Approximate Representation of Unknown Objects with a Single-line Scanning Lidar and a Video Camera

Ki Ho Kwak

July, 2012

Electrical and Computer Engineering

Carnegie Mellon University

Pittsburgh, PA 15213

Thesis Committees:

Takeo Kanade

Daniel Huber

Marios Savvides

Robert T. Collins(PSU)

Submitted in partial fulfillment of the requirements

for the degree of Doctor of Philosophy

Abstract

Models are useful for many computer vision tasks, such as object detection, recognition, and tracking. Computer vision tasks must handle situations where unknown objects appear and must detect and track some object which is not in the trained database. In such cases, the system must learn or, otherwise derive, descriptions of new objects.

In this dissertation, we investigate creating a representation of previously unknown objects that newly appear in the scene. The representation is to create a viewpoint-invariant and scale-normalized model approximately describing an unknown object. Those properties of the representation facilitate 3D tracking of the object using 2D-to-2D image matching. The representation has both benefits of an *implicit* model (referred to as a view-based model) and an *explicit* model (referred to as a shape-based model). The object representation is created using multi-modal sensors. We illustrate the benefits of the object representation with two applications: object detection and 3D tracking. We extend the object representation to an explicit model by imposing a shape prior and combining two existing approaches.

Acknowledgements

I am sincerely and heartily grateful to my advisor, Takeo Kanade, for his excellent guidance, caring, patience, and providing me with an excellent atmosphere for doing research. The knowledge and skills that I learned from him will be the guideline in my life.

I would like to thank Daniel Huber, who let me experience the research in the field and practical issues beyond the textbooks, patiently corrected my writing. I would like to thank the other members of my thesis committee, Marios Savvides and Robert Collins for their advice and comments on my thesis.

I would like to thank to the Agency for Defence Development in Korea, that gave me a chance to study in Carnegie Mellon University and supported my research. I also would like to thank to Yong Woon Park and my colleagues in ADD for their belief and support.

Many thanks to my friends in CMU. To Jun Sik Kim, Young Woo Seo and Hernan Badin, for valuable discussions and their advice. To Santosh Kumar Divvala and Paul Vernaza, for sharing the office and their friendship. To my Korean friends in CMU for making my time in graduate school productive and rewarding.

I am very thankful to the understanding and unconditioned support from my lovely wife, Hyun Hee Lim, and my children, Seoyeon and Minsung. I also would like to thank my family in Korea, for their patience and believing in me.

This dissertation is dedicated to my wife and my children, and to all my family.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Imagine vehicles passing an intersection. Your visual system can easily detect the shape and position of the vehicle and persistently track them, and identify the brand and year of the vehicles. However, it is very difficult for a computer because of the problem of variability. A vision system needs to generalize across huge variation in the appearance of an object - due for instance to viewpoint, illumination, scale or occlusion. At the same time, the system must maintain specificity because an object can be recognized at a variety of levels of specificity: a car can be recognized as “a black car” by the color, or “a sedan” by the shape [1].

Models are important for the robustness and efficiency of computer vision applications. One can manage to perform tasks without models, but with models, one can perform tasks better [2]. For example, when recognizing a car in an image, if we have car models which are manually trained or generated, we can recognize the car by searching in the previously constructed database. Furthermore, models are useful for detecting and tracking objects in challenging conditions, such as occlusion, illumination, and viewpoint change. Models are generally built based on prior knowledge about the object appearance, shape, or both, and model learning is accomplished with user supervision.

Many computer vision tasks must handle situations where previously unseen objects appear and must detect and track an object which is not in the trained model. In such cases, the system must learn the new objects or simply derive the descriptions for the objects never seen before [3] [4]. For example, if we trained a perception system with only car and person models, the system can not recognize animals. Therefore, building object models while keeping track of the objects is necessary to accomplish the tasks of computer vision applications.

To build object models online, the first question to be address is how to represent objects. The focus of representation is to describe implicitly or explicitly the geometry and appearance of 3D objects [5], [6]. The implicit representation (referred to as a view-based model) models an object as a set of representative images taken from different viewpoints. The explicit representation (referred to as a view-based model) is to create the 3D geometric shape of the object. The former has the advantage of simple matching to an image at runtime (assuming that other variations like lighting and color can be minimized), but it requires a large number of viewpoints and therefore a lot of data storage. The latter makes storage more efficient, since there is only one model per object, but the model creation is more challenging.

Object representation is influenced by the available sensors. Some sensors are more appropriate for shape representation, and some are appropriate for appearance representation. Vision provides dense data with fast updates, but it does not give a sense of scale. Lidar can give accurate geometric information, but only at sparse locations, and much of the appearance information cannot be captured. The combination of the two sensors gains the benefits of both. The tradeoff is the additional effort needed to calibrate the sensors with respect to each other. It can be solved by finding corresponding features in the lidar and vision data. However, finding such correspondences is difficult because the lidar spot usually cannot be seen directly in the image, since a lidar typically operates outside the visual spectrum. Full 3D sensors are expensive and somewhat bulky, but single line lidars, which provide limited 3D information, are compact and lower cost.

1.2 Related Work

This dissertation work focuses on creating an approximate model of unknown objects online using pervasively combined sensor data. Therefore, it is useful to examine existing object representation methods, online modeling approaches, and existing modeling approaches using sensor fusion techniques.

An object should be defined as something that is of interest for further analysis. The common approach (referred to as the model-based approach) models objects by their shape, appearance, or both. Other ways include detecting and tracking the motion blobs of moving objects without recognition (referred to as the non-model-based approach). The non-model-based approaches have an advantage in which they can detect and track moving objects without prior knowledge. However, they cope poorly with occlusion and cannot re-acquire the tracked object after losing it because they describe objects as moving blobs or points instead of using shape or appearance of the object [7] [8] [9] [10]. However, model-based detection and tracking approaches have achieved more reliable performance in vision-based applications [11] [12] and in lidar-based approaches [13] [14] [15].

A 3D object model is more robust and able to detect and track objects over longer durations than 2D object models. Since the 3D object models contain geometric and appearance information, they have much higher stability with respect to viewpoint change, occlusion, and shape deformation [16] [17]. For example, in [2] [18] [19] [20], the authors used 3D wire-frame vehicle models to track various types of moving vehicles with a stationary camera and showed that the 3D model is more robust in occlusion and viewpoint changes. The 3D object representation can be either implicit (view-dependent or viewer-centered) or explicit (view-invariant or object-centered) [21]. The implicit model features multi-view-based models encoding different views of an object [22]. The explicit model contains the 3D shape [23] [24] or the 3D shape and appearance of an object [25]. In the aforementioned research, the 3D object models are manually created in limited environments. These approaches have failed to detect and track objects previously unseen or not in their training databases.

Once we create 3D models online, it is possible to detect and track objects when they appear again. Online object modeling approaches can be categorized into two types. The first approach is to make a robust and adaptive model varying in shape and appearance online [3] [26]. The second is to build a model of previously unseen objects on-the-fly [27] [4]. Both are identical in terms of making models online. In [27], the authors introduced an online object modeling approach for use in airborne images. They built a set of view-dependent object appearance models adaptively and automatically while tracking an object under different viewing angles. Although they built viewpoint dependent object models, the models do not contain any shape information. In [4], the authors described exemplar-based online object modeling for representing the shape of objects in planar laser scans. They conducted experiments to build models for several types of moving people e.g., pedestrians, skaters, or cyclists in real environments.

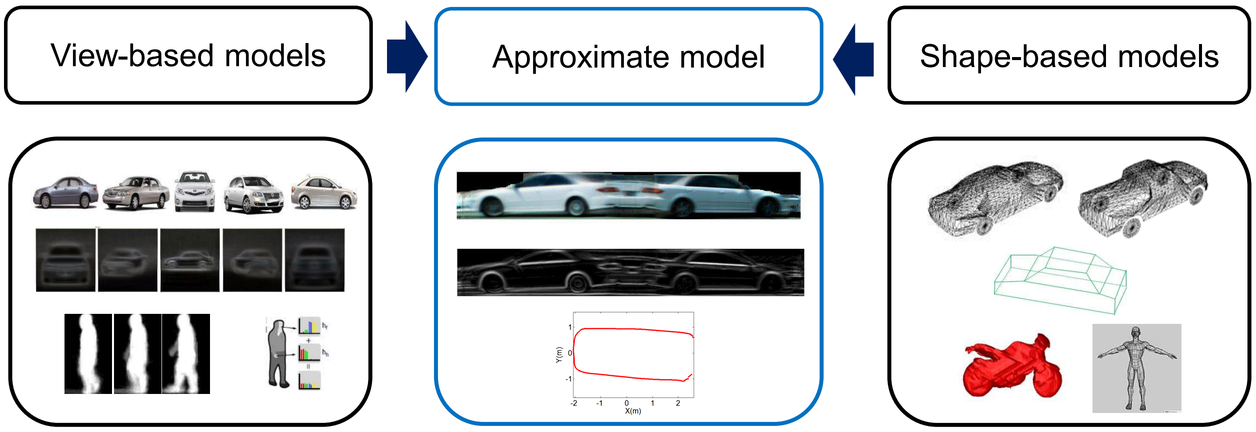


Figure 1.1: Overview of 3D object representation: The implicit 3D model represents objects as a collection of 2D views. The explicit 3D model uses volumetric representation and has an explicit model of the 3D geometry of the object. The approximate model consists of two main models: approximate unwrapped mosaics (color and edge gradient) and an approximate geometry. The approximate model has the benefits of both view-based and shape-based models; that is, we estimate 3D pose of an object by 2D-to-2D matching.

1.3 Dissertation Overview

In this dissertation, we create object models online while tracking. To trade-off between the implicit and explicit representation, we create an approximate model with the geometry and appearance of an object as shown in Figure 1.1. Our sensor configuration is a combination of a single line scanning lidar and a vision sensor.

The organization of this dissertation is as follows: In Chapter 2, we introduce preliminary work for the approximate representation - vertical boundary detection and data association by sensor fusion. We describe the approach of the approximate representation and the performance of detection and tracking with the model in Chapter 3. Approximate modeling of a specific object (e.g., cars) and a method for 3D car shape recovery are introduced in Chapter 4. Finally, we conclude with a summary of this work and suggest future extensions in Chapter 5. In the Appendix, we introduce a new extrinsic calibration algorithm that we developed for combining lidar measurements and imagery.

1.4 Dissertation Contributions

The contributions of this dissertation include approximate representation of unknown objects using lidar measurements and appearance, detection and 3D tracking of the objects, and 3D shape recovery of a specific object, as shown in Figure 1.2. The details of the contributions made are:

Approximate Representation of Unknown Objects

We create a model which is invariant to viewpoint and scale. We first create an approximate geometric model that provides the geometry for an unwrapped mosaic model of appearance. The approximate model consists of an approximate unwrapped mosaic along with an approximate geometry. The unwrapped mosaicing is achieved by creating a fronto-parallel image using the corresponding geometry. The appearance model consists of color and edge gradient mosaics, both created independently and normalized by viewpoint and scale. Using the approximate model, we can obtain the benefits of both implicit and explicit modeling approaches, since we can perform 2D image matching for detecting an object and estimating its 3D pose.

Detection and Tracking using Approximate Model

We build a classifier to distinguish between different objects when we detect an object with the

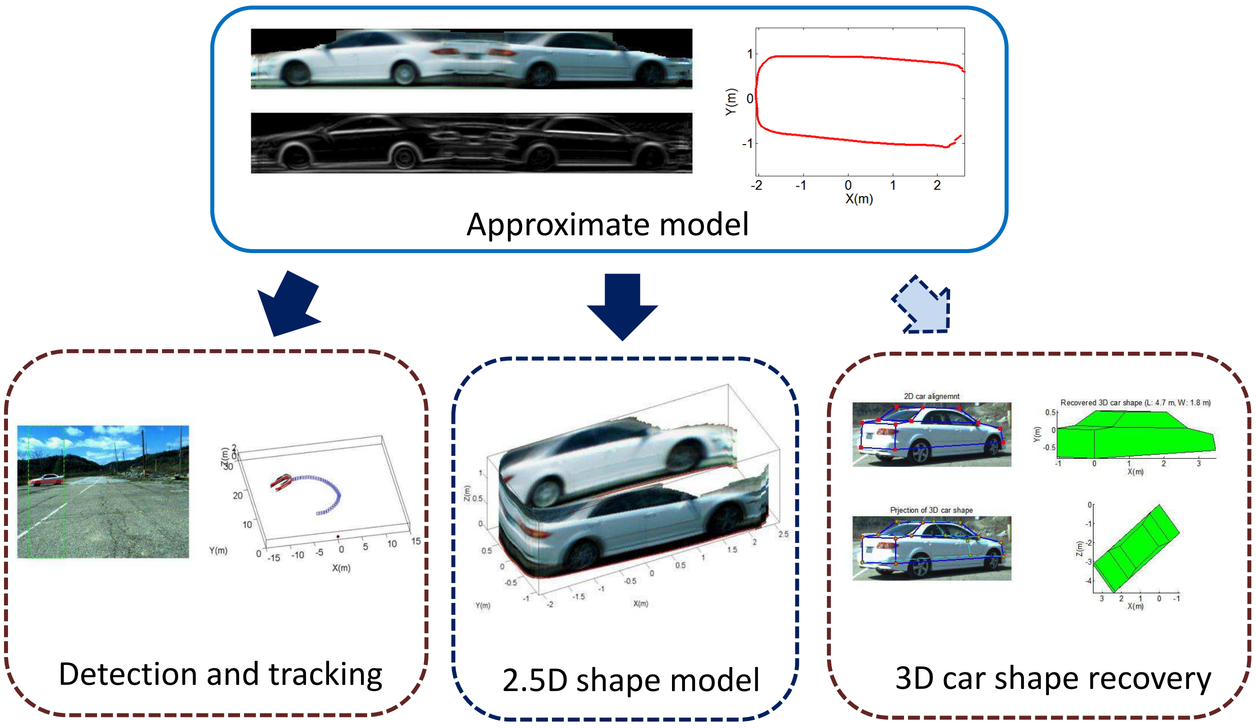


Figure 1.2: Dissertation contributions: We create an approximate representation method for unknown objects. The approximate model estimates the 3D pose of an object using 2D image matching. We recover the 2.5D shape model with the approximate model. We also recover an explicit 3D car shape using the shape prior of a car and the lidar constraints.

approximate model. Each approximate model is used to build a probabilistic model that can be used to assign a probability score to each query object, from which a classifier is obtained. We evaluate the performance of our detector using data sets obtained in outdoor environments.

3D Shape Recovery Imposing Shape Prior

We recover a 3D car shape of real scale by imposing the shape prior of a car and geometric constraints from lidar measurements. We develop an approach that combines three ideas: (i) an existing method for aligning a deformable model of a car to an image [28]; (ii) an algorithm for 3D shape recovery using the Origami world [29]; and (iii) a new method for applying knowledge-based constraints and lidar geometry constraints to recover the geometry of real scale.

Pervasive Sensor Fusion

In order to create the models, we pervasively combine a single-line scanning lidar and vision data. We compute features from the data, and then fuse the information at the feature-level. Many fusion approaches use data from different modalities independently and then fuse the resulting output of the algorithm. This approach has some advantages, since the data from each sensor can be treated independently. However, it does not take advantage of all the available information from a calibrated multi-modal sensor system. The benefit of our approach is that we can integrate fusion throughout the different stages in the process. The feature-level fusion approach can be more robust than the decision-level fusion because both sensors can contribute to the solution at each step in a sequence of operations, whereas decision-level fusion is limited by any one step that performs poorly for a given sensor.

Chapter 2

Object Region Detection and Association

2.1 Introduction

In order to create a model of an object, we first need to find correspondences of the tracked object between frames. Searching for the correspondences across frames is equivalent to tracking the object over time. To search for the correspondences, we first detect the *foreground* in each frame, that is, we should identify the locations of potential object regions in the frame. Hereafter, we refer to the potential object region as an *object region* or an *object image*.

The object region can be estimated by detecting the boundary of an object. An object boundary occurs where one object occludes another from the viewpoint of a sensor. Detecting object boundaries in images is difficult because appearance may change more dramatically within an object than at its boundary. Lidar offers a good alternative to monocular vision approaches for boundary detection. These sensors directly estimate the distance to surfaces in the scene and therefore have the potential to detect depth discontinuities that often appear at object boundaries. Lidars have the advantage of lower uncertainty and reliable performance in untextured regions, generally at the expense of lower angular resolution. However, even the high-precision range measurements of laser scanners do not trivially solve the boundary detection problem.

A common approach for detecting boundaries using range data is to compute differences in the range of adjacent measurements and to declare any range difference above a given threshold to be an object boundary. While this simple thresholding approach can work well for straightforward cases, it does not perform as well in more challenging situations that occur frequently worth addressing. For example, if two objects are close together, they may be incorrectly classified as a single object with no boundary between them. For these challenging cases, we considered whether the information from lidar could be helpful in detecting such boundaries and also whether imagery could be helpful.

Combining both sensors' data can also be beneficial for tracking objects in a scene. It can be awkward to consistently track an object using vision-based tracking because of information loss [15]. The vision-based tracking can be difficult to consistently track an object because of loss of information caused by projection of the 3D world on a 2D image, appearance change by viewpoints and scene illumination, and partial and full occlusions. On the other hand, lidar-based tracking can struggle to maintain correspondences across frames due to sparse depth information. Properly combining individual sensor's measurements has a benefit because one sensor's data can provide complimentary information to the other sensor.

Vision-based tracking has been studied in numerous ways for many decades. Most of the approaches are divided into three major categories based on object representation and image features:

(i) discrete feature tracker, (ii) contour tracker (e.g. edge-gradient), and (iii) region-based tracker. The tracking is performed in two main stages. First, each tracker represents objects as discrete features (e.g. points and lines), object boundaries (e.g. edge-gradient), or simple geometric shape (e.g. rectangular or elliptical patch). Second, the tracker establishes correspondences between the object instances across frames [30]. Tracking objects with only visual information remains a difficult problem within the field of computer vision because of viewpoint and illumination change [30] [31].

Lidar can offer a good alternative to vision-based approaches for object tracking. The sensor directly estimates the distance to surfaces in the scene and therefore has the potential to detect depth discontinuities that often appear at object boundaries. Lidars have the advantage of lower uncertainty and reliable performance in untextured regions, generally at the expense of lower angular resolution [13]. However, even the high-precision depth measurements of lidars do not trivially solve the object tracking problem. Moreover, if the lidar scans a single line, it is more difficult than a multi-line or a 3D scanning lidar because we cannot have depth information at object regions except in the scanning direction

We propose a region-based tracking approach by pervasive sensor fusion of a single line scanning lidar data and imagery. The methodology for performing object tracking using lidar data and imagery can be summarized in two main stages: (i) vertical boundary detection of objects in a single frame and (ii) data association of the same object across frames.

2.2 Related Work

There is relatively little research on explicitly detecting object boundaries with sensor fusion. However, the boundary detection problem can be viewed as the dual of the more commonly studied problem of segmentation, where the goal is to find all the points (or pixels) belonging to a single object. Boundary detection (or segmentation) algorithms can be broadly classified based on whether they use range data or imagery.

The most common method for segmenting range data is to use jump distance thresholds [32] [33] [34] [35]. Boundaries are identified wherever the Euclidean distance between two adjacent points is larger than a threshold. Arras [34] and Spinello [35] observed that although fixed threshold methods work well in indoor environments, they do not work well in outdoor environments due to the longer ranges involved and more challenging conditions. For example, if the threshold is chosen to be large enough to cluster objects at long distances, then nearby objects are not separated. To solve these problems, new segmentation methods using more complex and adaptive thresholds (typically a function of range) were developed [32] [33]. While these methods address the problem of long-range objects, they still have difficulty with challenging scenarios such as occlusions and viewpoint changes in outdoor environments.

There is much existing literature related to object tracking and data association. In this chapter, we will briefly introduce several of the most common algorithms used to address the data association problem because our object tracking is focused not on solving a general tracking problem but on keeping track of an object of interest over time.

The most commonly used algorithm is the *Nearest Neighbor* (NN) approach, which assigns each observation to the most probable object [36] [37] [38]. The biggest limitation of this approach is that it makes irreversible assignment decisions immediately. Information obtained in the future can be invaluable at resolving ambiguities at the current time step. The *Joint Probabilistic Data Association* (JPDA) algorithm is a popular approach to tracking multiple moving objects [39] [37] [38]. The JPDA computes a Bayesian estimate of the correspondence between features detected in the sensor data and the different objects to be tracked. The JPDA can be seen as an approximation of the exact Bayesian solution of the data association problem. The *Multiple Hypothesis Tracking* (MHT) algorithm keeps several correspondence hypotheses for each object at each time frame [40] [37] [38].

2.3 Vertical Boundary Detection

Vertical boundary detection is a key step in segmenting an object from its background, which is known as figure-ground segmentation. The ability to distinguish a foreground object from the background in a scene can simplify object detection and recognition, and distinguishing a foreground object is a first step in detecting, modeling, and tracking moving objects in a scene.

We cast the problem of vertical boundary detection as a classification task. A single line-scanning lidar produces a sequential series of range measurements with fixed angular increments in polar coordinates. The vertical boundary of an object may lie at any point in this continuum, and therefore, the true location of the boundary is likely to lie at some position between the measured rays. The classification task is to determine whether an object boundary exists between two consecutive range measurements.

We consider the space between consecutive lidar measurements, and predict whether an object boundary exists within this region or not. Given a set of labeled local features derived from the

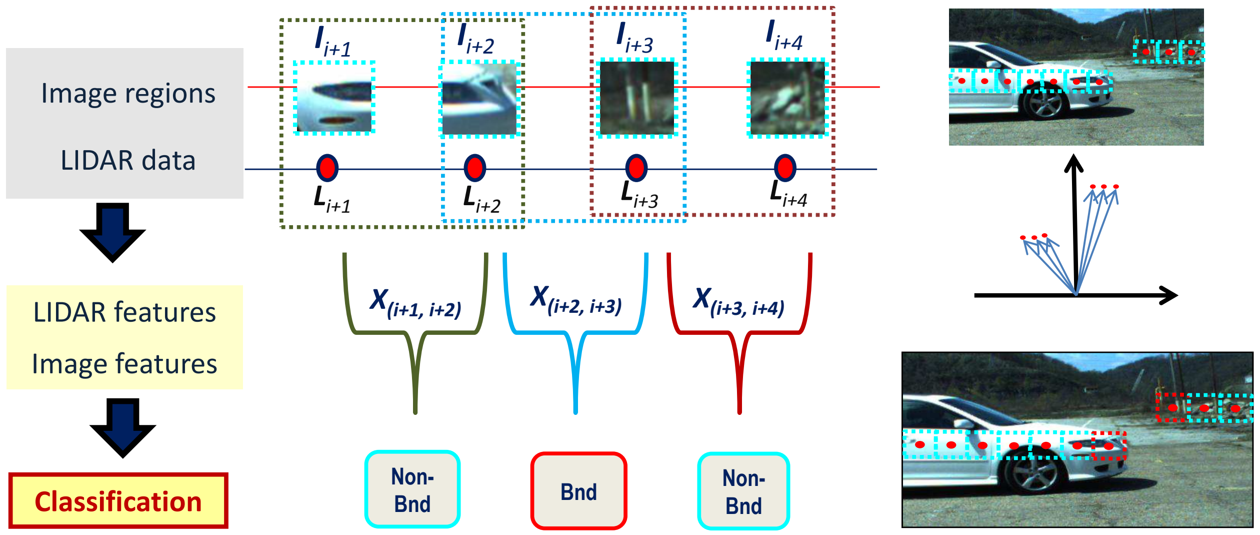


Figure 2.1: Overview of vertical boundary detection. The boundary detection problem in the framework of classification (Nb: Non-boundary, B: boundary).

data, standard machine learning tools, such as support vector machines (SVMs), can be used to learn the relationship between features computed from the data and the boundary/non-boundary classification labels.

If the scene is observed simultaneously with a camera, then it is possible to calibrate the camera pose with respect to the lidar, which allows the range measurement points to be projected onto the camera image. The centers of projection of the camera and lidar need to be as close together as possible to minimize the effects of parallax. Camera image resolution is typically significantly higher than the angular resolution of lidars, which means that the region between two projected laser points will contain multiple pixels. Consequently, information from image patches must be aggregated. We use rectangular image patches surrounding the projected lidar points for computing image-based features. Intuitively, if the image patches surrounding two points belong to the same object, then the statistics of the image patches are more likely to be similar than if they belong to two different objects.

Evidence of a boundary existing between two lidar points may be based on features derived from the range, including distance between real returned points (except intervening points with no return from the laser scanner, which we refer to as "no-return points"), range from origin to mid-point, and surface orientation. Evidence of a boundary between two points in an image is based on image texture features, histogram similarity, mean intensity, standard deviation of intensity.

Given a set of features derived from the lidar and image data, along with the ground truth labeling of whether a boundary exists between each pair of lidar measurements, we use standard machine learning techniques to train a classifier to correctly label novel data queries.

2.3.1 Lidar Features

When a lidar returns consecutive range scan points ($p_i,\ i = 1, \ldots, n$) in a full scan, each scanned point can be defined as (r_i, ϕ_i) in polar coordinates and as ($x_i = r_i cos\phi_i,\ y_i = r_i sin\phi_i$) in Cartesian coordinates. Here, n denotes the number of scan points, including valid and invalid scanned points in a full scan. A point is considered invalid if the sensor did not receive a return for that measurement, which occurs, for example, when there is no surface within the sensor's range. In this dissertation, we only consider the relation between valid range measurements ($p_j,\ j = 1, \ldots, n_r$), where n_r is defined as the number of valid returned points. The lidar features are defined as a function ($f_j : P_j \to \mathbb{R}^m$) that takes a pair of consecutive valid points ($P_j = (p_j, p_{j+1}),\ j = 1, \ldots, (n_r - 1)$) and returns an m-dimensional value ($X_j^L = (d_j, l_j, \theta_j)^T,\ X_j^L \in \mathbb{R}^3$). We define the following lidar-based features (refer to Figure 2.2):

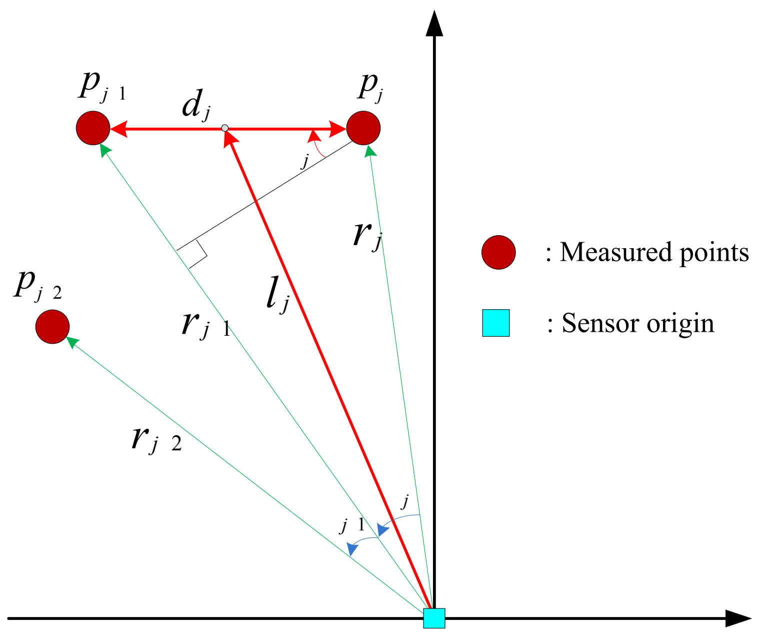


Figure 2.2: Lidar features.

- *Distance between valid returned points* $(d_{j,j+1})$: This feature is the Euclidean distance between points p_j and p_{j+1} $(j = 1, \dots, (n_r - 1))$ and is given by

$$d_j = \|p_j - p_{j+1}\|_2. \tag{2.1}$$

- *Range from origin to mid-point* $(l_{j,j+1})$: This feature measures the range from the lidar sensor's origin to the mid-point between points p_j and p_{j+1}, which is given by

$$l_j = \left\| \frac{p_j + p_{j+1}}{2} \right\|_2. \tag{2.2}$$

- *Angle between valid returned points* $(\theta_{j,j+1})$: This feature is designed to measure the orientation of the surface with respect to the viewing direction and is given by

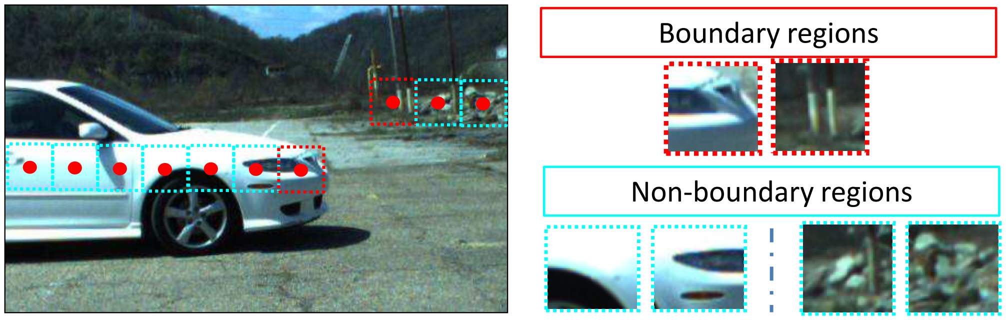


Figure 2.3: Image features. Left: Examples of image patches used for computing image-based features. Right: Close-ups of some representative pairs of adjacent regions from the left image.

$$\theta_j = \arctan\left(\frac{r_{j+1}}{r_j \sin\phi_j} - \cot\phi_j\right). \tag{2.3}$$

2.3.2 Image Features

Each image feature is based on two rectangular regions, R_j and R_{j+1}, extracted from image locations centered on the projection of the lidar measurements, p_k and p_{k+1} into the image (Figure 2.3). The image features are defined as a function ($f_j : P_j \rightarrow \mathbb{R}^m$) that takes a pair of consecutive valid patches ($I_j = (R_j, R_{j+1})$, $j = 1, \ldots, (n_r - 1)$) and returns a 3-dimensional value ($X_j^I = (h_j, m_j, s_j)^T, X_j^I \in \mathbb{R}^3$). In our experiments, because the original patch size is varying by the lidar measurement, we normalized the size (N) of each patch to be 11 by 15 pixels.

- *Histogram intersection ($h_{j,j+1}$)*: This feature measures the similarity between two histograms computed over adjacent regions. This is calculated as

$$h_j = \frac{\sum_B \min(H_j, H_{j+1})}{\sum_B H_j} \tag{2.4}$$

where H_j and H_{j+1} are the histograms acquired from the image patches R_j, R_{j+1}, and B is the number of bins in the histogram. In our experiments, B is set to 16.

- *Difference of mean intensity of two patches ($m_{j,j+1}$)*: This feature measures the difference of average intensity of adjacent patches

$$m_j = \frac{1}{N}\left(\sum_{m,n}\left(R_j\left(m,n\right) - R_{j+1}\left(m,n\right)\right)\right) \tag{2.5}$$

- *Difference of standard deviation of two patches (s_j)*: This feature returns the difference of standard deviation of the intensity of adjacent patches.

$$s_j = \sigma_j - \sigma_{j+1} \tag{2.6}$$

where

$$\sigma_j = \sqrt{\frac{1}{N-1}\sum_{m,n}\left(R_j(m,n) - \mu_j\right)^2} \tag{2.7}$$

and μ_j is the mean intensity value of patch R_j. The value of σ_{j+1} is defined analogously.

2.3.3 Boundary Classification

Features derived from the lidar and imagery are used to train a support vector machine (SVM) classifier to label pairs of range measurements as boundary or non-boundary. SVM is a common supervised learning algorithm for binary classification. An SVM seeks the hyperplane that maximizes a notion of margin between classes [41]. Linear SVM is fast, has publicly available implementations, and handles high-dimensional feature spaces well.

2.4 Association of Object Regions

Once we detect the object regions in a scene through vertical boundary detection, we assign the objects to the object labels, which is commonly referred to as the data association problem. Data association is a straightforward problem if we track a single object in a scene. Assuming no measurement noise and only a single object, there will be just one measurement at every time instant and therefore data association is trivial. However, in most real situations, multiple objects are detected that sometimes move near each other or occlude one another, making data association difficult.

Combining geometric information and appearance of objects can help to find correspondences across frames. When two objects appear superimposed in our perspective, depth information may be used to rule out matches where appearance cannot. However, when two objects move close to one another, appearance may contribute to the decision of matching candidates because the appearance of an object does not change dramatically in a short time period.

We detect objects in every frame and seek to match a set of tracks across frames. Once objects have been detected, data association can be performed to link the object tracks to the currently detected object. Since the lidar provides accurate depth information of the objects, some matches are easily filtered out when the matches are geometrically unlikely from the position in the previous frame. This process is is referred to as gating. The gating process decomposes data association into sub-problems.

Once highly unlikely matches are filtered out, we evaluate each observation in the track gating region and choose the best one to incorporate into the track. The data association between the track and observations is accomplished in three main steps. We first evaluate the correspondence between the track and observations by measuring the distance of lidar measurements and appearance similarity. Second, we compute the relative likelihood between matching pairs with both similarity measures. Finally we choose the best matching pairs by solving an integer programming problem [42] [43].

2.4.1 Similarity of Lidar Measurements

A direct measurement of similarity between data points is the distance between two points. Considering N tracks and M observations, there are $N \times M$ possible matching combinations. In practice, some matches are more likely to originate from one track than another. Measuring the geometric distance is therefore an efficient way to estimate the likelihood of matches; the longer the distance between a object and an observation, the less likely the observation corresponds with the object.

Considering a pair of an object Z_n at time t and an observation Z_m at time $t+1$, the similarity of two lidar measurements is estimated by a distance between the center points $\bar{\mathbf{z}}_n$ and $\bar{\mathbf{z}}_m$ of both measurements. The lidar likelihood S_L for matching the Z_m to the Z_n is estimated by an exponential function:

$$S_L(Z_n, Z_m) = \exp\left(-\frac{\|\bar{\mathbf{z}}_n - \bar{\mathbf{z}}_m\|^2}{\sigma^2}\right), \tag{2.8}$$

where the S_L is normalized from 0 to 1.

2.4.2 Similarity of Image Regions

Image similarity is measured with sub-image regions I_n and I_m in the object region of interests (ROIs) which are cropped from the vertical boundary detection. Since a single line scanning lidar measures depth of a stripe line of an object, we define the sub-image region to be a rectangular region. The width of the rectangular region is the same as the object ROIs, while height is measured using a pre-defined size onto the object ROIs centered on the projection of the lidar measurements.

Considering the normalized color histograms h_n and h_m extracted from sub-images I_n and I_m respectively, the image similarity S_I is estimated by computing the Bhattacharyya measure between distributions of h_n and h_m [44]. However, the Bhattacharyya measure is a divergence-type measure between distribution and does not impose a metric structure. We use the modification of the Bhattacharyya measure proposed by [45]. This measure represents a metric distance between two distributions as:

$$S_I(I_n, I_m) = \sqrt{1 - \rho(\mathbf{h}_n, \mathbf{h}_m)}, \tag{2.9}$$

where

$$\rho(\mathbf{h}_n, \mathbf{h}_m) = \sum_{k=1}^{K} \sqrt{(h_n(k), h_m(k))},$$

and K is the number of bins.

2.4.3 Data Association

Given N tracks at time t and M observations at time $t+1$, we compute a maximum-likelihood matching between the objects and observations using the approach proposed by Li [43]. The algorithm relies on two inputs: a likelihood vector $\mathbf{c}$ of V matching hypotheses and a constraint matrix $\mathbf{A}$, where $\mathbf{c}$ is a length V vector and $\mathbf{A}$ is a $V \times (N+M)$ matrix. We create the vector $\mathbf{c}$ and the matrix $\mathbf{A}$ as follows.

When the similarity measures S_L and S_I are given, the vth likelihood c_v is estimated as:

$$\mathbf{c}_v = S_L(Z_n, Z_m)\ S_I(I_n, I_m). \tag{2.10}$$

The constraint matrix $\mathbf{A}$ whose row index corresponds to the vth hypothesis is built as:

$$\mathbf{A}(v,i) = \begin{cases} 1 & \text{if } i = n \text{ or } i = N + m \\ 0 & \text{otherwise.} \end{cases} \tag{2.11}$$

Given the likelihood vector $\mathbf{c}$ and the constraint matrix $\mathbf{A}$, selecting the best matching pairs between the tracks and the observations is equivalent to finding a subset of rows of $\mathbf{A}$ such that the sum of corresponding elements in $\mathbf{c}$ is maximized, under the constraint that no two rows share common non-zero entries. This can be posed as the following integer programming problem:

$$\max_{\mathbf{x}} \mathbf{c}^T\mathbf{x} \quad s.t \quad \mathbf{A}^T\mathbf{x} \leq \mathbf{b}, \tag{2.12}$$

where $\mathbf{x}$ is a binary $V \times 1$ vector which is 1 if vth row is in the solution, or 0 otherwise, and $\mathbf{b}$ is a $V \times 1$ identity vector. While integer programming problems are in general NP-hard, the problem given in Equation 2.12 can be solved exactly using linear programming. This is due to the fact that the constraint matrix $\mathbf{A}$ is totally unimodular[1], and the right-hand sides of the constraints are all integers. In fact, if the above two conditions are satisfied, a linear programming problem will always have an integer-valued solution [46]. We solve the integer programming problem with the MATLAB function, *linprog*.

[1]A matrix is totally unimodular if the determinant of any square submatrix takes one of the values in -1, 0, 1.

2.5 Experiments

We evaluate the performance of the vertical boundary detection approach in this section. The performance evaluation of the data association approach is given in Chapter 3 because it is tightly coupled with the object representation approach that will introduce in Chapter 3.

Using the framework described in [47], we performed a series of experiments to evaluate the effectiveness of our boundary detection approach. The experiments include: a) a comparison of different types of classifiers; b) a comparison of our algorithm with a state-of-the-art method that uses dynamic jump distance thresholding; and c) a comparison of our algorithm using lidar features, image features, and a combination of lidar and image features.

For our experiments, a SICK LMS-221 lidar and a PointGrey Flea2 camera were used to acquire the data sets. The lidar has a 180° degree horizontal field of view (FOV), with a line scanning frequency of 75 Hz and a 0.5° angular resolution. The camera has a 60° degree horizontal field of view, with a frame rate of 30 Hz and a resolution of 800 by 600 pixels. These sensors were mounted on front of a vehicle. The lidar was placed at a height of 70 cm, and the camera was installed 30 cm above the lidar. The data from each sensor was time-stamped to enable synchronization.

2.5.1 Choice of Classifier

We compared the performance of three different classifiers: SVMs using a linear kernel, SVMs using an RBF kernel, and logistic regression. For the SVM classifier implementation, we used LibSVM [48] and determined the penalty parameter ($C = 10$) for the linear kernel SVM and the penalty and kernel parameters ($C = 10$, $\gamma = 0.1$) for the RBF kernel SVM by cross validation. The results of the experiment show that there is not much differentiation between the performance of the chosen

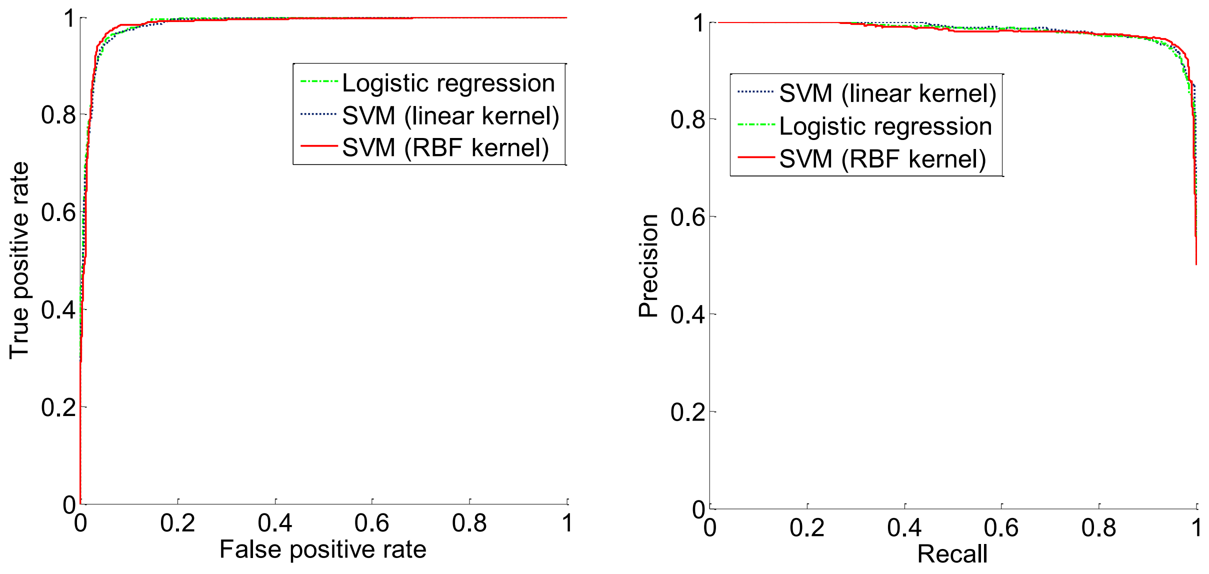


Figure 2.4: Comparison of classifiers. The performance comparison of three different classifiers. ROC (left) and Precision/Recall (right) curves.

classifiers (Figure 2.4). In the ensuing experiments, we use the SVM classifier with a linear kernel, which is sufficiently fast and does not suffer from the possibility of overfitting than when we use a radial basis function as kernel.

2.5.2 Comparison with Dynamic Thresholding

In our second experiment, we compared the performance of our algorithm, using lidar features only, with the adaptive breakpoint detector (ABD) algorithm described by Borges and Aldon [32]. The ABD algorithm uses a threshold that scales as a function of distance from the sensor to determine the boundaries between objects.

We evaluated the two algorithms on the same data sets, which included all instances inside and also outside the camera FOV. The results, shown in Figure 2.5, show that our algorithm outperforms the ABD algorithm for all threshold values. We also manually extracted a subset of

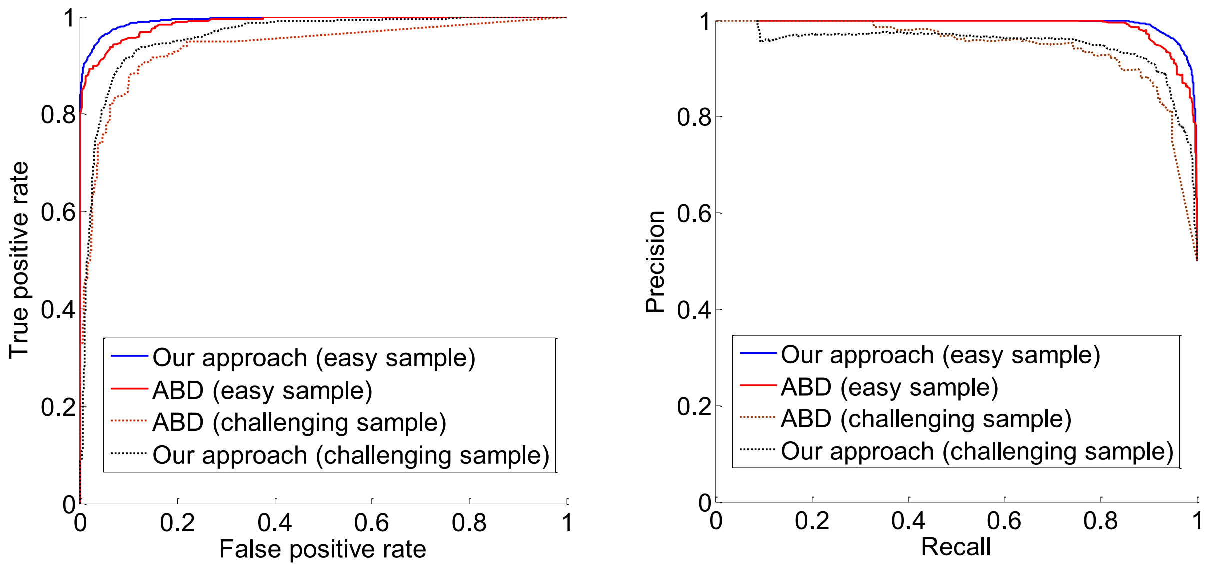
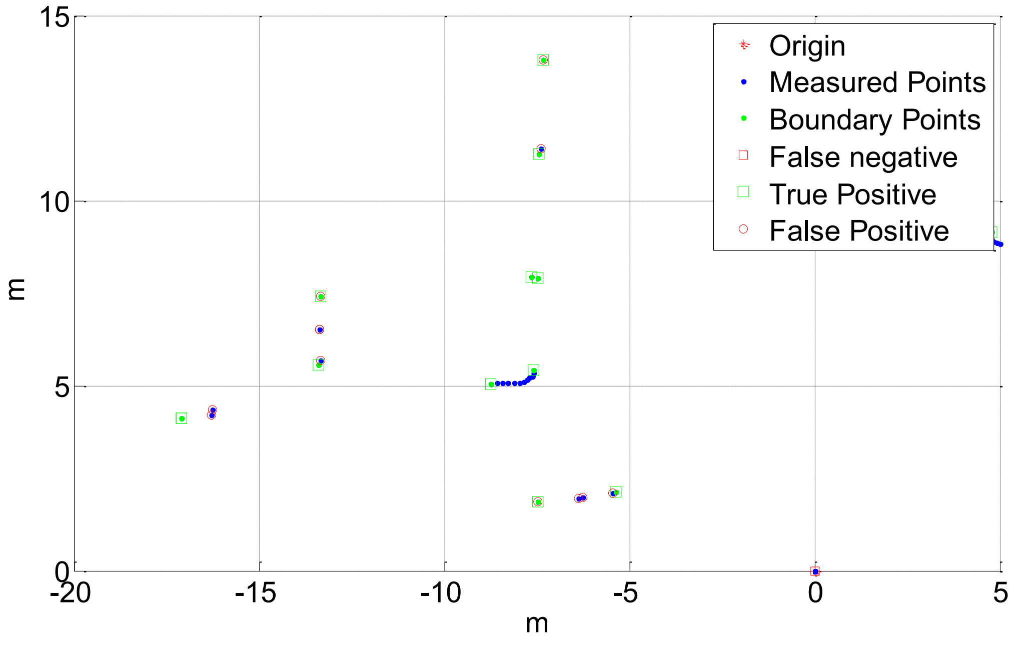


Figure 2.5: Comparison with Dynamic Thresholding. Threshold averaging ROC (left) and Precision/Recall (right) curves. These illustrate that our method improves boundary detection performance when compared to the ABD algorithm.

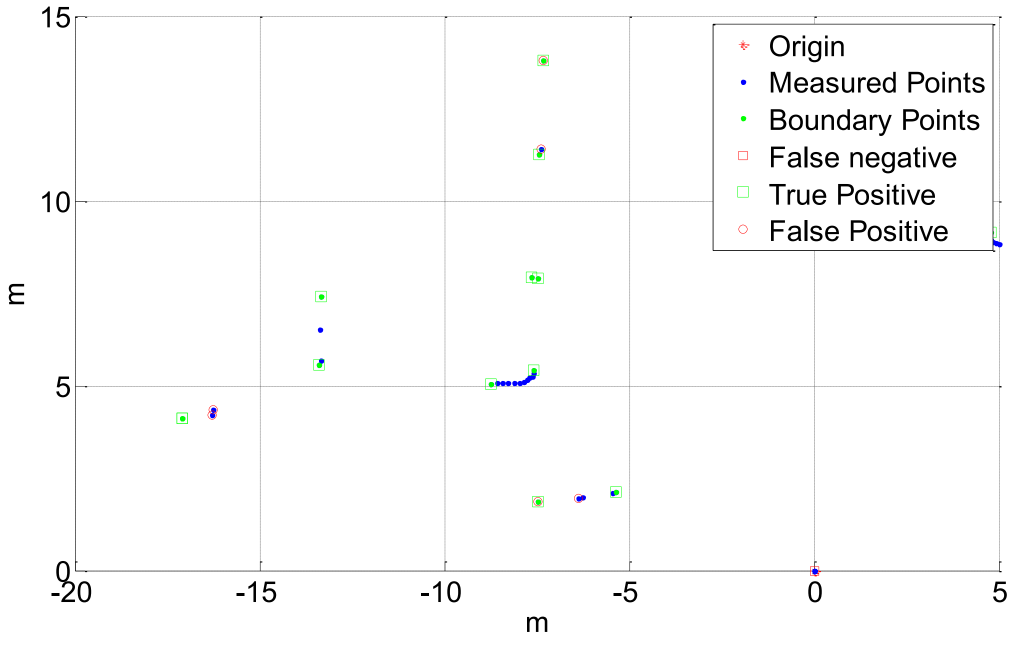
the full data that consisted entirely of challenging cases such as bushes, occlusion, and transparent surfaces. The performance of both algorithms on this challenging data set is lower than on the full data set, but our algorithm still outperforms the ABD method. Figure 2.6 shows an example of the boundary classification results of the two algorithms for a portion of an example scan. In this example, the ABD algorithm breaks up single objects when they are viewed obliquely (for example, a car seen from a slanted angle), whereas our method classifies these instances correctly.

2.5.3 Effect of Sensor Modality

The next experiment was designed to test the contribution of the lidar-based features versus the imagery-based features. In this experiment, only instances within the camera's FOV were used. We evaluated three cases: lidar features only, image features only, and lidar and image features



(a) Segmentation result (ABD)



(b) Segmentation result (our approach)

Figure 2.6: An example showing the segmentation performance of the ABD method and our algorithm for a single scan.

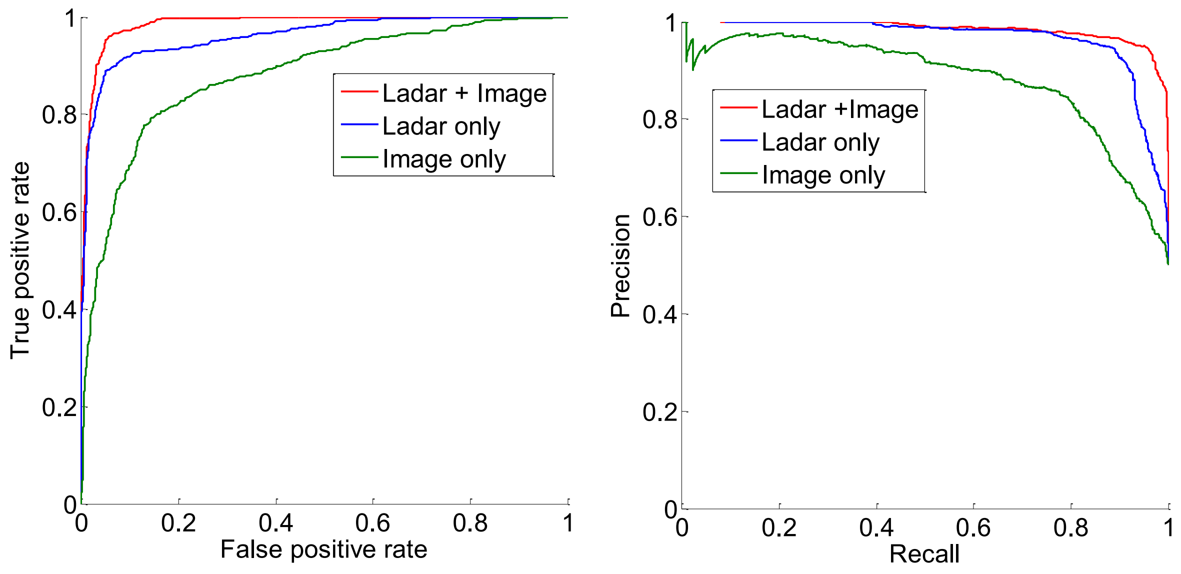


Figure 2.7: Effect of Sensor Modality. Threshold averaging ROC (left) and Precision/Recall (right) curves when image only, lidar only, and combined features are used.

combined.

Figure 2.7 provides the threshold averaging ROC curves with respect to the three cases. As can be seen, the lidar features are more discriminative than the image-based features. This agrees with the intuition that depth discontinuities from 2D laser range measurements are a more reliable indicator of object boundaries than differences in image intensity or texture. However, the combination of lidar and image-based features performs better than features from either modality individually. This result suggests that the information in the lidar and image features is not entirely redundant.

Figure 2.8 shows the segmentation result of lidar, image, and combined data when we consider a sample frame of data in the camera FOV. As shown in Figure 2.8 (a), the car on the left and the person standing 0.5 m in front are grouped into a single object. Discriminating the objects into two different things is difficult because of their close proximity to each other.

Figure 2.8 (b) shows the segmentation result when only image features are used for segmentation.

As can be seen, the segmentation result is not as good in this case. In the figure, the marked points indicate the manually labelled ground truth and each colored box encompasses a set of points classified as non-boundary. In other words, the boxes represent the segmentation result. The height is fixed at 60 pixels.

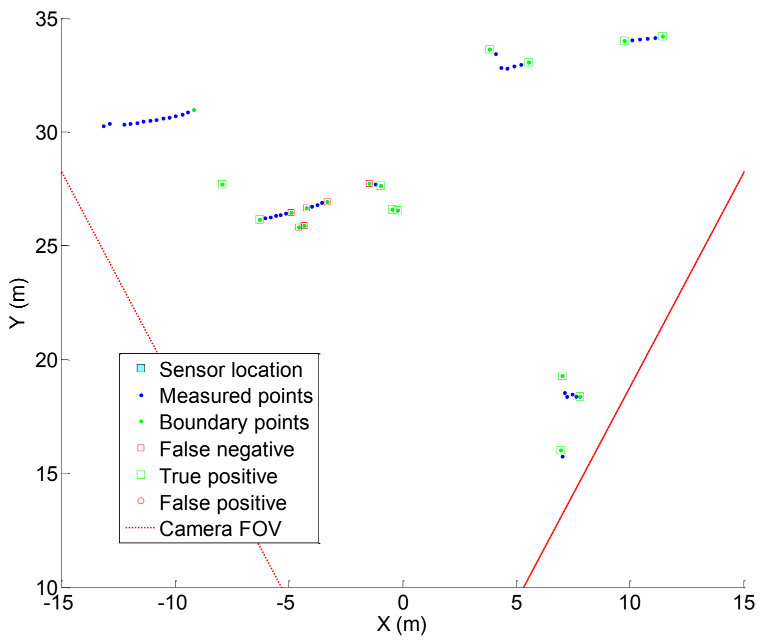
The method using combined lidar and imagery features was evaluated in the same frame of data (Figure 2.8 (c)). In this case, boundaries that were misclassified by the image feature method are now correctly classified. Specifically, the person and the car, which were separated by 0.5 m are now correctly divided into two segments. When only image or lidar features were used, these objects were segmented as one group.

2.6 Summary

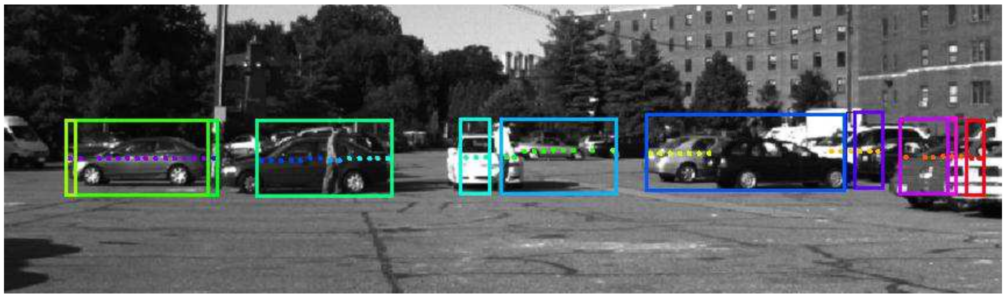
We introduced vertical boundary detection and data association as preliminary tools for object representation. Both approaches are performed using lidar measurement and corresponding imagery.

First, we have proposed a supervised learning approach for object boundary detection. The method is aimed at line scanning lidars and can be augmented with imagery if available. We formulate the boundary detection task as a classification problem and use features derived from the lidar measurements and image patches surrounding the lidar measurements as inputs to a classifier. The classifier is learned using a set of labeled training examples. Experiments show that the SVM classifier performs slightly better than logistic regression. The proposed method improves on the performance of an existing dynamic thresholding boundary detection algorithm, especially for challenging boundary cases. Further improvement is achieved by incorporating visual features derived from co-registered imagery.

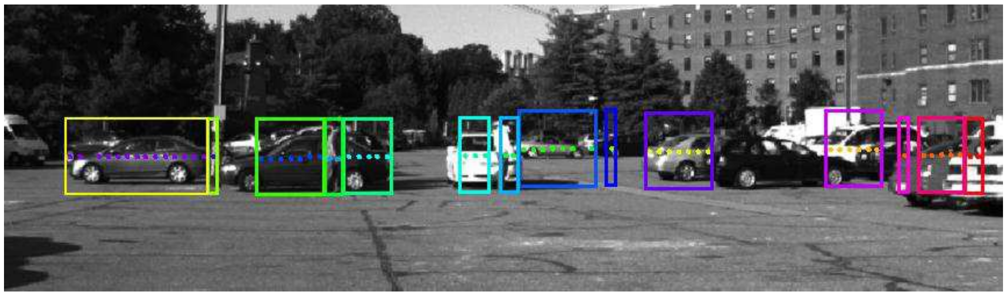
We have introduced a data association approach measuring the maximum-likelihood between objects and observations. The likelihood of each association pair is estimated by measuring appearance similarity using histogram intersection and by computing the euclidean distance between candidate matching pairs. The performance of data association is evaluated together with the object representation approach in Chap 3.



(a) The segmentation result of a single scan lidar data.



(b) The segmentation result of a single frame image data.



(c) The segmentation result based on image and lidar features

Figure 2.8: The boxes represent each cluster. The distance between the person and the car is only 0.5 m and the sensors' origin is placed at 25 m from the objects.

Chapter 3

Approximate Representation of Unknown Objects

3.1 Introduction

In this chapter, we propose an approximate object representation with a single line scanning lidar and camera data. The representation consists of an approximate unwrapped mosaic along with an approximate geometric model. The main concept behind the proposed representation technique is that, by unwrapping an object image using the corresponding lidar measurements, we can estimate the 3D pose of the object using 2D image matching. The goal is to create a viewpoint-invariant and scale-normalized object model. The algorithm learns the object model online without human intervention. The set of models created is used to detect and track previously seen objects and improve tracking performance in challenging situations such as occlusion, abrupt view changes and illumination changes.

Object detection and tracking are challenging problems in computer vision applications because the system has to manage many critical situations such as occlusion, viewpoint change, and intra-class variation. There are a variety of approaches, depending on the type of object and the sensor configuration.

Many approaches are based on 2D models; that is, the approaches track a set of point features, optical flow, or blobs in 2D image space [30]. Many approaches produce good results when objects are well-separated and move smoothly. However, performance falls dramatically when these conditions are not met; e.g., when the objects' visual appearances change significantly due to turns in their trajectory, when occlusions occur due to proximity between objects during periods of congestion, and when there is abrupt illumination change in outdoor environments.

In contrast with the 2D approaches, 3D model-based approaches have many advantages [2] [16]. First, prior knowledge of the 3D shape of an object can result in better performance under occlusion. Also, once we know the 3D geometry of an object, we can estimate the 3D pose of the object. Furthermore, 3D models can be applied even if objects change their orientation during the motion. The critical problem of the 3D model-based approach is that features extracted from a 2D image do not immediately correspond to the 3D object models. In other words, we must obtain the transformation between image and object coordinates.

In order to consistently track an object in 3D, we need a robust and effective object representation approach because tracking and modeling are interdependent. If we have a good estimate of the motion of an object, then modeling becomes simpler, and if we have a model of the object, then tracking becomes simpler. We create a model with the benefits of 2D and 3D models; the model can detect an object by 2D-to-2D image matching like the 2D model-based approach, and directly estimate the 3D pose of the object from the matching result. This model can be built by unwrapping an object image with the corresponding lidar measurement.

Unwrapping refers to the process of creating a 2D image representation of a 3D model. In this thesis, we define the unwrapped image as a fronto-parallel view of an object generated with its geometry [49]. Once we transform sequential object images to unwrapped images, we can easily align those unwrapped images by 2D-to-2D image matching because the unwrapped images are viewpoint-invariant and scale-normalized [50]. Matching two unwrapped images, we can directly estimate the 3D pose of the object.

The methodology for creating the approximate model of unknown objects is summarized as follows. First, we create an unwrapped image of a tracked object ROI with its lidar measurements in each frame. Second, we align the unwrapped object images by simultaneously minimizing the image matching error and geometric alignment error. Finally, we create an approximate model that consists of an unwrapped mosaic and approximate geometry. A high-level diagram of the approximate representation of an object is shown in Figure 3.1

3.2 Related Work

An object's shape and appearance can be described in a diversity of ways. In general computer vision applications, an object can be defined as anything of interest that needs further analysis. After knowing the object of interest, we need to find a suitable representation of the object. Types of objects may need very different representations due to the trade-off between accuracy and efficiency. An approximate object representation is determined by the end application. For example, a point-based representation may be proper for a flock of birds in the sky [9] [10], a bounding box may

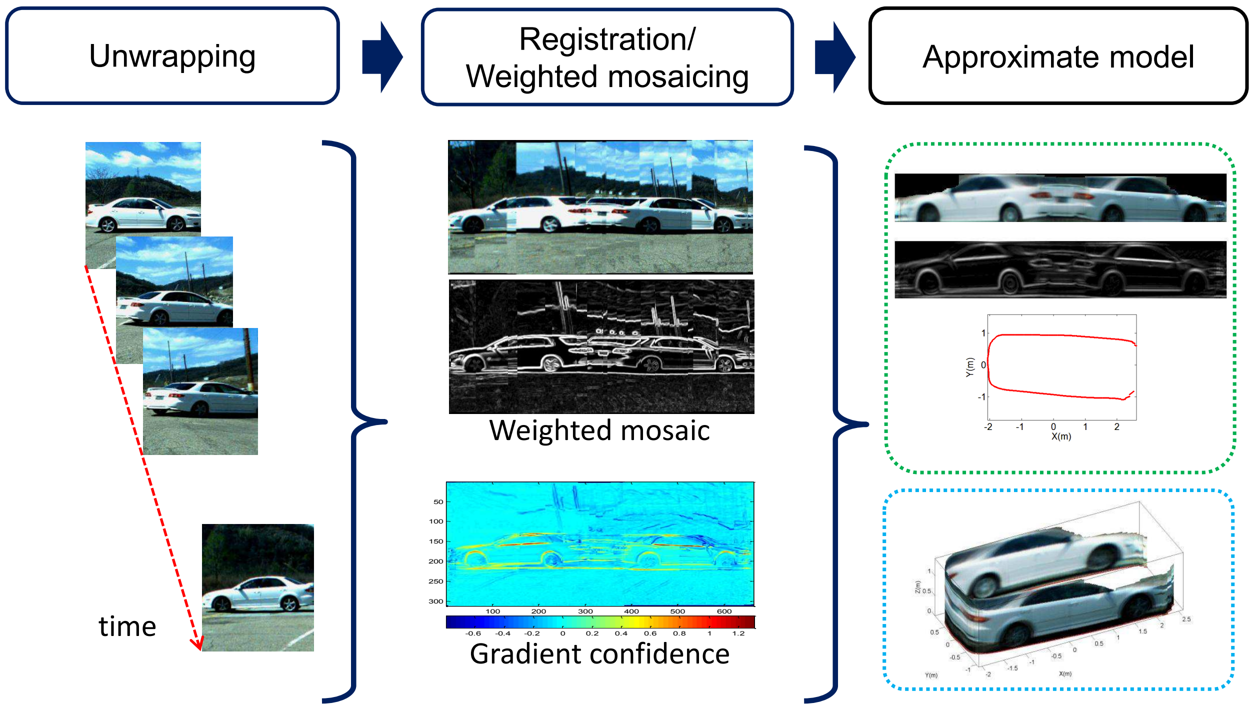


Figure 3.1: Overview of this chapter: Stages of the algorithm for the proposed approximate representation of unknown objects.

may be appropriate tracking people walking on the road [45], and a 3D model may be appropriate for tracking a rigid object in 3D space [2] [18] [20] [16].

There are many approaches to building 3D models of objects in controlled environments [51] [52] [24]. Critical to these approaches is the determination of correspondence between spatially localized features within each image. Most of the approaches model objects by aligning local features such as points and lines over time [53]. These approaches assume that two images taken from two barely different points of view are not very different and that there is a rich texture on the object surface. However, in an outdoor environment where both the object and sensor are moving, these assumptions are unrealistic, the object's illumination and viewpoint change dramatically.

Image-based matching may be superior to local feature-based algorithms in textureless cases [54].

On a sunny day, a car will violate the Lambertian assumption because parts of it are transparent and missing texture. This leads to many difficulties for detecting features. Since image matching uses an entire image, the approach can align images of a texture-less object. However, the objects may appear different from different views, and if the object view changes dramatically during tracking, the image-based matching may no longer be valid.

3.3 Image Unwrapping using Lidar Measurements

Once a piecewise linear fit of the 2D lidar geometry[1] is obtained, each fitted line directly corresponds to a plane in the image. The unwrapped image of an object is described as a set of warped image planes corresponding to the fitted lines. The unwrapping process is performed with two steps: (i) lidar geometry fitting and plane estimation, and (ii) object image warping with the fitted lidar geometry and weight estimation of each plane. The unwrapping process is illustrated in Figure 3.2.

3.3.1 Fitting Raw Lidar Measurements

A lidar returns sparse depth measurements of an object and sometimes provides unreliable depth information depending on the object's material characteristics. To represent the raw measurements as line components, we need to remove outliers of the raw lidar measurements by fitting. The line elements describing an object geometry are estimated by a piecewise approximation. The methodology for fitting a raw lidar measurement is summarized in two steps: (i) raw measurement clustering and polynomial curve fitting and (ii) piecewise linear fitting.

[1]We define our lidar scan data to a 2D geometry because the lidar provides depth information of a single line on an object's surface.

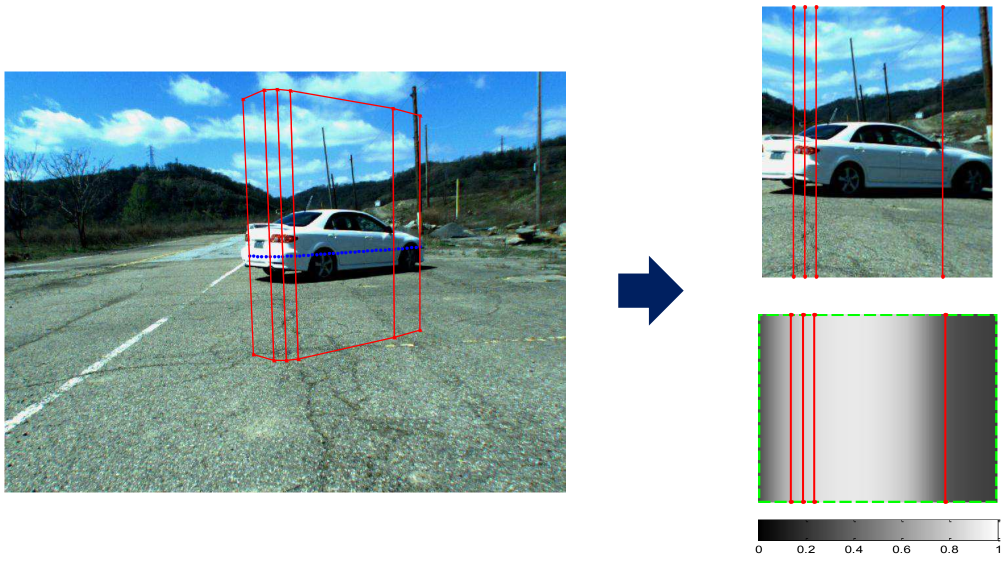


Figure 3.2: Image unwrapping procedure. Left image shows the planes which are estimated with the lidar geometry. Right images show the unwrapped object image and the weight of each unwrapped plane. The red lines describe the boundary of each plane. The green box describes the unwrapped image region which we use for the modeling.

Given a lidar geometry L of an object, we simplify the raw lidar measurements using the Douglas-Peucker (referred to as Iterative End Point Fit algorithm) [55] [56]. Using the algorithm, we detect corner points (c_i, $i = 1, \cdots, N$) over a threshold distance T_1 in the raw measurement. We separate the measurement $N+1$ clusters (L_j, $j = 1, \cdots, N+1$) based on the corner points c_i. In order to remove outliers, we fit a 2nd-order polynomial function $f(x_k)$ to the data points (x_k, y_k) of each cluster (L_j). The fit function $f(x_k)$ can be estimated minimizing the error E_f between the $f(x_k)$ and y_k in a least square sense:

$$E_f = \sum_{k=1}^{n_k} (y_k - f(x_k))^2 , \tag{3.1}$$

where $f(x_k) = a_0 + a_1 x + a_2 x^2$.

With the fitting functions of the cluster, we obtain a smooth estimate of the object's geometry. We refit piecewise linear functions to the smoothed geometry in order to estimate plane surfaces. In order to find feature points for piecewise linear fitting, we first detect corner points using the DouglasPeucker algorithm, in this case using a new threshold distance T_2 smaller than T_1. Once we find the corner points (c_i^*, $i = 1, \cdots, M-1$), the final lidar geometry consists of M lines, which corresponds by a description of the surface with M planes. In this thesis, we define the piecewise linear fitted lidar geometry as $L = \{L_j\}$, $j = 1, \cdots, M$ with respect to an object. Figure 3.3 shows a resulting of fitted raw lidar measurement.

3.3.2 Unwrapping of Object Image

We warp an object image to a frontal view image with lidar measurements; that is, we unwrap the set of imaged planes consist of the object image. We define α to be the inter-pixel distance in real-world units. For instance, if the inter-pixel distance is α and the height of model is h, the real height of the object is αh.

Each fitted line L_j of a lidar geometry directly corresponds to a plane of an object's surface because we assume that the surface is piecewise planar. Once we know the corresponding image region, that is, the object region image (I) with the lidar geometry, we can divide the object region image into M sub-image regions $I = \{I_j\}$, $j = 1, \cdots, M$ as shown in Figure 3.4. The L_j and I_j describe the geometry and sub-images of sub-surfaces of an object's surface (S), where S is a set of the geometries and sub-images, e.g., $S_j = \{L_j, I_j\}$.

In order to convert a perspective distorted image (referred to as the object image I) to a fronto-parallel image (referred to as the unwrapped image U), we estimate a homography matrix $\mathbf{H}$ which

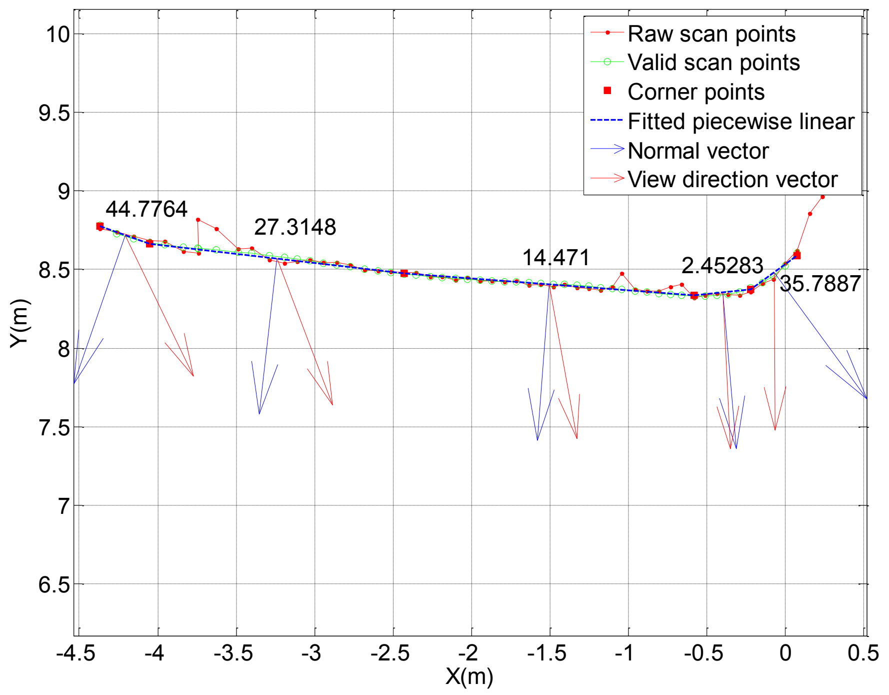


Figure 3.3: Fitted lidar measurement. The raw lidar measurement (red line) is fitted to the green line by curve fitting, and then the fitted line is re-fitted by a piecewise linear function (blue line). From the fitting result, we can estimate that the object surface consists of five planes. The numbers of each fitted line show the angle difference between the normal direction vector of each line and the viewing direction vector from the sensor.

is a transformation from the object image plane to the unwrapped image plane. Given a set of point pairs $\mathbf{p}_i = (u_i, v_i)$ in I and $\mathbf{q}_i = (x_i, y_i)$ in U respectively, we compute the transformation matrix $\mathbf{H}$.

We estimate vertices $(\mathbf{V}_i, i = 1, \cdots, 4)$ of sub-planes of the object's surface in 3D space. Projecting $\mathbf{V}_i$ onto the image, we can find the correspondences $\mathbf{p}_i$. The sub-image I_j is defined a region having $\mathbf{p}_i$ as vertices. The corresponding points on the unwrapped image are estimated with the inter-pixel distance α and the length of fitted lines l_j. The number of pixel rows in the

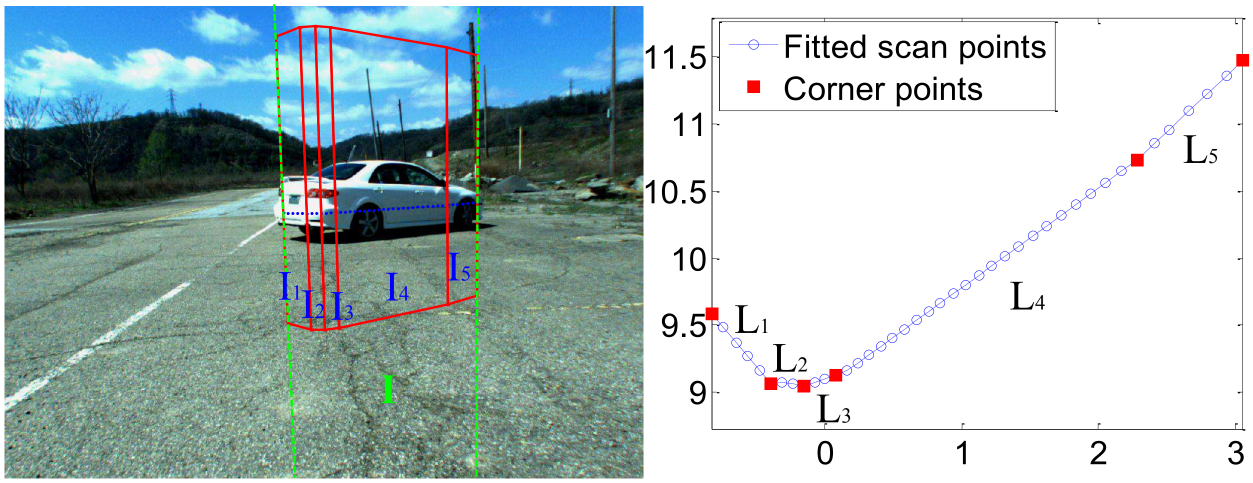


Figure 3.4: Estimated planes by lidar measurements. The imaged planes (I_1, I_2, I_3, I_4, I_5) in the object image **I** are estimated using the lines (L_1, L_2, L_3, L_4, L_5) obtained by piecewise linear fitting, respectively. Height of the imaged planes is fixed to a predefined height. In this thesis, we set the height to 6 m.

unwrapped image is computed from the fixed maximum height as $\frac{h}{\alpha}$. The number of columns is $\frac{l_j}{\alpha}$. The inter-pixel distance varies according to the working range R. If the working range is increased, the inter-pixel distance is decreased and vice-versa:

$$\alpha = R\,\frac{F\tan(\theta_L)}{W\theta_L}, \tag{3.2}$$

where θ_L is the angular resolution of the lidar, F is the horizontal Field of View (FOV) of a camera, and W is the horizontal resolution of an image.

Let $\mathbf{p}_i = (u_i, v_i, 1)^T$ denote the vertices in homogeneous coordinates of I_j, and let $\mathbf{H}_j$ denote the transformation from I_j to U_j. The corresponding vertices $\mathbf{q}_i = (x_i, y_i, 1)^T$ in U_j are estimated as [49]:

$$\mathbf{q}_i = \mathbf{H}_j \mathbf{p}_i. \tag{3.3}$$

Once we compute the transformation matrix (H_j), we map the I_j into U_j with $\mathbf{H}_j$. We use the inverse warping approach; that is, we do not warp each pixel (u, v) in I_j to its corresponding location (x, y) in U_j, but get (x, y) from its corresponding location (u, v). Since the mapping coordinate is not always an integer, we filter the pixel values by bilinear interpolation.

3.4 Motion Estimation

Consistent object tracking depends critically on motion estimation. This section addresses the motion estimation of a tracked object using its appearance and geometry. Combining appearance and geometry of an object may be helpful because certain aspects of the motion estimate can be difficult with each sensing modality individually. In the image domain, the translating motion of an object is easy to parse, since the appearance does not change much, but the rotating motion of the object may be hard to analyze. In the lidar domain, some data can be easy to match if it has sufficient constraints (such as the corner of a vehicle), but other data can be ambiguous (e.g., the flat side of a vehicle).

The motion of a tracked object is estimated by registering the currently unwrapped object image onto the previously unwrapped object image. The motion estimation is accomplished in two main stages: (i) unwrapped image registration and geometry alignment and (ii) four degree of freedom (DOF) motion estimation. The object in sequential frames is consistently tracked with the data association algorithm in Chapter 2.

Given two unwrapped object images U_{t-1} and U_t, and corresponding geometry data L_{t-1} and L_t

taken from different frames, we estimate the image matching error E_U and the geometry alignment error E_L.

Given the current image $U_t(\mathbf{x})$ sampled at discrete pixel locations, $\mathbf{x}_i = (x_i, y_i)$, we want to find where it is located in $U_{t-1}(x)$. The straightforward way to establish the alignment between two images is to shift one image relative to the other. We use normalized cross-correlation to measure quality of alignment of the two color images. The correlation value is computed by summing differences across all three color channels [57]:

$$S_{NCC}(\mathbf{u}) = \frac{\sum_i \{U_{t-1}(\mathbf{x}_i) - \overline{U}_{t-1}\}\{U_t(\mathbf{x}_i + \mathbf{u}) - \overline{U}_t\}}{\sqrt{\sum_i \{U_{t-1}(\mathbf{x}_i) - \overline{U}_{t-1}\}^2 \{U_t(\mathbf{x}_i + \mathbf{u}) - \overline{U}_t\}^2}}, \tag{3.4}$$

where $\mathbf{u} = (u, v)$ is the displacement, and

$$\begin{aligned} \overline{U}_{t-1} &= \frac{1}{N}\sum_i U_{t-1}(\mathbf{x}_i) \\ \overline{U}_t &= \frac{1}{N}\sum_i U_t(\mathbf{x}_i + \mathbf{u}). \end{aligned} \tag{3.5}$$

are the mean of the corresponding patches and N is the number of pixels in the patch. Figure 3.5 shows the NCC score between two unwrapped images. Different colors in Figure 3.5 indicate the level of cross correlation with blue being low and red being high values. Finally, the alignment position is obtained by locating the maximum correlation (red areas in the plot) as:

$$\mathbf{x}^* = \arg\max_{x,y} S_{NCC}(\mathbf{u}), \quad \forall (u, v) \in (x, y). \tag{3.6}$$

Given the alignment position $\mathbf{x}^* = (x^*, y^*)$ in U_{t-1}, we assign the matching score of the row y^* to the distribution of the matching score. The image matching error E_U is a N vector and is

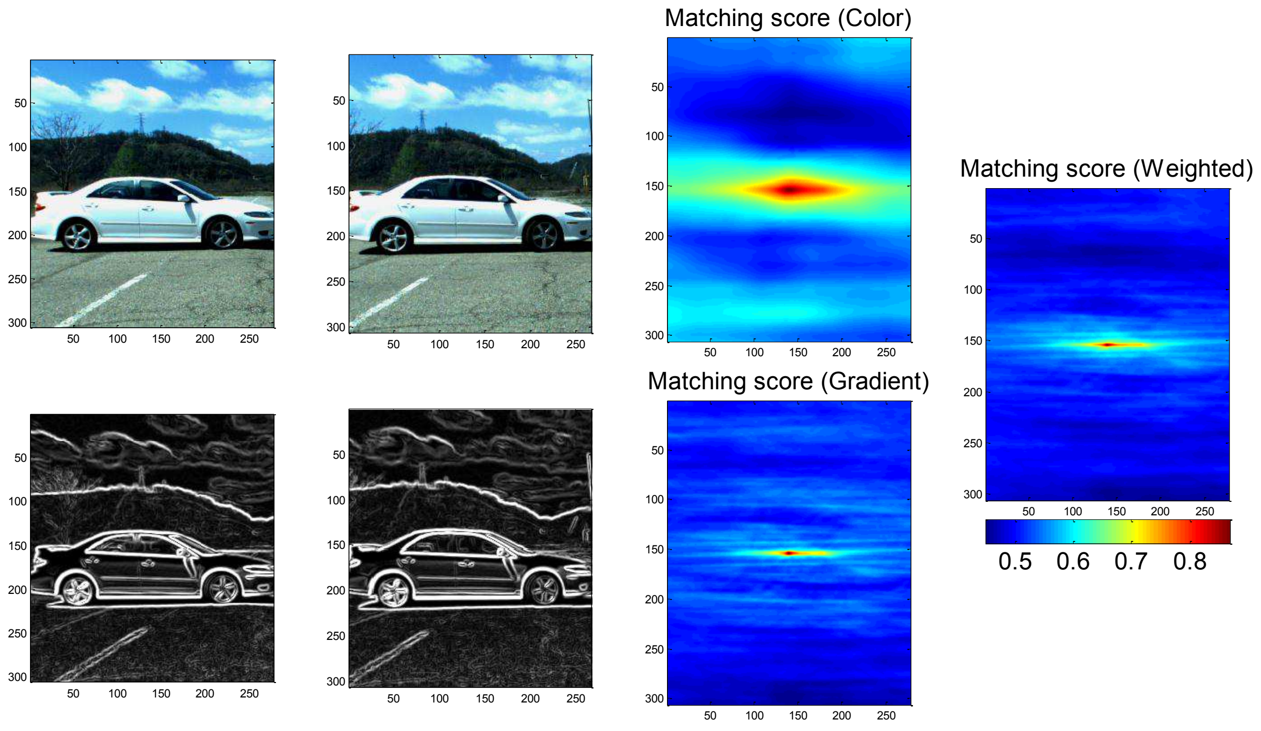


Figure 3.5: Similarity measure of two unwrapped object images. In the modeling stage, the similarity S_{NCC} is estimated from the weighted sum of the similarity of color images $S1_{NCC}$ and the similarity of gradient images $S2_{NCC}$; that is, $S_{NCC} = wS1_{NCC} + (1-w)S2_{NCC}$. The weight $w(=0.3)$ is estimated empirically.

obtained as:

$$E_U = 1 - S_{NCC}(x^*, y^*). \tag{3.7}$$

Denote by L_{t-1} the geometry data corresponding to U_{t-1} and by L_t the geometry data corresponding to U_t. We want to find a Euclidean transformation in 3D space consisting of rotation θ_Z and translation (t_x, t_y) that transforms the $\mathbf{x}$ of reference image onto the $\mathbf{x}^{'}$ of template image corresponding to the intersection region of the U_{t-1} and U_t. The alignment error is obtained as:

$$E_L(k) = \sum_{k=1}^{N} w_k \left\| \begin{bmatrix} \cos\theta_z & -\sin\theta_z \\ \sin\theta_z & \cos\theta_z \end{bmatrix} \begin{bmatrix} x_k \\ y_k \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} - \begin{bmatrix} x_k^{'} \\ y_k^{'} \end{bmatrix} \right\|_2, \quad k \in [1, N]\,, \tag{3.8}$$

where w_k is defined to the ratio of the column size of intersection region to the column size of U_{t-1}.

Measuring the image registration error and the corresponding geometry alignment error, we estimate a motion with four degrees of freedom $(\theta_z, t_x, t_y, t_z)$ which is described by translations and rotation in 3D space. The translation t_z is estimated from the image matching because this translation related to the vertical direction in image.

Figure 3.6 shows the motion estimation result by simultaneously minimizing image registration error and geometry alignment error. The 4-DOF motion is estimated by the weighted sum of both alignment errors:

$$\begin{aligned} [\theta_Z, t_x, t_y] &= \arg\min_k \{w_1\, E_U(k) \;+\; w_2\, E_L(k)\}, \\ t_z &= y^*, \end{aligned} \tag{3.9}$$

where the weight values, w_1 and w_2 are obtained experimentally ($w_1 = 0.7$, $w_2 = 4$).

3.5 Weighted-Mosaicing

In order to create a model incrementally, sequential images must be aligned to a reference image. We might envision two approaches for aligning images: (i) aligning two sequential images and estimate an offset relative to a reference image and (ii) aligning a current image onto a reference image which accumulates previous images. The first approach benefits matching two sequential

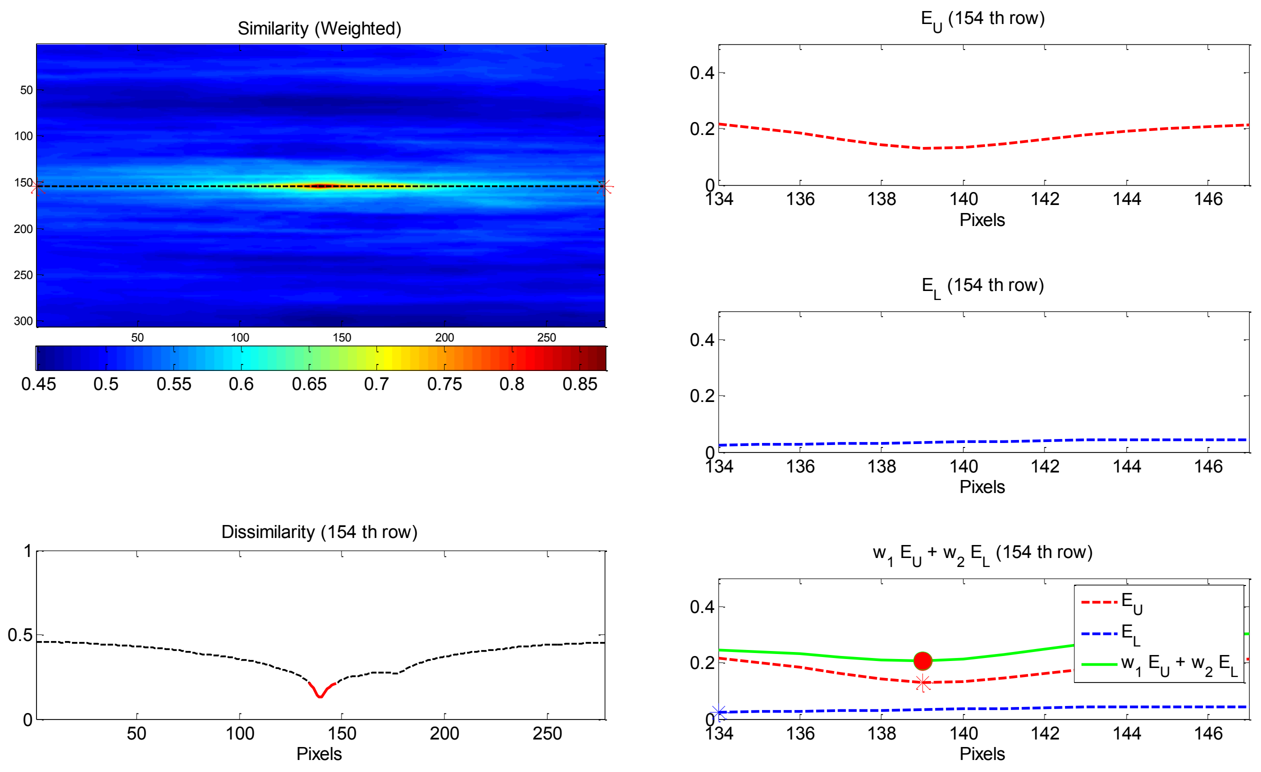


Figure 3.6: Motion estimation result. The image matching result shows that the alignment position is (154, 139) and the matching score is 0.75. The distribution of the matching error is the dissimilarity value at the 154th row. In the upper left image, the red stars show the number of the row. The geometric alignment error is estimated within the interval where the image matching score is higher than 0.9 times as much as the matching score at (154, 139). This interval is shown in the lower left image as the red line. In the lower right image, the red circle shows the estimated position minimizing the weighted sum of E_U and E_L.

images because the appearance of an object does not dramatically vary in a short period, but this approach has a drift problem which accumulates alignment error over time. The second approach avoids the drift problem because it aligns a current image onto a reference image created with all previous sequence images, but matching the current image onto the reference image is difficult because the reference image is blended with images from different views and different illuminations.

Image stitching is the process of merging multiple images with overlapped fields of view to generate a single image. This process can be divided into two main stages: image registration and blending. Since the unwrapped images are invariant with respect to scale and viewpoint changes, the unwrapped images are registered by 2D-to-2D matching. The critical issue is how to represent the registered-unwrapped images obtained under different illumination conditions to generate a single unwrapped image.

We update the reference image using a weighted mosaic approach. The weighted mosaic approach is a way to minimize the difference between the reference image and a current image when we update the reference image. We build the mosaic image with a set of column images of previous image frames which have a high weight. The weight is dependent on the angle between the sensor's view direction and the normal of an object's surface because quality of unwrapped images degrades as this angle increases. In order to minimize the difference between adjacent columns, we smooth the mosaic image in an MRF energy minimization framework [58], [59].

Our weighted-mosaicing approach is accomplished in two stages: (i) estimating weights of image planes and (ii) stitching image planes using their weights.

3.5.1 Plane Weight Estimation

Since the mosaic image is built with a set of column images accumulated over time, we estimate a weight for each column of the unwrapped image. Considering an unwrapped image ($\mathbf{U} = \{U_i\}$, $i = 1, \cdots, M$), the number of columns N_U of $\mathbf{U}$ is measured as follow:

$$N_U = \frac{1}{\alpha} \sum_{i=1}^{M} l_i, \tag{3.10}$$

where l_i is length of ith fitted line.

Given N_U, we estimate the weight ($w_k,\ k = 1, \cdots, N_U$) of each column based on the angle ϕ between the viewing direction of a sensor and the surface normal corresponding to the column:

$$w_k = \cos^2 \phi_k. \tag{3.11}$$

If the distance between an object and a sensor does not change dramatically, this measured weight is reasonable. Otherwise, w_k should depend on the object distance, since the quality of unwrapped images depends on distance. In that case, we redefine the weight function with a distance weight:

$$w_k^* = \cos^2 \phi_k \ \exp\left(-\frac{d_k}{2\,R}\right), \tag{3.12}$$

where d_k is geometric distance of the kth column and R is the working range we defined.

3.5.2 Weighted Stitching using MRF model

After the weights are computed, each column can be taken from the image with the highest weight. However, doing this may lead to discontinuities between adjacent columns. It is desirable to minimize discontinuities between consecutive columns of the unwrapped mosaic image. Our weighted mosaicing approach can be regarded as a labeling procedure in an MRF framework. We formulate the problem as a 1D MRF where each MRF label determines which image to use as the source for the mosaic image in each corresponding column. Figure 3.7 shows the mosaicing of unwrapped images in the modeling sequence.

We use graph cut optimization to estimate the label of each column in the mosaic image. The graph cut optimization solves the labeling problem minimizing an energy function with two energy terms [60]. Let X represent a set of labels for each column, where x_k denotes the label of the kth column. The energy function which we minimize is denoted by $\mathbf{E}(X)$:

$$\begin{aligned} \mathbf{E}(X) &= \mathbf{E}_D(X) + \lambda\, \mathbf{E}_S(X) \\ &= \sum_{k \in V} \mathbf{E}_D(x_k) + \lambda \sum_{(k,k+1) \in \mathbf{E}} \mathbf{E}_S(x_k, x_{k+1}) \end{aligned} \tag{3.13}$$

where the first energy term ($\mathbf{E}_D(x_k) = w^*_{(k,x_k)}$) is defined to be the data for assigning a column k to label x_k and the second energy term ($\mathbf{E}_S(x_k, x_{k+1}) = \|x_k - x_{k+1}\|$) is the pairwise penalty for smoothing the frame difference of neighbouring labels.

When different images are stitched together, for various reasons (changed lighting conditions, vignette effects) the adjacent pixel intensities differ enough to produce artifacts. To remove these artifacts, we use a feathering approach (a center weighting image blending) [57]. In this approach, the pixel values in the blended regions are a weighted average from the two overlapping images.

3.6 Approximate Representation

Tracking an object over time, we are able to obtain the mean appearance (e.g., color and edge-gradient) of the accumulated object images and the mean geometry of the tracked object. However, the mean appearance does not explain the foreground region in the image because we detected the

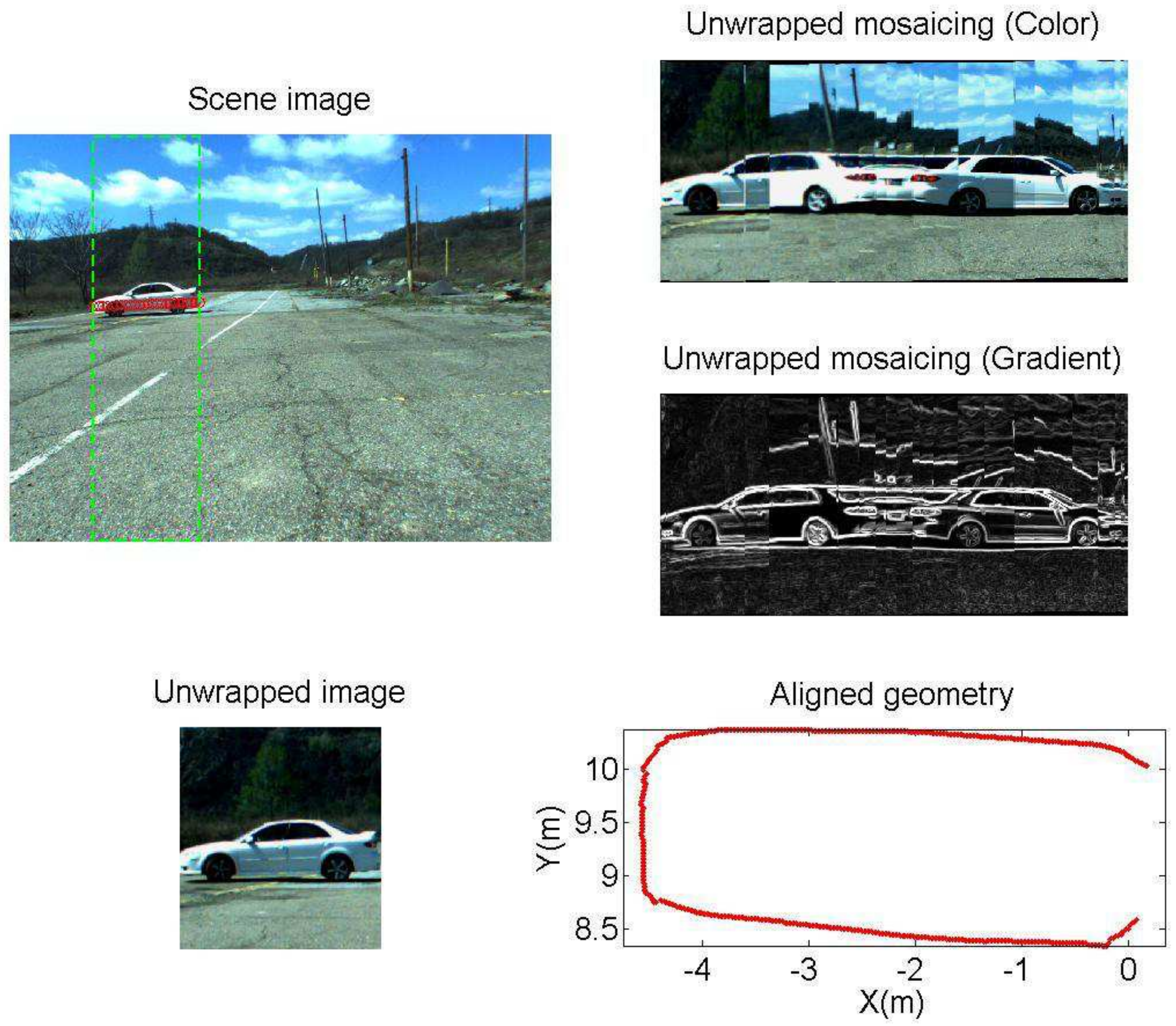
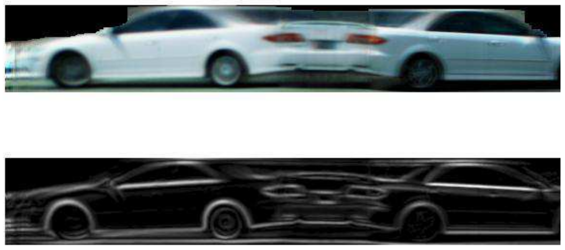
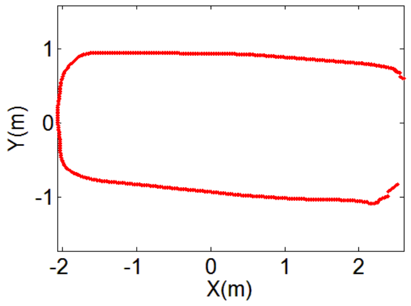


Figure 3.7: Weighted-mosaicing of unwrapped images. In the upper left image, the green box shows the object region of the tracked object. The lower left image shows the unwrapped image of the image in the green box. The upper right corner image shows the unwrapped mosaic of color images. The right center image shows the unwrapped mosaic of gradient images. Both unwrapped mosaics are obtained independently. The lower right image shows the approximate geometry corresponding to the unwrapped mosaics.


(a) Appreance (color/gradient)



(b) Geometry

Figure 3.8: Approximate model of a object. The approximate model consists of the unwrapped mosaic (color/gradient) and the approximate geometry of the object.

vertical boundary for the registration, but we did not estimate the height of the object in the object images. The goal of the approximate representation is to find a good representation of unknown objects for improving detection and tracking performance when they appear again.

In this section, we propose an approximate representation for an object. The final approximate model consists of the mean appearance of the foreground region and the mean geometry. Figure 3.8 shows the approximate model of an object consisting of the unwrapped mosaic (e.g., color and edge gradient) and the approximate geometry.

3.6.1 Height Estimation

The height is estimated by detecting a horizontal boundary in the accumulated object image. As the object moves, the background in the object image will typically change more dramatically than the object appearance, so we detect the point in the mean object image where the background variability begins to occur. In order to detect the height position, we need some discriminative features that can detect the appearance change.

We use graph-cut optimization to detect the object height. The graph cut optimization returns the label (e.g., $\{Foreground = 1, Background = 0\}$) of each row minimizing an energy function with two energy terms [60]. The first energy term is obtained by the gradient confidence and the second energy term is measured by the similarity between row pixels in the accumulated object images.

3.6.1.1 Energy Minimization

The horizontal boundary detection is a kind of binary labeling problem which determines whether a row is the foreground or not. With the gradient confidence $\mathbf{C}_G$ of registered gradient images and the color similarity H_C between two row images, the label of each row in the mean object image is assigned by minimizing the following energy function with graph-cut optimization.

$$E(X) = \sum_{j \in V} E_D(x_j) + \alpha \sum_{(j,j+1) \in E} E_S(x_j, x_{j+1}), \tag{3.14}$$

where $E_D(x_j)$ is the data term for each row j, and $E_S(x_j, x_{j+1})$ is the smoothness term related with two neighbouring rows j and $j+1$. The parameter α trades off the influence of these two energy terms.

The data term $E_D(x_j)$ is defined as the negative log likelihood of each row label l_j given the column image x_j.

$$E_D(x_j) = \begin{cases} -\ln p(l_j | x_j = 1) & \text{if } x_j \text{ is foreground} \\ -\ln p(l_j | x_j = 0) & \text{if } x_j \text{ is background} \end{cases}$$

The smoothness term $E_S(x_j, x_j + 1)$ penalizes the differences between adjacent labels. This term is defined as:

$$E_D(x_j, x_{j+1}) = \|x_j - x_{j+1}\| \exp\left(-\frac{\|H_j - H_{j+1}\|^2}{2\,\sigma^2}\right). \tag{3.15}$$

Figure 3.9 shows the estimated horizontal boundary in the mean object image.

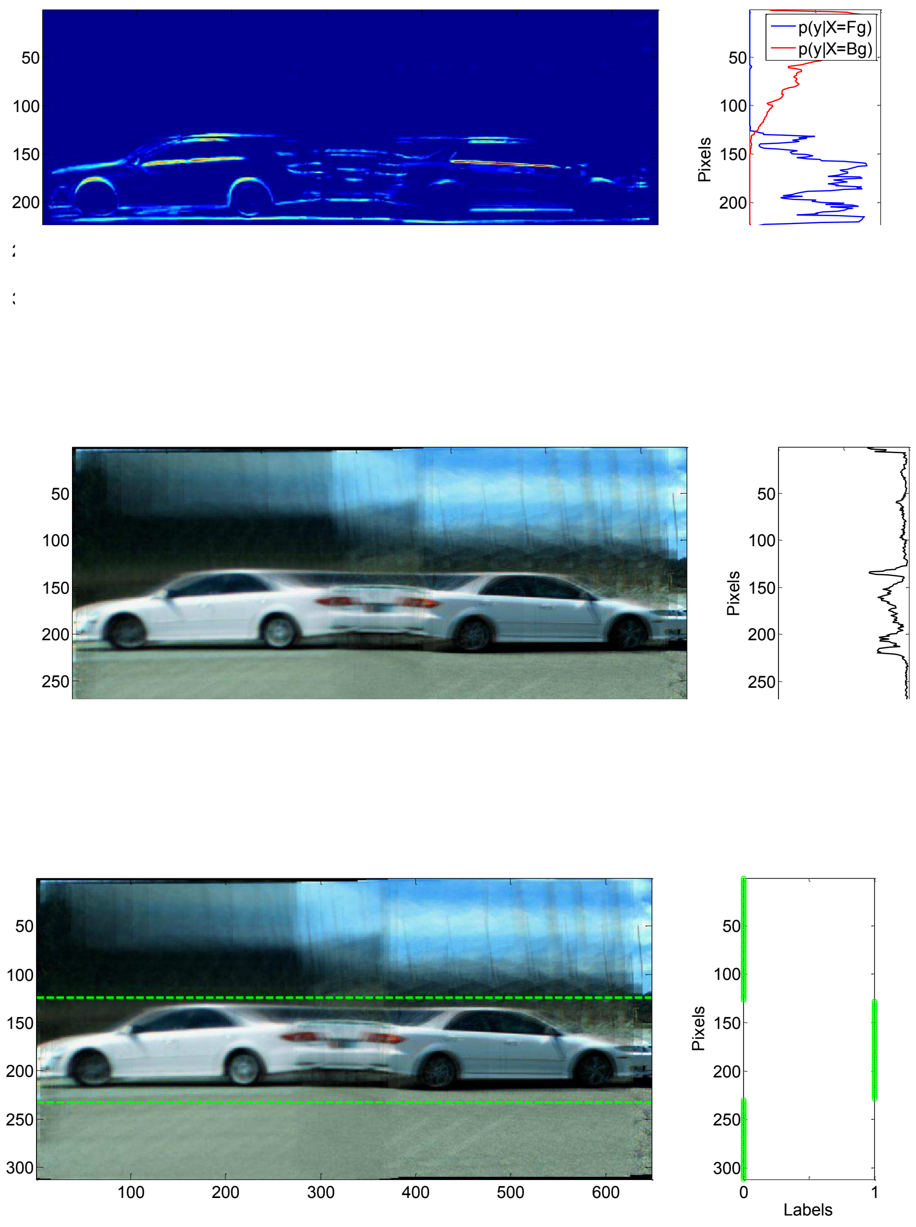
3.6.1.2 Gradient Confidence between Object Images

Suppose we have two unwrapped object images (U_{t-1}, U_t) which are already registered. We are able to measure the confidence of gradient field with an approach proposed in [61]. Let G_{t-1} and G_t be the gradient vector fields of the two images. Given the matching position $\mathbf{x}^* = (x^*, y^*)$, the gradient confidence C_G between the both images is given by:

$$\begin{aligned} C_G(U_{t-1} \cap U_t) &= \frac{1}{2}(\|G_{t-1}(x^*, y^*)\| + \|G_t(x^*, y^*)\|) \qquad (3.16)\\ &\quad -\|G_{t-1}(x^*, y^*) - G_t(x^*, y^*)\|, \\ C_G(U_{t-1} - (U_{t-1} \cap U_t)) &= \frac{1}{2}\|G_{t-1}(x^*, y^*)\|, \qquad (3.17)\\ C_G\{U_t - (U_{t-1} \cap U_t)\} &= \frac{1}{2}\|G_t(x^*, y^*)\|. \qquad (3.18)\end{aligned}$$

3.6.1.3 Histogram Intersection between Row Images

Let $U \in M \times N$ be the mean object image of the accumulated object images. Let H_j denote the color histogram of the jth row. We estimate the similarity between consecutive rows measuring the histogram intersection of H_j and H_{j+1} [57]. This is calculated as



(c)

Figure 3.9: Height estimation using 1D MRF model. Image (a) shows the likelihood of each row. Image (b) shows the similarity between adjacent rows. Image (c) shows the labeling result of each row in which the foreground row is labeled as 1 and the background is labeled as 0.

$$H_C(H_j, H_{j+1}) = \frac{\sum_{b=1}^{B} \min\left(\mathbf{h}_j(b), \mathbf{h}_{j+1}(b)\right)}{\sum_{b=1}^{B} \mathbf{h}_j(b)}, \tag{3.19}$$

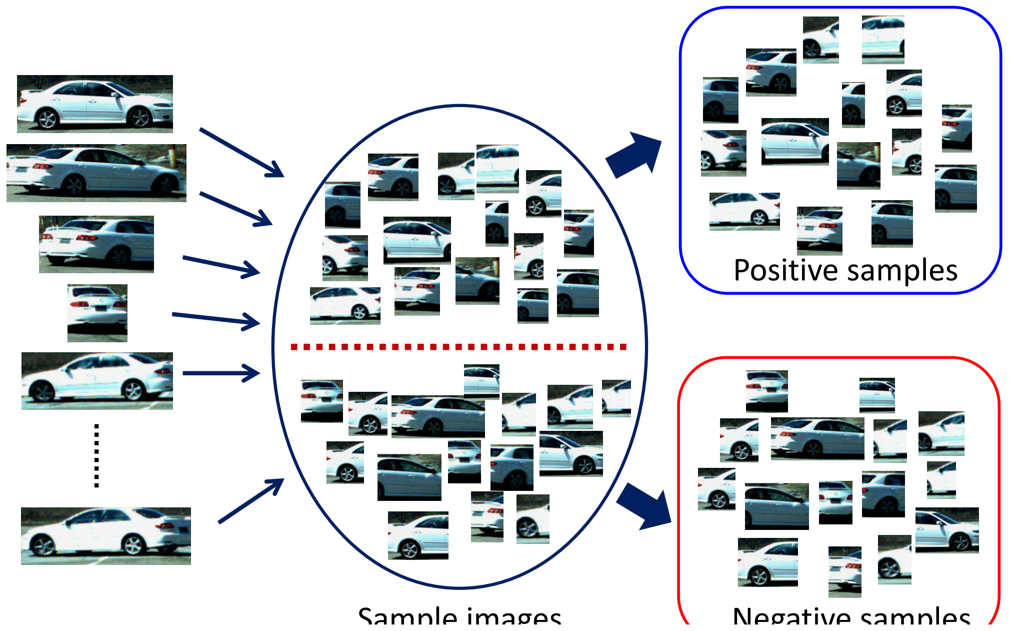
where B is the number of bins in the histogram. In our experiments, B is a set of 48 (i.e., each color channel has 16 bins).

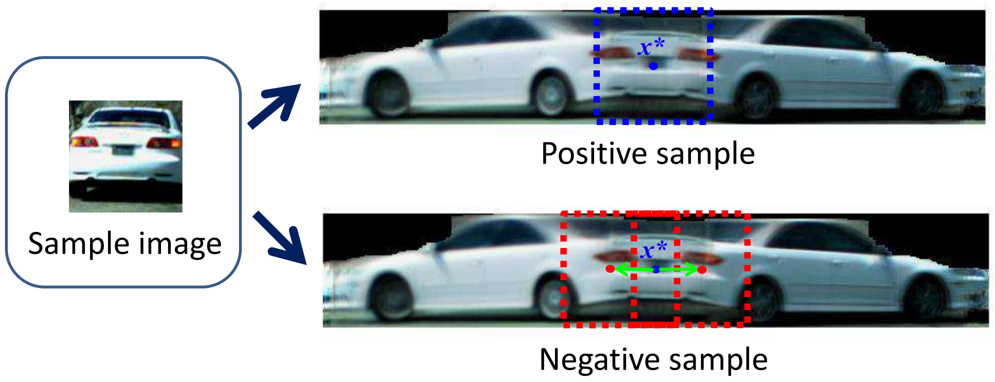
3.6.2 Approximate Model Training

The detection problem requires us to discriminate a target object in the scene. However, it is difficult to determine a threshold on a matching score to reject false positives. In order to solve the problem of threshold-based object detection, we apply supervised machine learning techniques using a support vector machine (SVM) and convert the output to a probability score.

Given the approximate model of an object and the unwrapped images which are used for the model, we can make training samples by registering the edge-gradient of the unwrapped images into the edge-gradient of the approximate model. The samples are generated by dividing each unwrapped image registered onto the approximate model into sub-images of different widths as shown in Figure 3.10. The width is limited to avoid generating sub-images that are too small, as shown in Figure 3.10 (a). We randomly choose half of the samples and assign them to the positive class. We measure the matching position on the approximate model, the size of width of the sub-image, and the cross-correlation score. We specify the remaining samples as negative samples and shift the original matching position of theses samples out of alignment in the column direction, as shown in Figure 3.10 (b). The matching position and the matching score are re-estimated.

We define the features ($\mathbf{x} \in Re_n$) as the column position of the matching position (x), the width of sub-image (w), and the cross-correlation score (m). Given the training input data $T =$





(b)

Figure 3.10: Sample generation for training the model. Image (a) shows sample generation with the unwrapped images which are used for creating the approximate model. Image (b) shows positive and negative sample generation. To generate the negative samples, we shift the matching position $\mathbf{x}^*$ from side to side by eight pixels.

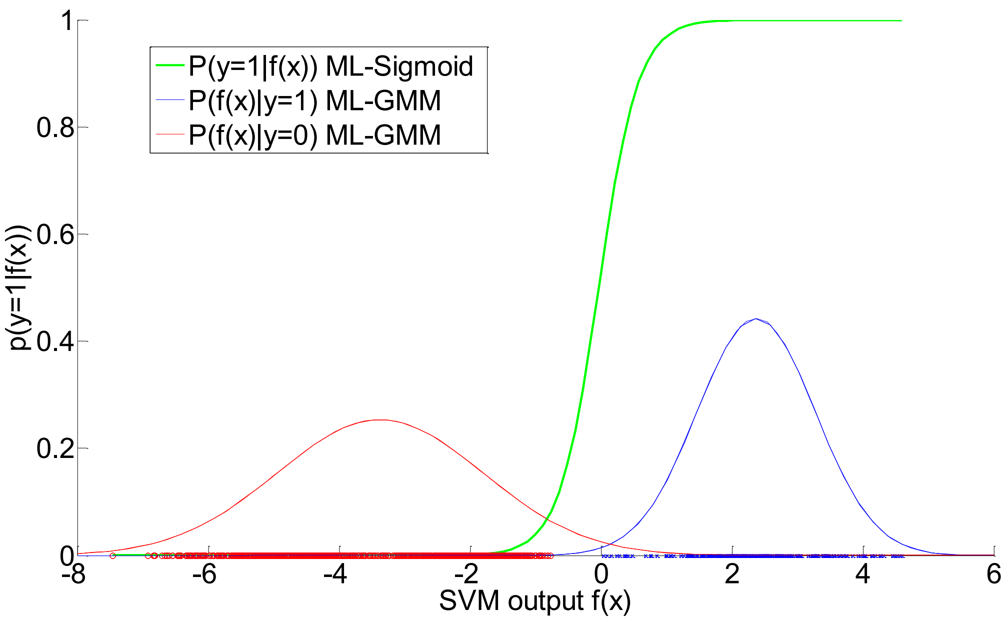


Figure 3.11: Probabilistic output of the training model.

$\{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_K, y_K)\}$ consisting of a set of features $\mathbf{x}_k$ derived from both samples and the binary hidden states y_k, we train a linear SVM classifier to correctly label novel data queries.

In order to compute the likelihood when an image is registered in the model image, we express the classifier output as a probability. Let $f : X \subseteq \Re^n \rightarrow \Re$ be a discriminant function trained from the input training data by SVMs. The aim is to estimate parameters of a posterior distribution $p(y|f(\mathbf{x}), \theta)$ of the hidden state y given the value of the discriminant function $f(\mathbf{x})$. The distribution is modeled by a sigmoid function as shown in Figure 3.11. We estimate the probabilistic output using the STPRtool in [62].

$$p(1|f(\mathbf{x}), \theta) = \frac{1}{1 + \exp\left(a_1 f(\mathbf{x}) + a_2\right)}, \tag{3.20}$$

which is determined by parameters $\theta = (a_1, a_2)$. The parameter θ is estimated by the maximum-likelihood estimation

$$\begin{aligned} \theta &= \arg\max_{\theta} \prod_{k=1}^{K} p(y_k|f(\mathbf{x_k}), \theta) \qquad (3.21) \\ &= \arg\max_{\theta} \sum_{k=1}^{K} \log p(y_k|f(\mathbf{x_k}), \theta). \end{aligned}$$

3.7 Experiments

A SICK LMS-221 lidar and a PointGrey Flea2 camera were used to acquire the data sets for our experiments. The lidar has a 180 degree horizontal field of view, with a line scanning frequency of 75 Hz and a 0.5° angular resolution. The camera has a 60° horizontal field of view, with a frame rate of 15 Hz and a resolution of 1024 by 768 pixels. These sensors were mounted on front of a vehicle. The lidar was placed at a height of 70 cm, and the camera was installed 30 cm above the lidar. The data from each sensor was time-stamped to enable synchronization.

Using the framework described above, we performed a series of experiments to evaluate the effectiveness of our approximate representation approach. In the first experiment, we evaluated the accuracy of the 3D sensor-pose estimation using our approximate representation algorithm. We then compared our representation algorithm to two state-of-the-art methods: iterative closet point (ICP) [63] and the color ICP approach [64]. Finally, we evaluated the detection and tracking performance of our approximate model.

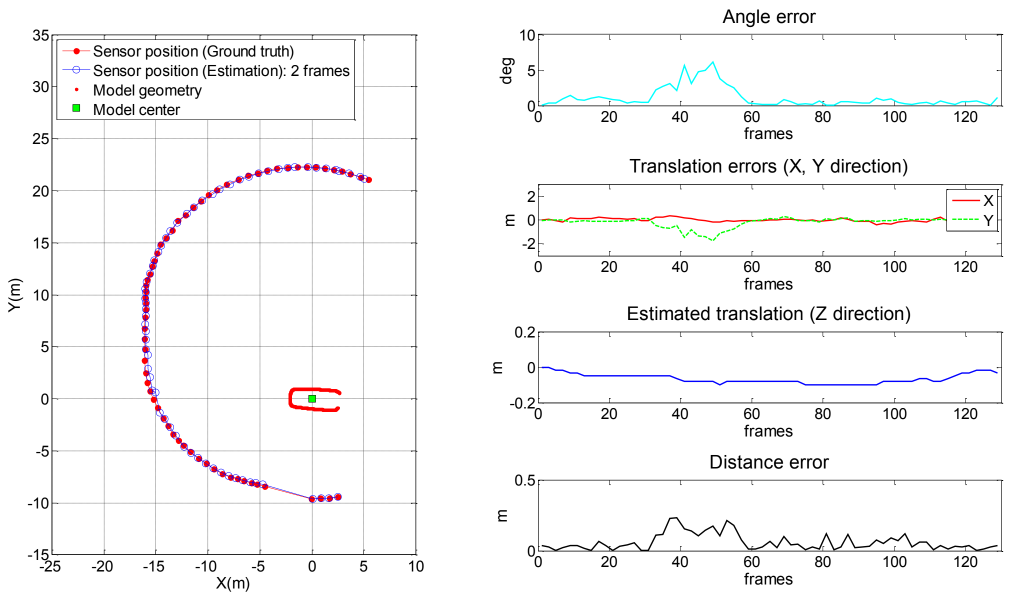
3.7.1 Evaluation of 3D Pose Estimation Accuracy

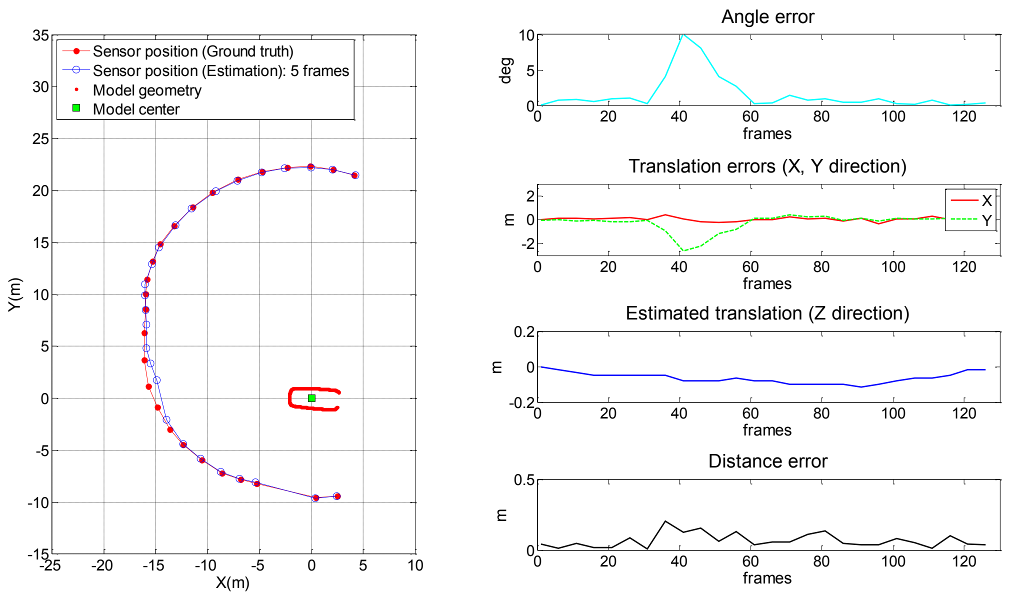
We evaluated the accuracy of the 3D sensor pose estimate. This evaluation was done using the best estimated sensor poses in object coordinate in which the poses are manually generated with

130 frames. The performance was evaluated as the angle error, the translation error on the XY-ground plane, and the distance error between the ground-truth position and the estimated position. Additionally, we estimated the translation in the Z direction. The testing was achieved with the two sets of different frame rates, i.e., two- and five-frame intervals. The object views vary more dramatically as the frame interval is increased.

Figure 3.12 shows the results of 3D sensor pose estimation. Our approach accurately estimates the sensor position independent of the frame rate. As shown in Figure 3.12 (a) and Figure 3.12 (b), the averages of the angle errors are 1.2° (2 frames) and 2.7° (5 frames), respectively. The translation errors in the Y direction on the XY-ground plane increase with the frame interval, but the translation errors in the Y direction do not vary significantly by the number of the frame interval. Angular error causes a large error in the Y translation direction because the angle error increases at the specific frames where the sensor position moves on the Y axis. Since we use lidar geometry for the modeling, the maximum distance errors of the two cases are less than $0.3m$. The largest angle errors occur when the object moves on the slanted ground because our approximate representation does not estimate roll and pitch motion.

Figure 3.13 shows the comparison of the sensor pose estimate using different frame rates. For the experiment, we used three different frame rates in which each performance was evaluated with two, three, and five frame intervals respectively. When we estimate the sensor pose with the frame rate of 3 frames/second, the frame interval is five, and the angle error is 2 times higher than the other cases, but it has similar error values in the other estimation interval. This result indicates that the lidar views a strong shape feature (such as a corner) in most frames except from 35 to 60. All of the three cases accurately estimate the pattern of the XY-ground plane because the estimated Z coordinate shows that the height of the ground plane in the middle frames is higher





(b)

Figure 3.12: Evaluation of 3D sensor pose estimate. The frame rate is 7.5 frames/second; that is, we use 65 frames of data with 2 frame intervals out of 130 data frames. The initial few frames are artificially generated for evaluating the robustness of our algorithm.

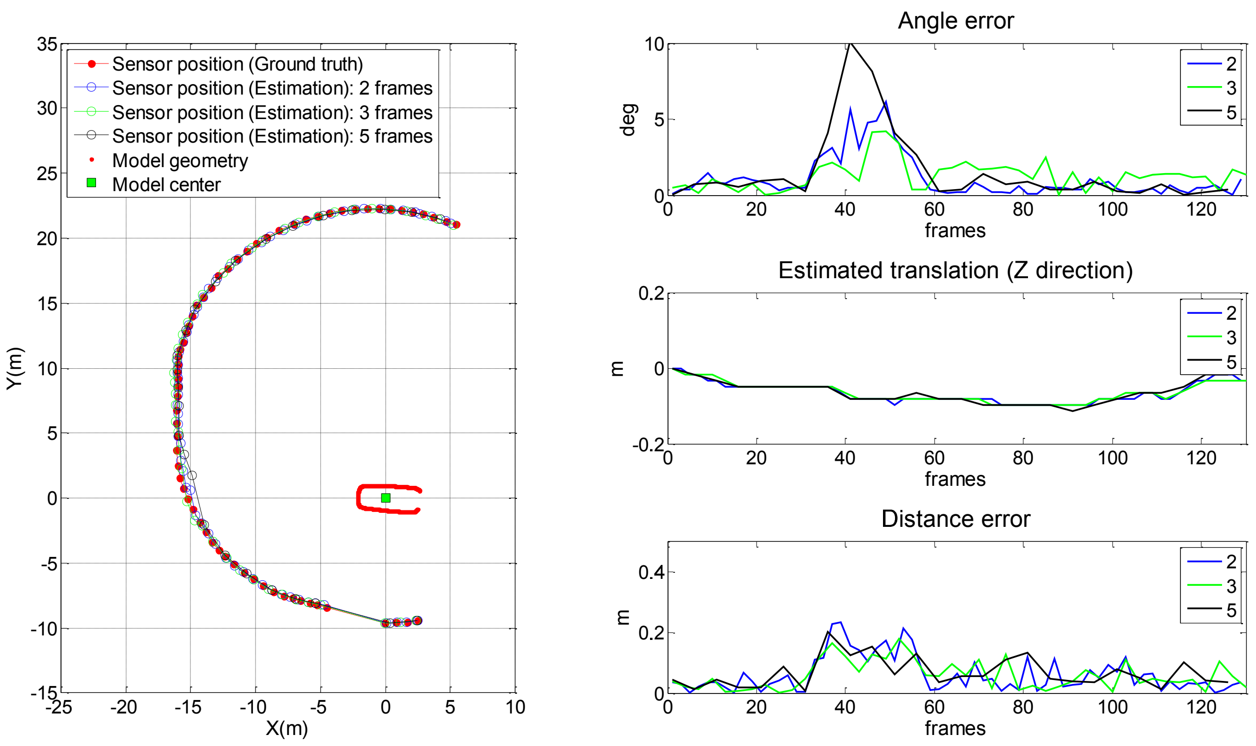


Figure 3.13: Comparison of 3D sensor pose estimates. The frame rates of each experiment are 7.5, 5, and 3 frames per second respectively. The model geometry shows the object shape created using the 7.5 frame per second.

3.7.2 Comparison against Existing Methods

In our second experiment, we compared the performance of our algorithm with two existing algorithms; the iterative closet points (ICP) approach and color ICP. Both algorithms are state-of-the-art approaches to register or align data, though there are many variants of these. The color ICP considers not only point data, but color as well. For the color ICP approach, the texture information was used with small image patches centered onthe projection of lidar measurements.

The results, shown in Figure 3.14, show that our algorithm outperforms both algorithms for the initial frame section of 1 to 40 frames. These frames consist of challenging cases such as abrupt view changes and featureless geometry data. The misalignment of the ICP algorithm was caused by some featureless geometry data (e.g., lines without corners). The ICP algorithm requires a good initial transformation in order to converge to the globally optimal solution, otherwise only a local optimum is achieved. The color ICP is robust with respect to featureless data, but the corresponding image patches are not discriminative because most cars have textureless surfaces and similar colors at the same height.

3.7.3 Evaluation of Detection and Tracking Performance

We test the performance of our detector and the tracking accuracy with 3D pose estimation of unknown object using our approximate model. For the experiment, we first create the approximate model of an object of interest using half of the frames in a full frame data set and then detect and track the object in the full frames. The rejection of wrong objects is achieved by checking the likelihood of each object using our training model.

Figure 3.15 shows the detection result using our approximate model. This dataset was obtained on a moving sensor platform. The approximate model was created using 10 frames in the dataset. We test the detection performance from the 1st frame to the 30th frames. The object was accurately detected with high probability in all frames. As shown in Figure 3.15 (d), when half of the object is out of the field of view (FOV) of the camera, our algorithm detected part of the object and aligned that part on the model.

Figure 3.16 shows the tracking result of an object in 3D space. For these experiments, we

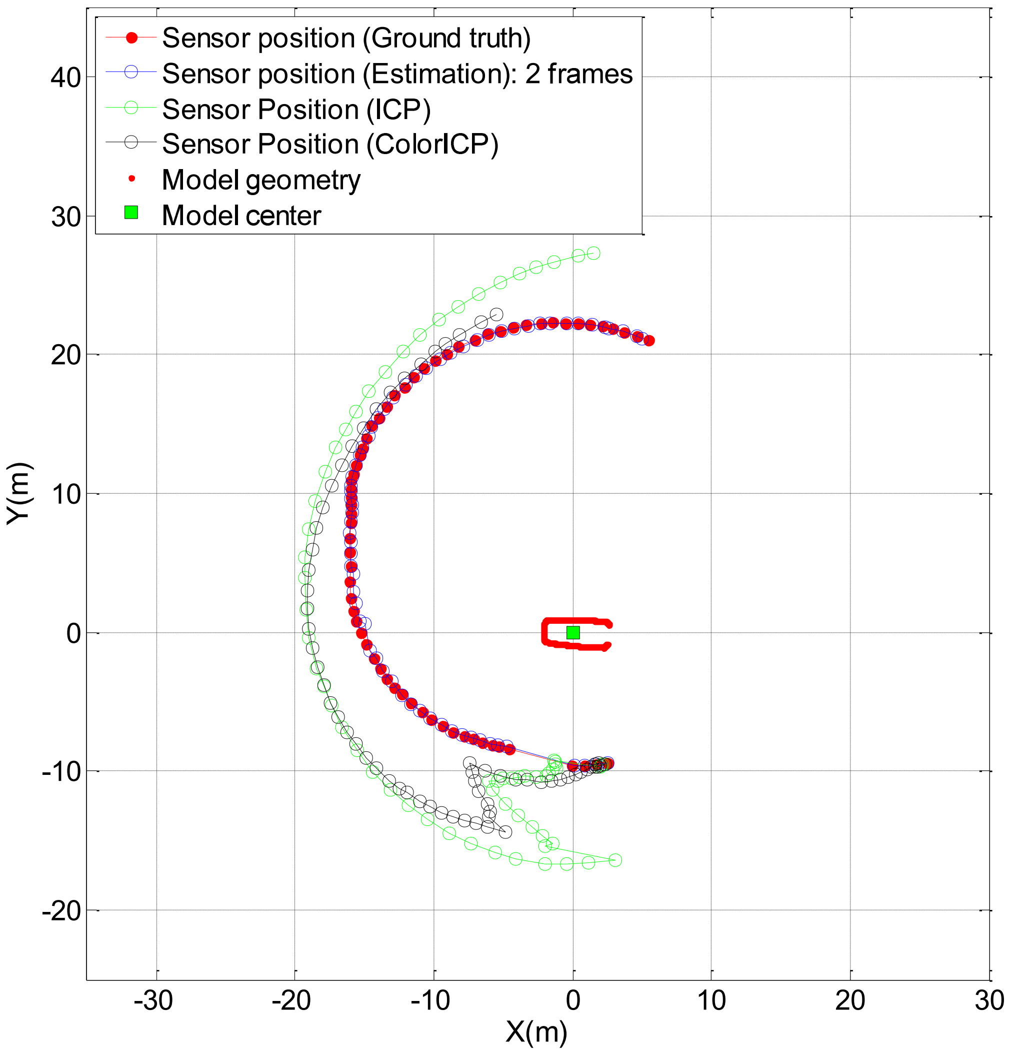
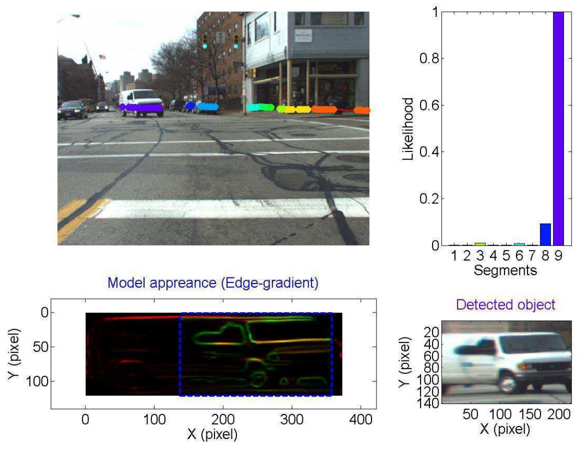
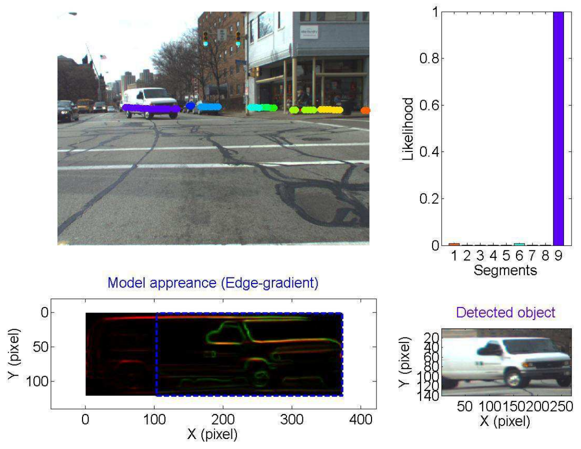
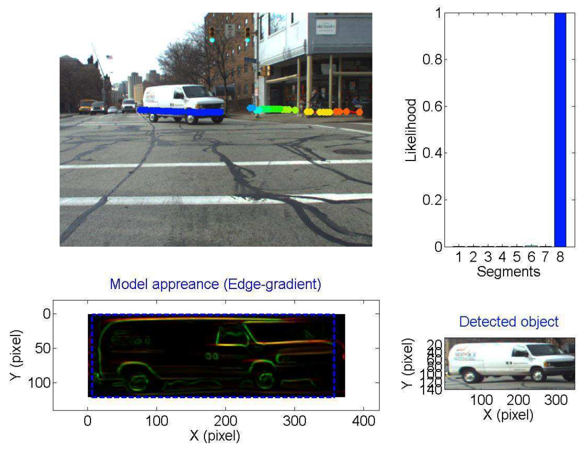


Figure 3.14: Comparison with ICP and color ICP. The color ICP algorithm performs better than the ICP algorithm in the case of abrupt view change, but color ICP also does not overcomes the estimation error of the sensor position in the case of featureless data sets.

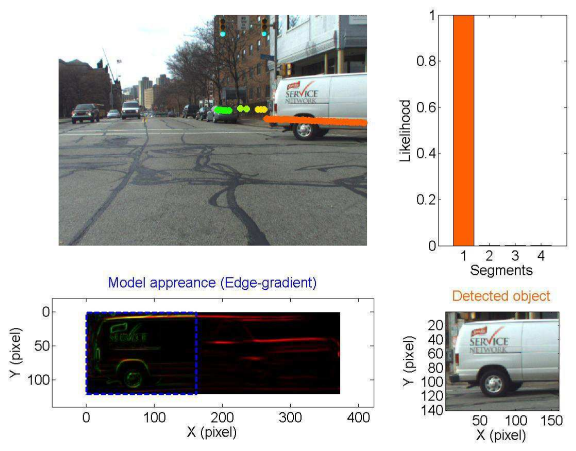
detect an object at 3 frame intervals in 130 frame data and estimate the 3D pose of the object. The tracking was performed by detecting the object and estimating the 3D pose. The tracking trajectory was obtained by tracking the estimated center of mass of the object. We predicted an accurate trajectory in all frames.







(c) t=15



(d) t=28

Figure 3.15: Object detection results. Image (a), (b), (c), (d) show the detection result at the 2nd frame, 7th frame, 15th frame, and 28th frame. In each result, the upper left image shows the segments in a scene, the upper right image shows the likelihood of each segment, the lower left image represents the matching position on the model, and the lower right image shows the detected object respectively. The bar color in the likelihood image represent the color of segments.

3.8 Summary

In summary, we have presented an algorithm approximately representing unknown objects in outdoor environments using lidar measurements and corresponding imagery. The success of the method

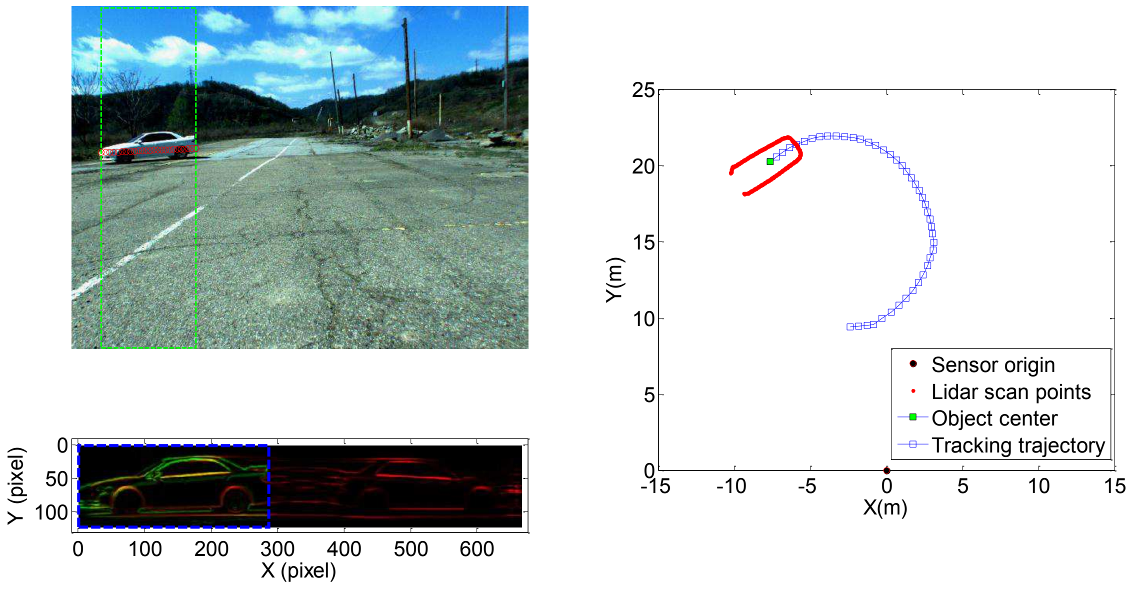


Figure 3.16: Object tracking with 3D pose estimation. We show the result at the last frame of the tracking sequence. In the scene image, the green box represents the object image region that is detected using the vertical boundary detection approach. The geometric model of the tracked object is shown as a red box in the right image. The image in the lower left corner shows the matching region on the model of the tracked object. The red shows the edge gradients of the model, the green describes the edge gradient of the object, and the yellow represents the edge gradient matched with high probability.

described in this chapter lies in three central stages which consist of unwrapping the object image with corresponding lidar geometry, registering unwrapped images, and training the approximate appearance model.

The approximate model consists of the unwrapped mosaics (i.e, color and edge-gradient), the approximate geometry, and the probabilistic training model of an object. These model components are created by an incremental modeling approach. Using the unwrapping approach, we create an appearance model which is invariant to viewpoint and scale changes. With the approximate model, we detect and track an object in a scene, and then estimate its 3D pose by 2D-to-2D matching between the unwrapped object image and its approximate model. We then use the approximate model to train a classifier and represent the detection result as a probabilistic score. The proposed algorithm is tested with the datasets we obtained in an outdoor environment. The result shows that the approximate model outperforms both ICP and color ICP alignment algorithms for estimating the sensor position.

Chapter 4

Car Modelling with Shape Prior

4.1 Introduction

In autonomous vehicle applications, some objects occur frequently (e.g., vehicles). Can we exploit knowledge about object class to do better for these objects? In this chapter, we apply the approximate representation approach to the specific object class of a car. Up to now, we were interested in creating a model without any prior knowledge of a target object. However, we can use prior knowledge to improve the accuracy of our models. For example, we can exploit the known symmetry of cars in order to recover a full model from a partial view.

We model cars using two different approaches. First, we create an explicit car model using the car shape prior and geometric constraints. Second, we extend the approximate modelling approach by exploiting prior knowledge such as symmetry of cars. The reason why we choose the specific object class of *car* is two-fold. The car is the class of object that describes the assumptions of a planar surface and rigid-body motion that we employed for the approximate model. Additionally, we can use the symmetry property and parallelism of edges.

Given a 2D car image, recovering a 3D car shape of real scale is a challenging problem. Much research has addressed the 3D car shape recovering problem, but there is little research on the 3D shape recovery of real scale. Kanade and others conducted early vision research on 3D reasoning from edges and their intersections in line drawings [29]. The idea is that reasoning about these junctions can constrain the interpretation of the drawings to allow 3D reasoning in the under-constrained case of 2D images. However, this approach has two central problems: (i) since they assume orthographic projection rather than perspective projection, it is difficult to recover a 3D object shape of real scale using the approach and (ii) the approach always needs line drawings of an object.

In the first approach, we recover a 3D car shape of real scale using the Origami world of a car. To do this, we impose a shape prior of a car and geometric constraints from lidar measurements. We can assume weak-perspective projection using lidar measurements instead of orthographic projection. We automatically obtain the line drawings of a car employing the 2D car alignment approach proposed by Li [28].

In the second model approach, we aim to model a car more explicitly than the approximate model with prior knowledge of a car. Since the approximate representation of an object is built using the unwrapped mosaicing approach, the model itself does not define a detailed shape. Knowing the object class is helpful for improving model accuracy with the approximate model. Extension of the approximate representation is summarized in two points of view: (i) once we build a half-car model including a side and rear or a side and front view, we create a full model by exploiting the symmetry of cars; and (ii) we recover the 3D car shape shape by carving out incorrectly projected regions resulting from the planar surface assumption.

4.2 Car Shape Recovery

In order to recover a 3D car shape, we first need to check the two existing methodologies introduced by Li and Kanade [28] [29]. First, Li's approach is to align a set of landmark points onto an image in which the points form the shape of a car. Second, Kanade suggests an approach to recover the 3D shape of an object given an orthographic perspective assumption.

The key idea of the robust 2D car alignment approach is to find optimal feature points minimizing a shape prediction error from random shape and pose hypotheses. The approach first detects a set of landmarks in the shape of a car. In this step, it uses the Histogram of Oriented Gradient (HOG) [65] to describe the image patches centered at the landmarks. After detecting a set of landmarks, the approach uses RANSAC to choose a subset consisting of correctly detected inliers [57]. Given the inliers, it estimates the car shape by solving a Bayesian inference problem. To evaluate the approach, Li manually labeled the landmarks of 5297 car images spanning a wide variety of sizes, types, viewpoints, lighting condition, and partial occlusions. He also manually classified the labeled data into five views: frontal, half-frontal, profile, half-back, and back views. In our approach, with the same labeled data and same parameter configuration, we estimate a 2D car alignment model at only the diagonal views such as profile and back or profile and frontal views. To recover a 3D car shape, we need information about at least two surfaces of a car.

The methodology for recovering 3D shape of an object consists of two stages: (i) apply a model of Origami World of an object and (ii) map image regularities into shape constraints. Origami World is a kind of model for converting line drawings to a set of surfaces and recovering geometrically plausible shapes (that is, 3D configurations). The world assumes that all the projection lines are orthogonal to the projection plane (i.e., orthographic projection) and surfaces in the Origami World are planar (i.e, the orientation is constant throughout a surface) [66]. The terminology for the Origami World generally follows the labeling theory of Waltz [67]. We first summarize the definitions of *edge*, *vertex*, *line*, *junction*, and *region* in the Origami World [66]:

- *Edge*: a straight boundary of a plane surface
- *Vertex*: a corner or point where edges of the surfaces meet
- *Line*: an orthographic projection of an edge onto the image plane
- *Junction*: the projection onto the image plane of a vertex or the point where an edge is interrupted by an occluding surface
- *Region*: an area in the image plane surrounded by lines

Here, an edge corresponds to a line in the image plane and a surface corresponds to a region in the image plane. Each edge is classified by its physical meaning in the scene. A line can be labeled with one of the four labels: $+$ (convex), $-$ (concave), $\leftarrow$, and $\rightarrow$ (occluding). See [66] for more details about Origami World.

Once we build a model of Origami World, we need additional constraints to recover the 3D shape of the object. In [29], Kanade used parallelism of lines and skewed-symmetry as the image regularities. He mapped the image regularities into shape constraints in the gradient space [1]. With

[1]With orthographic projection, a plane in the scene whose surface is visible by the viewer can be expressed as $-z = ax + by + c$. The two-dimensional space made of the ordered pairs (a, b) is called the gradient space $\mathbf{G}$. The gradient space is a good tool to represent surface orientation in the scene [68]

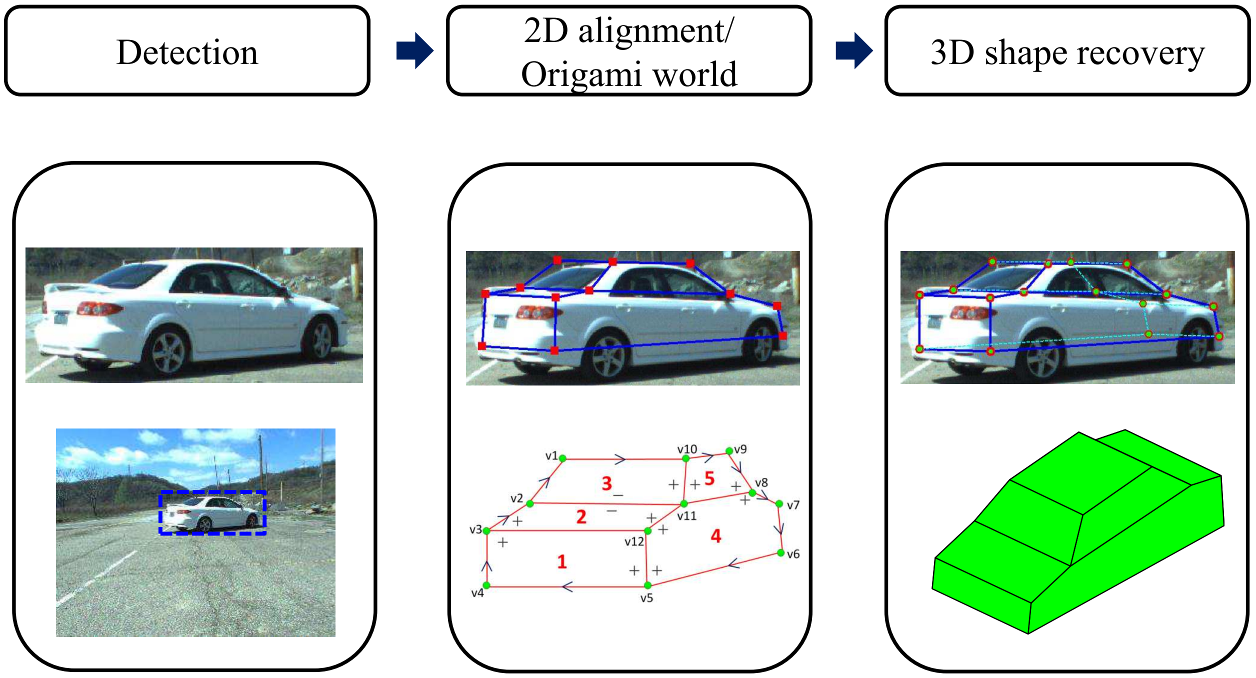


Figure 4.1: Overall process for a 3D car shape recovery. The processes consist of three steps: (i) car detection, (ii) 2D car alignment modelling, and (iii) 3D car shape recovery

the assumption of orthographic projection, we can explain the parallelism of lines in the gradient space. If a pair of boundary lines of two planes is parallel in the 3D space, the two planes intersect along those boundary lines moving one of the planes toward the other. Skew-symmetry property is a relaxed version of the real symmetry property. The skewed-symmetry means that a 2D shape in which the symmetry is found along lines not necessarily perpendicular to the symmetry axis, but at fixed angle to the axis.

Recovery of a 3D shape of a car is summarized in two stages: (i) we detect a car in the diagonal views and obtain the line drawings of the car using a 2D car alignment approach, and (ii) we recover a 3D car shape of real scale by solving an optimization problem to satisfy surface constraints and lidar constraints. Figure 4.1 shows the overall process for recovering a 3D car shape.

4.2.1 Car Detection and Alignment Model

In order to recover a 3D car shape, we need a specific view of a car, which means that a car should be detected at a diagonal view (i.e., profile and back or profile and frontal). Detecting the specific view of cars, we should make a model with many view-based car images and design a detector using the model. The car detector based on the multi-view model does not adequately detect the diagonal views of a car; it frequently detect the side view as the diagonal view [31]. Li's approach[2] fails to estimate the alignment model correctly if the views are incorrectly identified.

We use the approximate model to detect specific views of a car. Once we create an approximate model of a car, we can accurately estimate its 3D pose by matching an unwrapped image of the car onto the approximate model. When the unwrapped image of the car is aligned in specific regions on the approximate model, the view of the car is likely to be one of the diagonal views (i.e., half-frontal or half-back view). The specific regions on the approximate model are defined as areas including profile and back views or profile and frontal views at the same time. Once we detect a car in the diagonal views, we crop out the car region I_c.

In this thesis, we estimate the landmarks by using the 2D car alignment algorithm. The 2D car alignment algorithms returns optimal landmarks minimizing the alignment error of an input car image. In case of diagonal views, it provides 15 points which describe the shape of a car as shown in Figure 4.2. For recovering the 3D car shape, we use 13 landmarks, ignoring the two points obtained from both wheels. We will define the landmarks to be junctions in the terminology of Origami World.

[1]We does not use the multi-view detector which Li suggests in his paper.

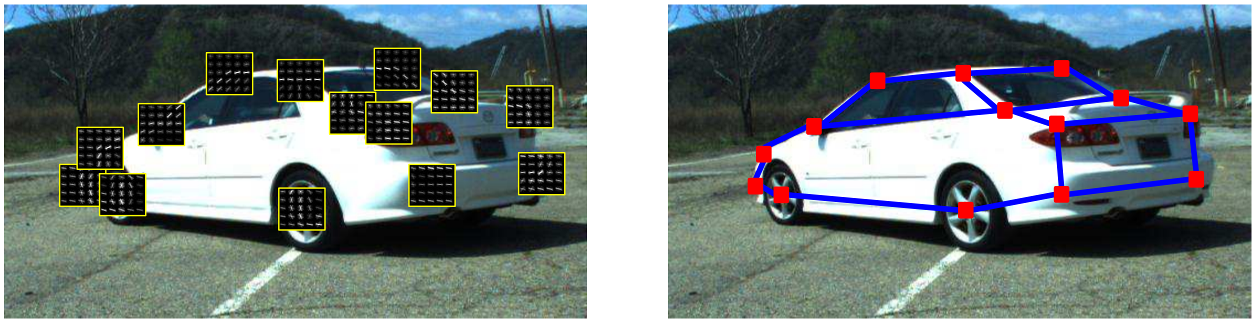


Figure 4.2: Landmarks detected in the 2D car alignment algorithm. Li's algorithm provides 15 landmarks in the diagonal view. We do not use the landmarks on the front and rear wheel in the alignment result. Left images shows the local descriptors used to detect the landmarks. Right image shows the 2D car alignment result obtained from Li's algorithm.

4.2.2 3D Car Shape Recovery using Origami World

From the result of 2D car alignment, we obtain the line drawings of a car. The line drawing of the car means not the 3D car shape, but a 2D projection of the 3D car shape. To recover a 3D car shape, we build a car model in the Origami world using the line drawing. The purpose of modeling in the Origami world is to obtain a labeled line drawing and a Surface Connection Graph (SCG). The labeling procedure consists of assigning one of the four labels ($+$, $-$, $\leftarrow$, or $\rightarrow$) to each line. The Surface Connection Graph (SCG) represents a surface as a node and a constraint between the surfaces as $+$ (Convex) or $-$ (concave). Figure 4.3 shows the results of line labeling and the SCG of a car.

In order to recover a 3D car shape at real scale, we first impose some constraints on the gradient space (intersection of surfaces, a skewed-symmetry heuristic, and a parallel-line heuristic, as previously defined). We define additional constraints from the shape prior of a car and geometric constraints from the lidar measurements (referred to as a vertical constraint and lidar constraint).

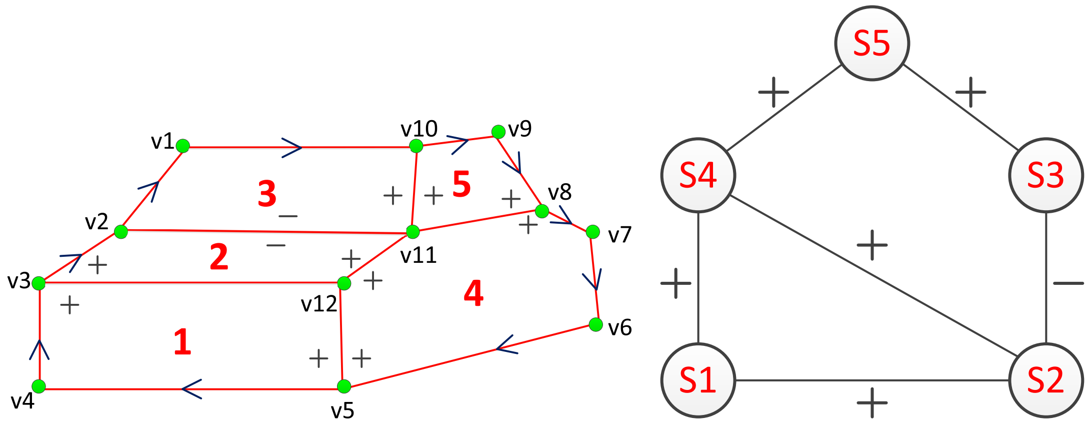
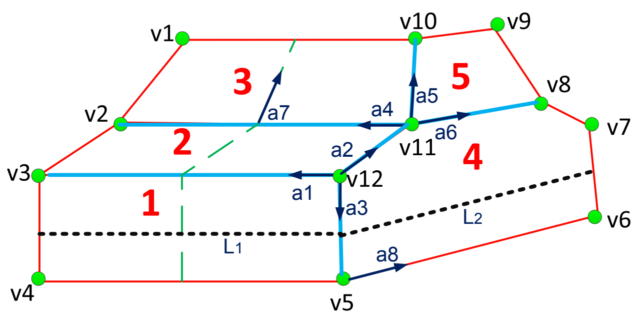


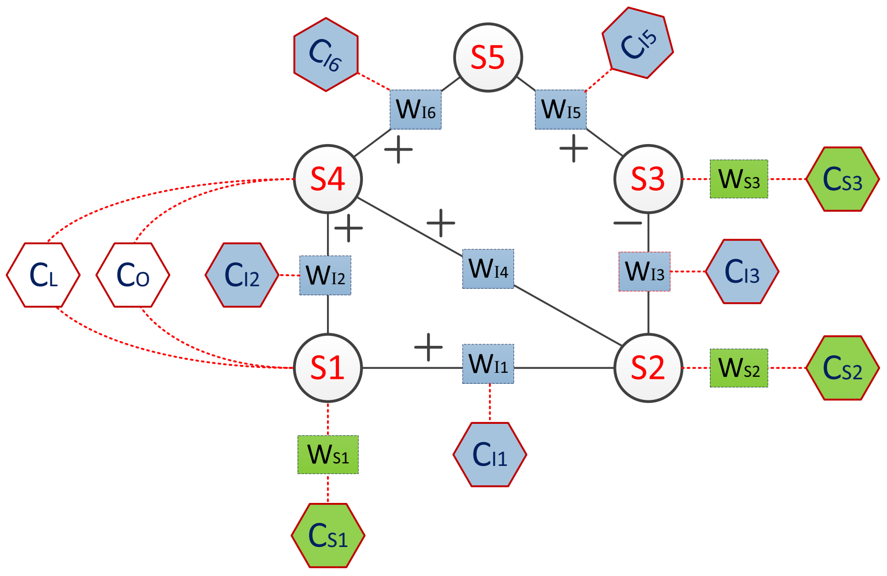
Figure 4.3: Labeled line drawing and surface connection graph of a car. In the left image, the numbers are region labels, 'v' describes the vertices of a car. In the right image, 'S' means the surface in 3D ,and the numbers correspond to the region labels in the left image.

Surface intersections require that two adjacent surfaces intersect at a given line. Parallelism constraints require that certain lines are parallel in 3D. Skewed-symmetry constraints require that both sides of a car are symmetric in 3D space. An orthogonal constraint requires that the side and the front or rear surfaces are orthogonal. A lidar constraint requires that the ratio of the length and width of a car measured from the lidar measurement is equal to the ratio of the estimated length and width satisfying the constraints. The constraints are represented in the surface connection graph as shown in Figure 4.4.

The 3D car shape is obtained by minimizing weighted sum of errors. Suppose that two regions are connected along a line with a direction ($\mathbf{a}_i = (\cos(\alpha_i), \sin(\alpha_i))$, $i = 1, \cdots, 8$) in the image as shown in Figure 4.4. The regions are given the gradients ($\mathbf{G}_i = (p, q)$, $i = 1, \cdots, 6$), and the real scales of length and width of a car are measured by lidar measurements. We can define each constraint as:



(a)



(b)

Figure 4.4: Surface connection graph with constraints. In top image, the blue lines describe the intersection line of two regions, arrows with 'a' labels indicate the direction vectors for each constraint, and the green line show the skewed-symmetry axis. In the bottom image, 'S' means the surface in 3D ,and the numbers correspond to the region labels in the top image, 'w' is the weight between connected surfaces, and 'C' is the constraints. The blue constraints show the intersection of surfaces, and the green constraints show the skewed-symmetry heuristic. The parallel-line heuristic is defined between 'S1' and 'S2', and between 'S2' and 'S3'. 'C_O shows the orthogonality between 'S1' and 'S4'. 'C_L shows the length constraint of 'S1' and 'S4'.

Intersection of surfaces: two adjacent surfaces intersect at a given line,

$$\begin{aligned}
\mathbf{G}_1 \cdot \mathbf{a}_1 &= \mathbf{G}_2 \cdot \mathbf{a}_1 \\
\mathbf{G}_2 \cdot \mathbf{a}_2 &= \mathbf{G}_4 \cdot \mathbf{a}_2 \\
\mathbf{G}_1 \cdot \mathbf{a}_3 &= \mathbf{G}_4 \cdot \mathbf{a}_3 \\
\mathbf{G}_2 \cdot \mathbf{a}_4 &= \mathbf{G}_3 \cdot \mathbf{a}_4 \\
\mathbf{G}_3 \cdot \mathbf{a}_5 &= \mathbf{G}_5 \cdot \mathbf{a}_5 \\
\mathbf{G}_4 \cdot \mathbf{a}_6 &= \mathbf{G}_5 \cdot \mathbf{a}_6,
\end{aligned}$$

$$\begin{aligned}
e_I &= \|(\mathbf{G}_1 - \mathbf{G}_2) \cdot \mathbf{a}_1\|^2 + \|(\mathbf{G}_2 - \mathbf{G}_4) \cdot \mathbf{a}_2\|^2 + \|(\mathbf{G}_1 - \mathbf{G}_4) \cdot \mathbf{a}_3\|^2 + \|(\mathbf{G}_2 - \mathbf{G}_3) \cdot \mathbf{a}_4\|^2 \\
&+ \|(\mathbf{G}_3 - \mathbf{G}_5) \cdot \mathbf{a}_5\|^2 + \|(\mathbf{G}_4 - \mathbf{G}_5) \cdot \mathbf{a}_6\|^2. \qquad (4.1)
\end{aligned}$$

Skewed-symmetry heuristic: both sides of a car is symmetric in 3D space,

$$\begin{aligned}
(\mathbf{G}_1 \cdot \mathbf{a}_1)(\mathbf{G}_1 \cdot \mathbf{a}_3) + \cos(\alpha_1 - \alpha_3) &= 0 \\
(\mathbf{G}_2 \cdot \mathbf{a}_4)(\mathbf{G}_2 \cdot \mathbf{a}_2) + \cos(\alpha_4 - \alpha_2) &= 0 \\
(\mathbf{G}_3 \cdot \mathbf{a}_4)(\mathbf{G}_3 \cdot \mathbf{a}_7) + \cos(\alpha_3 - \alpha_4) &= 0,
\end{aligned}$$

$$\begin{aligned}
e_S &= \|(\mathbf{G}_1 \cdot \mathbf{a}_1)(\mathbf{G}_1 \cdot \mathbf{a}_3) + \cos(\alpha_1 - \alpha_3)\|^2 + \|(\mathbf{G}_2 \cdot \mathbf{a}_4)(\mathbf{G}_2 \cdot \mathbf{a}_2) + \cos(\alpha_4 - \alpha_2)\|^2 \\
&+ \|(\mathbf{G}_3 \cdot \mathbf{a}_4)(\mathbf{G}_3 \cdot \mathbf{a}_7) + \cos(\alpha_3 - \alpha_4)\|^2. \qquad (4.2)
\end{aligned}$$

Parallel-line heuristic: certain lines are parallel in 3D,

$$e_P = \frac{1}{2}(\|(\mathbf{G}_1 - \mathbf{G}_3) \cdot \mathbf{a}_1\|^2 + \|(\mathbf{G}_1 - \mathbf{G}_3) \cdot \mathbf{a}_4\|^2). \tag{4.3}$$

Orthogonal constraints: the vertical component of the surface normal be consistent across surfaces,

$$e_O = \|(\mathbf{G}_1(1) - \mathbf{G}_2(1))\|^2 + \|(\mathbf{G}_1(2) - \mathbf{G}_4(2))\|^2. \tag{4.4}$$

Lidar constraints: the surface normal and lidar scan line is orthogonal and lidar scans of side and the front or real surface is also perpendicular,

$$e_L = \|\{L_1\|\mathbf{a}^{\cdot}{}_8\| - L_2\|\mathbf{a}^{\cdot}{}_1\|\}\|^2, \tag{4.5}$$

where $\|\mathbf{a}^{\cdot}{}_1\|$ and $\|\mathbf{a}^{\cdot}{}_8\|$ are the magnitude of the projection vector on the image of the $\mathbf{a}_1$ and $\mathbf{a}_8$ vectors.

These constraints can be solved for using a weighted least squares framework using the following equation 4.6. Each constraint is part of a weighted sum in an overall objective function to be minimized, with the total of the weighting coefficients constrained to sum to one. The errors due to each component are Intersection of surfaces(E_I), Skewed-symmetry heuristic(E_S), Parallelism heuristic(E_P), Orthogonal constraints(E_V), and Lidar constraints(E_L):

$$E = w_I \sum E_I + w_P \sum E_P + w_S \sum E_S + w_O \sum E_O + w_L \sum E_L \tag{4.6}$$

where the weight values are estimated empirically ($w_I = 0.25$, $w_P = 0.15$, $w_S = 0.3$, $w_O = 0.1$, $w_L = 0.2$), and $\sum(w_I + w_P + w_S + w_O + w_L) = 1$.

4.2.3 Experiments

We analyzed the effect of the lidar constraints by comparing to the car shape recovered without the lidar constraints. Figure 4.5 shows the recovered 3D car shape using the shape constraints based on orthographic projection. Although the vertices of the recovered car shape are accurately placed on the image, the shape does not represent the real scale of the car due to the orthographic assumption. Figure 4.6 shows the recovered car shape with both of the shape constraints and the lidar constraints. The recover car shape represents the real scale of the car. Using the lidar measurements makes it possible to use weak perspective projection to recover the 3D shape instead of orthographic projection.

We evaluate the performance of our approach recovering the 3D car shape of different types of cars. These results are shown in Figure 4.7.

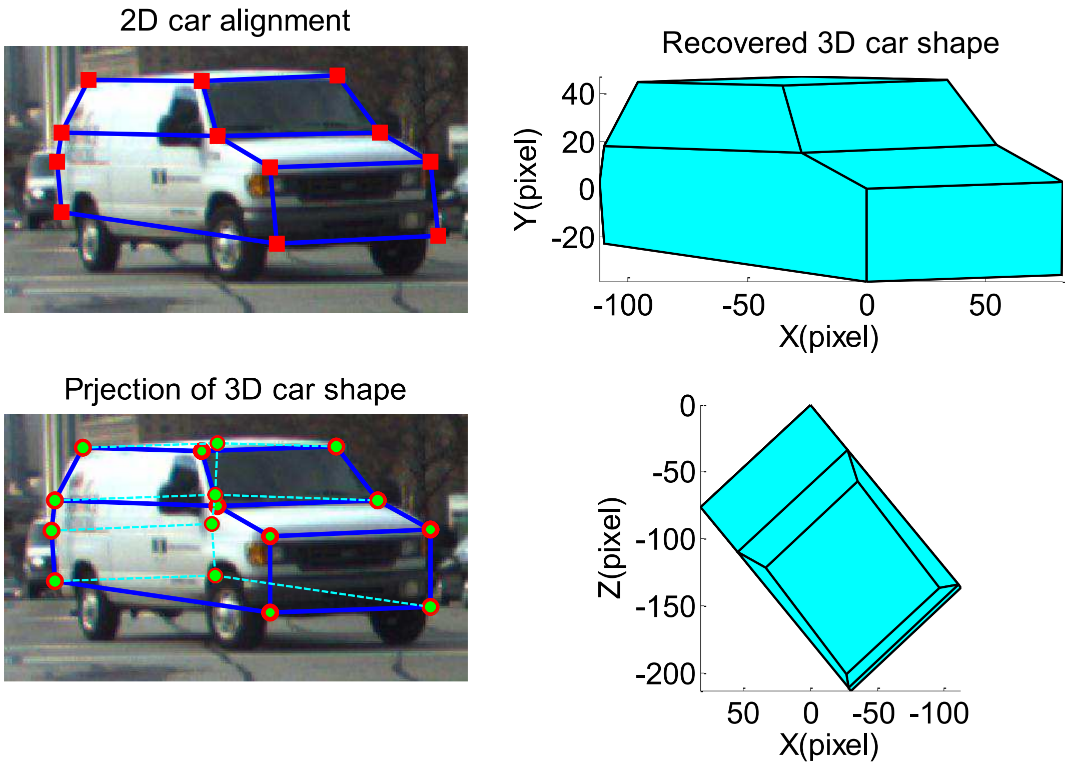


Figure 4.5: 3D car shape recovery without lidar constraints. The edge length of the recovered car is measured in pixels. The ratio of the width to length of the car is generally 1:2.5, but the ratio of the recovered car is 1:1.5. The red square shows the landmarks detected by the 2D car alignment approach. The red circle is the projection of the vertex of the recovered shape. The blue line shows the edge of the car.

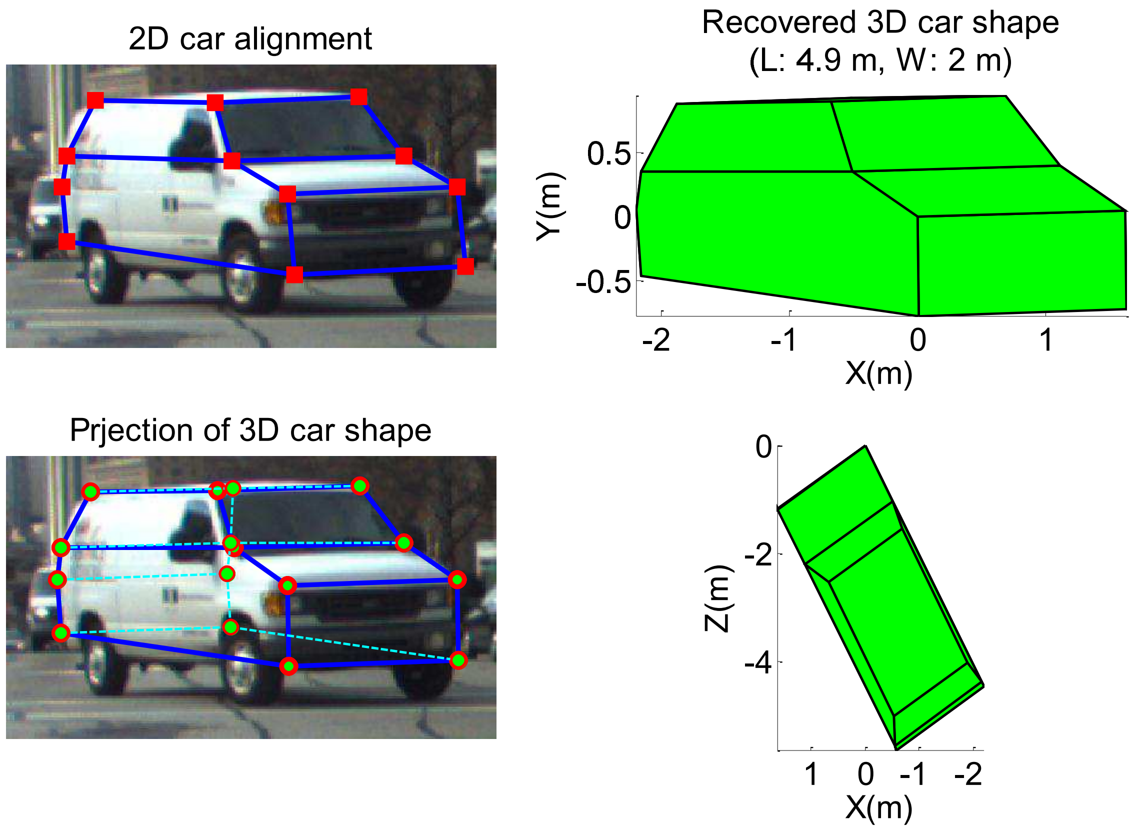
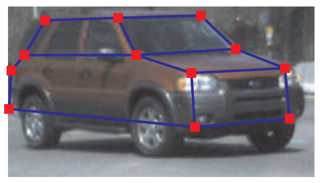
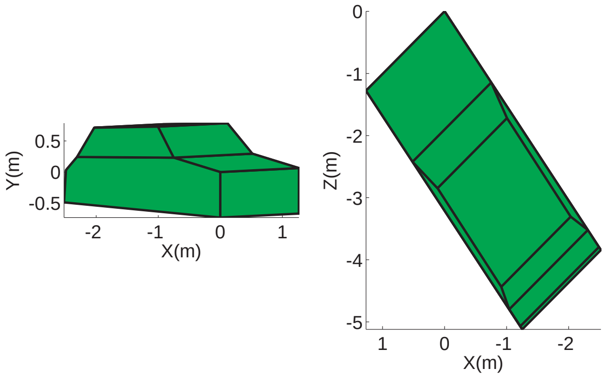


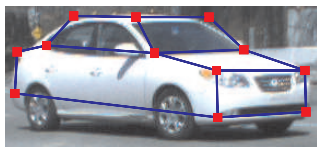
Figure 4.6: 3D car shape recovery with lidar constraints. The recovered car shape has real scale of the car with edge length being recovered in meters. The estimated length and width are 4.9 m and 2 m respectively.



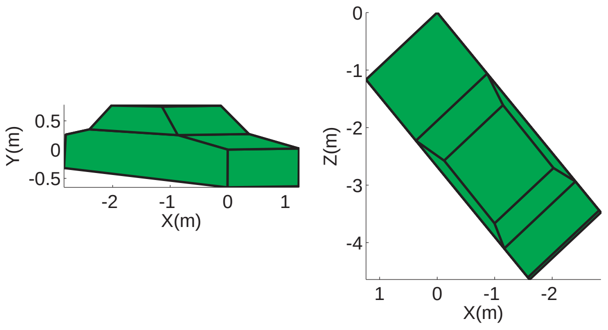
(a)



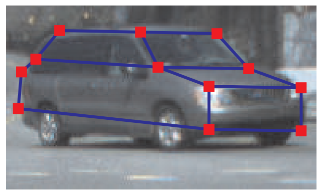
(b)



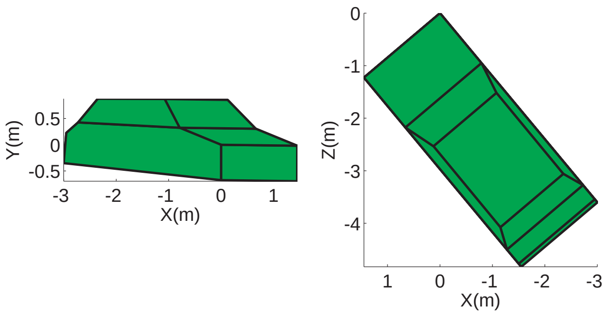
(c)



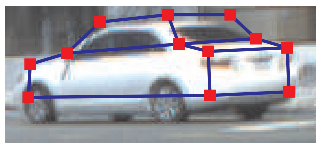
(d)



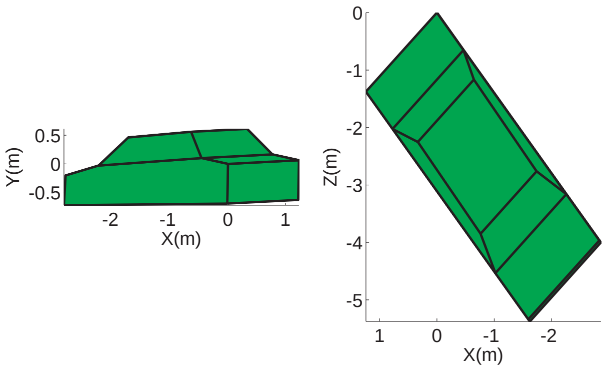
(e)



(f)



(g)



(h)

Figure 4.7: Recovered 3D car shapes.

4.3 Car Model using Approximate Representation

With the approximate model, we may reconstruct 3D models of cars because the approximate model includes the 2D geometry and height of a car. However, the approximate model represents a cylindrical shape, not a 3D shape of a car. The approximate model of a car is incrementally created with accumulated multiple views of the car. We need a method to recover a 3D from a set of images, such as *space carving.* While this technique works well under controlled conditions, very little is known about 3D object reconstruction under conditions with dramatically changing illumination.

We reconstruct a 3D car shape with a foreground segmentation approach and shape prior of a car. The explicit foreground region of an object can be segmented using the unwrapped mosaic image that we used in the approximate modelling step in Chapter 3 because the background typically changes more dramatically than the object appearance. However, it is difficult to reconstruct the 3D car shape through the segmentation process due to the appearance of the approximate model of the car. During the unwrapping process, parts of the image (such as the rear window) can be warped depending on their depth, since the lidar only returns the depth of a stripe on the object. Fortunately, in the case of a car, we can use two strong properties: (i) the two sides of a car are symmetric, and (ii) the profile view is orthogonal to the frontal and rear views.

A 3D car shape is created by mirroring a segmented car appearance on the symmetric axis of the model. The modeling process is consists of two stages: (i) foreground segmentation in the mean appearance of unwrapped images, and (ii) mirroring of car geometry and appearance.

4.3.1 Foreground Segmentation

The foreground segmentation is performed by energy minimization based on graph-cut optimization. Through the approximate modeling, we estimated the initial height and superposed the gradient confidence map on the unwrapped mosaic image of an object. We can make statistical models with respect to the foreground and background using the color texture of the unwrapped mosaic image. Given the statistical model of the foreground and background, we pose the segmentation problem in the binary labelling framework using graph-cut optimization [60].

Given the initial height estimation result in Chapter 3, we can deduce that the region below this height is likely to be in the foreground, and the region below it is more likely to be in the background. We segment the foreground region by building statistical models of pixel intensity or color. The statistical model can be represented as a mixture of Gaussian distribution of the pixel distributions over pixel-level characteristics [69]. However, segmentation based on the Gaussian mixture model (GMM) does not take into account spatial information [70]. Therefore, to improve the accuracy of segmentation, we apply a spatial MRF model for foreground segmentation [60].

4.3.1.1 Modelling of a Mixture of Gaussian Distribution

Once we select the foreground and background image regions, we represent those images as a mixture of K Gaussian distributions. We model the foreground and background with different number of Gaussians. The GMM distribution has the form:

$$p(x) = \sum_{i=1}^{K} w_i \cdot N(x|\mu_i, \Sigma_i), \text{ where} \sum_{i=1}^{K} w_i = 1, 0 \leq w_i \leq 1.$$

where $\mu_i$is the mean value, Σ_i is the covariance, and w_i is the prior probability (weight) of the ith

Gaussian.

Given a set of pixel values $X = x_1, x_2, \cdots, x_N$ drawn from an unknown distribution (assumed to be a GMM), the set can be modeled by estimating the parameters $\theta = [w_1, \cdots, w_K; \mu_1, \cdots, \mu_K; \Sigma_1, \cdots, \Sigma_K]$ of the Gaussian mixture model that best fits the data.

We can estimate the parameter θ by maximizing the likelihood $p(X|\theta)$ of the data with respect to the model parameters:

$$\theta^* = \arg\max_{\theta} p(X|\theta) = \arg\max \prod_{j=1}^{N} p(x_j|\theta). \tag{4.7}$$

One of the most popular approaches to maximize the likelihood is to use the Expectation-Maximization (EM) algorithm. The EM approach introduces a hidden variable such that its knowledge would simplify the maximization of the likelihood. At each iteration, the EM algorithm estimates the distribution of the hidden variable given the data and the current value of the parameters (E-step) and maximizes the joint distribution of the data and the hidden variables (M-step) [69].

4.3.1.2 Energy Minimization

Once we model the foreground and background using a GMM, segmenting the object image can be posed in the framework of a binary labeling problem. This labelling problem can be effectively solved using a Markov Random Field (MRF) model. The MRF is a graphical model of a joint probability distribution. It consists of an undirected graph $G = (V, E)$ in which the nodes V represent random variables and the edges E represent direct correlations between variables [60].

Considering an image (I) constituting of a set of pixels, we can define the set of all pixels with a given color as the nodes (V) and the set of all adjacent pixel pairs as the edges (E). Foreground and background segmentation problem consists to assigning a unique label x_i to each pixel $i \in V$, e.g. $x_i \in L$, where L is a set of region labels (foreground (= 1), background (= 0)). Given foreground and background GMMs $p(z_{Fg})$ and $p(z_{Bg})$, the label x_i of ith pixel can be estimated by inferring the Maximum a posterior solution (MAP) of the MRF model

$$x_{i^*} = \arg\max_{x_i} p(x_i|z) = \arg\max_{x_i} p(x_i|z_F g, z_B g) \tag{4.8}$$

which is equivalent to an optimization problem to minimize the following energy function $E(X)$ by the Hammersley-Clifford theorem [69].

$$E(X) = \sum_{i \in V} E_D(x_i) + \alpha \sum_{(i,j) \in E} E_S(x_i, x_j) \tag{4.9}$$

where $E_D(x_i)$ is the data term for each pixel i, and $E_S(x_i, x_j)$ is the smoothness term relating two neighbouring pixels i and j. The parameter α trades off the influence of these two energy terms.

The data term $E_D(x_i)$ is defined as the negative log likelihood of each pixel label l_i given the GMMs of the foreground and background:

$$E_D(x_i) = \begin{cases} -\ln p(l_i|x_i = 1) & \text{if } x_i \text{ is foreground} \\ -\ln p(l_i|x_i = 0) & \text{if } x_i \text{ is background} \end{cases}$$

The smoothness term $E_S(x_i, x_j)$ penalizes the difference between the labels of adjacent pixels i and j. The smoothness term is defined over 4-connected neighbors. We use the following penalty function [60]:

$$E_S(x_i, x_j) = \|x_i - x_j\| \, exp\left(- \frac{\| \|I_i - I_j\| \|^2}{2\,\sigma^2} \right) \tag{4.10}$$

4.3.2 Exploiting Prior Knowledge

Given the segmented foreground image, the 3D car shape is reconstructed by exploiting the car shape prior . The shape modelling is performed by mirroring the segmented appearance and the corresponding geometry across an axis of symmetry. The symmetry axis is estimated by aligning the mirror image onto the reference model image, because when we fold the aligned image, the center column is the symmetric column. The mirrored geometry is estimated by the symmetric axis which is measured by the corresponding geometry of the symmetric column.

In order to detect the symmetric column, the reference image and its mirror image should be aligned. Aligning between the both images is performed with sub-regions on the aligned image because the sub-regions are a common region between the reference image and its mirror image. Thesubregion consists of a frontal or rear region on the matched image. The sub-image region is estimated using the corresponding geometry because the shorter line generally represents the geometry of front or rear of a car.

The reference geometry corresponding to the reference image consists of two lines in which the number of points of the geometry is same with the number of columns of the image. Considering both end points $(\mathbf{p}_1, \mathbf{p}_N)$ of the geometry, the corner point $\mathbf{p}_c$ is a point which has the maximum distance from the line passing trough the both end points to points of the geometry. The column indexes of the sub-image region are defined as follow:

$$R_I = \begin{cases} [1, c] & \text{if } \|\mathbf{p}_1 - \mathbf{p}_c\|_2 < \|\mathbf{p}_c - \mathbf{p}_N\|_2 \\ [c, n] & \text{if } \|\mathbf{p}_1 - \mathbf{p}_c\|_2 > \|\mathbf{p}_c - \mathbf{p}_N\|_2 \end{cases}$$

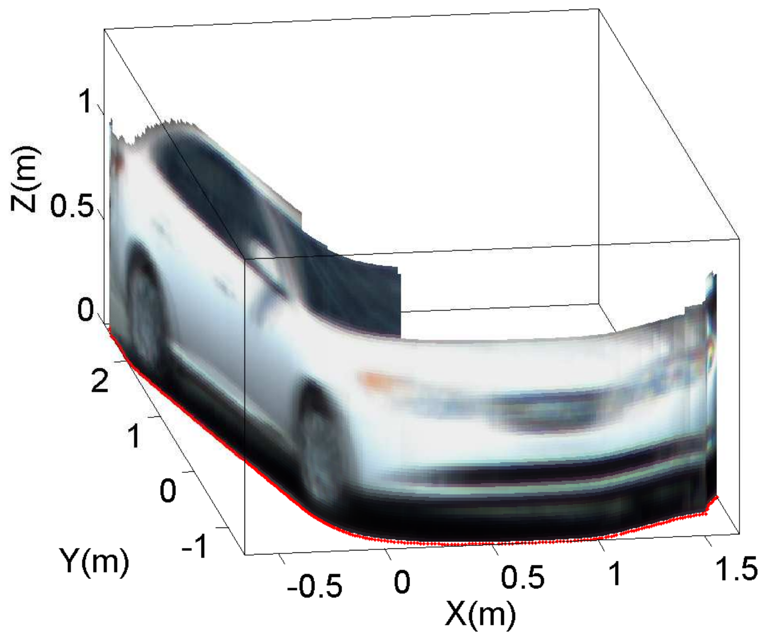
The aligned image is created by matching sub-images. The best matching position is estimated using a sliding window approach. The column of symmetry is easily measured by folding the aligned image until it overlaps perfectly. Once we estimate the symmetric column on the aligned image, the symmetric axis of the geometry is a line passing trough the point corresponding the symmetric column. The symmetric line is also parallel to the fitted side geometry of the car.

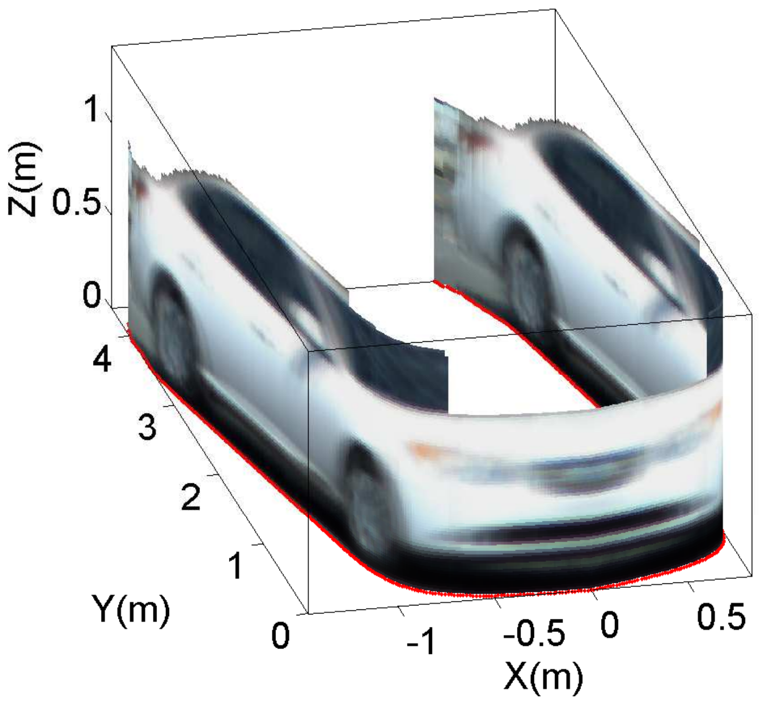
4.3.3 Experiments

Figure 4.8 shows a 3D car model built with the shape prior. For this experiment, we used the approximate model of a car whose side and front are detected in the modeling step as shown in Figure 4.8 (a). By taking into account symmetry and orthogonality, we re-built the approximate model of the car. Figure 4.8 (b) shows the re-generated approximate appearance, geometry, and 3D shape of the car.

4.4 Summary

We introduced two 3D car shape modeling approaches: real-scale car shape recovery and a 3D modeling using the approximate model. In the first approach, we recovered a real-scale car shape recovery using Origami World of a car. This work was achieved by imposing a shape prior of a car and geometric constraints from lidar measurements, and then combining two existing approaches such as the 2D car alignment approaches proposed by Li and the 3D recovery using a single image





(b)

Figure 4.8: Image (a) shows an original approximate model of a car, Image (b) shows a more complete 3D car model reconstructed with shape prior.

proposed by Kanade. In the second approach, we create more complete 3D car shape using the two strong properties: (i) the two sides of a car are symmetric, and (ii) the profile view is orthogonal to the frontal and rear views. This work was achieved by segmenting foreground in the approximate appearance model and estimating symmetry axis on the approximate model.

Chapter 5

Conclusions

This dissertation addressed the approximate representation of unknown objects, detection and tracking of previously seen objects in 3D, and extension of an approximate model by imposing a shape prior. We pervasively combined lidar sensor data and vision data throughout. The contributions of this dissertation are presented in four main categories:

Approximate Representation of Unknown Objects : We developed an algorithm that is approximately representing unknown objects in outdoor environments with lidar measurements and corresponding imagery. We created viewpoint-invariant and scale-normalized models of unknown objects. This work was achieved in three stages: (i) weighted-unwrapped mosaicing of object images with corresponding lidar geometry, (ii) registration of the unwrapped images, and (iii) foreground segmentation. The approximate model consists of the unwrapped mosaics (i.e, color and edge-gradient) and the approximate geometry. Using this object representation, we obtained both the benefits of view-based models and shape-based models; that is, we estimate the 3D pose of an object by 2D-to-2D matching.

Detection and Tracking using an Approximate Model : We designed a detector to distinguish between different objects when we detect an object with the approximate model. We trained the detector using positive and negative samples generated from the unwrapped mosaics in the approximate appearance model. We represented the training model as an probabilistic model to estimate the likelihood of new objects. We evaluated the performance of our detector using datasets obtained in outdoor environments. The detector identifies all of the objects which have their approximate model in detection.

3D Shape Recovery Imposing Shape Prior : We demonstrated recovery of real-scale 3D car shapes using Origami World. In order to recover a car shape, we imposed a shape prior of a car and geometric constraints from lidar measurements. This work is achieved by combining our new ideas with two existing approaches, the 2D car alignment approach for automatic line drawing and the Origami World for 3D shape recovery. We applied a weak-perspective projection assumption for shape recover with lidar measurements instead of orthographic projection. We automatically obtained the line drawings of a car by employing the 2D car alignment approach proposed by Li.

Pervasive Sensor Fusion : In this dissertation, we pervasively combined lidar and vision data. Both types of sensor data are fused at the feature-level by computing features from the data. The benefit of our approach was that we integrate both sensor data throughout the different stages in the process. Such an approach can be robust because both sensors can contribute to the solution at each step in a sequence of operations, whereas decision-level fusion is limited by any one step that performs poorly for a given sensor. We proved the benefits of the feature-level fusion of both sensors in the problem of boundary detection, data association, modeling, detection, and tracking. We estimated the extrinsic parameters between both sensors using a new extrinsic calibration method minimizing the corresponding error of the lidar measurement and vision data.

We developed a time-synchronization method minimizing the time difference between the lidar measurement and vision data.

To summary, the thesis develops a viewpoint-invariant and scale normalized model of unknown objects using pervasive sensor fusion of a lidar sensor and a vision sensor. We have demonstrated that the model has the benefits of view-based models and shape-based models, performing 3D pose estimation by 2D-to-2D matching. The detector using the model shows excellent performance on detecting the objects which have approximate models. The proposed 3D shape recovering method automatically builds explicit car shape models of real scale with a single image. We hope that the obtained results will contribute to solve other problems in computer vision and serve a basis for pursuing the dream of building autonomous vehicles.

Appendix A

Extrinsic Calibration of a Lidar and Camera

In order to convert data between the local coordinate systems, we must estimate the rigid body transformation between the sensors. In this thesis, we developed a robust-weighted extrinsic calibration algorithm that is implemented easily and has small calibration error [71]. The extrinsic calibration parameters are estimated by minimizing the distance between corresponding features projected onto the image plane. The features are edge and centerline features on a v-shaped calibration target (Figure A.1). The proposed algorithm contributes two ways to improve the calibration accuracy. First, we use different weights to distance between a point and a line feature according to the correspondence accuracy of the features. Second, we apply a penalizing function to exclude the influence of outliers in the calibration data sets. We conduct several experiments to evaluate the performance of our extrinsic calibration algorithm, such as comparison of the RMS distance of the ground truth and the projected points, the effect of the number of lidar scan and image, and

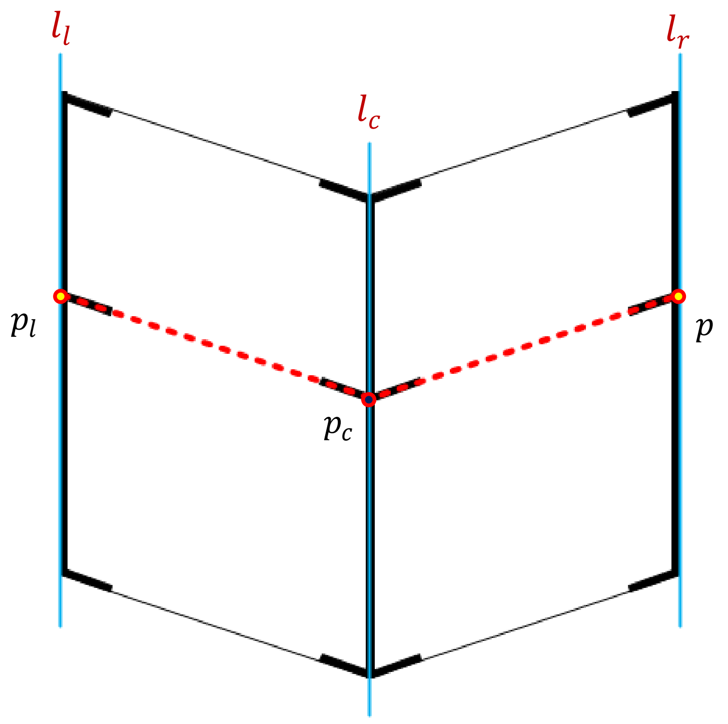


Figure A.1: Calibration board description used for the extrinsic calibration: The blue lines, (left (l_l), center (l_c), and right (l_r)) describe the line feature that we extracted in the image and the three red points (p_l, p_c, p_r) are the point features in 3D.

the effect of pose and range of the calibration target. In the experiments, we show our extrinsic calibration algorithm has calibration accuracy over 50% better than an existing state of the art approach. To evaluate the generality of our algorithm, we also colorize point clouds with different pairs of lidars and cameras calibrated by our algorithm.

A.1 Extrinsic Calibration between Lidar and Vision

Our extrinsic calibration algorithm consists of three steps: data acquisition, feature extraction, and optimization. These steps are described in the following subsections.

A.1.1 Calibration Target and Data Acquisition

The calibration target we designed consists of two planar boards arranged in a v-shape. The angle is 150°, based on the recommendation of Wasielewski [72]. We marked the centerline and left and right boundaries with black tape to enable those edges to be detected in images. We created our target from foamcore boards, which are low cost, light weight, and sufficiently planar. Using this calibration target, we obtain a set of images and lidar scans with the target in different poses and at different ranges. Both planes on the convex side must visible to each sensor. The lidar scan line must intersect both the left and right sides of the target. The lidar and camera must either be synchronized or held stationary during a scan.

A.1.2 Feature Definition and Extraction

We extract three line features (l_l, l_c, l_r) from each image and three point features (p_l, p_c, p_r) from each lidar scan, as shown in Figure A.1. The image-based line features correspond to the left and right edges and the centerline of the target. Currently, we manually select the line features because we found it difficult to automatically extract these lines in long range images. The 3D features extracted from the lidar scan include the left and right depth edges and the centerpoint (Figure A.2). First, we manually select the border points on the target on the left and right sides (p_L^j, p_R^j). Second, we segment the lidar data into two linear segments corresponding to the left and right sides of the target. The segmentation is accomplished using the Iterative-End-Point-Fit (IEPF) algorithm [73]. The IEPF algorithm recursively splits a set of points until a distance related criterion is satisfied. In our case, the algorithm is stopped after two segments are found. Next, we fit lines to each segment using the total least squares method. Finally, the intersection of those lines is defined as the centerpoint.

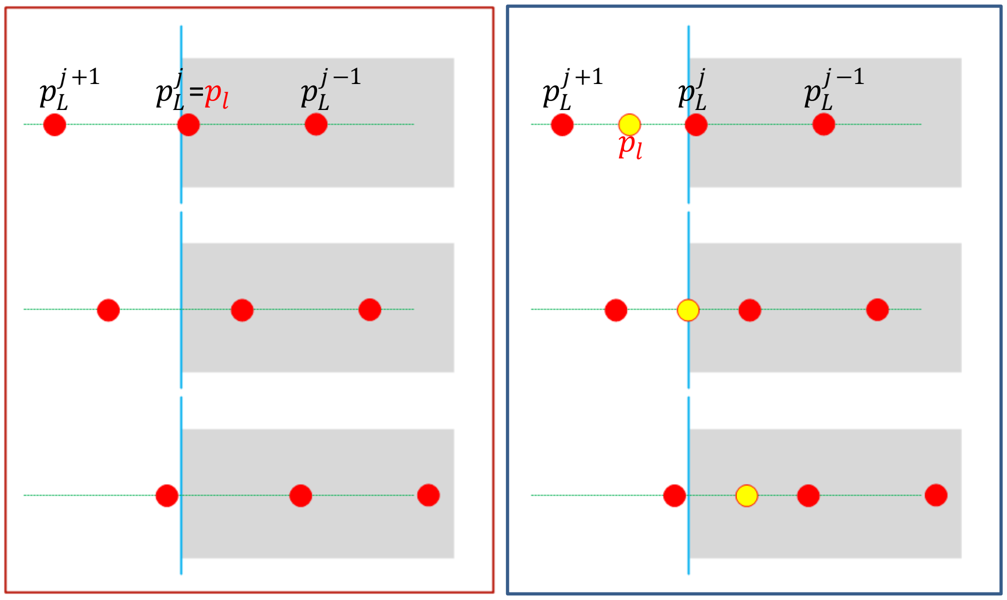
Li proposed using edge points as 3D features directly [74]. However, since the lidar can be sparse, the features may be far from the true edge of the target, especially at long distances. Therefore, we propose using virtual feature points placed at the expected value of the true edge locations, as shown in Fig. A.2 (a). If we assume that the true edges are uniformly distributed between the last point (p_L^j) and the next consecutive point (p_L^{j+1}), the expected value is the mid-point of the two points. The feature (p_l) is computed by selecting the intersection point of the angular mid-point and the fitted line as shown in Figure A.2 (b). The feature (p_r) is computed analogously.

A.1.3 Optimization

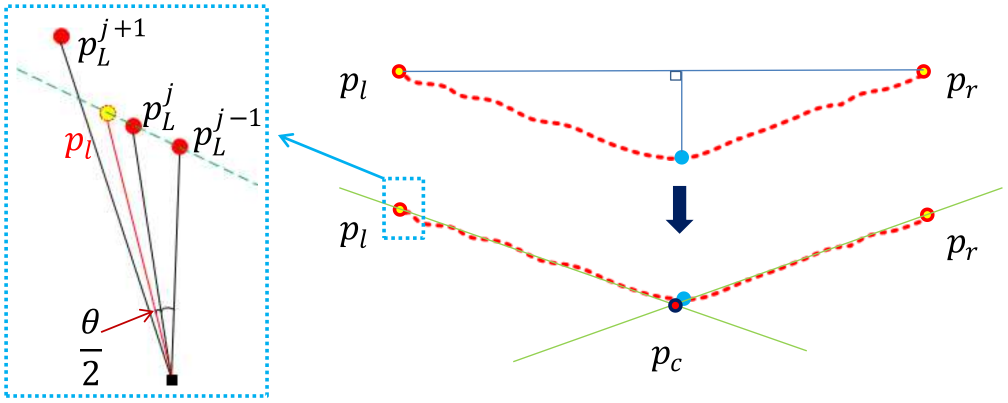
The extrinsic parameters are estimated by minimizing the distance between the line features and point features in 2D. Given a set of N paired images and lidar scans with the target in different poses and at different ranges, we project the point features (p_l, p_c, p_r) in 3D onto the images. In the image plane, we measure the normal distance between each line feature (l_l, l_c, l_r) and corresponding projected point feature ($\widetilde{p}_l$, $\widetilde{p}_c$, $\widetilde{p}_r$). We obtain three distance values per image (d_l, d_c, d_r), one each on the left and right borders and from the centerline. The error function $E(\mathbf{R},\ \mathbf{t})$ is defined as the squared sum of those distances over the N image and lidar pairs:

$$
\begin{gathered}
E(\mathbf{R},\ \mathbf{t}) = \sum_{i=1}^{N} (d_l^i)^2 + (d_c^i)^2 + (d_r^i)^2, \\
d_k^i = dist(\widetilde{p}_k^i,\ l_k^i), \quad k = \{left,\ center,\ right\}.
\end{gathered}
\tag{A.1}
$$

where the function *dist* is the minimum distance from point ($\widetilde{p}_k$) to line (l_k). We estimate the extrinsic parameters by minimizing (A.1) using the Levenberg-Marquardt (LM) method [75].



(a) The left boundary point (p_L^j) is never exactly placed on the left edge line of the calibration board. Instead, we select a virtual point (p_l) between the two consecutive points (p_L^j, p_L^{j+1}), located at the expected value of the edge line position (yellow dot). In the figure, the gray plane represents the calibration target and the blue line describes the left edge line of the calibration target.



(b) The virtual point (p_l) is selected as a point that is placed on the intersection position of the angular mid-point and the line fitted to the left side lidar points. θ is the angular resolution of the lidar. The centerpoint (p_c) is selected as the intersection of the left and right fitted lines.

Figure A.2: Lidar feature definition and extraction.

A.2 Extensions to the basic approach

We observed two characteristics in the basic approach: some points are more reliable than the others, and line features can be noisy due to manual extraction. Therefore, we extended the basic approach in two ways. First we applied different weights to the distance error for the center features versus the edge features because the center feature is more accurate than the others. Second, we incorporated a robust optimization approach to handle the effect of outliers in data sets.

A.2.1 Feature Weighting

Intuitively, the corresponding accuracy of center feature is higher than the others because the point is selected as the intersection of the left and right fitted lines. We apply more weight to the distance of center feature. The weighting values are computed by the following steps. First, we measure the average residuals (e_l, e_c, e_r) of each position from Equation (A.1). The average residuals are computed as $e_l = \sum_i^N (d_l^i)^2/N$ where N is number of image and lidar scan pairs. The center and right residuals (e_c and e_r) are expressed analogously. Second, we compute the weights (w_l, w_c, w_r) as $w_l = 1/e_l$. Each weight value describes the confidence of the distance measure. We also show that the correspondence of the center feature is more accurate through experiments in Section A.3. Once we have determined the weights, we re-estimate the extrinsic parameters by minimizing the following error function with the LM algorithm:

$$E_W(\mathbf{R},\ \mathbf{t}) = \sum_{i=1}^{N} w_l \cdot (d_l^i)^2 + w_c \cdot (d_c^i)^2 + w_r \cdot (d_r^i)^2 \tag{A.2}$$

A.2.2 Robust Optimization

Since we extract the corresponding line and point features manually, noise is likely to be included in image and lidar pairs. We extend our approach to be robust to the effect of outliers. Small errors can be managed by increasing the size of the data set, but outliers in the data set cause the accuracy of the extrinsic calibration to decrease. Therefore, we apply the concept of nonlinear robust regression to exclude the influence of these outliers [76]. If there exists a large pixel distance error between the selected edge line and the corresponding lidar point, the algorithm gives the penalty measured by the Huber penalty function. Otherwise, it returns the distance value. The equation is represented as follows [46]:

$$E_R(\mathbf{R},\ \mathbf{t}) = \sum_{i=1}^{n} w_l \cdot \phi_{huber}(d_l^i) + w_c \cdot \phi_{huber}(d_c^i) + w_r \cdot \phi_{huber}(d_r^i) \tag{A.3}$$

where $\phi_{huber}(\cdot)$ is the Huber penalty function:

$$\phi_{huber}(d) = \begin{cases} d^2 & \text{if } |d| \leq d_{max} \\ d_{max}(2 \cdot |d| - d_{max}) & \text{if } |d| > d_{max} \end{cases} . \tag{A.4}$$

Since we use the virtual points instead of the real border points, the maximum distance on image is bounded as follows:

$$d_l,\ d_r \lesssim d_{max} = f \tan\left(\frac{\theta}{2}\right) \tag{A.5}$$

where f is the focal length of the camera and θ is the angular resolution between successive lidar points. We use this bound to define d_{max}.

A.3 Experimental Results

In order to illustrate and evaluate our algorithm, we used three pairs of cameras and lidars. One pair consists of a SICK LMS-221 lidar and a PointGrey Flea2 camera. The other pairs are composed of a SICK LMS-291 lidar and a PointGrey Flea2 camera. The detailed specifications of the sensors are the same. The lidars have a 180° horizontal field of view, with a line scanning frequency of 37.5 Hz and a 0.5° angular resolution. The cameras have a 60° horizontal field of view, with a frame rate of 15 Hz and a resolution of 1024 by 768 pixels. The data from each sensor is time-stamped to enable synchronization. The intrinsic parameters of the cameras (e.g., focal length, optical center, pixel aspect ratio, and skew) were computed using the Matlab Camera Calibration Toolbox [77]. The images used for the experiment were rectified by the intrinsic parameters prior to feature extraction.

We analyzed the performance of our algorithm and compared to the state of the art using the ground truth image/scan pairs. In the first experiment, we evaluated the performance of our baseline algorithm and the two extensions – the weighting method and the robust optimization. We then compared our algorithm to two state-of-the-art previously published algorithms. In the third experiments, we evaluated the effect of the number of scan/image pairs. Finally, to evaluate the repeatability of our extrinsic calibration algorithm, we colorized point clouds using the resulting calibration and visually verified the results.

A.3.1 Comparison of Base Line and Extension Approaches

We compared the performance of our baseline approach and the two extensions described in Section A.2. Using a set of 250 image/scan pairs, we randomly selected subsets of 50, 100, and 150

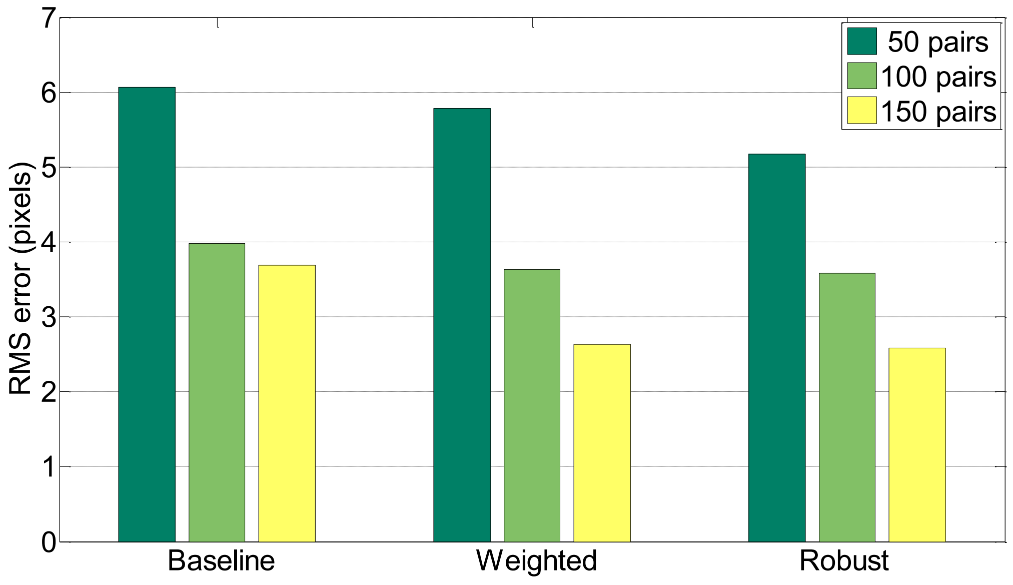


Figure A.3: Comparison of our baseline approach with the weighted and robust extensions. The results were computed as a function of number of image/scan pairs evaluated using 10-fold cross validation. The errors were estimated by measuring the RMS distance with the ground truth.

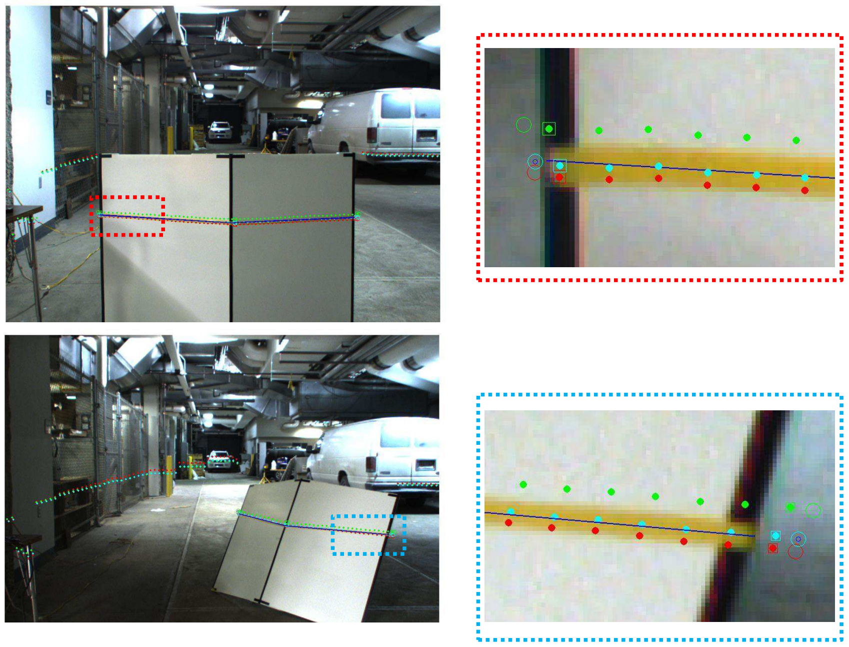
image/scan pairs and computed the extrinsic parameters using the features extracted from each subset. The experiment was repeated multiple times using 10-fold cross-validation and using the same picked points for each image/scan pair. The results were evaluated in terms of the line alignment error averaged over the trials within each group. Figure A.3 shows the comparison of our baseline, weighted, and robust approaches. Within each group, the robust approach performed the best. For example, when we used 150 pairs, the robust approach performed 20% better than the baseline approach and 5% better than the weighted approach. Therefore, we used the robust approach to evaluate the performance of our calibration algorithm in the following experiments.

A.3.2 Comparison with Previous Methods

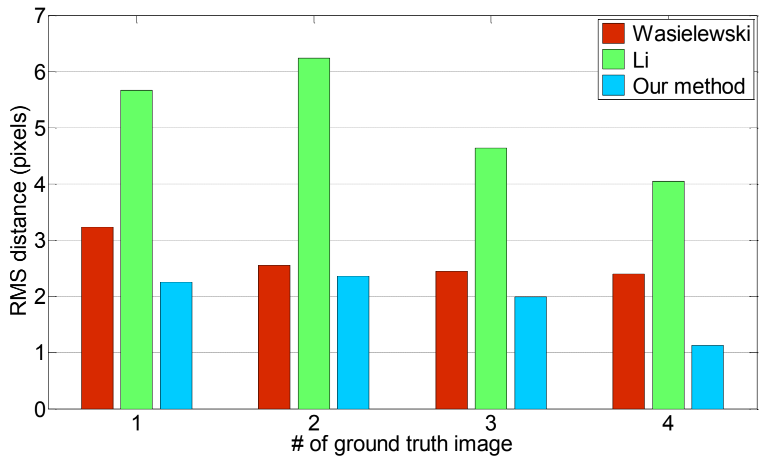
We compared our method to Li's [74] and Wasielewski's [72] algorithms. We used the same 250 image/scan pairs as in the first experiment. Figure A.4 (a) shows the lidar points projected onto a test image by the extrinsic parameters acquired from each algorithm. The projected lidar points for our algorithm are located closer to the ground truth line than the projected points of the other methods. The performance of each algorithm is objectively evaluated by computing the line alignment error, as shown in Figure A.4 (b). The average error of our method is 1.87 pixels. It is smaller than Wasielewski (2.7 pixels) and Li (5.5 pixels). In the case of Wasielewski and our method, the correspondence error of the centerpoint is smaller than both the side points which confirms our intuition that led to the weighted approach.

A.3.3 Effect of the Number of Scan/Image Pairs

We evaluated the line alignment accuracy as as function of number of image/scan pairs. This evaluation was done using 10-fold cross validation. The number of testing image/scan pairs were randomly selected from the 250 pairs. As shown in Figure A.5, our result improves when using more image/scan pairs and achieves better performance with less data. The average error of our method is 4.4 pixels. This error value is 2 times lower than Wasielewski's 8.5 pixels and 2.2 times lower than Li's 9.6 pixels. When using 50 pairs, our approach has 43% better performance than Wasielewski's algorithm using 100 pairs and 75% better than Li's method with 100 pairs. This result means that our approach performs as well as the previous approaches with half of the data. It is interesting to note that the calibration accuracy of Li's approach is not improved by increasing the number of the pairs.



(a) Projection of the lidar points onto the two ground truth images: The yellow band is the marked tape. The blue line is the mid-line of the marked tape that manually selected. The red and cyan boxes show close-up of the projection of the lidar points. Red points = Wasielewski, green points = Li, cyan = our method (robust).



(b) Line alignment error between the ground truth line and projected lidar points.

Figure A.4: The comparison of the extrinsic calibration algorithms which proposed by Walsielewski, Li, and us. To do this test, we used the four ground truth images we made.

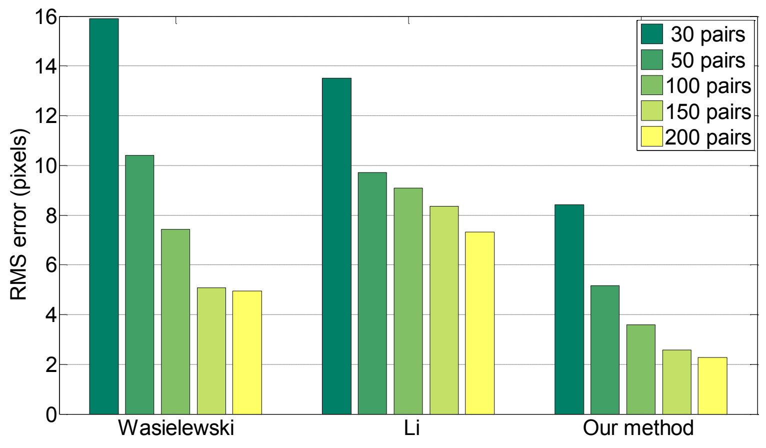


Figure A.5: Performance as a function of number of image and lidar scan pairs. We used a 10-fold cross validation to evaluate the effect that the number of calibration data give to the calibration accuracy. Our approach was also affected by the number of the calibration data, but the effect was smaller than other approaches).

A.3.4 Point Cloud Colorizing

We evaluated reliability and repeatability of our algorithm. This experiment was achieved with two pairs of lidars and cameras on the roof of a vehicle looking diagonally at 45° with respect to the vehicle. The lidar were mounted with a 90° roll rotation so vertical cuts of the environment are obtained. As the vehicle drives, lidar and image data are acquired, and the 3D points are accumulated in a common reference frame using the inertial sensors of the vehicle. Since the cameras and lidars were calibrated with our method, the image projection of each 3D lidar point is known. The color of each 3D point is then extracted and used to perform a photo-realistic 3D reconstruction of the environment. Figure A.6 shows examples of the 3D reconstruction obtained. A detailed visual inspection of Figure A.6 reveals the accuracy of our calibration algorithm (observe the lighting post and windows of the building).

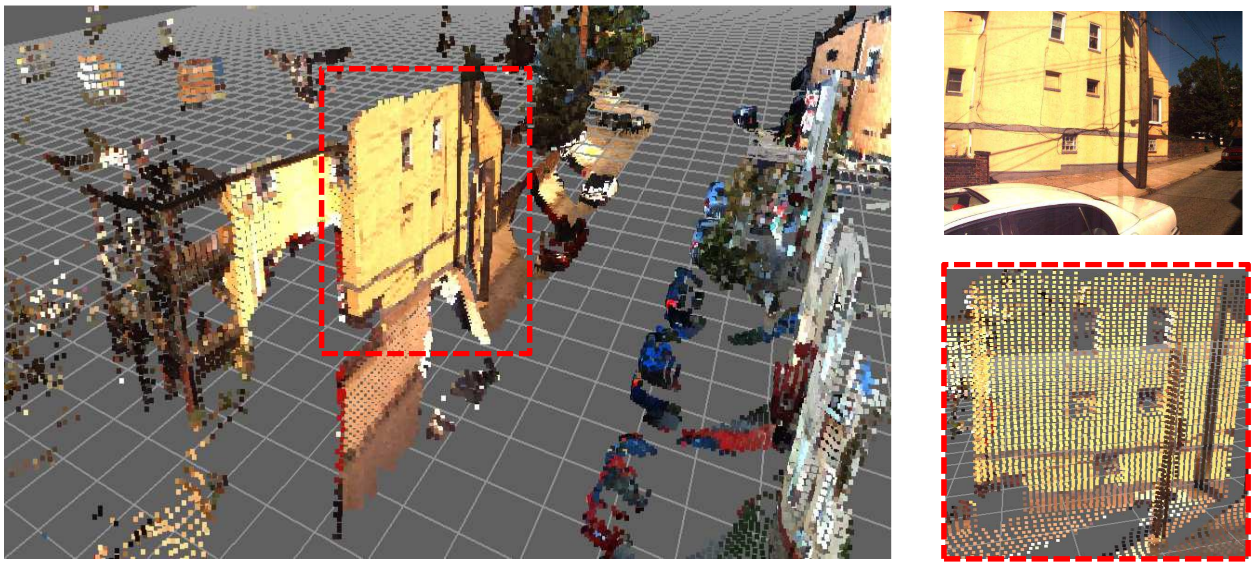


Figure A.6: Colorized point clouds of an outdoor street scene. Left: Overview of a section of the scene. Upper right: Corresponding camera image. Lower right: A close up of the region indicated by the red box in the left sub-figure. Notice that the a edges of the window openings are well-aligned, indicating that the calibration accuracy is good.

A.4 Summary

We developed an extrinsic calibration algorithm compensating for the drawbacks of existing approaches and providing highly accurate calibration performance. We conducted several experiments to evaluate the performance and to compare to existing state-of-the-art algorithms. The main findings of our experiments are:

- The alignment accuracy of our method is two times better than the compared state of the art algorithms.

- Our method requires fewer image/scan pairs to achieve the same calibration accuracy than the previous methods.
- Our method has robust performance regardless of pose and range of the calibration target.

Bibliography

[1] M. Riesenhuber and T. Poggio, "Models of object recognition," *Nat Neurosci*, vol. 3, pp. 1199–1204, 2000.

[2] D. Koller, K. Danilidis, and H.-H. Nagel, "Model-based object tracking in monocular image sequences of road traffic scenes," *International Journal of Computer Vision*, vol. 10, pp. 257–281, 1993.

[3] A. Jepson, D. Fleet, and T. El-Maraghi, "Robust online appearance models for visual tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 10, pp. 1296 – 1311, 2003.

[4] M. Luber, K. O. Arras, C. Plagemann, and W. Burgard, "Classifying dynamic objects," *Autonomous Robots*, vol. 26, pp. 141–151, 2009.

[5] S. Palmer, *Vision Science: Photons to Phenomenology.* MIT Press, 1999.

[6] J. L. Mundy, "Object recognition in the geometric era: A retrospective," in *Toward Category-Level Object Recognition*, 2006, pp. 3–28.

[7] B. K. P. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, pp. 185–203, 1981.

[8] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.

[9] C. Veenman, M. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, no. 1, pp. 54 –72, 2001.

[10] K. Shafique and M. Shah, "A noniterative greedy algorithm for multiframe point correspondence," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 27, no. 1, pp. 51 –65, 2005.

[11] A. Ess, K. Schindler, B. Leibe, and L. Van Gool, "Object detection and tracking for autonomous navigation in dynamic environments," *International Journal of Robotics Research*, vol. 29, pp. 1707–1725, 2010.

[12] B. Leibe, K. Schindler, N. Cornelis, and L. Van Gool, "Coupled object detection and tracking from static cameras and moving vehicles," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 30, pp. 1683–1698, 2008.

[13] R. MacLachlan, "Tracking moving objects from a moving vehicle using a laser scanner," Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-07, 2005.

[14] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, pp. 123–139, 2009.

[15] C. Premebida, O. Ludwig, and U. Nunes, "Lidar and vision-based pedestrian detection system," *Journal of Field Robotics*, vol. 26, pp. 696–711, 2009.

[16] V. Lepetit and P. Fua, "Monocular model-based 3d tracking of rigid objects," *Foundations and Trends in Computer Graphics and Vision*, vol. 1, pp. 1–89, 2005.

[17] P. Moreels and P. Perona, "Evaluation of features detectors and descriptors based on 3d objects," *International Journal of Computer Vision*, vol. 73, pp. 263–284, 2007.

[18] M. Haag and H.-H. Nagel, "Combination of edge element and optical flow estimates for 3d-model-based vehicle tracking in traffic image sequences," *International Journal of Computer Vision*, vol. 35, pp. 295–319, 1999.

[19] A. Ottlik and H. H. Nagel, "Initialization of model-based vehicle tracking in video sequences of inner-city intersections," *International Journal of Computer Vision*, vol. 80, pp. 211–225, 2008.

[20] J. Lou, T. Tan, W. Hu, H. Yang, and S. J. Maybank, "3-d model-based vehicle tracking," *IEEE Transactions on Image Processing*, vol. 14, pp. 1561–1569, 2005.

[21] S. J. Dickinson, "Object representation and recognition," in *What is Cognitive Science.* Basil Blackwell Publishers, 1999, pp. 172–207.

[22] H. Schneiderman and T. Kanade, "A statistical method for 3d object detection applied to faces and cars," in *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on*, 2000, pp. 1746–1759.

[23] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, "3d object modeling and recognition using affine-invariant patches and multi-view spatial constraints," in *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on*, vol. 2, 2003, pp. II – 272–7 vol.2.

[24] P. Yan, S. M. Khan, and M. Shah, "3d model based object class detection in an arbitrary view," in *Computer Vision. IEEE International Conference on*, vol. 0, 2007, pp. 1–6.

[25] Y. Sato, M. D. Wheeler, and K. Ikeuchi, "Object shape and reflectance modeling from observation," in *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997, pp. 379–387.

[26] N. Dowson and R. Bowden, "Simultaneous modeling and tracking (smat) of feature sets," in *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on*, vol. 2, 2005, pp. 99 – 105 vol. 2.

[27] Z. Yin and R. Collins, "On-the-fly object modeling while tracking," in *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on*, 2007, pp. 1 –8.

[28] Y. Li, L. Gu, and T. Kanade, "A robust shape model for multi-view car alignment," in *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on*, 2009.

[29] T. Kanade, "Recovery of the three-dimensional shape of an object from a single view," *Artificial Intelligence*, vol. 17, no. 1-3, pp. 409 – 460, 1981.

[30] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, pp. 13+, 2006.

[31] K. Cannons, "A review of visual tracking," York University, Tech. Rep., 2008.

[32] G. A. Borges and M.-J. Aldon, "Line extraction in 2d range images for mobile robotics," *Journal of Intelligent & Robotic Systems*, vol. 40, pp. 267–297, 2004.

[33] C. Premebida and U. Nunes, "Segmentation and geometric primitives extraction from 2d laser range data for mobile robot applications," *Robotica*, pp. 17–25, 2005.

[34] K. Arras, O. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2D rangedata," in *IEEE International Conference on Robotics and Automation*, 2007, pp. 3402–3407.

[35] L. Spinello and R. Siegwart, "Human detection using multimodal and multidimensional features," in *IEEE International Conference on Robotics and Automation*, 2008, pp. 3264–3269.

[36] J. Crowley, P. Stelmaszyk, and C. Discours, "Measuring image flow by tracking edge-lines," in *Computer Vision. IEEE International Conference on*, 1988, pp. 658 –664.

[37] I. J. Cox, "A review of statistical data association techniques for motion correspondence," *International Journal of Computer Vision*, vol. 10, pp. 53–66, 1993.

[38] S. Blackman, "Multiple hypothesis tracking for multiple target tracking," *Aerospace and Electronic Systems Magazine, IEEE*, vol. 19, no. 1, pp. 5 –18, 2004.

[39] Y. Bar-Shalom, *Tracking and data association.* Academic Press Professional, Inc., 1987.

[40] D. Reid, "An algorithm for tracking multiple targets," *Automatic Control, IEEE Transactions on*, vol. 24, no. 6, pp. 843 – 854, 1979.

[41] V. N. Vapnik, *The nature of statistical learning theory.* Springer-Verlag New York, Inc., 1995.

[42] O. Al-Kofahi, R. J. Radke, R. K. Goderie, Q. Shen, S. Temple, and B. Roysam, "Automated Cell Lineage Construction: A Rapid Method to Analyze Clonal Development Established with Murine Neural Progenitor Cells," *Cell Cycle*, vol. 5, no. 3, pp. 327–335, 2006.

[43] K. Li, M. Chen, and T. Kanade, "Cell population tracking and lineage construction with spatiotemporal context," in *Proceedings of the 10th International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2007, pp. 295 – 302.

[44] N. A. Thacker, F. J. Aherne, and P. I. Rockett, “The bhattacharyya metric as an absolute similarity measure for frequency coded data,” *Kybernetika*, vol. 34, pp. 363–368, 1997.

[45] D. Comaniciu, V. Ramesh, and P. Meer, “Kernel-based object tracking,” *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564–575, 2003.

[46] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[47] K. Kwak, D. Huber, J. Chae, and T. Kanade, “Boundary detection based on supervised learning,” in *IEEE International Conference on Robotics and Automation*, 2010.

[48] C. Chang and C. Lin, “LIBSVM: A library for support vector machines,” *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 3, pp. 1–27, 2011.

[49] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[50] A. Rav-Acha, P. Kohli, C. Rother, and A. Fitzgibbon, “Unwrap mosaics: A new representation for video editing,” in *Proceedings of ACM SIGGRAPH.* ACM, 2008.

[51] A. Pentland and S. Sclaroff, “Modal representations,” in *Proceedings of the International NSF-ARPA Workshop on Object Representation in Computer Vision.* Springer-Verlag, 1995, pp. 249–262.

[52] F. Rothganger, S. Lazebnik, C. Schmid, and J. Ponce, “3d object modeling and recognition using local affine-invariant image descriptors and multi-view spatial constraints,” *International Journal of Computer Vision*, vol. 66, no. 3, pp. 231–259, 2006.

[53] D. Lowe, “Local feature view clustering for 3d object recognition,” in *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on*, vol. 1, 2001, pp. 682–688.

[54] S. Hinterstoisser, C. Cagniart, S. Ilic, P. Sturm, N. Navab, P. Fua, and V. Lepetit, "Gradient response maps for real-time detection of textureless objects," *IEEE Transactions Pattern Analysis and Machine Intelligence*, vol. 34, no. 5, pp. 876–888, 2012.

[55] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[56] V. Nguyen, S. Gächter, A. Martinelli, N. Tomatis, and R. Siegwart, "A comparison of line extraction algorithms using 2d range data for indoor mobile robotics," *Autonomous Robots*, vol. 23, no. 2, pp. 97–111, 2007.

[57] R. Szeliski, *Computer Vision : Algorithms and Applications*, R. Szeliski, Ed. Springer-Verlag New York Inc, 2010.

[58] V. S. Lempitsky and D. V. Ivanov, "Seamless mosaicing of image-based texture maps," in *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on.* IEEE Computer Society, 2007.

[59] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys, "Interactive 3d architectural modeling from unordered photo collections," *ACM Trans. Graph.*, vol. 27, no. 5, pp. 159:1–159:10, 2008.

[60] Y. Y. Boykov and M. P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images," *Proceedings Eighth IEEE International Conference on Computer Vision ICCV 2001*, vol. 1, pp. 105 – 112, 2001.

[61] D. Scharstein, "Matching images by comparing their gradient fields," in *International Conference on Pattern Recognition*, 1994, pp. 572–575.

[62] V. Franc and V. Hlavac, "Statistical Pattern Recognition Toolbox for Matlab," *Prague, Czech: Center for Machine Perception, Czech Technical University*, 2004.

[63] Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119 – 152, 1994.

[64] A. E. Johnson and S. B. Kang, "Registration and integration of textured 3-d data," in *Proceedings of the International Conference on Recent Advances in 3-D Digital Imaging and Modeling.* IEEE Computer Society, 1997, pp. 234–.

[65] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 1, 2001, p. 511.

[66] T. Kanade, "A theory of origami world," *Artificial Intelligence*, vol. 13, pp. 279–311, 1980.

[67] D. L. Waltz, "Generating semantic descriptions from drawings of scenes with shadows," MIT, Tech. Rep., 1972.

[68] A. Macworth, "Interpreting pictures of polyhedral scenes," *Artificial Intelligence*, vol. 4, no. 2, pp. 121 – 137, 1973.

[69] C. M. Bishop, *Pattern Recognition and Machine Learning.* Springer-Verlag New York, Inc., 2006.

[70] J. Sun, W. Zhang, X. Tang, and H. yeung Shum, "Background cut," in *European Conference on Computer Vision*, 2006, pp. 628–641.

[71] K. Kwak, D. Huber, H. Badino, and T. Kanade, "Extrinsic calibration of a single line scanning lidar and a camera," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[72] S. Wasielewski and O. Strauss, "Calibration of a multi-sensor system laser rangefinder/camera," in *Intelligent Vehicles Symposium*, 1995, pp. 472–477.

[73] R. Duda and P. Hart, *Pattern Classification and Scene Analysis.* John Wiley and Sons Inc, 1973.

[74] G. Li, Y. Liu, L. Dong, X. Cai, and D. Zhou, "An algorithm for extrinsic parameters calibration of a camera and a laser range finder using line features," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3854–3859.

[75] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of Applied Mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[76] P. J. Huber, "Robust estimation of a location parameter," *The Annals of Mathematical Statistics*, vol. 35, no. 1, pp. 73–101, 1964.

[77] J. Bouguet, "Camera calibration toolbox for matlab," 2003.