VISUAL LOCALIZATION, SEMANTIC VIDEO SEGMENTATION AND LABELING USING SATELLITE MAPS

by

TURGAY SENLET

A Dissertation submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Computer Science

written under the direction of

Prof. Ahmed Elgammal

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

May, 2015

ABSTRACT OF THE DISSERTATION

Visual Localization, Semantic Video Segmentation and Labeling Using Satellite Maps

By TURGAY SENLET

Dissertation Director:

Prof. Ahmed Elgammal

In this dissertation, I propose vision-based geo-localization and segmentation methods that make use of semantic and appearance information from satellite images. First, I present a framework for vision-based localization of moving platforms by registering perspective camera images to satellite maps and by employing particle filter tracking techniques. I present different versions of the localization framework for stereo and monocular imagery. Second, I propose a novel computer vision method that uses semantic elements for efficient localization of a given aerial image in a large search area. In this method, I use buildings on the aerial image as semantic elements and make use of building neighborhood structures to obtain accurate localization results, efficiently. For this problem, I perform tests on a very large city building dataset with 300K buildings. Third, I propose a novel framework for semantic segmentation and labeling of videos that

propagates semantic information from satellite maps on to globally localized video frames. This method generates accurate labeling of semantic elements without performing any prior learning on the video itself. Finally, in order to understand and extract semantic information from satellite images, I investigate algorithms for semantically labeling satellite images; mainly focusing on labeling buildings, roads, sidewalks, and crosswalks from satellite images. I propose novel techniques to estimate sidewalk paths occluded by trees on satellite images.

DEDICATION

To my precious wife

Selen Esmeray-Senlet

ACKNOWLEDGEMENTS

I want to thank my advisor, Prof. Ahmed Elgammal, for his guidance and support all throughout my Ph.D. years and for always encouraging me to take the less-travelled road in my research topics. With his advice and ideas, I was able to shape my thoughts into my research and finally into useful outcomes.

I would like to thank my internal committee members Prof. Dimitris Metaxas, Prof. Kostas Bekris and my committee member Dr. Abhijit Ogale from Google Self-Driving Car team for their ideas, discussions, and contributions. I would also like to thank them for their comments and improvements on my dissertation.

I would like to thank the graduate students of Rutgers Computer Science Department and the Computational Biomedicine Imaging and Modeling Lab for the great years we spent together. Especially to my teammates; Tarek and Gokhan and the Computer Vision group Ali, Amr, Mohammed, Chetan, and Babak.

I want to thank my family for always being with me even when we are apart, and for encouraging me to achieve the hardest goals.

Finally, and most importantly, I would like to thank my precious wife Selen. Her support, encouragement, and love were the reasons that kept me going on this path and finally seeing the outcome of my efforts.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

In this thesis, my objectives are to solve visual geo-localization and video segmentation problems using new Computer Vision methods that utilize information from freely available satellite images. My motivation is to build systems that use simple vision sensors and efficiently generate accurate global localization information for ground and air vehicles, autonomous outdoor robots, and people with handheld or wearable devices.

Visual geo-localization is to localize a set of images globally using vision as the main source of information. Visual geo-localization can be useful in scenarios where GPS-like sensors are unavailable or unreliable. Visual localization results can also be used to improve accuracy of localization obtained from other sensors. Furthermore, for a video that has been recorded earlier, the sensor information does not exist. In this offline localization problem, visual localization can be the only solution.

In my work on visual localization, I focused on using satellite maps to perform global localization. My objective in localization is to solve two separate visual localization problems using satellite map information. First problem is to localize a given aerial image in a very large area without having prior information on the location. For this problem, my approach is to localize the image based on the layout of the semantic

elements on the query image and to find the same layout on the map. I devised a very efficient hierarchical semantic hashing algorithm that can localize an image on search region of a large city in the order of seconds, where a naïve appearance matching based method would take years' worth of computation time for the same problem.

The second localization problem is to continuously localize a moving platform using the images coming from its cameras. The problem can be a car moving on a road, a person carrying a handheld camera or a mobile phone or a robot moving on sidewalks of a city. This is a continuous tracking problem where previous location estimation is used to calculate localization for a new frame. I solve this problem by extracting incremental motion information from images and by correcting the possible localization drift by registering the images to the satellite map. I also make use of other information extracted from satellite maps to be priors to our localization process. All sources of information are combined under an efficient probabilistic framework. I built two separate localization frameworks for stereo and monocular images, where in stereo case 3D scene and top view image can be generated using stereo disparity estimation, in monocular case Structure from Motion methods have to be employed.

Extracting meaningful information from satellite maps is a well-studied task in Geographical Information Systems (GIS) area, where road, vegetation, or building segmentation is a common task. Appearance information from satellite maps is directly available from geo-referenced raster images, whereas semantic information has to be obtained by processing the satellite maps, raster images, and other sources. In map segmentation usually the objective is extract one or more specific classes from the map, but not to label each pixel on the map with a label. Thus, segmentation map solutions

commonly focus on segmenting the classes required for a given task using special ad-hoc segmentation methods for these classes. I approach the problem as a multi-class labeling problem and label each pixel on the satellite raster image with a class label. Using learning based methods we can obtain class probabilities for map pixels. These probabilities are improved using more semantic information on the classes themselves since pixel values can only represent local features and they can be ambiguous without use of other information.

Semantic video segmentation is the task of labeling each pixel in the video frames with a semantic label. Recent advances in learning, classification, and semantic reasoning techniques yielded to many successful video segmentation systems. Use of object detectors, geometric reasoning, 3D scene structure and deep network based classification techniques can generate semantic labels for common classes with high success. Especially the street-scene scenarios, where the images are taken from a vehicle travelling on a street is a recent focus area for video segmentation studies, due to its impactful applications on self-driving cars and other car safety technologies.

Instead of improving the existing techniques in video segmentation, I addressed the problem with a completely different approach. By projecting the semantic object class information from satellite maps in to the camera-view, we achieve a semantic segmentation of video frames with no learning performed on the camera view. Although this approach creates a completely new class of video segmentation techniques, it can also be combined with the existing learning-based techniques to improve their results and making their segmentation more consistent with the known classes on the satellite map.

1.1 Outline of the Dissertation and Published Work

Introduction to the dissertation is given in Chapter 1. Related work for each topic is discussed in detail in the corresponding chapter.

In Chapter 2, I present the outline of general multi-class satellite image segmentation and labeling algorithm and sidewalk completion method.

Basis for this work is the road detection algorithm appearing in "**A Framework for Global Vehicle Localization Using Stereo Images and Satellite and Road Maps**" [46] published in 2011 at IEEE International Conference on Computer Vision (ICCVW'11) in Computer Vision in Vehicle Technologies Workshop (CVVT) with co-author Ahmed Elgammal.

Later, I applied similar approaches to sidewalk detection in "**Satellite Image Based Precise Robot Localization on Sidewalks**" [47] published in 2012 at IEEE International Conference on Robotics and Automation (ICRA'12) with co-author Ahmed Elgammal.

More advanced sidewalk detection algorithms, sidewalk completion, crosswalk detection, tree detection and tree shadow detection algorithms are published in "**Segmentation of Occluded Sidewalks in Satellite Images**" [19] in 2012 at International Association for Pattern Recognition, International Conference on Pattern Recognition (ICPR'12) with co-author Ahmed Elgammal.

Building detection algorithm is presented in "**Hierarchical Semantic Hashing: Visual Localization from Buildings on Maps**" [48] published in 2014 at International

Association for Pattern Recognition, International Conference on Pattern Recognition (ICPR'14) with co-authors Tarek El-Gaaly and Ahmed Elgammal.

In Chapter 3, I present an efficient visual localization method that can localize a given aerial image in a very large city using the layouts of buildings. The presented method is called Hierarchical Semantic Hashing and is a version of Geometric Hashing algorithm, tailored for semantic localization problem. I investigate the accuracy and efficiency of the method with experimental results for localizing satellite images in a dense part of Downtown Seattle. This localization algorithm is presented in "**Hierarchical Semantic Hashing: Visual Localization from Buildings on Maps**" [48] published in 2014 at International Association for Pattern Recognition, International Conference on Pattern Recognition (ICPR'14) with co-authors Tarek El-Gaaly and Ahmed Elgammal.

In Chapter 4, I present a continuous visual localization framework that uses stereo camera images and matches them to satellite images to obtain GPS accuracy localization. In this chapter, a platform equipped with stereo camera that is moving continuously, is localized using stereo visual odometry, orthographic satellite image matching and road map priors.

The stereo visual localization framework applied to vehicle localization on roads, is presented in "**A Framework for Global Vehicle Localization Using Stereo Images and Satellite and Road Maps**" [46] published in 2011 at IEEE International Conference on Computer Vision (ICCV'11) in Computer Vision in Vehicle Technologies Workshop (CVVT) with co-author Ahmed Elgammal.

The stereo localization framework applied to mobile robot localization on sidewalks, is presented in **Satellite Image Based Precise Robot Localization on Sidewalks**" [47] published in 2012 at IEEE International Conference on Robotics and Automation (ICRA'12) with co-author Ahmed Elgammal.

In Chapter 5, I present a continuous visual localization framework that uses monocular camera images and satellite image matching to localize a moving platform with accuracies better than GPS. In this chapter, a video taken from a car or a robot with a single camera or a handheld camera is continuously localized with the help of using Structure from Motion, orthographic satellite image matching, and perspective satellite image matching. This work is being prepared for publication with co-author Ahmed Elgammal.

In Chapter 6, I present a novel semantic video segmentation and labeling technique that uses no learning on the video frames. First, I segment the satellite map of the area into semantic classes. I perform video segmentation by localizing the frames on the satellite map and then propagating the map segmentation to the video frames. I propagate labeling information from satellite image to a set of sparse points on the video images. I then perform dense segmentation and labeling on the images using these sparse labels. Experiments are performed on monocular city sequences, where high accuracy localization, high-resolution top view images and dense video segmentation and labeling results are obtained. I further discuss possible applications of this method like augmented reality. This work is being prepared for publication with co-author Ahmed Elgammal.

In all the previously published work that has been presented here, I am the principal researcher and the principal author.

Finally, in Chapter 7, we provide a summary of the conclusions of my dissertation research along with directions for future work.

1.2 Contributions

The contributions of this thesis are novel and efficient frameworks to solve semantic satellite image segmentation, visual localization and video segmentation and labeling. A summary of important contributions is listed below.

1.2.1 Contributions in Satellite Image Segmentation and Labeling

- Completion of occluded sidewalks on satellite maps
- Detection of crosswalks on satellite maps
- Use of multi-class semantic segmentation and labeling for satellite maps
- Incorporation of structure and inter-class relations in segmentation

1.2.2 Contributions in Visual Localization

- A method that uses semantic element layouts to very efficiently, accurately and robustly localize aerial images in large environments

- Continuous visual localization from camera images using satellite maps for stereo and monocular camera images
- Generating accurate, high-resolution top view images using camera images, scene structure and motion information

1.2.3 Contributions in Semantic Video Segmentation

- Semantic video segmentation and labeling by propagating information from satellite maps on the perspective camera view
- Obtaining segmented video frames that are precisely localized and aligned with satellite images using segmentation and visual localization techniques together

Chapter 2

Semantic Satellite Map Segmentation

2.1 Motivation

In the last decade, freely available road maps have been extensively used for vehicle navigation and vehicle path planning. In order to achieve the same tasks for pedestrians, precise sidewalk maps have to be present. Although Google Maps and Bing Maps began to provide walking directions, both lack precise sidewalk locations and sometimes assume that sides of the roads are walkable, which might generate dangerous paths.

Segmenting and labeling the pixels of satellite images can yield to very important information about a region and this information can be obtained just by processing the raster satellite images of this area using a combination of computer vision and image processing techniques. Semantic segmentation of satellite maps is to label each pixel on the satellite map with a semantic class label. Semantic classes are types of object classes that are meaningful to human understanding.

For various tasks in GIS applications researchers worked on segmenting specific semantic classes from satellite maps, aerial images, multi spectral imagery, and other aerial image sources. Segmenting orthographic aerial maps is a well-studied and more straightforward task than segmenting complex 3D scenes viewed by a perspective

camera. Furthermore, map segmentation can be performed offline and with the help of additional information, such as road maps, community entered building footprints, municipal, and transportation maps. We can label satellite maps for many different object or region classes including roads, sidewalks, buildings, trees, grass, cars, and crosswalks.

The results of satellite image segmentation in this chapter will be used extensively throughout the rest of the chapters. In Chapter 3, to achieve vision-based localization from building structures, we use segmentation of buildings from satellite maps. In Chapter 4 and 5, to provide prior location information for our camera based visual localization techniques; we make use of road and sidewalk segmentations from satellite maps. In Chapter 6, we use semantically segmented satellite images and propagate this information in to camera view, in order to segment out a given video in to similar classes.

The best resolution of publicly available satellite images goes down to 10-15 cm/pixels and view angle is orthographic top view, but it is also worth noting that there is an abundance of publicly available oblique orthographic images taken from low-altitude airplanes. With the low resolution and viewpoint of the satellite images, the object classes that can be resolved are usually limited to roads, cars, sidewalks, crosswalks, building roofs, grass, trees, soil, and water. Among these classes, only cars have the property of being non-stationary. Cars in the satellite image will not correspond to cars in the same area, since the satellite image has been taken a long time ago. All other classes are stationary objects that will still be there if the actual area is visited. Depending on the location, the satellite images will be months or up to years old and between the time the image taken and the current time, even the stationary objects may be changed, e.g.; buildings can be demolished, new buildings can be built, roads can be repainted or even

rerouted. Therefore, it is important to know that satellite images will not be giving an absolute reference to register to but partially correct information and they should be used as priors in our estimations.

We base our map segmentation approach on a multi-class segmentation that generates pixel class probabilities. On top of independent pixel probabilities, we apply structural rules to represent specific properties of each class. Final step is to exploit the relations between classes. We employ inter-class relationships to resolve possible ambiguities in final classification. For example, crosswalks have to be situated on the road. With our general goal of a fully autonomous sidewalk-navigating robot, we focus more in the details of estimating sidewalks and crosswalks, and we develop novel methods for estimation of whole sidewalk grid.

The results of satellite image segmentation in this chapter will be used extensively throughout the rest of the chapters. In Chapter 3, to achieve vision-based localization from building structures, we use segmentation of buildings from satellite maps. In Chapter 4 and 5, we make use of road and sidewalk segmentations from satellite maps to provide prior location information for our camera based visual localization techniques. In Chapter 6, we use semantically segmented satellite images and propagate this information in to camera view, in order to segment out a given video in to similar classes.

2.2 Related Work

Due to its possible usage in vehicle navigation and safety applications, road detection is a well-studied area in GIS, mapping, and robotics applications. Road information can be very useful in determining where a vehicle can drive in a city, and thus can be used directly in navigating an autonomous vehicle or a robot. Some specific road detection algorithms are given in [1]-[4], where semantic information, use of classifiers, and training neural networks methods are employed, besides appearance. An extensive review on the literature on road detection area is given in [5] with a summary of more than 200 research articles.

Similar to roads, building segmentation from satellite, aerial, oblique and LiDAR imagery, and Digital Elevation Maps (DEM) has been a popular topic and [6]-[10] and many others proposed multitudes of techniques ranging from root top detectors to image segmentation, and to neural network based classifiers. Training appearance classifiers from satellite maps with the help of community entered building footprints is also an approach to detect buildings on satellite maps.

To obtain statistics of vegetation in large areas, researcher has studied GIS analysis on vegetation like crops, grass and trees. Few examples for tree and vegetation detection from aerial images are given in [6], [11], [12].

Unlike the other classes, sidewalk detection from satellite images has not been studied extensively due to the fact that relatively higher resolution aerial images are needed to detect narrow sidewalks from above and due to heavy occlusions from trees

and buildings. Although research on sidewalk and crosswalk detection from street level view images exists [13]-[18], to our knowledge, there exist no work on an automatic method to perform complete sidewalk segmentation from aerial images prior to our sidewalk detection and completion work given in this chapter [19].

2.3 Satellite and Road Map Acquisition

We download freely available satellite and road maps from Google Maps and stitch pieces by estimating the homographies between pieces to build larger maps. We obtain *prior sidewalk probabilities* for sidewalks, based on their proximity to roads, as described in Chapter 3 [47]. Furthermore, we obtain road orientations that capture local road orientation angle for each road point.

We use Google Static Maps API [51] to download freely available satellite and road maps. This API allows users to obtain satellite images and road maps through HTTP requests. User specifies the location, zoom factor, map type, image size, and format in the query string, and receives an image file in response to the HTTP request.

Google Static Maps supports maps with custom appearance with the help of *styled maps* in addition to the default street, satellite, hybrid, and terrain map types. Using this feature, we create a custom map style that we call *road map,* where we remove all non-road features and text from the street maps (vector maps) and selecting foreground color as white and road color as black, Figure 2.1. The resulting binary map image is now suitable for image processing algorithms to recover road locations. Use of these road

maps simplifies the otherwise complicated road segmentation step (e.g., road extraction step performed in [52] manually and in [53] using structure tensors).

Static API image sizes are limited to 640x640 pixels. In order to obtain the satellite and road map of larger areas (e.g., entire Busch Campus of Rutgers University) we download small overlapping patches over the area and stitch them together.

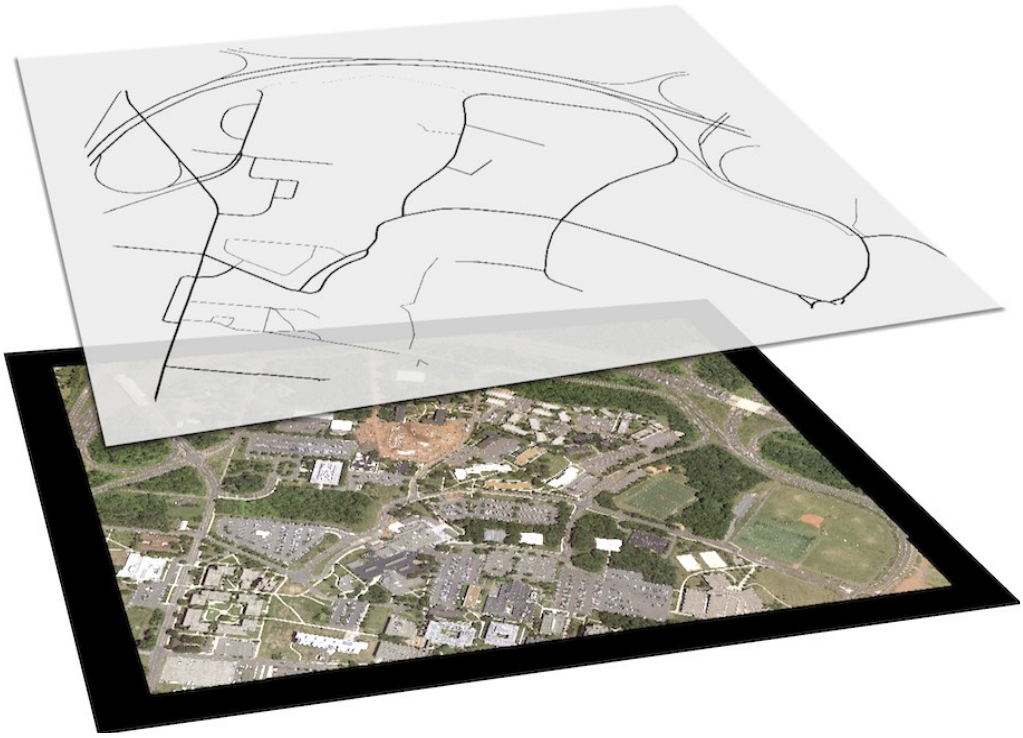


Figure 2.1 – Road map and satellite map of Rutgers Busch Campus obtained from Google Static Maps API on top of each other. A 1 km x 1 km area is covered by stitching together 9x9 patches.

2.4 Pixel Level Classification Using Random Forest Classifier

As a first step of the satellite image classification, we train a multi-label Random Forest classifier [76] over the map image to obtain pixel-based prior probabilities for

class memberships. Classifier is trained with samples from *road*, *grass*, *tree*, *building*, *sidewalk,* and *shadow* classes, although shadow class is only used to help determining a pixel's membership to other classes. For multi-label classification WEKA framework [20] is used with a combination of color, edge, and texture features for training and initial classification step.

2.5 Use of Structure

Although methods change from class to class, we use structural properties of each class to prune out possible misclassified pixels. Class-specific morphological filters are applied to the initial pixel-level classification results for each class. With the use of these morphological filters, we can guarantee minimum thickness, minimum area, maximum curvature or similar desired features of class pixels. A sample segmentation result is shown in Figure 2.13 for road, grass, tree, sidewalk, and building classes.

2.6 Use of Inter-Class Relationships

On top of the structural constraints applied to pixel wise probabilities of classes, some external information or inter-class relations can be used to further modify the posterior classification probabilities. Known relations between classes can be exploited to improve the classification results. Few examples to such relations are the relationship between roads and sidewalks, roads, and crosswalks. To improve our classification

results, we can use the facts that majority of sidewalks are next to roads and they have similar orientation to the road next to them. Most of the times crosswalks are located on roads and they are perpendicular to the road they are crossing. Furthermore, crossroads join two ends of sidewalks. Inter-class relations similar to these examples can be used to improve the results of our semantic satellite maps segmentation task.

2.7 Road Segmentation

When defining the road class, we consider regions of the map that are covered with asphalt or concrete roads. In general, road class includes any types of roads, streets, highways and parking lots. Structurally roads cannot be too thin or too short. Another structural property is that small holes in the road regions should be considered part of the road since roads generally don't have such holes. In addition, since roads are designed to be part of a road network, road regions should be connected with other road regions.

Along with the satellite maps obtained from Google Static Maps [51], we also acquire the road map of the area using styled maps feature of Google Static Maps API. Road map is a binary image, where roads are white and all non-road areas are black. Satellite maps and road maps requested for the same coordinates match in few pixels accuracy (except the faulty mapped areas in Google Maps). Road maps can be used to determine the locations of roads on satellite maps without using any complex image segmentation techniques.

If available, road maps of the area can be a good prior for modifying the road classification probabilities. Map providers store road maps as vector data and render them differently based on the output resolution. Road widths are not stored accurately and rendered differently based on the resolution for practical reasons. Moreover, road maps do not include parking lots or other road related structures that are connected to the road network. Because of these reasons and due to acquisition time differences, road maps never perfectly match the underlying satellite map roads pixel to pixel; that is why we only consider road maps as a prior modifier to pixel-wise probabilities and not as ground truth information.

It is also important to consider that some parts of the road can be occluded by building shadows or trees at the side of the road. Another important occluder for roads would be parked and traveling cars. Usually a region where a car is detected is a road region. This fact can also be used to improve car and road detection both ways.

Road segmentation uses both pixel-based road classification probabilities and road maps from Bing Maps and Google Maps. Ground truth road maps are not always pixel-wise accurate since they are formed of key points on the roads and the rest is generated by spline or line fitting to these key points. In addition, the width of the roads in the road maps is not accurate and usually it is fixed for the same type of roads. These road maps also do not include parking lots, and asphalt paved road pieces that go inwards to building entrances. To be pixel-wise accurate in road segmentation from maps, these problems have to be fixed.

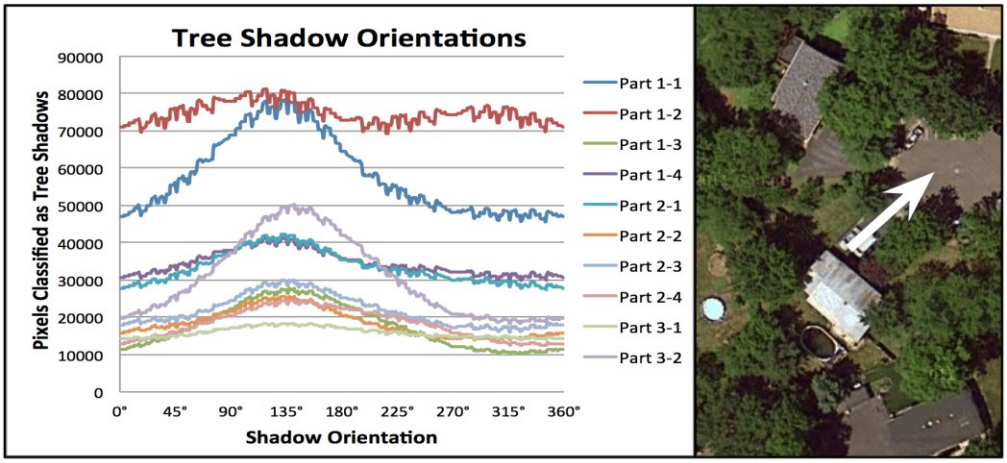
Our road segmentation starts with high road probability regions from pixel-wise classification results. These probabilities are modified using road map priors. Small road

patches are removed and holes are filled, since roads cannot have small holes in them and there cannot be independent small road pieces. Furthermore, pixels known to be grass or sidewalk are removed from the segmentation and known road ground truth locations are added to the segmentation.

2.8 Tree Segmentation

Besides other uses like the determination of the amount and locations of trees in an area, detecting trees is necessary to determine occluded sidewalks, since trees are the main occluders of sidewalks in satellite images. We use the classification results of random forest classifier, [76], trained on training images to classify trees on the satellite maps.

Together with the buildings, tress will be the major source of the shadows on a map. Thus, we apply shadow detection algorithm on the trees that we found on the map. We then detect the shadows pixels of the tree regions using a color classifier on top of the random forest classification for shadow class. In order to find the correct orientation of the shadows, we extend tree regions in all possible directions and find the angle that maximizes the number of pixels classified as tree shadows under the extended region Figure 2.2.



(a) (b)

Figure 2.2 – (a) Tree shadow orientations. Number of pixels classified as tree shadows for each shadow angle. Different lines correspond to separate non-overlapping satellite image parts. (b) Sample satellite map region and corresponding best shadow direction.

Tree class is made of all kind of trees and vegetation that has a distinguishable texture other than the grass areas. Although different trees can be of different sizes, we still consider that there is a minimum size for trees observed from above and we ignore trees that are not easily observable from above.

Segmented tree pixels are found by thresholding the tree pixel classification probabilities, where small holes are filled and very small islands are removed. Dark pixels inside tree regions that may be self-shadows of trees on themselves are also added to the final segmentation results.

2.8.1 Results of Tree Segmentation

Tree segmentation results without and with tree shadow detection are given in Figure 2.3 and a more detailed view is given in Figure 2.4. In these results, 100% detection

accuracy is achieved by detecting all trees in the input map without introducing any false positives. Furthermore, with the inclusion of shadow pixels, the pixel-wise detection accuracy is also greatly improved.

2.9 Sidewalk Segmentation

Accurate segmentation of sidewalks from satellite images can be required in various applications, such as giving walking directions to pedestrians or automatic robot navigation. We propose a framework to construct sidewalk and crosswalk maps from satellite images. This is a challenging task, since typically sidewalks in satellite images are highly occluded by trees and their shadows. Furthermore, there can be several objects on maps that have similar appearance to sidewalks. In our framework, we initially segment visible sidewalks by their appearance, and then complete the tree-occluded areas by posing the problem as an image-inpainting task with multiple priors. This fuses the knowledge about roads, occluders, and sidewalk structures. We present successful sidewalk segmentation results from satellite images, where sidewalks are highly occluded by trees.

Figure 2.3 – Tree segmentation results. (a) Satellite images. (b) Tree detection result. (c) Tree detection results with detected tree shadows included. Best viewed on computer screen.

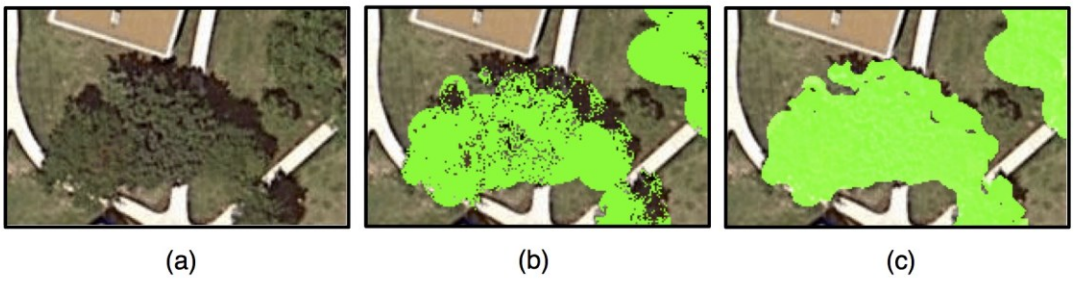


Figure 2.4 – Tree segmentation results details. (a) Satellite images. (b) Tree detection result. (c) Tree detection results with detected tree shadows included.

Besides pedestrian path-planning, another important application of sidewalk detection is navigating autonomous mobile robots on sidewalks in realistic city environments, [74]. Using the final sidewalk map, the robot can run a global path-planning algorithm. When executing this global plan, robot can perform local path-planning and obstacle avoidance and can further improve the sidewalk map by including its local observations.

Sidewalk class consists of any pedestrian walkable path or sidewalk. Sidewalks have a minimum thickness as their structural property. Although connectivity is a desired property, it does not always exist in real life scenarios where sidewalks discontinue due to roads or grass areas interfering.

Whenever available, pedestrian walkway maps provided by map providers is a good prior for detection but this information is frequently inaccurate due to vector data and rough estimations of actual paths made by the map provider and no thickness information provided with pedestrian walkway maps. In addition, current maps cover a very small percentage of the actual pedestrian walkways.

Another prior that can be used to estimate sidewalk segmentation is the existence of roads; in most cases sidewalks are walkways that are on the side of the roads (as the

name implies). Being close to road class pixels can be modeled in to a plausible sidewalk priority probability for sidewalks that are next to roads.

In some areas, sidewalks will be one of the most occluded classes. Trees planted next to the roads and their shadows can cover a large portion of the sidewalks and lower their visibility on the satellite image. We devised a novel image-inpainting like algorithm to complete the occluded sidewalks as an extension to visible sidewalks and by explicitly searching under the trees.

On top of the pixel-wise classification and structural filtering steps, we developed an adaptive approach for sidewalk segmentation, local appearance based region-growing, and adaptive outlier rejection.

First, we apply a color-based classifier over SV color space for initial sidewalk segmentation and train this classifier on several known sidewalk regions in training images. However, there are buildings and several other kinds of regions that can be wrongly segmented as sidewalks, because of their similar appearance to sidewalks. Therefore, we use oriented filters to classify segmented pixels as sidewalks, buildings or non-linear structures, by fitting thin structural elements in all directions and summing up the filter responses. Small response corresponds to linear structures and classified as sidewalks, whereas large response corresponds to large areas and classified as buildings. Then these regions are removed from sidewalk segmentation. Regions, where structural elements cannot fit, are classified as small non-linear structures, which are also removed from the segmentation.

We use sidewalk segmentation results from the structural classification as seeds for an appearance-based region-growing algorithm, which finds all connected pixels that

have similar appearance, measured on SV color space. Region growing can capture local color similarities of pixels in addition to the global color model. Finally, we adapt a color model from pixels classified as sidewalks, and based on this new model we reject regions that are very different in appearance then the rest.

Segmented sidewalk pixels are found by thresholding the sidewalk pixel classification probabilities, where small holes are filled and very small islands are removed. After this, sidewalk parts that are occluded by trees are filled using the image-inpainting sidewalk completion method as described below.

2.9.1 Completion of Occluded Sidewalks

Initial segmentation steps provide probability maps for visible sidewalks, roads, buildings, trees, and tree shadows. Using these, in this step, we determine the locations of occluded sidewalks to complete the sidewalk connectivity on the map. Once the occluded sidewalks are found, we pose the filling problem as an exemplar-based inpainting task with priors. We perform our proposed inpainting algorithm on the segmented sidewalk image. We use the area under trees and tree shadows as the area to be filled.

For inpainting task, we adapt and modified the approach described in [77], which can fill missing parts in natural images in a visually plausible manner. In this algorithm, texture filling is performed while linear structures and object contours are preserved. Sample patches are picked from known or newly filled areas of the image. However, there are major limitations in direct application of this approach to our problem. Method

described in [77]: (1) allows only linear/straight continuation of sidewalks; (2) can cause growing of non-endpoint areas; (3) can result in filling with flawed examples, Figure 2.5; (4) can cause different sidewalk branches to meet beyond their actual meeting points, Figure 2.6. Problem (4) happens when branches have different distances to the meeting point and get extended equally by the inpainting algorithm.

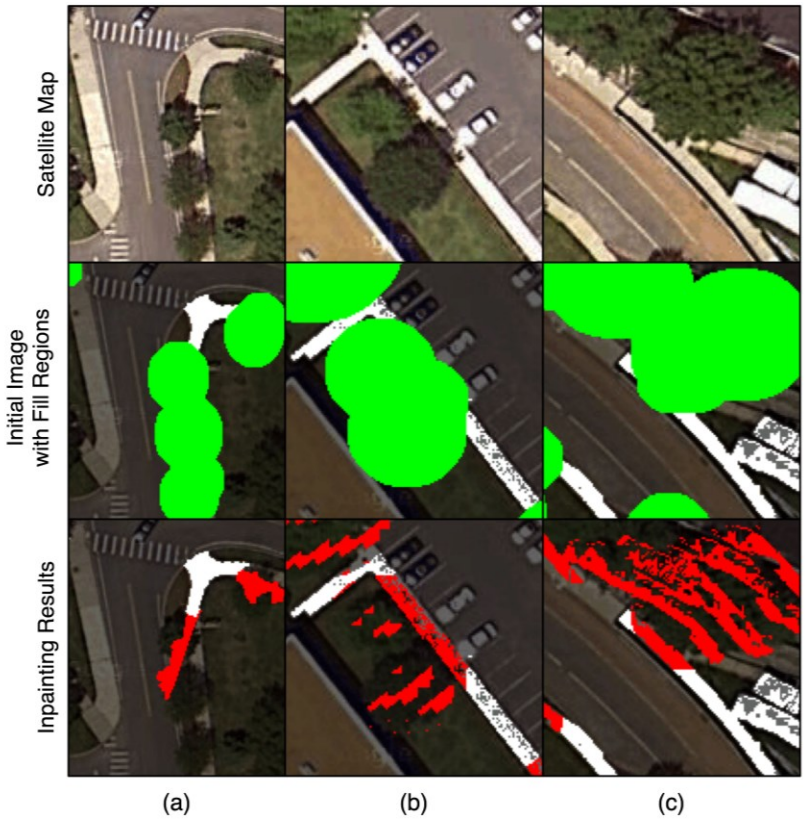


Figure 2.5 – Problems that arise when using inpainting for sidewalk completion. (a) Straight continuation. (b) Non-endpoint continuations. (c) Copying flawed examples. In the initial image, regions to be filled are green and actual inpainting source image is white. In the results, image inpainting results are red.

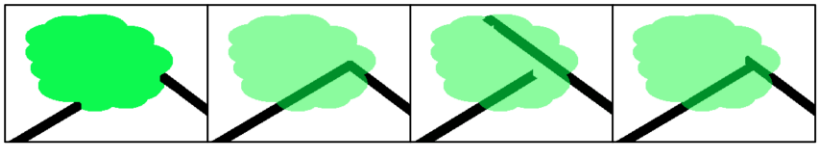


Figure 2.6 – Branch meeting problems that may arise in inpainting for sidewalk completion. (a) Initial image. (b) Expected - ground truth. (c) Inpainting results, where branches miss each other. (d) Structure growing - extend-trim method.

In order to overcome inpainting algorithm problems, we propose a structure growing based method, where blocks are selected from a pre-built library of oriented sidewalks. Moreover, growing starts only from sidewalks endpoints, allowing us to not fill all occluded regions. When growing an endpoint, next block to put is selected from the library based on several priors as outlined below.

Block matching – continue straight. Similar to other block based image-inpainting approaches, correlation score of library blocks with the known part of the current block helps continuation of linear elements.

Road orientation – go parallel with the road. Similarity of orientations of the current edge with the closest road point.

Prior sidewalk probability – go next to the road. Sum of the prior sidewalk probabilities in the area under the candidate block, favors sidewalk blocks closer to road.

Endpoint attraction – go towards other endpoints. Analogous to static electric fields being stronger in high-curvature points, we create a vector field that increases as it gets closer to an endpoint. Sum of probability vector field in the area under the candidate block guides the sample towards other map endpoints.

In order to overcome branch meeting problem, we develop a method called extend-trim, where we first extend all endpoints separately then find the maximum number of steps that an endpoint can extend until it touches another extended endpoint. We use the maximum number of steps as the actual extension amount, Figure 2.6.

2.9.2 Results of Sidewalk Segmentation

We test our algorithm on satellite images from Rutgers Busch Campus in New Jersey. Sidewalk ground truths are marked by operators, by referring to Street View images or by visiting the original places. Structure growing result details are shown in Figure 2.7.

Table 2.1 – Path prediction results

Correct concealed paths – true positive	Missed concealed paths – false negative	False predicted paths – false positive	Precision	Recall
41 / 54	13 / 54	6	87%	76%

Final results of the segmentation, structure growing and crosswalk detection together with the ground truth and original satellite imagery are shown in Figure 2.8. Table 2.1 gives the prediction results and precision-recall in terms of number of concealed/occluded paths.

In order to compare the results shown in Figure 2.8 with the ground truth pixel-wise, we use binary dilation by an amount of an average sidewalk width. If we tolerate predictions that are off by a width of a sidewalk to be correct, the resulting pixel-wise precision is 83% and recall is 80%. For the same dilation value, initial segmentation gives

85% precision and 62% recall. Using structure growing, we reduce pixel-wise missed paths by 18%, without sacrificing precision. Precision and recall are less than 70% when the regular inpainting algorithm in [77] is utilized, even if an operator-guided procedure is used to pick the correct samples for inpainting.

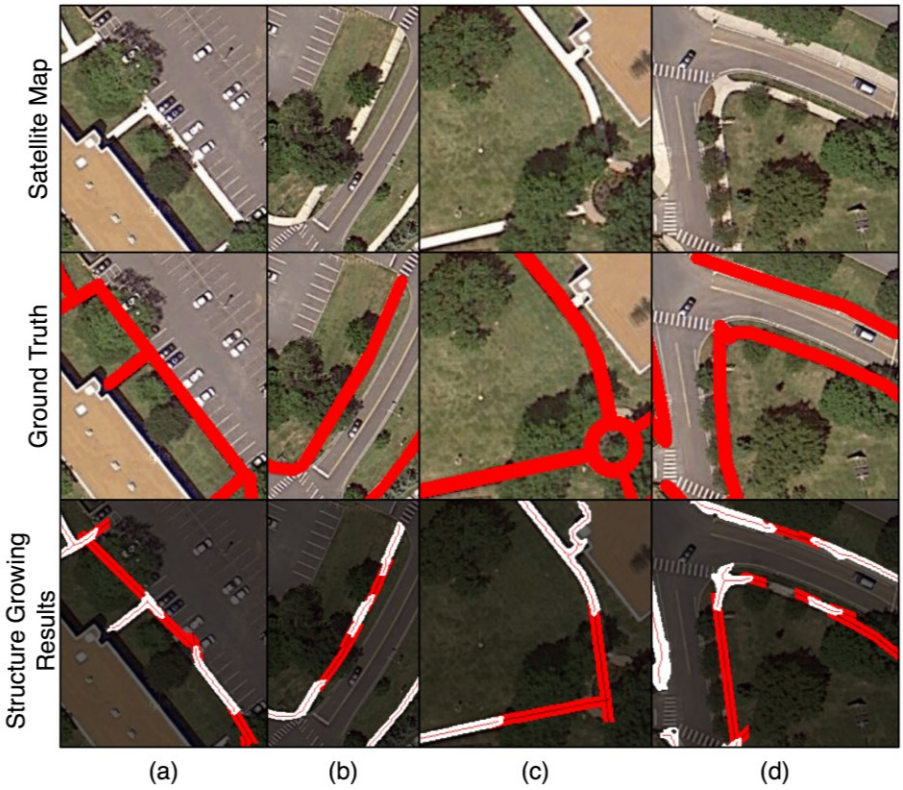


Figure 2.7 – Structure growing results details. In results images, segmentation outputs are white and structure-growing results are red. (a) Continuation of linear structures on small edges. (b) Straight continuation (c) Joining of concealed intersections. (d) Growing parallel to roads.

Figure 2.8 – Segmentation, structure growing and crosswalk detection results. (a) Satellite map. (b) Ground truth sidewalk locations. (c) Sidewalk segmentation (white), structure growing (red) and crosswalk detection (blue) results.

2.10 Crosswalk Detection

Crosswalk class consists of visibly distinguishable crosswalk regions in satellite images. Although in our work we only focus on zebra pattern crosswalks (parallel white lines), this can easily be extended to other local crosswalk structures. In our work for crosswalk detection, we focus more on detecting the location and orientation of the crosswalks instead of the perfect pixel-wise segmentation. Crosswalks have a minimum length and thickness. Zebra pattern is easy to search with wavelet, cross correlation and other image and signal processing techniques, but not all successful detections of zebra pattern may mean a crosswalk. Very important inter-class relations that can be used are: 1) crosswalks exist on the roads; 2) they are usually perpendicular to roads; 3) they connect two sidewalks together. This information can be used to successfully filter out erroneous raw detections of zebra patterns.

2.10.1 Using Results of Sidewalk and Road Segmentation

To complete the connectivity of the sidewalk maps, we extract the locations of possible crosswalks –or zebra crossings- in the satellite images. Without knowing the locations of crosswalks, acceptable path-planning cannot be performed.

Similar to [3], we use a filter bank that contains matching filters with different frequencies and all possible orientations. We run the filters on the bright pixels that are on the road or near the road regions. We replace high response points that are close to each other with a single crosswalk center point that sits in their centroid and set its orientation to the median of the orientations of these points. Orientations of crosswalk center points that are on the road are compared to the road orientations at those locations, and they are rejected if they are not close to being perpendicular to the road. Each center point is grown both ways along its orientation until it touches a segmented sidewalk point, Figure 2.9. If the crosswalk grows too long, the point is rejected, based on the assumption that crosswalks should be set between two sidewalks.

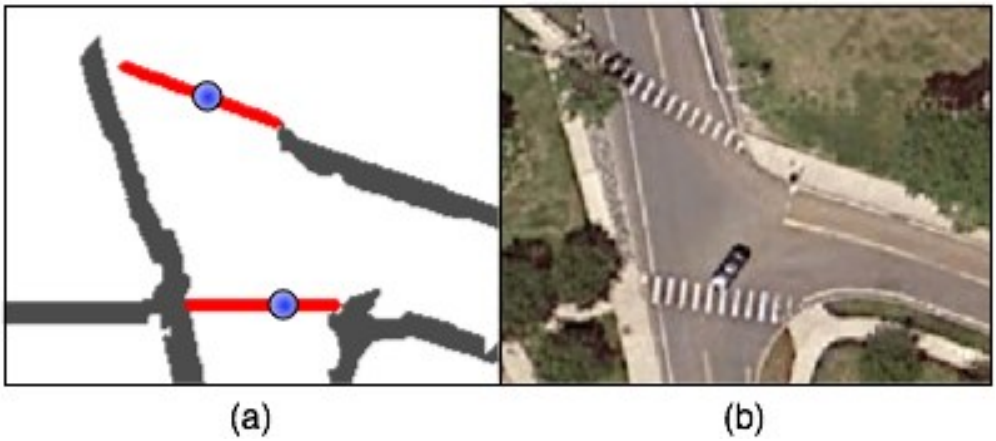


Figure 2.9 – (a) Segmented sidewalks (gray), detected crossing center points (blue) and grown crosswalk lines (red). (b) Corresponding original satellite image.

2.11 Building Segmentation

Segmented building pixels are found by thresholding the building pixel classification probabilities, where small holes are filled and islands that are smaller than possible building sizes are removed. The variety in building roofs causes them to be mixed with most of the other classes in pixel-wise classification, so we remove known road, grass, tree and sidewalk pixels from the thresholded results to obtain the final building segmentation results.

Building class consists of observed buildings on the map and other building like structures. For example, a big tent, a stadium, a swimming pool, a large storage container should also be considered as buildings. Alternatively, a swimming pool can also be a part of water class. Buildings cannot be too small in area or they cannot be too thin. Buildings are one of the major sources of shadows in the map. Buildings can cast shadow on all

other classes, even on other buildings. If the building footprint map of the area is available, this is a good prior to the building detection. This information is usually more pixels-wise accurate when compared to road map information for roads.

2.11.1 Large-Scale Building Dataset

In order to test our building detection algorithm, we used a building footprint dataset of Seattle, provided by Seattle City GIS Program [93]. In the provided GIS dataset, geographic coordinates and detailed outline contours of buildings in Seattle are given. In the dataset, buildings are extracted from imagery acquired in 2009 and further manually processed for higher precision, in the order of centimeters. Data provided is composed of individual building contour polygons with WGS84 and feet coordinates. Whole building footprint data consists of 284,017 buildings from Seattle, covering an area of approximately 23 km x 38 km (Figure 2.11). To give a comparison; as of 2014, Manhattan Island in New York has roughly 100,000 buildings.

The part of the dataset we used in our building segmentation experiments covers a dense 16.5 km^2 map area that contains 7,000 buildings. We obtained satellite images of the area from Google Maps, which is shown in Figure 2.10 (a). GIS building outline data overlaid on top of Google Maps images is shown in Figure 2.10 (b). We chose this part of the city to include both the downtown area containing large and sparse buildings Figure 2.10 west/left) and the residential Central District area containing neighborhoods of small and dense buildings (Figure 2.10 right side of the figure).

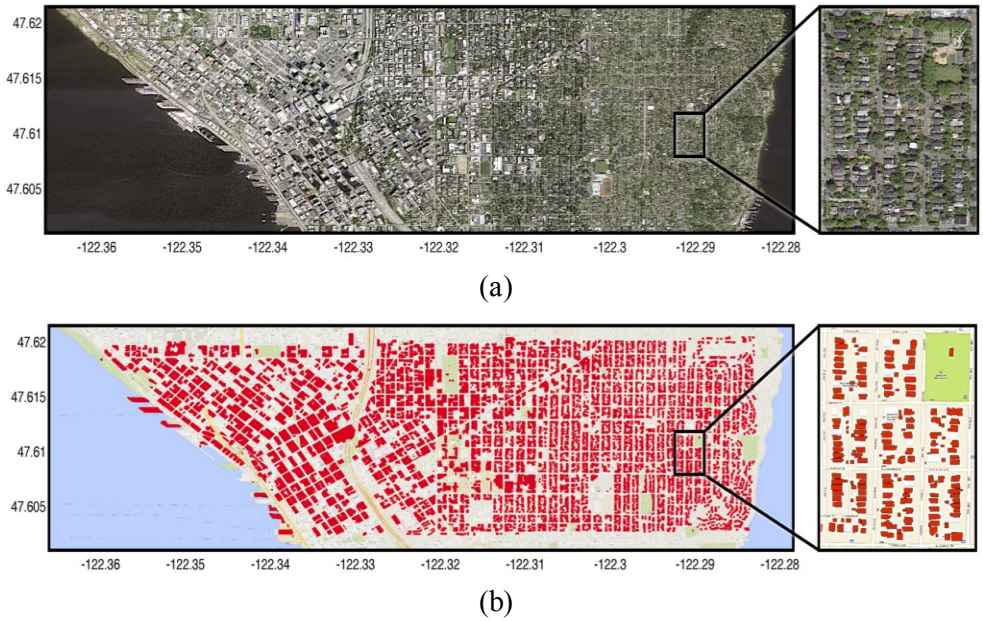


Figure 2.10 – (a) Google Maps Satellite image of the area that has been used in the localization queries. (b) Same area with ground truth GIS building outlines overlayed on top of the Google Maps image of the area. All 7,000 buildings with their outlines are shown on the map. Best viewed on computer screen and in color.

2.11.2 Building Detection Results

Building detection algorithm extracts building footprints from a given image (*e.g.* Figure 2.12 (a)) even in the presence of occlusion from vegetation and similarities in the appearances of roads and building roofs. As a first step, we use image classifier to obtain pixel-wise classification probabilities for buildings (Figure 2.12 (c)). The classifier is a three-class Random Forest classifier trained over training samples of vegetation, road and building regions. Building classification results are binarized and noise reduction filters are applied to remove unwanted salt and pepper noise and small components. Each connected component in the resulting image that has a large enough area and that is

structurally not too thin for a building is considered as a building seed region. Holes are filled and building seeds are expanded outwards based on color similarity to complete missing parts of the buildings. Final detection results are shown in Figure 2.12 (d). Missed detections and false detections are highlighted in Figure 2.12 (f). Most of the missed detections are due to tree occlusions and buildings with similar appearance to roads. False detections occur when isolated road patches are detected as buildings. With our method, we obtain satisfactory results and the detection accuracy is listed in results section.

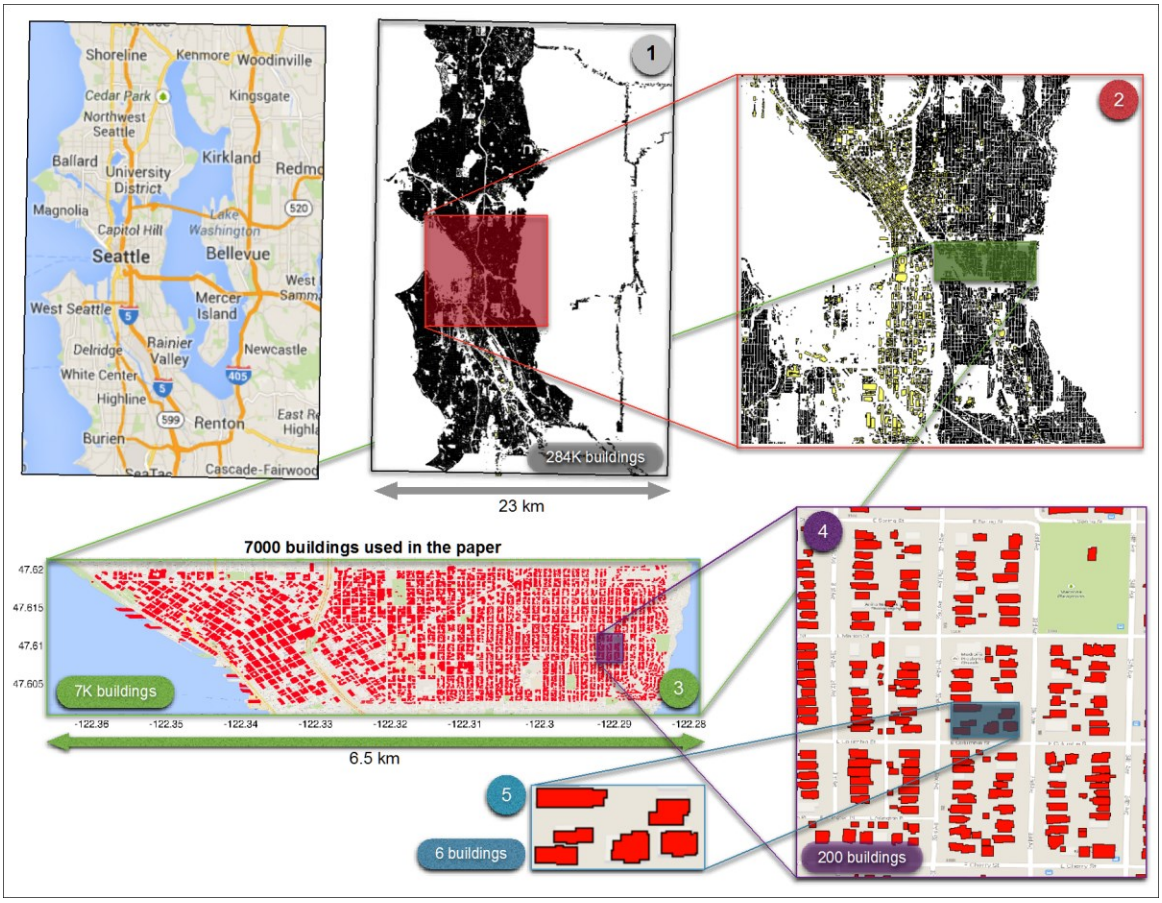


Figure 2.11 – Full Seattle building footprint GIS dataset with 284,017 buildings. Showing the central downtown area that contains 7,000 buildings used in some parts of the localization experiments.

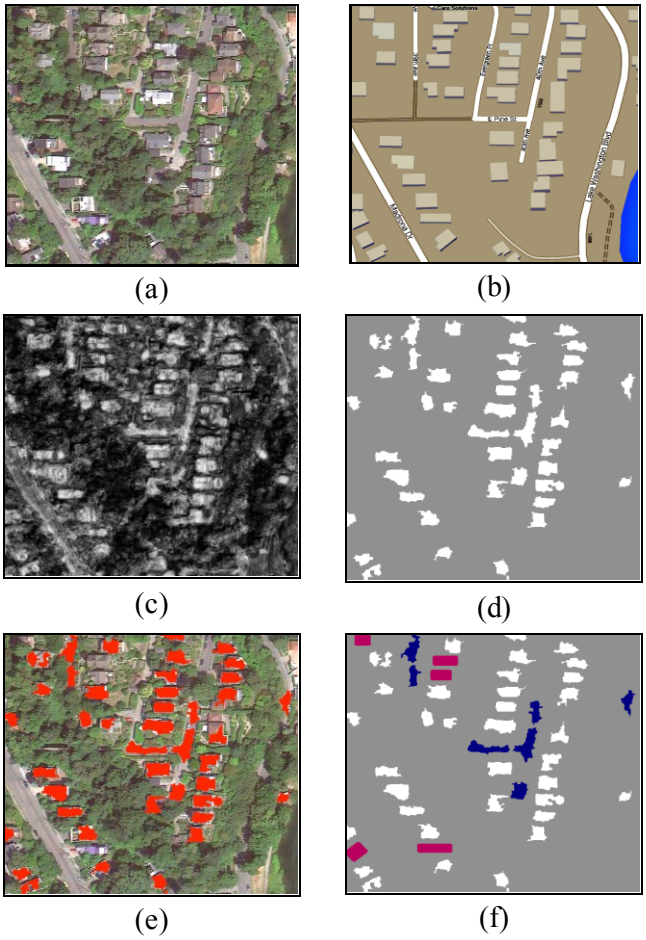


Figure 2.12 – Building detection results on a sample tile. (a) Google Maps Satellite image tile of size 1024 x 1024 px. (b) Google Maps tile. (c) Pixel-based building classification results obtained from satellite image, brighter values are higher building probabilities. (d) Final building detection results. (e) Building detection results overlayed on satellite image. (f) Building detection results showing correct (white), missing (magenta) and false (dark blue) detections. Best viewed on computer screen and in color.

Building detection results based on building locations have **65% *precision*** and **81% *recall***, where precision and recall are defined on correctly detecting a building with its majority of pixels and not matching all building pixels exactly. Our method has high recall, but also suffers from false positives that decrease the precision. Majority of false positives come from roads having similar appearance to rooftops and false negatives come from tree-occluded buildings and complex building structures.

2.12 Grass Segmentation

Segmented grass pixels are found by thresholding the grass pixel classification probabilities, where small holes are filled and very small islands are removed. In addition, known tree pixels are removed since trees can reside inside grass regions.

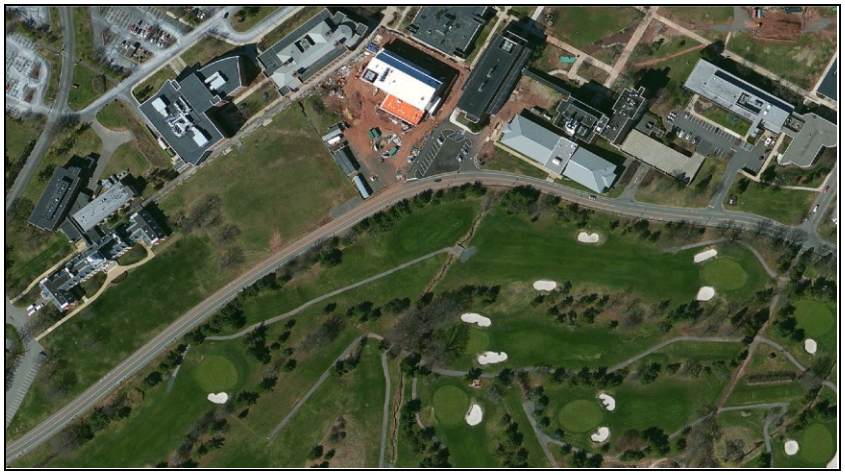
Grass and soil class is made of flat regions with soil and areas partially or fully covered with grass. These regions do not have any structural properties, since they can be as small as a grass area surrounding an electric pole or as thin as a grassed or soil road divider. They can be disconnected from each other and can have other areas of other classes inside them like trees, sidewalks or others.

2.13 Combined Segmentation Results

Results for combined satellite image segmentation for road, grass, tree, sidewalk, and building classes are shown in Figure 2.13. Original satellite images of the area are shown in the same figure including Google Maps and Bing Maps images. The results are very promising except some discontinuities in sidewalk parts that lie under building shadows. At this experiment, we have not extend the sidewalk completion algorithm to complete sidewalks under building shadows, but only under trees as the initial design of our algorithm suggests.

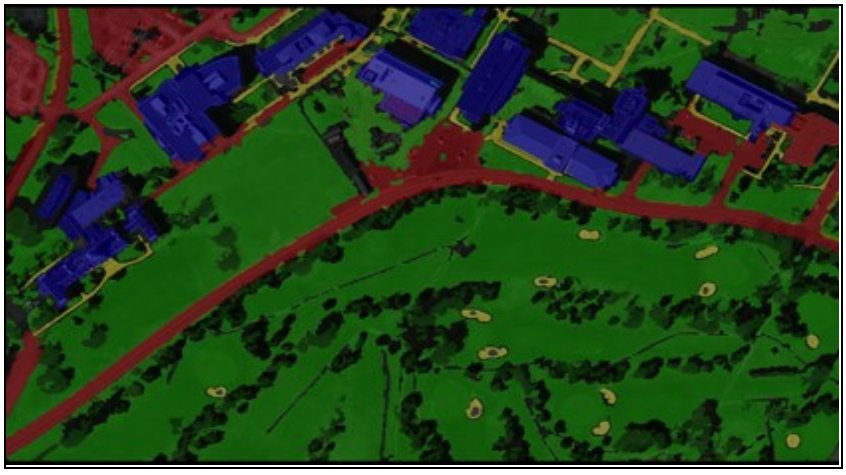
The results for initial pixel classification probabilities obtained from Random Forest Classifier are given in Figure 2.14. In here, probabilities for pixels belonging to shadow

class are also shown. Later this information is used in classifying other classes. Final segmentation results for each class are shown in Figure 2.15. Golf course puddle pixels have a very high resemblance to Random Forest Classifier samples for sidewalk class has been trained on and thus these also appear sidewalks in the final results. This and similar possible errors can be fixed using more wide variety of training samples for the classifier.



(a)

(b)



(c)

Figure 2.13 – (a) Bing Satellite image (April 2011). (b) Google Maps Satellite image (September 2013). (c) Satellite map segmentation and labeling. Showing class labels obtained for road (red), grass (light green), tree (dark green), sidewalk (yellow) and buildings (blue).

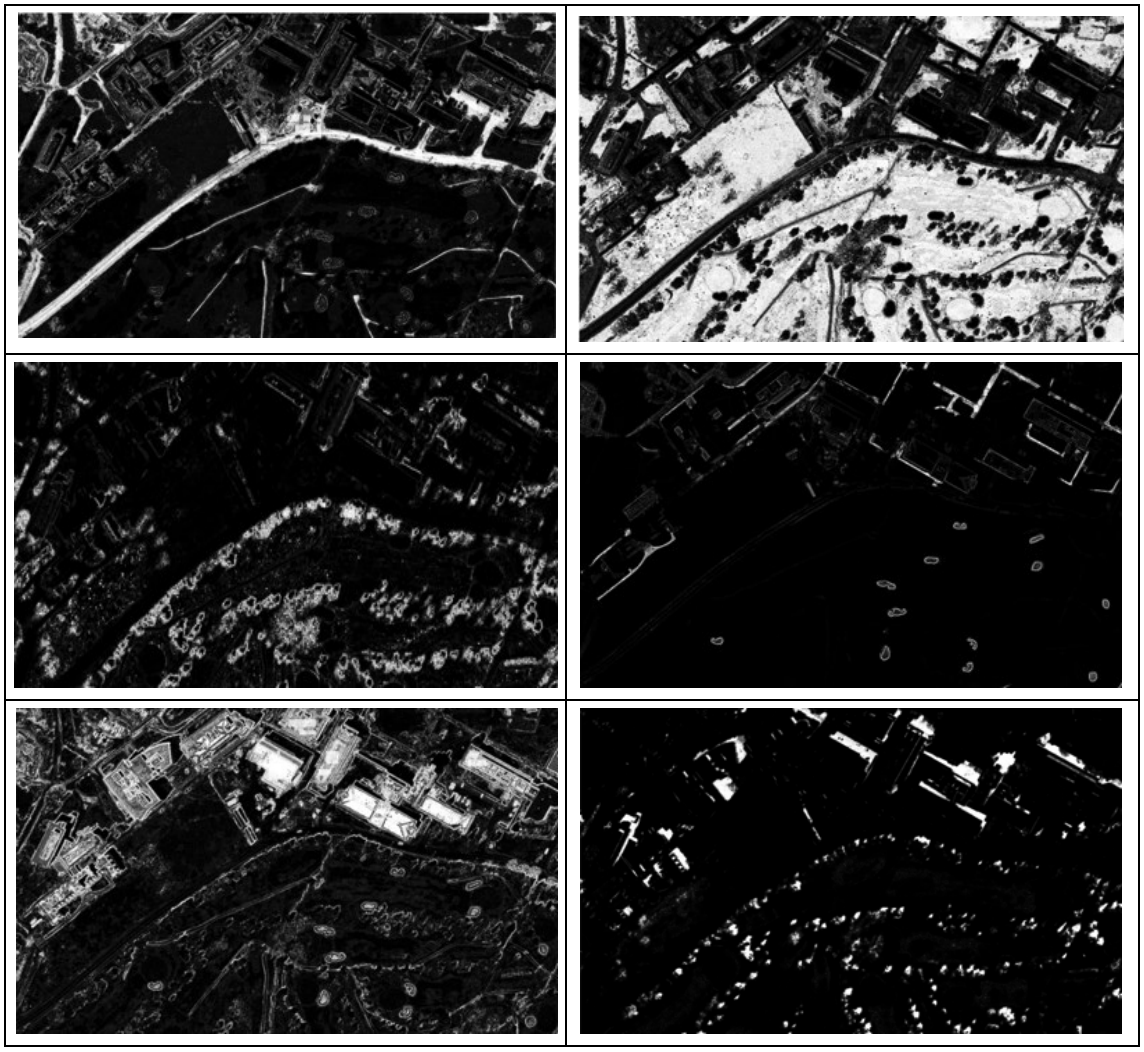


Figure 2.14 – Initial pixel classification class probabilities obtained from multi-label Random Forest Classifier. First row: Road, grass; second row: tree, sidewalk; third row: building and shadow probabilities obtained from Google Maps are shown for the map for campus road dataset.

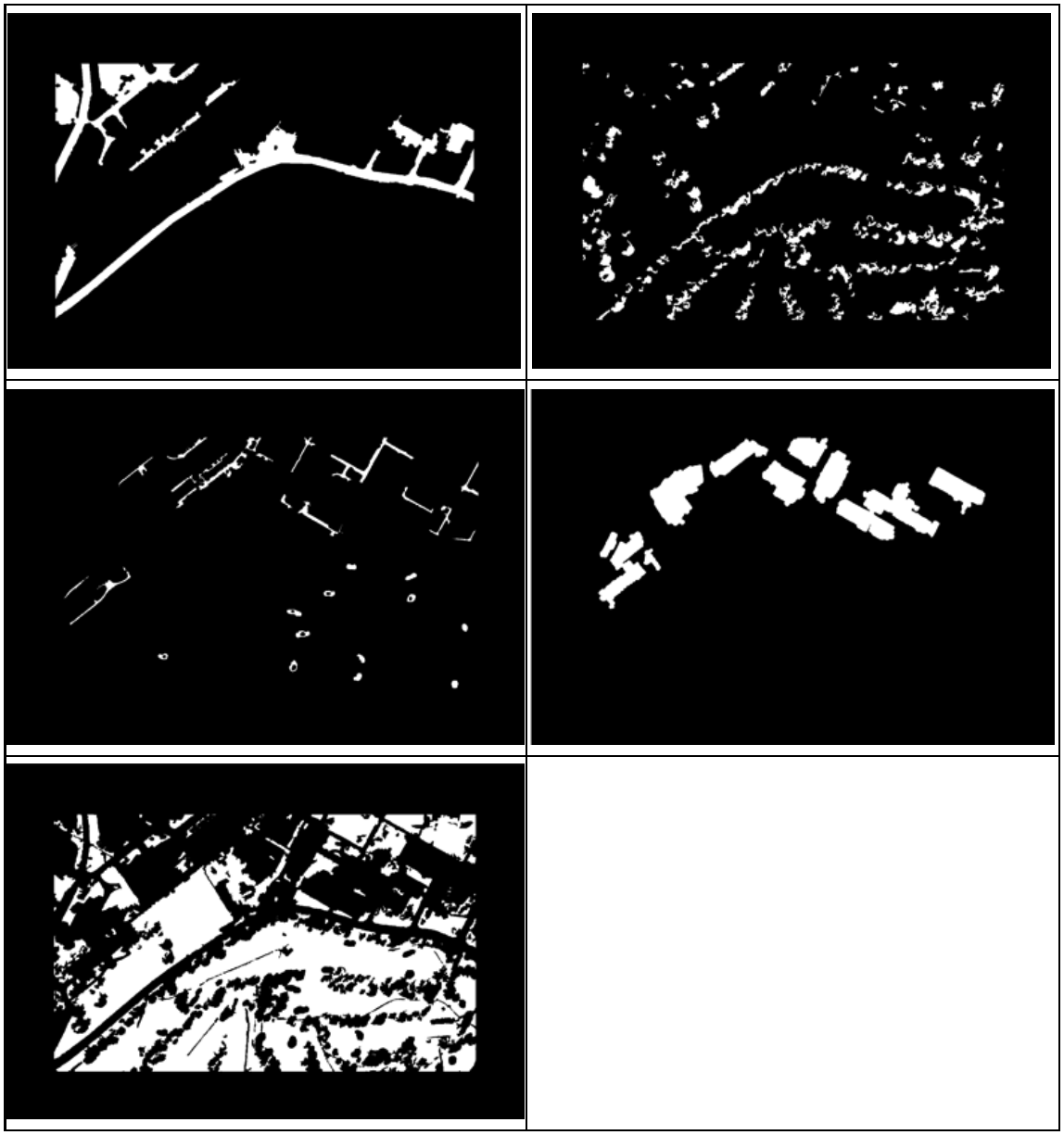


Figure 2.15 – Final segmentation of classes obtained after applying noise reduction and structural constrains for each class. First row: Road, grass; second row: tree, sidewalk; third row: shadow segmentation results obtained from Google Maps are shown for the map for campus road dataset.

2.14 Discussion and Conclusions

In this chapter we present our general framework for segmenting semantic classes from satellite images. With this method, we can segment multiple classes simultaneously and making use of the relation between these classes.

We show tree segmentation with a special focus on shadow detection for trees. Grass segmentation is also closely related to tree segmentation results. We present accurate grass segmentation results.

We also show a similar building detection solution that is tested on a large building dataset, giving good detection and segmentation results even in cluttered environments and ambiguous situations.

We show successful road segmentation with the help of road maps obtained from map providers and making use of other classes.

We present a novel crosswalk detection algorithm that can successfully detect cross walks from satellite images by making use of appearance, detected road locations, road orientations and sidewalk locations. The results of this work are very important for pedestrian navigation systems' safety and for realistic autonomous robot navigation.

In this chapter, we present a novel solution to complete sidewalk detection and segmentation problem that can even segment out sidewalks that are not non-observable on the map. Sidewalk detection method makes use of road and tree detection algorithms. Sidewalk detection results are very promising, where both pixel-wise and path-wise prediction rates are good with few false predictions. Predictions can further be improved

by improving the sidewalk and tree pre-segmentation routines. The results can be used to extract a complete sidewalk network of an area for an autonomous robot that plans its path, executes the plan, crosses crosswalks and reaches any given point in the map, all done in real-time and just by using live camera input and satellite maps downloaded at that instant. Most importantly, in this chapter, we present the bases of a common framework for segmenting multiple semantic classes from satellite maps.

Chapter 3

Hierarchical Semantic Hashing:

Visual Localization from Buildings on Maps

In this chapter, we present a vision-based method for instant global localization from a given aerial image. The approach mimics how humans localize themselves on maps using spatial layouts of *semantic* elements on the map. Instead of using pixel values or appearance-based features, we make use of semantic features that we extract from the image and satellite view. The main important point in our approach, instead of trying to locate a set of features appearing in the image, we efficiently store and query for the relative locations or layouts of the features to each other. With a geometric hashing scheme implemented on this relative layout, we are able to query a given image with very high localization accuracy in a very short query time. Our method relies on robust and consistently detectable semantic elements that are invariant to illumination, temporal variations and occlusions. In our experiments, we used buildings on a map as our semantic features. We show our experimental results for localizing satellite image tiles from a 16.5 km sq dense city map with over 7,000 buildings. We also show different performance analyses of our method on the full 284,017 buildings over 800 km sq area in the dataset.

3.1 Motivation

Visual geo-localization is the task of globally localizing a given set of images. In this chapter, we localize the given orthographic images by registering them to geo-referenced satellite maps. Here, we define our visual localization problem to be efficiently localizing a given aerial image in a large area without having prior information on the location. This particular problem is useful for flying systems to visually localize themselves in the absence of GPS signal just by using a downward facing camera. As it is, the problem is specific to outdoor localization, but the same idea can be extended to indoor localization if similar semantic features and maps can be built, where an important difference would the perspective projection in the images instead of the current simple orthographic top view projection.

For autonomous flight of Unmanned Aerial Vehicles (UAV) and rather smaller Micro Air Vehicles (MAV), accurate localization is a key element. A standard way of localizing air vehicles is use of regular GPS in conjunction with inertial sensors. The accuracy of the global localization obtained from these types of sensors can be low, especially in cases of low GPS signal quality or if small, lighter and cheaper sensors have to be used. Work on using Real-Time Kinematic Global Positioning Systems (RTK-GPS) for global Quadrocopter MAV localization has been presented in [21] and combination of visual and inertial sensors for the same purpose is given in [22]. Although SLAM and other feature tracking visual methods and inertial sensor based methods can generate accurate localization results for small areas or shorter flights, global registration based localization has to be performed to avoid the position and attitude drift that can occur in

using these methods. By registering downward facing camera images to geo-referenced satellite images, we achieve this goal without the use of expensive and heavy sensors onboard.

For visual localization, a simple pixel-wise image matching between aerial images and satellite maps of the area may not always be successful because of the appearance difference between these two images. Furthermore, the approach would be very infeasible in terms of its computational needs, where a simple mega-pixel aerial image with unknown scale and orientation that needs to be searched in a multi giga-pixel satellite image will require more than 10^{18} pixel comparisons. To put this in a clearer setting, searching a single image in a city with this method would take more than 3 years' worth of computation for a regular 3 GHz multi-core computer.

Appearance dissimilarity in aerial and satellite images can be solved up to an extent by using invariant visual features like image corners, area feature descriptors, texture, edge features and more complicated multi-level features learned through deep neural networks or convolutional networks. Use of these features no doubt would improve the runtime performance of the search too. Bag-of-words (BoW) [23]-[26] approaches try to solve the problem this way. Since searching for individual features would be very infeasible, BoW approach also groups features in images in to bags and improves uniqueness and search performance at the same time.

Our localization method consists of hashing and querying stages, Figure 3.1. Hashing stage is the offline process, where the known building locations on the map are used to build an efficient hash table structure for fast and robust querying. In the hashing step, we store relative positions and properties of buildings with respect to their close

neighbors. Querying stage is performed online and provides a single-shot global localization using an orthographic query image of a small area with buildings. Buildings on the query image are extracted by our proposed building detection method. Resulting buildings and their geometric layouts are used to obtain queries for the hash table from hashing stage. After the query, we receive votes for possible locations on the map and further checks can be done to determine the best matching location.

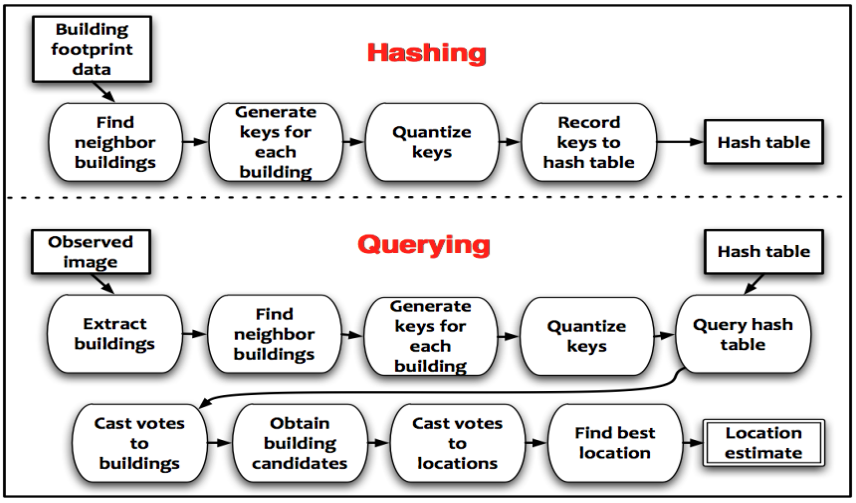


Figure 3.1 – Algorithm overview. Algorithm consists of two main stages: the offline hashing stage and online querying stage.

Other semantic features can also be employed for localization purpose when needed. For example, buildings are very well suited for localization in dense city environments, but for areas with not many buildings, roads, electric poles, hills, rivers and other man-made or natural structures may be used all with their own special detection, processing and matching needs.

Our contribution in this chapter are, a novel hashing scheme and a full pipeline around it to enable instantaneous localization in large city environments and using semantic features in images to perform fast and efficient localization.

3.2 Related Work

The broad category of computer vision methods that provide global localization using only a single observation (i.e. instantaneous), can use feature matching, bag-of-words (BoW) [26], map-matching (template-matching/correlation) [78] and image retrieval algorithms [79]. Feature matching and BoW algorithms come up with efficient ways to search features coming from query observations in a vast, prebuilt feature set of the environment. BoW can be thought of as attempting to capture higher-level groups of features that commonly occur together in the environment and by this; it takes a step closer to semantic-level features. On large urban maps, a lot of visually similar regions occur. This is a challenge for feature matching algorithms. Sliding window methods like template matching, suffer in performance with increased scale. They become even in orders of magnitude more computationally expensive when searched over possible rotations and scales. They are also more susceptible to appearance changes like view angle, lighting and seasonal changes. There is a large body of work on image retrieval algorithms (a comprehensive list is given in [79]), where the class of Content-Based Image Retrieval (CBIR) techniques to find the closest looking image to a given image can be employed for localization purposes. Unfortunately, many of the CBIR methods rely on the basis of the aforementioned methods and have similar shortcomings.

Semantic-level features are more robust against visual uncertainties in uncontrolled environments than local point features. Methods in [80] have focused on building semantic representations of indoor environments using the presence of informative objects. There are other approaches that focused on constructing semantic-level

information from low-level features [81] using supervised learning in order to perform image retrieval.

Methods for localizing camera images using street view images and satellite maps are proposed in [82] and the method in Chapter 4 [46] respectively. For querying, [82] uses SIFT features of street view images stored on a tree structure. Our localization method proposed in Chapter 4 [46] uses a Bayesian tracking framework to iteratively localize vehicle position by matching camera view to satellite map, but does not attempt to solve the instantaneous or initial localization problem.

One of the most relevant work is the method proposed in [83] for visual localization on satellite imagery using low-level local SIFT features. This work uses feature matching and a BoW-type approach to quantize the features. Unfortunately, matching performance is only evaluated for visually discriminative areas of the map and approach does not address disambiguating between similar regions for localizing in urban environments. The search is will suffer in performance as search and query images grow. The approach is also affected from scale and rotation since it has to search over multiple scales.

Map matching based UAV localization in a tracking framework is presented in [84], where image-to-image and image-to-map registration is performed iteratively. SLAM from downward facing camera has been affectively applied to the recently growing field of UAV localization [22]. These approaches are for rather small-scale environments and do not provide instantaneous global localization on maps.

Although Geometric hashing has not been directly used for general map based localization methods, it has been used as coarse global localization methods by several works in other scenarios: edge based indoor localization, where each room is a model

[85] and building-facade based 2D localization on maps [86]. The latter work uses hand-made ground truth of building facades on satellite images and matches them to buildings seen in images taken from a ground robot. Although this method is promising in its nature, it does not scale to larger areas nicely. Experiments in this work are performed on a small number of buildings and accuracy of the method depends on the accuracy of detailed hand-made façade data.

3.3 Semantic Geometric Hashing for Localization

In this chapter, we first describe the construction of the semantic geometric hash table that encodes the layouts of small neighborhoods of buildings on the map. We then describe at query time, given an observation image, how we localize this image by extracting the buildings from the image and finding the closest location to the query image using the semantic geometric hash table.

Semantic hashing is to build a hash table of semantic element layouts and querying this hash table with similar hash keys. For a map that we need to do localization on (e.g. a city), we extract or obtain the semantic features to build a hash table with their relative layouts as the key. In the query stage, we extract the same kind of semantic features from the query image and build the same layout keys to query the hash table. A hierarchical voting scheme first localizes individual buildings and then localizes the whole image from these individual building localization candidates.

We apply the idea behind geometric hashing to localization problem. Geometric hashing is a general technique for model-based object recognition, where a model with multiple 3D points can be identified by looking at the layout of these points. The main important properties of geometric hashing are its query time efficiency, its inherent scale, rotation and translation invariance and its robustness against missing data. Geometric hashing hashes a model by encoding a normalized coordinate for all its model elements using baselines generated from combinations of model elements themselves.

When designing semantic hashing, we had to take in to account real life implementation problems like uniqueness of normalized coordinates and size and loading factor of the hash table. We used some other distinctive features from semantic elements as a part of the hash key to increase the uniqueness of hash keys and altered the coordinate generation algorithm to reduce the number of hash entries generated for each semantic element. We also defined semantic element neighborhoods that correspond to the concept of models in the original geometric hashing algorithm. The rotation and scale invariance of geometric hashing proves to be very useful in querying an image with unknown scale and orientation. The robustness of geometric hashing to missing data allows us to obtain a good localization result even under non-perfect semantic element detections.

3.3.1 Constructing the Semantic Hash Table

Geometric hashing [87] has been proposed as a model-based efficient object recognition scheme that has later been used for many pattern recognition tasks including

3D shape recognition [88], pose estimation, hand-written image matching; generally by using point or line features [89], [90].

Unlike the traditional geometric hashing applications that use low-level point or line features in an image, we present an algorithm that makes use of higher-level semantic features in the scene. We hash the building centroid locations and building features such as the area ratios. This approach is more robust against noise or detection errors when compared to using low-level visual features in the image. Furthermore using semantic features enables us to match across different data modalities, e.g. matching satellite images to GIS building contour polygon data or other street maps data.

A *model* in geometric hashing is the set of feature points of a single entity to be detected in the scene. Performance of hashing degrades when the feature points to be hashed in the model increases too much. Thus, we chose to have each model to correspond to a building on the map and its K nearest neighbor buildings. For each building, we find at most K nearest neighbors inside a neighborhood R (Figure 3.2) to be the model for that building. The hashing step is where we store the geometric inter-relations (i.e. layout/structure) of the elements of these models. In the query stage, when one of the models is detected, this will suggest localizing the center building (key building) of that model. We build our hash table using ground truth locations and areas of buildings obtained from GIS data.

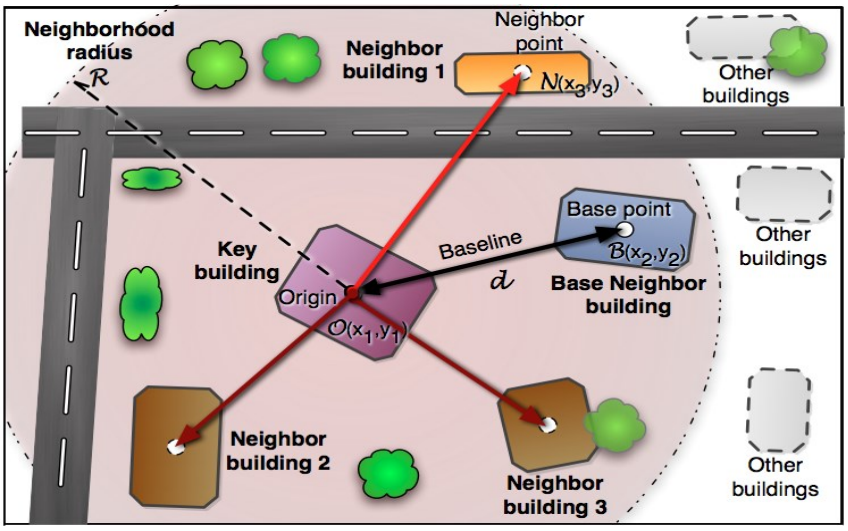


Figure 3.2 – Building hashing scheme. Hashing baseline is formed by the key building and base neighbor centers.

Algorithm 1 – Hashing

Input: buildings
hash_table ← []
for all *key_building* in *buildings* **do**
  (x_1,y_1) ← centroid(*key_building*)
  a_1 ← area(*key_building*)
  N ← neighbors(*key_building*, *max_neighbors*, R)
  for all *base_neighbor* in N **do**
    (x_2,y_2) ← centroid(*base_neighbor*)
    a_2 ← area(*base_neighbor*)
    baseline ← vector((x_2, y_2), (x_1, y_1))
    other_neighbors ← N – {*base_neighbor*}
    for all *neighbor* in *other_neighbors* **do**
      (x_3, y_3) ← centroid(*neighbor*)
      a_3 ← area(*neighbor*)
      (f_x, f_y) ← normalize((x_3,y_3), *baseline*)
      (q_x, q_y) ← quantize(f_x, f_y)
      q_{a13} ← quantize(a_1 / a_3), q_{a23} ← quantize(a_2 / a_3)
      hash_key ← [q_x, q_y, q_{a13}, q_{a23}]
      value ← [*key_building*, *base_neighbor*]
      hash_table[*hash_key*].append(*value*)
return: *hash_table*

Detailed steps of hashing buildings are given in Algorithm 1. To normalize the coordinates of neighbor buildings in the model, we take one neighbor building to be the base neighbor to form a baseline with the key building, Figure 3.2. Each neighbor becomes a base neighbor, where rest of the neighbors is normalized based on this baseline. Normalization is performed by considering a new coordinate system where the key building center is the origin (0, 0) and base neighbor center is the point (1, 0). Normalized coordinates of neighbor buildings along with the area ratios to key and base neighbor buildings are quantized to generate hashing keys for the model. Area ratios are used to generate more unique hash keys and like the normalized locations, this feature is also translation, scale and rotation invariant. Corresponding to each key, we note down key building number and base neighbor number in the hash table. The K nearest neighbor approach and the selected baseline with center point being fixed, reduces the per model geometric hashing complexity to $\mathbf{O}(K^2)$. In the original geometric hashing algorithm, for a model with K points, all possible point pairs in a model (all neighbor pairs) are used as basis, which leads to $\mathbf{O}(K^3)$ per model complexity. The complete hashing time complexity for our approach becomes $\mathbf{O}(nK^2)$, where n is the number of buildings to be hashed in the whole map.

Geometric hashing is inherently invariant to scale and rotation, due to its normalized coordinate system and it is also robust against occlusion and missing data points since multiple entries for a model is stored in the hash table and using a voting scheme. As a simplification, we chose to select the model (building location) with the maximum number of votes as the best localized model. As long as the number of votes per each

model stored in the hash table is close to each other, this simplification does not pose a significant problem to final localization.

3.3.2 Querying and Localization Steps

3.3.2.1 Automatic Building Detection from Satellite Images

In order to query the location of a given image, we need to extract the semantic features from the image for semantic hashing queries. By processing a given query image, we automatically determine building locations and sizes by employing a combination of machine learning and image processing techniques as explained in Chapter 2.

Although the query images can come from a variety of different sources like UAV's or aerial imagery, for our testing purposes, we chose to use satellite images for queries because of the availability of these images.

3.3.2.2 Hierarchical Querying

We localize the query image by employing a hierarchical querying approach. We use the buildings extracted from the query image and the previously built hash table. As detailed in Algorithm 2, we first independently localize every detected building in the query image.

Building neighborhoods are generated from the detected buildings. A similar baseline and key generation procedure is followed to generate a query key to the hash table. All the values in the hash table that correspond to the generated keys are candidate buildings for the query building. Using the key building and baseline neighbor counts in each table entry, probable orientation and scales votes for each candidate building are generated. The output of each single building localization step is the unique identifier of the best matching building on the map as well as the scale and orientation hypotheses for the query image.

We then combine these single building localization results that agree on image center locations to obtain final localization results as explained in Algorithm 3. We cast votes for image orientations, scales and center locations. Votes are quantized in order to compensate for mismatches due to individual localization errors and inaccuracies. The highest voted area is the localization estimate for the query image center. An improvement step can also be applied to this quantized localization estimate to further fine-tune it and get more precise localization results. Our hierarchical approach enables the method to be robust against missing or false building detections and possible incorrect localizations of individual buildings.

During testing, we take the area with the maximum number of votes as the estimated location of the query image. This is a hard localization decision. It is also possible to make a soft localization decision, which would examine the top k voted locations within the large-scale map. Later, these candidate regions can be examined more closely. In the results section, we briefly look in to possible improvement we can obtain from using the top-k locations approach.

Algorithm 2 – Locate Single Building

Input: key_building, hash_table
$(x_1,y_1) \leftarrow$ centroid(*key_building*)
$a_1 \leftarrow$ area(*key_building*)
$N \leftarrow$ neighbors(*key_building, max_neighbors, R*)
query_hash_table $\leftarrow$ []
for all *base_neighbor* in N **do**
  $(x_2,y_2) \leftarrow$ centroid(*base_neighbor*)
  $a_2 \leftarrow$ area(*base_neighbor*)
  baseline $\leftarrow$ vector((x_2, y_2), (x_1, y_1))
  other_neighbors $\leftarrow N -$ {*base_neighbor*}
  for all *neighbor* in *other_neighbors* **do**
    $(x_3, y_3) \leftarrow$ centroid(*neighbor*)
    $a_3 \leftarrow$ area(*neighbor*)
    $p \leftarrow [(x_1, y_1), (x_2, y_2), (x_3, y_3)]$
    $(f_x, f_y) \leftarrow$ normalize((x_3,y_3), *baseline*)
    $(q_x, q_y) \leftarrow$ quantize(f_x, f_y)
    $q_{a13} \leftarrow$ quantize(a_1 / a_3), $q_{a23} \leftarrow$ quantize(a_2 / a_3)
    hash_key $\leftarrow [q_x, q_y, q_{a13}, q_{a23}]$
    values $\leftarrow$ *hash_table*[*hash_key*]
    for all *value* in *values* **do**
      baseline $\leftarrow$ *value*
      building_id $\leftarrow$ *value*[0]
      building_theta $\leftarrow$ calculate_theta(*baseline, p)*
      building_scale $\leftarrow$ calculate_scale(*baseline, p)*
      $q_{theta} \leftarrow$ quantize(*building_theta*)
      $q_{scale} \leftarrow$ quantize(*building_scale*)
      building_key $\leftarrow [building\_id, q_{theta}, q_{scale}]$
      query_hash_table[*building_key*]++
match $\leftarrow$ {scale, theta, id} of max(*query_hash_table*)
return: *match*

Algorithm 3 –Locate Image with Hierarchical Querying

```
1:  Input: query_image, hash_table
2:  image_center_votes ← []
3:  query_buildings ← detect_buildings(query_image)
4:  for all key_building in query_buildings do
5:      match ← LocateSingleBuilding(key_building, hash_table)
6:      image_center_vote ←calculate_image_center(match)
7:      q_center ← quantize(image_center_vote)
8:      image_center_votes[q_center]++
9:  best_center ← {x, y} of max(image_center_votes)
10: return: best_center
```

3.4 Experimental Results

To test the semantic geometric hashing method we present, we used building contour data from Seattle GIS building data to construct the hash table. In these tests, locations of non-overlapping tiles from Google Maps satellite images with approximately 40 cm/px resolution are used for building detection and querying for localization. The test area with 7,000 buildings (shown in Chapter 2, Figure 2.10) is fully covered with 80 image tiles of size 1024x1024 px. We report on the accuracy of the building detection and localization stages and analyze the localization step under different parameter configurations.

The effects of the accuracy of building detection algorithm from Chapter 2 are also investigated by comparing the localization results of using detected buildings and ground truth buildings.

3.4.1 Semantic Localization Results

We considered tiles that have more than 80 buildings and used 18 neighbors for each model. With these settings, the overall localization accuracy we obtained with our localization algorithm is **91%**. In Figure 3.3, we further investigate the effect of the neighborhood size parameter.

To see effects of building detection error on the localization results, in Figure 3.3, we compare localization results for the detected buildings and the ground truth building locations. It is worth noting that difference in accuracy does not only come from incompleteness of building detection results, but also from building differences between 2009 GIS information and 2014 satellite images. Even with the missing and mismatching building locations (due to GT being acquired at a different time), our localization method still performs robustly and it accurately localizes the given query images.

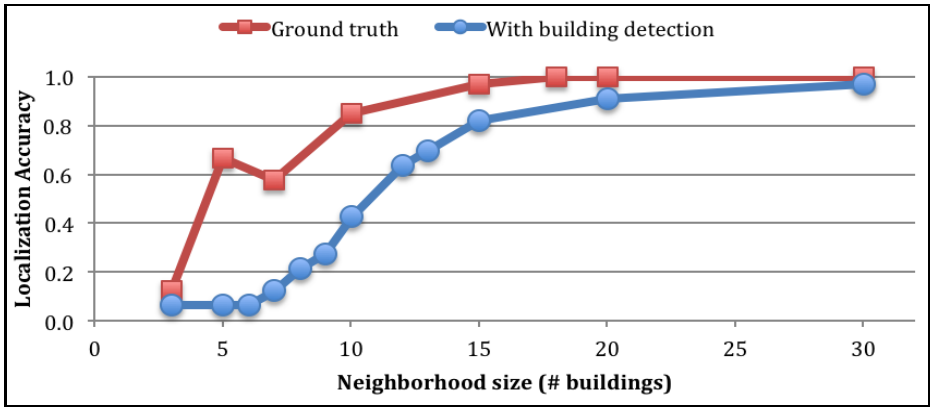


Figure 3.3 – Localization accuracy vs. model neighborhood size. Localization accuracy increases as the number of buildings hashed in a neighborhood increases.

In Figure 3.4, we report localization accuracy divided into *denseness* categories of tiles, where residential areas are denser than downtown areas. It can be observed that as

denseness increases, localization accuracy increases. This is both due to dense regions providing more semantic information and also building detection algorithm working better in the dense regions where buildings are smaller and rectangular.

We also explore how much accuracy we get if we look in to top k votes locations instead of picking the maximum voted location. Figure 3.5 shows accuracy vs. number of top locations considered for different neighborhoods. From these results, we see that it is highly probable to find the correct location if top 4 locations are investigated as opposed to using top voted location.

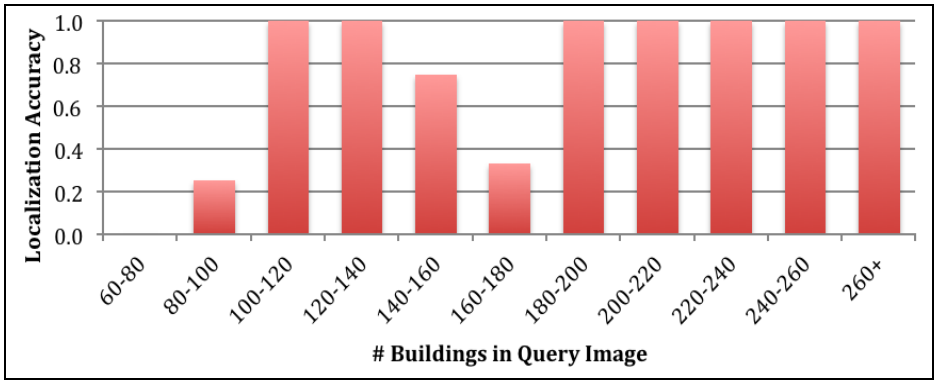


Figure 3.4 – Localization accuracy vs. the number of buildings in the query image. Although localization accuracy depends on multiple factors including building detection accuracy and uniqueness of the layout, in general the localization accuracy increases as the number of buildings in the query increases.

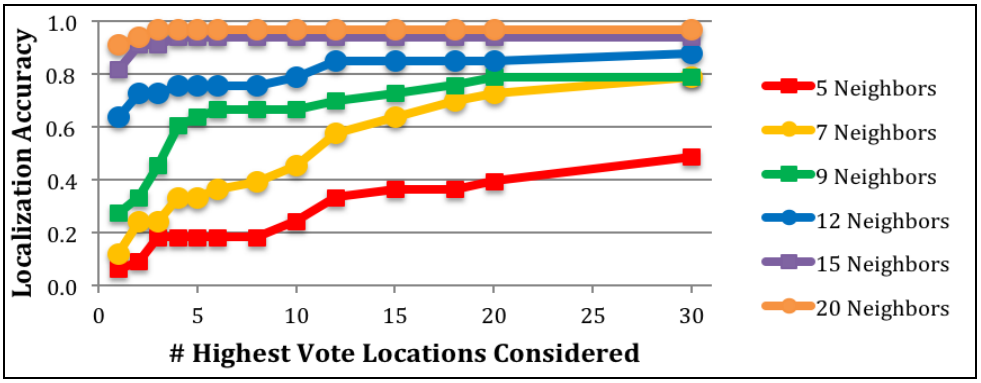


Figure 3.5 – Localization accuracy vs. number of top vote candidates considered, where each graph shows different neighborhood sizes. In this result the accuracy calculated by considering if the actual ground truth location is among the top voted locations. Localization accuracy increases as the number of buildings increases. With enough neighbors, close to perfect localization can be achieved by reevaluating multiple top voted location candidates.

3.4.2 Performance Evaluation and Scalability Results

To evaluate the approach we present from different performance aspects, we conducted various experiments on our large-scale dataset. For the localization part of the experiments, we used 7,000 query buildings in order to keep average number of buildings above a minimum number. Figure 3.6 (a) shows that the accuracy of our method is not adversely affected when additional buildings are added to the hash-table, hence proves an aspect of the scalability of the method. In Figure 3.6 (b) we show the rotation invariance of the results by using extreme cases of 90, 180 and 270-degree rotations, which was already expected from the rotation invariant construction of the method. Figure 3.6 (c) and (d) show respectively the linear and quadratic relations of total number of buildings and number of neighbor buildings to the hash-table size, again showing the scalability of

the method. In this test we hashed up to 250,000 buildings on a 3.0 GHz Intel i7 computer with 16 GB memory, where in-memory hash table size reached up to 6.5 GBs.

3.5 Conclusions

We present a complete vision-based instantaneous localization pipeline that has no manual steps, where we can localize a given single aerial image in a very large map, in a very short time. For semantic localization, we present a novel hierarchical semantic hashing scheme that is invariant to scale and rotation and robust to occlusions and missing data. Our method scales linearly with increasing number of buildings stored in the database. Our approach can be modified to work with arbitrary top view images and with other semantic landmark information in addition to buildings. We present results for using our method to perform efficient localization of satellite images on large-scale GIS maps. For experiments in an urban area with large number of buildings, method achieves high localization accuracy as long as sufficient buildings exist in the query image.

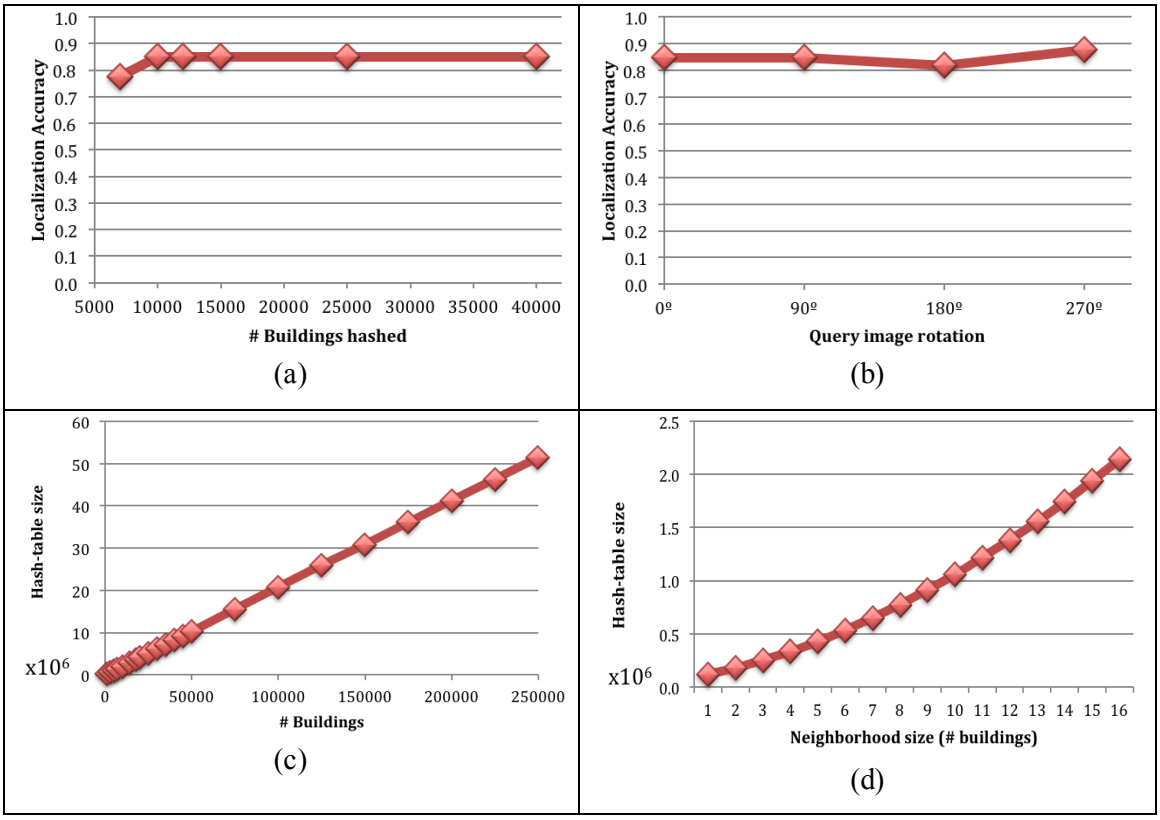


Figure 3.6 – (a) Localization accuracy vs. number of buildings hashed (queried with 7,000 buildings fixed). (b) Localization accuracy vs. rotation. (c) Total hash-table entries for number of buildings hashed. (d) Total hash-table entries vs. model neighborhood size.

In this chapter, we present an instantaneous visual localization method based on the semantic structures –namely building locations- found on maps. We use a form of geometric hashing that utilizes semantic information. We show that localization method is able to localize a single view within a large-scale map very efficiently, where the map covers 7,000 buildings in a portion of downtown Seattle with 6.5 x 2.5 km (16.5 km2) area.

Chapter 4

Visual Localization from Stereo Image Sequences

In this chapter, we present a framework to localize a vehicle by matching stereo camera images taken from the same vehicle to the satellite maps of the area. Our global vehicle localization framework combines stereo visual odometry, satellite images, and road maps under a particle-filtering architecture. It also generates 3D point cloud reconstructions and high-resolution top view images of the scene on top of accurate localization results. The framework focuses on the general vehicle localization scenario for urban and rural environments with the presence of moving objects without the use of global positioning system. The main novelties of our approach are using road maps, rendering accurate top views using stereo reconstruction, and matching these views with the satellite images in order to eliminate drifts and obtain accurate global localization. For vehicle localization, road maps are used to generate prior road probabilities. A sidewalk probability transfer function is used to calculate prior sidewalk probabilities for robot localization on sidewalks.

We show that our method is practicable by presenting experimental results for vehicle localization on roads and robot localization on sidewalks. We compare the results of our visual localization method to show that our method yields similar accuracy results

compared to GPS localization and superior localization results compared to visual odometry.

4.1 Motivation

After the keystone robotic competitions organized by DARPA; Grand Challenge in 2005 [49], and Urban Challenge in 2006 [50], the automotive industry accepted the possibility of fully autonomous driving. Since then, researchers have been working on the problems encountered in previous systems, improving their results and trying to reduce the extensive sensor load that was present in the competition vehicles. Promising work on stereo camera systems suggest that they might replace the functionality of radars, laser scanners, and other sensors in the coming years for similar autonomous robotics applications.

An important part of the autonomous driving problem is the self-localization of the vehicle. Although localization problem can be solved with the help of Global Positioning System (GPS) and Inertial Navigation System (INS) [27], [28]; there are some environments where GPS does not function well. Our framework is suitable for *GPS-denied navigation*, for example in urban environments, where reception is low or in the areas, where GPS signal is intentionally blocked or falsified.

We propose a framework that uses stereo camera images and freely available satellite and road maps to automatically obtain accurate global vehicle localization, pose estimation, and realistic dense 3D point cloud reconstructions of the road and its

surroundings. The proposed framework depends solely on stereo image data and does not require GPS or INS inputs; nonetheless, it can make use of this information if available.

The visual localization problem we are dealing with is to localize a moving platform using the images taken from its forward facing camera. The platform can be a vehicle moving on the road, a robot moving on city sidewalks or a person walking with a cellphone or a wearable camera like Google Glass or GoPro. Our localization problem can be to localize the platform as the images arrive or the video to be localized could already been captured and stored. Thus, our method can also be used to perform *offline localization*, where video is already stored and GPS information is not present.

We address the general problem of global localization with the presence of moving objects in the scene. Feature tracking based stereo Visual Odometry (VO) results are combined with matching results of image data to satellite maps, for obtaining global pose corrections. In order to register perspective camera images to orthographic satellite maps, we generate top view equivalent images from camera images using image transformations. We also include other priors to our probabilistic framework to improve localization. We employ RANSAC-based outlier rejection for robust ego-motion calculation in dynamic environments and define a Particle-Filtering (PF) framework to combine results of aerial image matching, road map following, and VO.

The framework is well suited for real-time implementation since no global optimization phase is performed and incremental steps do not need storing localization results from previous frames, except the very last frame.

This work introduces four significant contributions to the field. The first contribution is a complete vision based localization framework that can achieve GPS accuracy

localization for moving platforms. The second contribution is reconstructing accurate metric top view images of the road region from stereo images. The third contribution is the use of readily available road map images obtained from the Internet in a computer vision framework to enable global corrections in localization. The final one is the use of a multi-resolution approach to traditional block-based stereo depth estimation methods in order to attain denser depth images.

In this chapter, we first introduce the problem and list the related work and provide an overview of our framework. We present the components of the system, which are stereo estimation, map acquisition, visual odometry, top view construction, map matching, particle-filtering, and 3D reconstruction. Finally, we give results, conclusions, observations, and future research goals.

4.2 Related Work

Accurate self-localization is an essential part of a robot's navigation requirements. Researchers have built systems that navigate in urban environments and find their way by interacting with pedestrians [62]. A real world robot challenge is being carried out in Japan since 2007, where robots have to navigate autonomously, through a given 1-km walkway course [63]. The importance of this challenge is that the course is in a city center but not in a controlled environment. Navigating on sidewalks effectively is a crucial task for a robot that has to function in cities. Robots having sidewalk navigation capabilities can travel in cities to perform high-level tasks like guiding tourists or

assisting the visually impaired. Several systems have been proposed to help visually impaired [64], [65], focusing on hardware design, navigation and human-robot interaction issues.

Over the last decades, researchers have been proposing different outdoor localization methods that use various sensors. Among these, global positioning system (GPS) and inertial navigation system (INS) based methods [28], [66], landmark based Simultaneous Localization and Mapping (SLAM) methods [67], [68], and map-matching methods [46], [52], [53], [69] are the most common ones.

Stereo and monocular SLAM techniques (see [70], [71] for overview) are widely used in robotics applications [29]-[31] to localize a moving platform and at the same time build a feature map of the environment around it. Common SLAM algorithms track multiple visual feature points (landmarks) and estimate location of the robot based on the estimated locations of the feature points. Although over the recent years very efficient SLAM techniques have been devised, SLAM still may require a lot of memory and computation power and most implementations are prone to drift kind of errors. Visual odometry [34], [59], which estimates the relative pose of a robot at each frame using stereo visual features, are also similar to SLAM. To our knowledge, a SLAM technique that incorporates registration to an already existing metric map to remove the drift does not exist.

In GPS-based navigation, absolute location of the robot can be obtained with a bounded error. However, due to weather and environmental conditions (e.g., urban canyons) GPS localization performance can be degraded.

Map-matching based localization algorithms exist, where satellite or aerial maps of the area are matched to onboard camera images. There have been an increasing number of studies in this topic in the recent years. The study in [69] has used iterative corresponding point (ICP) algorithm to match lane markings on aerial maps with features on camera images. A similar work in [52] used the homography between camera and satellite maps in order to warp camera images to their top view equivalents and to match on-road features, where a feature map of the environment is prebuilt. In this work, we obtain accurate top view equivalents of onboard stereo images by projecting the 3D point cloud of the scene observed by the camera and compare top view images to satellite images in order to eliminate the need for computationally expensive feature-matching steps. We automatically obtain precise localization for a vehicle or robot that is moving without the use of GPS, INS or wheel odometry.

4.3 Framework Overview

Our proposed framework combines stereo VO results with satellite and road maps under a particle filter architecture in order to obtain globally corrected vehicle poses, high-resolution top view images, and 3D reconstructions.

Algorithmic flow chart of the system is shown in Figure 4.1. Stereo camera system is used to acquire stereo image pairs observed from the road. Depth images and VO are calculated from these stereo pairs. Depth images are obtained using stereo disparity estimation (Figure 4.8 (c)), and then they are converted to frame point clouds using

projective geometry (Figure 4.8 (d)). Top views are obtained from the point clouds by a *voxel* (3D volumetric pixels) projection (Figure 4.8 (e)). Satellite and road maps of the area are downloaded from the Internet. These maps are processed to get road or sidewalk maps as explained in Chapter 2 and later converted to road or sidewalk prior probability maps using distance transform. Calculated VO transformations, satellite maps, prior probability maps, and constructed top view images are integrated into a PF framework as observations.

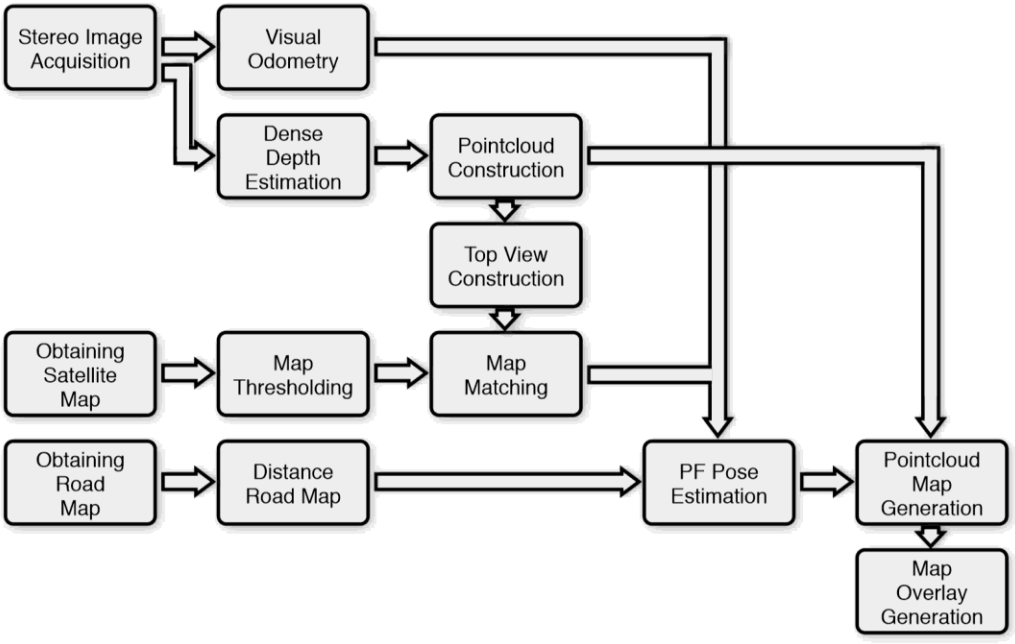


Figure 4.1 – Framework flow chart for obtaining PF poses, point clouds and map overlays from stereo images, satellite and road maps.

4.4 Stereo Image Acquisition

In our system, stereo image pairs are acquired by the onboard BumbleBee2 [58] camera system, which is parallel to the ground and facing the front of the vehicle (Figure

4.2). Stereo images acquired by the stereo camera are rectified using the intrinsic and extrinsic parameters of the camera system (Figure 4.8 (a), (b)). In Figure 4.2, stereo image frames 1 and 10 are displayed from the point of view that they are acquired from the vehicle. These images are shown parallel to the camera sensor plane. Satellite map of the area and the aligned top view reconstruction image are shown on top of each other.

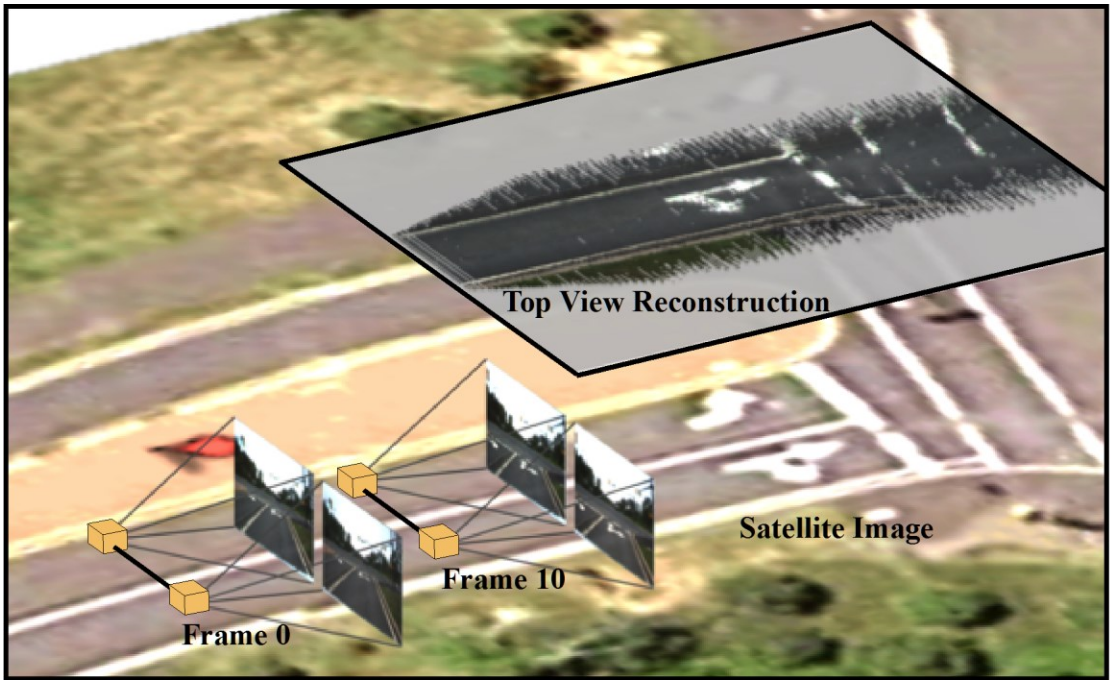


Figure 4.2 – The setting for framework elements. Stereo image pairs number 0 and 10, satellite image of the area and top view construction of the environment obtained from stereo images.

4.5 Stereo Depth Estimation Using Multi-Resolution Semi-Global Block Matching

Here we introduce a multi-resolution approach to available stereo depth estimation methods to obtain a denser depth image of the ground region. Over the past decade,

stereo vision has been one of the rapidly developing areas of computer vision. Real-time dense stereo reconstruction of high-resolution images has been recently made possible by better exploiting real-world constraints, making use of new optimization techniques, and taking advantage of new CPU/GPU architectures [54]-[56].

Semi-Global Block Matching (SGBM) [54], [57] disparity estimation algorithm is an efficient and accurate block-based stereo disparity estimation method, which makes use of mutual information and 2D smoothness constraints to obtain disparity images from rectified stereo image pairs. The resulting dense disparity image contains sub-pixel-accuracy disparity values corresponding to the pixels in the stereo image pair (Figure 4.8 (c)).

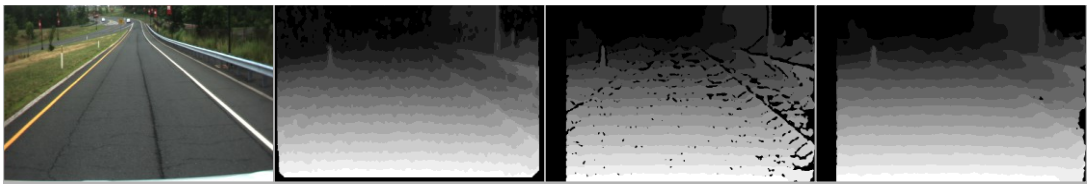


Figure 4.3 – Stereo depth estimation methods. (a) Left camera image. (b) ELAS disparity (c) SGBM disparity. (d) MR-SGBM disparity. MR-SGBM gives denser results with least errors in road regions.

In our work, in order to obtain denser depth image, we created a multi-resolution variant of the existing SGBM algorithm. In our approach, disparity estimations on lower resolution images are used to fill in the gaps of the disparities of higher resolution images. Disparities for each image pair is found at four different resolution levels and combined to obtain the final disparity image. This process can be parallelized since calculation for each level can be performed independently. We call this approach Multi-Resolution Semi-Global Block Matching (MR-SGBM). This method gives denser results

compared to SGBM and less noisy results in the road regions when compared to Efficient LArge-scale Stereo (ELAS) [55], another state-of-the-art dense disparity estimation method. Comparison of these depth estimation methods is presented in Figure 4.3 and comparison of the top view images generated from each method is given in Figure 4.7.

4.6 Visual Odometry and Preliminary Pose Estimation

In order to estimate incremental motion from the given stereo images, we use a stereo VO system, where rectified stereo image pairs are fed and incremental 6-DOF vehicle poses are obtained at each frame. We made use of the VO algorithm proposed and implemented by Geiger et al. [59]. This algorithm matches features from two consecutive stereo image pairs in a circular fashion among and across image pairs. The algorithm also reduces the number of features and uniformly distributes the selected features throughout the image using spatial bucketing. Algorithm then estimates the vehicle pose using a RANSAC outlier rejection scheme by re-projecting rest of the points using the estimations calculated from randomly drawn feature point sets. On top of everything, the algorithm refines the incremental pose estimations using a Kalman filter with constant acceleration motion model.

This VO algorithm is also suitable for real-time operation and one of its main advantages is its robustness against moving objects in the scene. The results of stereo VO are more accurate than MonoSLAM [31] and Scene-Flow [60] techniques as a consequence of its knowledge of accurate 3D locations of each feature point in the scene

through matching of feature points in stereo. Nevertheless, still the incremental calculation of vehicle poses in VO may bring substantial drifts in the long run. There exists no global correction in VO framework to correct these kinds of drifts. The large overall effects of accumulation from small (less than 1° in rotation) estimation errors at each frame are shown in Figure 4.4. In the following sections of this chapter, we show how to fuse VO results with satellite map matching to correct these drifts. Corrections will be done only for x and y position and z rotation corresponding to the 3-DOF part of the whole pose and rest of the 6-DOF pose is omitted.

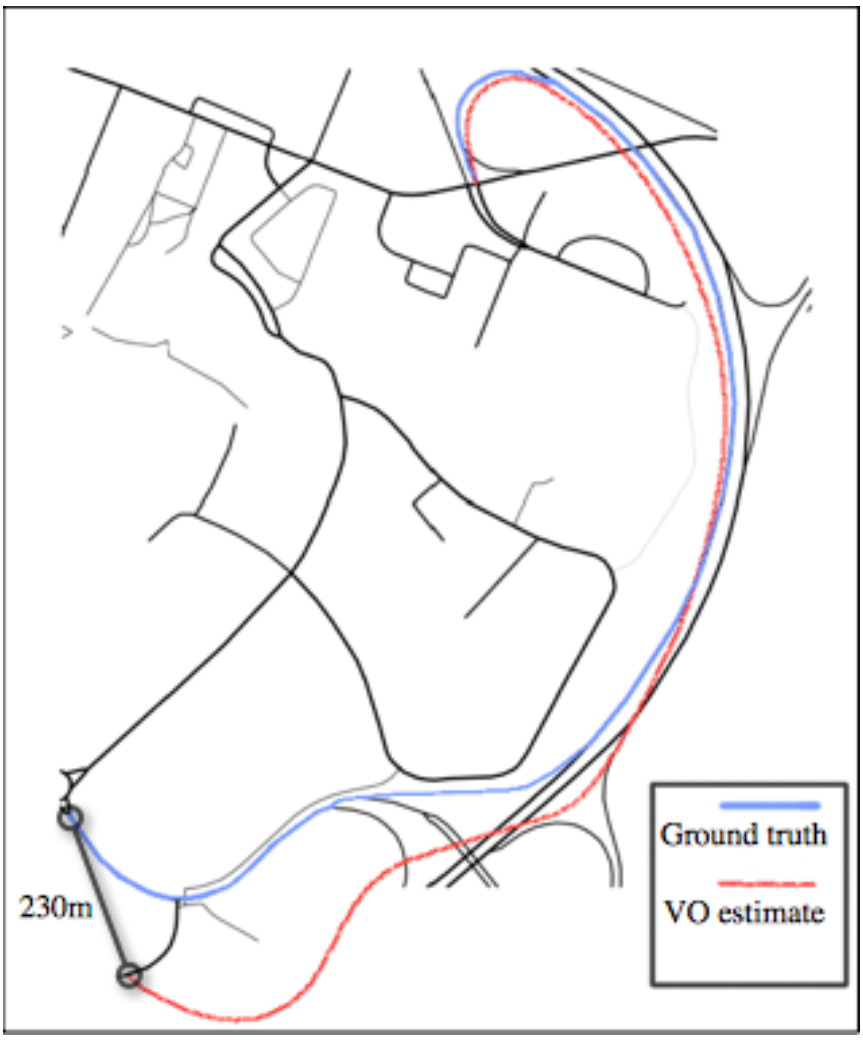


Figure 4.4 – VO estimated path and ground truth shown on road map. Estimated path drifts from the ground truth due to small estimation errors in incremental rotations.

4.7 Localization Priors

In addition to information coming from other sources, for various applications we can also make use of domain specific knowledge in our localization framework. If we know that we are localizing a vehicle travelling on the roads, we can make use of road

locations. On the other hand, if we know we are localizing robot that is travelling on the sidewalks; we can make use of sidewalk and crosswalk locations on the map. Below, we investigate these two possible scenarios in more detail and explain how to make use of this information in localization.

4.7.1 Road Probability

After performing a detailed road segmentation step as described in Chapter 2, or just by using available road maps of the area, we can generate a roads probability distribution map. We propose using a Gaussian distribution that has probabilities that decay exponentially by distance from known segmented road locations in order to incorporate possible segmentation errors or unexpected road thicknesses. It is also important not to assign probabilities close to zero to any pixel, because otherwise the road prior will prohibit us from successfully localizing a vehicle that goes out of the segmented road network. We assign a small constant probability to road probability values that are below a certain threshold.

4.7.2 Sidewalk Probability

To generate the probability distribution of sidewalk pixels on the map, we propose a probability transfer function that takes the segmented road network map and generated the sidewalk probability map (Figure 4.5, Figure 4.6).

In order to build this probability transfer function, our first assumption is that sidewalks are usually next to and parallel to the roads in cities. Our second assumption is that if there is a road in a location, there cannot be a sidewalk at the same location. There are exceptions that have to be addressed for these assumptions; first, in areas closed to traffic, sidewalks may exist without the existence of roads; second, usually there are no sidewalks next to highways; and third, if pedestrian crosswalks have to be considered as a part of sidewalks, their locations usually coincide with road regions.

Based on our assumptions and the exceptions we mentioned, we define the *probability distance transfer function P*.

$$P(d)=\begin{cases} th_{lowest} & |d| < w_r \\ e^{\frac{|d|-w_r-d_{s2}}{s2}} & w_r \le |d| < w_r + d_{s1} \\ 1 & w_r + d_{s1} \le |d| < w_r + d_{s2} \\ e^{-\frac{|d|-w_r-d_{s1}}{s1}} & w_r + d_{s2} \le |d| < w_r + d_{s3} \\ th_{lowest} & w_r + d_{s3} \le |d| \end{cases} \quad (1)$$

$$s_1=\frac{d_{s1}}{\ln(th_{lowest})} \quad s_2=\frac{d_{s3}-d_{s2}}{\ln(th_{lowest})} \quad (2)$$

P gets distance to the road center, *d*, and returns the related sidewalk probability (Figure 4.5). Constants are half of expected width of the road region, w_r, and d_{s1}, d_{s2}, d_{s3} expected sidewalk start, sidewalk end and no information zone distances, respectively. In the road region, sidewalk probability is constant and equal to th_{lowest}, corresponding to having a small probability for the robot being on the road. This region is reserved for road crossing scenarios. Between d_{s1} and d_{s2} distance from road, sidewalk probability is one. Majority of sidewalks are expected to remain in this region. Before d_{s1}, and after d_{s2}, probability attenuates exponentially with the distance. After d_{s3}, sidewalk probability is a

small constant equal to th_{lowest}, corresponding to no information coming from the road map. This region is reserved to give equal probability to inner sidewalks that are not related to the road network. These constant vales are determined by experimentation as d_{s1}=0.4 m, d_{s2}=1.0 m and d_{s3}=3.0 m. w_r is the half width of the road given in the road map. In order to maintain continuity of the transfer function at each point, attenuation sigmas s_1 and s_2 have to have the values given in Equation (2).

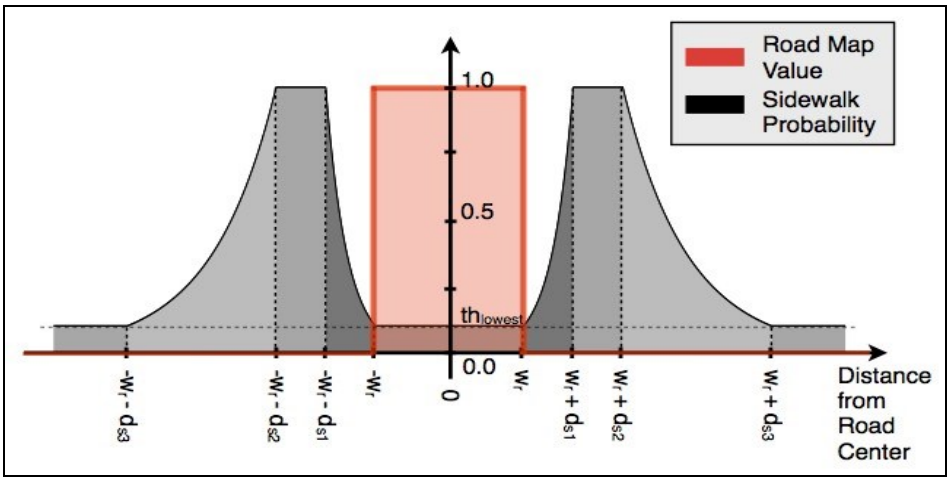


Figure 4.5 – Probability distance transfer function *P*. Road map values and corresponding sidewalk probabilities for road cross-section.

In order to obtain the prior sidewalk probability value for a map location, we find the closest road distance to that point and apply *P* to this road distance (Figure 4.5 and Figure 4.6). We call *P*, the *sidewalk probability transfer function*. Closest distance to roads for each map point can be effectively evaluated using *image distance transform*.

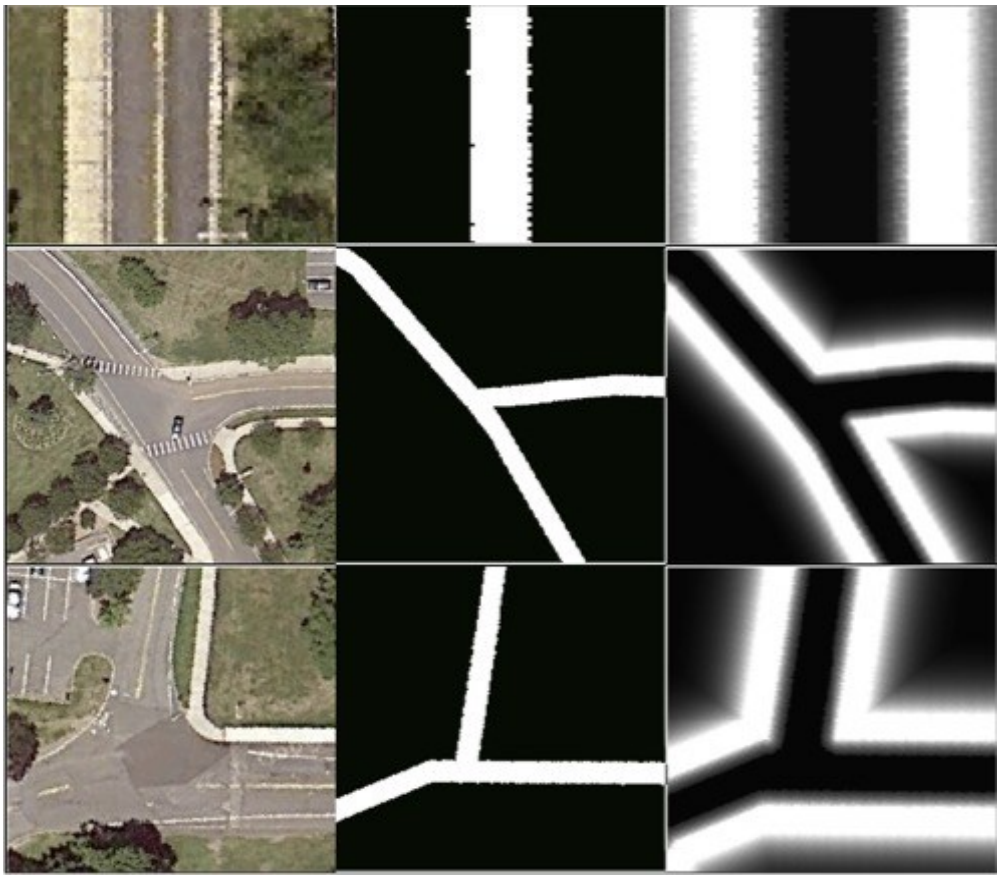


Figure 4.6 – From left to right: Examples of satellite maps, corresponding road maps and estimated prior sidewalk probabilities.

4.8 Top View Reconstruction from Stereo Depth Images

At this step, we build an accurate top view of the scene that is in front of the vehicle using only the acquired camera images. Noda et al. [52], suggests a method to build per-frame top view images by transforming the camera images using a (projective) transformation. Although this method generates dense top view images, these images

show large distortions as the objects move away from the camera and the method implicitly assumes the entire road region lies on a single plane.

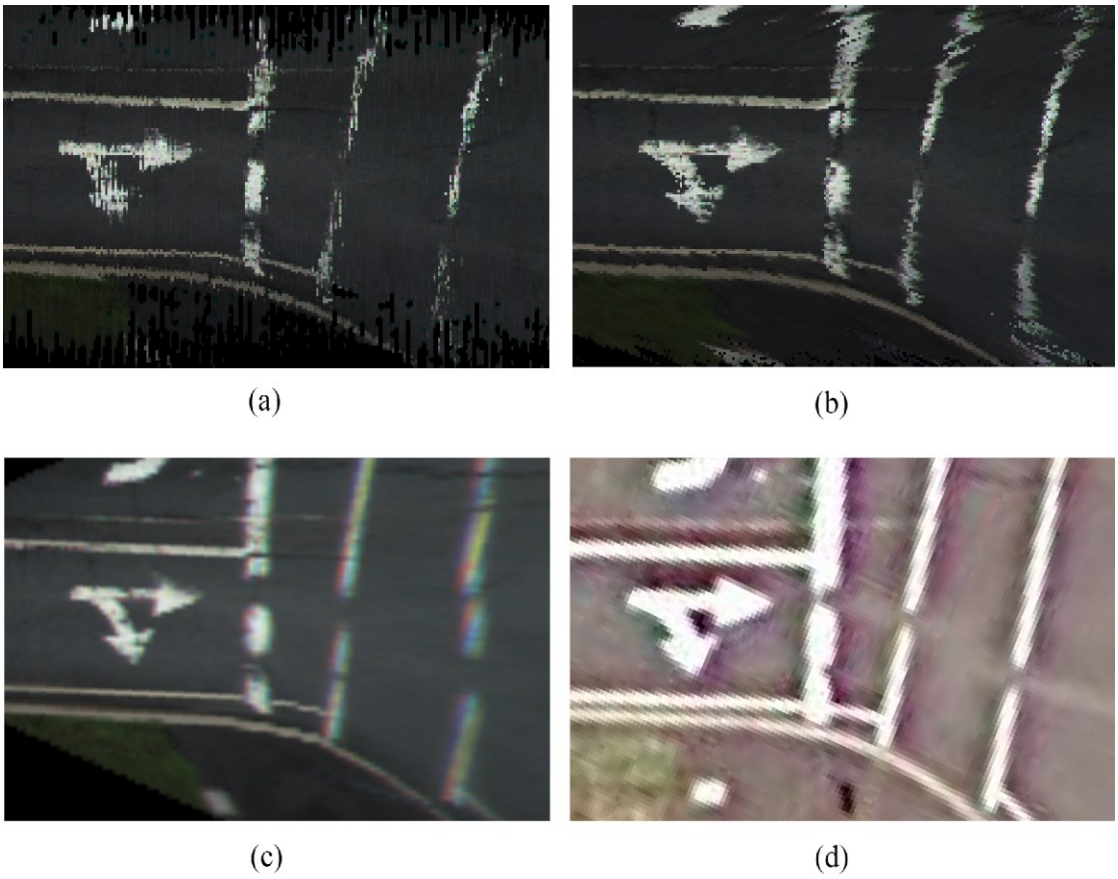


Figure 4.7 – Top view reconstructions; MR-SGBM (a), ELAS (b) and fixed homography (c). Satellite image (d). Distortions for distant objects can be observed in ELAS and fixed homography top views.

In our approach, to avoid these distortions, we do not follow the single-plane/single-homography assumption. We build a 3D point cloud of the scene seen by the stereo camera, and then project these points to a top view image. We crop the points that are above the camera level that would correspond to side or bottom faces of high objects (e.g., trees over the road). This method of building top images avoids distortion of distant objects and allows capturing more accurate top view for non-planar (curved cross-

sectioned, hill, downhill, bumpy, etc.) road regions and off-road objects that do not lie on a similar plane with the road.

4.8.1 Point Cloud Generation

A colored point cloud of the scene is constructed using the calculated disparity and color of each pixel. For consistency in the pixel color values, we chose to use the color value coming from the left image. Colored point cloud consists of colored voxels that correspond to each image pixel that has a valid disparity value. Metric coordinates of each point in the point cloud are calculated using epipolar geometry and triangulation.

4.8.2 Top View Projection

Stereo scene top view generation takes 3D scene estimated for a single frame pair, and projects it to a top view image. When the camera pose is known, top view projection is to project all pixels to the satellite image plane. In the projection, moving objects or fixed objects that cannot be observed from satellite image should be removed. Even if the stereo case objects that do not belong to the ground plane, they still can be projected to top view successfully. It is simpler to limit the projection to objects only on the ground plane. This approach, only using pixels from ground plane, also eliminates some of the moving objects, such as cars and pedestrians in a street scene. Otherwise, moving objects can be removed by analyzing the movement of points on them over multiple frames.

Figure 4.8 – Various frame examples for left (a) and right (b) stereo images, estimated disparity image (c), single-frame top view reconstruction (d) and combined top view image (e). Current pose and location of the robot are marked on the top view images.

By projecting the voxels of the point cloud on to local x-y plane where the local origin is taken as the focal point of the left camera, a single-frame top view image is generated for each frame (Figure 4.8 (d)). At this step, to get rid of points belonging to objects taller than our vehicle, only points that are below the horizon are projected to the top view image. Normally, points above the horizon can belong to side or bottom faces of tall objects (e.g., leaves of a tree or sidewalls of a building). Because of its situation, the vehicle cannot observe the top faces for these objects and thus the voxels above horizon coming from these objects should not contribute to the top view. There is an exception for this case; whenever there is an oblique surface higher than the vehicle, it can still be observable by the vehicle and contribute to the top view image. We do not address this case since it is not very common for our localization scenario. Omitting this case does not reduce the localization performance for inclined paths, since they fall under the horizon when they get closer to the camera.

We compare satellite image and results of several top view reconstruction methods in Figure 4.7. MR-SGBM has more unknown pixels than the other two methods, but it gives the most exact results (Figure 4.7 (a)). ELAS and fixed homography results are fully dense, but have more distortions in distant parts of the image.

4.8.3 Top View Integration

In order to obtain denser top view images, several consecutive top view images are combined using the estimated initial VO transformations for poses of each top view image. Dense combined top view images are obtained by combining individual top views of past frames and coming frames. At this step, we do not worry about the possible small errors in the VO estimations, since we are only building multi frame top view images using few (5 to 20) frames and the error accumulation will be negligible. Taking the origin as the first frame camera location in the top view image, we blend multiple single-frame top view images, into their appropriate locations and poses on this local coordinate system (Figure 4.8 (e)).

A stitched top view image, which covers enough area to be matched with satellite image, can be generated using this method. For each new frame, this stitched image can be extended to include the new frame excluding one frame from the beginning.

We also employ an alpha-beta filtering scheme to determine the pixel values when combining overlapping pixels of multiple top view images. Each pixel observed for a location is averaged with the existing value of the pixel. This way, newer observations

have more influence on the final value, which are pixels closer to the camera when the vehicle is moving forward.

4.9 Map Matching

After preparing the top view images, we can match them to satellite map regions to localize our platform for each frame. This matching can be done using directly pixel color comparison or specific matching features in the top view and satellite image. Features common in both images has to be selected carefully, since all types of features will not appear on both images in the same way, if they appear in both at all.

Point features detected from noticeable corners in the images are widely used in many computer vision algorithms due to their ease of detection and efficient matching methods. However, in top view to satellite image matching, not many unique features may exist in the either of the images to match. A simple example would be a highway road scene seen from above would have pixels belonging to the road, grass on the road side of the road, and some road markings. If the road markings are continuous lines, then there exist no detectable or matchable point features in this image. Use of intensity edges would lead to good and accurate matching in the road markings. Use of direct color matching will further improve the matching by differentiating the road pixels and grass pixels and at the same time successfully matching the road markings. Furthermore, if point features were detected on the top view image, any discontinuities occurring due to the problems in the top view generation process will be detected as a point feature. When

point features are detected on the satellite image, most of the point features will appear on the trees due to their textured look from above. However none of the trees will be matchable to the built top view images, since treetops do not appear in the top view images built from viewpoint of the camera. Because of these reasons, throughout our work we use edge matching and color matching methods, which are explained in detail below (Figure 4.9).

Searching for an area on the map that is similar to the current top view can be costly due to large map area and the three-dimensional search space (x-y locations and heading angle). Instead of search, edge-segmented images for top view of the current location and the map portion observed from each proposed possible pose are compared using chamfer matching method to obtain a matching score for these poses. Probable poses for a given time instance are calculated using the particle-filtering framework that we introduce in the next section. Edge chamfer matching score normalized to [0,1] interval is used as the map matching observation likelihood. When comparing two binary images, chamfer matching returns a result that is robust against small translational, scaling and rotational inconsistencies and pixel errors.

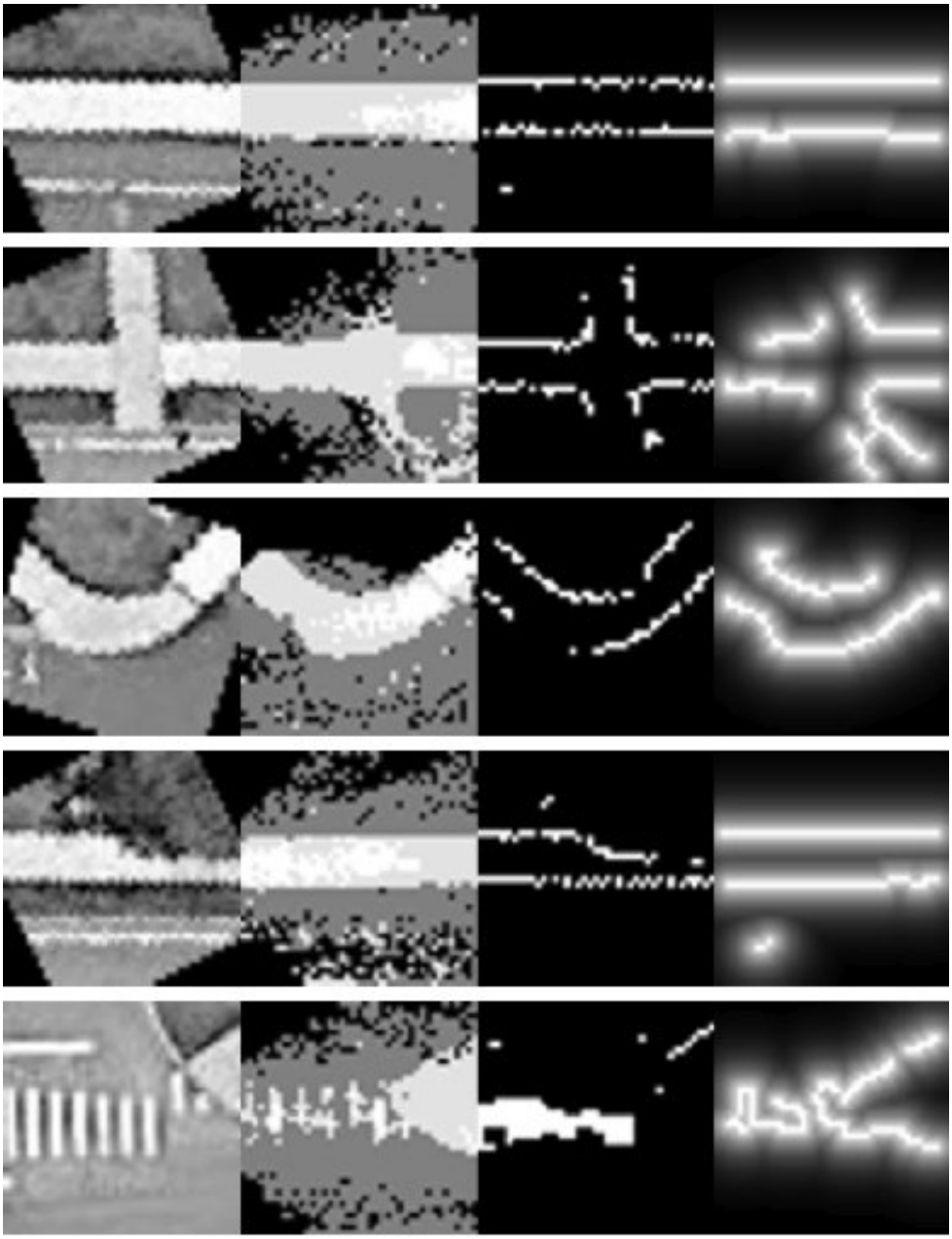


Figure 4.9 – Map matching results. From left to right: Satellite image, top view image for best matching robot pose, edge-segmented satellite image and distance transformed edge-segmented top view image for best matching robot pose. From top to bottom: Selected high matching scored examples for regular sidewalks, sidewalk crossings, turns, occluded satellite views and road crosswalks. Alignment is shown as the matching process suggests.

After the orthographic matching to the top view image, an optional more accurate *perspective-matching* step can be applied. This step is explained in detail in Chapter 5 for the monocular localization case. In the perspective-matching step, possible satellite map regions are transformed to the camera view using the inverse of the transformation used in the top view generation. Camera image and the transformed satellite images are compared in the perspective image domain to find the best matching location among the map regions. Perspective and orthographic matching compares the same regions with different transformations applied to them. The main difference is, in orthographic matching all pixels are compared with the same importance, whereas in the perspective matching the closer pixels have more weights in the comparison due to the perspective effect. Using the perspective matching, the localization accuracy can be as small as the map resolution or even more with sub-pixel accuracy. We use perspective matching to well satellite images to camera images so that we can use the map information for image segmentation and labeling purposes.

4.10 Vehicle Model, Particle-Filtering and Final Pose Estimation

In order to obtain precise localization results, we combine all our observations under a particle-filtering framework. At each frame, we carry particles that correspond to probable localization states. Each particle holds the x-y location (m), heading (degrees), tangential speed (m/s) and weight of that particle. At each step, using S*equential Importance Sampling* (SIS) [73], new particles are sampled from the old ones with giving more samples to higher weight particles. This way, the underlying probability distribution

is sampled finer around high weight particles and they have more chance of survival in the future frames.

Particle weights are calculated based on the likelihood of the particle state with current observations. Likelihood of an observation with a particle state gives the probability that this observation has been made from that state. Observation likelihood is calculated differently for different types of observations. Final particle weight is calculated by multiplying each individual observation probability for that particle.

Visual odometry observation gives the estimated relative transformation between current frame and the previous frame. This 6-DOF transformation is converted in to angular representation to obtain relative heading and speed. In order to find visual odometry observation likelihood, these observations are compared to particle headings relative to previous frame and particle speeds. Throughout the framework, visual odometry observations for altitude, roll and pitch are assumed correct, since no global observations can be made for these quantities within the setting of this framework.

Based on the application, sidewalk or road prior probability values for each particle location is used to modify the observed weights of the particles. This observation mainly helps to fix incremental errors in rotation estimation of visual odometry.

Map is matched to top view images by comparing edge-segmented versions of the scaled top view image and the satellite map area that should be observed for the given particle state (Figure 4.9). This observation helps fix translational and rotational errors in visual odometry. If an area with features along the direction of motion (i.e. lane markers on the road or edges of sidewalks), matching will only contribute in fixing the rotational error, since the appearance of map regions with the same heading might be very similar.

If the observation contains areas that are distinguishable in their appearances (e.g., crosswalks, sharp turns, stationary objects; see Figure 4.9), this will help fix both translational and rotational errors.

We combined all observations coming from different sources and system states under a Bayesian tracking framework. Particle-filtering is a very powerful Bayesian framework where multiple hypotheses of the system states are maintained throughout the tracking process and Gaussian/non-Gaussian and potentially multimodal probability densities can be estimated with low computational power.

Each particle holds a hypothesis of vehicle location, pose and speed. Particle state is (x, y, θ, v) where x and y represent the position of the vehicle in meters with respect to a fixed origin. In the PF, local Euclidean coordinate representation of the map has been used for its simplicity. Parameter θ denotes the absolute heading of the vehicle in degrees with respect to North direction of the map. Parameter v corresponds to the linear/tangential speed of the vehicle in the direction of the heading with unit of m/s.

Each frame's localization distribution is calculated from previous frame's distribution, the motion model, observations from that frame and the map matching scores. Particle filter resamples the particles with importance resampling according to their weights. We use a constant speed motion model that simulates the particles to move with a constant speed between two frames. A particle holds the location, orientation and speed information for a localization hypothesis.

In particle filtering framework, there is a compromise between computation time and accuracy in selecting the number of particles to use. As more particles are used, the underlying probability distribution can be modeled more closely and accuracy will

increase, but the computational power needed will be increased. Our computation time is linearly proportional to number of particles, since the most demanding map matching step is performed separately for each particle. However, since each particle is independent, this process can be easily parallelized.

4.10.1 Prediction and Resampling

A two-step process is used to update the particle filter. New particles are resampled from the current ones, where the number of new particles generated from each current particle is proportional to its weight and the total number of particles is fixed. Each new particle attains a state that is a sample from the probability distribution of the next noisy state of the current (parent) particle according to the vehicle state transition model. Equation (3) shows the noiseless constant speed vehicle state transition model. Equation (4) shows the noisy vehicle transition model where $\tilde{\theta}$ and $\tilde{v}$ denote the heading and speed noise which are modeled as zero-mean Gaussian distributions.

$$\begin{pmatrix} x \\ y \\ \theta \\ v \end{pmatrix} \rightarrow \begin{pmatrix} x + \Delta t \cdot \cos(\theta) \cdot (v) \\ y + \Delta t \cdot \sin(\theta) \cdot (v) \\ \theta \\ v \end{pmatrix} \tag{3}$$

$$\begin{pmatrix} x \\ y \\ \theta \\ v \end{pmatrix} \rightarrow \begin{pmatrix} x + \Delta t \cdot \cos(\theta + \tilde{\theta}) \cdot (v + \tilde{v}) \\ y + \Delta t \cdot \sin(\theta + \tilde{\theta}) \cdot (v + \tilde{v}) \\ \theta + \tilde{\theta} \\ v + \tilde{v} \end{pmatrix} \tag{4}$$

4.10.2 Weight Update

Weights of the new particles are calculated based on their agreement with the current observations. The weight of a particle is proportional to the product of the probabilities of each observation coming from a different source, belonging to that particle. In Equation (5), we show the weight calculation formula for new particle i at time step t where particle j at time step $t-1$ is its parent. Here, $\Delta\theta$ and Δd show how well the particle state is supported by the incremental VO observations $\Delta\theta^t_{observed}$ and $\Delta d^t_{observed}$; $I_{RD}(x,y)$ is the value of distance transform image for the binary road map at position (x, y). I_{MDT} is the distance transform of the thresholded map image and $I_{MDT}[x,y,\theta]$ is the sub image of I_{MDT} that should be observed from vehicle location (x, y) with heading θ. I_{TT} is the thresholded top image constructed from the stereo image pair at this time step. $C_{\max}$ is the maximum possible match value for the chamfer matching and it is used for normalizing the matching results. The weight of a particle is product of the weight of its parent, similarity of its speed and incremental rotation to the observations, how close it is to a road on the map and how well the map image observed at the particle location matches the observed top view image in the sense of chamfer matching.

$$
\begin{aligned}
w_i^t &= \mathrm{e}^{-\frac{(\Delta\theta)^2}{\sigma_\theta}-\frac{(\Delta d)^2}{\sigma_d}-\frac{I_{RD}(x_i^t,y_i^t)}{\sigma_I}+\frac{corr(\mathrm{I}_{MDT}[x_i^t,y_i^t,\theta_i^t],I_{TT})}{C_{\max}}} \\
\Delta\theta &= \Delta\theta_i^t = \left|\Delta\theta^t_{observed} - (\theta_j^{t-1} - \theta_i^t)\right| \\
\Delta d &= \Delta d_i^t = \left|\Delta d^t_{observed} - \Delta t^{[t,t-1]}\cdot v_i^t\right|
\end{aligned}
\tag{5}
$$

By this scheme, particles that are supported more by the observations obtain more successors for the next frame and the state probability distribution function around these states is sampled denser. It is important to note that information coming from other sensors and observations can be fused easily in the same framework via customizing weight calculation formula in Equation (5).

As a result, the most probable particle will have the highest weight and will denote the possible location and pose of the vehicle at that frame. An online approach is using a Kalman filter and a model switching technique to track several highest weight particles and avoid rapid switches between them. An offline solution to precise localization for each frame in a sequence is to take the highest weight particle in the final frame and to recursively follow its parents backwards. This way, particles states that led to the final frame localization result will be taken as the localization path.

4.11 3D World and Map Overlay Reconstruction

4.11.1 3D Point Clouds

After obtaining the resulting precise vehicle pose from PF at each frame, we transform each frame point cloud using these results and combine them into a global 3D world map. We can now use the voxels both above and below the eye level to better model the environment.

When several frame point clouds for the same scene are combined, the result may contain redundant observations and this may lead to huge files beyond the capacity of personal computers. For example, when point clouds for 3 minutes' drive are combined, the result becomes a 6.5 GB file. In order to get rid of the information redundancy and to reduce the resulting file sizes we define a pixel grid in x-y plane and merge the height and color values that correspond to same grid square. To avoid merging voxels from different objects, only values from voxels close in z-values are merged and ones that correspond to very dissimilar z-values are kept separately.

4.11.2 Map Overlay

We overlay top view images on to the satellite map and obtain a map where road and its surroundings have higher resolution than the original. Moreover, we perform basic dynamic object removal by using voxels that have the lowest z value for same x and y values in the point cloud integration step.

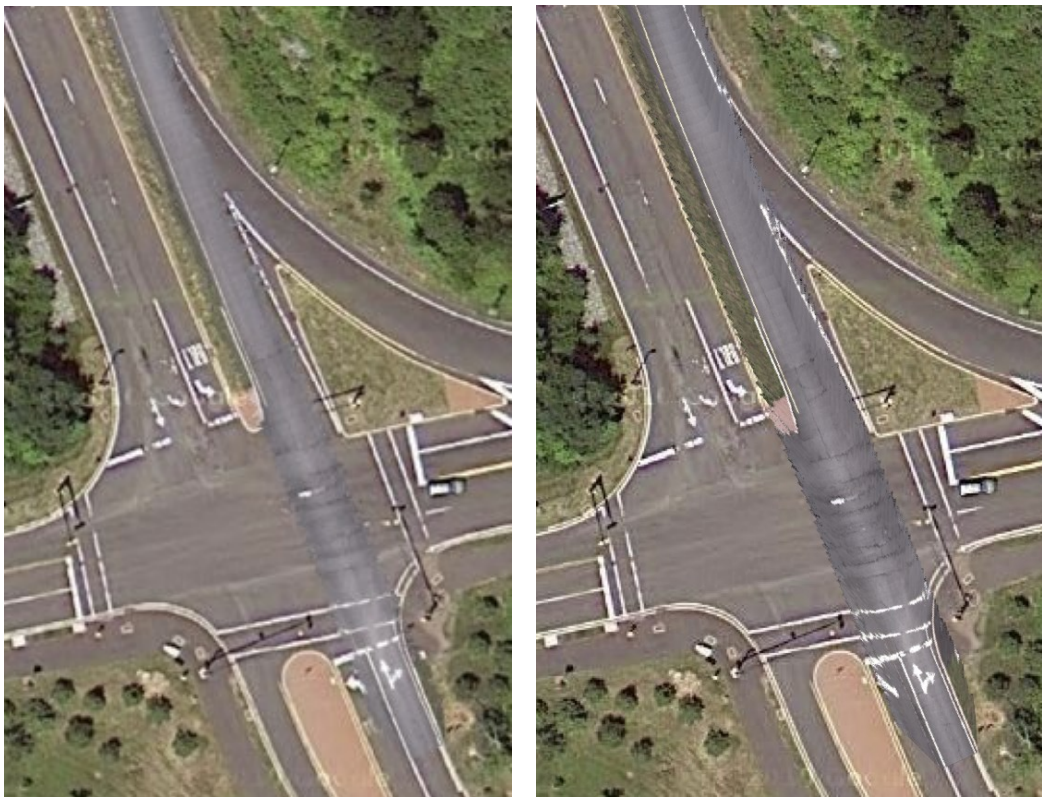
Using the stereo vision based top view reconstruction method, we have achieved superior results to the work in [61], which uses homography based image registration and image mosaicking methods. Due to their inherent assumption of single plane roads and two-dimensional vehicle motion, these methods may introduce distortions in the built road images. Furthermore, occlusion removal process that is uninformed of the 3D structure may fail for both static and dynamic obstacles.

4.12 Results

We show that accurate visual localization on a map is possible by only utilizing stereo images, satellite and road maps, in cases where feature-based map matching methods would fail. In order to test our framework in real-world scenarios, we collected a datasets for two scenarios: a car moving on a highway route and a mobile robot moving on sidewalks. Using the proposed framework, we obtained the corresponding localization information. For the same datasets, we also obtained the 3D point cloud reconstruction and map overlays.

4.12.1 Results for Map Overlay

In Figure 4.10, we show the map overlay where the results of two different top view construction methods are overlaid on to the satellite map. Other moving vehicles are removed automatically in the reconstruction process. Here, we exhibit the closeness of the overlay with the original map while its resolution is improved substantially without introducing any significant distortions. See the distortions in the crosswalk in Figure 4.10 (b) corresponding to ELAS method. A closer look in to the details in the map overlay is shown in Figure 4.11, where details that cannot be resolved in the satellite map are easily observed in the map overlay created by our method.



(a) (b)

Figure 4.10 – (a) Map overlay for MR-SGBM. (b) Map overlay for ELAS. Note that overlaid areas have considerably higher resolution, e.g. two close white road markings can only be resolved on the overlaid areas.

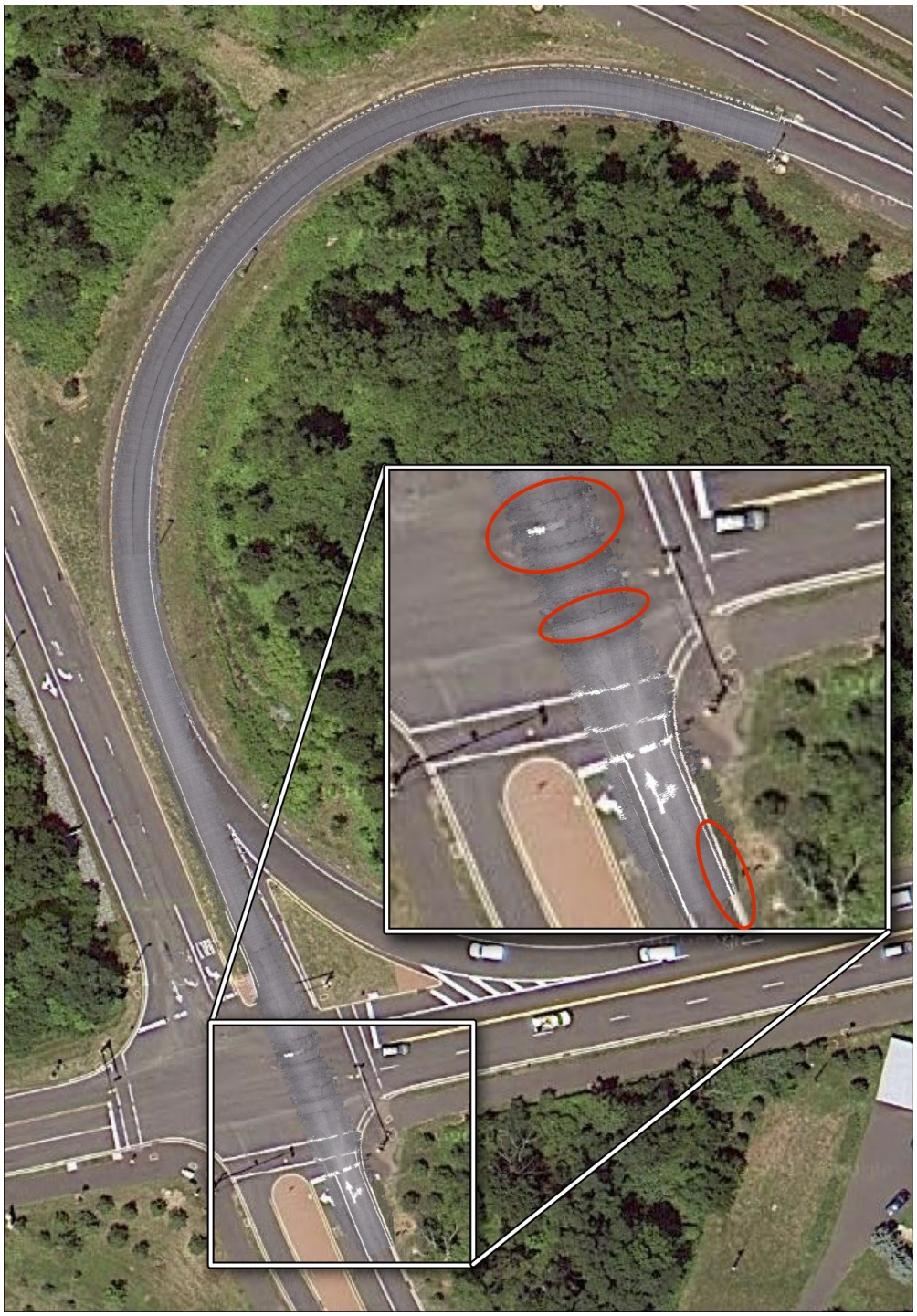


Figure 4.11 – High-resolution road region image overlaid on the low-resolution satellite map. Visible road cracks and resolvable double-lines are marked in the zoomed in high-resolution overlay image obtained from our method.

4.12.2 Results for 3D World Reconstruction

Results for 3D reconstruction of a part of a rural road from our dataset are shown in Figure 4.12. Defects on the road can be easily observed from this reconstruction.

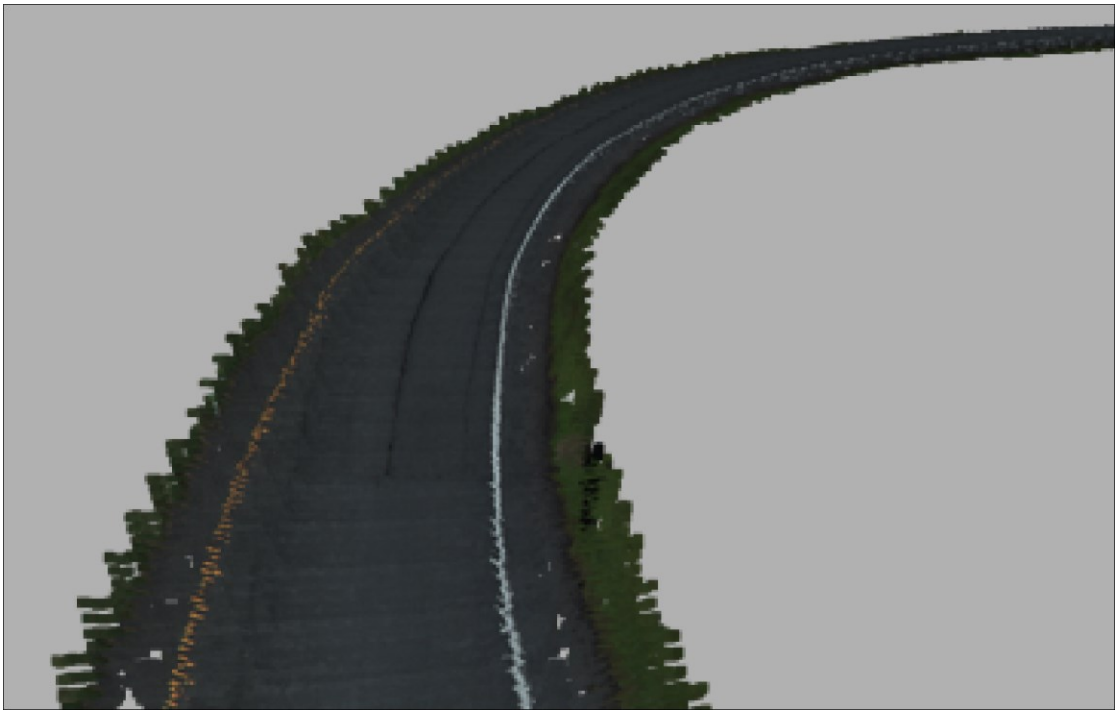


Figure 4.12 – 3D Reconstruction of a rural road and its surroundings. MR-SGBM point clouds are combined for 100-frames.

4.12.3 Results for Vehicle Localization on Roads

For dataset collection, we used a sport utility vehicle with a PointGrey BumbleBee2 [58] stereo camera installed onboard. One of the reasons this specific camera system is chosen is its accurate factory-calibration for lens distortion and camera misalignments, which eliminates the requirement for user calibration. It is stated by PointGrey that “the

rectification process has misalignment of epipolar lines within 0.05 pixels (RMS)" [56], [58].

We acquired stereo images for a 2-km drive in and out of the Rutgers Campus. The chosen area is a typical rural road example where on-road features are very scarce. The dataset consists of 1580 stereo image pairs, where each image has 1024x768 pixels resolution. Images are taken at 10 FPS and the vehicle is driven at an average speed of 48 kph (30 mph).

The vehicle localization results for PF step are shown in Figure 4.13 (c) and Figure 4.13 (d) for the 2 km path. The actual path driven is shown in Figure 4.13 (a). When these results are compared with original VO localization results (Figure 4.4), a substantial improvement in the localization drift can be observed. The original VO drift is 230 meters and approximately 16° for a 3-minute, 2 km drive. In contrast, our approach results in at most 15 meters linear and 0° angular drift for the same path.

(a) (b) (c) (d)

Figure 4.13 – PF localization results shown on satellite image. (a) Google Maps snapshot of the area and the path followed. (b) Satellite map of the area acquired and stitched by the system. (c) All particles for all frames - particles are shown in shades of red depending on their weights where brighter colors denote higher weights. (d) Locations of the most probable particle and its parents.

4.12.4 Results for Robot Localization on Sidewalks

For our experiments, we collected two datasets for different time of day, different weather conditions and slightly different paths. Both datasets are from a suburban sidewalk inside Rutgers campus. Datasets contain real-life elements like pedestrians, moving cars and areas occluded by high trees. First dataset is 570 m in length with a road crossing. Second data set is for a 478 m sidewalk path. The first dataset is taken on a sunny day at sunset where the second is taken at a partially cloudy day mid-day time. First dataset contains GPS points taken by the onboard GPS module. Frames are captured at 2 fps and 6 fps respectively. It is also worth noting that the average robot velocities are reasonably different on both datasets (0.9 m/s and 1.3 m/s). Ground truth paths are marked on the map manually.

Using our proposed framework, we obtained particle-filtering localization results for dataset 1 (Figure 4.14) and dataset 2 (Figure 4.15). These results show the location of the highest weight particles for particle-filtering, where observation sources are visual odometry (VO), combination of visual odometry and sidewalk probabilities (VO, SP) and combination of visual odometry, sidewalk probabilities and map-matching (VO, SP, MM). Furthermore, GPS sample locations (Figure 4.14) and ground truth is shown for comparison (Figure 4.14, Figure 4.15). Localization with map matching is able to situate the robot on the map with few errors. It is also able to find the correct location where the road crossing occurs, even though the framework does not incorporate any special crosswalk-matching component.

We show that, using the proposed framework we reduce the average sidewalk localization error of the mobile robot up to 10 times when compared to our baseline visual odometry method. This result is also slightly better than the GPS localization result for our dataset. All tests are performed with 2,000 particles, which is a good compromise between accuracy of the result and the memory usage.

Table 4.1 lists the quantitative summary of dataset 1 and dataset 2 results in terms of deviation from the ground truth, where deviation is defined as the Euclidean distance to the closest ground truth point. It can be observed that by introducing sidewalk probabilities, we improve visual odometry localization results by 7 times. Moreover, with the help of map matching, these results are further improved and became as accurate as GPS localization on the average and with a 45% less maximum error. Deviations from the ground truth for the path travelled are shown in Figure 4.16 to get a detailed view on the localization error.

Table 4.1 – Error results for different localization methods

Localization Error	GPS	PF VO	PF VO, SP	PF VO, SP, MM
Dataset 1				
Mean	1.52 m	15.22 m	2.00 m	1.55 m
Max	6.79 m	57.41 m	7.00 m	4.67 m
Std. dev.	1.22 m	11.48 m	1.69 m	1.28 m
Dataset 2				
Mean	N/A	4.56 m	1.52 m	1.24 m
Max	N/A	17.60 m	4.10 m	3.30 m
Std. dev.	N/A	4.06 m	0.93 m	0.96 m

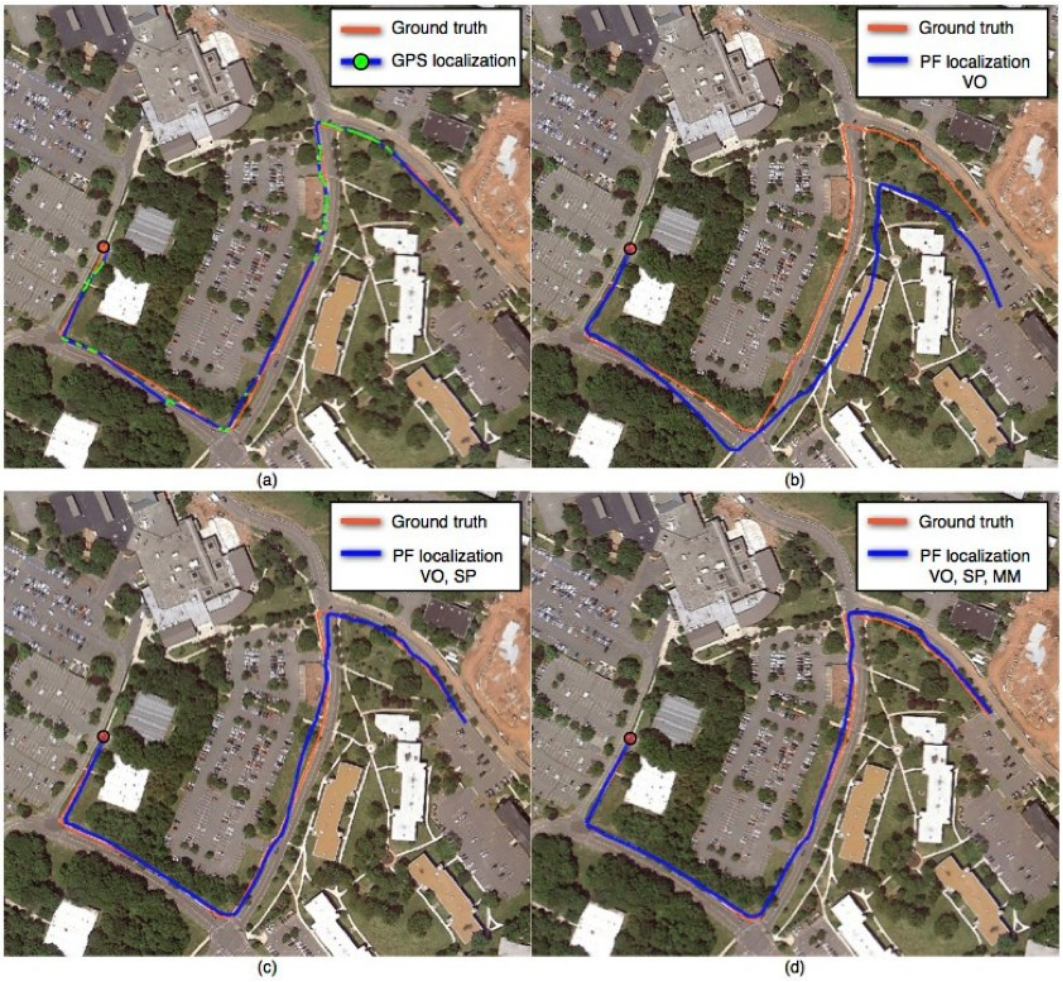


Figure 4.14 – Ground truth, GPS and particle-filtering localization results for dataset 1. (a) GPS localization showing GPS points taken along the route. (b) Particle-filtering localization results for VO. (c) PF results for VO. (d) SP. PF results for VO, SP, MM. Red circles mark the start point.

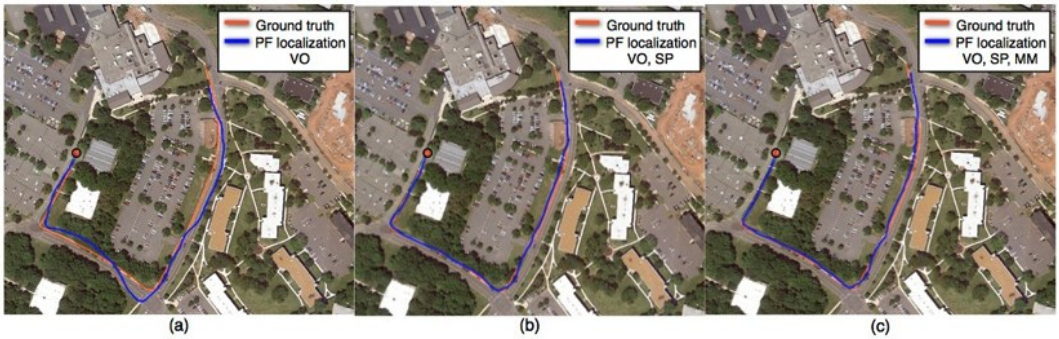


Figure 4.15 – Ground truth and particle-filtering localization results for dataset 2. (a) Particle-filtering localization results for VO. (b) PF results for VO, SP. (c) PF results for VO, SP, MM. Red circles mark the start point.

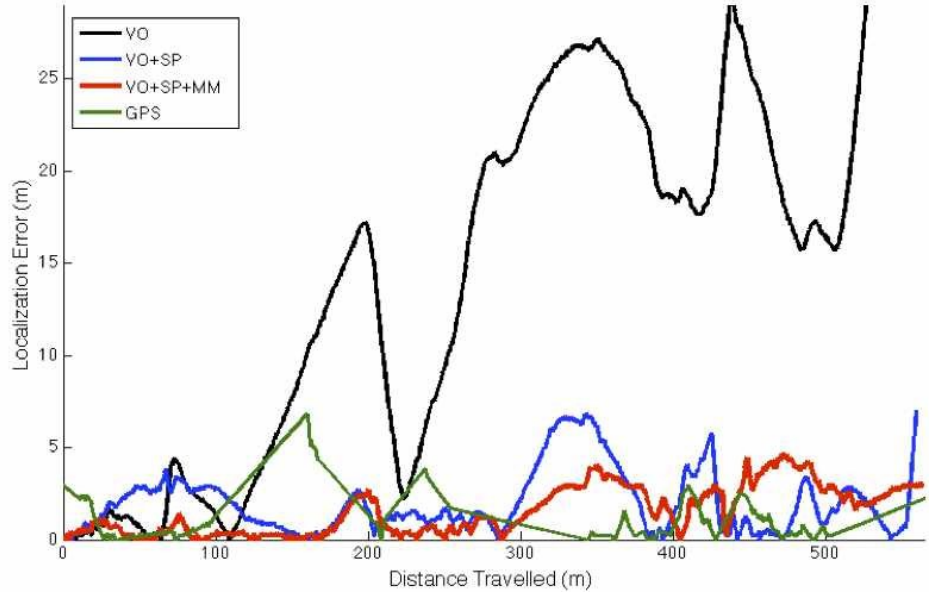


Figure 4.16 – GPS and particle-filtering localization errors for dataset 1. Different observation sources for particle-filtering are shown. Localization error is the minimum distance of the estimated location to the actual path. In the figure, localization error values (Y-axis) are restricted to 30 m to show more details in the lower parts. Actual data values for VO go up to 57.41 m. High error areas for PF results usually happen after a sharp turn has occurred. High error areas for GPS results usually happen when the time between valid GPS sample points are high, due to low GPS reception under tree-occluded areas or a valid GPS fix could not be obtained.

Map-matching component is especially helpful in correcting the incremental translational errors, which cannot be corrected easily with the other methods. Occlusion of sidewalks on the map is an issue that can be addressed in further studies. For now, we fix this issue by simply omitting the map-matching observations of a frame if the matching score is below a fixed threshold. In Figure 4.14, it can be observed that map-matching, when compared to sidewalk probability results, helps fix the robot location in turns and crossings on the map as well as in straight sidewalk areas.

4.13 Conclusions

The results we present here show that our method is capable of fusing inputs from very different sources to obtain reasonable localization outputs. We show that it is possible to obtain GPS accuracy localization results just by using stereo images and low-resolution free Internet maps. GPS, INS and wheel odometry data are not used as inputs to our system, however if available, they can be directly incorporated as new observations to the framework in order to further improve the localization results.

Using stereo images, satellite and road maps, we have obtained accurate vehicle poses, high-resolution top view images, map overlays and 3D reconstructions of the road and its surroundings. These outputs of the system can be used in many different and interesting areas besides our main goal of vehicle navigation. Some of these application areas are road maintenance for road defects (see road defects in Figure 4.12 and Figure 4.11), realistic driving simulations, detailed 3D virtual tours, driving assistance using augmented reality and servicing of outdated satellite images.

Since no global optimization step is necessary in the process, real-time implementation of the algorithm is possible on ordinary personal computers, which opens doors for many fascinating vehicle applications. We also show that the framework can work on relatively small frame rates (e.g. 2 fps) without a performance degradation, which increases the available frame processing time for real-time implementations.

Chapter 5

Visual Localization from Monocular Image Sequences

In this chapter, we solve the vision based geo-localization problem for monocular image sequences. Unlike the method in Chapter 4, we do not have access to stereo images, and thus 3D scene points, top view images and incremental motion estimates all have to be estimated using structure from motions methods. We localize given images on the map by matching top view image of the scene to the map, and then match the perspective projected map portions to the images. We show that with this method we can achieve high accuracy localization results for each image along with 3D scene points and dense and accurate top view images.

An outline of the steps of the localization pipeline is shown in Figure 5.1, where stereo localization steps from Chapter 4 are also shown.

5.1 Motivation

In this chapter we perform vision based global localization with matching monocular camera images to satellite maps. For the stereo localization method in Chapter 4, we make use of stereo matching and stereo visual odometry (VO), for a monocular video case, we make use of structure from motion (SfM) techniques in order to perform scene

building and localization. For our *orthographic matching* step, similar to the previous methods, we first build the top view image and roughly localize our frames on the map by tracking or performing exhaustive search. In the *perspective matching* step, using the inverse of the obtained top view projection transformation, we project the map portion seen by the camera in to the perspective camera view to further adjust the location and orientation of the camera on the map. With this step, we can achieve up to 5-10 cm horizontal localization accuracy on free satellite maps with comparable pixel resolutions. As long as there are enough features on the video to match to the satellite maps, we are not bound by specific video locations. Hence, we can perform localization on handheld walking sequences, city and highway vehicle sequences as well as other possible sequences with sufficient motion and enough features.

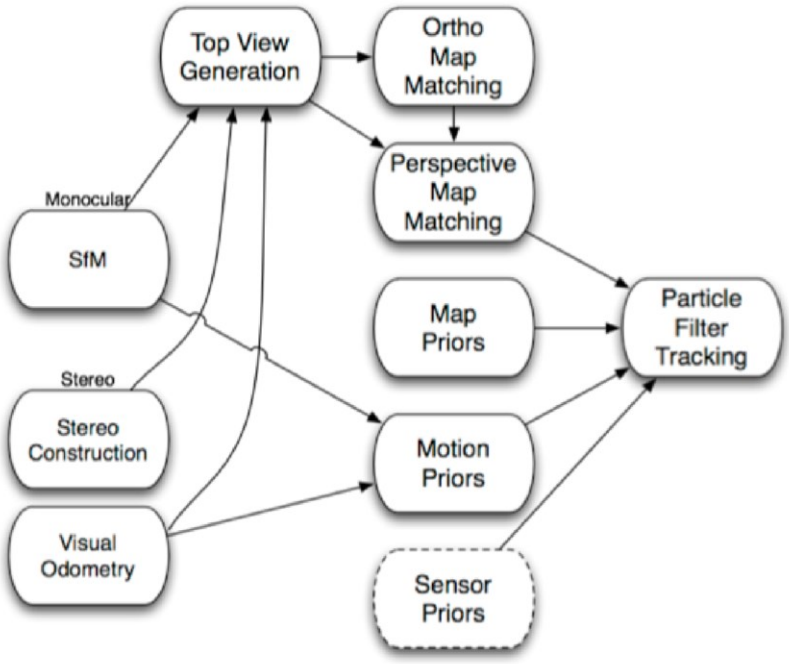


Figure 5.1 – Visual localization pipeline. Showing common steps of the framework for both stereo and monocular camera cases.

5.2 Related Work

Visual localization from satellite maps is one of the emerging topics that recently started gaining interest with the availability of free high resolution satellite maps and several advancements in computer vision techniques. Although camera image to map matching concept dates back to almost three decades ago [98], the modern work on this area that makes use of satellite maps starts with the works where point to line feature matching [96] and fixed homography and point feature matching based localization [52] were used. Following these, in the framework described in Chapter 4 [46], [47], a complete stereo image framework that can localize a vehicle with GPS accuracy and can generate accurate top view images has been presented. After these works, methods for large scale localization based only on visual odometry and road map information has been proposed and evaluated by [99], [100] in independent efforts both using OpenStreetMaps data.

Given a set of monocular images, estimating a sparse representation of the 3D scene structure has been available through recent advances in Structure from Motion (SfM) [32], [33] and Simultaneous Localization and Mapping (SLAM) [30] techniques. These techniques tracks many features through frames to both determine the 3D positions of each feature point in space and at the same time estimate the position and orientation of the camera for each frame. SfM applies techniques generally apply an optimization step over camera locations and scene structure, whereas SLAM techniques try to localize the camera at each frame in the map built by the features in the scene so far.

5.3 Scene Modeling

Our visual localization pipeline has scene building, top view generation, map matching and particle filtering steps. The outline is shown in Figure 5.1.

The visual localization pipeline starts by processing the camera images and obtaining the scene structure and camera locations. We call this first step as scene building step. Scene structure is the amount of 3D information that needs to be obtained from the images to be able to estimate the ground plane, remove unwanted objects, perform image transformations and build the top view image. In stereo scenes, 3D information about the scene can be obtained per frame using stereo disparity estimation, pair without the need for motion. In monocular scenes, use of motion and multiple frames is essential to obtain the required 3D information.

Camera locations are the positions and orientations of each camera frame relative to its previous frame. Camera locations are used in building the scene structure, building a larger top view image from single top view images and estimating an initial relative localization before the global registration step. For a monocular scene, motion information has to be employed to build the required 3D information. Structure from Motion methods track point features in frames over multiple frames and based on geometric reasoning, estimate the 3D position of each point in the scene and camera pose in each frame. Usually SfM techniques require a global optimization over all frames, but in recent methods, this has been reduced to optimization over a group or bundle of frames. With the global optimization requirement removed, we can use SfM techniques to build an accurate sparse 3D scene for the last few frames and make to pipeline work

for online problems. For offline localization problems, a global optimization over all the frames can be applied.

Top view generation algorithm differs from stereo to mono sequences. In stereo, dense top view images can be obtained from each image pair after estimating the stereo disparity between the images. In monocular case, only a sparse 3D scene can be estimated using Structure from Motion (SfM) [32], [33] or Simultaneous Localization and Mapping (SLAM) techniques. Thus, a direct transformation from this scene will lead to a very sparse top view image. For generating a dense top view image, in the monocular case, we assume that there is a dominant ground plane in the scene. We estimate the ground plane equation from sparse 3D points, find the pixels that are on the ground plane and transform these image pixels densely to the top view to generate a dense top view image. In both monocular and stereo cases, multiple frame top view images are combined using relative camera motion information to generate a larger, denser and more consistent top view image. Furthermore, this way, regions of top view image that are not observed in one frame can be filled from other frames to avoid holes in the final top view image.

Monocular scene top view generation for a group of frames takes estimated sparse 3D scene points, the corresponding ground plane equation and warps the relevant part of the images using the ground plane equation and each frame's camera pose. Although warping is done for each frame separately, finding the ground plane and 3D points belonging to the ground plane, need the multiple frames SfM analysis. Thus, unlike the stereo top view building, this method of finding the top view image works only in the presence of camera motion. Removing the moving objects and fixed objects that do not belong to ground plane is crucial to monocular top view generation, since pixels are

warped to top view using a single per-frame transformation. Any pixel that does not belong to the ground plane will be transformed incorrectly with this transformation and will appear smeared or stretched in the top view image. Sparse points on moving objects can be estimated in the SfM process and they can be grouped per object using their pixel similarity and motion similarity. This way, dense exclusion masks for each moving object can be obtained. For finding the inclusion masks for pixels on the ground plane, we apply segmentation and labeling techniques by giving sparse points on the ground plane as prior labels. We limit our segmentation to the convex hull of all ground plane points. The final mask for pixels to be transformed is found by removing all exclusion masks from the ground plane inclusion mask. For each frame, all the pixels in this final mask are warped to top view image using the transformation obtained from the ground plane equation of the frame group. A stitched top view image similar to stereo scene case is created using the camera poses and stitching and blending the multiple top view images.

When the top view image for each frame is constructed, in the ortho matching step, they can be matched to satellite image using one of the various methods depending on the application. Matching intensity edges from top view image to satellite map would be useful if the region is known to have lines observable from the satellite image. Road lane markings and sidewalk edges are good for edge matching. If the application is not limited to roads or sidewalks, color matching is a better choice. Color matching has to take in to account that white balance, exposure, noise level and resolution of the top view image and satellite may differ substantially.

Perspective matching is a finer matching step that can generate highest accuracy visual localization which has to come after a good localization is achieved with the

orthographic matching step. It also would be more efficient to apply perspective matching after candidate locations found with orthographic matching step, due to the larger images under comparison in perspective case. Perspective matching applies the inverse of the top view image generation process and transforms a satellite image region to the perspective camera view.

For a monocular scene, motion information has to be employed to build the required 3D information. Structure from Motion methods track point features in frames over multiple frames and based on geometric reasoning, estimate the 3D position of each point in the scene and camera pose in each frame. Usually SfM techniques require a global optimization over all frames, but in recent methods, this has been reduced to optimization over a group or bundle of frames. With the global optimization requirement removed, we can use SfM techniques to build an accurate sparse 3D scene for the last few frames and make to pipeline work for online problems. For offline localization problems, a global optimization over all the frames can be applied.

For a stereo scene, relative camera poses for each frame can be obtained using the very accurate Visual Odometry (VO) techniques [34]. Modern stereo VO techniques ([35], [36]) track multiple features over multiple frames and match these features from left image to right image. This way both 3D scene points and camera poses are consistently estimated. Outlier rejection step removes 3D points that do not belong to the rigid scene. With stereo camera calibration parameters supplied, stereo scene building and stereo VO generates correct metric results.

For a monocular scene, relative camera poses are estimated as a part of the SfM. SfM techniques can estimate the camera parameters for an uncalibrated camera or can use

parameters from user calibration. Both scene structure and camera pose results from SfM are correct up to a scaling factor.

5.3.1 Structure from Motion

For a monocular video sequence, Structure from Motion (SfM) [32], [33] is used to estimate camera parameters, camera locations and 3D point locations in the scene using camera images. Harris corners found in the frames are tracked across multiple frames using a KLT tracker [101]. 3D locations of tracked features and camera poses for each frame are estimated. Camera focal length is given to SfM and rest of the camera intrinsic parameters are estimated automatically using [102]. Camera locations and scene points calculated using SfM have an arbitrary starting point and an arbitrary scale.

Given a set of monocular images, estimating a sparse representation of the 3D scene structure has been available through recent advances in Structure from Motion (SfM) [32], [33] and Simultaneous Localization and Mapping (SLAM) [30] techniques. These techniques tracks many features through frames to both determine the 3D positions of each feature point in space and at the same time estimate the position and orientation of the camera for each frame. SfM applies techniques generally apply an optimization step over camera locations and scene structure, whereas SLAM techniques try to localize the camera at each frame in the map built by the features in the scene so far.

5.3.2 Frame Grouping

We divide the whole set of frames in to smaller groups to cover larger distances and to group more number of points together. For frame grouping we use k-means clustering on camera locations and obtain groups of consecutive frames. Unless there is an abrupt change in the camera location, groups have similar number of frames. Having frame groups instead of individual frames help us have more consistent plane estimations (Figure 5.2) and by using a single frame per group as key frames we can lower the computational load in localization steps.

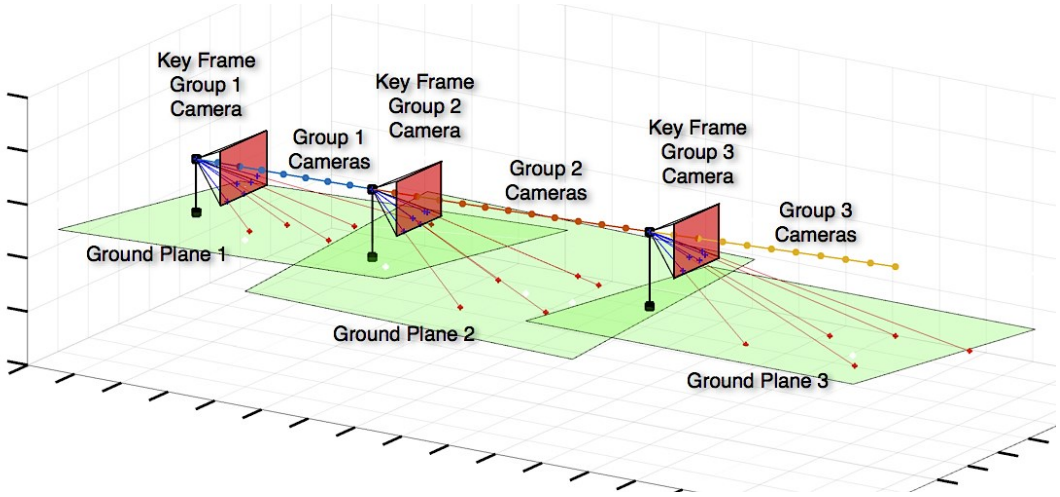


Figure 5.2 – Multi-plane representation of world. All frames in a frame group share the same ground plane. Key frames for each group are the first frames in the group. Orthographic matching is performed on key frame cameras using the combined top view image for the group.

5.3.3 Ground Plane Estimation

For each frame group, we estimate the ground plane using RANdom SAmple Consensus (RANSAC) plane estimation procedure [72], [105]. With RANSAC, we find the plane supported by most number of points in the scene. If the normal of the estimated plane is not aligned with up vector of the camera, we remove the points belonging to this plane and restart plane estimation on the remaining of the points. We continue the procedure until a ground plane is found or there are too few points left for estimation.

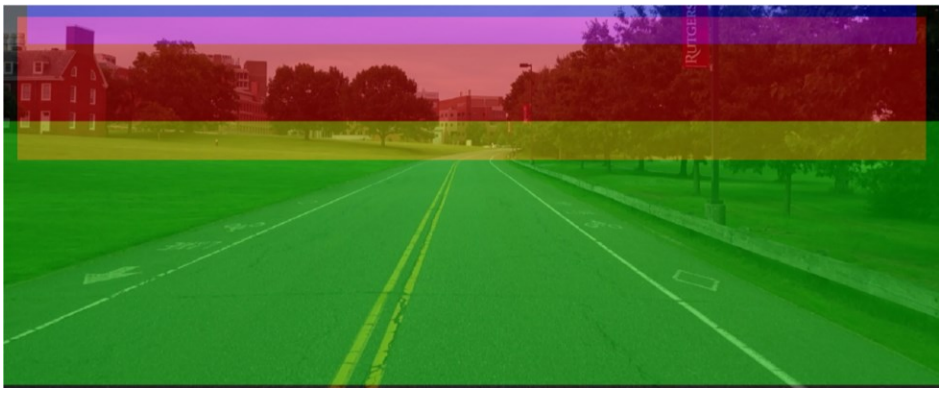


Figure 5.3 – Limiting pixel locations for ground plane and 3D object point estimations. Feature points in the green region are considered for ground plane estimation. Feature points and their corresponding 3D scene points are considered for 3D object point estimation.

To limit our plane estimation only to reasonable points in the scene, we restrict point set to points that are below the horizon in the image. Without explicitly searching for the horizon in the image, assuming the camera is roughly parallel to the ground plane, we take points that are below the optical center of the image for plane estimation. These image limits are shown in Figure 5.3 and resulting feature points considered for ground plane estimations and 3D object estimations are shown in Figure 5.4.

Figure 5.4 – Point considered for ground plane estimation (red) and 3D object point estimations (blue).

Ground plane estimated for each group is shared between all the frames in the group. A sample frame ground plane points are shown Figure 5.5.

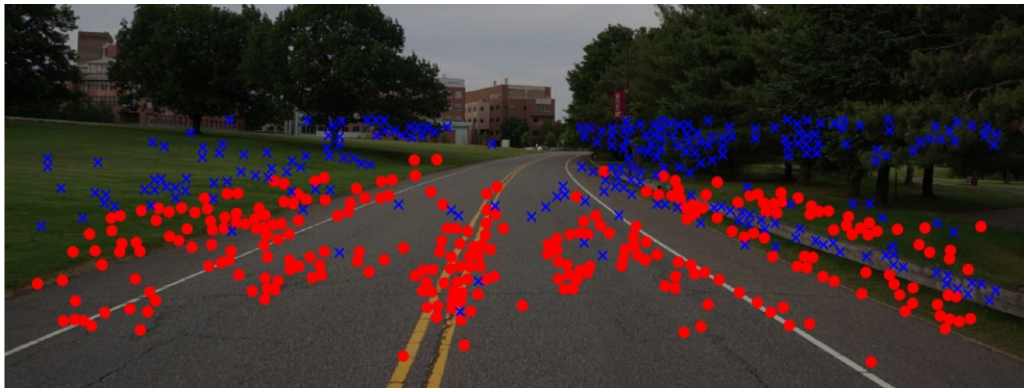


Figure 5.5 – Inlier (red) and outlier (blue) points for estimated ground plane.

5.3.4 Scene Building Results

For monocular datasets we estimate the camera poses and 3D scene points using SfM method outlined in [102] and using the tool in [108]. Scene points and camera locations for campus road dataset is shown in Figure 5.6.

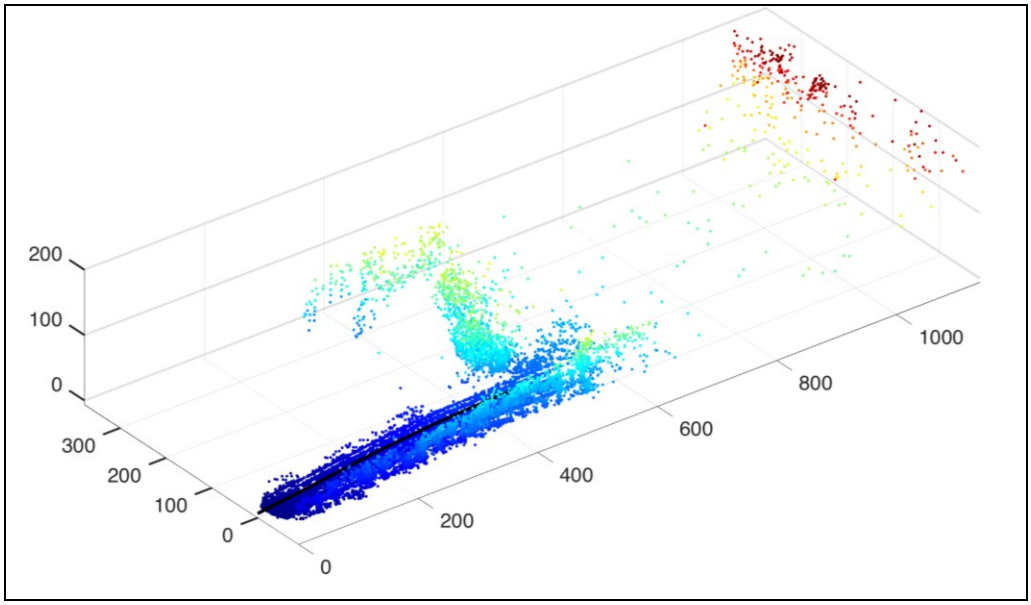


Figure 5.6 – Scene points for campus road dataset. Estimated camera positions and scene points, colored by height from the ground.

5.4 Localization with Orthographic Matching

In the orthographic matching localization step, we start from a known location on the map and estimated relative position and orientation of the camera for each frame. In this step, we improve the estimated camera positions and orientations by registering each frame to a region in satellite image. This step eliminates the accumulated error in the camera pose estimation by registering camera poses to the map. It is important to note that at this step, we do not estimate a full 6-DOF pose of the camera but instead we only estimate the orientation and projected (x, y) location of the camera on the map.

We model the world as a series of planes and we generate dense top view images corresponding to frame groups to perform localization for the key frames of each group.

5.4.1 Top View Mask Generation

After finding the ground plane and sparse points on this plane, we find a mask of pixels in the image belonging to the ground plane to transform the image to top view. If we transform the entire image without applying this mask, pixels that do not belong to the ground plane will be stretched. For our mask on the frame image, we find the region that covers ground plane points and also have sufficient visual similarity to these points.

We use convex hull of the pixel locations of the points on the plane as the region covering the points, as seen in Figure 5.7. We also perform a binary MRF labeling of the image by giving plane inlier points as the first class and other scene points as the second class. Result of this labeling gives pixels that look like the given ground points. This step is done to remove any different looking objects that may exist partially or fully in the convex hull (e.g. cars, pedestrians, fences, etc.). Final top view mask is the intersection of these convex hull and similarity masks.

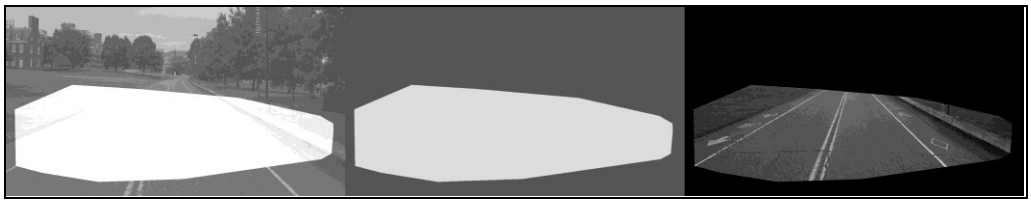


Figure 5.7 – Convex hull top view mask.

5.4.2 Inverse Perspective Mapping and Top View Image Generation

Based on estimated camera position, camera parameters and ground plane equation, we estimate an inverse perspective image transformation that generates the orthographic top view image of the scene, which is a homography. Unlike the previous work, we do not require the ground plane to be exactly parallel to the world x-y plane or camera view direction to be parallel to the ground plane or x-y plane.

Although we are solving the general case, we first give the projection equations for the easier case where camera center lies in origin of the world coordinate system, camera optical axis pointing in the +x direction and camera up vector pointing at +z direction as shown in Figure 5.8.

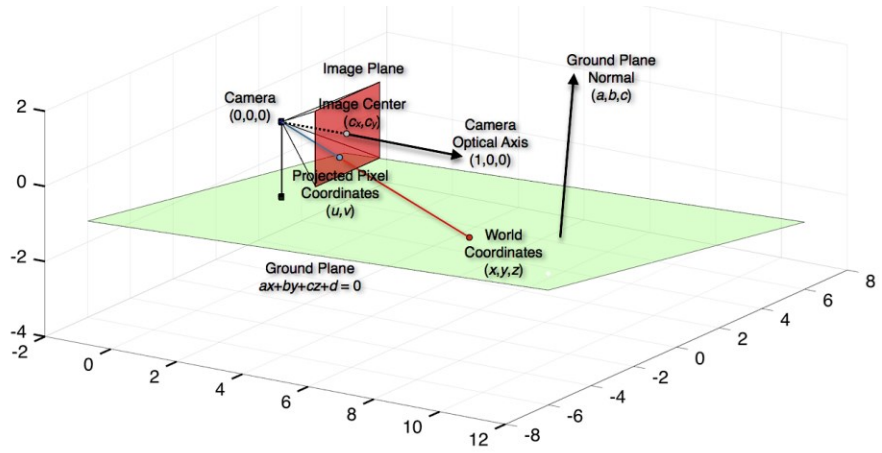


Figure 5.8 – Camera, ground plane and plane normal. Formulation is shown in with camera in the origin and looking through positive x-axis.

For a 3D world point *(x, y, z)* the corresponding projected 2D image point *(u, v)* can be calculated using the pinhole camera model by Equation (6), where camera focal length is given as f.

$$(u, v) = \left(f\frac{y}{x}, f\frac{z}{x}\right) \tag{6}$$

Coordinates (u, v) are pixel coordinates centered at the image center (c_x, c_y) and correspond to image pixel coordinates (p_x, p_y) as shown in Equation (7). Camera optical axis is the hypothetical ray that starts from camera location and passes through the image center.

$$(u, v) = (p_x - c_x, p_y - c_y) \tag{7}$$

For the ground plane as given in Equation (8), the normal vector is (a, b, c).

$$ax + by + cz + d = 0 \tag{8}$$

For a pixel with centered pixel coordinates (u, v) the corresponding world coordinates can be found by intersecting the ground plane and the ray that starts from camera location and passes through the pixel. This ray can be given in parametric form as in Equation (9). Here it is worth noting that world y-axis corresponds to -x axis in image coordinates and world z-axis corresponds to -y axis in image coordinates. When we

intersect the ray and the plane, we can find the intersection point through the parametric form Equation (10), (11). If the denominator in Equation (12) becomes 0, this means that the pixel is on the image horizon and the ray is parallel to the plane and will not intersect. If for a given point, t is negative, then the point given is above the horizon and the intersection world point will be behind the camera and thus it is invalid.

$$t(f, -u, -v) = 0, t > 0 \tag{9}$$

$$atf - btu - ctv + d = 0 \tag{10}$$

$$t = -\frac{d}{af - bu - cv} \tag{11}$$

For the more general case, where camera is located on (cam_x, cam_y, cam_z) and still pointing the same direction, the ray in Equation (9) now can be formulated as Equation (12). When the camera has a rotation around its location around +z axis by $\gamma°$, the ray rotates the same amount around the camera location Equation (13).

$$(cam_x, cam_y, cam_z) + t(f, -u, -v) = 0, t > 0 \tag{12}$$

$$(cam_x, cam_y, cam_z) + t(f\cos(\gamma) - c_x\sin(\gamma), -c_x\cos(\gamma) - f\sin(\gamma), -c_y) = 0,\ t > 0 \tag{13}$$

Top view images are obtained by projecting the dense set of pixels in the masked region using the homography between image plane and ground plane. Each frame in the group is projected to its individual top view using the same group plane equation and then merged in to a group top view using the relative camera locations of each frame.

When stitching and merging top view for frames, pixels from different frames that correspond to the same top view location are blended by favoring pixels closer to the camera, since they will have more details (Figure 5.9).

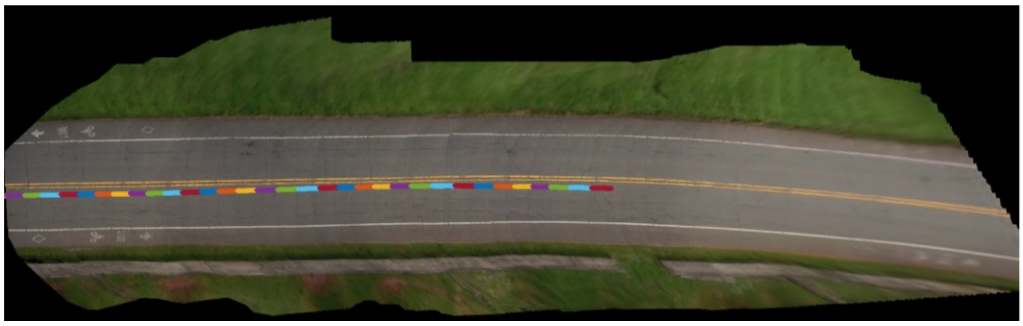


Figure 5.9 – Top view image and camera location. Each camera group is colored with a different color.

5.4.3 Top View Image to Map Matching

We are localizing the camera at each frame of the sequence precisely as the camera moves. We assume that the starting position and orientation on the map are roughly known. From the scene estimation step, the relative camera locations based on observed camera motion in the frames are available. Since these observations are incremental, these locations are susceptible to noise and accumulation of error as the sequence

becomes longer. We need to fix the localization error using a global reference in order to eliminate the accumulated localization error.

To fix the relative camera locations, we only match group top view images to their corresponding satellite map regions and interpolate the camera locations for intermediate frame. For a frame group the corresponding map region is the rectangular region on the map that will be observed by the camera at the given map location and with the given orientation. In addition, we assume that the camera's view optical axis is parallel to the local ground plane, which is roughly true for forward-facing vehicle mounted camera scenarios.

For each frame group we run a search on a range of camera orientations and match the observed top view image with the map region of each hypothetical camera orientation.

Camera angle with best correlation score is chosen as that frame's corrected camera orientation. Starting from the first frame group we sequentially advance among groups and fix the relative orientation of each group, which also affects the orientations and locations of the rest of the groups. This way, error accumulations among multiple frames are corrected. A sample top view image and its map region for the best matched camera pose is shown in Figure 5.10.

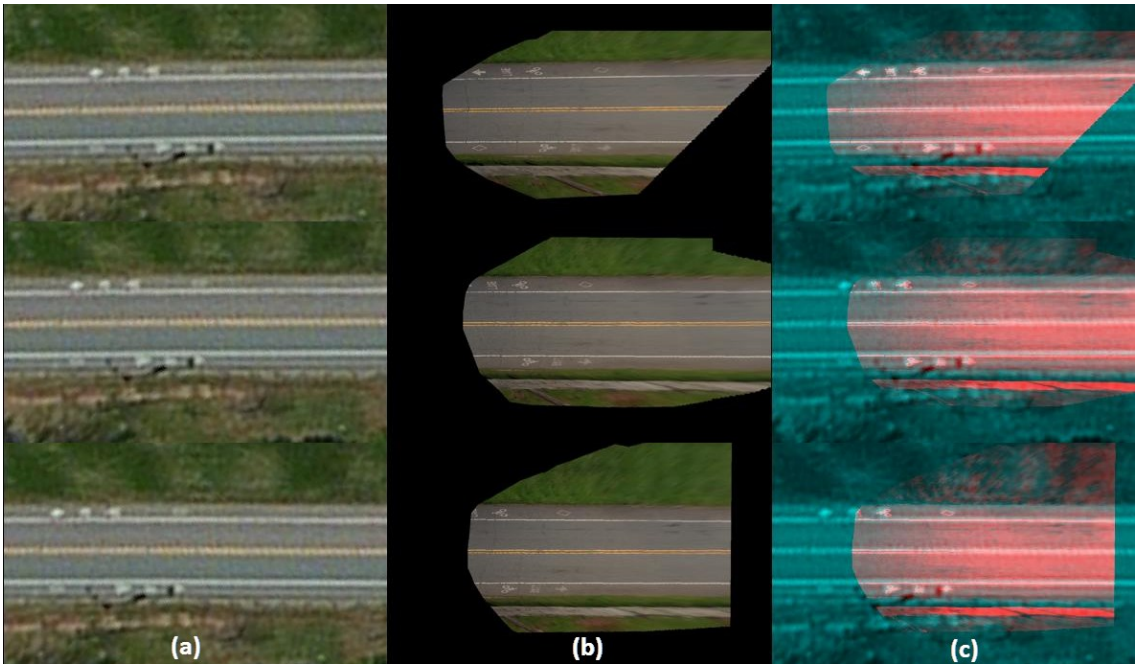


Figure 5.10 – Orthographic matching to top view images of frame groups. (a) Satellite map region for best matching pose to the give n top view image. (b) Group top view image. (c) Top view image overlayed on to the matching map region.

5.4.4 Color Correlation for Image Matching

In aerial image to ground image matching, it is often (e.g. [52], [106]) stated that the color discrepancies and the appearance differences in the camera image and aerial images are too high to have a good matching if appearance matching is to be performed. As a solution to this, researchers suggested using matching of various features including edge [46], [47] (methods in Chapter 4), [96], line [106] and point [52], [96] features. In works of [46] (Chapter 4), [52], [96] the approaches are tuned for street and highway driving scenes and for [52], [96] and several others, the matching is performed on very large map areas (100 m x 50 m) where finding enough unique features is possible. A more realistic approach that can work for smaller regions (10 m x 5 m) has to be found, since these

regions are more plausible to be observed by a single-frame camera image. In addition, the scene should not only be limited to driving sequences, where unique lane markers and road markings are assumed present.

Although it is easy to extract and match point, edge and line features, finding enough number of features that appear in both views may not always be possible; especially in case of environments with fewer features. Thus, in this work we used color matching to match satellite map images to generated top view images.

We match top view images and map regions using normalized cross correlation over RGB color space. Although other color spaces are cognitively more uniform in matching and comparison, RGB color space is less susceptible to encoding errors in color values occurring due to JPEG and MPEG compressions compared to HSV, YUV, XYZ and LAB color spaces. To successfully compare two images in RGB color space directly, brightness and white balance of the images have to be matched before the comparison. Additionally, we have an observable level-of-detail difference between the top view image and the satellite map (Figure 5.11). Therefore, before the matching step, we fix both satellite maps and top view images in order to match each other in appearance. Details of the top view image are reduced using a Gaussian blur filter to recreate the blur in the low-resolution satellite images. In addition, a sharpening filter is applied to the blurred top view image to replicate the post processing applied to the satellite image. We apply color correction to satellite image to match its white balance and brightness to top view image. Assuming the respective map area has a uniform white balance, color correction parameters to be applied are determined in the first few frames of the sequence for the whole map and kept fix throughout the sequence.

Color and resolution correction results are shown for Bing and Google Maps images and top view image in Figure 5.11. In the figure, it can be qualitatively observed that color and resolution corrected top view images look more similar to the maps than the original images.

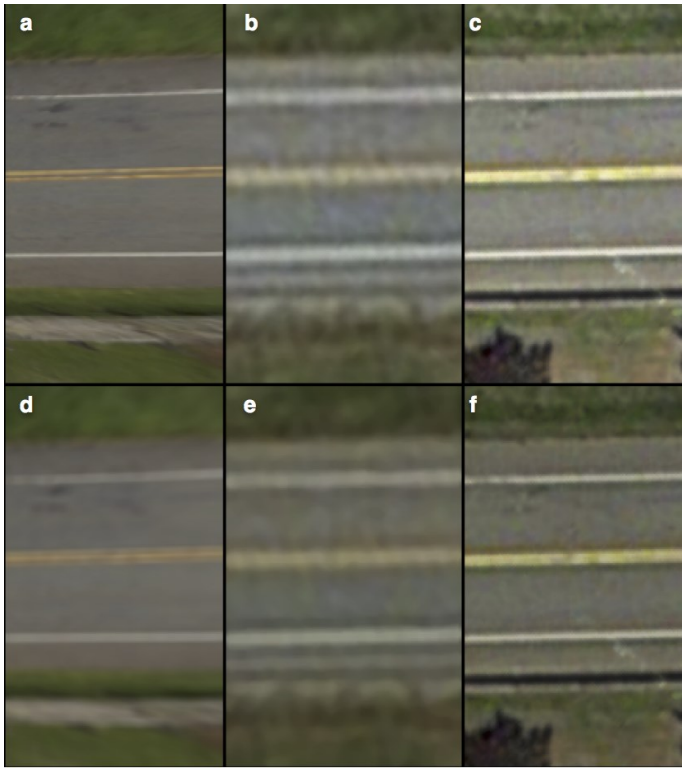


Figure 5.11 – Image color and detail correction. (a) Original top view image. (b) Bing Maps zoom level 19, captured on April 2011. (c) Google Maps zoom level 20 captured on September 2013. (d) Blurred and sharpened top view image. (e) Color corrected and noise reduced Bing Maps image. (f) Color corrected and noise reduced Google Maps image. Figures best viewed in color and in digital version.

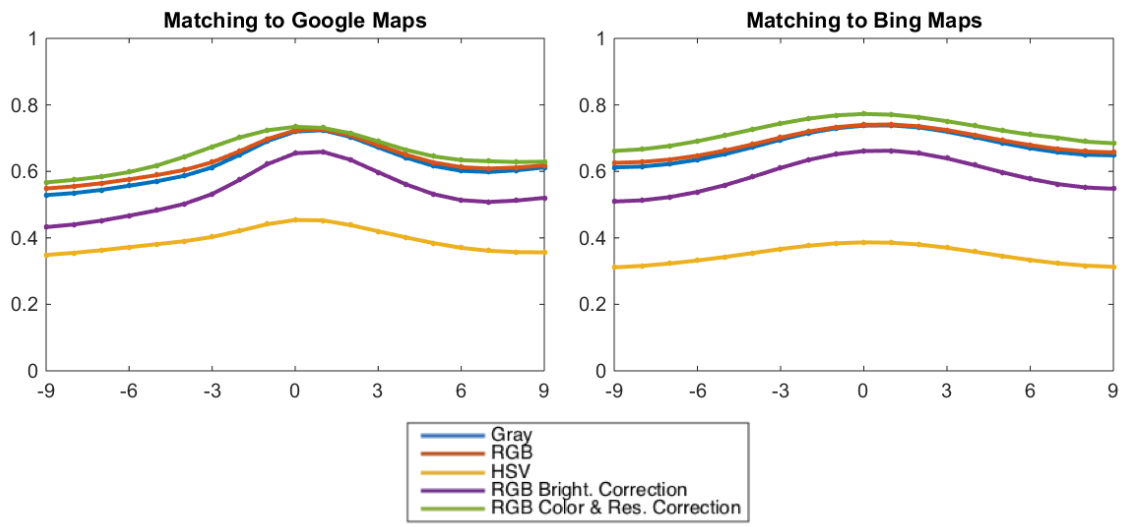


Figure 5.12 – Effect of color and resolution correction to image matching. Normalized cross correlation results for matching Bing and Google Maps images to top view image. Cross correlation values on y-axis are in the range of [-1,+1], where +1 is a perfect match. In the plot, x-axis shows pixel-wise search range in a vertical search across a road top view, where zero value is the correct match.

In Figure 5.12, we show the matching results for a vertical search for the pixel location of a top view image. The figure shows the quantitative effects of color and resolution correction on the image matching process. We match Bing and Google Maps to top view image using various color space options and with and without color and resolution corrections. In both plots, RGB matching with color and resolution corrections outperforms the others and HSV correlation has the lowest score.

5.5 Localization with Perspective Matching

After correcting the accumulative error up to the order of 10's of cm's using orthographic matching, we further improve our localization up to cm accuracy using perspective matching. As we search for best camera pose, for each possible camera

location and orientation we transform satellite map to the camera view by a perspective transformation and match it to the frame image.

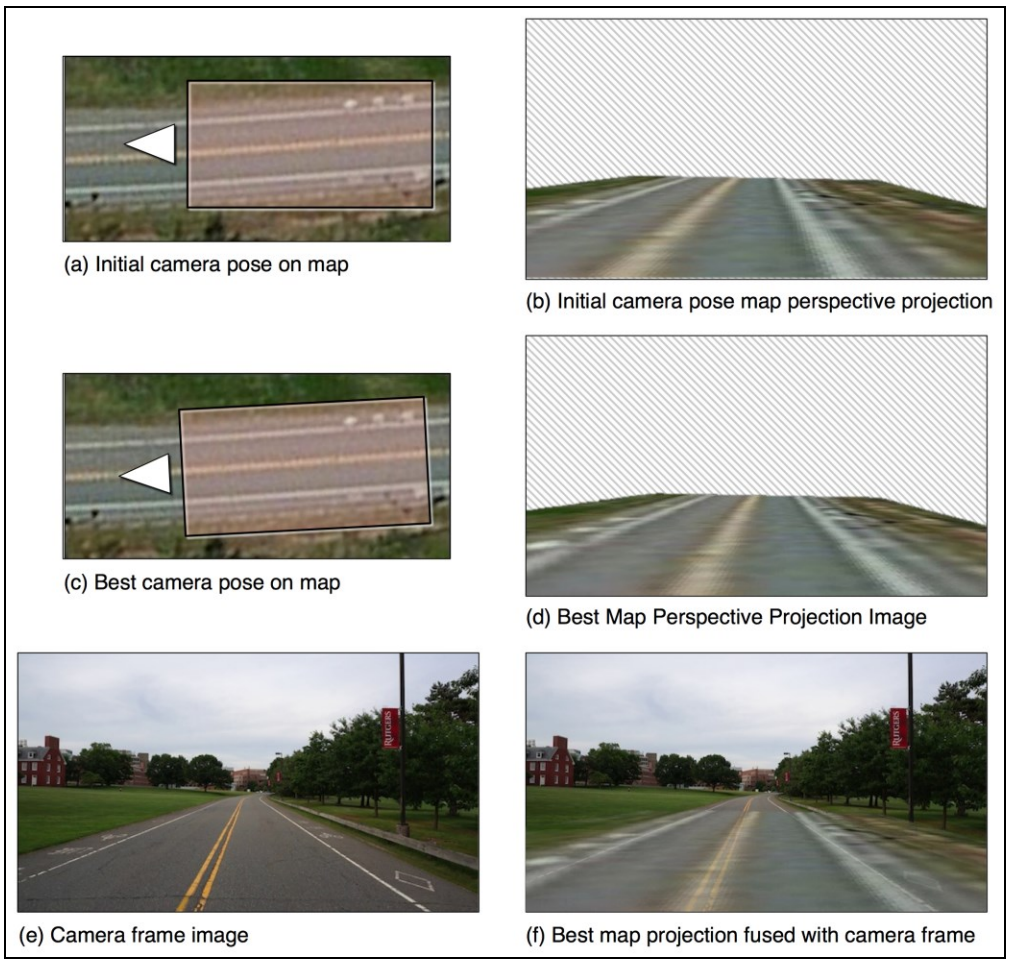


Figure 5.13 – Perspective transformed map to camera image matching. Perspective projections corresponding to camera poses searched on the map are matched to camera frame image to find the best matching pose. In (b) and (d) parts that are not projected are marked with hatching.

5.5.1 Perspective Projected Map Image Generation

We transform the satellite map region observed by the group key frame camera location and orientation in to the camera view by using the group plane equation. Map

regions to be transformed are limited to small rectangular regions. Figure 5.13 (b) and (d) show two perspective map projections for two different camera poses shown in Figure 5.13 (a) and (c), respectively.

5.5.2 Perspective Transformed Map to Camera Image Matching

We search on a range of camera locations and orientations and compare their resulting transformed map images using the same color correlation scheme to find the best matching transformed map image to the camera image. We fill in the intermediate frame camera locations and orientations using linear interpolation between key frames of consecutive groups.

In this step because of the perspective view, the matching score is more dependent on matching of objects closer to the camera than the ones farther. As opposed to this, in the orthographic matching step, matching score is determined equivalently from all pixels, regardless of their distance to the camera.

5.6 Results

Starting with roughly correct initial camera location and orientation we generate camera locations and orientations on the map first by applying ortho matching and then applying perspective matching. Localization results for campus road dataset are shown on

map in Figure 5.14. Camera locations are given in Figure 5.15 and camera orientations are in Figure 5.16.

Figure 5.14 – Campus road dataset frame localization results on map. SfM camera locations (blue), ortho matching corrected locations (red), perspective matching corrected locations (green).

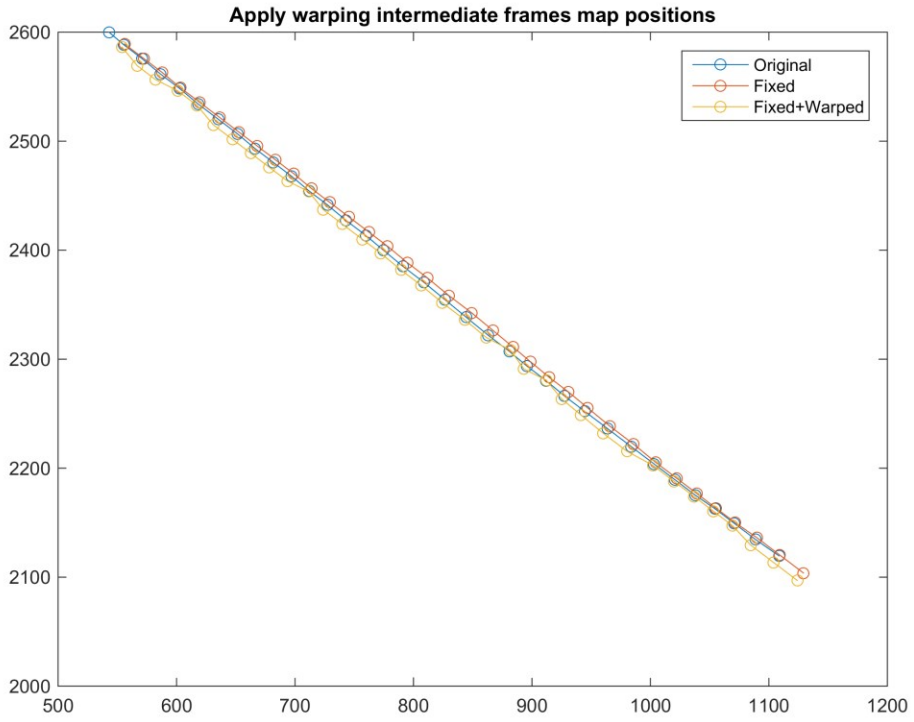


Figure 5.15 – Campus road dataset localization camera locations. SfM camera locations (blue), ortho matching corrected locations (red), perspective matching corrected locations (green).

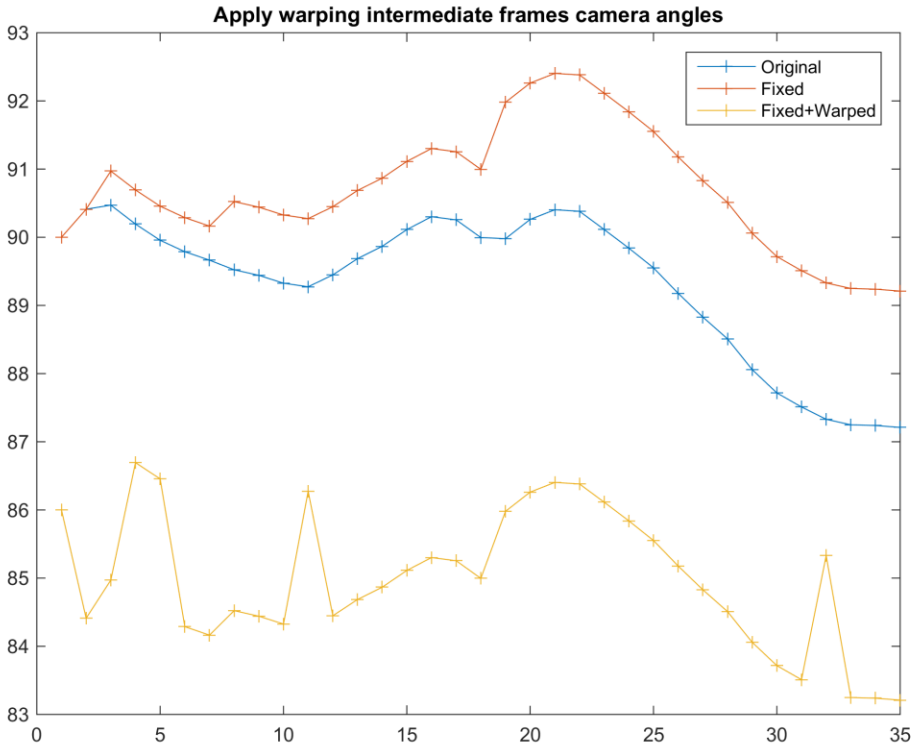


Figure 5.16 – Campus road dataset localization camera orientations. SfM camera locations (blue), ortho matching corrected locations (red), perspective matching corrected locations (green).

Chapter 6

Semantic Video Segmentation and Labeling using Satellite Maps

This chapter presents a novel idea for video segmentation and semantic labeling for videos taken from moving platforms. Typical approaches to video labeling involve supervised or semi-supervised learning from given camera images or a similar test set of images. In contrast to this, in our work, we accomplish video segmentation and labeling without performing learning on the camera images but by projecting labels that we obtain from satellite maps on to the perspective view of the video frames. Our approach has many advantages to existing methods, because semantic labeling on orthographic aerial maps is a less complicated task than labeling in complex 3D environments viewed by a perspective camera.

Projecting map label information to original camera view depends highly on accurate localization of video frames on the map. Hence, we first solve the localization problem by using accurate vision based geo-localization methods. We localize video frames on the map by employing orthographic and perspective matching steps from top view image of the scene to the map, as explained in Chapters 4 and 5. With this method, we obtain high accuracy localization for each video frame. From projections of segmented satellite maps on to the camera view, we generate labeled sparse seed points and then using these seed

points we perform a multi-label MRF segmentation for the final dense labeling. We show our results on monocular camera sequences of handheld and street scenes.

6.1 Motivation

Densely labeling a given video in to multiple semantic labels is a complex task that can have uses in several fields for understanding the environment, decision-making and planning. Video segmentation and labeling is well studied [43], [45], [94], [95] as an extension of image segmentation and labeling with the use of additional temporal continuity.

We are approaching the video segmentation problem from a completely new angle, where most of the segmentation and labeling is done on the aerial maps of the area and then projected on to the video frame, Figure 6.1. With our approach, we can achieve high accuracy video segmentation and semantic labeling without learning label classes on the camera view. We only target outdoor videos for our segmentation approach, since we are making use of aerial maps of the area.

To label pixels in the video frames we localized each frame precisely on the satellite map and projected labeling information from the labeled satellite map of the area to the video frame. This method generates sparse labeled points on the video frame, which then can be used as prior labeling for a denser multi-label segmentation step.

Precise global localization is essential in registering the camera images and satellite maps. Since GPS information for a video may not always be present or be sufficiently

accurate, in our framework we perform visual localization from satellite maps with map matching we present in Chapter 4 and Chapter 5.

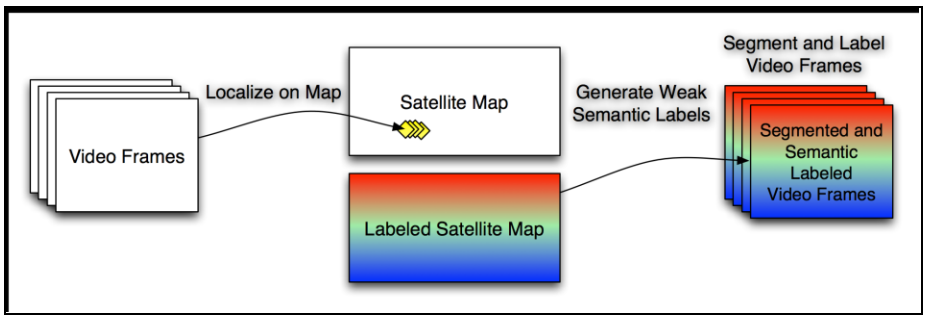


Figure 6.1 – Video segmentation and semantic labeling from satellite maps. Video frames are localized on the map, satellite map labeled with semantic class labels, scene points on frames are weakly labeled by projecting information from the labeled satellite map.

Satellite image segmentation methods presented earlier in Chapter 2 can be used to obtain semantic classes from satellite maps. After 3D scene building step is performed using SfM methods, the estimated sparse scene points can be labeled using satellite maps.

When accurate localization is present, sparse scene points lying on or close to ground plane are directly labeled by projecting the labeled satellite maps. Points that do not belong to ground plane require more indirect approach. These points are tracked throughout the video and clustered in to objects in 3D world coordinates. These objects are then projected to the map and labeled and their map locations are compared to labeled regions on the maps.

Finally, the labeled sparse points on the video frames are used as seeds to a per-frame multi-label MRF classifier in order to obtain the dense segmentation of the video.

Our proposed satellite image based video segmentation framework can be used standalone to segment given user videos or segment images in real-time. Alternatively, the framework can be used in combination with some learning based video and image

segmentation methods to improve their results. In video segmentation and labeling from images, some information may not be inferred from the camera view itself and map information can help disambiguate these cases.

Real-time video segmentation combined with accurate localization of frames has appealing applications that were not possible before. An important application is mobile markerless augmented reality. Augmented reality is bringing virtual augmented objects in to a view of real-world environments. With labeling, localization and map registration, we can create augmented reality applications that attach information tied to satellite maps to the pixel locations of the objects on the video. For example, in navigation applications important buildings can be highlighted, virtual street address labels can follow locations correctly and exact distances to objects can be marked on video.

We can use segmented frames and map localization of 3D objects in order to further augment the video with other useful information. For example, we can show the names and addresses of the buildings in view, mark a particular building, show how tall trees are or display how far a parked vehicle is on the video. We can attach this information to the pixel locations of the objects on the video to generate labels following the real world objects in the video.

6.2 Related Work

Pedestrian detection from live videos became a topic of interest with the advances in part based object detection [37] and HOG descriptors [38]. At the same time, successful

image segmentation methods like super pixel analysis, Conditional Random Fields (CRF) [39] and Markov Random Fields (MRF) [40], and methods to effectively solve these problems like GraphCut [41], [42] are applied to image and video segmentation area [43]-[45]. In video segmentation, frame continuity through temporal label propagation algorithms [95] are employed and understanding the 3D scene structure [94] also played an important role in generating temporally consistent labels throughout the whole sequence.

In this chapter, although we are presenting a very different approach to video segmentation, we are making use of some of the bases these methods. For our dense segmentation step, we use CRF's and employ GraphCut method to segment frames densely from known sparse labels. We also follow a similar approach to methods that make use of 3D scene structure.

6.3 Sparse Semantic Label Generation

After correct localization is achieved, we propagate the map segmentation results to image frames as labeled sparse points. In order to project information coming from segmented satellite maps in to the camera view, we generate the 3D scene points and mark these points with a label from the segmented satellite map. For stereo scenes, every pixel can be precisely mapped to satellite map coordinates and hence can be labeled with the label appearing in the map. For monocular sequences, point projections are done differently for points on the ground plane and point that are above the ground plane.

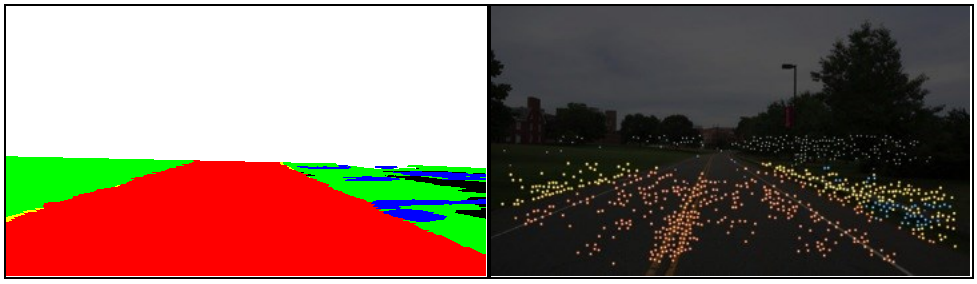
To label the points on the ground plane, we can project the satellite map densely on to the camera view and label scene points with their corresponding map labels. To label the points above the ground plane, we cannot use the image projection that converts satellite image to perspective view, since these points do not fit in to the transformation. In this case, we project the 3D points on to the map and label their classes by searching the closest possible labeled regions on the map.

Due to localization errors, there is a possibility of positional mismatch between projected map and actual camera image. There is also the possibility of object mismatches, for example a car appearing on the map may not appear on the camera image or a part of the map may be wrongly labeled. Because of these reasons, we label the sparse pixel labels we generate at this step as weak labels rather than hard labels. Furthermore, since the segmented map contains class probabilities, we generate class label probabilities for image feature points (i.e. a pixel belonging to multiple classes with some probabilities) instead of hard class membership labels (i.e. a pixel belonging to a single class).

6.3.1 Label Generation for Ground Plane Points

Since points belonging to the ground plane have a correct projection to map, they can be labeled from the projected labeled map image, Figure 6.2 (b). Labeled map contains probabilities of each pixel being a member of each class. We only propagate probabilities for planar object classes (e.g. road, grass, sidewalk, etc.). Points that have their highest probability class belonging to one of the 3D object classes (e.g. tree, building, and car)

are removed. This way we make sure that using the planar projective transformation we do not propagate probabilities for non-planar objects.



(a) (b)

Figure 6.2 – Sparse semantic label for ground points propagated from segmented satellite map. (a) Labeled map projected to perspective view. Parts that are not projected are marked with hatching. (b) Semantically labeled ground points in camera view. Red shows road, green shows grass.

6.3.2 Label Generation for 3D Points

In order to semantically label feature points that are not on the ground plane, we have to determine their locations on the map, since we cannot use the plane equation for these points. Scene modeling step gives us the 3D world locations of each point. We can project the 3D coordinates on to the 2D map plane to obtain their map location. After this, we can assign probabilities of class memberships to each 3D point. From the scene-modeling step, we also know the corresponding pixel location of 3D points on camera frame images. Thus, we can label frame pixels for feature points on 3D objects in the scene.

For the i[th] 3D point in world coordinates (x_i, y_i, z_i), there are n_i pixels with pixel coordinates $(u_{i,j}, v_{i,j})$; for $j = 1 \dots n_i$:

$$(x_i, y_i, z_i) \equiv (u_{i,1}, v_{i,1}), (u_{i,2}, v_{i,2}), \dots, (u_{i,n_i}, v_{i,n_i}) \tag{14}$$

Projection of a 3D point on to the map is its projection on to the x-y plane:

$$(x_i, y_i, z_i) \rightarrow (x_i, y_i) \tag{15}$$

Probability of a feature point pixel $(u_{i,j}, v_{i,j})$ on the camera frame image of belonging to class c out of possible k classes is the probability of the corresponding projected map point belonging to class c .

$$P(c \mid u_{i,j}, v_{i,j}) = P(\mathrm{c} \mid x_i, y_i) \tag{16}$$

Since we are dealing only with 3D object points, this time we only use probabilities for the non-planar classes (e.g. tree, building, and car).

In the outdoor scenes since sky rarely has trackable feature points, when necessary we also add few fixed points in the sky as hard labels for sky class.

6.4 Semantic Video Segmentation and Labeling from Weak Labels

Starting from the weak labels on the frame as sparse priors, we can achieve dense segmentation of the frame. On to each frame in the sequence, we put the weak labels for ground features (e.g. road, grass, sidewalk, etc.) and weak labels for 3D object features (i.e. building, tree, car, etc.). We also include four sky points with fixed positions, to have some sample points from the sky. In case a tree or a building covers some a part of the sky such that some of these four points do not belong to sky anymore, we do not label them as sky.

Given probabilistic sparse labels on an image, we can use several segmentation and labeling methods. We posed this single image multi-label segmentation and labeling problem as a multi-label Graph Cut problem with given probabilistic label priors. Due to its high efficiency compared to other methods, we solve this problem with MRF Alpha Expansion method [41], [107] to perform our dense multi-label classification. Given prior sparse points with labels, this method densely labels each pixel using pixel similarities and neighborhood information. The method also treats labels as weak labels and the dense labeling results may not exactly follow the sparse priors given to the method.

In order to increase the temporal continuity of the labels, on top of these results other methods can also be applied.

6.5 Video Augmentation on Globally Localized and Labeled Videos

After localization, dense segmentation and labeling of the video frames, we can use this information to augment several types of information to the video that is not normally possible with other segmentation or labeling schemes. With the use of labeled maps and 3D scene structure, we can obtain and augment interesting information about objects like buildings, trees, sidewalks or crosswalks on to the video frames. Many different and exciting applications can be built with this type of augmentation and we like to share some of the ideas that we came up with. We also demonstrate the building highlighting and labeling idea to show the possibilities of augmentation.

6.5.1 Sidewalk and Crosswalk Highlighting on Video

As a pedestrian navigation aid, walkable sidewalks and crosswalks can be highlighted on video frames. With the help of walking directions views of map providers, sidewalk and crosswalk segmentation from satellite maps can be achieved. Sidewalks can be easily highlighted with a different color and crosswalks can be pointed out with 3D arrows in the videos.

6.5.2 Road Enhancement, Highlighting and Labeling

As a driving assistance and navigation tool, contrast and brightness of road pixels on the video frames can be enhanced, roads to navigate can be highlighted and road names can be put on top of the roads on the video frame. Although road labeling is often performed on online maps and street-view images (e.g. Google Street View and Bing Street Side), the information that is used in these cases only depends on map and GPS information, where with the help of our method this information can be assigned to image pixels and image regions.

6.5.3 Building Highlighting and Labeling on Video

As another pedestrian navigation aid, a particular building can be marked, highlighted and/or labeled on video frames. Nametags for buildings on a campus can be put exactly on top of the buildings in the video. As opposed to regular (GPS based) augmented reality applications that can augment only the direction and distance of a building, here with the help of accurate localization and map labeling the building pixels can be altered and any desired effect can be created. For example, while rest of the frame is dimmed, the target building can be made brighter and painted red. We show the necessary information to build this augmentation.

Figure 6.3 – Building names overlayed on the map. Segmented buildings are labeled with different color and their names and addresses obtained from reverse-geocoding service are shown.

In order to uniquely identify each building on the map and to augment building names and addresses on the camera frames, we automatically can extract information about buildings from Google Maps Figure 6.3. For each separate building segmented on the map, we query Google Maps reverse-geocoding service with the building center coordinate. Center coordinate of a building is obtained by converting the centroid pixel location to Latitude-Longitude coordinates, where coordinates of corners of the map are known. For some buildings, Google Maps have both name and address information, where for others it only has the address. Whenever the building name is not available, we substitute the short address for the building name (e.g. 174 Frelinghuysen Rd.).

6.6 Experimental Results

Our datasets are composed of frames from videos taken from a moving platform with mono camera setup. Camera motion is unknown, but it is assumed that the camera moves roughly in the direction of its optical axis, i.e. towards what it sees. In addition, we assume that the camera optical axis is roughly parallel to the local ground plane and the camera motion is on the ground plane. We still allow localized camera shakes and non-flat or non-parallel surfaces like hills or ramps as ground planes. Videos can be taken from a car traveling on a road or a person holding a handheld video recording device.

6.6.1 Campus Road Dataset

This dataset is captured with a video camera mounted on a car driving on the road. Dataset is on a campus road that is not much curved. There are distant buildings and road is mostly surrounded with trees and grass areas. Dataset covers 300 m distance on the road and has 1920x1080 pixel frames at 60 fps. For our experiments frames rates at 12 fps are used Figure 6.4.

Figure 6.4 – Campus road dataset sample frames.

6.6.2 Video Segmentation Results

Video segmentation results are shown in Figure 6.5 with sparse weak labels and dense segmentation results. Although some segmentation errors can be observed in the results, considering that these results have no training for the given classes in the images these are very promising results. In order to evaluate the per frame accuracy of the method, no temporal continuity criteria is applied and all frames are segmented independently.

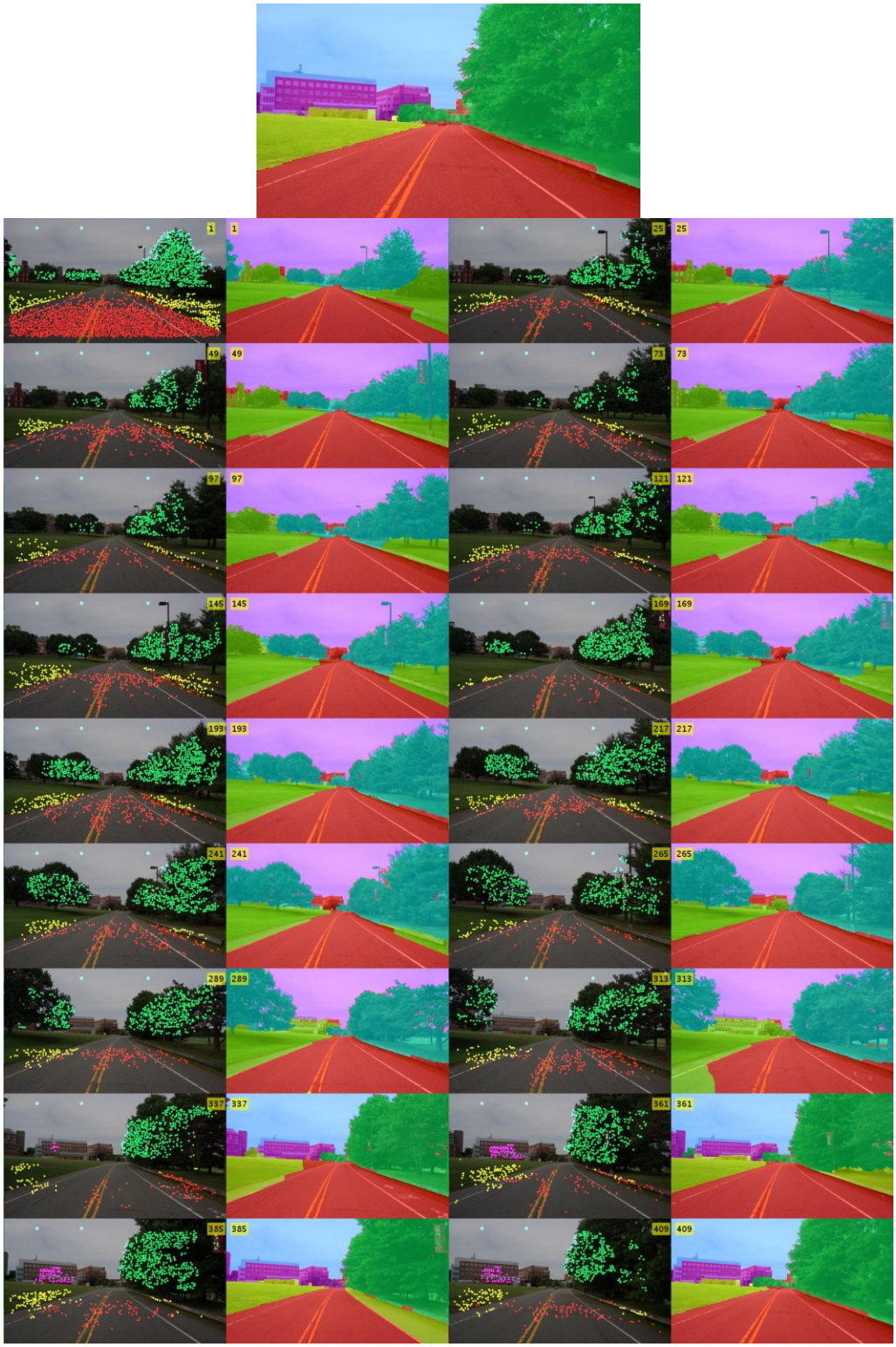


Figure 6.5 – Video segmentation results for campus road dataset. Sparsely labeled points and dense segmentation results are shown for selected frames. Planar points for road (red) and grass (yellow) and 3D points for tree (green) and building (magenta) are shown. Sky pixels are marked with teal.

6.7 Conclusions

In this chapter, we present a novel approach to dense semantic video segmentation and labeling that propagates labeling information from satellite maps on to the view of the video.

Although here we present a completely new class of video segmentation techniques, it can also be combined with the existing learning-based techniques to improve their results and making their segmentation more consistent with the known classes on the satellite map. This way, shortcomings of both methods can be avoided.

Results of the algorithm are segmented video frames with one to one matching to their satellite map location. These results can be used in many different applications, including markerless augmented reality applications on mobile phones.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In Chapter 2 of this thesis, I show a multi-class satellite image segmentation method for road, sidewalk, crosswalk, building, grass and tree classes. In this method, I made use of class-specific structural properties to enhance map segmentation. Furthermore, I made use of inter-class relations to disambiguate classes and enhance map segmentation.

In Chapter 2, I also describe a complete framework to automatically complete sidewalks in satellite images occluded by tree regions. I also present a method that detects and validates zebra crossings from satellite images, by combining appearance information and sidewalk and road segmentations.

In Chapter 3, I explain Hierarchical Semantic Hashing, an efficient geo-localization method on satellite maps that uses layout of semantic map elements. This method can scale to large number of semantic elements with linear storage requirements. Method can perform large scale localization very efficiently, orders of magnitude faster than image comparison methods. With experimental results, I show that Hierarchical Semantic Hashing can be used to efficiently localize a given aerial image in a large city area by using buildings as the semantic elements.

In Chapter 4 and 5, I present methods to localize a moving platform using monocular or stereo camera images taken from the platform by performing satellite map matching. In these methods, I generate accurate top view images of the area using perspective camera images from monocular or stereo cameras, in presence of moving objects and occluders. I developed a Bayesian tracking method to accurately localize a given set of images or a video by matching generated top view images to satellite images. As an additional algorithm, camera images are matched to satellite images projected on to camera view to further improve visual localization accuracy.

In Chapter 4, localization results of a vehicle on the road and robot travelling on sidewalks using stereo camera images and satellite map matching are given. In Chapter 5, localization results for vehicles with monocular camera images are given, where satellite map matching to top view images and matching of camera images to perspective projected satellite images are used.

In Chapter 6, I present an accurate video segmentation method that segments the video pixels densely in to multiple classes without performing training on video frames. This method propagates labels of feature points from segmented satellite images on to video frames.

7.2 Future Work

Localization by Semantic Hashing: Inclusion of more semantic features like building roof colors, distances to roads and building shapes will be investigated to

improve the uniqueness of generated hash keys and thus further decreasing the query time.

Localization by from Ground-Camera Video: Work for visual localization will focus on intensive evaluation of the algorithms via more test sets, where longer paths, loops, denser traffic and changing vehicle speeds exist. We would like to establish measures on rural and urban scenarios for the evaluation of the framework. We also would like to work on the applications of the framework on to real-time systems; especially to a driving-assistance application using augmented reality, where reconstructed 3D environment is rendered on to real world images in order to improve the awareness of the driver.

In this framework we did not address the initial localization problem, which we left as a future research direction. A modified version of our localization framework can be used to solve the initial localization problem.

Semantic video segmentation: Improving the temporal continuity among frame segmentations will be one of the future works. Real-time implementation and full offline video segmentation are among our future goals.

Bibliography

[1] G. Maderlechner, and H. Mayer, "Automated acquisition of geographic information from scanned maps for GIS using frames and semantic networks" Proceedings of the 12th IEEE International Conference on Pattern Recognition (IAPR), vol 2, 1994.

[2] D. Guo, A. Weeks, and H. Klee, “Segmentations of road area in high resolution images”, Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, vol. 6, pp. 3810–3813, 2004.

[3] G. Pacher, S. Kluckner, and H. Bischof, “An improved car detection using street layer extraction”, Proceedings of Computer Vision Winter Workshop, pp. 1–8, 2008.

[4] V. Mnih and G. E. Hinton, “Learning to detect roads in high-resolution aerial images”, Proceedings of the 11th European Conference on Computer Vision (ECCV), 2010.

[5] J. B. Mena, “State of the art on automatic road extraction for GIS update: a novel classification”, Pattern Recognition Letters, vol. 24, no. 16, pp. 3037–3058, 2003.

[6] N. Shorter and T. Kasparis, “Automatic vegetation identification and building detection from a single nadir aerial image”, Remote Sensing, vol. 1, no. 4, pp. 731–757, 2009.

[7] M. Persson, M. Sandvall, and T. Duckett, “Automatic building detection from aerial images for mobile robot mapping”, Proceedings of the IEEE International Computational Intelligence in Robotics and Automation Symposium (CIRA), pp. 273–278, 2005.

[8] W. Willuhn and L. Van Gool, “Building reconstruction from aerial images using efficient semiautomatic building detection”, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Hannover Workshop, 2005.

[9] J. Xiao, “Automatic building detection using oblique imagery”, 2013.

[10] P. Saeedi and H. Zwick, “Automatic building detection in aerial and satellite images”, 10th International Conference on Control, Automation, Robotics and Vision (ICARCV), pp. 623–629, 2008.

[11] C. Burnett and T. Blaschke, “A multi-scale segmentation/object relationship modelling methodology for landscape analysis”, Ecological Modelling, vol. 168, no. 3, pp. 233–249, 2003.

[12] A. Carleer and O. Debeir, “Assessment of very high spatial resolution satellite image segmentations”, Photogrammetric Engineering and Remote Sensing, vol. 71, no. 11, pp. 1285–1294, Nov. 2005.

[13] V. Smith, J. Malik, and D. Culler, “Classification of sidewalks in street view images”, International Green Computing Conference (IGCC), pp. 1–6, 2013.

[14] A. Chand and S. Yuta, “Navigation strategy and path planning for autonomous road crossing by outdoor mobile robots”, 15th International Conference on Advanced Robotics (ICAR), 2011.

[15] D. Castells and J. Rodrigues, “Sapientia: Obstacle detection and avoidance on sidewalks”, In Proc. Int. Conf. on Computer Vision Theory and Applications (VISAPP) Angers, France, 2010.

[16] Q. Mühlbauer, S. Sosnowski, and T. Xu, "The autonomous city explorer project: Towards navigation by interaction and visual perception", Proceedings of the International Workshop on Cognition for Technical Systems (CoTeSys), 2009.

[17] Q. Lin, Y. Han, and H. Hahn, "Vision-based navigation using top-view transform and beam-ray model", International Conference on Advanced Computer Science and Information System (ICACSIS), 2011.

[18] H. Lee, "Application of machine vision techniques for the evaluation of highway pavements in unstructured environments", Fifth International Conference on Advanced Robotics, (ICAR), pp. 1425–1428 vol. 2, 1991.

[19] T. Senlet and A. Elgammal, "Segmentation of occluded sidewalks in satellite images", IEEE International Conference on Pattern Recognition (ICPR), pp. 805–808, 2012.

[20] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software", SIGKDD Explor. Newsl., vol. 11, no. 1, pp. 10–18, Nov. 2009.

[21] U. Pilz, W. Gropengießer, F. Walder, J. Witt, and H. Werner, "Quadrocopter Localization Using RTK-GPS and Vision-Based Trajectory Tracking", 4th International Conference in Intelligent Robotics and Applications (ICIRA), Aachen, Germany, vol. 7101, no. 2, pp. 12–21, 2011.

[22] M. W. Achtelik, S. Lynen, S. Weiss, L. Kneip, M. Chli, and R. Siegwart, "Visual-inertial SLAM for a small helicopter in large outdoor environments", IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2651–2652, 2012.

[23] T. Botterill, S. Mills, and R. Green, “Bag-of-words-driven, single-camera simultaneous localization and mapping”, Journal of Field Robotics, vol. 28, no. 2, pp. 204–226, Oct. 2010.

[24] T. Botterill, R. Green, and S. Mills, “A bag-of-words speedometer for single camera SLAM”, 24th International Conference on Image and Vision Computing New Zealand, pp. 91–96, 2009.

[25] P. Quelhas, F. Monay, J.-M. Odobez, D. Gatica-Perez, and T. Tuytelaars, “A Thousand Words in a Scene”, IEEE Trans. Pattern Anal. Mach. Intell., vol. 29, no. 9, pp. 1575–1589, Sep. 2007.

[26] D. Filliat, “A visual bag of words method for interactive qualitative localization and mapping”, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp. 3921–3926, 2007.

[27] B. Barshan and H. F. Durrant-Whyte, “Inertial navigation systems for mobile robots”, IEEE Transactions on Robot Automation, vol. 11, no. 3, pp. 328–342, Jun. 1995.

[28] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte, “A high integrity IMU/GPS navigation loop for autonomous land vehicle applications”, IEEE Transactions on Robotics and Automation, vol. 15, no. 3, pp. 572–578, 1999.

[29] H. Lategahn, A. Geiger, and B. Kitt, “Visual SLAM for autonomous ground vehicles”, IEEE International Conference on Robotics and Automation (ICRA), pp. 1732–1737, 2011.

[30] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem", Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton, Canada, vol. 1, p. 3, 2002.

[31] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM", IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 29, no. 6, pp. 1052–1067, 2007.

[32] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle adjustment—a modern synthesis", Vision Algorithms: Theory and Practice, pp. 153–177, 2000.

[33] T. S. Huang and A. N. Netravali, "Motion and structure from feature correspondences: A review", Proceedings of the IEEE, vol. 82, no. 2, pp. 252–268, 1994.

[34] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry", Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, 2004.

[35] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme", IEEE Intelligent Vehicles Symposium (IV), pp. 486–492, 2010.

[36] B. Kitt, F. Moosmann, and C. Stiller, "Moving on to dynamic environments: Visual odometry using feature classification", IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 5551–5556, 2010.

[37] B. Leibe, A. Leonardis, and B. Schiele, "Robust Object Detection with Interleaved Categorization and Segmentation", Int Journal of Computer Vision, vol. 77, no. 1, May 2008.

[38] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection”, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 886–893, 2005.

[39] J. Lafferty, A. McCallum, and F. C. Pereira, “Conditional random fields: Probabilistic models for segmenting and labeling sequence data”, Proceedings of the Eighteenth International Conference on Machine Learning (ICML), pp. 282–289, 2001.

[40] Y. Boykov, O. Veksler, and R. Zabih, “Markov random fields with efficient approximations”, Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 648–655, 1998.

[41] Y. Boykov, O. Veksler, and R. Zabih, “Fast approximate energy minimization via graph cuts”, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 23, no. 11, pp. 1222–1239, 2001.

[42] Y. Boykov and G. Funka-Lea, “Graph Cuts and Efficient N-D Image Segmentation”, International Journal on Computer Vision, vol. 70, no. 2, Nov. 2006.

[43] C. Wojek and B. Schiele, “A Dynamic Conditional Random Field Model for Joint Labeling of Object and Scene Classes”, European Conference on Computer Vision (ECCV), Berlin Heidelberg, vol. 5305, no. 54, pp. 733–747, 2008.

[44] B. Micusik and J. Košecká, “Semantic segmentation of street scenes by superpixel co-occurrence and 3D geometry”, Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on, pp. 625–632, 2009.

[45] B. Micusik, J. Košecká, and G. Singh, "Semantic parsing of street scenes from video", The International Journal of Robotics Research, vol. 31, no. 4, pp. 484–497, Apr. 2012.

[46] T. Senlet and A. Elgammal, "A framework for global vehicle localization using stereo images and satellite and road maps", IEEE International Conference on Computer Vision Workshops (ICCVW), Barcelona, pp. 2034–2041, 2011.

[47] T. Senlet and A. Elgammal, "Satellite image based precise robot localization on sidewalks", IEEE International Conference on Robotics and Automation (ICRA), pp. 2647–2653, 2012.

[48] T. Senlet, T. El-Gaaly, and A. Elgammal, "Hierarchical Semantic Hashing: Visual Localization from Buildings on Maps", International Conference on Pattern Recognition (ICPR), 2014.

[49] M. Montemerlo, S. Thrun, H. Dahlkamp, D. Stavens, and S. Strohband, "Winning the DARPA Grand Challenge with an AI robot", Proceedings of the AAAI National Conference on Artificial Intelligence, vol. 21, no. 1, p. 982, 2006.

[50] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. N. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. Rybski, B. Salesky, Y.-W. Seo, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson, "Autonomous driving in urban environments: Boss and the Urban Challenge", Journal of Field Robotics, vol. 25, no. 8, Aug. 2008.

[51] Google, Google Static Maps API, [Online], http://code.google.com/apis/maps/documentation/staticmaps, Accessed [06-May-2012].

[52] M. Noda, T. Takahashi, D. Deguchi, I. Ide, H. Murase, Y. Kojima, and T. Naito, "Vehicle ego-localization by matching in-vehicle camera images to an aerial image", Asian Conference on Computer Vision Workshops (ACCVW), pp. 163–173, 2011.

[53] N. Mattern, R. Schubert, and G. Wanielik, "High-accurate vehicle localization using digital maps and coherency images", IEEE Intelligent Vehicles Symposium (IV), pp. 462–469, 2010.

[54] H. Hirschmuller, "Stereo Processing by Semiglobal Matching and Mutual Information", IEEE Trans. Pattern Anal. Mach. Intell., vol. 30, no. 2, pp. 328–341, 2008.

[55] A. Geiger, M. Roser, and R. Urtasun, "Efficient large-scale stereo matching", Asian Conference on Computer Vision (ACCV), pp. 25–38, 2010.

[56] PointGrey, Triclops SDK, [Online], http://www.ptgrey.com/products/triclopsSDK/index.asp, Accessed [06-May-2012].

[57] Willow Garage, OpenCV, [Online], http://opencv.willowgarage.com, Accessed [06-May-2012].

[58] PointGrey, Bumblebee2 CCD FireWire Camera, [Online], http://www.ptgrey.com/products/bumblebee2/bumblebee2_stereo_camera.asp, Accessed [06-May-2012].

[59] A. Geiger, J. Ziegler, and C. Stiller, "StereoScan: Dense 3D reconstruction in real-time", IEEE Intelligent Vehicles Symposium (IV), pp. 963–968, 2011.

[60] S. Vedula, S. Baker, P. Rander, R. Collins, and T. Kanade, “Three-dimensional scene flow”, IEEE International Conference on Computer Vision (ICCV), vol. 2, pp. 722–729, 1999.

[61] M. Noda, T. Takahashi, D. Deguchi, I. Ide, H. Murase, Y. Kojima, and T. Naito, “Road image update using in-vehicle camera images and aerial image”, IEEE Intelligent Vehicles Symposium (IV), pp. 460–465, 2011.

[62] G. Lidoris, K. Klasing, A. Bauer, T. Xu, K. Kuhnlenz, D. Wollherr, and M. Buss, “The autonomous city explorer project: Aims and system overview”, IROS'07, pp. 560–565, 2007.

[63] S. Yuta, M. Mizukawa, H. Hashimoto, H. Tashiro, and T. Okubo, “An open experiment of mobile robot autonomous navigation at the pedestrian streets in the city—Tsukuba challenge”, Int. Conf. on Mechatronics and Automation, pp. 904–909, 2011.

[64] H. Iikura, K. Kobayashi, and K. Watanabe, “Development of Intelligent wheelchair employing omni-directional camera and slightly tilted laser rangefinder”, SICE 2004 Annual Conference, vol. 3, pp. 1989–1993 vol. 3, 2004.

[65] H. Mori, S. Kotani, and N. Kiyohiro, “Human interface of a robotic travel aid”, Proceedings of the 3rd IEEE International Workshop on Robot and Human Communication (RO-MAN), Nagoya, pp. 90–94, 1994.

[66] S. Shair, J. H. Chandler, V. J. González-Villela, R. M. Parkin, and M. R. Jackson, “The Use of Aerial Images and GPS for Mobile Robot Waypoint Navigation”, IEEE/ASME Trans. Mechatron., vol. 13, no. 6, pp. 692–699, 2008.

[67] M. Montemerlo and S. Thrun, “Simultaneous localization and mapping with unknown data association using FastSLAM”, Proceedings of IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 1985–1991 vol. 2, 2003.

[68] H. Casarrubias-Vargas, A. Petrilli-Barcelo, and E. Bayro-Corrochano, “EKF-SLAM and Machine Learning Techniques for Visual Robot Navigation”, 20th International Conference on Pattern Recognition (ICPR), pp. 396–399, 2010.

[69] O. Pink, “Visual map matching and localization using a global feature map”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pp. 1–7, 2008.

[70] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (SLAM): part II”, Robotics & Automation Magazine, IEEE, vol. 13, no. 3, pp. 108–117, 2006.

[71] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part I”, Robotics & Automation Magazine, IEEE, vol. 13, no. 2, pp. 99–110, 2006.

[72] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography”, Communications of the ACM, vol. 24, no. 6, pp. 381–395, 1981.

[73] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking”, Signal Processing, IEEE Transactions on, vol. 50, no. 2, pp. 174–188, 2002.

[74] Y. Morales, E. Takeuchi, A. Carballo, W. Tokunaga, H. Kuniyoshi, A. Aburadani, A. Hirosawa, Y. Nagasaka, Y. Suzuki, and T. Tsubouchi, “1Km autonomous robot

navigation on outdoor pedestrian paths ‘running the Tsukuba challenge 2007’”, Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), , pp. 219–225, 2008.

[75] T. Shioyama, H. Wu, Y. Nishibe, N. Nakamura, and S. Kitawaki, “Image analysis of crosswalk”, Proceedings of International Conference on Image Analysis and Processing, pp. 168–173, 2001.

[76] L. Breiman, “Random forests”, Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.

[77] A. Criminisi, P. Pérez, and K. Toyama, “Object removal by exemplar-based inpainting”, Proc. IEEE Computer Soc. Conf. on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. II–721–II–728 vol. 2, 2003.

[78] J. P. Lewis, “Fast normalized cross-correlation”, Vision Interface, vol. 10, no. 1, 1995.

[79] R. Datta, D. Joshi, J. Li, and J. Z. Wang, “Image retrieval: Ideas, influences, and trends of the new age”, ACM Computing Surveys (CSUR), vol. 40, no. 2, pp. 1–60, Apr. 2008.

[80] A. Pronobis, “Semantic mapping with mobile robots”, KTH Royal Institute of Technology, 2011.

[81] W. Jiang, K. L. Chan, M. Li, and H. Zhang, “Mapping low-level features to high-level semantic concepts in region-based image retrieval”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 244–249, 2005.

[82] A. R. Zamir and M. Shah, “Accurate Image Localization Based on Google Maps Street View”, European Conference on Computer Vision (ECCV), pp. 255–268, 2010.

[83] C. Wu, F. Fraundorfer, J.-M. Frahm, and J. Snoeyink, “Image localization in satellite imagery with feature-based indexing”, Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Beijing, pp. 197–202, 2008.

[84] Y. Lin and G. Medioni, “Map-enhanced UAV image sequence registration and synchronization of multiple image sequences”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–7, 2007.

[85] Y. B. Yang and H. T. Tsui, “Mobile robot localization by geometric hashing and model-based scene matching”, Proceedings of the 13th International Conference on Pattern Recognition (CVPR), vol. 1, pp. 181–185, 1996.

[86] T. J. Cham, A. Ciptadi, W. C. Tan, M. T. Pham, and L. T. Chia, “Estimating camera pose from a single urban ground-view omnidirectional image and a 2D building outline map”, Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), pp. 366–373, 2010.

[87] I. Rigoutsos and H. J. Wolfson, “Geometric Hashing”, IEEE Computational Science & Engineering, vol. 4, no. 4, p. 9, 1997.

[88] H. Van Dijck, M. Korsten, and F. Van Der Heijden, “Robust 3-dimensional object recognition using stereo vision and geometric hashing”, Proceedings of International Conference on Image Processing, vol. 1, pp. 329–332, 1996.

[89] J.-J. Liu and R. Hummel, “Geometric hashing with attributed features”, Proceedings of the CAD-Based Vision Workshop, pp. 9–16, 1994.

[90] F. C. Tsai, “A probabilistic approach to geometric hashing using line features”, Computer Vision and Image Understanding, vol. 63, no. 1, pp. 182–195, 1996.

[91] B. P. Olsen, “Automatic change detection for validation of digital map databases”, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. 34, pp. 569–574, 2004.

[92] N. Ekhtari, M. R. Sahebi, M. V. Zoej, and A. Mohammadzadeh, “Automatic building detection from Lidar point cloud data”, Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Beijing, vol. 4, 2008.

[93] City of Seattle, Seattle's Data Site, [Online], https://data.seattle.gov/dataset/2009-Building-Outlines/y7u8-vad7, Accessed [19-Dec-2013].

[94] P. Sturgess, K. Alahari, L. Ladicky, and P. Torr, “Combining appearance and structure from motion features for road scene understanding”, British Machine Vision Conference (BMVC), 2009.

[95] V. Badrinarayanan, I. Budvytis, and R. Cipolla, “Semi-supervised video segmentation using tree structured graphical models”, IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), vol. 35, no. 11, pp. 2751–2764, 2013.

[96] O. Pink, F. Moosmann, and A. Bachmann, “Visual features for vehicle localization and ego-motion estimation”, IEEE Intelligent Vehicles Symposium (IV), pp. 254–260, 2009.

[97] A. M. Muad, A. Hussain, S. A. Samad, M. M. Mustaffa, and B. Y. Majlis, “Implementation of inverse perspective mapping algorithm for the development of an

automatic lane tracking system", IEEE Region Conference TENCON, vol. A, pp. 207–210, 2004.

[98] R. Wallace, K. Matsuzaki, Y. Goto, J. Crisman, J. Webb, and T. Kanade, "Progress in robot road-following", IEEE International Conference on Robotics and Automation (ICRA), vol. 3, pp. 1615–1621, 1986.

[99] M. A. Brubaker, A. Geiger, and R. Urtasun, "Lost! Leveraging the Crowd for Probabilistic Visual Self-Localization", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013.

[100] G. Floros, B. van der Zander, and B. Leibe, "OpenStreetSLAM: Global Vehicle Localization Using OpenStreetMaps", IEEE International Conference on Robotics and Automation (ICRA), 2013.

[101] C. Tomasi and T. Kanade, "Detection and tracking of point features", International Journal of Computer Vision, 1991.

[102] H. Broszio and O. Grau, "Robust estimation of camera parameters pan, tilt and zoom for integration of virtual objects into video sequences", International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging, Santorini, Greece, pp. 15–17, 1999.

[103] H. Hirschmuller, "Accurate and efficient stereo processing by semi-global matching and mutual information", IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 807–814 vol. 2, 2005.

[104] C. Mei, G. Sibley, M. Cummins, P. M. Newman, and I. D. Reid, "A Constant-Time Efficient Stereo SLAM System.", British Machine Vision Conference (BMVC), London, pp. 1–11, 2009.

[105] P. H. S. Torr and A. Zisserman, "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry", Computer Vision and Image Understanding, vol. 78, no. 1, pp. 138–156, Apr. 2000.

[106] A. Li, V. I. Morariu, and L. S. Davis, "Planar Structure Matching under Projective Uncertainty for Geolocation", European Conference on Computer Vision (ECCV), 2014.

[107] A. Delong, A. Osokin, H. N. Isack, and Y. Boykov, "Fast approximate energy minimization with label costs", International Journal of Computer Vision, vol. 96, no. 1, pp. 1–27, 2012.

[108] Viscoda, Voodoo Camera Tracker, [Online], http://www.viscoda.com/index.php/en/products/non-commercial/voodoo-camera-tracker, Accessed [03-Sep-2014].