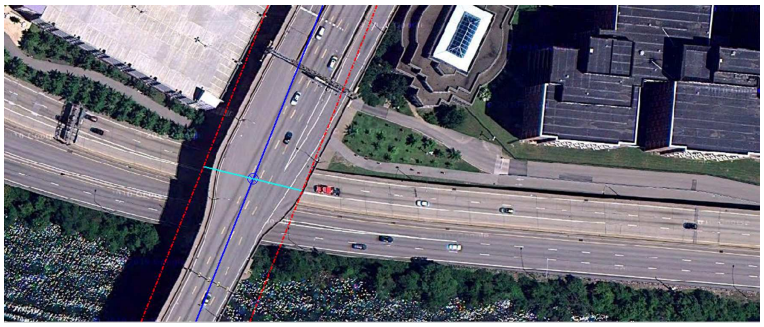


(c) Results of road-region segmentation.

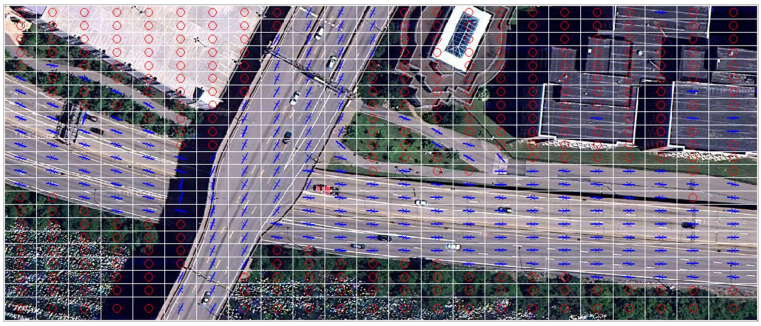


(d) Results of highway map generation.

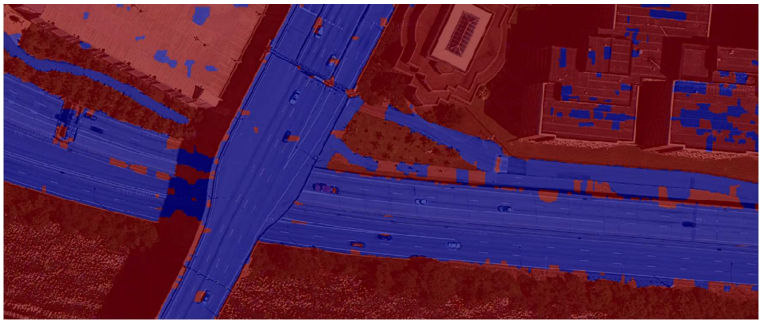
Figure C.8: Test highway orthoimage 8.



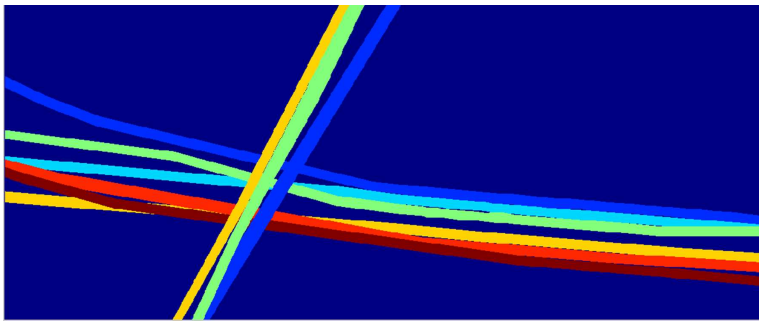
(a) Results of overpass detection.



(b) Results of driving direction estimation.

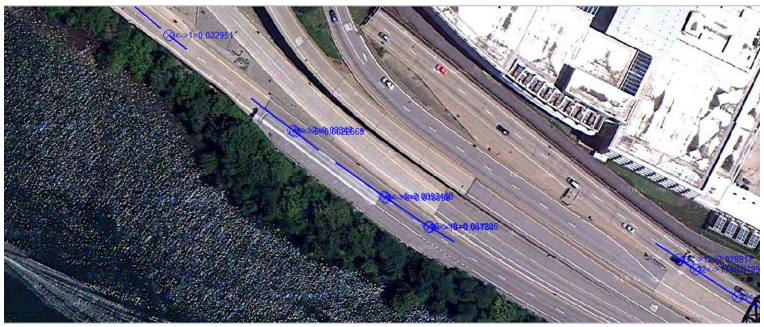


(c) Results of road-region segmentation.

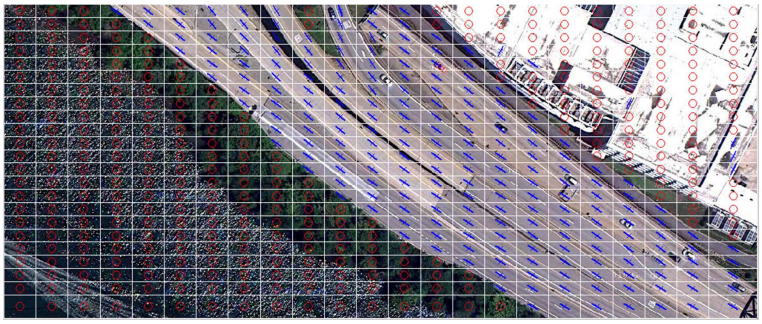


(d) Results of highway map generation.

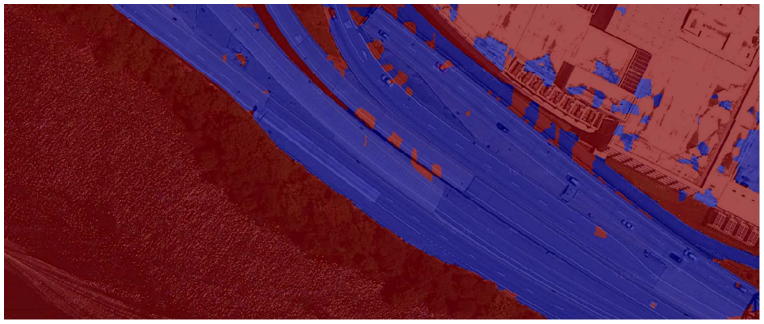
Figure C.9: Test highway orthoimage 9.



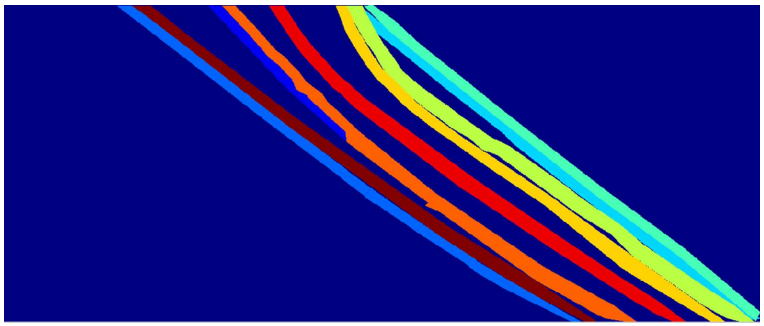
(a) Results of overpass detection.



(b) Results of driving direction estimation.

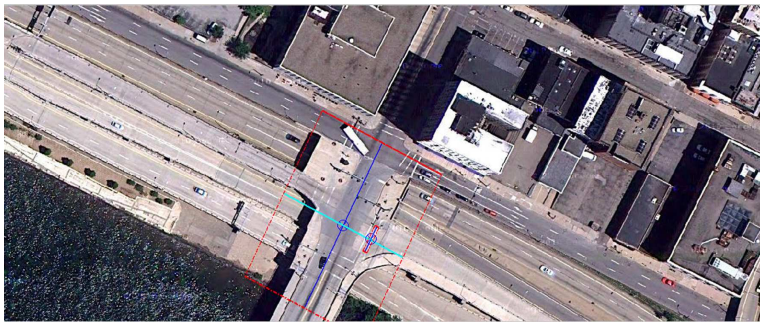


(c) Results of road-region segmentation.

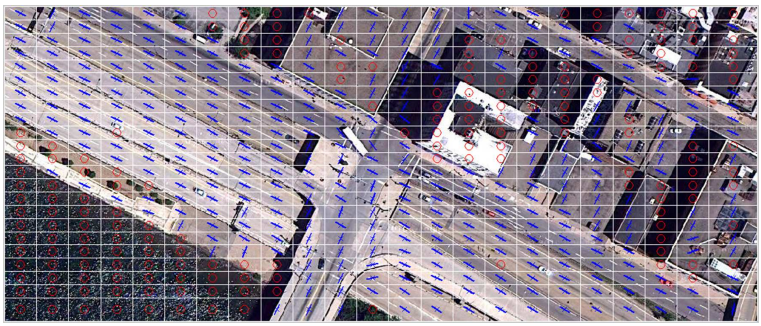


(d) Results of highway map generation.

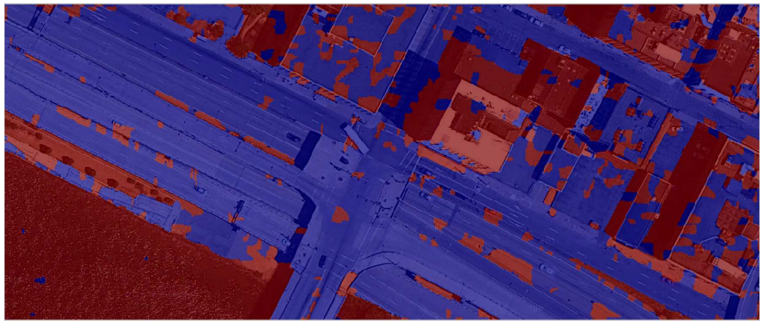
Figure C.10: Test highway orthoimage 10.



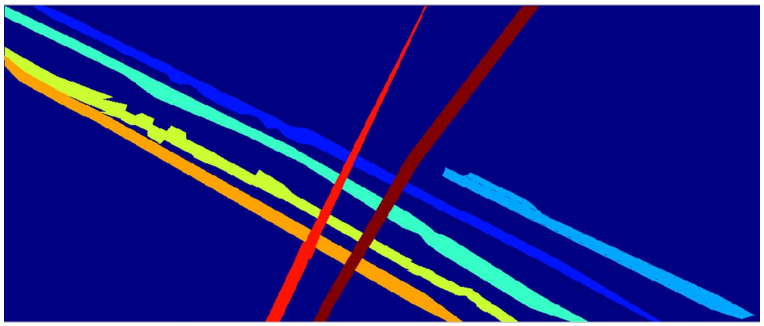
(a) Results of overpass detection.



(b) Results of driving direction estimation.

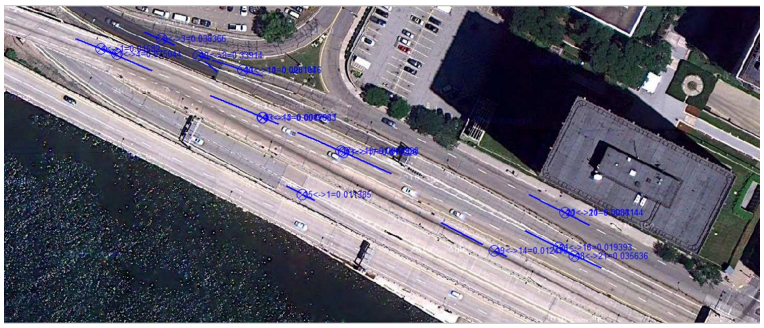


(c) Results of road-region segmentation.

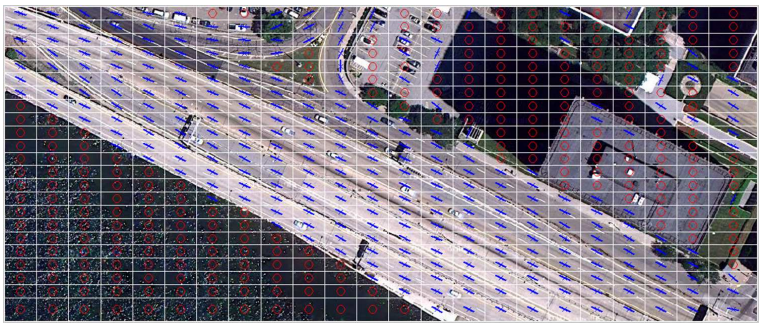


(d) Results of highway map generation.

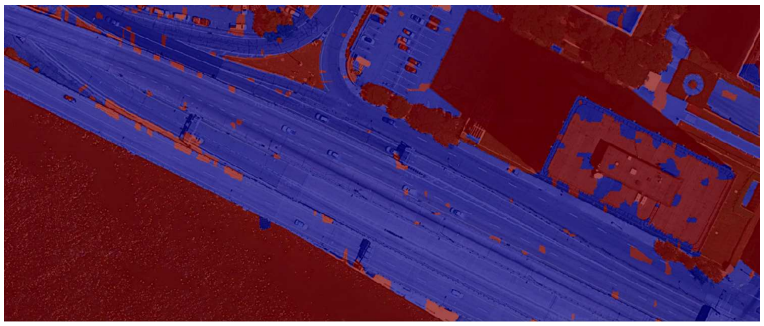
Figure C.11: Test highway orthoimage 11.



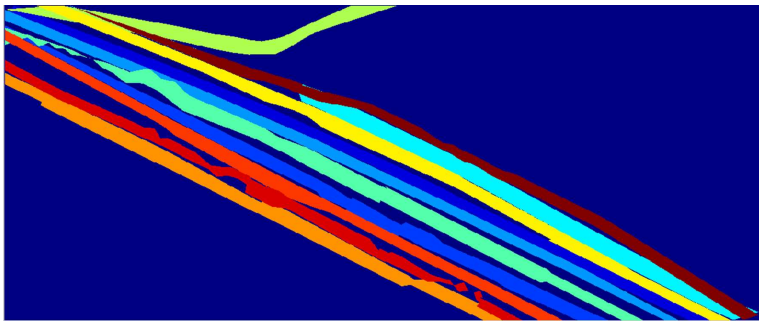
(a) Results of overpass detection.



(b) Results of driving direction estimation.



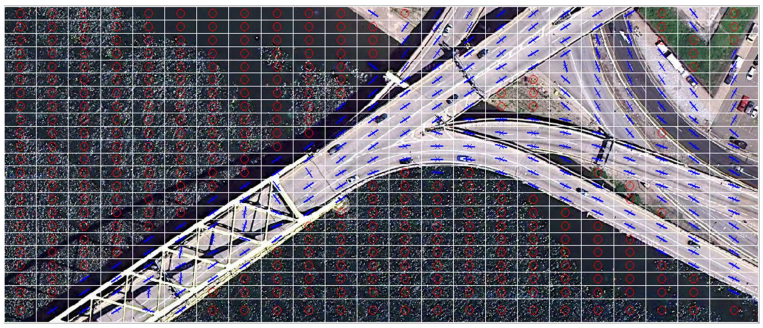
(c) Results of road-region segmentation.



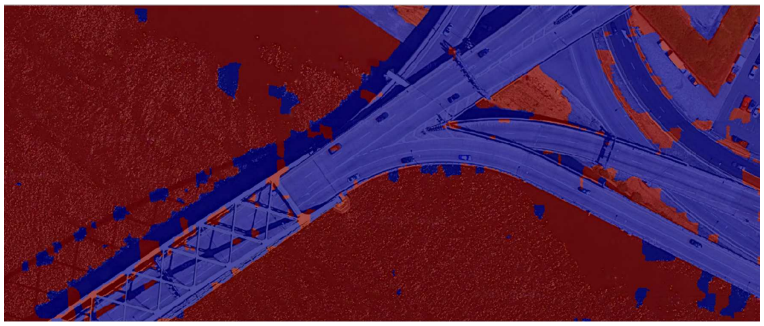
(d) Results of highway map generation.

Figure C.12: Test highway orthoimage 12.

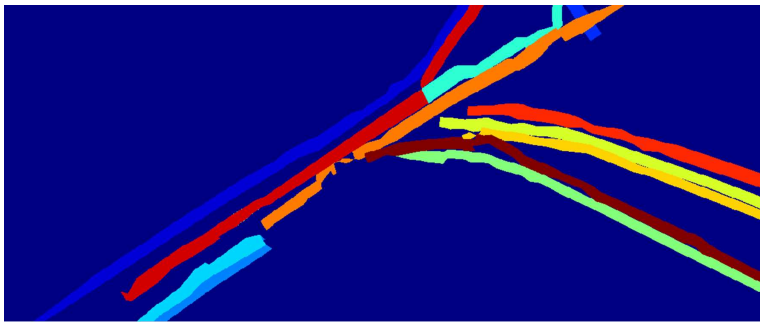
(a) Results of overpass detection.



(b) Results of driving direction estimation.

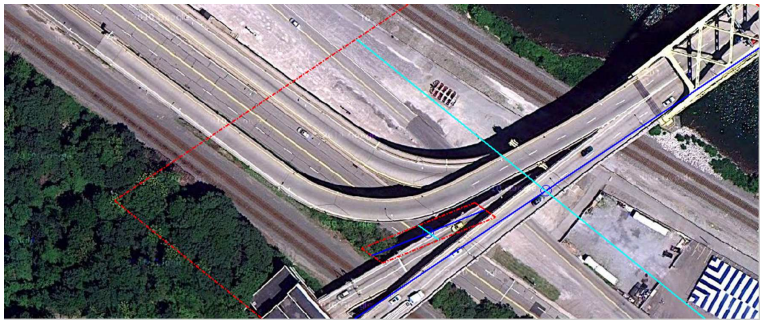


(c) Results of road-region segmentation.

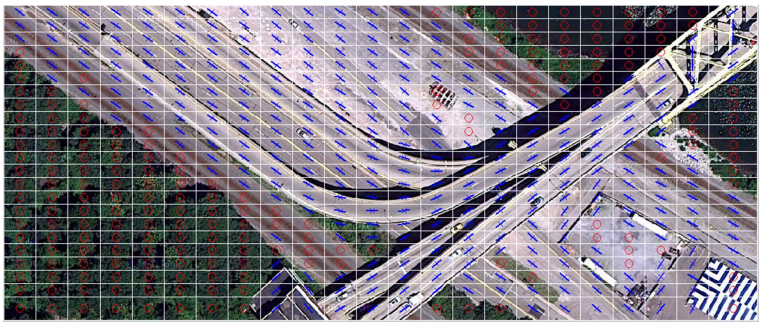


(d) Results of highway map generation.

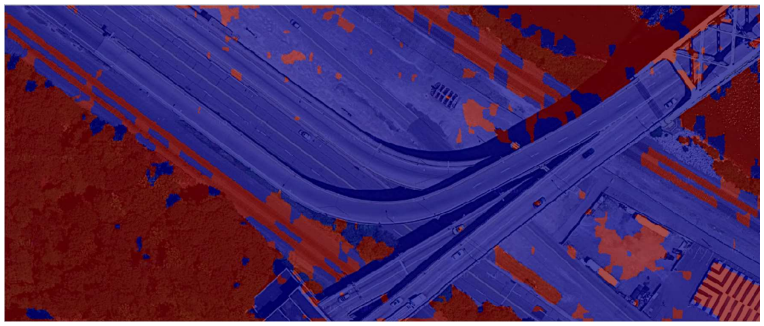
Figure C.13: Test highway orthoimage 13.



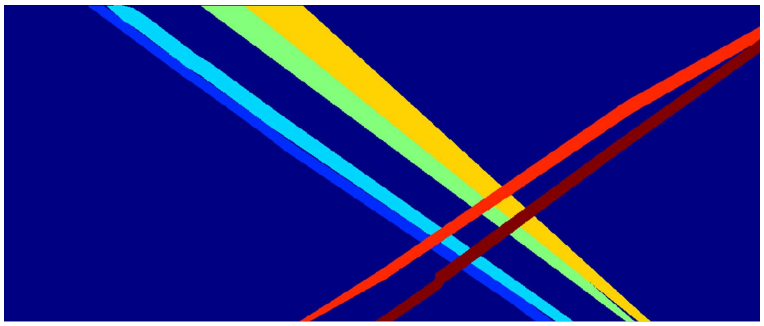
(a) Results of overpass detection.



(b) Results of driving direction estimation.

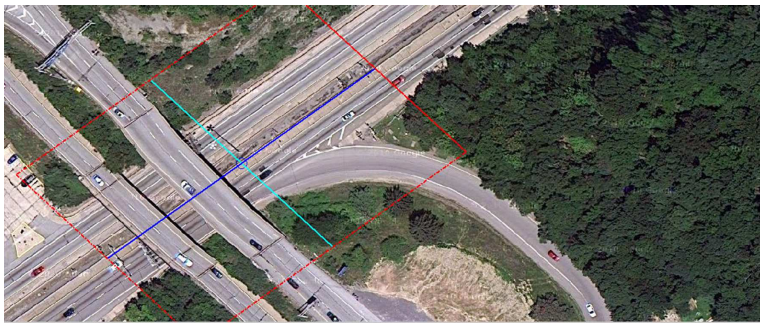


(c) Results of road-region segmentation.

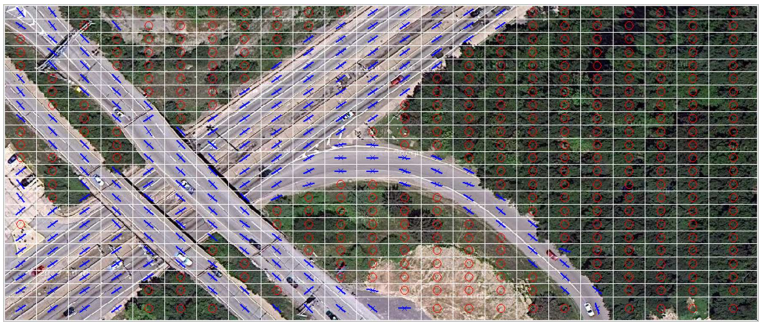


(d) Results of highway map generation.

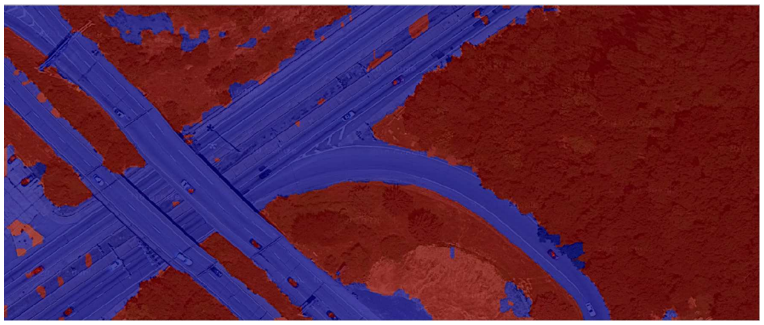
Figure C.14: Test highway orthoimage 14.



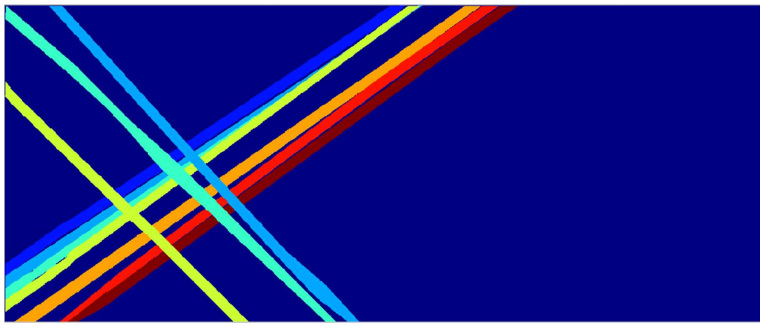
(a) Results of overpass detection.



(b) Results of driving direction estimation.

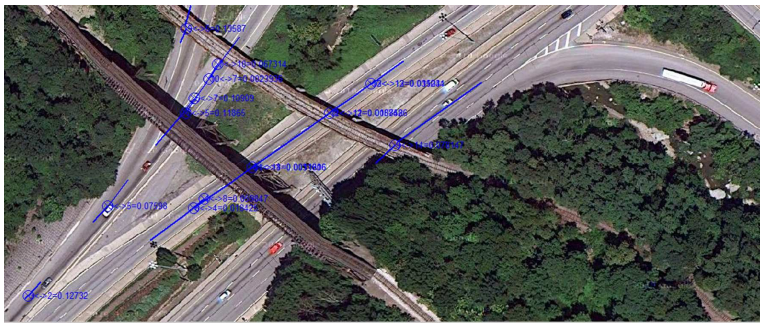


(c) Results of road-region segmentation.

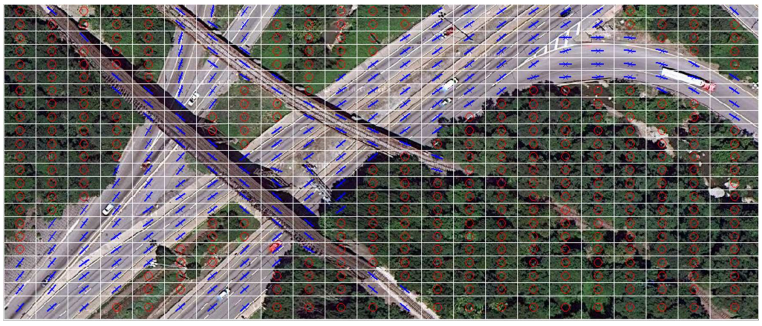


(d) Results of highway map generation.

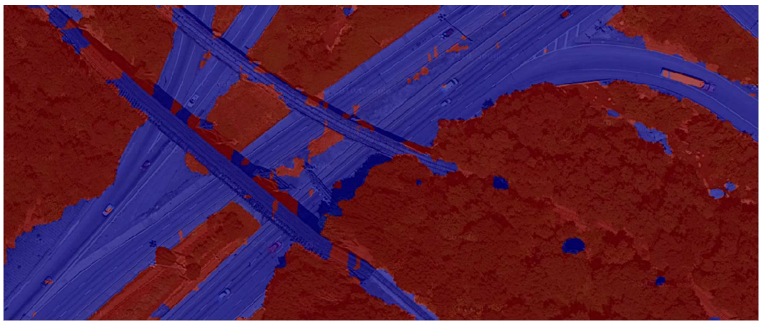
Figure C.15: Test highway orthoimage 15.



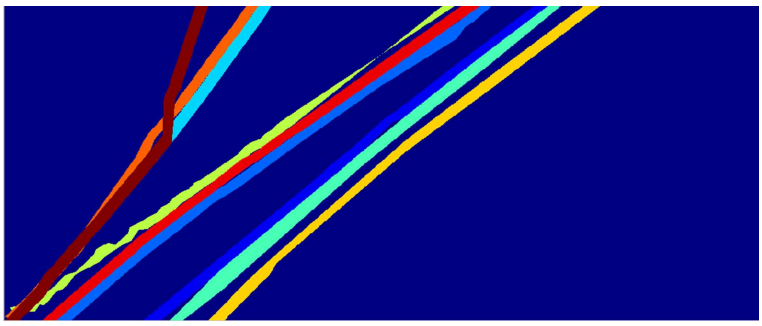
(a) Results of overpass detection.



(b) Results of driving direction estimation.

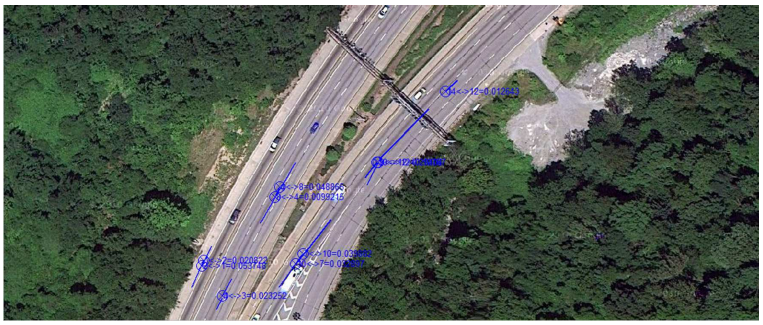


(c) Results of road-region segmentation.

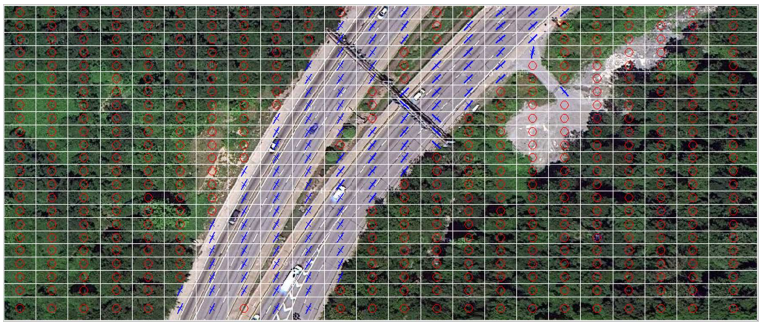


(d) Results of highway map generation.

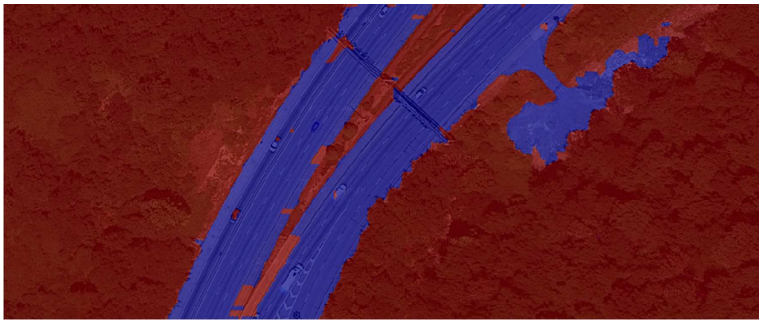
Figure C.16: Test highway orthoimage 16.



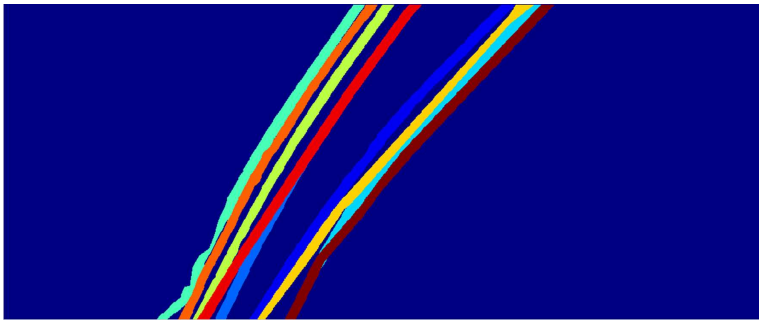
(a) Results of overpass detection.



(b) Results of driving direction estimation.

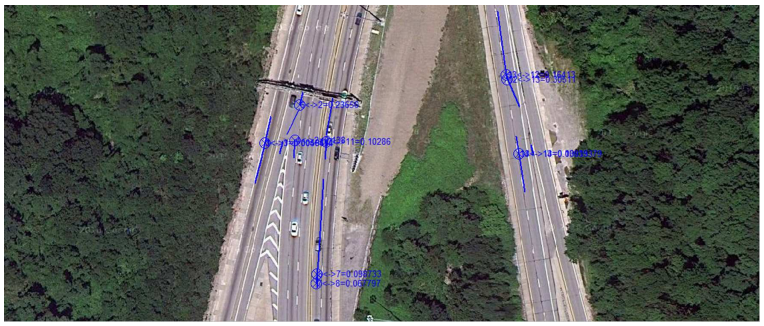


(c) Results of road-region segmentation.

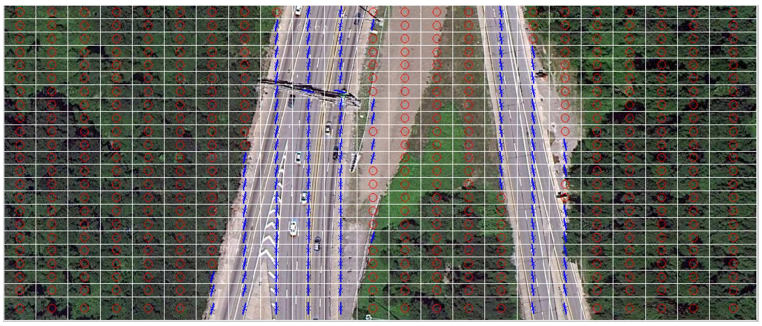


(d) Results of highway map generation.

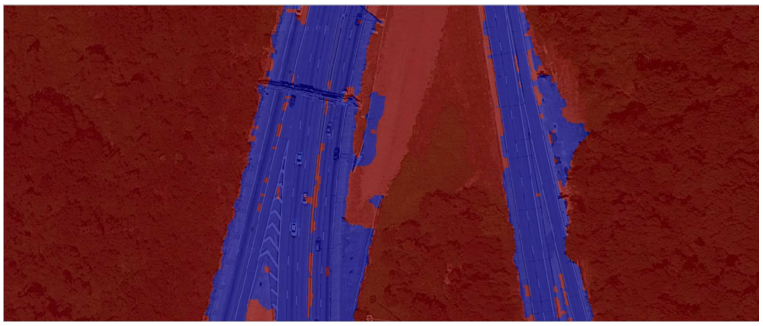
Figure C.17: Test highway orthoimage 17.



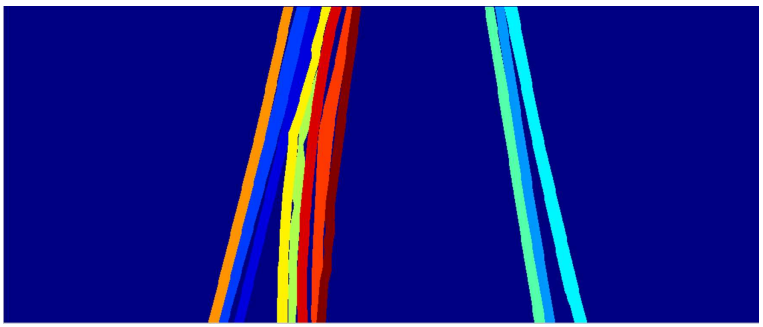
(a) Results of overpass detection.



(b) Results of driving direction estimation.

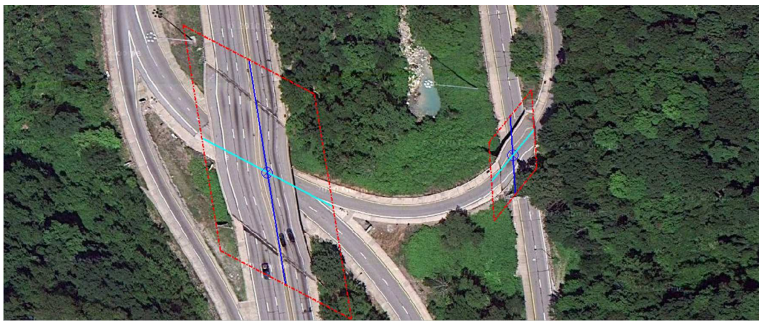


(c) Results of road-region segmentation.

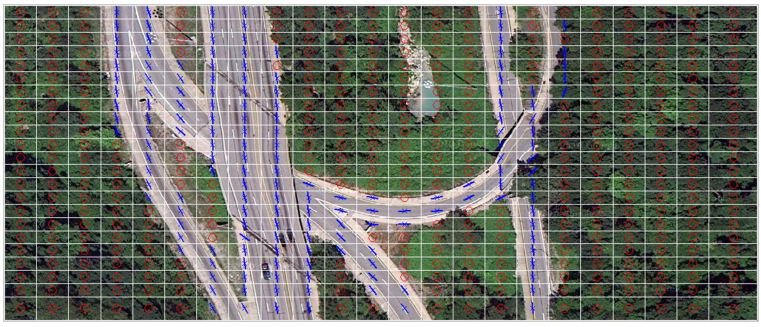


(d) Results of highway map generation.

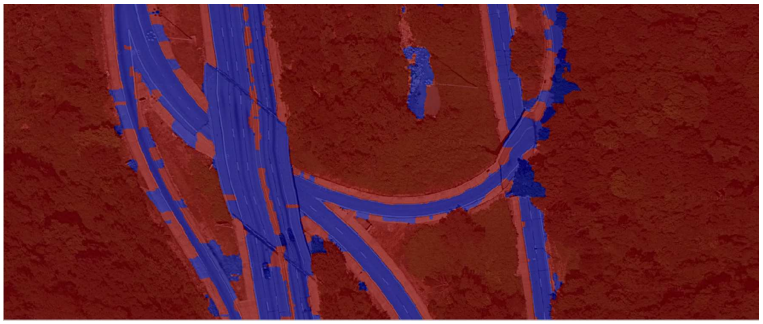
Figure C.18: Test highway orthoimage 18.



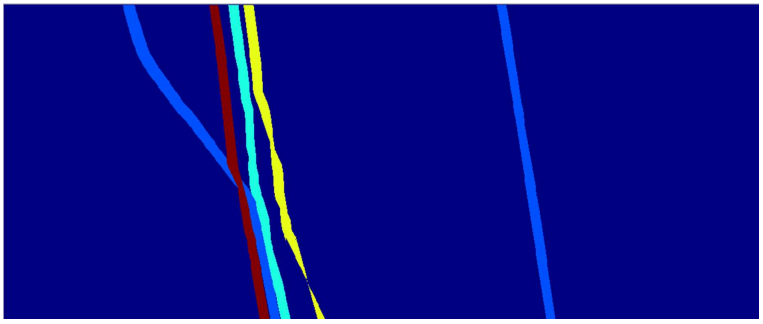
(a) Results of overpass detection.



(b) Results of driving direction estimation.

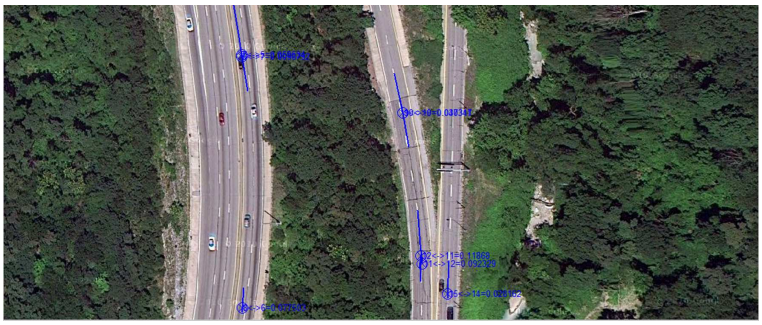


(c) Results of road-region segmentation.

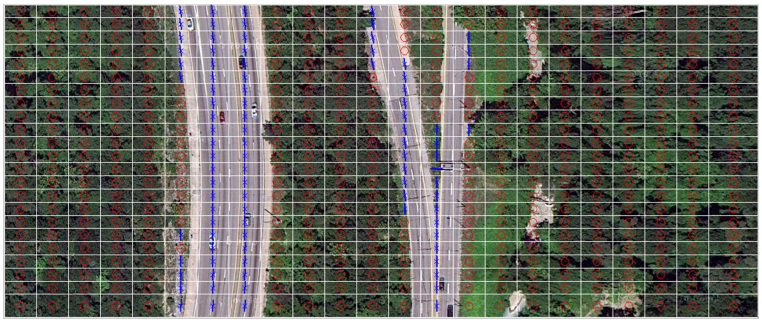


(d) Results of highway map generation.

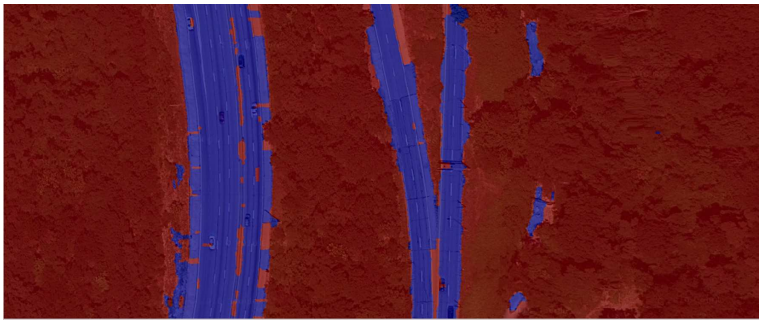
Figure C.19: Test highway orthoimage 19.



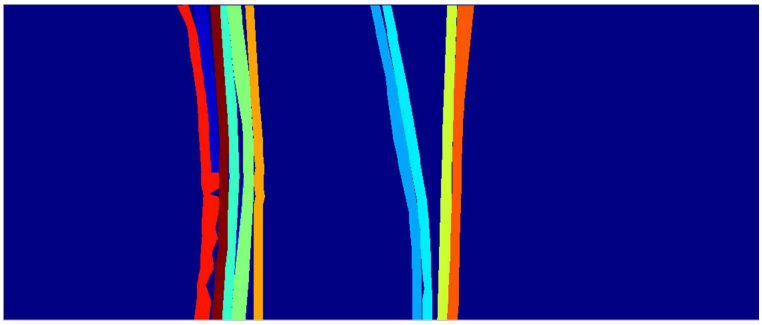
(a) Results of overpass detection.



(b) Results of driving direction estimation.



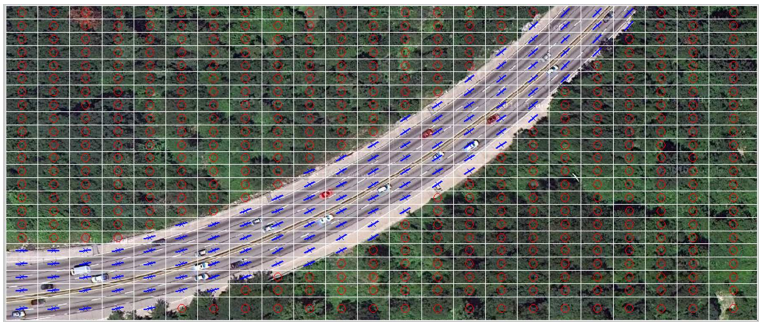
(c) Results of road-region segmentation.



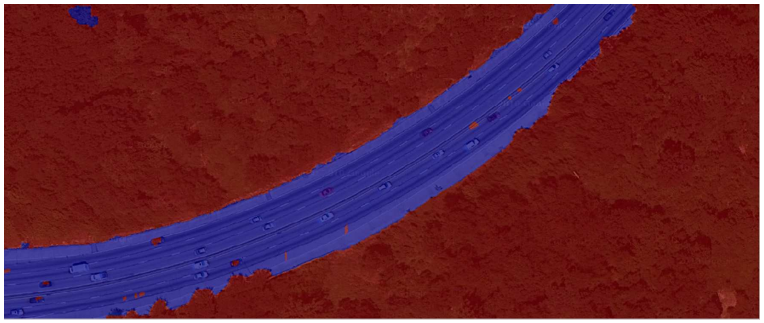
(d) Results of highway map generation.

Figure C.20: Test highway orthoimage 20.

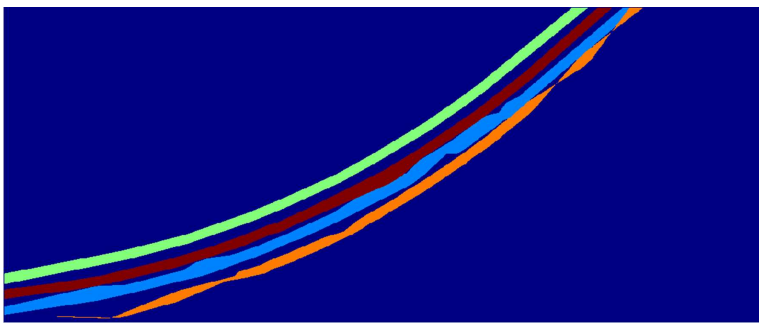
(a) Results of overpass detection.



(b) Results of driving direction estimation.



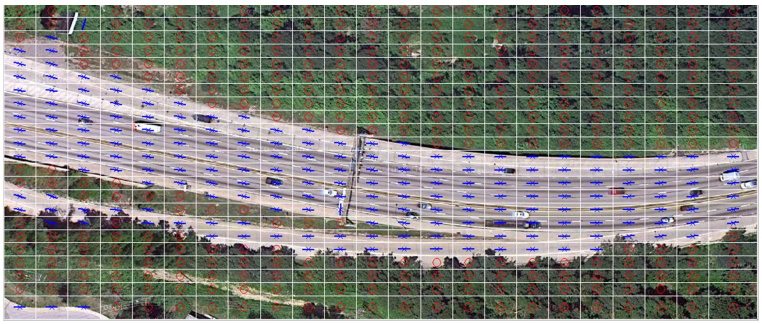
(c) Results of road-region segmentation.



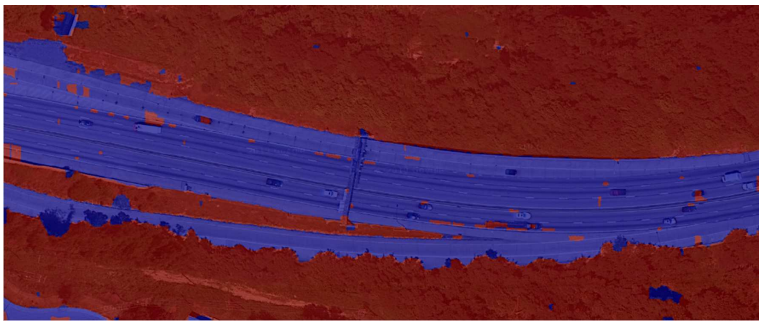
(d) Results of highway map generation.

Figure C.21: Test highway orthoimage 21.

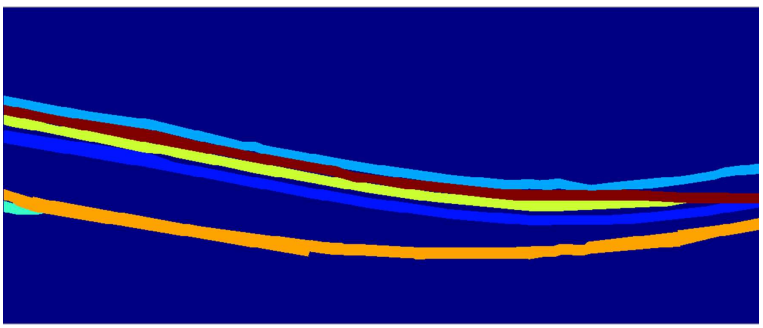
(a) Results of overpass detection.



(b) Results of driving direction estimation.

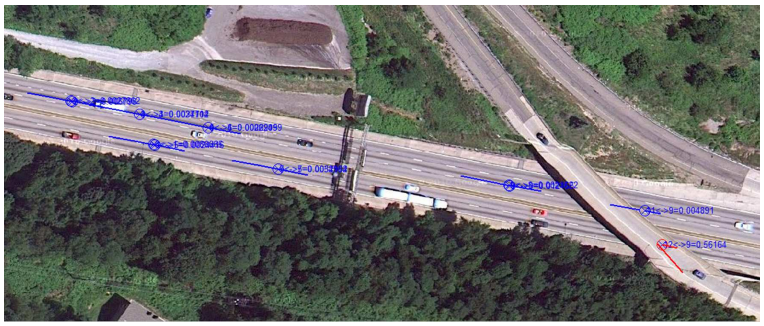


(c) Results of road-region segmentation.

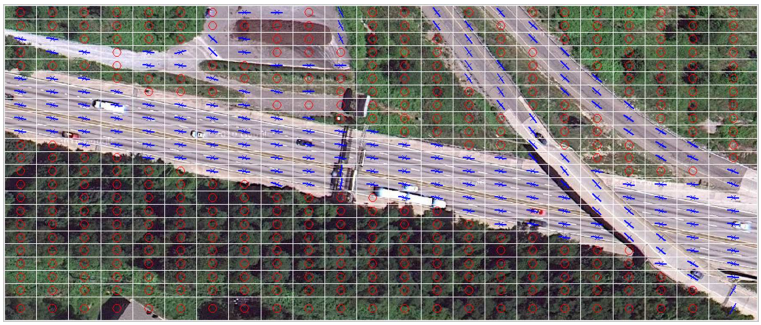


(d) Results of highway map generation.

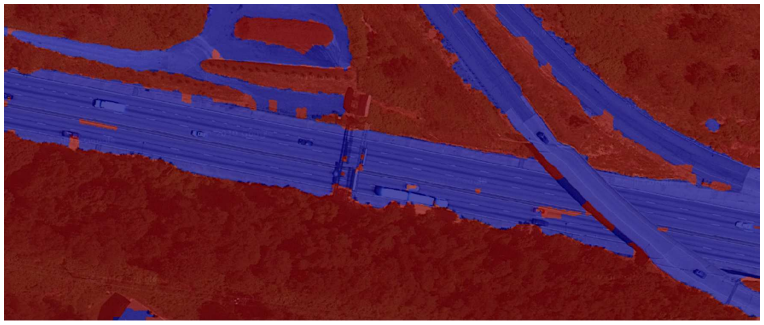
Figure C.22: Test highway orthoimage 22.



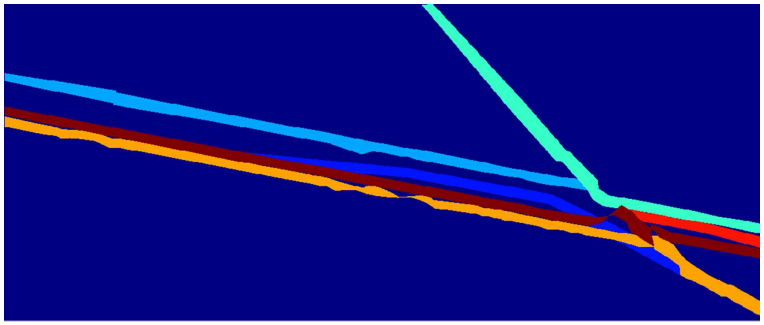
(a) Results of overpass detection.



(b) Results of driving direction estimation.

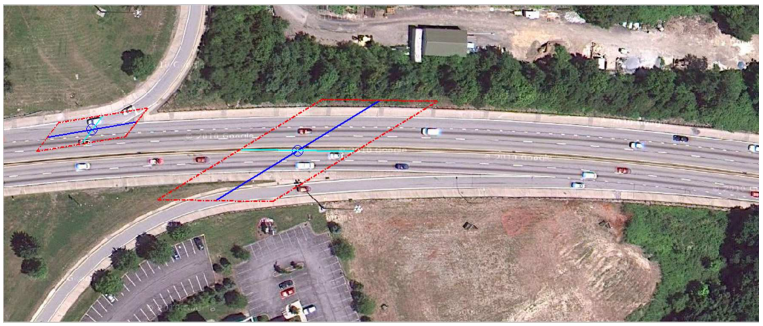


(c) Results of road-region segmentation.

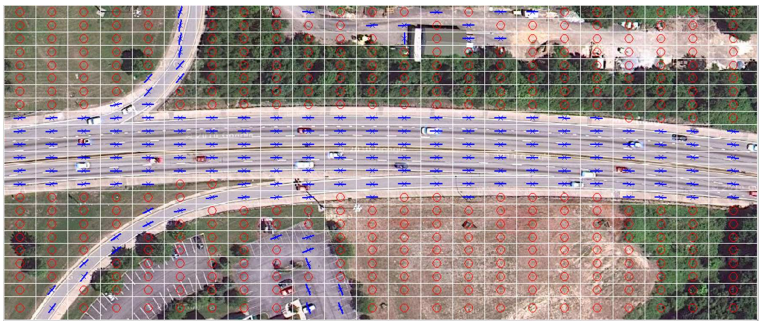


(d) Results of highway map generation.

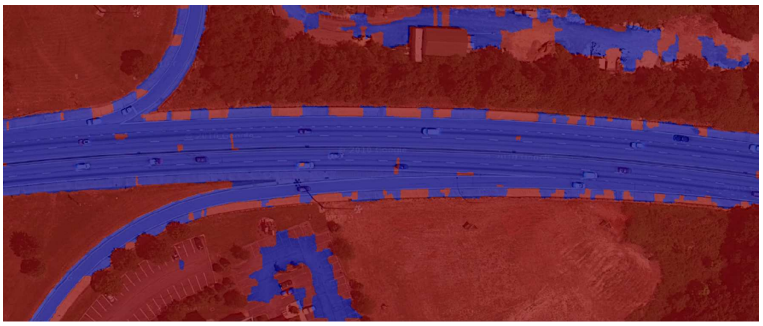
Figure C.23: Test highway orthoimage 23.



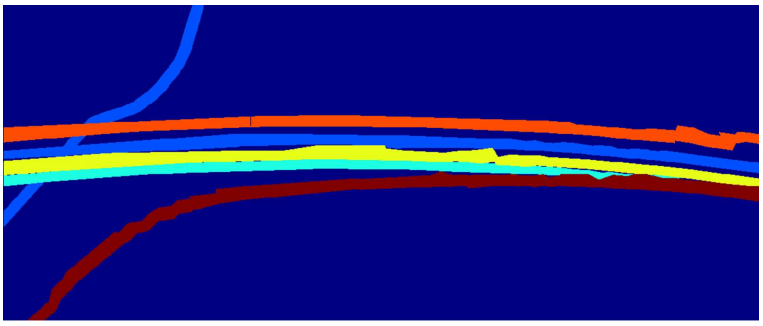
(a) Results of overpass detection.



(b) Results of driving direction estimation.



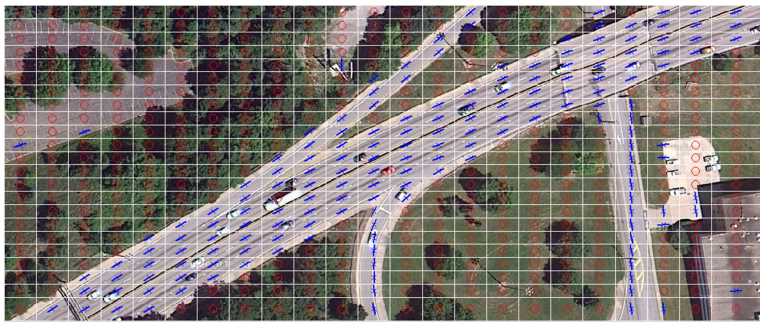
(c) Results of road-region segmentation.



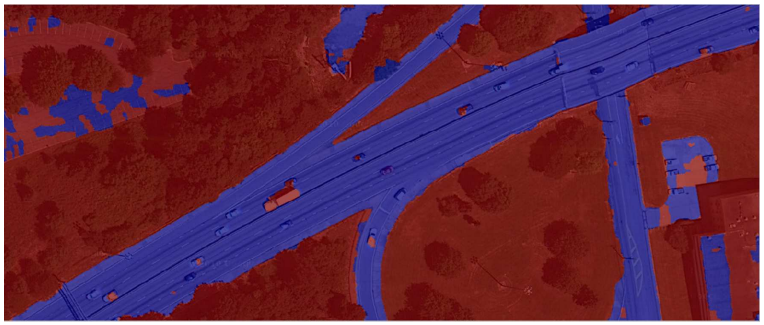
(d) Results of highway map generation.

Figure C.24: Test highway orthoimage 24.

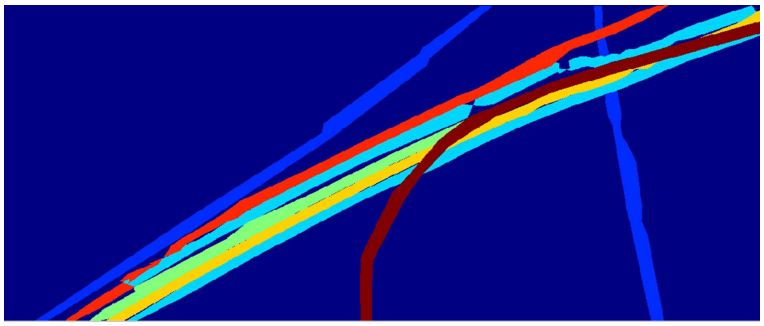
(a) Results of overpass detection.



(b) Results of driving direction estimation.



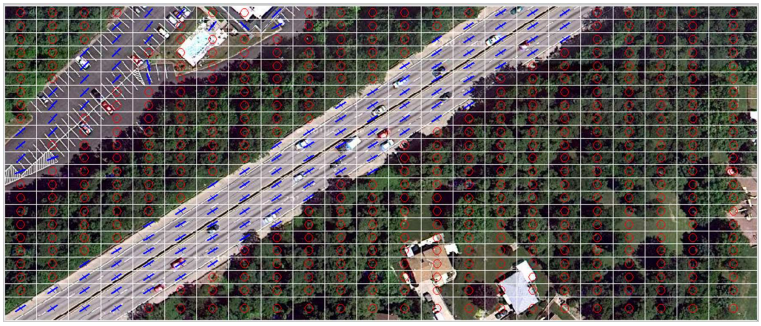
(c) Results of road-region segmentation.



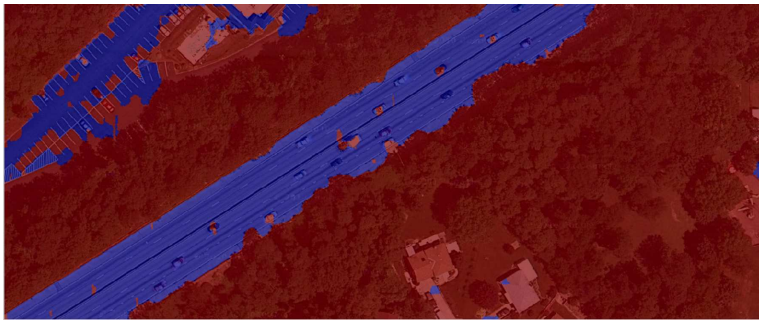
(d) Results of highway map generation.

Figure C.25: Test highway orthoimage 25.

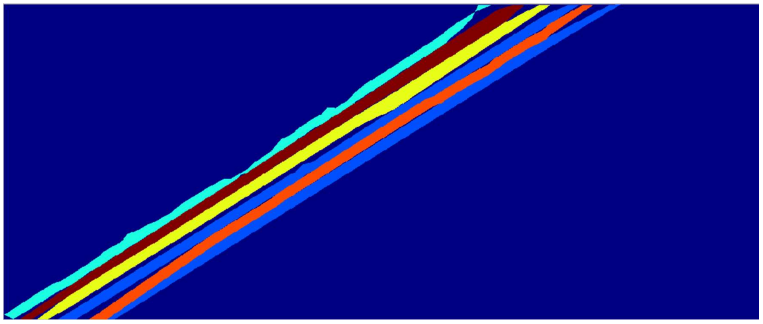
(a) Results of overpass detection.



(b) Results of driving direction estimation.



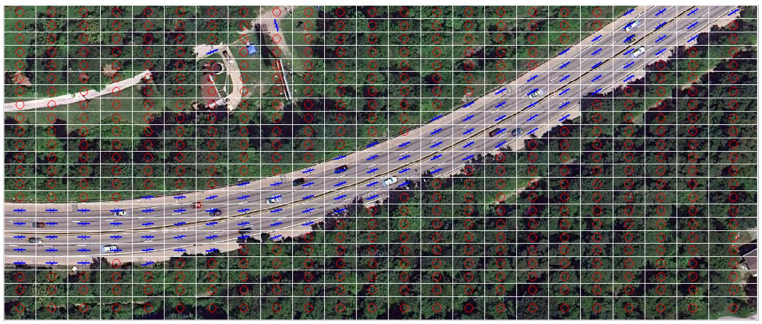
(c) Results of road-region segmentation.



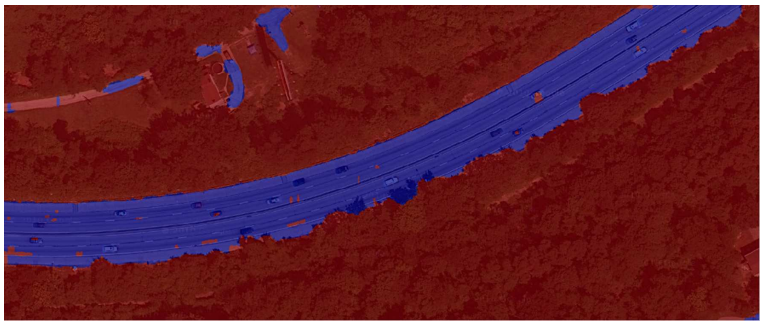
(d) Results of highway map generation.

Figure C.26: Test highway orthoimage 26.

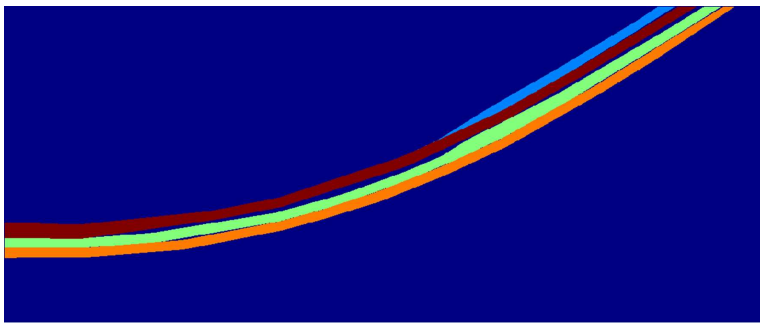
(a) Results of overpass detection.



(b) Results of driving direction estimation.



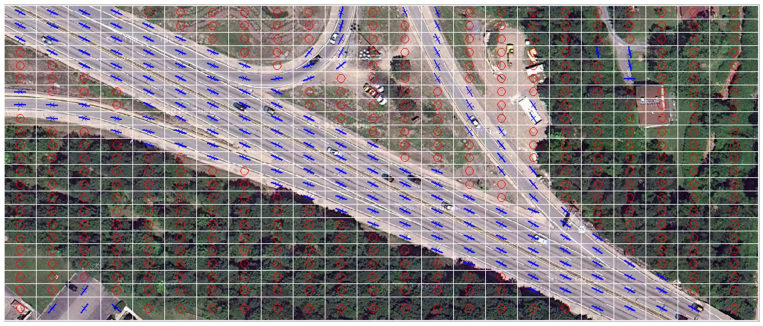
(c) Results of road-region segmentation.



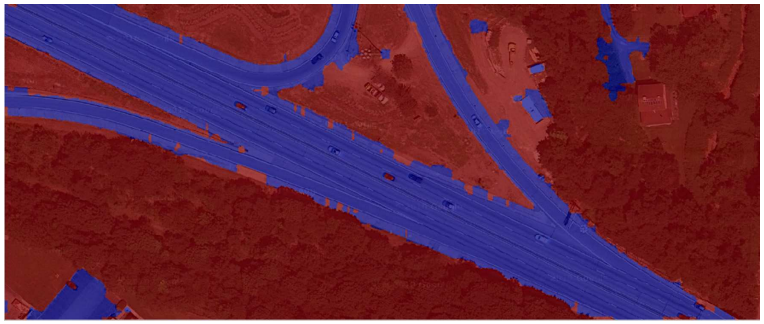
(d) Results of highway map generation.

Figure C.27: Test highway orthoimage 27.

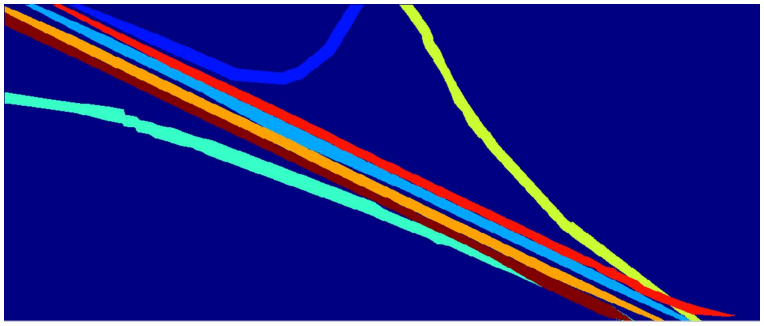
(a) Results of overpass detection.



(b) Results of driving direction estimation.

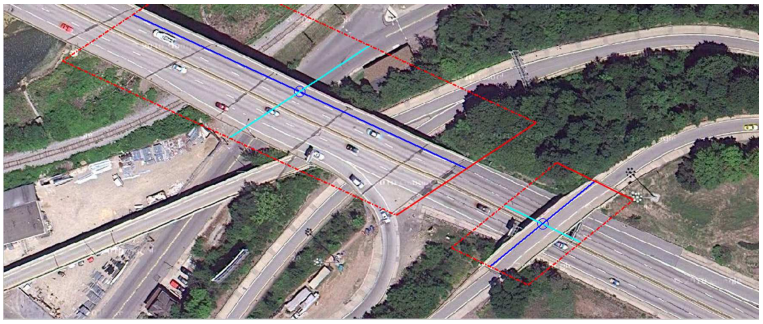


(c) Results of road-region segmentation.

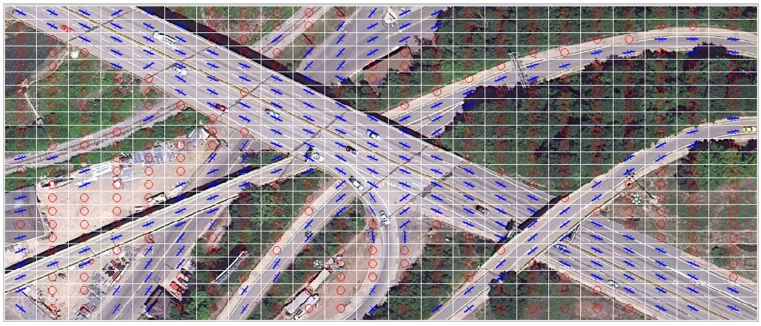


(d) Results of highway map generation.

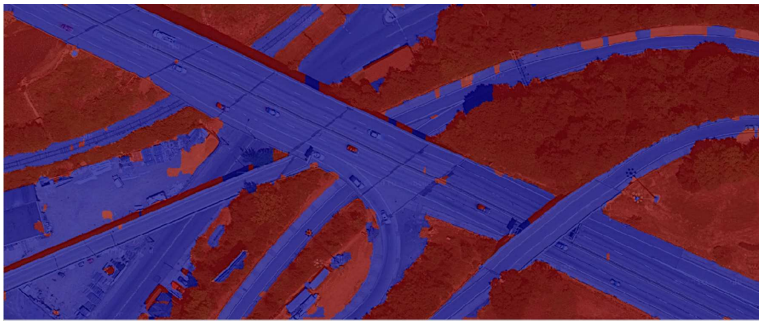
Figure C.28: Test highway orthoimage 28.



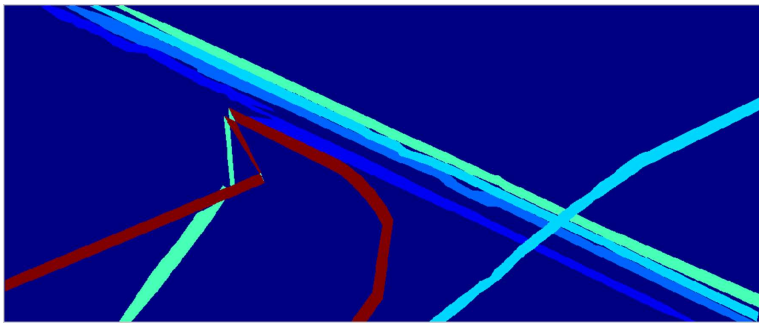
(a) Results of overpass detection.



(b) Results of driving direction estimation.

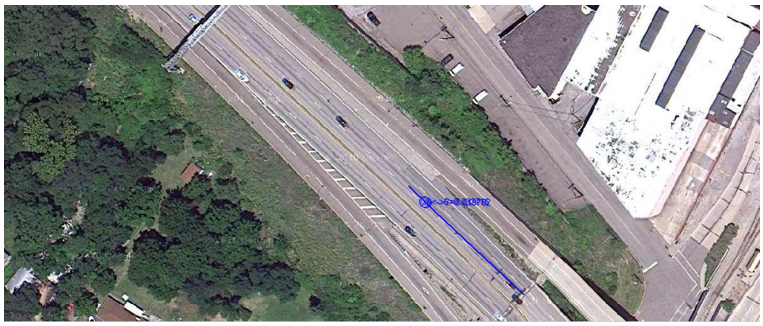


(c) Results of road-region segmentation.

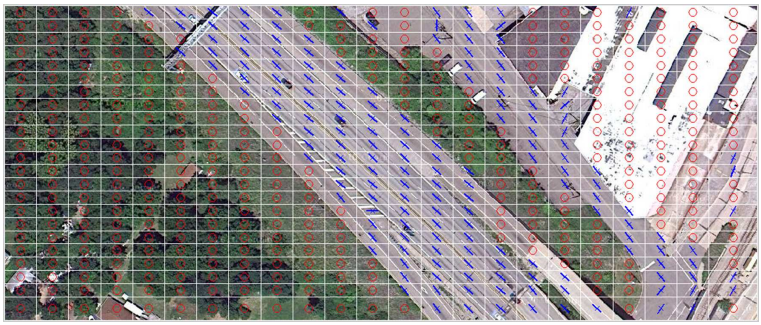


(d) Results of highway map generation.

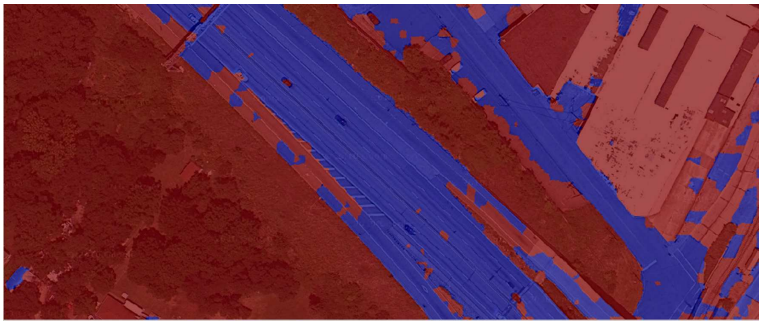
Figure C.29: Test highway orthoimage 29.



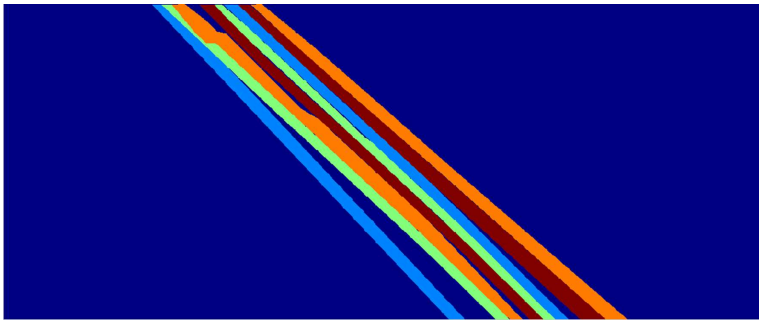
(a) Results of overpass detection.



(b) Results of driving direction estimation.

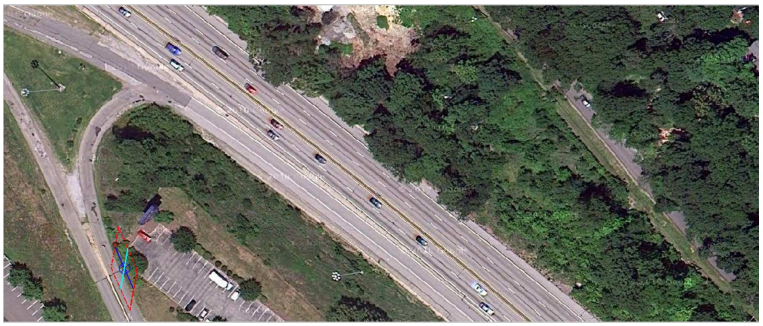


(c) Results of road-region segmentation.

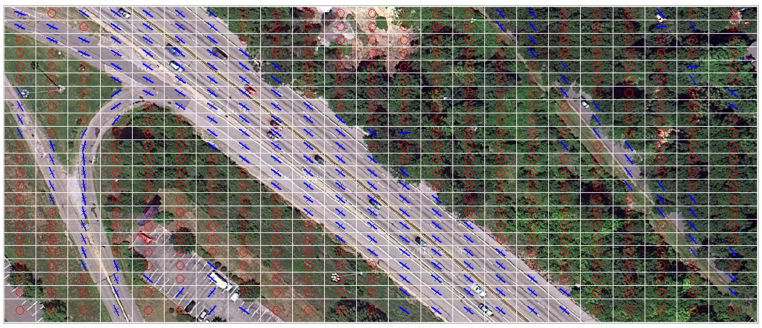


(d) Results of highway map generation.

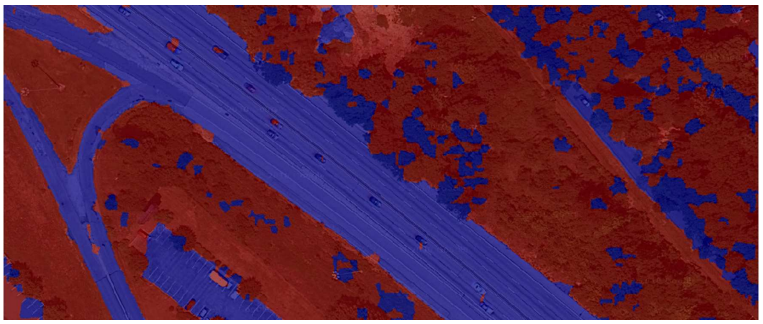
Figure C.30: Test highway orthoimage 30.



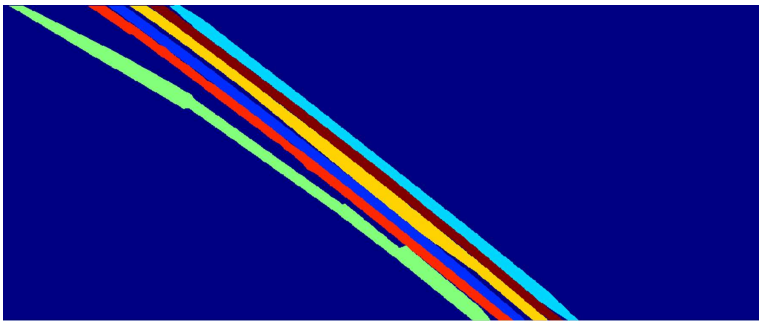
(a) Results of overpass detection.



(b) Results of driving direction estimation.

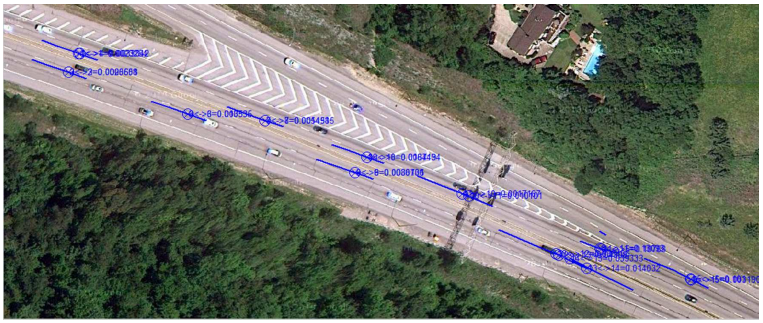


(c) Results of road-region segmentation.

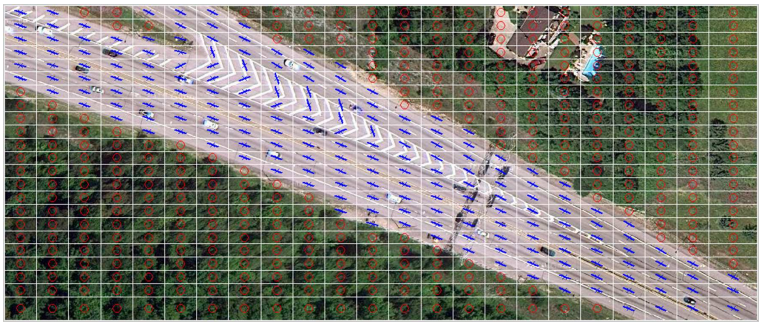


(d) Results of highway map generation.

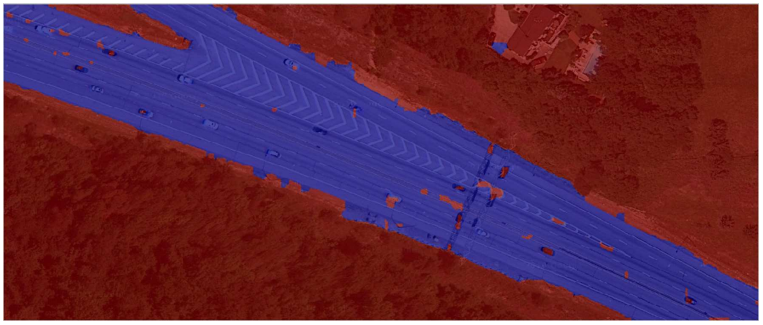
Figure C.31: Test highway orthoimage 31.



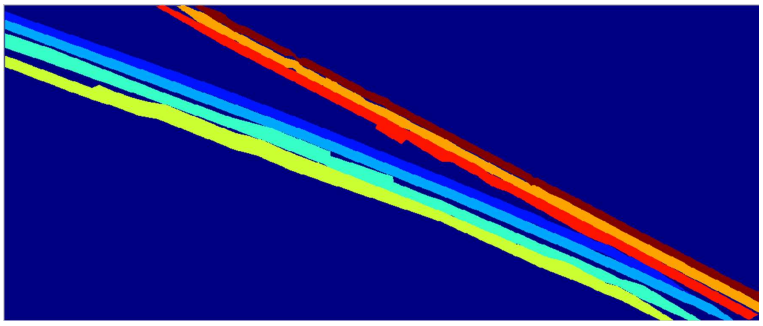
(a) Results of overpass detection.



(b) Results of driving direction estimation.

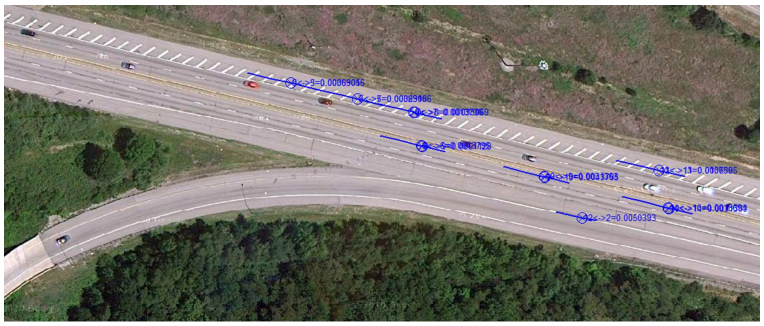


(c) Results of road-region segmentation.

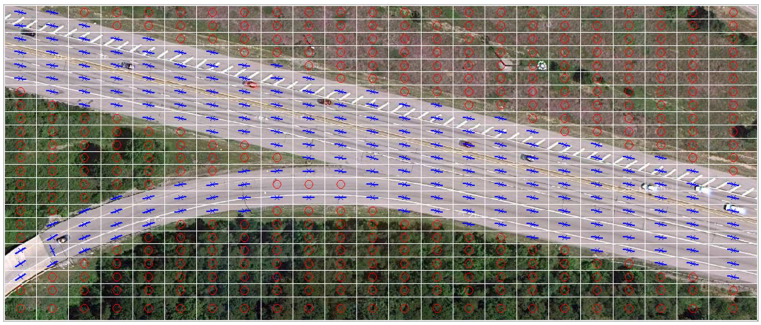


(d) Results of highway map generation.

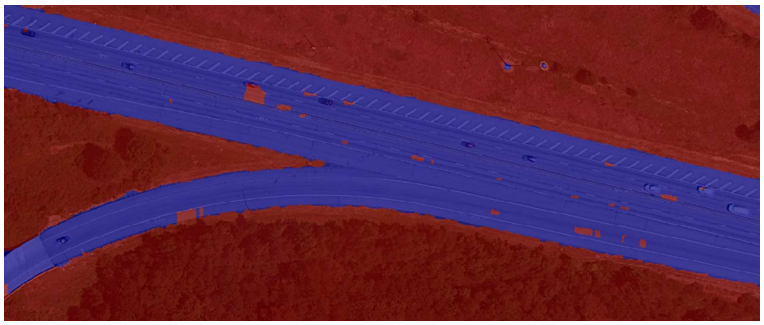
Figure C.32: Test highway orthoimage 32.



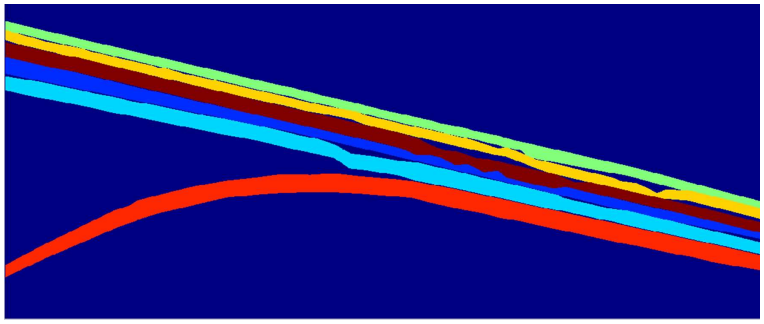
(a) Results of overpass detection.



(b) Results of driving direction estimation.

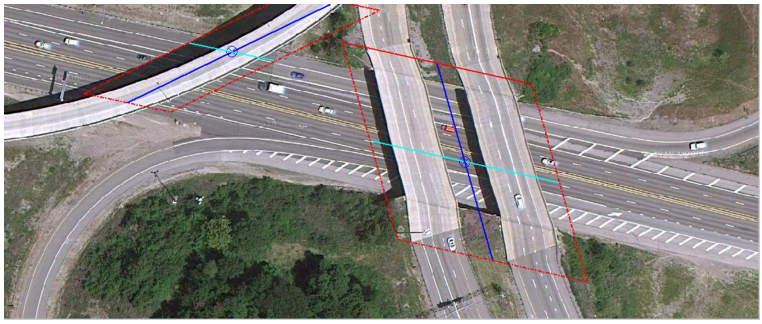


(c) Results of road-region segmentation.

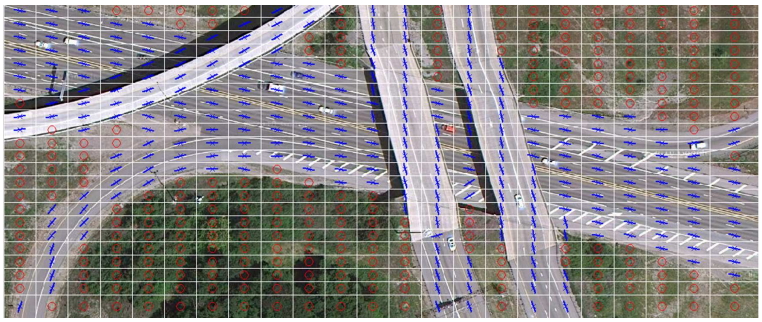


(d) Results of highway map generation.

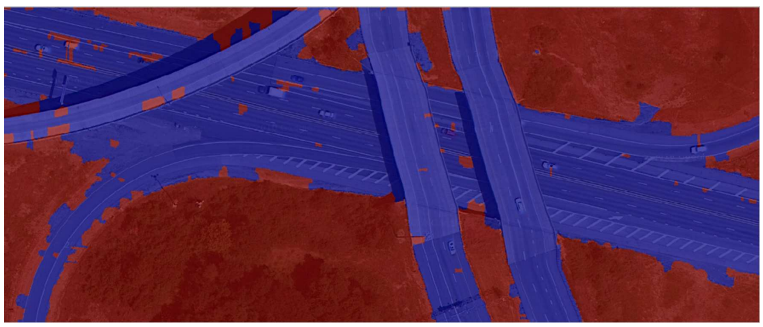
Figure C.33: Test highway orthoimage 33.



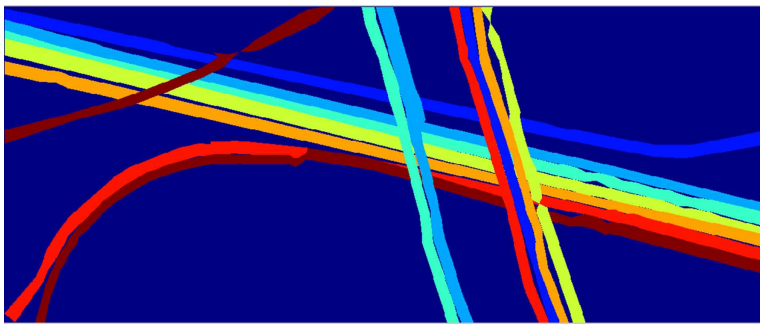
(a) Results of overpass detection.



(b) Results of driving direction estimation.

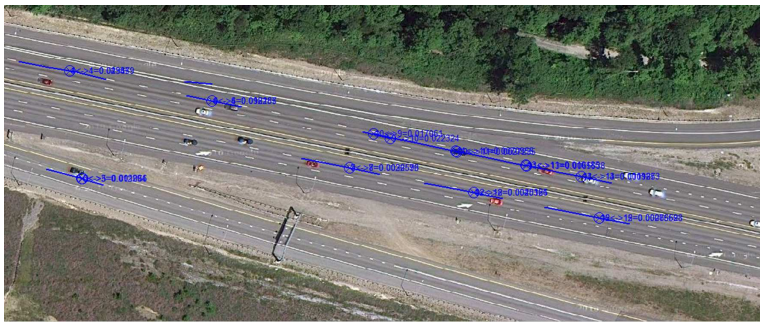


(c) Results of road-region segmentation.

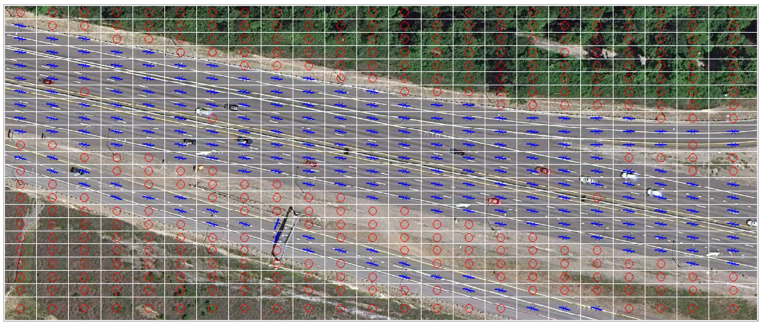


(d) Results of highway map generation.

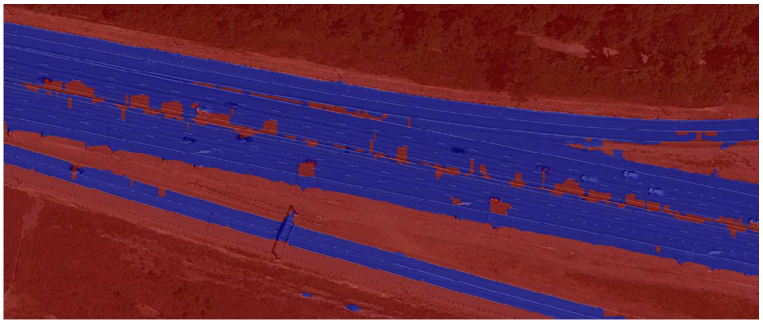
Figure C.34: Test highway orthoimage 34.



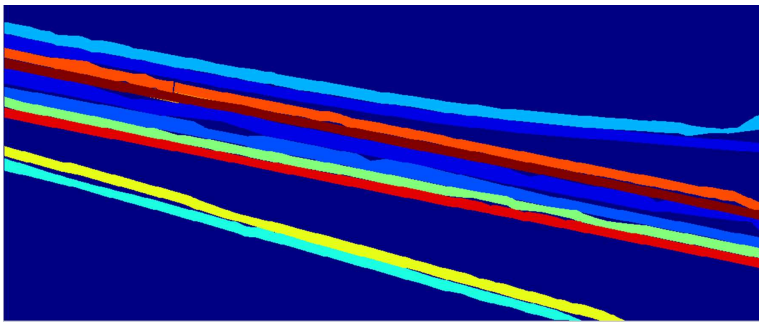
(a) Results of overpass detection.



(b) Results of driving direction estimation.



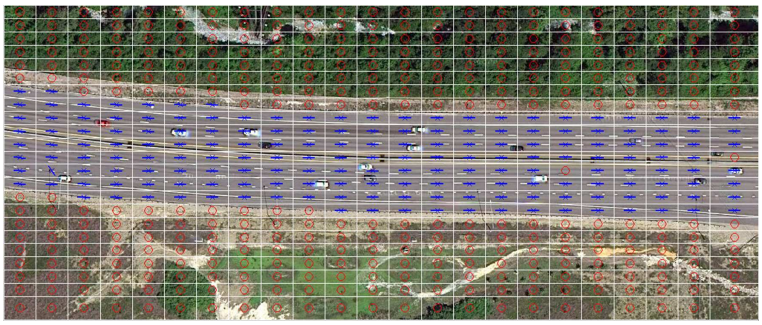
(c) Results of road-region segmentation.



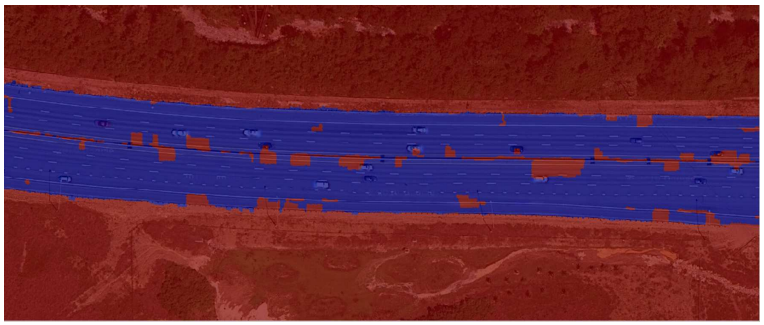
(d) Results of highway map generation.

Figure C.35: Test highway orthoimage 35.

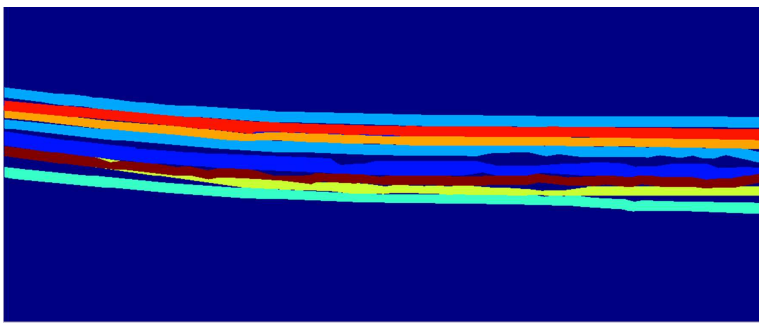
(a) Results of overpass detection.



(b) Results of driving direction estimation.



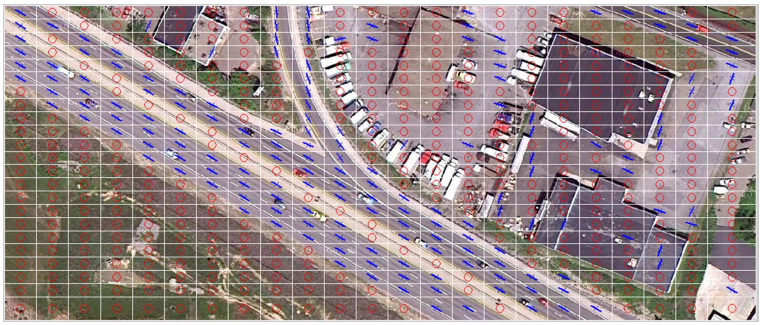
(c) Results of road-region segmentation.



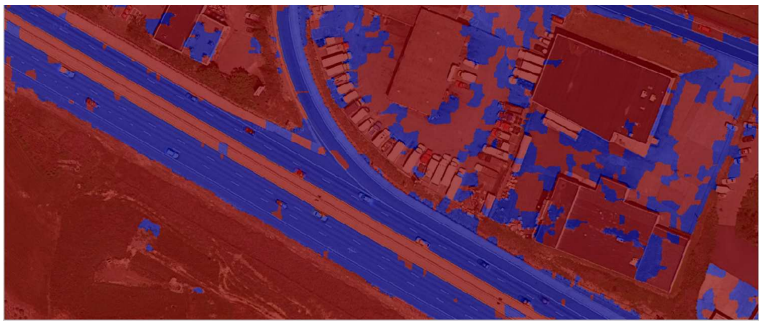
(d) Results of highway map generation.

Figure C.36: Test highway orthoimage 36.

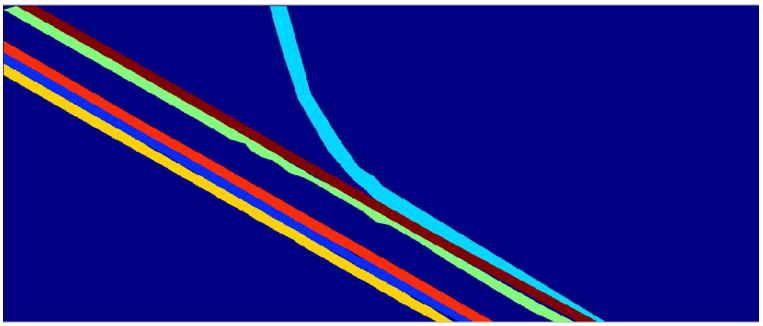
(a) Results of overpass detection.



(b) Results of driving direction estimation.

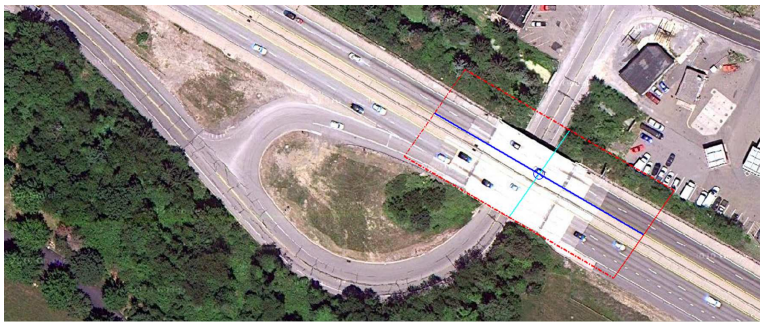


(c) Results of road-region segmentation.

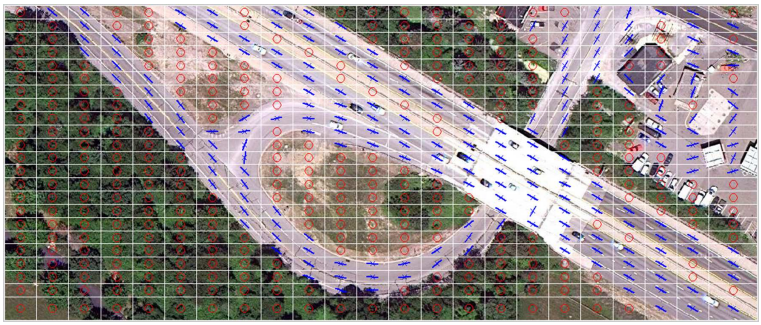


(d) Results of highway map generation.

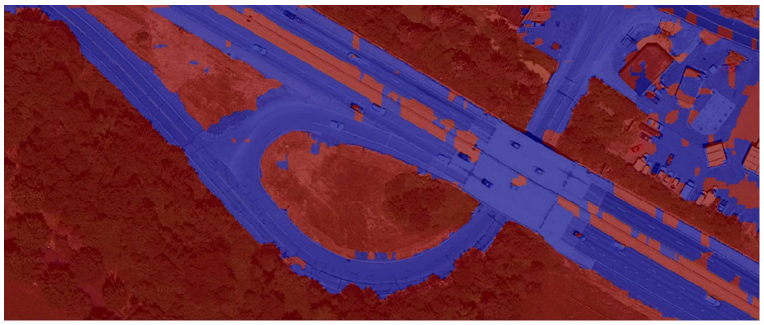
Figure C.37: Test highway orthoimage 37.



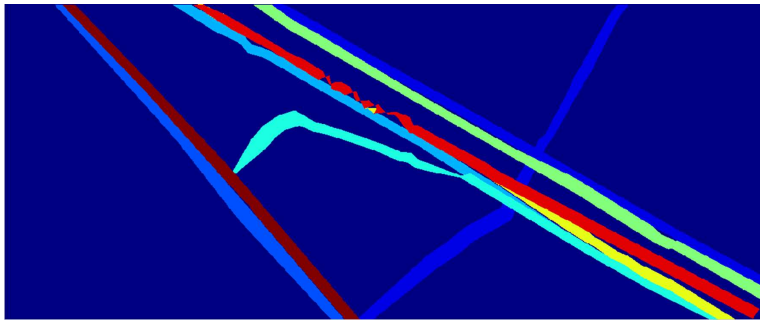
(a) Results of overpass detection.



(b) Results of driving direction estimation.

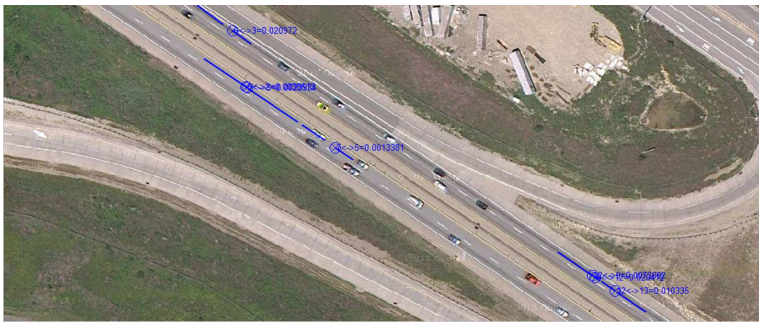


(c) Results of road-region segmentation.

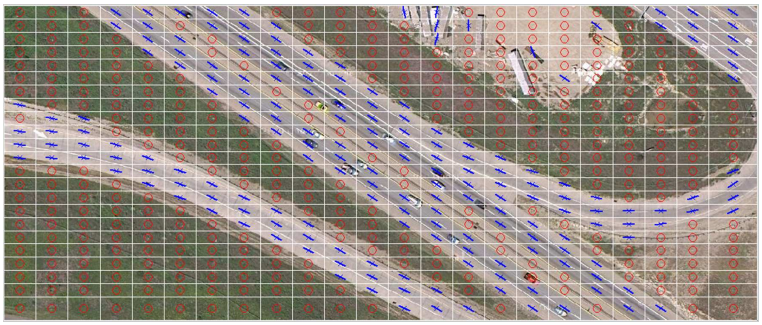


(d) Results of highway map generation.

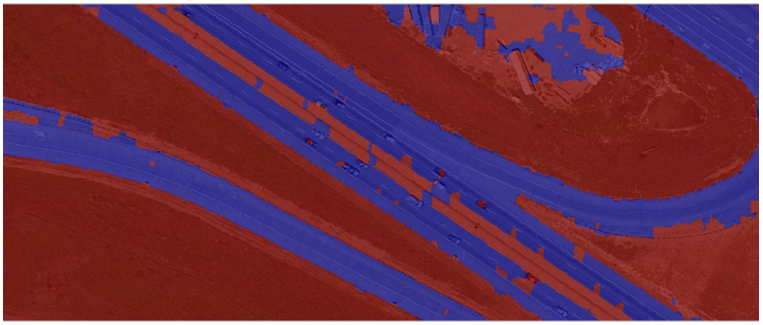
Figure C.38: Test highway orthoimage 38.



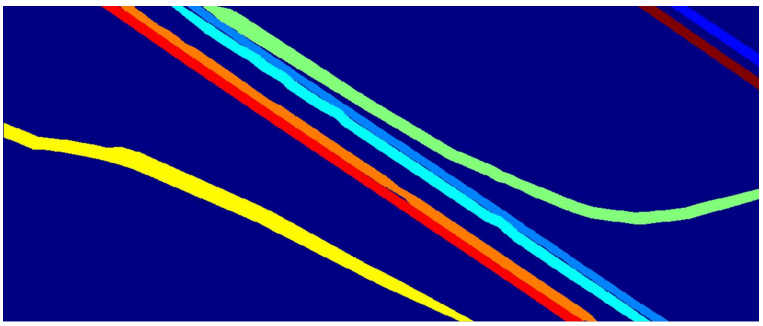
(a) Results of overpass detection.



(b) Results of driving direction estimation.



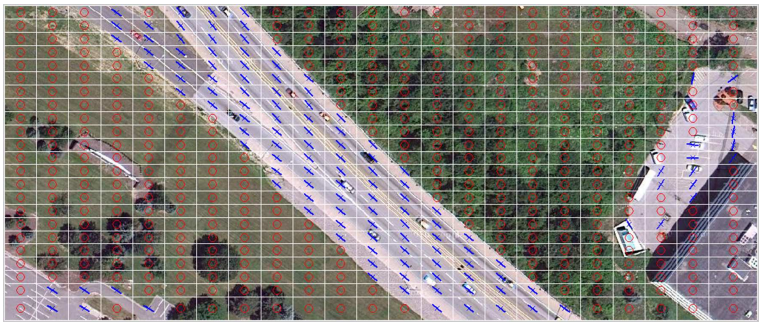
(c) Results of road-region segmentation.



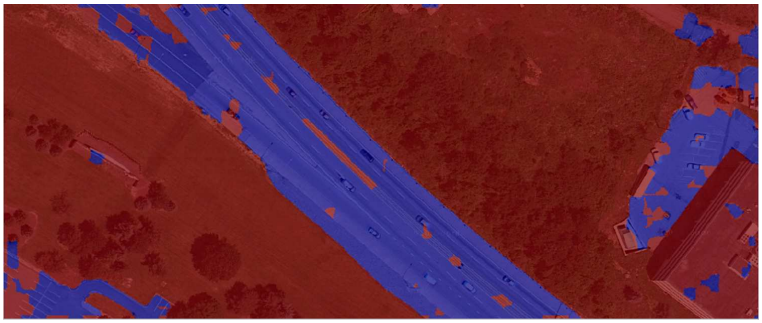
(d) Results of highway map generation.

Figure C.39: Test highway orthoimage 39.

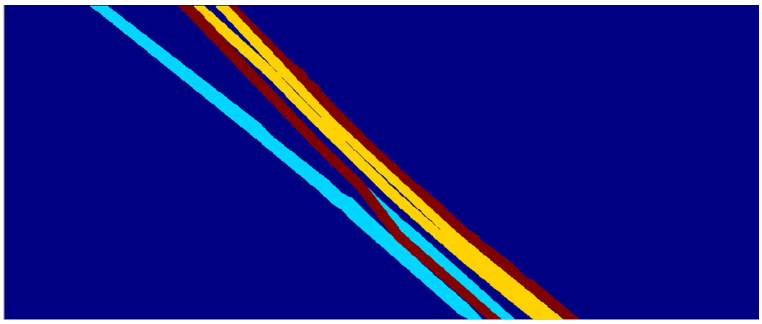
(a) Results of overpass detection.



(b) Results of driving direction estimation.

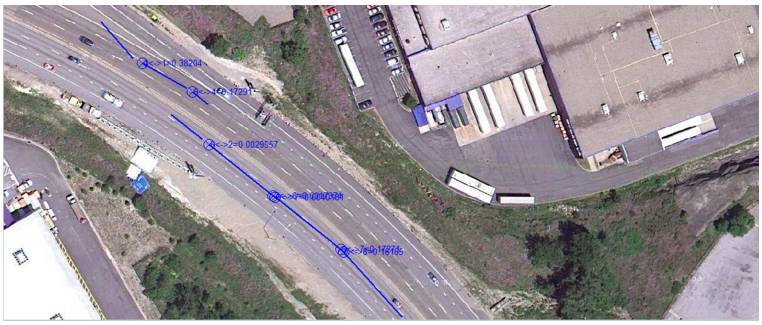


(c) Results of road-region segmentation.

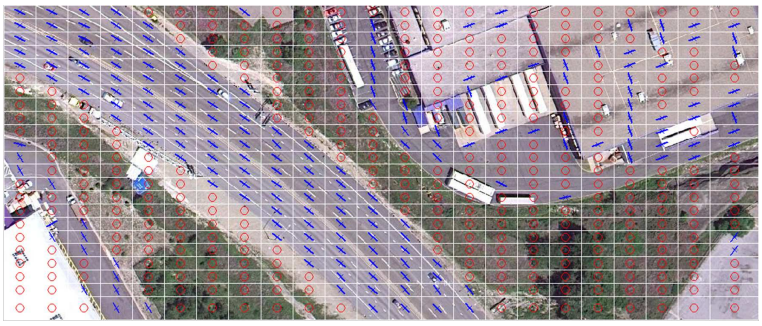


(d) Results of highway map generation.

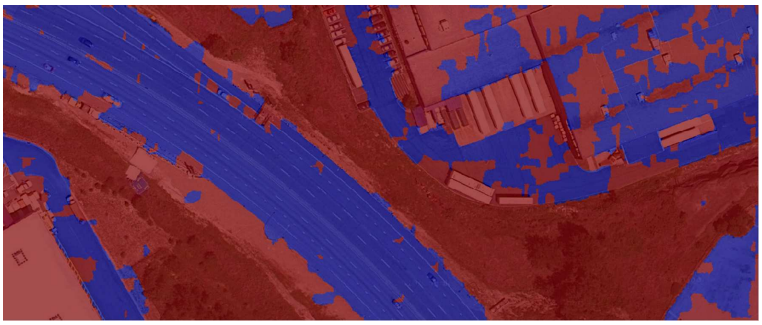
Figure C.40: Test highway orthoimage 40.



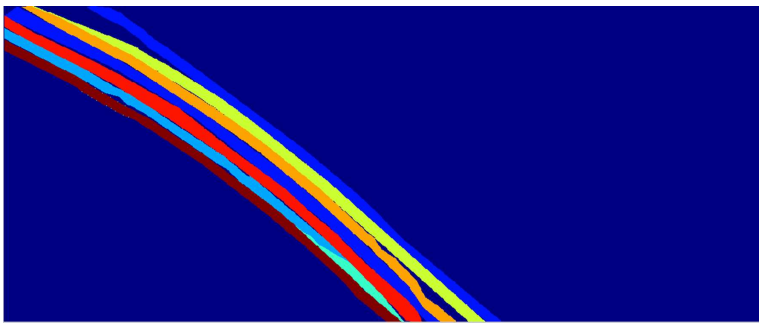
(a) Results of overpass detection.



(b) Results of driving direction estimation.

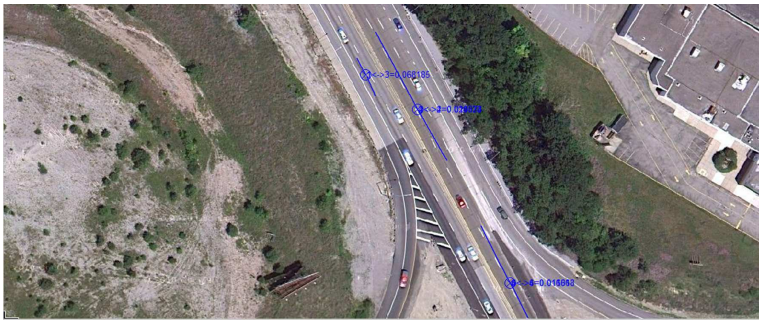


(c) Results of road-region segmentation.

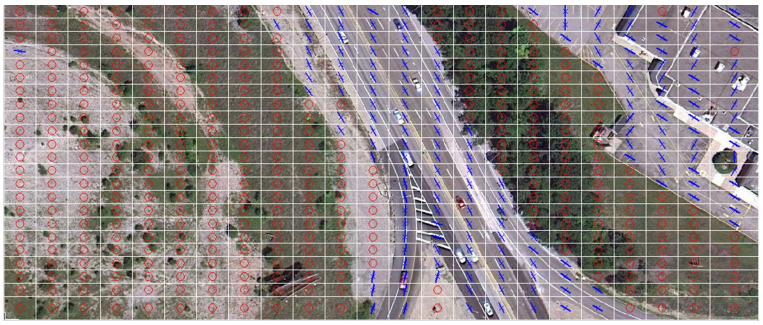


(d) Results of highway map generation.

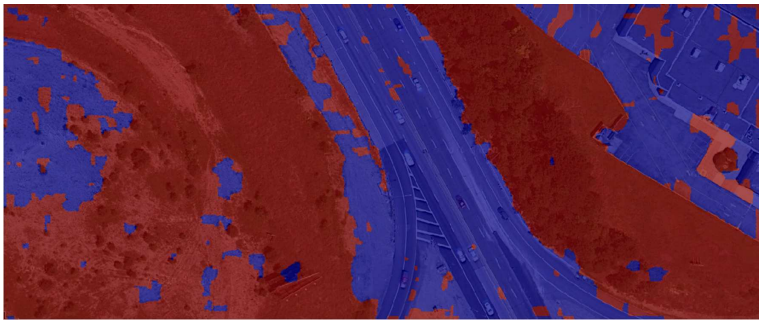
Figure C.41: Test highway orthoimage 41.



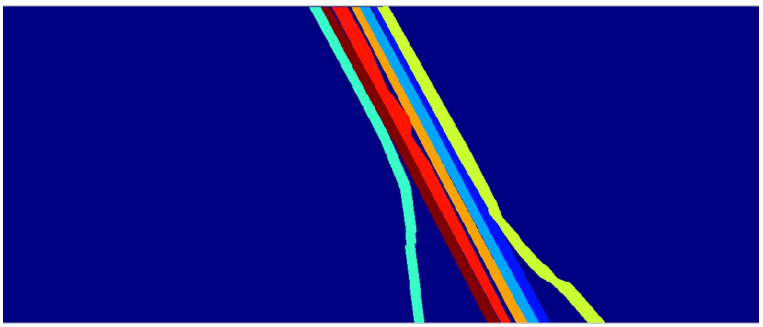
(a) Results of overpass detection.



(b) Results of driving direction estimation.

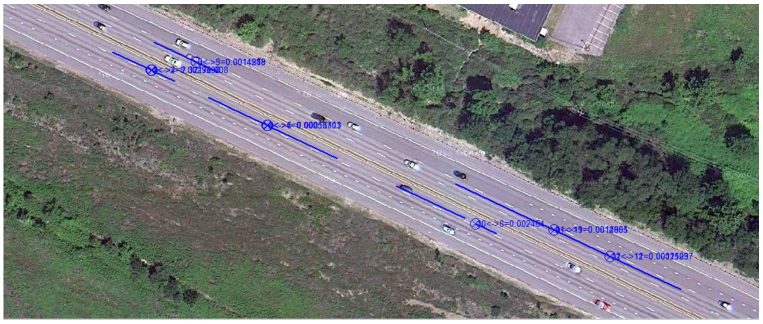


(c) Results of road-region segmentation.

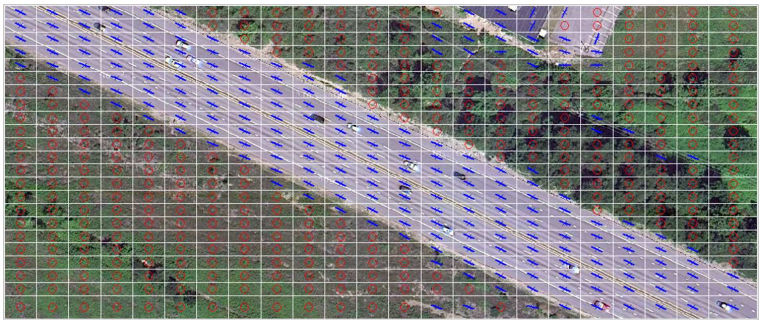


(d) Results of highway map generation.

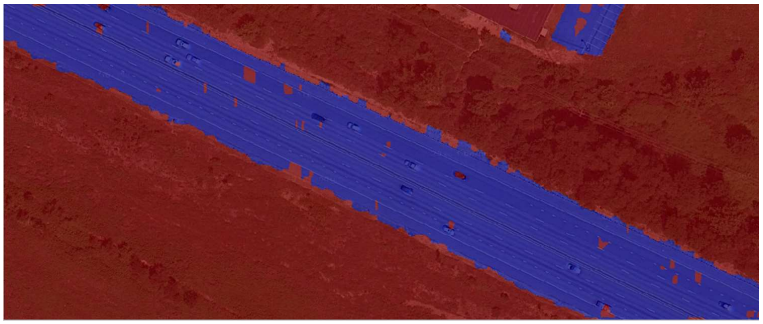
Figure C.42: Test highway orthoimage 42.



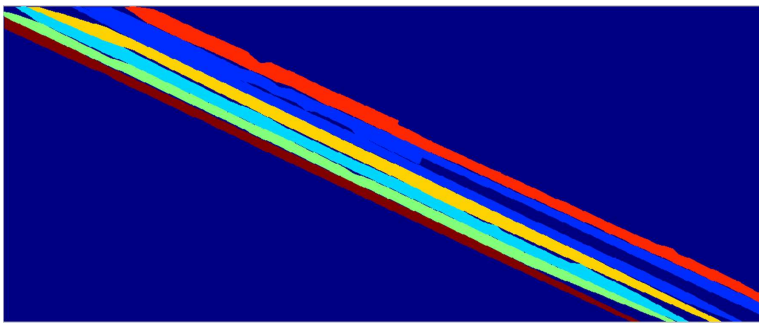
(a) Results of overpass detection.



(b) Results of driving direction estimation.



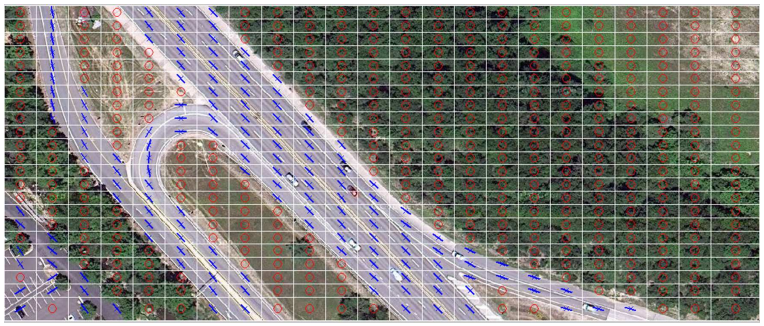
(c) Results of road-region segmentation.



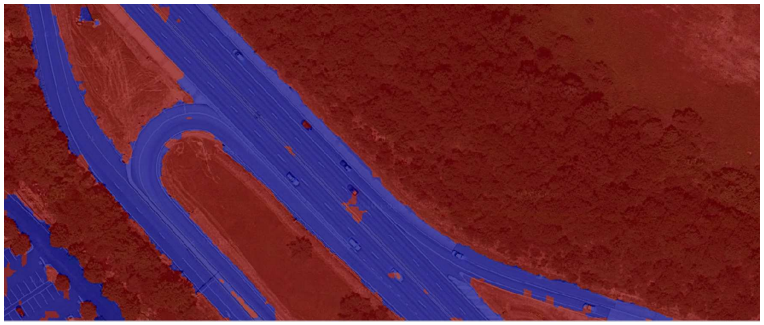
(d) Results of highway map generation.

Figure C.43: Test highway orthoimage 43.

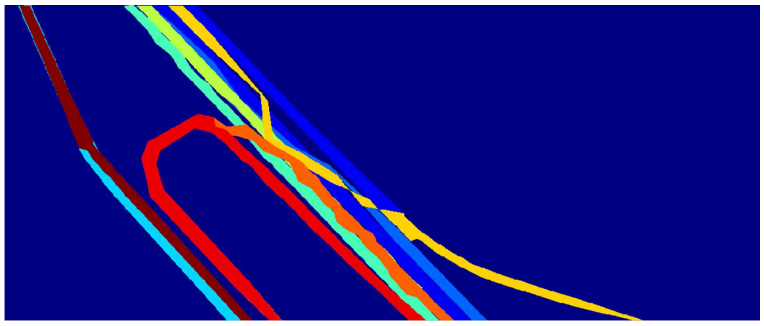
(a) Results of overpass detection.



(b) Results of driving direction estimation.

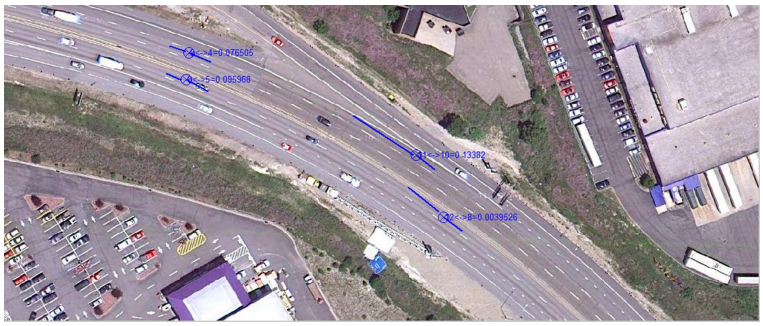


(c) Results of road-region segmentation.

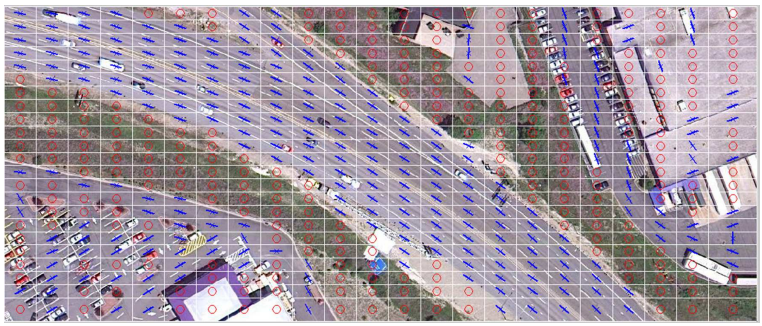


(d) Results of highway map generation.

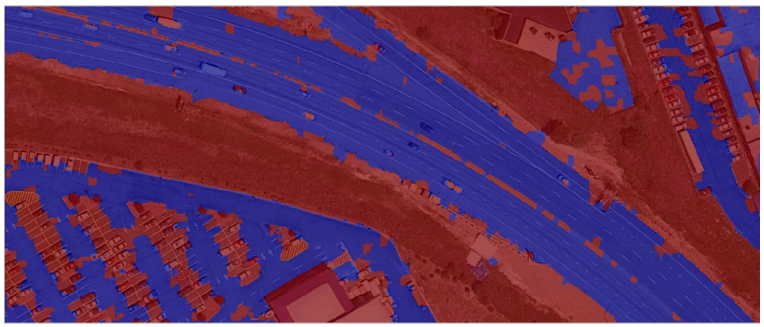
Figure C.44: Test highway orthoimage 44.



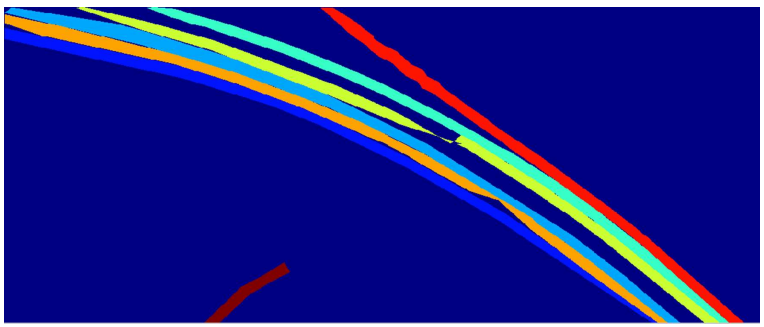
(a) Results of overpass detection.



(b) Results of driving direction estimation.



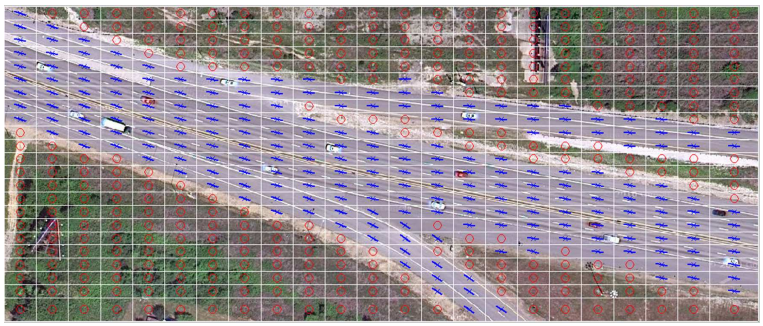
(c) Results of road-region segmentation.



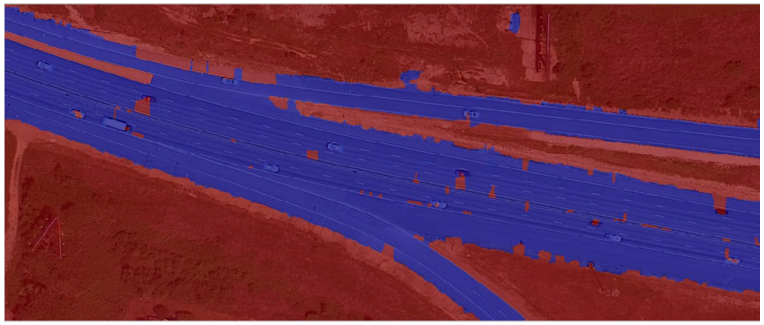
(d) Results of highway map generation.

Figure C.45: Test highway orthoimage 45.

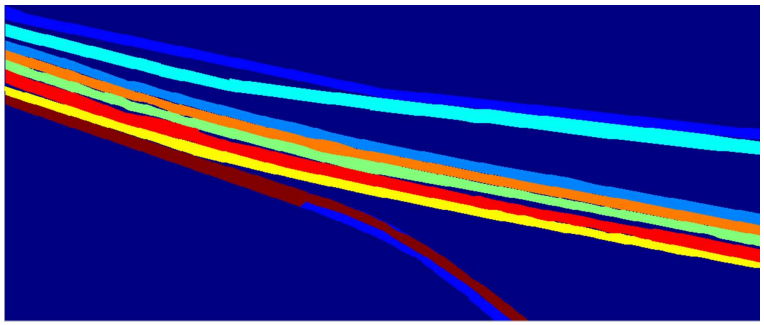
(a) Results of overpass detection.



(b) Results of driving direction estimation.

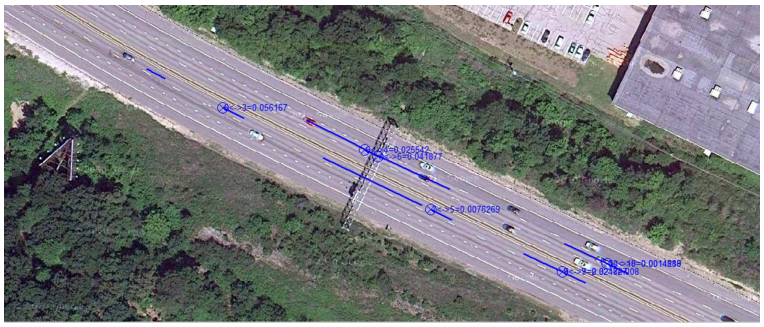


(c) Results of road-region segmentation.

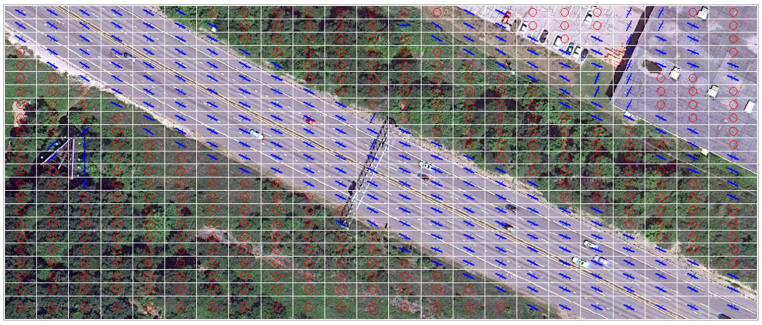


(d) Results of highway map generation.

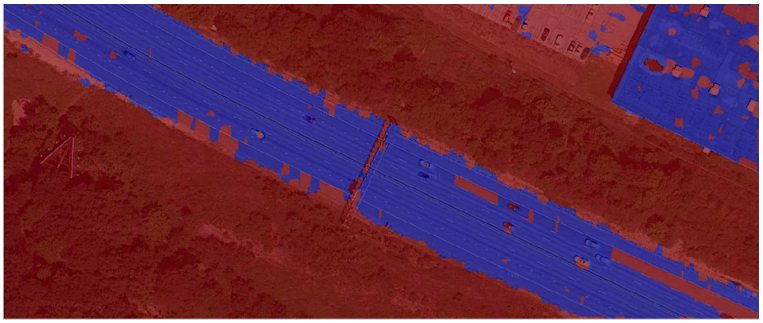
Figure C.46: Test highway orthoimage 46.



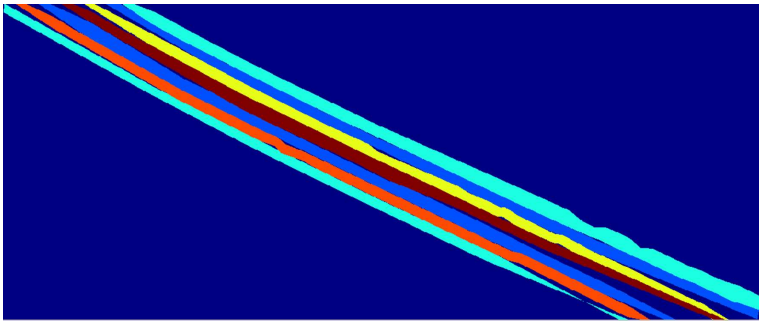
(a) Results of overpass detection.



(b) Results of driving direction estimation.

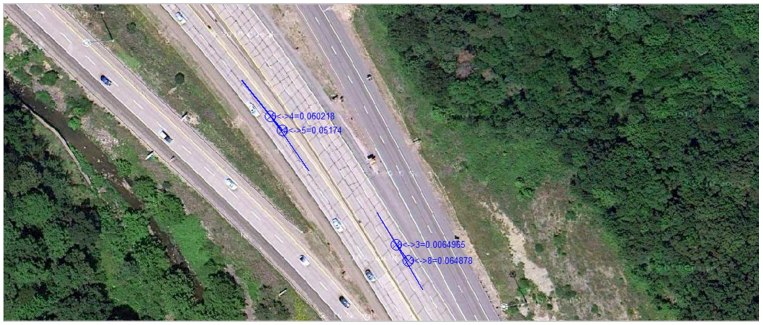


(c) Results of road-region segmentation.

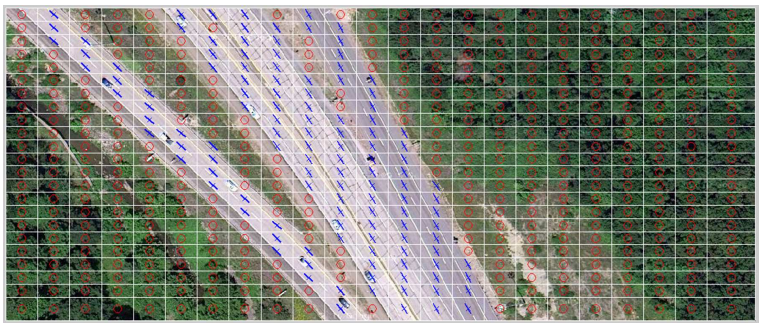


(d) Results of highway map generation.

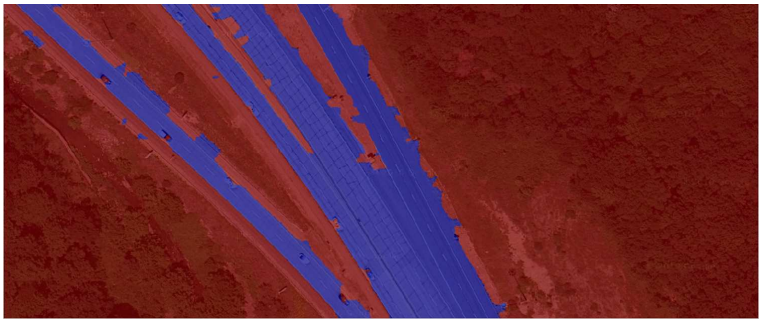
Figure C.47: Test highway orthoimage 47.



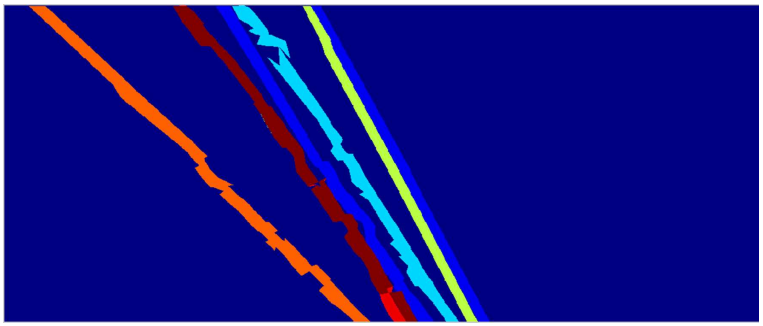
(a) Results of overpass detection.



(b) Results of driving direction estimation.

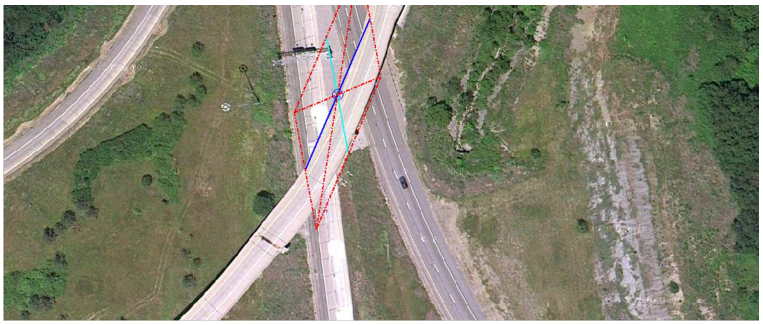


(c) Results of road-region segmentation.

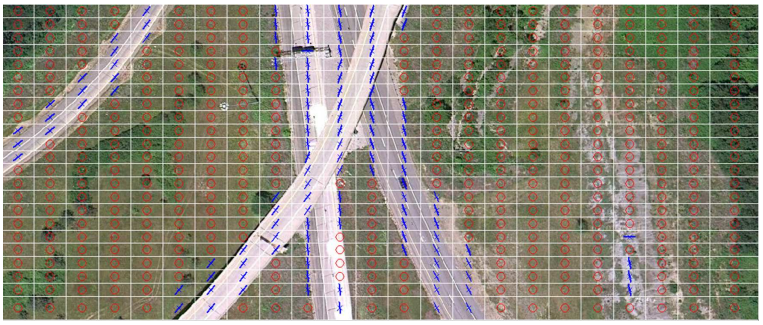


(d) Results of highway map generation.

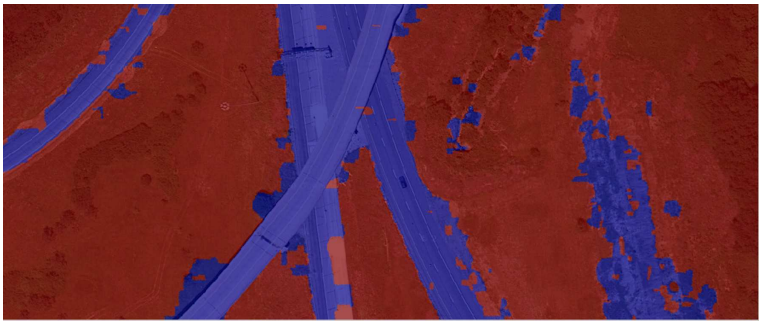
Figure C.48: Test highway orthoimage 48.



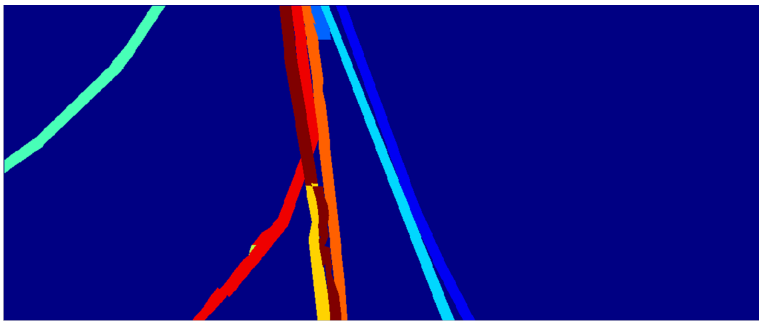
(a) Results of overpass detection.



(b) Results of driving direction estimation.

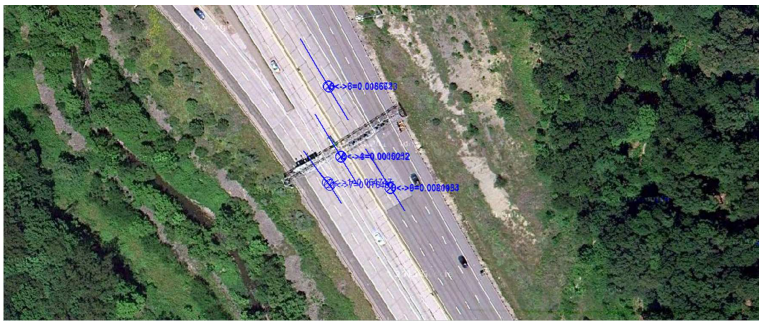


(c) Results of road-region segmentation.

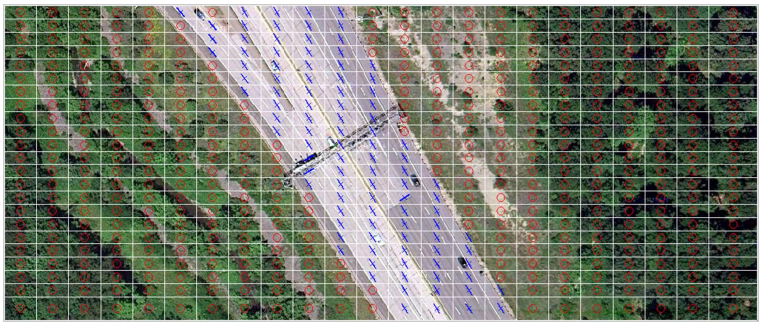


(d) Results of highway map generation.

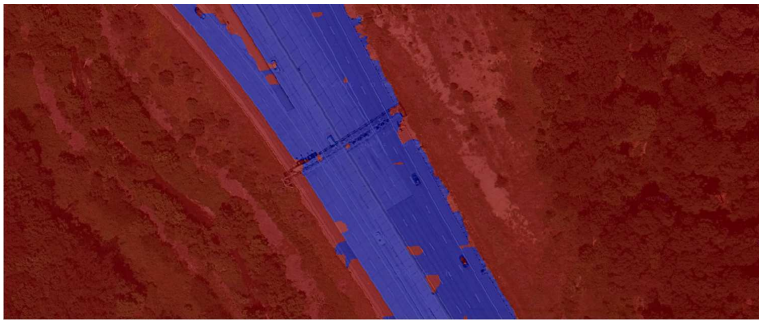
Figure C.49: Test highway orthoimage 49.



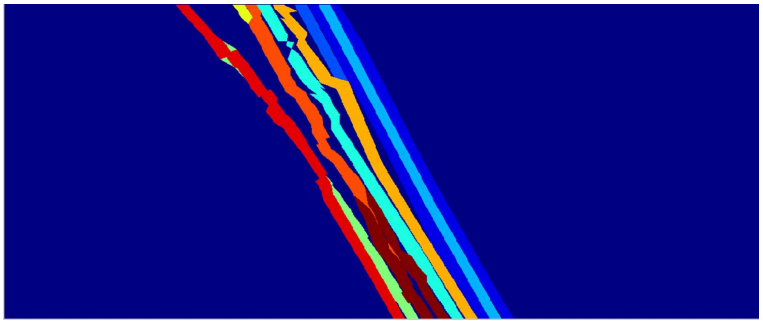
(a) Results of overpass detection.



(b) Results of driving direction estimation.



(c) Results of road-region segmentation.



(d) Results of highway map generation.

Figure C.50: Test highway orthoimage 50.

Bibliography

[Bacha et al., 2008] Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong, D., Wicks, A., Alberi, T., Anderson, D., Cacciola, S., Currier, P., Dalton, A., Farmer, J., Hurdus, J., Kimmel, S., King, P., Taylor, A., Covern, D. V., and Webster, M. (2008). Odin: Team victortango's entry in the darpa urban challenge. *Journal of Field Robotics*, 25(8):467–492. 2.1.2

[Bahlmann et al., 2005] Bahlmann, C., Zhu, Y., Ramesh, V., Pellkofer, M., and Koehler, T. (2005). A system for traffic sign detection, tracking, and recognition using color, shape, motion information. In *Proceedings of IEEE Symposiums on Intelligent Vehicles*, pages 255–260. 2.4.1

[Baltsavias and Zhang, 2005] Baltsavias, E. and Zhang, C. (2005). Automated updating of road databases from aerial imagery. *International Journal of Applied Earth Observation and Geoinformation*, 6:199–213. 2.3.1

[Barnes et al., 2008] Barnes, N., Zelinksky, A., , and Fletcher, L. S. (2008). Real-time speed sign detection using radial symmetry detector. *IEEE Transactions on Intelligent Transportation Systems*, 9(2):322–332. 2.4.1, 4.1.1, 4.1.1

[Baro et al., 2009] Baro, X., Escalera, S., Vitria, J., Pujol, O., and Radeva, P. (2009). Traffic sign recognition using evolutionary adaboost detection and forest-ecoc classification. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):113–126. 2.4.1

[Belongie et al., 2002] Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape context. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24):509–522. 4.1.2

[Bishop, 2006] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. 4.1.1, 5.1.2

[Bohren et al., 2008] Bohren, J., Foote, T., Keller, J., Kushleyev, A., Lee, D., Stewart, A., Vernaza, P., Derenick, J., Spletzer, J., and Satterfield, B. (2008). Little ben: The ben franklin racing team's entry in the 2007 darpa urban challenge. *Journal of Field Robotics*, 25(9):598–614. 2.1.2

[Breiman, 2001] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32. 2.3.3

[Brooks and Iagnemma, 2007] Brooks, C. A. and Iagnemma, K. D. (2007). Self-supervised classification for planetary rover terrain sensing. In *IEEE Conference on Aerospace*, pages 1–9. 2.3.4

[Carbonetto et al., 2004] Carbonetto, P., Freitas, O. D., and Barnard, K. (2004). A statistical model for general contextual object recognition. In *Proceedings of European Conference on Computer Vision*, pages 350–362. 2.3.3

[Carle and Barfoot, 2010] Carle, P. J. and Barfoot, T. D. (2010). Global rover localization by matching lidar and orbital 3d maps. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 881–886. 2.3.2

[Chapelle et al., 2006] Chapelle, O., Scholkopf, B., and Zien, A. (2006). *Semi-supervised Learning*. MIT Press. 2.5

[Chen et al., 2006a] Chen, C.-C., Knoblock, C. A., and Shahabi, C. (2006a). Automatically conflating road vector data with orthoimagery. *GeoInformation*, 10:495–530. 2.3.1

[Chen et al., 2006b] Chen, Y., Wang, R., and Qian, J. (2006b). Extracting contour lines from common-conditioned topographic maps. *IEEE Transactions on Geoscience and Remote Sensing*, 44(4):1048–1057. 2.3.1

[Chiang and Knoblock, 2008] Chiang, Y.-Y. and Knoblock, C. A. (2008). Automatic extraction of road intersection position, connectivity and orientations from raster maps. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2.3.1

[Dahlkamp et al., 2006] Dahlkamp, H., Kaehler, A., Stavens, D., Thrun, S., and Bradski, G. (2006). Self-supervised monocular road detection in desert terrain. In *Proceedings of Robotics: Science and Systems*. 2.3.4

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of Computer Vision and Pattern Recognition*, pages 886–893. 2.3.3, 5.1.2

[DARPA, 2007] DARPA (2007). Route network definition file (rndf) and mission data file (mdf) formats. Technical report, Defense Advanced Research Projects Agency (DARPA). `http://www.darpa.mil/grandchallenge/docs/RNDF_MDF_Formats_031407.pdf`. 2.1.2

[de la Escalera et al., 2004] de la Escalera, A., Armingol, J. M., Pastor, J. M., and Rodriguez, F. J. (2004). Visual sign information extraction and identification by deformable models for intelligent vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 5(2):57–68. 2.4.1

[Diebel and Thrun, 2005] Diebel, J. and Thrun, S. (2005). An application of makov random fields to range sensing. In *Proceedings of Neural Information Processing Systems*, pages 291–298. 2.3.3

[Dogruer et al., 2008] Dogruer, C., Koku, B., and Dolen, M. (2008). Global urban localization of outdoor mobile robots using satellite images. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3927–3932. 2.3.2

[Dolgov and Thrun, 2008] Dolgov, D. and Thrun, S. (2008). Detection of principal directions in unknown environments for autonomous navigation. In *Proceedings of the Robotics: Science and Systems*. 3.1.2

[Dolgov and Thrun, 2009] Dolgov, D. and Thrun, S. (2009). Autonomous driving in semi-structured environments: Mapping and planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 3407–3414. 2.1.1, 2.5, 5

[Dollar et al., 2006] Dollar, P., Tu, Z., and Belongie, S. (2006). Supervised learning of edges and object boundaries. In *Proceedings of Computer Vision and Pattern Recognition*, pages 1964–1971. 2.3.3

[Doucette et al., 2004] Doucette, P., Agouris, P., and Stefanidis, A. (2004). Automated road extraction from high resolution multispectral imagery. *Photogrammetric Engineering and Remote Sensing*, 70(12):1405–1416. 2.3.1

[Duda et al., 2001] Duda, R. O., Hart, P. E., and Stork, D. G. (2001). *Pattern Classification*. Wiley-Interscience. 2.3.3

[Durrant-Whyte and Bailey, 2006] Durrant-Whyte, H. and Bailey, T. (2006). Simultaneous localization and mapping: Part i the essential algorithms. *Robotics and Automation Magazine*, 13:99–110. 2.1.1

[Eichner and Breckon, 2008] Eichner, M. L. and Breckon, T. P. (2008). Integrated speed limit detection and recognition from real-time video. In *Proceedings of IEEE Symposiums on Intelligent Vehicles*, pages 626–631. 2.4.1

[Elder and Zucker, 1996] Elder, J. H. and Zucker, S. W. (1996). Computing contour closure. In *Proceedings of European Conference on Computer Vision*, pages 399–412. 2.3.3

[Eppstein and Goodrich, 2008] Eppstein, D. and Goodrich, M. T. (2008). Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2.3.1

[Estrada and Jepson, 2006] Estrada, F. J. and Jepson, A. D. (2006). Robust boundary detection with adaptive grouping. In *Proceedings of Conference on Computer Vision and Pattern Recognition Workshop*, pages 184–191. 2.3.3

[Fabian, 2008] Fabian, T. (2008). An algorithm for parking lot occupation detection. In *Proceedings of IEEE Computer Information Systems and Industrial Management Applications*. 2.3.3

[Fairfield and Urmson, 2011] Fairfield, N. and Urmson, C. (2011). Traffic light mapping and detection. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 5421–5426. 2.1.2

[Felzenszwalb and McAllester, 2006] Felzenszwalb, P. and McAllester, D. (2006). A min-cover approach for finding salient curves. In *Proceedings of IEEE Workshop on Perceptual Organization in Computer Vision*. 3.2.2, 3.2.2

[Felzenszwalb et al., 2008] Felzenszwalb, P., McAllester, D., and Ramanan, D. (2008). A discriminatively trained, multiscale, deformable part model. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2.3.3, 2.5

[Felzenszwalb and Huttenlocher, 2005] Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1):55–79. 2.3.3

[Ferguson et al., 2008a] Ferguson, D., Howard, T. M., and Likhachev, M. (2008a). Motion planning in urban environments: Part i. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1063–1069. 2.1.2

[Ferguson et al., 2008b] Ferguson, D., Howard, T. M., and Likhachev, M. (2008b). Motion planning in urban environments: Part ii. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1070–1076. 2.1.2

[Fischler and Bolles, 1981] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381395. 2.3.2

[flavie Auclair Fortier et al., 2000] flavie Auclair Fortier, M., Ziou, D., Armenakis, C., and Wang, S. (2000). Automated updating of road information from aerial images. In *Proceedings of American Society Photogrammetry and Remote Sensing*, pages 16–23. 2.3.1

[Frankel et al., 2010] Frankel, R., Gudmundsson, O., Miller, B., Potter, J., Sullivan, T., Syed, S., Hoang, D., min John, J., Liao, K.-S., Nahass, P., Schwab, A., Yuan, J., Stavens, D., Plagemann, C., and Thrun, S. (2010). Assisted highway lane changing with rascl. In *Proceedings of AAAI Spring Symposium*. 2.1.2

[Freeman et al., 2000] Freeman, W. T., Pasztor, E. C., and Carmichael, O. T. (2000). Learning low-level vision. *International Journal of Computer Vision*, 40(1):25–47. 2.3.3

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of International Conference on Machine Learning*, pages 148–156. 4.1.1

[Freund and Shapire, 1997] Freund, Y. and Shapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139. 5.4.1

[Friedman et al., 2000] Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–407. 4.1.1

[Gallagher and Chen, 2007] Gallagher, A. C. and Chen, T. (2007). Using a markov network to recognize people in consumer images. In *Proceedings of International Conference on Image Processing*, pages 489–492. 2.3.3

[Geman and Jedynak, 1996] Geman, D. and Jedynak, B. (1996). An active testing model for tracking roads in satellite images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(1). 2.3.1

[Geraud and Mouret, 2004] Geraud, T. and Mouret, J.-B. (2004). Fast road network extraction in satellite images using mathematical morphology and markov random fields. *Journal on Applied Signal Processing*, 16:1–12. 2.3.3

[Grote et al., 2007] Grote, A., Butenuth, M., and Heipke, C. (2007). Road extraction in suburban areas based on normalized cuts. In *Proceedings of International Archieves of Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 51–56. 2.3.1

[Gruen and Li, 1995] Gruen, A. and Li, H. (1995). Road extraction from aerial and satellite images by dynamic programming. *ISPRS Journal of Photogrammetry and Remote Sensing*,

50(4):11–20. 2.3.1

[Hartley and Zisserman, 2003] Hartley, R. and Zisserman, A. (2003). *Multiple View Geometry in Computer Vision*. Cambridge University Press. 4

[Haverkamp, 2002] Haverkamp, D. (2002). Extracting straight road structure in urban environments using ikonos satellite imagery. *Optical Engineering*, 41(9):2107–2110. 2.3.1

[Hebert, 1989] Hebert, M. (1989). Building and navigating maps of road scenes using an active sensor. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1136–1142. 1, 1.1, 2.1.1, 2.5

[Heitz and Koller, 2008] Heitz, G. and Koller, D. (2008). Learning spatial context: Using stuff to find things. In *Proceedings of European Conference on Computer Vision*. 2.3.3, 2.5

[Hu et al., 2007] Hu, J., Razdan, A., Femiani, J. C., Cui, M., and Wonka, P. (2007). Road network extraction and intersection detection from aerial images by tracking road footprints. *IEEE Transactions on Geoscience and Remote Sensing*, 45(12):4144–4157. 2.3.1

[Huang et al., 2008] Huang, C.-C., Wang, S.-J., Chang, Y.-J., and Chen, T. (2008). A bayesian hierarchical detection framework for parking space detection. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 2097–2100. 2.3.3

[Jordan, 2004] Jordan, M. I. (2004). Graphical models. *Statistical Science Special Issues on Bayesian Statistics*, 19:140–155. 2.3.3

[Kahn et al., 1990] Kahn, P., Kitchen, L., and Riseman, E. (1990). A fast line finder for vision-guided robot navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(11):1098–1102. 3.1.1, 5.1.1

[Katz et al., 2008] Katz, R., Douillard, B., Nieto, J., and Nebot, E. (2008). A self-supervised architecture for moving obstacles classification. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 155–160. 2.3.4

[Khotanzad and Zink, 2003] Khotanzad, A. and Zink, E. (2003). Contour line and geographic feature extraction from usgs color topological paper maps. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(1):18–31. 2.3.1

[Kim et al., 2006] Kim, D., Sun, J., Oh, S. M., Rehg, J. M., and Bobick, A. F. (2006). Traversability classification using unsupervised on-line visual learning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 518–525. 2.3.4

[Kimmel et al., 2003] Kimmel, R., Brostein, M., and Brostein, A. (2003). *Numerical Geometry of Images: Theory, Algorithms, and Applications*. Springer. 5.3.2

[Kluckner et al., 2009] Kluckner, S., Mauthner, T., Roth, P. M., and Bischof, H. (2009). Semantic classification in aerial imagery by integrating appearance and height information. In *Proceedings of Asian Conference on Computer Vision*, pages 477–488. 2.3.3

[Koller et al., 2007] Koller, D., Friedman, N., Getoor, L., and Taskar, B. (2007). *Introduction to Statistical Relational Learning*, chapter 2: Graphical Models in a Nutshell. MIT Press. 2.3.3

[Kumar and Hebert, 2005] Kumar, S. and Hebert, M. (2005). A hierarchical field framework for

unified context-based classification. In *Proceedings of International Conference on Computer Vision*, pages 1284–1291. 2.3.3

[Kummerle et al., 2009] Kummerle, R., Hahnel, D., Dolgov, D., Thrun, S., and Burgard, W. (2009). Autonomous driving in a multi-level parking structure. In *Proceedings of International Conference on Robotics and Automation*, pages 3395–3400. 2.1.1, 2.5, 5

[Lacoste et al., 2005] Lacoste, C., Descombes, X., and Zerubia, J. (2005). Point processes for unsupervised line network extraction in remote sensing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10). 2.3.3

[Lalonde et al., 2010] Lalonde, J.-F., Efros, A. A., and Narasimhan, S. G. (2010). Detecting ground shadows in outdoor consumer photographs. In *Proceedings of European Conference on Computer Vision*, pages 322–335. 3.1.1

[Leonard et al., 2008] Leonard, J., How, J., Teller, S., Berger, M., Campbell, S., Fiore, G., Fletcher, L., Frazzoli, E., Huang, A., Karaman, S., Koch, O., Kuwata, Y., Moore, D., Olson, E., Peters, S., Teo, J., Truax, R., Walter, M., Barrett, D., Epstein, A., Maheloni, K., Moyer, K., Jones, T., Buckley, R., Antone, M., Galejs, R., Krishnamurthy, S., and Williams, J. (2008). A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25:727–774. 2.1.2

[Leung and Malik, 2001] Leung, T. and Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44. 3.3.1

[Lewis and Catlett, 1994] Lewis, D. D. and Catlett, J. (1994). Heterogeneous uncertainty sampling for supervised learning. In *Proceedings of the International Conference on Machine Learning*, pages 148–156. 5.4.1

[Li et al., 2003] Li, Q., Zheng, N., and Cheng, H. (2003). An adaptive approach to lane markings detection. In *Proceedings of Intelligent Transportation Systems*, pages 510–514. 2.3.3

[Li, 2000] Li, S. Z. (2000). *Markov Random Fields Modeling in Computer Vision*. Springer-Verlag. 2.3.3, 5.1.2

[Lieb et al., 2005] Lieb, D., Lookingbill, A., and Thrun, S. (2005). Adaptive road following using self-supervised learning and reverse optical flow. In *Proceedings of Robotics Science and Systems*. 2.3.4

[Lipski et al., 2008] Lipski, C., Scholz, B., Berger, K., Linz, C., Stich, T., and Magnor, M. (2008). A fast and robust approach to lane marking detection and lane tracking. In *IEEE Symposium on Image Analysis and Interpretation*, pages 57–60. 2.3.3

[Loy and Barnes, 2004] Loy, G. and Barnes, N. (2004). Fast shape-based road sign detection for a driver assistance system. In *Proceedings of IEEE Intelligent Robots and Systems*, pages 70–75. 2.4.1

[Ma et al., 2009] Ma, Y., Chowdhury, M., Sadek, A., and Jeihani, M. (2009). Real-time highway traffic condition assessment framework using vehicle-infrastructure integration (vii) with artificial intelligence. *IEEE Transactions on Intelligent Transportation Systems*, 10(4):615–627. 2.4.2, 2.5

[Malik et al., 2001] Malik, J., Belongie, S., Leung, T., and Shi, J. (2001). Contour and texture

analysis for image segmentation. *International Journal of Computer Vision*, 43(1):7–27. 2.3.3

[Marr, 1982] Marr, D. (1982). *Vision*. Freeman. 3

[Martin et al., 2004] Martin, D. R., Fowlkes, C. C., and Malik, J. (2004). Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(1). 2.3.3, 3.1.2, 3.3.2

[Miller et al., 2008] Miller, I., Campbell, M., Huttenlocher, D., Kline, F.-R., Nathan, A., Lupashin, S., Catlin, J., Schimpf, B., Moran, P., Zych, N., Garcia, E., Kurdziel, M., and Fujishima, H. (2008). Team cornell's skynet: Robust perception and planning in an urban environment. *Journal of Field Robotics*, 25(8):493–527. 2.1.2

[Mnih and Hinton, 2010] Mnih, V. and Hinton, G. E. (2010). Learning to detect roads in high-resolution aerial images. In *Proceedings of European Conference on Computer Vision*, pages 210–223. 2.3.3

[Montemerlo et al., 2008] Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger, S., Haehnel, D., Hilden, T., Hoffmann, G., Huhnke, B., Johnston, D., Klumpp, S., Langer, D., Levandowski, A., Levinson, J., Orenstein, D., Paefgen, J., Penny, I., Petrovskaya, A., Pflueger, M., Stanek, G., Stavens, D., Vogt, A., and Thrun, S. (2008). Junior: The stanford entry in the urban challenge. *Journal of Field Robotics*, 25(9):569–597. 1.1, 2.1.2

[Movaghati and Moghaddamjoo, 2008] Movaghati, S. and Moghaddamjoo, A. (2008). Using unscented kalman filter for road tracing from satellite images. In *Proceedings of Asia International Conference on Modelling and Simulation*, pages 379–384. 2.3.3, 3.2.2

[Nabbe et al., 2004] Nabbe, B., Kumar, S., and Hebert, M. (2004). Path planning with hallucinated worlds. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3123–3130. 2.3.3

[Nair and Clark, 2004] Nair, V. and Clark, J. J. (2004). An unsupervised, online learning framework for moving object detection. In *Proceedings of Computer Vision and Pattern Recognition*, pages 317–324. 2.3.4, 2.5

[Niu and Liu, 2011] Niu, S. and Liu, H. (2011). Probabilistic neural networks for the identification of traffic state. In *Proceedings of International IEEE Conference on Intelligent Transportation Systems*, pages 754–759. 2.4.2

[Ojala et al., 2002] Ojala, T., Pietikainen, M., and Maenpaa, T. (2002). Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987. 2.3.3, 3.1.2

[Oliva and Torralba, 2007] Oliva, A. and Torralba, A. (2007). The role of context in object recognition. *Trends in Cognitive Sciences*, 11(12):520–527. 2.3.3

[Overett and Petersson, 2011] Overett, G. and Petersson, L. (2011). Large scale sign detection using hog feature variants. In *Proceedings of IEEE Intelligent Vehicles Symposium*, pages 326–331. 2.4.1, 2.4.2, 2.5

[Palmer, 1999] Palmer, S. E. (1999). *Vision Science: Photons to Phenomenology*. A Bradford Book. 2.3.3, 3

[Persson et al., 2008] Persson, M., Duckett, T., and Lilienthal, A. (2008). Improved mapping

and image segmentation by using semantic information to link aerial images and ground-level information. In *Recent Progress in Robotics*. 2.3.2

[Platt, 2000] Platt, J. C. (2000). Probabilities for support vector machines. In *Advances in Large Margin Classifiers*, pages 61–73. MIT Press. 5.4.1

[Poggio, 1981] Poggio, T. (1981). Marr's approach to vision. *Trends in NeuroSciences*, 4(10):258–262. 3

[Ponce et al., 2006] Ponce, J., Berg, T., Everingham, M., Forsyth, D., Hebert, M., Lazebnik, S., Marszalek, M., Schmid, C., Russell, C., Torralba, A., Williams, C., Zhang, J., and Zisserman, A. (2006). Dataset issues on object recognition. In *Towards Category-Level Object Recognition*, pages 29–48. Springer. 4.1.1

[Porway et al., 2008] Porway, J., Wang, K., Yao, B., and Zhu, S. C. (2008). A hierarchical and contextual model for aerial image understanding. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. 2.3.3, 2.5

[Qian et al., 2010] Qian, C., Gale, B., and Bach, J. (2010). Earth documentation: Overpass detection using mobile lidar. In *Proceedings of IEEE International Conference on Image Processing*, pages 3901–3904. 2.3.1

[Rosenberg et al., 2005] Rosenberg, C., Hebert, M., and Schneiderman, H. (2005). Semi-supervised self-training of object detection models. In *Proceedings of the 7th IEEE Workshops on Application of Computer Vision*, pages 29–36. 2.3.4, 2.5

[Saragih et al., 2009] Saragih, J. M., Lucey, S., and Cohn, J. F. (2009). Face alignment through subspace constrained mean-shifts. In *Proceedings of IEEE International Conference on Computer Vision*. 2.3.3, 2.5

[Saudi et al., 2008] Saudi, A., Teo, J., H.A.Mohd, H., and Sulaiman, J. (2008). Fase lane detection with randomized hough transform. In *Proceedings on International Symposium on Information Technology*, pages 1–5. 2.3.3

[Saxena et al., 2007] Saxena, A., Chung, S. H., and Ng, A. Y. (2007). 3-d depth reconstruction from a single still image. *International Journal of Computer Vision*. 2.3.3

[Schlosser et al., 2010] Schlosser, J., Montemerlo, M., and Salisbury, K. (2010). Intelligent road sign detection using 3d scene geometry. In *Proceedings of IEEE Intelligent Robots and Systems*, pages 740–745. 2.4.1

[Schpok, 2011] Schpok, J. (2011). Geometric overpass extraction from vector road data and dsms. In *Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2.3.1

[Scrapper et al., 2003] Scrapper, C., Takeuchi, A., Chang, T., Hong, T., and Shneier, M. (2003). Using a priori data for prediction and object recognition in an autonomous mobile vehicle. In *Proceedings of the SPIE Aerosense Conference*. 2.3.2

[Seo et al., 2009a] Seo, Y.-W., Ratliff, N., and Urmson, C. (2009a). Self-supervised aerial image analysis for extracting parking lot structures. In *Proceedings of International Joint Conference on Artificial Intelligence*, pages 1837–1842. 1.1, 2.3.4, 5.4, 5.4.1

[Seo and Urmson, 2008] Seo, Y.-W. and Urmson, C. (2008). A perception mechanism for sup-

porting autonomous intersection handling in urban driving. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1830–1835. 1.1

[Seo and Urmson, 2009a] Seo, Y.-W. and Urmson, C. (2009a). A hierarchical image analysis for extracting parking lot structures from aerial image. Technical Report CMU-RI-TR-09-03, Robotics Institute, Carnegie Mellon Univ. 1

[Seo and Urmson, 2009b] Seo, Y.-W. and Urmson, C. (2009b). Utilizing prior information to enhance self-supervised aerial image analysis for extracting parking lot structures. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 339–344. 5.4.1

[Seo et al., 2009b] Seo, Y.-W., Urmson, C., Wettergreen, D., and Lee, J.-W. (2009b). Augmenting cartographic resources for autonomous driving. In *Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 13–22. 2.3.3, 2.3.4, 5.3.1

[Seo et al., 2010] Seo, Y.-W., Urmson, C., Wettergreen, D., and Lee, J.-W. (2010). Building lane-graphs for autonomous parking. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6052–6057.

[Seo et al., 2011a] Seo, Y.-W., Wettergreen, D., and Zhang, W. (2011a). Recognizing highway workzone for active safety and driver assistant applications. Technical Report Collaborative Report CL-11-34-ECI, GM Research and Development Center. 4

[Seo et al., 2011b] Seo, Y.-W., Wettergreen, D., and Zhang, W. (2011b). Understanding temporary changes of driving conditions for active safety and driver assistant applications. Technical Report Collaborative Report CL-11-40-ECI, GM Research and Development Center. 4

[Silver et al., 2006] Silver, D., Sofman, B., Vandapel, N., Bagnell, J. A., and Stentz, A. (2006). Experimental analysis of overhead data processing to support long range navigation. In *Proceedings of IEEE/IRJ International Conference on Intelligent Robots and Systems*, pages 2443–2450. 2.3.2

[Singhal et al., 2003] Singhal, A., Luo, J., and Zhu, W. (2003). Probabilistic spatial context models for scene content understanding. In *Proceedings on Computer Vision and Pattern Recognition*, pages 235–241. 2.3.3

[Sofman et al., 2006] Sofman, B., Lin, E., Bagnell, J. A., Vandapel, N., and Stentz, A. (2006). Improving robot navigation through self-supervised online learing. In *Proceedings of Robotics Science and Systems*. 2.3.2, 2.3.4, 2.5

[Soleimani et al., 2010] Soleimani, B., Ashtiani, M.-H. Z., Soleimani, B. H., and Moradi, H. (2010). A disaster invariant feature for localization. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1096–1101. 2.3.2

[Stahl and Wang, 2006] Stahl, J. S. and Wang, S. (2006). Globally optimal grouping for symmetric boundaries. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1030–1037. 2.3.3

[Stavens and Thrun, 2006] Stavens, D. and Thrun, S. (2006). A self-supervised terrain roughness estimator for off-road autonomous driving. In *Proceedings of Conference in Uncertainty*

in Artificial Intelligence. 2.3.4

[Sun et al., 2006] Sun, T.-Y., Tsai, S.-J., and Chan, V. (2006). Hsi color model based lane-marking detection. In *Proceedings of the IEEE Intelligent Transportation Systems Conference*, pages 1168–1172. 2.3.3

[Swain and Ballard, 1991] Swain, M. J. and Ballard, D. H. (1991). Color indexing. *International Journal of Computer Vision*, 7(1):11–32. 5.4.1

[Tabibiazar and Basir, 2011] Tabibiazar, A. and Basir, O. (2011). Kernel-based modeling and optimization for density estimation in transportation systems using floating car data. In *Proceedings of International IEEE Conference on Intelligent Transportation Systems*, pages 576–581. 2.4.2

[Thompson et al., 2005] Thompson, D., Niekum, S., Smith, T., and Wettergreen, D. (2005). Automatic detection and classification of features of geologic interest. In *Proceedings of IEEE Conference on Aerospace*. 2.3.4

[Thorpe et al., 1991] Thorpe, C., Hebert, M., Kanade, T., and Shafer, S. (1991). Toward autonomous driving: The cmu navlab. part ii: System and architecture. *IEEE Expert*, 6(4):44–52. 1.1, 2.1.1, 2.5

[Thrun, 1996] Thrun, S. (1996). Is learning the nth thing any easier than learning the first? In *Advances in Neural Information Processing Systems*, pages 640–646. 2.3.4

[Thrun et al., 2005] Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic Robotics*. MIT Press. 1.1, 2.1.1

[Thrun and Mitchell, 1995] Thrun, S. and Mitchell, T. M. (1995). Lifelong robot learning. *Robotics and Autonomous Systems*, 15:25–46. 2.3.4

[Thrun and Montemerlo, 2005] Thrun, S. and Montemerlo, M. (2005). The graphslam algorithm with applications to large-scale mapping of urban structures. *International Journal of Robotics Research*, 25(5-6). 2.1.1

[Timofte et al., 2009] Timofte, R., Zimmermann, K., and Gool, L. V. (2009). Multi-view traffic sign detection, recognition, and 3d localisation. In *Proceedings of Workshop on Applications of Computer Vision*, pages 1–8. 2.4.1, 2.4.2, 2.5

[Tomasi and Manduchi, 1998] Tomasi, C. and Manduchi, R. (1998). Bilateral filtering for gray and color images. In *Proceedings of International Conference on Computer Vision*, pages 839–846. 3.1.1

[Tonjes and Growe, 1998] Tonjes, R. and Growe, S. (1998). Knowledge based road extraction from multisensor imagery. *International Archives of Photogrammetry and Remote Sensing*, 32(3/1):387–393. 2

[Turk and Pentland, 1991] Turk, M. and Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86. 2.3.3, 5.1.2

[Urmson et al., 2008] Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M. N., Dolan, J., Duggins, D., Galatali, T., Geyer, C., Gittleman, M., Harbaugh, S., Hebert, M., Howard, T. M., Kolski, S., Kelly, A., Likhachev, M., McNaughton, M., Miller, N., Peterson, K., Pilnick, B., Rajkumar, R., Rybski, P., Salesky, B., Seo, Y.-W., Singh, S., Snider, J., Stentz,

A., Whittaker, W. R., Wolkowicki, Z., Ziglar, J., Bae, H., Brown, T., Demitrish, D., Litkouhi, B., Nickolaou, J., Sadekar, V., Zhang, W., Struble, J., Taylor, M., Darms, M., and Ferguson, D. (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics: Special Issues on the 2007 DARPA Urban Challenge*, pages 425–466. 1, 1.1, 2.1.2, 2.3.2

[Urmson et al., 2009] Urmson, C., Baker, C., Dolan, J., Rybski, P., Salesky, B., Whittaker, W. R., Ferguson, D., and Darms, M. (2009). Autonomous driving in traffic: Boss and the urban challenge. *AI Magazine*, 30(2):17–28. 2, 1

[U.S. Department of Transportation, 2000] U.S. Department of Transportation, F. H. A. (2000). *FHWA Functional Classification Guidelines*. http://www.fhwa.dot.gov/planning/fctoc.htm. 3

[U.S. Department of Transportation, 2007] U.S. Department of Transportation, F. H. A. (2007). *Mitigation Strategies for Design Exceptions*. http://safety.fhwa.dot.gov/geometric/pubs/mitigationstrategies/index.htm. 3.2.1

[U.S. Department of Transportation, 2009] U.S. Department of Transportation, F. H. A. (2009). *Manual on uniform traffic control devices for streets and highways*. http://mutcd.fhwa.dot.gov/. 1, 4, 4.1, 4.3.1

[Vandapel et al., 2003] Vandapel, N., Donamukkala, R., and Hebert, M. (2003). Experimental results in using aerial ladar data for mobile robot navigation. In *Proceedings of International Conference on Field and Service Robotics*, pages 103–112. 2.3.2

[Vapnik, 1995] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer-Verlag. 5.1.2

[Vazirani, 2004] Vazirani, V. V. (2004). *Approximation Algorithms*. Springer. 3

[Verbeek and Triggs, 2007] Verbeek, J. and Triggs, B. (2007). Scene segmentation with conditional random fields learned from partially labeled images. In *Proceedings of Neural Information Processing Systems*. 2.3.3

[Viola and Jones, 2004] Viola, P. and Jones, M. (2004). Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154. 2.3.3, 2.4.1, 2.4.2, 5.4.1

[Voisin et al., 2005] Voisin, V., Avila, M., Emile, B., Begot, S., and Barget, J.-C. (2005). Road markings detection and tracking using hough transform and kalman filter. In *Proceedings of Conference on Advanced Concepts for Intelligent Vision Systems*, pages 76–83. 2.3.3

[Wang and Zimmermann, 2008] Wang, H. and Zimmermann, R. (2008). Snapshot location-based query processing on moving objects in road networks. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2.3.1

[Wang and Hanson, 1998] Wang, X. and Hanson, A. R. (1998). Parking lot analysis and visualization from aerial images. In *Proceedings of the IEEE Workshop on Applications of Computer Vision*, pages 36–41. 2.3.3, 2.5

[Wang et al., 2000] Wang, Y., Shen, D., and Teoh, E. K. (2000). Lane detection using spline model. *Pattern Recognition Letters*, 21(9):677–689. 2.3.3

[Wang et al., 2004] Wang, Y., Teoh, E. K., and Shen, D. (2004). Lane detection and tracking

using b-snake. *Image and Vision Computing*, 22:269–280. 2.3.3

[Warren and Warren, 1968] Warren, R. M. and Warren, R. P. (1968). *Helmholtz on Perception: Its Physiology and Development*. John Wiley and Sons, Inc. 1

[Weinman et al., 2004] Weinman, J., Hanson, A., and McCallum, A. (2004). Sign detection in natural images with conditional random fields. In *Proceedings of International Workshop on Machine Learning for Signal Processing*, pages 548–558. 2.3.3

[Weiss, 1999] Weiss, Y. (1999). Segmentation using eigenvectors: A unifying view. In *Proceedings of International Conference on Computer Vision*, pages 975–982. 5.4

[Wu et al., 2007] Wu, Q., Huang, C., yu Wang, S., chen Chiu, W., and Chen, T. (2007). Robust parking space detection considering inter-space correlation. In *Proceedings of IEEE International Conference on Multimedia and Expo*, pages 659–662. 2.3.3

[Yedidia et al., 2002] Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2002). Understanding belief propagation and its generalizations. Technical Report TR2001-022, Mitsubishi Electric Research Laboratories. 5.1.2

[Yedidia et al., 2003] Yedidia, J. S., Freeman, W. T., and Weiss, Y. (2003). *Exploring Artificial Intelligence in the New Millennium*, chapter 8: Understanding belief propagation and its generalizations. Elsevier Science. 2.3.3

[Yu and Jain, 1997] Yu, B. and Jain, A. K. (1997). Lane boundary detection using a multiresolution hough transform. In *Proceedings of International Conference on Image Processing*, pages 26–29. 2.3.3

[Yuille et al., 1998] Yuille, A., Snow, D., and Nitzberg, M. (1998). Signfinder: Using color to detect, localize and identify informational signs. In *Proceedings of International Conference on Computer Vision*, pages 628–633. 2.4.1

[Zhang, 2006] Zhang, Q. (2006). *Automated road network extraction from high spatial resolution multi-spectral imagery*. PhD thesis, University of Calgary. 2.3.1

[Zheng et al., 2007] Zheng, S., Tu, Z., and Yuille, A. (2007). Detecting object boundaries using low-, mid-, and high-level information. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. 2.3.3

[Zhou et al., 2006] Zhou, J., Bischof, W. F., and Caelli, T. (2006). Road tracking in aerial images based on human-computer interaction and bayesian filtering. *Journal of Photogrammetry and Remote Sensing*, 61(2). 2.3.3, 3.2.2

[Zhu and Mordohai, 2009] Zhu, Q. and Mordohai, P. (2009). A minimum cover approach for extracting the road network from airborne lidar data. In *Proceedings of ICCV Workshop on 3D Digital Imaging and Modeling*, pages 1582–1589. 2.3.1, 3.2.2, 3.2.2