Probabilistic Online Learning Of Appearance And Structure For Robotics

Bhoram Lee
*University of Pennsylvania*, bhoram.lee@gmail.com

Follow this and additional works at: https://repository.upenn.edu/edissertations

Part of the Robotics Commons

Recommended Citation
https://repository.upenn.edu/edissertations/3322

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/edissertations/3322
For more information, please contact repository@pobox.upenn.edu.
Probabilistic Online Learning Of Appearance And Structure For Robotics

Abstract
A robotic system can be characterized by its interactions with environments. With growing demand for robots deployed in various scenarios, the ability to perform physical interaction in uncontrolled environments has become of great interest. While a robot performs interactive tasks, its visual and spatial sensing plays a critical role. Being a major source of learning, vision not only guides immediate actions, but also indirectly improves future actions and decisions. How visual information is gathered and represented will significantly influence how a robot can plan and act. Although recent advances in machine perception have presented unprecedented performance in some areas, there still exist challenges in various aspects. In this dissertation, I will address two such issues and suggest an online probabilistic approach to each problem.

Most successful approaches in visual learning depend on fragments of exemplars prepared by humans. It is simply unaffordable to provide constant human supervision to a robotic system that would receive tens of new image frames per second. Ideally, a robotic system is required to gather information from its unique experience and keep growing knowledge on the fly without such external aids. One way to implement the self-learning is to take advantage of the naturally correlated sensations of different sensory modalities. The first part of this talk presents a probabilistic online self-learning framework to alleviate the dependency in robotic visual learning by leveraging structural priors.

Another challenge in robotics is its spatial understanding. Aside from planning and performing actions, spatial representation itself still largely requires more research. While point or grid-based representations are currently being employed for practical conveniences, these methods suffer from discretization and disconnected spatial information. On the other hand, Gaussian Processes (GP) have recently gained attention as an alternative to represent the distance field of structures continuously and probabilistically. It is not only the seamless expression of structures, but also direct access to the distance and direction to obstacles that make the representation invaluable. The second part of the talk presents an online framework for continuous spatial mapping using GP.

Degree Type
Dissertation

Degree Name
Doctor of Philosophy (PhD)

Graduate Group
Electrical & Systems Engineering

First Advisor
Daniel Lee

Keywords
Mapping, Vision

Subject Categories
Robotics

This dissertation is available at ScholarlyCommons: https://repository.upenn.edu/edissertations/3322
PROBABILISTIC ONLINE LEARNING OF APPEARANCE AND STRUCTURE
FOR ROBOTICS
Bhoram Lee
A DISSERTATION
in
Electrical and Systems Engineering
Presented to the Faculties of the University of Pennsylvania
in
Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
2019

Daniel D. Lee, Supervisor of Dissertation
Professor

Victor M. Preciado, Graduate Group Chairperson
Professor

Dissertation Committee

Kostas Daniilidis, Professor of Computer and Information Science
Daniel Lee, Professor of Electrical and Systems Engineering
Rene Vidal, Professor of Biomedical Engineering, Johns Hopkins University
Renaud Detry, Research Scientist, NASA/Caltech Jet Propulsion Laboratory
PROBABILISTIC ONLINE LEARNING OF APPEARANCE AND STRUCTURE
FOR ROBOTICS
© COPYRIGHT
2019
Bhoram Lee
ABSTRACT

PROBABILISTIC ONLINE LEARNING OF APPEARANCE AND STRUCTURE
FOR ROBOTICS

Bhoram Lee
Daniel D. Lee

A robotic system can be characterized by its interactions with environments. With growing demand for robots deployed in various scenarios, the ability to perform physical interaction in uncontrolled environments has become of great interest. While a robot performs interactive tasks, its visual and spatial sensing plays a critical role. Being a major source of learning, vision not only guides immediate actions, but also indirectly improves future actions and decisions. How visual information is gathered and represented will significantly influence how a robot can plan and act. Although recent advances in machine perception have presented unprecedented performance in some areas, there still exist challenges in various aspects. In this dissertation, I will address two such issues and suggest an online probabilistic approach to each problem.

Most successful approaches in visual learning depend on fragments of exemplars prepared by humans. It is simply unaffordable to provide constant human supervision to a robotic system that would receive tens of new image frames per second. Ideally, a robotic system is required to gather information from its unique experience and keep growing knowledge on the fly without such external aids. One way to implement the self-learning is to take advantage of the naturally correlated sensations of different sensory modalities. The first part of this talk presents a probabilistic online self-learning framework to alleviate the dependency in robotic visual learning by leveraging structural priors.

Another challenge in robotics is its spatial understanding. Aside from planning and performing actions, spatial representation itself still largely requires more research. While point or grid-based representations are currently being employed for practical conveniences, these methods suffer from discretization and disconnected spatial information. On the other hand, Gaussian Processes (GP) have recently gained attention as an alternative to represent
the distance field of structures continuously and probabilistically. It is not only the seamless expression of structures, but also direct access to the distance and direction to obstacles that make the representation invaluable. The second part of the talk presents an online framework for continuous spatial mapping using GP.
# Contents

Abstract iii  
Contents v  
List of Tables vii  
List of Figures viii  
1 Introduction 1  

I SELF-SUPERVISED APPEARANCE LEARNING 4  
2 Definition 6  

3 Related Work 10  
3.1 Ground Recognition 10  
3.1.1 Terrain Traversability in Wild Environment 11  
3.1.2 Road Detection in Urban Settings 12  
3.1.3 Ground in Monocular Vision 13  
3.2 Object Pose Recognition in 3D 15  
3.2.1 Pose Estimation 15  
3.2.2 Refined Pose Estimation 18  
3.2.3 3D Pose Annotation 18  

4 Online Ground Segmentation and Estimation 21  
4.1 Contributions 22  
4.2 Models 22  
4.2.1 Ground Vehicle Motion and Image Kinematics 23  
4.2.2 Ground Surface Model 25  
4.2.3 Ground Color Classifier 25  
4.3 Online Self-Supervision Framework 26  
4.3.1 Motion and Inverse Depth Estimation 26  
4.3.2 Ground Surface and Scale Estimation 28  
4.3.3 Ground Classification 29  
4.4 Results 29
List of Tables

5.1 RMSE of Estimates on Choi’s Dataset [31] ................................. 45
5.2 Success Rate on Krull’s Dataset[94] .............................. 46
## List of Figures

1.1 Team THOR’s robot at DARPA Robotics Challenge [118] performing the "valve" task: visual and spatial perception plays a critical role for interactive tasks as well as navigation in an autonomous robotic system. .......... 2

2.1 Illustration of (a) Offline Batch Learning, (b) Online Learning with Convergence, and (c) Online Learning for Non-Stationary Concept. ............. 7

2.2 The natural correlation of multi-modal sensations is a great source of self-learning. ................................................................. 8

3.1 Self-Supervised Learning by Multi-Modal Sensors for Terrain Traversability in Wild Environment. .............................................. 11

3.2 Drivable Road Detection in Urban Environment: Since this task requires understanding traffic rules and other objects, existing road or lane maps can be valuable resources for learning. ................................. 12

3.3 Ground as a Reference Object in Monocular Vision: Finding ground can help to recover the scale of the scene. ................................. 14

3.4 Various Pose Estimation Approaches ......................................... 16

3.5 Image Labeling with 3D Pose Annotation .................................... 19

4.1 Finding Ground via Combining (Left) Geometric Estimation and (Right) Appearance-Based Segmentation on Image Plane .................... 21

4.2 Coordinate Frames and Motion Parameters .................................. 23

4.3 Illustration of Parametric Ground Models: in order of simplicity, (a) flat plane with no tilt, (b) general plane possibly with tilt, and (c) curved surface up to the 2<sup>nd</sup> order. ......................................................... 25

4.4 Overview of Online Ground Estimation and Segmentation ................ 27

4.5 An Example of Uniform vs. Biased Weights of Points in the RANSAC Domain of the 2-parameter Model with Real Data. ....................... 29

4.6 Learning Ground Feature (Images from KITTI Benchmark Odometry Dataset [55] No.03): The classifier updates the histogram over the color labels and predicts the probability that each pixel is from the ground based on the learned feature. ......................................................... 30

4.7 Success Rate ............................................................................. 31

4.8 Velocity (Scale) Error ............................................................. 32

4.9 Translation Errors of Sequences No.00-10 Compared to VISO2-M ........ 33
7.5 Illustration of Surface Point and Normal Inference (Sec. 7.3.3): After (a) The range regression and (b) occupancy test converted to Cartesian coordinates, (c) point location and (d) surface normal can be inferred. .......................... 71
7.6 Illustration of ICP without Closest Point Matching on SDF ........................................ 72
7.7 Data Structure Implementation in QuadTree: The clusters for local GPs are centered at the level $l_c$, being a distance $r_{lc}$ apart from each other. The radius of each cluster is larger than $r_{lc}$ to allow proper overlapping areas between clusters. .................................................................................. 74
7.8 The Update Process ................................................................................................. 75
7.9 Visual Comparison of Surface Modeling in 2D ..................................................... 77
7.10 Surface Prediction Accuracy vs. Reduced Data Rate. The left image shows the reduced data rate of 1m interval and the colored patches on the graphs represent minimum/maximum errors for each case. ..................................................... 79
7.11 Hausdorff Distance Analysis: In the left column, Blue dotted lines are the ground true surface, and the red points are surface prediction by each method. The right columns show the close-up look at where the maximum distance occurs: corners for GPIS-SDF (c), and the walls with fewer measurements for other discrete methods (a) and (b). .............................................................. 80
7.12 Computing Time: (a) Update time distribution for mapping Intel Lab (Fig. 7.19 Top-left). (b) Median update time for a large map growing in 2D. Median values are reported since the distribution is not normal. ......................... 81
7.13 Simulation Data used for Localization Evaluation ................................................... 82
7.14 Pre-Computed Likelihood Field on the OG Map (Used for ‘PF’) .............................. 83
7.15 Ground Truth Discrete SDF with Extracted Surfaces (Used for ‘ICP-D’) ............... 83
7.16 GPIS map with Extracted Surfaces (Used for ‘ICP-G’) ............................................. 83
7.17 Localization Error Comparison .............................................................................. 85
7.18 Localization Error Close-Up Look of ICP-G in Fig. 7.17 ...................................... 85
7.19 Mapping Results on Real 2D Data: (Left) Intel Lab and (Right) Freiburg Building 079 from [9] (Best viewed in color) ...................................................... 87
7.20 Surface Modeling Results on Real 3D Data: Daily Objects from YCB [25] and BigBIRD [167] datasets. The SDF is visualized in the 'jet' colormap with an arbitrary scale. ................................................................. 89
7.21 Sampled Surface Points (black) vs. Point Cloud from Poisson Mesh of the Datasets (blue). ...................................................................................... 90
A robotic system can be characterized by its interactions with the world. With growing demand for robots deployed in various scenarios, the ability to perform physical interaction in uncontrolled environments has become of great interest [3][1][8]. While a robot performs interactive tasks, its visual and spatial sensing plays a critical role. This is not surprising. The importance of visual modality in the physical world is evident from the evolutionary history of the animal kingdom: vision is a principal sensory modality for 96% of known species [97]. Being a major source of perception, vision not only guides immediate actions, but also indirectly improves future actions and decisions. Thus, how visual information is processed and represented will significantly influence how a robot can plan and act.

Powered by modern computing technologies and big data, richness of visual information has considerably contributed to the recent advances in machine learning and robotics. Unprecedented performance is witnessed in certain areas such as image recognition and segmentation ([93][67][109][202] to name a few). However, we still find it hard to apply the advances in computer vision and machine learning directly to problems of real robotic systems. In fact, the physically embodied nature of robots requires a different quality of data and processing from pure computer vision [176].

First, there is a gap between what most benchmark datasets provide and what a robotic vision system would perceive. Most large-scale datasets such as [42][197] consist of images of
independent moments curated by humans. However, a robot takes a stream of consecutive and correlated image frames, the viewpoints of which will vary as the robot moves around. It is also questionable how web-based data can help a single robot’s unique visual experience. In order to successfully apply learning algorithms, it will be important to enable robots to use priors for gathering data by themselves.

Another important aspect of robot vision is that vision is one part of the whole robotic system. It is expected that the visual perception can lead to physical actions or abstract decisions. It will be much more interesting and useful if a robot can pick up an object rather than just telling its location and object category. However, the link from vision to action is a big question in both human and robot intelligence research. As a fundamental issue, how the spatial information should be represented requires more research.

Acknowledging that it will be a long way for the mentioned challenges to be resolved, this dissertation seeks principled methods for robot vision to enrich and represent knowledge utilizing underlying structural priors.

In Part I, an online self-supervised framework for segmentation and pose tracking is proposed. The framework is able to self-segment appearance information from a stream of
data while annotating the 3D target pose information by utilizing geometric priors. Unlike conventional approaches, it does not involve human hand-labeling to obtain image labels or 3D pose. Thus this approach enables collecting unlimited amounts of data given the shape models under moderate changes. The self-supervised learning framework is demonstrated via two example systems in Ch.4 and Ch.5.

In Part II, an online update method for continuous spatial representation is proposed. It adopts a continuous surface representation of the environment in the form of distance fields using Gaussian Processes. Unlike conventional methods based on points or grids, this representation does not suffer from artifacts of disconnected information, naturally taking the correlation of neighbor points into consideration. Also, the distance and its gradient fields provides rich information for determining actions, which binary occupancy maps cannot do. The proposed method contributes to the first probabilistic online implementation of the representation.

List of Related Publications and Releases

  
  Video: https://youtu.be/lpYHxgBTpxA,
  Code: https://github.com/leebhoram/OnlineLearningGround

- [100] *Online Learning of Visibility and Appearance for Object Pose Estimation*, IROS 2016,
  [101] *Self-Supervised Online Learning of Appearance for 3D Tracking*, IROS 2017

  Video: https://youtu.be/PSyGC452E3k

- [103] *Continuous Online Mapping using Gaussian Process Implicit Surfaces*, ICRA 2019

  Video: https://youtu.be/_EqeoLeHzXU
  Code: https://github.com/leebhoram/GPisMap
Part I

SELF-SUPERVISED APPEARANCE LEARNING
Part I presents an online method for self-labeling by utilizing shape priors. One limiting aspect of currently dominant machine vision approaches lies on its offline training scheme based on supervision by humans. The standard way of getting training data heavily depends on human labor using web-based platforms [2][7][5]. However, no matter how much data is provided, offline data collection by humans will lead to temporal and qualitative gaps between the training data and the actual data that a robot would observe. After introducing the key concepts of Part I in Ch.2, Ch.3 reviews related work on each domain of the following chapters, Ch.4 and Ch.5, which demonstrate an online self-supervision framework for visual learning in the domain.

In Ch.4, the suggested self-supervised learning is applied to ground estimation and segmentation for monocular visual odometry. It improves the odometry performance in terms of scale recovery of the monocular scene by finding the ground region. The visual segmentation does not require any prior on ground appearance, but learns from online samples provided by the geometric prior.

In Ch.5, the self-supervision concept is extended to 3D rigid-form object instances. We explicitly incorporate motion into the framework and track the 3D pose of the target. Without any prior on the target’s appearance, the suggested method achieves accurate segmentation under challenging conditions such as occlusion by touch. Also, by directly using the shape prior for optimal pose estimation, it demonstrates precise and accurate 3D pose tracking performance.
Chapter 2

Definition

Before proceeding to present details, let us clarify the key concepts used in the following chapters in Part I. More technical terms will be defined and discussed in each chapter, but here we list the keywords of Part I and provide the intended meanings.

Online Learning

When learning is viewed as an optimization problem with respect to a set of examples, two main training paradigms emerge: batch learning and online learning [160]. The batch learning paradigm aims to solve the problem while consuming the entire set of examples simultaneously. Once the model is trained, it can be used for test afterwards (Fig.2.1 (a)). On the other hand, in online learning, parameters are updated as each example is presented. It can be shown that, as more and more examples are consumed, the model converges under a proper learning rate (Fig.2.1 (b)). Whereas batch learning could suffer the burden of storing and processing a large training set, online learning can be described even without reference to a training set [21]. Online learning can be naturally applied to non-stationary problems (Fig.2.1 (c)) while offline batch learning would require retraining on new varying data sets. This is very important for robotics applications since robots would be deployed in the real world and they would need to react immediately to changes in their status or the environment. This study considers potentially non-stationary tasks and is interested in
Figure 2.1: Illustration of (a) Offline Batch Learning, (b) Online Learning with Convergence, and (c) Online Learning for Non-Stationary Concept.

‘tracking’ the changes. An implied assumption is that the changes occur gradually. This follows the common notion in computation learning theory that any algorithm will fail if the concept drifts rapidly or randomly before observing enough examples [95].

Self-Supervised Learning

Self-supervised learning refers to a supervised learning scheme involving no human supervision. Some studies using synthetic data also employ this term since the labels are generated by the synthesization process itself. On the other hand, our setting does not involve synthesizing data but considers learning from real data. The way we implement self-supervision on real data is to take advantage of the naturally available correlated sensations of sensory modalities [41]. To illustrate this concept, consider a real life moment encountering a scene of dog and it is likely that the visual experience of a dog is often accompanied by the sound or the touch of dog. This applies among various sensory modalities, but the scope we discuss remains within vision and deals with shape and appearance. Assuming that any change in
Figure 2.2: The natural correlation of multi-modal sensations is a great source of self-learning.

the correlation of shape and appearance occur gradually, we can track the changes in appearance without relying on its prior. In Ch.4 and Ch.5, We demonstrates self-supervised learning by leveraging relevant shape priors in order to alleviate the human dependency in appearance learning.

**Shape and Appearance**

Shape and appearance, being the most important visual sub-modalities, have a long and rich history in vision research. Although it is hard to completely separate one from the other, it is known that the two modalities are handles somewhat independently in our brains [84]. In this dissertation, we borrow the simplified view of independent shape and appearance processing while allowing interactions.

In general, appearance comprises aspects such as illumination, reflectivity, color, and texture. When we say *learning an appearance model* in the following chapters, it is assumed that a rich feature space exists and what is learned is the distribution of the particular target object in the feature space. To give a simple illustration, an apple could be the target, HSV color could be the appearance feature space, and the model could be a vector value \((142, 40, 56)\). This differs from the problem of learning representation itself. Our self-supervised learning scheme has no restriction to a particular choice of feature, but it is implemented using a quantized color representation (Ch.4) and a Convolutional Neural Network (CNN)-based higher dimensional representation (Ch.5).

Shape is the spatial and geometrical characteristics of the object. By shape, we usually mean something that remains invariant to certain transformations such as translation and rotation [84]. Nevertheless, it often requires using some coordinates to represents math-
In order to implement the self-supervised learning framework in Part I, shape priors are expressed using widely-used representations: parametric surface models in Ch.4 and point cloud representations in Ch.5. Furthermore, we propose a new online method for continuous shape representations in Part II. *Shape, structure, and map* may be used interchangeably with the subtle difference that map is often used to refer to spatial information of the whole environment.
Chapter 3

Related Work

3.1 Ground Recognition

Recognizing the ground surface is one of the most important visual tasks for the perception of the world [134]. The ground or floor gives a relative context to the scene providing complementary information to other complex tasks such as object localization and recognition. Imagine an object floating in dark space versus the same object on a continuous background surface; The latter provides a better context for spatial and semantic recognition of the object [58]. In applications of ground robots, the problem of accurately estimating the ground surface has been particularly important for autonomous navigation and exploration.

In this chapter, we survey various efforts on robotic applications for which ground recognition is important and discuss their approaches. Specifically, we consider three scenarios:

1) navigation in wild environment where geometric and mechanical properties of terrain matters, 2) navigation in urban settings where finer semantic classification of the scene including roads and lanes is more important, and 3) navigation with monocular vision where the ground surface can help recovery of the scale of the visual scene. In this review, our discussion focuses on what types of data or information are used and how they are collected.
3.1.1 Terrain Traversability in Wild Environment

Earlier autonomous vehicle studies considered exploration scenarios in unstructured or rough-terrain environment [4, 79], where the vehicles were expected to figure out safe path to the goal location. **Geometry-based approaches to this problem concern the analysis of elevation profile of terrain using range or depth sensing.** If we rely only on this approach, due to the limited look-ahead distance of 3D sensors, the speed of vehicles has to be remain slow for safety. Moreover, using only terrain geometry often fails to assess traversability since it does not reveal properties and conditions of terrain. **Approaches involving other sensing modalities, especially proprioceptive and visual sensing, are suggested** to overcome these issues [136]. Proprioceptive-based methods such as [78, 105, 191] are capable of directly learning the difficulty in traversing different types of terrain by analyzing various sensory inputs, for example, vibrations and wheel slips. Appearance-based studies such as [17, 75, 86] focus on learning visual features to classify the types of terrain or regress traversability measures.

There have been **hybrid approaches combining these ideas to associate the visual appearance of the ground** with its traversability, or with its height profile. For example, in [76], Howard *et al.* suggested learning near-field traversability from proprioception, and color-based visual learning for mid- and far-field traversability from 3D geometry. Similarly,
the study by Thrun et al. [38, 182] used LIDAR to scan the near-distance terrain and to train a Gaussian mixture model-based ground classifier in order to identify drivable path in color images. Vernaza et al. [186] used a stereo camera to provide supervised inputs to the Markov Random Field classifier based on monocular color images. Brooks et al. [23] suggested online self-supervision for visual classification using a trained traversability model based on proprioception. More recent reviews on approaches on the same line for the integration of RADAR and monocular vision can be found in Milella et al.’s work[123]. These self-supervised learning methods are established based on the correlation of 3D geometric properties and appearance of obstacles versus drivable terrain (Fig. 3.1).

3.1.2 Road Detection in Urban Settings

More recent autonomous vehicle studies consider driving on paved roads in the urban environment, where the distinction between drivable roads, lanes, pedestrians and so on is critical. Since roads and lanes for cars are defined by man-made rules rather than geometric properties, most studies on road detection use appearance information with or without a geometric analysis [70, 124]. Various strategies are found in the literature; To give some examples, texture statistics [61], inverse perspective mapping [27], disparity of stereo vision with clustering in color space [172], homography estimation with learned appearance [63], Omni-directional images and optical flow [200], and more ideas have been applied to drivable road detection.
Compared to those conventional approaches, recent CNN-based approaches to segmentation eliminate the steps involving hand-designing features or descriptors. CNN models developed for general classification tasks (e.g., [67, 93, 166]) are being adopted and fine-tuned or augmented with additional architectures for autonomously driving cars [29, 106, 122, 177]. One of the drawbacks of these data-driven approaches is that they require a large amount of labeled data, such as the Cityscapes dataset [35] or the KITTI dataset [55].

With a growing concern for the labeling issue, researchers have considered self-supervised approaches to the road detection problem. In [140], Paz et al. exploited a small region of the inverse depth image as a seed to propagate road regions in the sequence of images. Although they failed to achieve full self-supervision reporting ‘reset’ cases, their approach is better scalable than pure hand-labeling. Laddha et al. of [96] presented a drivable road detection method for monocular vision with self-annotations generated by using 3D maps (OpenStreetMap [13]) and localization information from GPS/IMU sensors. The authors reconstructed local 3D scenes based on the map data and projected the road regions on the images captured from a vehicle using the estimated pose. After this initial labeling step, some low-level image processing is involved to further improve the labels. Another large-scale multi-source dataset utilizing automated techniques to create ground truth labels includes the TorontoCity dataset [190]. They developed algorithms to align the map data including road to their source of imagery. This map-based self-supervision approach shares the high level motivation with our work. They take advantage of other existing data in the form of map that provides spatial information of roads.

3.1.3 Ground in Monocular Vision

Finding ground also has been special for monocular vision, which suffers from the unknown scale of the scene. There have been various ways suggested to resolve the scale problem. Depending upon the desired application, explicit initialization steps are required in order to obtain the scale. For example, MonoSLAM [40] set an initialization target and PTAM [92] devised a predefined initialization step by users, both specifically targeting indoor augmented reality (AR) applications. In the work on real-time monocular visual odometry called SVO
Figure 3.3: Ground as a Reference Object in Monocular Vision: Finding ground can help to recover the scale of the scene.

[52], the ground truth scale of the first few frames was used for initialization and the scale was propagated afterwards. Other work that attempted to compute the scale every frame had to rely on external references. With today’s proliferation of inertial sensors, the main approach to estimate scale from monocular sequences is the fusion of camera and IMU data inertial sensors [130], [69]. Other sensors such as air pressure sensors [15] or wheel odometers [161] could be valuable resources if they are readily available in an existing vehicle platform. Some recent data-driven approaches include using stereo vision for training a neural network to obtain scaled monocular depth [107], and using full 3D poses for training a neural network to directly predict 3D pose between image frames [189].

In applications for ground vehicles, the ground plane has been used as the reference of the scale. If the distance between the camera and the ground surface is known, the scale problem is reduced to finding the ground as a reference scale from the scene (Fig. 3.3). Ground plane can be detected in monocular images geometrically in a way similar to some of the approaches described in Sec. 3.1.1. The difference is that the problem is reversed: 

**Instead of looking for the ground surface at a certain elevation, we may first attempt to find the dominant planar surface and expect the surface to be at the ground elevation.** For example, one of the early studies by Mallot *et al.* [113] applied an inverse perspective mapping to find the ground plane and obstacles assuming that the brightness of pixels on the ground is constant. In [91], Kitt *et al.* adopted the idea that the ground is usually captured in the lower center part of the image and used a predefined region of interest (ROI) for referring the ground plane. Choi *et al.* [32] adopted a texture
analysis algorithm to predict the ground region first and applied the planar homography to recover the scale. In LibVISO by Geiger et al. [54], a density estimation technique was utilized to detect the ground plane and a similar approach by Choi et al. can be found in [33]. The work by Song et al. [170] suggested using a combination of several cues of the above-mentioned approaches including density estimation and ROI.

3.2 Object Pose Recognition in 3D

In order for robots to interact with the environment, for example via grasping or manipulation, the ability to discriminate targets from the scene and to know the targets’ spatial information is very important. Although there have been many model-free 3D tracking studies, for instance [20, 68, 128, 171, 196], in this chapter we concentrate on model-specific pose recognition, for which objects can be identified from their known shapes and the 3D pose can be clearly defined. We acknowledge that the 6DOF pose might not be the best way to represent an object’s status. However, 6DOF pose estimates provide a practical solution to object manipulation planning, and can be further processed to another form of representations.

We first review approaches on pose estimation in Sec. 3.2.1 and pose refinement methods in Sec. 3.2.2. Lastly, in Sec. 3.2.3, we review recent 3D datasets with a focus on how ground truth pose has been collected and what the limitations are.

3.2.1 Pose Estimation

There exists a large body of work dedicated on this topic. This section does not include an exhaustive list by no means. The intention of this section is to provide a brief overview on various approaches on instance-specific rigid body pose estimation (Fig. 3.4) and discuss the value of self-supervision for them. Good references to literature survey for pose estimation include [104, 114].

Shape appears as the primitive feature for object recognition in earlier studies on robot vision [126, 127, 156]. Various geometric features have been used for object pose recognition via shape analysis since then. Similar to image descriptors, 3D point descriptors have been
proposed based on the distribution of surface normal (“Splash” [173]), surface curvature ([43]), spin image [81], angular features between neighboring normals ([158, 159]) for shape recognition. Instead of defining a descriptor per point, features defined for pairs of points are also used to describe and recognize shape of objects [45]. Also, there exist global shape registration solutions [26, 174, 199] that works in the Branch-and-Bound scheme. Note that these approaches largely depend on 3D geometric information and do not address the target segmentation issue.

Appearance is the other major axis of object recognition studies. Especially during the past couple of decades, various visual feature or model representations [34, 39, 48, 67, 93, 111] are explored and they have made a significant contribution to pose recognition studies. Also, with the increased popularity of depth sensors, much recent work has explored directly using depth data with RGB-based appearance to derive shape information. Whether RGB or RGBD, there are approaches in which a model is defined by descriptors for a set of spare points and 3D pose is recovered using the geometric relation of the points. For example, Gordon et al. [60] suggested a method that uses the scale-invariant feature transform (SIFT) keypoints [111] to reconstruct 3D model as well as to match them with observations to estimate the pose. Pauwels et al. [138] also used SIFT features for their detection module which competed with a rendering-based pose tracking module. The handcrafted features such as SIFT are only applicable to texture-rich objects. Hodañ et al. [72]

---

<table>
<thead>
<tr>
<th>Approach</th>
<th>3D Shape Analysis</th>
<th>Direct Optimization</th>
<th>Sparse Features</th>
<th>Learning from Holistic Images</th>
<th>Rendering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Examples</td>
<td>Surface normal</td>
<td>Brand-n-Bound</td>
<td>Keypoints</td>
<td>Templates</td>
<td>Full image, Gradient, Boundary, Flow</td>
</tr>
<tr>
<td></td>
<td>Curvature</td>
<td></td>
<td>(corners, SIFT, learned,…)</td>
<td>DPM</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Spin image</td>
<td></td>
<td>Edges</td>
<td>CNNs</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Point-Pair features</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 3.4: Various Pose Estimation Approaches
employed templates based on a geometric descriptor defined on depth image to deal with
texture-less objects. In a study by Pavlakos et al. [139], semantic keypoints are found using
a CNN (‘stacked hourglass’[129]) and the 3D pose is recovered by an optimization procedure
for finding the best linear combination of basis models.

Some approaches use holistic images of different views rather than descriptors for some chosen points. Template-based approaches has been successful, where the target is modeled as a set of templates annotated with 3D poses. In particular, a method called LineMOD proposed by Hinterstoisser et al. [71] has been implemented in OpenCV [12] as well as PCL [14], and stimulated many other studies. For example, Rios-Cabrera et al. [155] and Wohlhart et al. [194] presented improved template-based methods via discriminative learning. Instead of having holistic-appearance templates for the whole object, researcher also explored ways to predict pose based on parts or points. Zhu et al. [204] suggested a pose estimation method that uses Deformable Part Model (DPM)[47] to learn discretized views of targets. Brachmann et al. [22] proposed a per-pixel object classification and coordinate regression method using Random Forest.

We can also find approaches that sample poses and render the scenes given 3D models with or without predefined appearance models. Among the mentioned studies, Zhu et al. [204] suggested refining the pose estimate from DPM by fitting rendered the object boundary to observed image boundary, and Pauwels et al. [138] suggested a similar rendering method using ‘AR flow’ to better track the pose. Online sampling methods such as particle filter (PF) are also suggested to track 3D pose as found in Choi et al. and Krull et al. [94].

One approach in deep-learning-based studies involving pose estimation is to have two-stage strategy of semantic segmentation or detection followed by pose estimation [64, 175, 183]. These studies suggested coarse pose prediction methods treating the problem as classification. Poirson et al. [146] also handled the problem as classification but without separating the detection from pose estimation. Tekin et al. [178] used CNNs for predicting target locations and bound boxes, followed by the classical PnP solution for pose estimation. Xiang et
3.2.2 Refined Pose Estimation

Pose estimation performance of template, sampling, or learning-based approaches usually show limited accuracy or precision. Since 3D pose is in SE(3), it is hard to have templates or training samples enough to cover possible 3D poses in fine resolutions. Thus, many pose estimation methods require a refinement step to achieve more precise performance. A large portion of the mentioned studies above include or mention refinement steps.

One popular method based on an optimization scheme is called Iterative Closest Point (ICP). Since its introduction, the ICP algorithm [18, 28] has been a popular shape registration method, many variants of which have been published ([157, 165]). However, there are some restrictions in applying ICP successfully: It requires a good initial pose and a good segmentation of the target, since it is a local optimal solution. For this reason, the use of ICP has been mostly restricted for mapping applications where the whole scene is the target [11], or pose refinement if for object-level targets [64, 198]. However, some recent studies presented interesting ideas in using ICP for object tracking. For example, Wang et al. suggested using ICP for object pose estimation after segmentation CNN. Also, Schmidt et al [164] proposed a self-supervised descriptor learning method using non-rigid ICP. Recent approaches based on deep learning [108, 187] also have presented iterative pose refinement concepts.

3.2.3 3D Pose Annotation

We have reviewed various approaches to 3D pose estimation in the previous sections. One common issue across all the studies will be how to obtain or prepare data for training and evaluation. Thanks to generous researchers of the community, now we have numerous public datasets that can be freely used for research. A recent survey on RGBD datasets for various applications can be found in [50]. In this section, we discuss how ground truth 3D pose of
objects are annotated for such datasets. It should be noted that there exist not so many of them since obtaining 3D pose is usually expensive as we will discuss. Although synthetic datasets are useful providing ground truth by design, we focus on real object data.

First, we may rely on manual annotation as done in PASCAL3D+[197]1 (Fig. 3.5). However, even for humans, there exist limitations in labeling 3D pose since it involves first defining the object-centered coordinate with a 3D model. Without a proper tool to visualize and manipulate the model, it is hard for a person to produce 6 DOF pose information just from an image. Thus, this annotation method should include steps to choose the right 3D model and adjust its pose on the image. Optionally, keypoints can be defined per the model to make the registration easy. If multiple views of the same scene are available, it is possible to make annotations for the whole 3D scene, which can be shared across different views. This idea is found in LabelFusion [115] and the Rutgers APC dataset [154].

There have been efforts to automate the data capture or labeling process. If we have access to the background-only image, then it is possible to subtract the background from the scene. This provides isolated object images making the segmentation easy, but there must be other process to compute pose. In the study by Zeng et al. [201], a robotic arm is used to automatically capture different views around targets. They also utilized

1The labeling process can be seen at https://youtu.be/5Yeus0x6fo0
image processing techniques together with ICP on depth images to get clean labels.

Visual fiducial markers such as AprilTag [132] are probably the most popular way to estimate the camera pose and infer object 3D poses as well. Use of checkerboard patterns or visual markers are found in T-LESS dataset [72], the BigBIRD dataset [167], LineMOD dataset (ground truth for test) [71]. Drawbacks of using visual markers include that it is hard to capture natural scenes, and that the object must remain static relative to the markers. External motion capture systems can be also used to precisely calibrate and measure 3D information on the environment and objects [100, 195]. However, it is an expensive method with spatial constraints in using the system.
Chapter 4

Online Ground Segmentation and Estimation

In this chapter, we delineate our self-supervised online learning approach for visual odometry and ground segmentation in monocular vision. While stereo or depth cameras directly provide 3D structure information, monocular vision presents several difficulties in solving the geometry problem in general. Especially the absolute scale of motion and scene structure is unknown if only using a monocular camera. In ground vehicle applications, the distance between the camera and the ground surface can be used to recover the scale and then the scale issue is reduced to finding the ground as a reference scale from the scene. Thus, the problem of estimating the ground surface in 3D has naturally arisen in the domain of autonomous navigation and exploration [113][91][54][32][33][170]. On the other hand, rich

Figure 4.1: Finding Ground via Combining (Left) Geometric Estimation and (Right) Appearance-Based Segmentation on Image Plane
appearance information can be used to find the ground as well. Given proper samples of pixels, a classifier can be trained to form priors of target appearance models in a machine learning framework. We integrates both geometric estimation and appearance-based classification for ground surface (Fig. 4.1) in an online self-learning scheme to achieve scale recovery and segmentation for monocular vision (Fig. 4.4).

4.1 Contributions

• We seek to recover the scale while finding where the ground is and learning how the ground appears simultaneously. This differentiates our approach from work that rely on either geometry or appearance.

• We utilize geometric ground estimation as self-labeling to train a classifier that segments the ground based on appearance on the fly. Unlike traditional supervised learning for segmentation, this approach does not require expensive, offline hand-labeled data, but uses automatically found data that is very relevant to the present moment. Also visual segmentation by the classifier improves the accuracy of ground surface estimation compared to naive estimation.

• Our effort to find the ground can also contribute beyond the scale recovery problem. Finding background surface in general is complementary to other complex tasks such as object localization and recognition. Thus, our method could be combined with other classes of objects to achieve a higher-level goal.

4.2 Models

We first review the background of our mathematical models for camera motion, ground model, and ground vs. non-ground classifier, which are incorporated in our system described in Sec. 4.3. The fundamental coordinate frames—the body coordinate frame $C^b$, the camera coordinate frame $C^c$, and the image plane coordinates $C^i$—are indicated in Fig. 4.2. We define ground as simply the lowest portion of the 3D scene where the wheels of ground vehicle would contact.
suppose that the camera center is fixed at the vehicle center and that the camera motion represents the vehicle motion well enough. The camera axes rotation is also assumed to be aligned since it can be easily compensated once the relative pose is known. Although we demonstrate the framework using the image kinematics based on the nonholonomic motion model for its simplicity, it can be generalized to 3D motion. Thus one may apply the standard SFM (Structure from Motion) [66] to obtain image kinematics and reconstructed scene from the SFM method instead of following 4.2.1.

4.2.1 Ground Vehicle Motion and Image Kinematics

Previous works such as [161, 162] have explored constrained motion models for a camera mounted on a ground vehicle. These works solve a parameterized essential matrix to estimate motion, which is based on a homogeneous equation. Although we also consider a constrained motion, our method uses the explicit kinematics of the camera motion and the image flows to obtain motion and structure. If the robot is wheeled, we may take advantage of the constrained motion of the wheeled vehicles when interpreting motion from images. For instance, nonholonomic vehicle motion along the ground surface constraints the number of motion parameters for computational efficiency.

A ground vehicle motion is nonholonomic and dominant in two dimensions. (Detailed kinematics of it can be found in [83].) We adopt this nonholonomic motion model and describe the vehicle motion only using the translational speed \( V \), the heading orientation \( \theta \), and the rotational velocity \( \omega \). Also we let \( \omega = \dot{\theta} \) under the assumption of tangential motion, i.e., the heading is assumed to be tangential to the path. Then, the vehicle trajectory can
be obtained by accumulating translational and rotational velocity estimates, \( V \) and \( \omega \).

Now let us review the explicit relation of \( V \) and \( \omega \) of the camera with the image flow vectors. (The full derivation can be found in [77].) Based on the pin-hole camera model, the projective mapping from a point in 3D in \( \mathbb{C}^c \), 
\[
x = \begin{bmatrix} x & y & z \end{bmatrix}^T,
\]
to a point in \( \mathbb{C}^i \),
\[
x = \begin{bmatrix} u & v \end{bmatrix}^T,
\]
can be written as,
\[
\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{z} \begin{bmatrix} x \\ y \end{bmatrix}, \tag{4.1}
\]
where the image coordinates are normalized with respect to the focal length.

Suppose that the camera is in motion with translation velocity \( \mathbf{V} = \begin{bmatrix} 0 & 0 & V \end{bmatrix}^T \) and rotational velocity \( \mathbf{\Omega} = \begin{bmatrix} 0 & \omega & 0 \end{bmatrix}^T \) with respect to the current camera frame \( \mathbb{C}^c \). The instantaneous velocity of the point with respect to \( \mathbb{C}^c \),
\[
\dot{\mathbf{X}} = -\mathbf{\Omega} \times \mathbf{X} - \mathbf{V},
\]
can be expressed component-wisely as,
\[
\dot{x} = -z\omega, \quad \dot{y} = 0, \quad \dot{z} = z(\mathbf{\omega}) - V. \tag{4.2}
\]
From Eq.(4.1) and Eq.(4.2), the derivatives, \( \dot{u}_i = (z_i\dot{x}_i - x_i\dot{z}_i)/z_i^2 \) and \( \dot{v}_i = (z_i\dot{y}_i - y_i\dot{z}_i)/z_i^2 \) for the \( i \)-th point can be written as,
\[
\begin{bmatrix} \dot{u}_i \\ \dot{v}_i \end{bmatrix} = \begin{bmatrix} -(1 + u_i^2) & u_i \\ -u_iv_i & v_i \end{bmatrix} \begin{bmatrix} \omega \\ Vd_i \end{bmatrix}, \tag{4.3}
\]
where \( d \) denotes \( z^{-1} \). Note that, by letting \( \bar{d}_i := Vd_i \), the unknowns are reduced to \( \omega \) and \( \bar{d}_i \) for \( i = 1, 2, \ldots, N \) and now we have a set of nonhomogeneous linear equations. The linear equations can be easily solved to obtain the angular velocity \( \omega \) and the scaled inverse depths, \( \bar{d}_i \), of the \( N \) point correspondences.
Figure 4.3: Illustration of Parametric Ground Models: in order of simplicity, (a) flat plane with no tilt, (b) general plane possibly with tilt, and (c) curved surface up to the 2nd order.

4.2.2 Ground Surface Model

Let $h$ be the camera height from the ground and let us assume it is fixed. We consider the ground as a possibly non-flat surface using a general 2nd-order polynomial model (Fig. 4.3). It can be written, with respect to the current camera-centered frame, as $y(x, z) = a_1 x^2 + a_2 x + a_3 z^2 + a_4 z + a_5 xz + a_6$. Equivalently,

$$vd = a_1 u^2 + a_2 ud + a_3 + a_4 d + a_5 u + a_6 d^2,$$

(4.4)

where $d$ denotes $z^{-1}$ and $a$’s are used to indicate arbitrary coefficients except that $a_6 = h$. Given a enough number of ground points, it is possible to estimate the model parameters in a least-squares sense. Also, we may regularize the problem to prevent an overfit if necessary.

Note that it represents a general 3D plane if $a_1, a_3,$ and $a_5$ are zeros as used in [186]. When $a_2$ is also zero, it is equivalent to the tilted plane about the $x$-axis (Fig. 4.3 (b)) as in [170].

4.2.3 Ground Color Classifier

In order to associate the geometric ground estimation with the ground classification problem, we use a Naive Bayes classifier based on a quantized color space. The Naive Bayes approach is simple yet effective for inductive learning algorithms. Nevertheless, this is a particular chosen feature and classifier. This implementation can be modified to have other features and classifiers. For example, as we will see in Ch. 7, using latest CNN with a neural network classifier can be another choice.
The color space can be quantized into \( n_c \) labels and the color label set can be expressed as \( L = \{1, 2, \ldots, n_c\} \). Let \( l \) indicate the color label of each pixel. Then \( l^{(m)} = j \) implies that the \( m \)-th pixel in the neighborhood has the color value corresponding to the label \( j \in L \). Also, let \( w \) be the number of pixels in the neighborhood of the \( k \)-th point of an image and \( X_k = \{l^{(1)}, l^{(2)}, \ldots, l^{(w)}\}_k \) be the set of color labels of the neighborhood of the point. Then we seek the class label

\[
C = \arg \max_{C \in \{0, 1\}} P(C|X = X_k). \tag{4.5}
\]

for the \( k \)-th point. Assuming conditional independency, \((l^{(m1)} \perp l^{(m2)}|C)\), we may write

\[
P(C|X = X_k) = \frac{\prod_{m=1}^{w} P(l^{(m)}|C)P(C)}{P(X = X_k)}. \tag{4.6}
\]

Given the statistics \( P(l|C) \) and \( P(C) \) learned from examples, we can evaluate \( P(C = 1|X) \) versus \( P(C = 0|X) \) and decide whether a point is ground or not.

### 4.3 Online Self-Supervision Framework

The overall workflow of the suggested scheme is illustrated in Fig. 4.4. In the odometry part, \( S = \left[ \omega \{d_i\}_{i=1}^{N} \right] \) is estimated from image observations in a Bayesian framework that maximizes the posterior distribution of it. The likelihood of \( \bar{S} = \left[ \omega \{\bar{d}_i\}_{i=1}^{N} \right] \) is first computed and \( S \) is obtained after the scale, \( V \), is recovered by finding the ground surface. In ground classification, the classifier is updated when new ground samples are received. The updated classifier then predicts the probability that a point is from the ground in the incoming image. This probability is used for generating better hypotheses of the geometric ground model.

#### 4.3.1 Motion and Inverse Depth Estimation

The kinematic model Eq. (4.3) can be written as

\[
A\bar{S} = O, \tag{4.7}
\]
where $A$ is a $2N \times (N+1)$ matrix and $O$ represents the observation vector, $O = \begin{bmatrix} o_1 & o_2 & \ldots & o_N \end{bmatrix}^T$, where $o_i = \begin{bmatrix} \dot{u}_i & \dot{v}_i \end{bmatrix}$. Note that the squared matrix $A_s = A^T A$ is a symmetric arrow-headed matrix. A least squares solution of Eq. (4.7) can be solved very efficiently to obtain the likelihood distribution $p(O|S)$. It is supposed that every image observation has Gaussian noise given $S$ and is measured independently.

Given the likelihood and a prior distribution, $p(S) \sim N\left(S_0, \Sigma_0\right)$, the MAP estimate

$$\hat{S} = \underset{S}{\text{arg max}} \ p(O|S)p(S)$$

and its covariance matrix can be updated as the following,

$$\hat{S} = \hat{\Sigma} S (\Sigma^{-1} S_0 \Sigma^{-1} + \Sigma^{-1}_{ML} S_0)$$

$$\hat{\Sigma}^{-1} = \Sigma^{-1}_{ML} + \Sigma^{-1}_{S_0}. \tag{4.8}$$

The propagation of $z$ can be written as $z_{k+1} = z_k + \dot{z}dt$ and from this, for $d$,

$$d_{k+1} = (d_k^{-1} + d_k^{-1} u \hat{\omega}_k dt - \hat{V}_k dt)^{-1}. \tag{4.9}$$

For the variance in the linearized propagation, we have

$$\sigma_{d_{k+1}}^2 = J^2 \sigma_{d_k}^2 + n, \tag{4.10}$$

where $J$ are given as,

$$J := \frac{\partial d_{k+1}}{\partial d_k} = \left(\frac{d_{k+1}}{d_k}\right)^2 (1 + u \hat{\omega}_k dt), \tag{4.11}$$

27
and $n$ denotes the propagation model uncertainty.

We avoid directly attacking the issue of depth ambiguity of monocular vision in this scope, considering possible dynamic objects as noisy measurement. However, the inverse depth of every feature point and its uncertainty are tracked in a Bayesian framework, thus such noisy points tend to be filtered out of the predicted search ranges.

### 4.3.2 Ground Surface and Scale Estimation

The scale factor $V$ is recovered by finding the ground points and fitting the points based on one of the ground surface models discussed in Sec. 4.2.2. To this end, we need to modify Eq. (4.4) with $\bar{d} := Vd$ in place of $d$ like the following,

$$
\bar{d}^2 = a_1' u^2 + a_2' u \bar{d} + a_3' \bar{d} + a_4' \bar{d} u + a_5' v \bar{d}.
$$

(4.12)

It can be easily shown that $a_6' = V/h$, from which $V = a_6'h$ is computed. (The prime symbol is used to denote that the parameters are different from $a$'s of Eq. (4.4).)

In order to find points that agree well with the model, we chose the RANSAC algorithm [51], accepting the assumption that non-ground points tend to form gross errors. Unlike the standard RANSAC algorithm (Fig. 4.5, left), our RANSAC loop selectively samples points based on the appearance-based ground probability (Fig. 4.5, right). Fig. 4.5 visualizes the biased weights of points in the RANSAC domain of the two-parameter model ($a_1' = a_2' = a_3' = a_5' = 0$). If we know the probability that each point belongs to the ground, then it is reasonable to sample more probable points to generate a hypothesis. Yet, since there is uncertainty, we still allow a chance of every point to be sampled by assigning a weight proportional to the probability. We contrast the two-parameter model versus the six-parameter model in Eq. (4.12), in order to show the effectiveness of our suggested learning scheme in Sec. 4.4. After the RANSAC procedure, $V$ is filtered by an adaptive Kalman Filter to handle occasional failures of RANSAC.
4.3.3 Ground Classification

The inliers of the estimated ground model are used as the training samples for the class $C = 1$ (Ground) and others are for the class $C = 0$ (Non-ground). Once the histogram of color labels for each class is initialized with the first set of samples ($H_1$), it can be updated for the $k$-th frame with a decaying rate $0 \leq \alpha \leq 0$ in the following fashion,

$$H_k = \alpha H_{k-1} + H_{\text{new}}. \quad (4.13)$$

Based on the current histograms, we evaluate $P(C|X)$ as described in Sec. 4.2.3. An example is shown in Fig. 4.6. As the environment changes (the first column), the classifier learns the changed feature (the middle column). When the next frame is available, the classifier predicts the class of each pixels (the last column) and the predicted probability is used to bias the weights of points in the RANSAC sampling for ground surface estimation (Fig. 4.5, right). Consequently, since the biased sampling will generate hypotheses from more probable points, we can expect that the performance gets better or that the number of repeating generating hypothesis is reduced. The experimental evidence of our argument is shown in the following section.

4.4 Results

This section describes an implementation of the proposed online ground estimation and segmentation method and show the experimental results. We used [54]'s method for feature
Figure 4.6: Learning Ground Feature (Images from KITTI Benchmark Odometry Dataset [55] No.03): The classifier updates the histogram over the color labels and predicts the probability that each pixel is from the ground based on the learned feature.

detection and matching but without the refinement of matching and the outlier removal. Instead of using matched points between two images, we employed points that survived over three sequential images to obtain the image flow of the centered image. This required no additional computation since points were being tracked already, rather it resulted in removal of noisy points and more accurate observation of instantaneous image velocity.

Static scene with no motion was detected by checking the ratio of static points. If the vehicle was detected to be static, the computation for odometry was not done and ground appearance was not updated. Otherwise, an iteratively re-weighted least squares (IRLS) solution of Eq. 4.7 was computed with a maximum number of iterations (= 10) to obtain the motion and inverse depths.

As for the ground classifier, the color values of an image were converted into HSV and quantized into 216 labels \( n_c = 6^3 \), 6 per channel). The decaying rate of the histogram update was set to be 0.7 and the window size for the neighborhood was set to be 3 for learning and 1 for testing.

Although the current implementation was written in MATLAB and C++, a future real-time implementation could be easily parallelized. The odometry and the feature learning can be run in separate threads due to the nature of the suggested scheme. The average
computation time ranged from 0.1 to 0.16 seconds per frame according to experimental conditions when tested on a laptop PC with Intel Core i7 3GHz Processor.

Our methods were tested on the KITTI dataset [55] from No.00 to 10 that are publicly available with the reference pose data. We first show two aspects of online learning for ground estimation: computational efficiency and accuracy. Next, the overall odometry performance based on the LibVISO devkit is presented. Two conditions of sampling in ground estimation (i.e. scale recovery) are compared according to the number of iteration of sampling and the order of the ground surface model. The condition denoted as ‘weighted’ or ‘w’ and marked in red of the figures involves online learning of the ground appearance feature. The other condition in blue is the case without online learning and it is marked as ‘uniform’ or ‘u’.

First, $p_s$, the proportion of successfully detected inliers using RANSAC, was evaluated. We used hand-labeled ground regions over 100 images of the first dataset as ground truth (GT) for this evaluation. The experiment was repeated 30 times for each case to correctly measure the performance. The boxplot on the left in Fig.4.7 illustrates the distribution of repeated results. (The boxplots of 6-param models are omitted for better visual readability and $\lambda$ indicates different regularization conditions.) The values of $p_s$ were fit with a general exponential form,

$$p_s = p_{max} - (p_{max} - p_0)p_i^{N_{iter}}$$

(4.14)

where $p_i$ is the probability that the minimal sample set are not entirely composed by inliers
and $p_{max}$ represents the maximum reachable $p_s$. Compared to the ideal RANSAC model, 
$p_s = 1 - p_i^{N_{iter}}$, it was observed that $p_{max}$ was saturated below one and $p_0$ appeared nonzero. 
This is because of the error distribution of the $\tilde{d}$’s that are estimated values themselves. The
inliers had non-ground points that happened to be inliers and some GT ground points also
happened to be mismatched and thus far from the set of inliers. The line plots of Fig.4.7
display the fitted models of Eq.(4.15). The results suggest that the performance improved
in an exponential sense as it is expected. The most noticeable fact from the plots is that
the weighted condition outperforms the cases without online learning for all numbers of
iterations. This implies that when we sample a single hypothesis based on the learned
weight, it is likely to have more inliers, and that the weighted sampling requires a less
number of iterations to achieve the same success rate than the uniform sampling does.

Second, we investigated the errors of instantaneous velocity (or scale) recovered directly
from the RANSAC loop. Here the regularization parameter $\lambda = 0.1$ was chosen for the 6-
parameter model. The absolute error of velocity estimate, $\Delta V$, was reasonably well modeled
as Laplace distribution according to,

$$f(\Delta V) = \frac{1}{2\Delta V_{mean}} \exp \left( -\frac{\Delta V}{\Delta V_{mean}} \right).$$

(4.15)

The values of $\Delta V_{mean}$ are plotted in Fig.4.8. We repeated the experiment ten times for
each case and took the average. Notice in Fig.4.8 that $\Delta V_{mean}$’s are smaller in weighted
samplings for all cases, which implies that the errors were smaller on average. Thus we can
observe that sampling based on the biased weights actually contributed to the improved
accuracy of geometric ground estimation. The effect of reduced computation might appear little for the two-parameter model since the range of required number of iterations is usually small. On the other hand, the effect becomes clearer for the higher order ground model since it basically requires much greater number of iterations for any significant improvement.

Finally, the overall odometry performance was evaluated for comparison of the different ground surface models as well as comparison with other works. Fig. 4.9 displays the average translation errors of each sequence and Fig.4.10 shows the evaluated translational and rotational error of all the sequences of No.00 to 10. The errors of VISO2-S (stereo) and VISO2-M (monocular) [55] provide a comparative performance.\textsuperscript{2} Even though our algorithm adopted a constrained kinematic model, and our results were obtained only from frame-to-frame estimation without an optimization technique over multiple frames, the translation performance of our system is more stable and significantly better on average than VISO2-M. The results shown in Fig.4.10 are based on the 2-parameter model of 16 iteration for RANSAC (‘u2′ and ‘w2′) and the 6-parameter model of 55 iteration (‘u6′ and ‘w6′).

Fig.4.11 shows the two best (top) and the two worst (bottom) of trajectory recoveries in terms of translational error (‘w6′). The average rotation error was the same for the four conditions as the conditions do not affect the estimation of the rotational velocity. The average translation accuracy of the 2-parameter model did not appear significantly different between two conditions. However, we can observe a significant improvement for

\textsuperscript{2}These systems are made publicly accessible by the authors who also provide the KITTI benchmark dataset. The evaluation is based on the average values of translational and rotational errors for all possible sub-sequences of length (100,200,...,800) meters. The parameters for VISO2-S/M were set as nms_n = 3, pitch = -0.03, half_resolution= 0, and height = 1.7 (only for VISO2-M). See http://www.cvlibs.net/datasets/kitti/eval_odometry.php
the weighted sampling condition of the higher-order model. Thus, increasing flexibility of the ground surface model became effective over all when the self-supervised learning was engaged.
Figure 4.11: Estimated Trajectory
Chapter 5

3D Object Segmentation and Tracking

In this chapter, we extend the self-supervision concept introduced in Chapter 4 to 3D object instances which have free rigid forms. As opposed to the ground surface, objects are dynamic and we explicitly incorporate motion into our model and track the pose of the target. Whereas a parametric surface model is used to estimate the ground surface in space, now we adopt the point cloud representation of object surfaces and use ICP for pose estimation. In addition, the segmentation is implemented with NN classifiers based on rich appearance features of CNNs.

Figure 5.1: Finding Ground via Combining (Left) Geometric Estimation and (Right) Appearance-Based Segmentation on Image Plane
5.1 Contributions

- Our method provides high-precision 6DOF pose of rigid objects from a RGBD stream given 3D point cloud models. It does not require training an appearance model of the target, and the target is allowed to move while being tracked. Once started under the common tabletop setting, it tracks the 3D pose of the target and provides the pixel-wise segmentation as well.

- We suggest an online classification method that learns from self-supervision by shape priors. This functions as a segmentation module for the online pose tracking, as well as an auto-labeling module for data gathering.

- This framework contributes as a pose tracking method with little efforts for training, and as a 3D object data annotation method with little restriction on the scene and little human intervention.

5.2 Models

We have designed our pipeline to have modules that are complementary but support each other. Before describing the probabilistic model of our framework, let us clarify what each module is.

- **Shape**  We use a shape prior as a means of identification and localization of an object instance. The colorless 3D point cloud representation is adopted, which is the most generic representation of shape, directly obtained from a depth sensor or a lidar scanner. (Sec.5.2.2)

- **Appearance**  Our approach treats shape and appearance rather independently. Our appearance-based classifier captures shape-derived appearance like the silhouette without explicitly relating appearance with viewpoint or shape. The classifier predicts the target region at fine-grained level in the absence of pretraining, learning from online
examples. We experimentally investigate two basic ways to combine features of various receptive field sizes for the online classification. (Sec.5.2.1 and Sec. 5.3.1)

- **Motion**  Motion in our work refers to kinematic state of both translation and rotation. In order to relate an object’s motion state and its probabilistic spatial occupancy, our motion-based prediction incorporates an approximate 3D volume representation of the object (Sec.5.2.1 and Sec. 5.3.2). One assumption we hold for the online learning to work is that motion changes incrementally in the sense that it could be generated by a moderate human action.

Now let us turn to the integration of these components. Studies in the human vision research community have explained the mechanism of object recognition as a probabilistic inference process [30, 85]. However, these models lack the notion of 3D and movement. Although the aim of this study is not building a fully Bayesian model, our probabilistic model provides a formal way to integrate 3D shape, depth observation, and motion in 3D with conventional models.

Let the color observation be $I$ and the depth observation be $D$ at time $t$ (See Fig. 5.2). Let $O \in \{0, 1\}$ be the variable to indicate the object which can be described by $S$ and $A$ and $T \in SE(3)$ be a variable representing the 3D pose of an instance of the object. Note that we describe the binary case for clarity but it can be extended to a multi-class problem.
We want to find the posterior pose,

\[ \hat{T}_t = \arg \max_{T_t} P(T_t | I, D, T_{t-1}). \]

(5.1)

Letting \( L \) be a variable referring to the occupancy of the object instance projected on the image space, a.k.a mask, the probability can be written as a marginalized probability over \( L \):

\[ P(T_t | I, D, T_{t-1}) = \sum_L P(T_t | L) P(L | I, D, T_{t-1}). \]

(5.2)

Note that \( S \) and \( A \) follow from \( O = 1 \) (the target is given), and that we consider \( S \) and \( A \) as given and hence omit them for brevity.

5.2.1 Prediction

First, we may interpret \( P(L | I, D, T_{t-1}) \), the term of the right side in Eq. (5.2), as a saliency map. This can be rewritten as

\[ P(L | I, D, T_{t-1}) \propto P(I, D | L, T_{t-1}) P(L | I, T_{t-1}) \]

\[ = P(I | L) P(D | L) P(L | T_{t-1}), \]

(5.3)

treating \( I \) and \( D \) independently given \( L \). We will call \( P(I | L) = P(I | L, A) \), the appearance-based saliency likelihood. A detail of the appearance model \( A \) will be discussed in later section. \( P(I | L) \) is simply computed from the likelihood from the appearance-based classifier. \( X_a \) in the graph can be interpreted as the location mapping of features onto \( I \).

We call the term \( P(D | L) P(L | T_{t-1}) \) in Eq. (5.3) motion-shape-based saliency. Let us use the notation \( x \) and \( R \) to represent the 3D location and orientation of the object respectively (\( T_{t-1} = \{ R, x \} \)), \( \tilde{D}(L) \) to represent the 3D coordinate of the observed 2D point at location \( L \), and a covariance matrix \( C = C(S) \) for an approximate distribution of the 3D shape of the object. We model the motion-shape-based saliency as a Mahalanobis distance map: Instead of using a single Gaussian, we may consider a Gaussian Mixture distribution if the volume is hard to approximate as an ellipse.
5.2.2 Update via Shape Registration

Next, finding the 3D motion given 2D location, $P(T_t|L)$ of Eq. (5.2), and marginalizing over all $L$ is a hard problem in general. Instead of directly computing marginalization, we guess a best $L_*$ assuming that $P(T_t|L)$ is very small for other possible $L$’s. That is,

$$
\hat{T}_t = \arg \max_{T_t} P(T_t|I, D, T_{t-1})
$$

$$
= \arg \max_{T_t} \sum_{L} P(T_t|L) P(L|I, D, T_{t-1})
$$

$$
\approx \arg \max_{T_t} P(T_t|L_*) P(L_*|I, D, T_{t-1}).
$$

(5.4)
We may choose $L_*$ to be for a threshold probability $p$,

$$L_* = \{y | P(y|I,D,T_{t-1}) > p\}. \quad (5.5)$$

Once $L_*$ is fixed, the term $P(L_*/I,D,T_{t-1})$ does not affect the maximizing argument. Therefore Eq. (5.4) can be further reduced to,

$$\hat{T}_t \approx \text{arg max}_{T_t} P(T_t|L_*). \quad (5.6)$$

Now, the interpretation is that given the best mask prediction $L_*$ we want to fine the best 3D pose that explains the mask. Then we further approximate the problem to finding a local solution of $\hat{T}_t$ by shape registration between the shape model, $S$, and the 3D coordinates of $L_*$. Let $Y$ be the 3D coordinates for points in $L_*$, which can be computed from the given image coordinate and depth values. Then Eq. (5.6) is practically solved as,

$$\hat{T} = \text{arg min}_T \sum_i d(S_i, T(Y_i)), \quad (5.7)$$

in an Expectation-Maximization (EM) fashion, where $d(\cdot,\cdot)$ represents a distance and $i$ indicates the correspondence between points in $S$ and points in $Y$. This is equivalent to ICP, and we use a variant of the point-to-plane algorithm [110] based on a projected distance. Note that even though solving $P(T_t|L_*)$ implies independence from $D$ and $T_{t-1}$, its computation is implemented as ICP which practically requires using $T_{t-1}$ and $Y$, the depth values corresponding to $L_*$. 

The predicted pose by the motion filter (Sec. 5.3.2) is used as the initial values for ICP. Although ICP could fail as any locally-optimal algorithm might (unless it has a global strategy as in [199]), it is still one of the most popular accurate methods to align two point sets with reasonable initial poses.

After having observed a new 3D pose, we can update the posterior of the motion state. Also, each point is checked if it is an inlier of the registered shape. Then the classifiers
Figure 5.4: Classifiers Implemented Using Neural Nets. (a) Multi-scale features are combined to form a single higher dimensional feature and fed into a single layer classifier. (c) The activation from each feature map is weighted-summed, while the weights being learned as well. Note that upsampling is applied to meet consistency of the dimension of the visual field.

discriminatively learn from these self-supervised inlier ($O = 1$) and outlier ($O = 0$) samples (Fig. 5.3 (b)). The learning models will be described in more detail in the following section. If we have more than one targets, then the prediction map will have more channels for the additional targets and ICP will be performed for each target object.

5.3 Online Update

Now that we have reviewed the high-level probabilistic framework of the suggested method, the model of each module used in our implementation is described in this section.

5.3.1 Appearance Classifier

In order to have a set of hard-wired features that capture visual characteristics of different scale and/or different dimensions, we chose to test the pretrained convolutional layers. It is known that the CNN structure is inspired by the biological structure. Yet, it is much less known how the information from the lower visual areas is combined to produce the whole visual perceptual experience. Among many possible implementations to integrate the features, we report a few chosen variations after our private preliminary tests. First, we fix the number of features layers to three (Details will be described in Sec. 5.4), and investigate two different implementations of classifier (Fig. 5.4). Second, we vary the sampling method
of examples (Note: sampling is necessary because there are too many training example points.), uniform sampling versus weighted sampling by physical adjacency.

**Single Classifier in Higher Dimension**

One possible way to combine different groups of features is augmentation of them, treating them independently. This allows us to obtain a high-dimensional vector, which is powerful in discriminative classification tasks. A similar idea of concatenation can be found in many studies, for example [65]. We use the resulting high-dimensional input with a $1 \times 1$ convolution followed by a logistic regression node per pixel (See Fig. 5.4 (a)). This can be extended to a soft-max layer for multi-target cases. We apply backpropagation only to the classifier layer (not to the feature layers) based on squared loss with respect to the labels supervised by shape registration.

**Sum of Multiple Classifiers**

Another idea to combine features of different depths of CNN will be having skip layers and a summing layer as in [109]. However, the naive average of predictions from three classifiers
would be less expressive due to the constraints of 2D spatial correlation within the same layer. In order to add some expressive nonlinearity with this model, we test a classifier that learns the weights of a weighted sum of prediction based on each skip layer. (Fig. 5.4 (b)).

### 5.3.2 Motion Filter

A linear Kalman filter (KF) [82] based on a first-order kinematic model with viscous damping is used for 3D translation estimation. The measurement of KF is the observed 3D location from ICP. The filter is designed to adaptively ignore an abrupt change of pose and only propagate according to the motion model. This includes when visibility of the target is less than 3% of the target size. In this study, we do not use a filter for orientation, but use direct solutions from ICP unless it fails.

### 5.4 Experiments

In order to evaluate the suggested method, we need 1) RGBD sequences of natural scenes containing target objects, 2) the 3D models of the objects, and ideally 3) a ground truth poses at each frame. There have not been many datasets that come with the ground truth pose of objects in dynamic scenes [50] since it is very hard to obtain the pose as discussed in our previous paper [100]. Two publicly available datasets [31, 94] are used for quantitative analysis. Furthermore, we have compiled 15+ real RGBD sequences accounting for over 10,000 frames, eight sequences of which includes ground truth segmentation and 3D poses.\(^1\) Fig. 5.5 shows the nine objects and samples of the scenes included in the data. The first six models are selected from [167] and [25], which is developed for object recognition and manipulation research. We added three models to have more diverse sizes and shapes. Ground truth poses are generated by hand-labeling the RGB images followed by ICP on the segmented point cloud.

Four versions of our implementations of the framework are reported in this paper\(^2\):

---

\(^1\)Available at: https://sites.google.com/site/bhoramlee/data

\(^2\)Having tested with three popular CNNs, AlexNet[93], VGGNet[166], and ResNet[67], we report the result from the AlexNet features due to the limited space. The others showed very similar performance, but it took longer time to pass the data through their nets.
- Single-layer NN Classifier (Fig. 5.4(a)) with uniform sampling (denoted as ‘Su’) or weighted sampling (‘Sw’)

- Weighted-sum NN Classifier (Fig. 5.4(b)) with uniform sampling (‘Wu’) or weighted sampling (‘Ww’)

The inputs of the NN classifiers are taken from the activation of conv1, conv2, and conv4 of AlexNet, and normalized before being passed to the classifier. L2-norm regularization is used for gradient descent, and fixed learning rates are used. The implementation of the algorithm is written in MATLAB and we used MatConvNet [6] for extracting CNN features. Computation times ranged from 0.15 to 0.3 seconds per object per frame when tested on a PC with an Intel Core i7 2.6GHz processor. This implementation would be easily parallelized for future real-time applications.

Table 5.1: RMSE of Estimates on Choi’s Dataset [31]

<table>
<thead>
<tr>
<th></th>
<th>Tide</th>
<th>Milk</th>
<th>Juice</th>
<th>Box</th>
<th>Tot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trans (nm)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C&lt;sub&gt;800&lt;/sub&gt;[31]</td>
<td>4.16</td>
<td>4.13</td>
<td>3.86</td>
<td>6.93</td>
<td>4.93</td>
</tr>
<tr>
<td>C&lt;sub&gt;12k&lt;/sub&gt;[31]</td>
<td>2.00</td>
<td>2.41</td>
<td>2.09</td>
<td>3.20</td>
<td>2.47</td>
</tr>
<tr>
<td>Krull [94]</td>
<td>1.34</td>
<td>1.51</td>
<td>1.10</td>
<td>2.03</td>
<td>1.53</td>
</tr>
<tr>
<td>Our-Su</td>
<td>.081</td>
<td>.034</td>
<td>.040</td>
<td>.15</td>
<td>.089</td>
</tr>
<tr>
<td>Our-Sw</td>
<td>.062</td>
<td>.028</td>
<td>.034</td>
<td>.14</td>
<td>.080</td>
</tr>
<tr>
<td>Our-Wu</td>
<td>.053</td>
<td>.025</td>
<td>.038</td>
<td>.11</td>
<td>.065</td>
</tr>
<tr>
<td><strong>Our-Ww</strong></td>
<td><strong>.048</strong></td>
<td><strong>.023</strong></td>
<td><strong>.033</strong></td>
<td><strong>.10</strong></td>
<td><strong>.060</strong></td>
</tr>
<tr>
<td>Rot (deg)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C&lt;sub&gt;800&lt;/sub&gt;[31]</td>
<td>4.84</td>
<td>10.63</td>
<td>4.05</td>
<td>12.0</td>
<td>8.6</td>
</tr>
<tr>
<td>C&lt;sub&gt;12k&lt;/sub&gt;[31]</td>
<td>2.37</td>
<td>5.22</td>
<td>2.06</td>
<td>9.03</td>
<td>5.45</td>
</tr>
<tr>
<td>Krull [94]</td>
<td>2.82</td>
<td>3.18</td>
<td>2.06</td>
<td>1.62</td>
<td>2.49</td>
</tr>
<tr>
<td>Our-Su</td>
<td>.15</td>
<td>.035</td>
<td>.046</td>
<td>.040</td>
<td>.082</td>
</tr>
<tr>
<td>Our-Sw</td>
<td>.11</td>
<td>.028</td>
<td>.039</td>
<td>.032</td>
<td>.063</td>
</tr>
<tr>
<td>Our-Wu</td>
<td>.098</td>
<td>.027</td>
<td>.045</td>
<td>.029</td>
<td>.058</td>
</tr>
<tr>
<td><strong>Our-Ww</strong></td>
<td><strong>.089</strong></td>
<td><strong>.023</strong></td>
<td><strong>.039</strong></td>
<td><strong>.025</strong></td>
<td><strong>.052</strong></td>
</tr>
</tbody>
</table>

5.4.1 Quantitative Evaluation

Two existing datasets from [31] and [94] are used for quantitative analysis of tracking and pose estimation performance of our method. Both papers studied 3D model-based tracking and provided their data as well as evaluation results. We cite their numbers from their paper for comparison. Choi’s dataset [31] is noise-free synthetic RGBD sequences of four objects
with ground truth pose that generated the scenes. Krull’s dataset [94] has six real RGBD sequences of three objects with ground truth obtained by human segmentation followed by ICP. The RGB images of Krull’s dataset are readily registered to depth images, so both RGB and depth images have ‘holes’ in the images. We simply regarded these holes as noise and tested our method on the noisy input for the evaluation without any problem.

As shown in Table 1, our result is very close to the ground truth pose of the synthetic dataset: translation error less than 0.1 mm and rotation error is less than 0.1 deg. Our near-perfect result is due to 1) the high quality of predicted saliency, and 2) the direct optimization (ICP) on the selected data over the pose space. On the other hand, the other two algorithms [31] and [94] primarily rely on a finite number of hypotheses of 6DOF pose. The number of particles used to obtain Choi’s best result in Table 1 is 12800. It is reported that Krull’s method uses a smaller number of particles (70 to 120), but it requires an intense pre-training session for both the target and background [22].

Compared to graphically synthesized scenes, real RGBD scenes of Krull’s dataset has challenging factors. The objects were moved by human hands generating occlusion with contact on the surface. The level of occlusion by other non-target objects ranged from mild to severe. Table 2 shows that the variations of our method achieved stable success rates, all cases above 95%. We use the same definition of the success rate used in [94] and [22]: The pose is ‘successfully’ estimated if the average distance of model point correspondences of estimate and ground truth is less than 10% of the diameter of the model. (This metric is used in [71] as well.) Note that our method does not need any pretraining whereas [94] and [22] do.

Table 5.2: Success Rate on Krull’s Dataset [94]

<table>
<thead>
<tr>
<th></th>
<th>Cat1</th>
<th>Cat2</th>
<th>Sam1</th>
<th>Sam2</th>
<th>Box1</th>
<th>Box2</th>
<th>Tot</th>
</tr>
</thead>
<tbody>
<tr>
<td>B [22]</td>
<td>66.6</td>
<td>44.2</td>
<td>72</td>
<td>33.7</td>
<td>54.7</td>
<td>59.4</td>
<td>58.9</td>
</tr>
<tr>
<td>K [94]</td>
<td>100</td>
<td>99.4</td>
<td>96.3</td>
<td>92.3</td>
<td>88.8</td>
<td>100</td>
<td>96.2</td>
</tr>
<tr>
<td>Our-Su</td>
<td>100</td>
<td>100</td>
<td>99.0</td>
<td>99.2</td>
<td>98.2</td>
<td>100</td>
<td>99.4</td>
</tr>
<tr>
<td>Our-Sw</td>
<td>100</td>
<td>100</td>
<td>97.8</td>
<td>99.3</td>
<td>98.2</td>
<td>100</td>
<td>99.2</td>
</tr>
<tr>
<td>Our-Wu</td>
<td>100</td>
<td>100</td>
<td>99.2</td>
<td>97.5</td>
<td>98.2</td>
<td>100</td>
<td>99.1</td>
</tr>
<tr>
<td>Our-Ww</td>
<td>100</td>
<td>100</td>
<td>96.9</td>
<td>98.2</td>
<td>98.2</td>
<td>100</td>
<td>98.8</td>
</tr>
</tbody>
</table>
We further investigated how the performance index varies with different values of the success threshold. As shown in Fig. 5.6, the average success rate achieved is higher than 95% for all our four implementations even when the success threshold was set as low as 5% of the diameter of the model. We observed that our method had more noisy segmentation in the Cat sequences than in others, because of deformations made by touching. In spite of that, it exhibited a robust overall tracking performance. The Samurai sequences were hardest in that they include some frames of as lower visibility as 3%, which was considered as a loss of the target in our current tracking implementation.

Now we provide quantitative results on a subset of our dataset for which we have ground truth. Whereas all Krull's image sequences were taken in the same place with mostly flat background, our sequences are taken in various places—kitchen, bedroom, laundry room, office, and so on—with many distractors including identical objects (See Fig. 5.5). All sequences of our dataset start from an initial scene where objects are placed on a supporting plane, allowing a table-based initial detection pipeline. Our initialization is also implemented based on plane segmentation followed by clustering object candidate point clouds and then pose search of a given target using its shape model.

Fig. 5.7 shows the average tracking success rate and the average 2D segmentation success rate for the four implementations of our framework. Surprisingly, the performance of the variations did not show significant differences. Although we see that the single-layer classifier performed slightly better than the two-layer weighted-sum implementation, we may conclude that our framework is proven to be effective for both types of classifier. Some interesting qualitative observations include that (1) the \(-w\) versions (weighted-sampling) showed a tendency of overfitting in cases where no adjacent distractor exists or under poor visibility, (2) prediction maps of the \(-u\) versions (uniform sampling) are shown discriminant enough around the target regions in most cases, resulting in reasonable final prediction maps (See the second row of Fig. 5.9).

Fig. 5.8 shows the results on individual sequences of our dataset using the \(Ww\) version of our method. Overall our method achieves very high success rates for tight thresholds for
most of the objects. The sequences with ‘banana’, plotted in the lower part in the figure, was the hardest in many aspects: the object is small, textureless, thin, and long. As it is known, such shapes tend to be less stable in ICP.

5.4.2 Discussion

Under good visibility and few distractors, our method gave reasonable prediction maps, and the pose tracking accuracy was high. Even when there are distractors in terms of both spatial adjacency and similar (or identical) appearance, it managed to remove the distractors. Our method not only resulted in robust shape registration but also gave final prediction maps that are sensible to the eye as shown in Fig. 5.9. We were unable to test other 3D pose tracking approaches on our dataset, because either we could not find the codes or they require intensive training.

With no additional strategy applied for recovery from complete occlusion, we have observed some failure in tracking (due to failure of ICP) under prolonged severe occlusions or large translations in other sequences. However, we remain focused on demonstrating the feasibility of this online learning appearance framework, instead of trying to fix the inherent problem of ICP. Using the registration score for online self-evaluation, or incorporating appearance for 3D pose prediction may ameliorate this issue in a future study.

One practical consideration may be the need of a shape prior that we hold in this work. However, some robotics applications readily require known shapes, and an extension of our framework to simultaneously build a shape model and track the target is ripe for future study.

Although we held some conditions fixed in order to keep our focus on the high-level idea, it will be interesting to explore principled methods to make the modules adjust the parameters throughout the online pipeline. Our framework may be extended to non-rigid objects, which can be more useful for various applications yet more challenging. It will be also interesting to use the proposed method in other applications such as precise manipulation of objects.
Figure 5.6: Tracking Performance on Krull’s dataset: (Left) Average pose tracking success rate for the four implementations of our method. (Right) Pose tracking success rate of individual sequences by one of our implementation, ‘Ww’.

Figure 5.7: Average Performance on Our Dataset: (Left) Pose tracking success rate and (Right) 2D target segmentation success rate. (Note: IOU is short for Intersection of Union. Best viewed in color.)

Figure 5.8: Performance Evaluation of ‘Ww’ on Individual Sequences. Larger objects of distinct shape show higher success rates. Other versions also have a similar tendency in graphs.
Figure 5.9: Highlight of Our Video Demonstration: Among the first four rows, the point clouds are displayed at the top, being overlapped with model point clouds in blue. The appearance-based prediction (using ‘Su’) and spatial attention map are showed in the second and the third row respectively. The predicted target region is highlighted in color on gray-scale images in the fourth row. The other sequences are shown with point cloud and the final prediction visualizations.
Part II

ONLINE CONTINUOUS MAPPING
Whereas Part I presents a learning framework that utilizes shape priors, Part II considers the shape representation itself in depth. In studying vision as an information-processing task, the representation of information is as important as how the information is extracted and processed [116]. There have been various suggested ways to incorporate local geometric properties and uncertainties into spatial representations. For example, a simple parametric surface model is used in Ch.4 and the point cloud representation is adopted to model 3D rigid object shapes in Ch.5. There is no single best representation for every application, and we still lack rich, efficient ways to express complex geometric information for robotic tasks.

In Ch.6, we will review some of the representations found in robotics literature and highlight more studies using occupancy and signed distance fields (SDF). After clarifying the definition of SDF, Gaussian Processes (GP) and Gaussian Process Implicit Surfaces (GPIS), a study on online update for spatial mapping using GPIS is proposed in Ch.7.
Chapter 6

Background and Related Work

6.1 Structure Representations

There exist a wide variety of approaches and techniques for geometric modeling and shape representation, and the particular choice usually depends on the application [16, 99, 203]. More emphasis on geometric structure can be found in robotics literature whereas the global description is also of great importance in visual recognition problems. Acknowledging the large body of work in this topic, we attend to metric representations that encode the geometry of the environment in robotics applications.

The most generic representation of structure boundaries would be point sets. Point cloud has been widely used, representing the surface of an object directly observed from a depth sensor or a LIDAR scanner. Since a point set is merely samples of the underlying surface for which it actually represents, it is often augmented with surface normals and accompanied with a notion of uncertainties or neighborhood. These ideas lead to other representations such as Gaussian Mixture Models (GMM) [46, 80], Normal Distribution Transform (NDT) [19], and Surfels [142]. GMM and NDT view the point measurements as samples from the underlying probability distribution representing the surface. NDT limits the scope of each distribution within a cell of regular grids. A surfel-based map is an unordered set of surfels, each of which is a disk defined by a point, a normal, and a radius. Although the definition itself does not involve probabilistic modeling, an additional variable may be attached to each
surfel to denote the uncertainty. Another popular method to represent surfaces is polygon mesh, which consists of points and edges that connect the points in a systematic way. By forming connected closed polygons, it is possible to represent seamless, directed surfaces of a closed volume when using mesh.

On the other hand, there are volumetric methods to explicitly model empty space or solid interior region together with the boundaries. Occupancy Grid (OG) [74, 125, 179], one of the most successful methods in robotics, represents the world as discretized cells, each of which holds a binary random variable to indicate its occupied or free state. Since each cell is treated independently in OG mapping, an update or access of the occupancy value of the cells can be very fast.

Signed Distance Functions (SDF) (See Sec. 6.3 and Fig. 6.3) also have been often employed as a mapping representation ([11, 24, 192]). It is an implicit representation of surfaces as a level-set of a function and the exterior or interior regions are expressed as the signed-distance to the boundary. Since its first introduction in [133], Level-set methods have been used as a tool for numerical analysis of surfaces and shapes. In particular, this approach is frequently used in computer vision and computer graphics applications including early shape modeling studies [37, 112] and more recent rendering techniques [62]. Although SDF is mathematically defined as a continuous function, it is often implemented in a grid structure.

### 6.2 Related Work: Continuous Representations

While having experienced considerable success with grid-based representations, there have been studies exploring spatial representations without regularity or discretization. In par-
ticular, the idea of using lines for robotic mapping and/or localization has been around since 1980’s [36, 143, 185]. These studies presented continuous representations of 2D structure boundaries as a set of line segments from laser range scans. This kind of representation compacts the size of the map since it does not model unexplored region or empty space but only the boundaries. However, the drawbacks of this approach include inaccuracy caused by the approximation of the world as a set of flat surfaces, non-trivial 3D extension, and lack of proper probabilistic frameworks. Spline-based representations such as [141] can overcome the limitation in shape, but still share the other drawbacks of line-based representations.

Non-Grid Occupancy Map

The work presented in [137] by Paskin et al. used a non-parametric method called polygon random field to probabilistically model occupancy rather than the boundaries. Although new and interesting, this approach has a significant disadvantage of computation complexity required to get the random fields to converge as noted by the authors. A recent study on non-regular representation by Fridovich-Keil et al. [53] suggested using a collection of non-overlapping sphere elements, which they call “Atoms”, to represent occupancy. Unlike most representations that are in the position domain, Schaefer et al. [163] suggest storing the map parameters in the discrete frequency domain, which allows the continuous and differentiable expression in the position domain via the inverse discrete cosine transform.

Recently, there have been studies on continuous occupancy representations, which uses the same concept of ‘occupancy’ but without discretization of the world. O’Callaghan et al. [135] pointed out that in OG mapping structural correlations between nearby cells are ignored and we often observe artifacts. They explored the idea of building continuous occupancy maps using GP that encode the correlation of observed points. They used the resulting predictive mean and variance distributions to classify regions of the robot’s surroundings into areas of occupancy or free space. Later, as an effort to avoid the cubic complexity of GP, Kim et al. [87, 89] suggested employing multiple local GPs with systematic spatial partitioning. Ramos et al. [148] also addressed the issue and presented kernel approximation methods as parametric alternatives. An efficient data handling method for
GP occupancy map is presented in [188].

Although the continuity in occupancy representation eliminates discretization artifacts and introduces dependency between points, it has some mathematical difficulties. It is not proper to model a random variable that is actually expected to jump from 0 (empty) to 1 (occupied) at the boundary as a continuous function. As a result, there must appear uncertain bands near the boundaries that continuously transition from 0 to 1. Moreover, by forcing the function to be continuous, the predicted values can be bigger than 1 or less than 0, and thus they lose the exact meaning of the intended occupancy random variable any longer (See Fig. 6.2).

**Continuous SDF**

On the other hand, the SDF representation is mathematically defined as a continuous function as will be shown in Sec. 6.3. Although the values of a SDF are often stored in a discrete grid for fast lookup, there have been studies that treated continuous SDF. In particular, Gaussian process implicit surfaces (GPIS) is a non-parametric regression model for an implicit SDF that provides the variance for every prediction [193].

Since the distance field and its gradients can be utilized in planning for collision checks and trajectory optimization [131, 153], the robotics community has paid attention to GP representations. Dragiev et al. [44] utilized GPIS as shape representations for robotic
grasping. They viewed the GPIS representation as a good choice for the application because it allows uncertain sensory fusion in a probabilistic way and it serves as a basis for efficient grasp and motion generation. Hollinger et al. [73] suggested a reconstruction method for closed 3D meshes from acoustic range data, and used GPIS to represent uncertainty on the meshes for underwater inspection applications. They developed a path planner for minimizing uncertainty based on the GP representation. Gerardo-Castro et al. [57] presented data fusion of laser and radar for shape representation within a GPIS framework. Kim et al. [90] and Martens et al. [117] explored using non-zero mean functions for a more complete shape representation. Also, as will be discussed in 7.4, when given the distance field, the Iterative Closest Point (ICP) algorithm [18] for surface registration can be performed without an expensive nearest neighbor search [11, 147].

While previous studies lack the notion of building GPIS from online measurements, this dissertation contributes a method that allows incrementally building GPIS. This is presented throughout Ch. 7.

Other Representations using GP

Applications of GP-based representation in robotics are not limited to occupancy map or SDF. GP has been used for terrain mapping [98, 184] to make continuous predictions on the terrain shape with uncertainty. Plagemann et al. [145] also took this approach for motion planning of legged robots. In order to avoid the limited expression of vertical terrain profile, [168] suggested a GP regression on the range measurement in a parametrized sensor-centric configuration space.

Our method also adopts another GP regression on local observation data in addition to the global map. The locally defined GP regressor is used to infer surface point locations and normals (See 7.3.2). A related work to this is found in [144], where the authors introduced a GP regression for range measurement and applied the method on Monte-Carlo localization and tracking problems.
6.3 Signed Distance Function (SDF)

We will mathematically define SDF and describe its properties in this section. Let us consider a volumetric object $\Omega$ in a metric space and its boundary $\partial \Omega$. We call a function $f : \mathbb{R}^D \rightarrow \mathbb{R}$ a signed distance function (SDF) if the function takes a point $x \in \mathbb{R}^D$ and outputs the distance of the point to the boundary of the target object, that is, $d(x, \partial \Omega)$. A 2D illustration is shown in Fig. 6.3. The surface points are implicitly defined as the zero-level set of the signed distance function, and the internal points are denoted as negative values. Formally, SDF of an object $\Omega$ can be written as,

$$
    f(x) = \begin{cases} 
        +d(x, \partial \Omega) & \text{outside surface} \\
        0 & \text{on surface} \\
        -d(x, \partial \Omega) & \text{inside surface},
    \end{cases}
$$

(6.1)

where $d$ is the distance of the point from the surface. The zero-level set defined as,

$$
    X_{zero} = \{x : f(x) = 0\},
$$

(6.2)

represents all the points on the boundary. Also the Note that, distance is a global property, which means that the distance values can be correctly determined only given the global structure.

![Figure 6.3: Illustration of Signed Distance Function of 2D Object.](image)
6.4 Gaussian Process Implicit Surfaces with Derivatives

Now let us consider using a GP regressor to approximate the SDF near the surface of a structure. We will begin with a general description of standard GP following the definitions and notations from [150]. Then the particular case to represent SDF with its derivatives will be discussed [152][169]. A full and rich description of GP can be found in [149].

6.4.1 Gaussian Processes

A GP is completely specified by its mean function and covariance function. We define the mean function \( m(x) \) and the covariance function \( k(x, x') \) of a real process \( f(x) \) as,

\[
m(x) = \mathbb{E}[f(x)]
\]

\[
k(x, x') = \mathbb{E}[(f(x) - m(x))(f(x') - m(x'))],
\]

and will write the Gaussian process as

\[
f(x) \sim \mathcal{GP}(m(x), k(x, x')).
\]

A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. This implies that if we have the GP that specifies \((y_1, y_2) \sim \mathcal{N}(\mu, \Sigma)\), then it must also specify \( y_2 \sim \mathcal{N}(\mu_2, \Sigma_{22}) \) where \( \Sigma_{22} \) is the relevant submatrix of \( \Sigma \). By this implication, we may define a GP with a set of training data (corresponding to \( y_1 \)) to make inference on new data (corresponding to \( y_2 \)). The details follow.

Assume we have a set of data \( \{(x_i, y_i) | i = 1, ..., n\} \), where \( y \) denotes a scalar output. We further assume the noise of the observed values \( y \) follows an i.i.d. Gaussian distribution, \( \epsilon_y \sim \mathcal{N}(0, \sigma_y^2) \). The prior on the noisy observations becomes,

\[
\text{cov}(y_p, y_q) = k(x_p, x_q) + \sigma_y^2 \delta_{pq},
\]

where \( \delta_{pq} \) is a Kronecker delta which is one iff \( p = q \) and zero otherwise. Using the notation \( K(X, X) \) to denote the square matrix of the covariances evaluated at all pairs of \( x_i \)'s, it can
be written as,
\[
\text{cov}(y) = K(X, X) + \sigma_y^2 I, \quad (6.6)
\]

Given a set of test points \( x_* \), we can write the joint distribution of the observed target values and the function values at the test locations \( (f_* := f(x_*)) \) under the prior as,
\[
\begin{bmatrix}
y \\
f_*
\end{bmatrix}
\sim \mathcal{N}
\begin{pmatrix}
K(X, X) + \sigma_y^2 I & K(X, X_*) \\
K(X_*, X) & K(X_*, X_*)
\end{pmatrix}
(6.7)
\]

Now we can derive the conditional distribution of \( f_* \mid X, y, X_* \sim \mathcal{N}(\bar{f}_*, \text{cov}(f_*)), \) with the mean,
\[
\bar{f}_* \triangleq \mathbb{E}[f_* \mid X, y, X_*] = K(X_*, X) \left[ K(X, X) + \sigma_y^2 I \right]^{-1} y,
(6.8)
\]

and the covariance,
\[
\text{cov}(f_*) = K(X_*, X_*) - K(X_*, X) \left[ K(X, X) + \sigma_y^2 I \right]^{-1} K(X, X_*).
(6.9)
\]

This is the key predictive equations for GP regression.

Note that the covariance function is the crucial ingredient of the equations. Only a function that satisfies the positive semi-definiteness can be a valid covariance function. Let us give some examples of covariance functions [151]. The squared exponential covariance function is one of the most commonly used covariance functions:
\[
k_{SE}(r) = \exp \left( -\frac{r^2}{2l^2} \right)
(6.10)
\]
where \( r \triangleq ||x_1 - x_2|| \).

The Matérn class of covariance functions is given by
\[
k_{\text{Matern}}(r) = \frac{2^{1-\nu}}{\Gamma(\nu)} \left( \frac{\sqrt{2\nu r}}{l} \right)^\nu K_\nu \left( \frac{\sqrt{2\nu r}}{l} \right),
(6.11)
\]
where \( \nu \) and \( l \) are positive parameters and \( K_\nu \) is a modified Bessel function \(|\). The Matérn
covariance function has a simpler form when \( \nu = p + 1/2 \), where \( p \) is a non-negative integer. For \( \nu = 1/2 \), it gives the exponential covariance function \( k(r) = \exp(-r/l) \), for which the process becomes very rough.

\[
k_{\nu=1/2}(r) = \exp\left(-\frac{r}{l}\right). \tag{6.12}
\]

For \( \nu = 3/2 \) and \( \nu = 5/2 \),

\[
k_{\nu=3/2}(r) = \left(1 + \frac{\sqrt{3}r}{l}\right) \exp\left(-\frac{\sqrt{3}r}{l}\right) \tag{6.13}
\]

\[
k_{\nu=5/2}(r) = \left(1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2}\right) \exp\left(-\frac{\sqrt{5}r}{l}\right) \tag{6.14}
\]

The choice of covariance function is important for many reasons. It specifies the covariance between pairs of random variables, defining a sort of similarity between data points. The values of covariance functions like Eq. (6.10)-(6.14) do not exactly zero even for very large \( r \). On the other hand, there exist covariance functions with compact support, which means that the covariance matrix can become sparse by construction [121]. Also, for some covariance functions such as Eq. (6.12), the \( 2k \)th-order partial derivative \( \partial^{2k} k(x)/\partial^2 x_{i_1} \ldots \partial^2 x_{i_k} \) may not exist. This is related to the existence of the \( k \)-th order partial derivative \( \partial^k f(x)/\partial x_{i_1} \ldots \partial x_{i_k} \), which is discussed in the following section.

### 6.4.2 Gaussian Process with Derivative Observations

Since differentiation is a linear operator, the derivative of a GP is another GP [152]. Thus we can use GPs to make predictions about derivatives, and also to make inference based on derivative information. In general, we can make inference based on the joint Gaussian distribution of function values and partial derivatives. A covariance function \( k(\cdot, \cdot) \) on function values implies the following (mixed) covariance between function values and partial derivatives, and between partial derivatives,
\[
\text{cov}(f_i, \frac{\partial f_j}{\partial x_{d_j}}) = \frac{\partial k(x_i, x_j)}{\partial x_{d_j}}
\]  
(6.15)

and

\[
\text{cov}(\frac{\partial f_i}{\partial x_{d_i}}, \frac{\partial f_j}{\partial x_{e_j}}) = \frac{\partial k(x_i, x_j)}{\partial x_{d_i} \partial x_{e_j}}
\]  
(6.16)

With \( n \) data points in \( D \) dimensions, the complete covariance of \( f \) and its \( D \) partial derivatives involves \( n(D + 1) \) quantities. In a typical application we may only have access to a subset of them. In that case, we simply remove the rows and columns from the joint matrix which are not needed.
Chapter 7

Online Mapping using GPIS

An accurate and efficient representation of spatial structures is crucial for successful planning and control in navigation or manipulation tasks. We explore a new method to express the structure of the environments continuously with an underlying assumption that there exist continuous surfaces of volumetric objects.

Currently the point cloud representation and the grid-based occupancy representation are widely employed [179] in robotics. Gaussian Processes (GPs) have been suggested as an alternative [44, 135] as discussed in 6.2. Unlike grid-based methods, GP does not depend upon a particular choice of coordinates while providing a compact representation of continuous functions. They can continuously interpolate surface structures and can be nat-

Figure 7.1: (a) Continuous Mapping vs. (b) Grid-based Mapping
urally extended to incorporate environmental priors and derivatives as well. Specifically, GP can be used as a regression to a distance function and build a map as the zero level set of the function. In this way, the map is represented as an implicit function, for which the GP regression maintains a probabilistic estimate of the SDF to objects in the environment (Fig. 7.1). The representation itself includes distance and its gradient, which are critical information for online obstacle avoidance and trajectory planning.

7.1 Contributions

- While prior research has focused on modeling the GP without much discussion on an online update, our study contributes a principled way to update GPIS models incrementally with noisy sensor measurements. This is necessary for a robot to continually plan while navigating in an uncertain environment.

- The update process we present modifies the existing map by comparing it with new observations in a Bayesian update scheme. We introduce a GP regressor on measurements and infer corresponding surface points and normals continuously. The measurement noise model considered is heteroscedastic and accounts for noise resulting from the geometric relationship between the surface and the observer’s viewpoint.

- Our GPIS implementation consists of multiple small GPs that are updated and tested locally to mitigate the notorious complexity of the standard GP model. A spatial partitioning data structure is exploited to manage this data efficiently.

- We experimentally demonstrate the performance of our method in terms of accuracy for mapping and localization. It is evaluated in comparison with other grid-based representations including occupancy grids and discrete SDFs.
7.2 Models

This section presents our GPIS representation as an approximate SDF, which is based on Sec. 6.4 with subtle differences from the standard GP model. Our approach is specifically based on the noisy input GP model [119], where the input $\tilde{x}$ of GP is corrupted by a Gaussian noise $\epsilon_x \sim \mathcal{N}(0, \Sigma_x)$, $\text{e.i.}$, $\tilde{x} = x + \epsilon_x$. Then the observed output can be written as, $y = f(\tilde{x} - \epsilon_x) + \epsilon_y$, where $\epsilon_y \sim \mathcal{N}(0, \sigma_y^2)$ is a noise to the output variable. Then, when its first order Taylor expansion is considered, the probability of an observation, $y$, can be expressed as,

$$P(y|f) = \mathcal{N}(f, \sigma_y^2 + \partial_f^\top \Sigma_x \partial_f), \quad (7.1)$$

where $\partial_f$ is denoted as the derivative of the function with respect to the input, which corresponds to the surface normal in our problem. $\Sigma_x$ can be modeled as an isotropic Gaussian covariance with parameter $\sigma_x$, assuming surface observation noise is not directional. Also the derivatives satisfy $||\partial_f||_2 = 1$ by definition of surface normal. Then, Eq.(7.1) can be rewritten as,

$$P(y|f) = \mathcal{N}(f, \sigma_y^2 + \sigma_x^2). \quad (7.2)$$

While the standard GP considers the output noise only, having non-zero the input noise allows us to incorporate the priors of measurement noise to the surface update. The noise may be further modeled as another process as in [144]. However, in order to render the problem tractable, we assume that the noise is independent. Thus, the noise covariance of $n$ input points can be written as $\Sigma_x = \text{diag}\{\sigma_{x_1}^2, \sigma_{x_2}^2, ..., \sigma_{x_n}^2\}$. Then the inference equations for the mean $\hat{f}_*$ and the variance $\mathbb{V}[f_*]$ of the function value given a test point $x_*$ can be summarized as,

$$\hat{f}_* = k_*^\top (K + \Sigma_x)^{-1}y,$$

$$\mathbb{V}[f_*] = k(x_*, x_*) - k_*^\top (K + \Sigma_x)^{-1}k_*.$$  

(7.3)

Note that we follow the simpler notations $k_*$, $K$, and $k(x_*, x_*)$ to indicate the same as
Eq.(6.8) and Eq.(6.9). These terms represent the vector of covariances between \( \mathbf{x}_* \) and \( n \) training points, the \( n \times n \) covariance matrix of the training points, and the covariance function value of \( \mathbf{x}_* \) respectively. \( \mathbf{y} \) is the target vector of the training points.

Now let us consider available training data for the function \( f \). While surface points are observed, we do not explicitly observe exemplars of off-surface or internal points. Some methods employ control points of the two types to express the directional information of the surface ([57]). In this study, we consider jointly modeling the SDF and the derivatives to encode the surface direction as in [44, 117]. In our implementation, a Matérn class covariance function with \( \nu = 3/2 \), Eq.(6.13), is used with its first-order derivatives.

### 7.3 Updates

Now we will describe the main idea of the suggested incremental update method for GPIS. Given a set of training surface points of GPIS and new observations, if the correspondences are known then the surface points can be updated in a Bayesian fashion. However, it is hard to find corresponding points directly from discrete measurements. We will show how to bypass the correspondence problem by using observation GP regressors, and describe the noise models used for the update. Note that all descriptions and illustrations are restricted to 2D for clarity, but can be easily extended to 3D as shown in Sec. 7.6. Fig. 7.2 illustrates what our method maintains and returns while being updated online.

In Fig. 7.2 (a), the arrow indicates the pose of the robot, and the colored patches and points visualize the range measurement at time \( t_1, t_2, \) and \( t_3 \). The gray points are the accumulated range measurements over time, the dotted black line shows the trajectory of the robot. Images in (b) and (c) show the boxed area within the bold dashed line. The method updates the training surface points and normals of GPIS with incremental observations (b). The resulting continuous SDF is visualized in a yellow-blue colormap, and its sampled surfaces in black. In (b), the gradient fields of SDF are visualized in a red-green colormap.
Figure 7.2: The suggested online GPIS update
7.3.1 Bayesian Update of Surface Points

Let us begin with the higher-level idea of the update. Consider a surface point \( x_k^- \) from the GPIS model and assume a new observation of that surface point as \( \tilde{x}_k \). Supposing that both the model point and observed point are Gaussian random variables with respective variances of \( \sigma_{x_k}^2 \) and \( \tilde{\sigma}_{x_k}^2 \). The surface point in the GPIS can be updated to have the distribution \( x_k \sim N(x_k^+, \sigma_{x_k}^+) \) with,

\[
x_k^+ = \frac{\sigma_{x_k}^0 x_k^- + \sigma_{x_k}^2 \tilde{x}_k}{\sigma_{x_k}^2 + \sigma_{x_k}^2}
\]

(7.4)

\[
(\sigma_{x_k}^2)^{-1} = (\sigma_{x_k}^2)^{-1} + (\sigma_{x_k}^2)^{-1}
\]

The GPIS model also contains surface normals which are updated separately from the points: a normal vector update is treated as weighted rotation instead of the arithmetic weighted average. After updating the input points and normals, we may update the GP inference parameters for prediction. However, implementing this idea involves technical issues, which will be discussed in the following sections.

7.3.2 Observation GP

The Bayesian update is simple when we correspond a measurement \( \tilde{x}_k \) with a point \( x_k \) in the GPIS model. However, this is not usually obvious. Instead of matching a measurement directly, we use a GP regressor on range measurements to infer \( \tilde{x}_k \) and its surface normal.

Let \( z_i \) be the range measurement at angle \( \theta_i \) in a local polar coordinate frame (Fig. 7.3 (a)). Then the GP regressor is the function (Fig. 7.3 (b)),

\[
f_z : \theta \rightarrow z^{-1}.
\]

(7.5)

Note that \( z^{-1} \) is used since it nicely scales from 0 to the inverse of the maximum limit range which is a finite positive value. This GP continuously predicts the range values at any given valid \( \theta \) (Fig. 7.5 (a)). The Ornsten-Uhlenbeck covariance function (Eq.(6.12)) is used because it expresses detailed curves without excessive smoothing.
Now, for a test point $x_*$ (or equivalently $(\theta_*, z_*^{-1})$ in polar coordinates), if $x_*$ is visible or free then $z_*^{-1} > \tilde{f}_z(\theta^*)$. If it is unobservable then $z_*^{-1} < \tilde{f}_z(\theta^*)$. We may soften this occupancy test by taking a logistic function along the ray as,

$$occ(z_*^{-1}, \theta_*) = 1 - \frac{2}{1 + \exp\{-a(z_*^{-1} - \tilde{f}_z(\theta^*))\}}.$$ \hspace{1cm} (7.6)

where $a$ is a slope parameter. An example of this occupancy test using Eq.(7.6) is visualized in Fig. 7.5 (b). The angular ranges with very high variances are considered as being unobservable or invalid.

### 7.3.3 Surface Point and Normal Inference

With this continuous local representation of the occupancy, a corresponding measurement point can be numerically inferred (Fig. 7.5(c)) as the following. Given an old point $x_k^-$ with normal $n_k^-.$

1. Test its occupancy with Eq.(7.6).

2. If the absolute value of occupancy is larger than a small number $\epsilon$, and

   (a) if the value is negative (beyond 'surface') then move a step along the direction of $n_k^-$, or

---

**Figure 7.3:** (a) Range observation in the local Cartesian coordinate. (b) GP regression in the polar coordinate for the range observation (Eq.(7.5)).
(b) if the value is positive ('free') then move a step along the opposite direction of $n_k$.

3. Otherwise, stop sampling points. Take the current location with a small occupancy value as $\tilde{x}_k$.

We may iterate this step to obtain a new point very close to the surface.

Next, the gradients of a measurement point $\tilde{x}_k$ can be inferred by computing (Fig. 7.5(d)),

$$\frac{\partial \tilde{x}_k}{\partial v} \approx \frac{\text{occ}(\tilde{x}_k + \delta e_v) - \text{occ}(\tilde{x}_k - \delta e_v)}{2\delta}$$

(7.7)

where $v$ indicates an axis of the $D$-dimensional Cartesian space and $e_v$ is the unit vector along the axis. Also $\delta$ is a small positive value that represents a reasonable neighborhood range.

Then the normalized vector of the gradients is used as the surface normal $\partial f$. The absolute gradient values are determined by $a$ of Eq.(7.6) which is somewhat arbitrary. However, this is not problematic since we only need the directional information of the gradient near the surface. This way, the normal vector can be obtained at any surface point within the visible range.

![Figure 7.4](image)

Figure 7.4: (a) Occupancy inference using Eq.(7.6). (b) The same result shown in the local Cartesian coordinate.
7.3.4 Noise Models

As mentioned in Sec. 7.2, we treat a training surface point $x$ as a Gaussian variable. The noise variance is modeled with two additive terms,

$$\sigma_x^2 = \sigma_z^2 + \sigma_\phi^2. \quad (7.8)$$

The first term, $\sigma_z^2$, is a function of the range value at the time of measurement, and the second term, $\sigma_\phi^2$, is a function of the relative view angle ($\phi$) of the surface. Intuitively, a measurement will be less noisy when observed at a closer distance and orthogonal to the observer. One way to implement this model is to use $\sigma_z^2 = \alpha_z z^{-2}$ and $\sigma_\phi^2 = \alpha_\phi \tan(\phi)$, where $\alpha_z$ and $\alpha_\phi$ are parameters.

The surface normal requires a separate noise model from the point itself. Although the noise for each axis is not independent because of the constraint $|\partial f| = 1$, we assume a spherical Gaussian noise for simplicity. We model the uncertainty $\sigma_\theta$ considering that the normals are numerically obtained. In an ideal case, the mean occupancy values of the neighbor points, $\sum_v \text{occ}(x_* + \delta e_v) + \sum_v \text{occ}(x_* - \delta e_v)$ would be exactly zero. However, if for some reason the neighbor is not representing the local surface properly, it will have some non-zero values. We use this mean value to indicate how much the surface normal computation can be trusted. This gives a reasonable range of absolute value of $\sigma_\theta^2$ which is the variance of surface normal vectors.

![Figure 7.5: Illustration of Surface Point and Normal Inference (Sec. 7.3.3): After (a) The range regression and (b) occupancy test converted to Cartesian coordinates, (c) point location and (d) surface normal can be inferred.](image)
7.4 Localization with GPIS

When considering localization on the GPIS map representation, the Iterative Closest Point algorithm [18][28] is a natural choice. ICP is most well-known as a point set registration method. However, in a general framework [165], it also applies to surface-to-point or surface-to-surface registration. The GPIS representation readily provides the gradient field near the surface, along which a point can flow toward the surface without explicitly finding the closest point (Fig. 7.6). The related concept and formula can be found in other references [11, 147] as well, but we include the derivation of pose estimation solution for clarity.

We may define the problem as finding the transformation on points \( x_i \) that minimizes the squared sum of distance to the surface:

\[
R_*, t_* = \arg\min \sum_i ||f(Rx_i + t)||^2
\]

\[
= \arg\min \sum_i ||f(x_i + \delta x_i)||^2
\]  

(7.9)

Assuming small rotation, i.e. \( R \approx I + [\omega]_x \), we may write the incremental displacement of each point as,

\[
\delta x_i = (Rx_i + t) - x_i
\]

\[
= (R - I)x_i + t
\]

\[
\approx [\omega]_x x_i + t
\]  

(7.10)

Figure 7.6: Illustration of ICP without Closest Point Matching on SDF
Now let us approximate $f$ in Eq.(7.9) using the Taylor expansion up to the first order,

$$
f(x_i + \delta x_i) \approx f(x_i) + \nabla f_{x_i} \delta x_i
$$

$$= f(x_i) + \begin{bmatrix} \nabla f_{x_i} & \nabla f_{x_i} \times x_i \end{bmatrix} \begin{bmatrix} t \\ \omega \end{bmatrix}
$$

(7.11)

Plugging it back into Eq.(7.9), we have

$$\xi^* = \arg\min \sum_i \left( f(x_i)^2 + 2f(x_i)b_i^\top \xi + \xi^\top b_i b_i^\top \xi \right)
$$

$$= \arg\min \left( 2 \sum_i f(x_i)b_i^\top \right) \xi + \xi^\top \left( \sum_i b_i b_i^\top \right) \xi
$$

(7.12)

where $b_i \triangleq \begin{bmatrix} \Delta f_{x_i} & \Delta f_{x_i} \times x_i \end{bmatrix}^\top$ and $\xi \triangleq \begin{bmatrix} t \\ \omega \end{bmatrix}^\top$. The solution for this quadratic problem should satisfy the optimality condition $\partial(\cdot)/\partial \xi = 0$, which reduces the problem to computing the following least square solution:

$$\xi = - \left( \sum_i b_i b_i^\top \right)^{-1} \left( \sum_i f(x_i)b_i^\top \right)
$$

(7.13)

Note that we use the notation $\xi^* = A \backslash B$ to imply the solution of the linear system $A\xi = B$. The signed distance values $f$ and the gradients $\Delta f$ are inferred from the GPIS map. Since the GPIS map representation includes the variance of prediction, we may consider a weighted version of the Least-Square solution with the weight $w_i$ being a function of the variance. In that case, Eq.(7.13) can be modified as,

$$\xi = - \left( \sum_i w_i b_i b_i^\top \right)^{-1} \left( \sum_i w_i f(x_i)b_i^\top \right)
$$

(7.14)

To summarize, the localization process can be summarized as the following:

1. Set the initial locations of the measurement points $\{x_i\}$ based on the current pose estimate of the robot.
2. Solve for $\xi_*$ using Eq.(7.14).
   
   (a) If the norm of $\xi_*$ is smaller than some threshold, stop the iteration. The accumulated transformation will be the solution.
   
   (b) Otherwise, update the locations of $x_i$ and the current pose according to $\xi_*$ and repeat the process.

7.5 Implementation

The foremost concern in using GP would be its computational complexity. Given $N$ points, computing the inverse covariance matrix in a standard GP regression model takes time $O(N^3)$ and making inferences for $M$ points takes time $O(MN^2)$. There have been various approximation approaches for this issue [149], and we chose a method similar to the spatial partitioning described in [89, 90]. The basic idea is to divide points into clusters and train each cluster locally. This reduces the time for computing an inverse to $O(KN_k^3)$ and testing to $O(MKN_k^2)$ where $K$ is the number of clusters and $N_k \ll N$ is the largest number of points per cluster. Note that in this approach, each cluster must overlap with its neighbors to prevent discontinuity.

The practical issue of this idea is then how to divide space and to what extent the clusters
should overlap. We used the QuadTree [49] and OcTree [120] to partition and access the surface points, as well as perform a range search for clustering efficiently. Specifically, we set the minimum resolution of the leaf node \( r^0 \), the maximum limit of map size \( r^{l_{\text{max}}} \), the intervals of local GP clusters \( r^{l_c} > r^0 \) (Fig. 7.7). Note that the radius of each cluster \( R_c \) must satisfy \( R_c > r^{l_c} \) to allow overlapped areas with neighboring clusters. When there are changed points, any local GPs within the range \( R_c \) from the points are re-trained. In order to make updating efficient, the algorithm keeps track of cluster nodes that are affected by inserting or removing points, which we call active clusters.

The overall online update process is summarized on the right in Fig. 7.7. When new range observations are received, we first compute a GP regression on them (Step 1). Given the current pose, we may query potentially visible clusters within the maximum range measurement value, and reject those that are beyond visible angular ranges. If an existing surface point is considered visible, the measurement regression is used to find the correspondence of the point, as well as to infer surface normals (Step 2). After moving existing surface points, test if newly observed points can be inserted into the map (Step 3). Only the GPs in the active set are re-computed (Step 4). The practical computation of Eq.(7.3) involves,

\[
L := \text{Chol}(K + K_x) \quad \text{and} \quad \alpha := L^\top \backslash (L \backslash y). \quad (7.15)
\]

When testing a point, a range search is performed to find nearby local GPs first. Then, the
predictive mean and variance (Eq.(7.3)) of the point can be computed as,

\[
\bar{f}_* = \mathbf{k}_*^\top \mathbf{\alpha} \quad \text{and} \quad \nabla[f_*] = L \mathbf{k}_*.
\]  

(7.16)
on the \( n \) closest GPs. Depending on the confidence level, our method chooses the best inference if the variance is below a certain value, or takes a weighted sum otherwise. This preserves the continuity of the map.

### 7.6 Evaluation

In this section, we demonstrate the characteristics and performance of the suggested method. We evaluated the accuracy in structure prediction and localization on 2D simulations. Some of the visual results and computational times on 2D and 3D real data are also reported. All programs are written in Matlab and C++ and all computations are performed on a CPU of a laptop (Intel i7-6700HQ @ 2.60GHz). The simulation data of range measurements and ground truth poses were obtained in Gazebo using a Hokuyo range sensor model on a Turtlebot with a Gaussian noise \( \sigma = 0.01m \). The angular range of the lidar sensor was \(-135^\circ\) to \(+135^\circ\) and the sensing resolution was \(1^\circ\).

#### 7.6.1 Mapping

We compared our method with OG mapping [10], Hilbert maps [148], and TSDF [37]. The three methods are chosen to have all combinations of discrete/continuous and occupancy/SDF representations. The simulated environment (Fig. 7.9) was made to duplicate the one from previous studies [88, 135, 148] with a slight rotation to avoid any artifacts caused by axis-aligned data. The trajectory was generated in the way that the accumulated raw measurements were dense enough to cover the structure as seen in Fig. 7.2 (a). The ground truth SDF of the environment is shown in Fig. 7.9 (a) and the trajectory used for mapping is shown in Fig. 7.9 (b). The distance values of the 1D dotted cross-section is visualized under the map. The four other graphs Fig. 7.9 (c)-(f) display the results of the compared methods in the form of either occupancy or SDF. While (b) and (c) model

\footnote{The source code and the parameter settings can be found at https://github.com/leebhoram/GPisMap}
Figure 7.9: Visual Comparison of Surface Modeling in 2D
the structure using the occupancy representation, (d) and (e,Ours) model the surface using
the SDF representation. On the other hand, (b) and (d) are grid-based implementations
whereas (c) and (e,Ours) are continuous functions. The red solid lines on the 2D maps
are the surface (or occupied) parts of the structure. The occupancy/distance values of the
dotted section are visualized under the maps. The gray lines in (d) and (e) are the ground
truth scaled properly for comparisons. Note that the colormap scales vary according to the
range of each result.

The first two gray-scale maps are occupancy maps. The OG map in Fig. 7.9 (c) has
a grid resolution of 0.1m and cells that have probability values above a threshold (0.65,
default by [10]) are considered occupied and are marked in red. Although the measurements
completely covered the structure, a close look into the map reveals missing surface cells on
the bottom left side. This is due to the fact that those areas were only observed from afar for
a short period of time so that the cells could not accumulate enough measurements to trust
its occupancy. This occurs frequently in OG maps. Fig. 7.9 (d) shows the Hilbert map using
the Nyström method, for which we set Kernel width $\gamma = 2.5$, number of features = 3000,
and default values by the authors for other parameters. As shown in the 1D visualization,
this method expresses probabilities which ranges between 0 to 1 as a smooth function. As
a result, its predictions for surfaces come as wide regions and it is hard to obtain a precise
surface (curve) in the same way as done in Fig. 7.9 (c). Instead, we visualized the boundary
pixels of empty space with the threshold 0.55 to match with other methods.

The next two maps are SDFs. Fig. 7.9 (d) shows the TSDF map of resolution 0.1m. In
the figure, cells with zero weights are plotted in white. Our test adopted the moving average
with the unit weight as done in [11] for simplicity. The surface estimate as the zero-level
isocontour is shown in red. Often the distance update is restricted to the vicinity of the
hit points, but we did not restrict it so that we could better observe the SDF values. The
obtained distance values tend to be overestimated as shown in the 1D plot. This is because
distances are taken along the the line of sight to the sensor rather than the closest distance
to the hit point. Lastly, the SDF and surface estimates by GPIS is shown in Fig. 7.9 (f).
Figure 7.10: Surface Prediction Accuracy vs. Reduced Data Rate. The left image shows the reduced data rate of 1m interval and the colored patches on the graphs represent minimum/maximum errors for each case.

The minimum resolution $r$ was 0.1 and the scale parameter of the covariance function was 1.2. The map is masked with an alpha (transparency) layer of the variance values, and thus regions of high variances appear white. The variance is visualized as the green envelop in the 1D profile plot. It gives high uncertainty in free space since our implementation does not explicitly model those free regions. However, the predicted signed-distances at the vicinity of the structure are close to actual values with high confidence.

The accuracy of surface predictions are evaluated with reduced data rates. The Hilbert map was omitted since its result was not comparable without devising a method to precisely extract the surface. The original data used in Fig. 7.9 was collected roughly every 1cm while the robot traveled 37m. We skipped regular numbers of frames to simulate longer sampling intervals (Fig. 7.10). For instance, in the ‘1m’ case, the robot is assumed to travel 1m before it receives new measurement. Often, in practical situations, only glimpses of parts of the surrounding are available for a robot due to high-speed motion, occlusions, or degraded bandwidth. The experiment simulated such a condition, in which the quantity of observation is degraded. The metric, the Hausdorff distance ($d_H$), is considered. $d_H$ measures an adversarial case of errors and becomes large when there are consecutive missing points or large deviations. Overall, the accuracy of TSDF and GPIS was higher than OG map because TSDF achieves the sub-grid precision by interpolation and GPIS deals with continuous values. With lesser data, $d_H$ becomes larger with larger variances in the case of
OGM and TSDF. There were higher chances to miss each cell and $d_H$ was largely affected by the luck on which specific data frames are received. It should be noted that the surface prediction for GPIS-SDF is done by testing on a regular grid of the same resolution with other maps and followed by the iso-contour extraction. When identifying where the maximum distance occurs, the major error source of GPIS-SDF was found to be the corner shaped
areas whereas for TSDF and OGM the errors were disconnected walls. While the grid-based methods suffered a lack of repeated observations, the suggested GPIS showed robust and consistent results for sparse observations.

**Computation Time**

The computation time per frame for 2D cases including real data usually took less than 20 ms with a multi-thread implementation of Step 4 (Fig. 7.12 (a)). In order to observe the computational complexity with respect to the map size, we generated a large 2D map by increasingly tiling the Intel Lab map of Fig. 7.19 which is roughly $1,000 \text{m}^2$. The parameters used to obtain the result were as follows: the minimum resolution $r = 0.1$, the cluster interval $r_l^c = 0.4$, and the cluster radius $R_c = 2r_l^c$. Fig. 7.12 (b) shows the resulting median update time for completing each tile. The majority of the time was spent in Step 2 and 4. The testing time per point did not vary much as the map grows, ranging from 4 to 6 \(\mu\text{sec}\). Of course, the absolute computing time will vary according to parameter settings and the complexity of scenes. However, this logarithmic tendency demonstrates the potential scalability of our method. The computation time for our current 3D implementation took longer, about 0.5 to 1.5 second per frame depending on setting. In 3D, the dimension of the covariance matrix is larger in addition to a larger number of points per local GP. Having said that, the reported numbers are based on serial computation of the standard Cholesky decomposition. A potential parallel Cholesky factorization [56, 59] and a more systematic multi-threaded implementation could further speed up the process.
7.6.2 Localization

We evaluated the localization method described in Sec. 7.4 on the suggested GPIS map in comparison with the same localization method on discrete SDF map, and a Monte-Carlo (MC) localization on OG map using a particle filter (PF) [180]. In order to evaluate the localization performance in isolation from any defects of the map, another Gazebo simulation is used. This time, the simulated environment was made to have a similar complexity of the Intel Lab data [9] and the trajectory of the robot was also controlled to follow the trajectory of the same data. Other settings such as sensor noise and angular range or resolution are the same to 7.6.1.

The following describes the compared methods and conditions:

First, we tested the PF localization on the ground truth OG map (‘PF’). As a measurement model, the likelihood field sensor model [181] is adopted for its good performance and convenience. The likelihood field is pre-computed from the OG map of the resolution 5cm, 10cm and 20cm, which are shown in Fig.7.14. Note that this sensor model in effect renders the map a Gaussian-blurred version of the structure. We implemented PF with some assumptions favorable to the method. First, the map is assumed be known perfectly and the
Figure 7.14: Pre-Computed Likelihood Field on the OG Map (Used for ‘PF’)

Figure 7.15: Ground Truth Discrete SDF with Extracted Surfaces (Used for ‘ICP-D’)

Figure 7.16: GPIS map with Extracted Surfaces (Used for ‘ICP-G’)
unexplored area is assumed to be known inaccessible, which is true. Thus when a particle passes through the inaccessible regions, it is removed from the set of hypotheses. Second, it is assumed that the computational efficiency is not primary concern and resampling is performed every step. The number of particles for PF was varied from $5k$, $10k$ to $20k$. In general, the pose estimation improves as the number increases.

Second, the ICP-based localization method is tested on ground truth discrete SDF (‘ICP-D’). We chose to use the ground truth SDF since TSDF mapping results may vary according to specific implementations and could bias the localization performance. The SDF was generated from the ground truth binary (occupancy) grid map of the resolution $20cm$, $10cm$ and $5cm$ (Fig. 7.15). In the figure, the extracted surface points are visualized in red, and the signed distance values are color coded in an arbitrary scale. The distance values are obtained by bilateral interpolation for ICP to achieve sub-grid accuracy.

Lastly, the same ICP-based localization method is tested on GPIS maps (‘ICP-G’). The GPIS maps are generated as suggested in Sec. 7.3, fully implementing our method by updating the map points in the continuous domain. We varied the resolution of leaf nodes from $5cm$ to $50cm$. The numbers of input points used for mapping are, $14.3k$ for $5cm$ leaf, $6.8k$ for $10cm$, $3k$ for $20cm$, $2.3k$ for $25cm$, and $1k$ for $50cm$.

In our localization tests, no odometry is involved but a simple KF is used for prediction of the next pose. Also, since ICP is a local registration, the best solution is found among multiple perturbed hypotheses. We tested with 10 to 100 hypotheses and the results with 25 are reported since it was the smallest number that gives a reliable variance of the results. As for the criteria for ICP estimates, the mean absolute distance,

$$\frac{1}{N} \sum_{i=1}^{N} |f(x_i)|,$$

is used for ICP-D, and the mean of absolute expected distance multiplied by its variance,

$$\frac{1}{N} \sum_{i=1}^{N} \mathbb{V}[f(x_i)] |f(x_i)|,$$
The compared localization performance is shown in Fig. 7.17. The position error of PF and ICP-D increases as the grid unit increases. This tendency is expected from the limited resolution of discretized map cells. The angular error of ICP-D has the similar tendency mostly dependent on the map resolution, whereas the angular error of PF seems more dependent on the sampling resolution rather than the map resolution. When the number of samples of PFs increased from 1k to 10k, the median angular error decreased roughly in half. However, in the lowest resolution map (25 cm grid unit), this improvement became insignificant both in position and angular errors.
We can observe that the errors of ICP-G is much smaller than the others and the increasing tendency is very mild. Since the values for ICP-G are not noticeable in the comparison figure, a close-up plot for ICP-G is shown in Fig. 7.18 for clarity. The position accuracy of our method differs almost in one order from other methods with any map resolution. The only competitive method in terms of angular error is ICP on discrete SDF, from which we can conclude that using the local gradient is a powerful pose estimation method. The superior performance of our method, ICP on GPIS, over other methods seems due to the combination of the continuous expression of the map information and using the distance gradient field.

One interesting observation in our method is that the performance did not vary much for node limits 5 to 25 cm. It seems to be related to the structure’s scale and details: the extracted surfaces of the first two map images in Fig. 7.16 look almost the same even though the first map includes 10k more input points. On the other hand, in an extreme case of node limit 50cm (Fig. 7.16 last image), we can see that thin walls are not well captured since the thickness was beyond the limit of leaf nodes. Accordingly, the performance for this limit is not as good as other cases with finer node limits. As we will discuss in Sec. xx, a hierarchical implementation of our suggested GPIS could increase both efficiency and accuracy of structure representations.

7.6.3 Experiments on Real Data

Unlike simulation with nicely arranged surfaces and simpler sensor models, real world data includes uncertain pose estimates, structures of various sizes, non-Gaussian sensor noise. Although the scope of this study does not cover all the related topics, we tested our mapping method on public datasets and some results are presented. Although the results are mostly limited to qualitative observation without true information to compare with, they provide a sense of how the suggested method works on real data.
2D Lidar Mapping

We tested our method using real lidar data from two SLAM datasets, the Intel Lab dataset and the Freiburg Building dataset, which are available from [9]. The Intel Lab data includes raw lidar measurement of 180° angular range with 1° resolution. It consists of about 1000 frames while visiting each room on the floor. The Freiburg Building dataset has the same angular range with 0.5° resolution and about 4800 frames. The dataset [9] also provides pose estimates based on a SLAM algorithm called RBPF (Rao-Blackwellization Particle Filter). We used the estimates for the mapping experiments, which results are shown in Fig. 7.19. The maps of the figure are built with the node length limit of 0.1 m. Other parameter settings are the same to the previous experiment in Sec. 7.6.1, which also can be found in
the released code\(^2\). Although the details beyond the particular resolution are limited, we can clearly see the surfaces of the walls for the rooms and hallways. From our observation of the mapping process, the errors of pose estimate were the main cause for the mapping result not as clear as the simulation presented in Fig. 7.16.

**3D Object Surface Modeling**

As mentioned, the suggested method can be easily applied to 3D in principle, and we present an application of 3D Object Surface Modeling using real data. The images in Fig.7.20 show object surface modeling results using depth images and ground truth poses from YCB [25] and BigBIRD [167] datasets. In the figure, the zero-level iso-surfaces are visualized as a mesh and the SDF values of selected planes are colored in an arbitrary scale. In order to build the models, depth images from forty different viewpoints including five different elevation angles are used. The resolution of the model was chosen as \( r = 0.01 \text{m} \) to reflect the scale of objects. The target object area of each depth image was segmented using the corresponding mask provided in the datasets. We further pre-processed the masks using morphological erosion techniques since the masks were still noisy. Note that the representation requires sampling for visualization because it contains the surface implicitly. We simply tested regular grid points and used iso-level interpolation for the visualization in Fig.7.20.

When compared with the Poisson meshes that the datasets provide, the mean deviation of our surface modeling results was \(<2.5\text{mm}\). The surface points are sampled in the resolution of 5\text{mm}. Considering the discretized comparison, the modeled surface is very much close to the ground truth mesh. The visualized surface points in Fig. 7.21 have 5\text{mm} resolution for ‘detergent’ and ‘drill’, and 2.5\text{mm} for ‘mustard’ due to its small size. The number of sampled points (black points) are about 5k, 3k, and 8k respectively. whereas the ground truth point cloud (blue points) has 20k, 14k, and 8k.

The computation time for our current 3D implementation took longer, about 0.5 to 1.5 second per frame depending on settings as well as the number of observed points. As mentioned, in 3D, the dimension of the covariance matrix is larger in addition to a larger
number of points per local GP. In addition, the number of observation points from depth image ($O(10^4)$) is much larger than the 1D scanning lidar used in the previous experiments ($180$). What limited the 3D experiment more was the computing time for visualization. The naive grid-based test required testing points of $O(10^4)$ to $(O)(10^5)$ for decent visualization. The test time per point was about $8.0 \times 10^{-5}$ sec at the final frame in the experiment shown in Fig.7.20. The reported time is based on serial computation of the standard Cholesky decomposition in multi-thread implementations.
Figure 7.21: Sampled Surface Points (black) vs. Point Cloud from Poisson Mesh of the Datasets (blue).
Chapter 8

Conclusion

This dissertation presented new online frameworks for robotic visual and spatial perception that allow robots to gather rich information of the environment as they are deployed in the real world.

In Part I, we presented a self-supervised appearance learning framework that utilizes geometric priors without manual-labeling by humans. Specifically, we have shown that the varying appearance of targets can be segmented from the background by comparing current observation with the known geometric priors. The idea is implemented and demonstrated for two types of visual objects, the ground and 3D rigid-form objects, which have been of significant importance in robotics research. The online ground segmentation method was applied to monocular vision based upon previous self-supervised approaches using multi-modal sensors. The further extension to 3D segmentation and tracking provides a more generalized framework for free-form objects in motion. As a 3D pose tracking method, our method outperformed other methods that require intense training or sampling. Nevertheless, the motivation of our study is not to compete with other studies but to provide them a principled way to obtain self-labeled data with little human intervention. This line of approach is valuable not only in terms of data scalability for using modern learning algorithms, but also because robots should be able to exploit their unique experiences for learning on the fly.
In Part II, we addressed the issue of geometric representation, explored an online method to represent structure continuously, and demonstrated the potential of the suggested method. While the widely employed point-based or grid-based representations have experienced great success, they also exhibit problems such as disconnected information between cells or points, discretization artifacts, and lack of means to interpret uncertainty. We considered the continuous, implicit representation of the signed distance function using Gaussian processes for an alternative method that overcomes all the mentioned problems. We suggested a method enabling the online use of the rich representation and demonstrated its potential as an accurate and efficient method to represent the geometry of the environment with experimental evidence. The online property of the method and its continuous nature are something that robotic systems inherently require.

The problems addressed in this dissertation are far from being solved. To conclude, the following sections highlight some of the limitations of the work, and present a vision for future research in the topic.

8.1 Limitations and Future Direction

Self-Supervised Learning

Our self-supervised learning framework provides a pragmatic approach to gather target information such as segmentation labels and 3D pose. The requirement for geometric priors in this framework may be seen as a limitation. While 3D data may not exit for random images from the web, it is more attainable in robotics applications. Unlike the past when obtaining 3D models were expensive, nowadays there exist low-cost off-the-shelf devices and solutions for 3D scanning and processing. For some industrial products, 3D models exist first even before physical instantiation. Thus, shape models can be an affordable resource compared to the cost of human labor for hand-labeling jobs.

As another limitation, the current implementation applies to certain types of objects, parametric surfaces and rigid objects. This limitation is in fact more related to the lack of definite solutions to deal with non-rigid geometry rather than the conceptual limitation of
our framework. At a higher level, this problem links to the motivation of second part of this dissertation, which is that we lack of good geometric representations.

One of the advantages of the suggested framework over other automated labeling methods is that it allows observation of targets under motion, and this can contribute beyond pose estimation study itself. For example, under certain controlled environments, it will be possible to observe the manipulator, whether it is a human or a robot, by identifying non-targets. Some interesting applications we can consider with this include learning for affordance or improved manipulation.

Other promising directions of research include an extension of the framework to non-rigid objects. Although not applicable to all kinds of objects, there exist several approaches to non-rigid representations available from the literature, such as regularization, warp-field estimation, or using a set of basis forms. Efforts toward this end will be beneficial for the robotics research community, because ultimately we want to have a self-supervised learning framework for any geometric objects.

Some simple ideas that immediately add more practicality to the current implementation are 1) to have a model acquisition step before it starts, 2) to have a re-initialization step for recovery from complete occlusions, and 3) coarse-to-fine registration for speed-up.

**Online GPIS**

As an initial study on an online method for GPIS, our work focused on demonstrating its potential as an efficient and accurate representation. A limitation of the current implementation is that a single scale parameter is used for the GP representation. Environments we observe may have geometric details of different scales. It is hard to model structures correctly with only one scale: It will be either redundant or too coarse depending on the complexity of structure. To improve this, the suggested GPIS representation can be further investigated under different settings in GP. Specific ideas for this include applying other covariance functions, involving learning the hyper-parameters for them, and implementing a hierarchical structure. The benefits of these research directions are that we will be able to represent the environment in a more efficient, rich, and flexible way.
A practical drawback in the current implementation of our GPIS method is the long computation time in the 3D case. Although this is partially due to the nature of 3D space and the complexity of GP inference computation, there is room for improvement. For example, parallelized computation of GP clusters will be one way to alleviate the computational complexity. A potential parallel Cholesky factorization [56, 59] and a more systematic multi-threaded implementation could further speed up the process. Approximation-based approaches [149] could reduce the complexity, although they also require computation for their own training. Also, surface sampling implemented as naive grid-based tests followed by iso-surface extraction can be improved. Current implementation employed no strategy for fast visualization, but the tree data structure with surface input points can be augmented with other information for the surface prediction step. As one example, instead of repeating tests for all given points from scratch, memorizing previously predicted surface or searching near surface input points will be computationally more efficient.

As for research directions, it will be promising to extend our method as an integrated SLAM algorithm. Although the scope of the dissertation did not cover the topic, our framework readily provides an online probabilistic update scheme and demonstrated that the GPIS representation is most suitable for the optimal registration. Hence, combining the suggested map update method with registration-based localization will be a natural future research direction. In addition to improving the representation, it will be promising to exploit the suggested GPIS representation for studies on planning or learning to plan. As reviewed in the previous sections, there have been previous work on planning utilizing the distance field and its gradients for collision checks and trajectory optimization. Such studies may take advantage of our method since it provides an online framework for the desirable representations. Finally, we have focused on a purely geometric aspect of the environment, setting aside issues related to appearance or semantics. It will be another interesting research direction to study how to combine appearance with the continuous geometric information to represent an object, which is largely an open question.
Bibliography


