Lifelong Robotic Object Perception

Alvaro Collet Romea

CMU-RI-TR-12-22

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

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

August 29, 2012

This research was supported in part by La Caixa fellowship, Caja Madrid fellowship, and NSF grant No. EEC-0540865.

Contents

List of Figures

List of Tables

Abstract

In this thesis, we study the topic of Lifelong Robotic Object Perception. We propose, as a long-term goal, a framework to recognize known objects and to discover unknown objects in the environment as the robot operates, for as long as the robot operates. We build the foundations for Lifelong Robotic Object Perception by focusing our study on the two critical components of this framework: 1) how to recognize and register known objects for robotic manipulation, and 2) how to automatically discover novel objects in the environment so that we can recognize them in the future.

Our work on Object Recognition and Pose Estimation addresses two main challenges in computer vision for robotics: robust performance in complex scenes, and low latency for real-time operation. We present MOPED, a framework for Multiple Object Pose Estimation and Detection that integrates single-image and multi-image object recognition and pose estimation in one optimized, robust, and scalable framework. We extend MOPED to leverage RGBD images using an adaptive image-depth fusion model based on maximum likelihood estimates. We incorporate this model to each stage of MOPED to achieve object recognition robust to imperfect depth data.

In Robotic Object Discovery, we address the challenges of scalability and robustness for long-term operation. As a first step towards Lifelong Robotic Object Perception, we aim to automatically process the raw video stream of an entire workday of a robotic agent to discover novel objects. The key to achieve this goal is to incorporate non-visual information—robotic metadata—in the discovery process. We encode the natural constraints and non-visual sensory information in service robotics to make long-term object discovery feasible. We introduce an optimized implementation, HerbDisc, that processes a video stream of 6 h 20 min of challenging human environments in under 19 min and discovers 206 novel objects.

We tailor our solutions to the sensing capabilities and requirements in service robotics, with the goal of enabling our service robot, HERB, to operate autonomously in human environments.

Acknowledgments

I would like to thank my advisors, Sidd Srinivasa and Martial Hebert, for their guidance and support. Thank you for giving me the freedom to explore the field and pushing me forward when I needed it. The robot HERB perceives objects thanks to you.

I would like to thank my committee members, Yaser Sheikh and David Lowe, for their invaluable comments and suggestions. I would also like to thank Fernando de la Torre for trusting me and giving me the opportunity to prove myself when I finished college, and Chris Atkeson for his help since I started my M.Sc. and throughout my PhD.

This research would not have been possible without my collaborators Manuel Martinez, David Fouhey, Bo Xiong, and Corina Gurau. Thank you for listening to my crazy ideas and making them work.

Learning to play Squash is one of the best decisions I ever made. I am grateful to Frank Pfenning, Sidd Srinivasa and the CMU Squash Team for the happy times on the courts.

I would like to thank my PR Lab buddies for their friendship, support, and endless discussions about anything and everything. Anca, your enthusiasm is contagious; do not lose it. Mehmet, your kindness and work ethic are unparalleled. Mike, the PR Lab could not work without you.

These long years at CMU would not have been the same without my RI buddies, my 2007 cube-mates Mike Furlong, Brian Becker, Dan Munoz, Santosh Divvala and Uma Nagarajan, and my Vision-aire friends. Keep up the good work.

I would like to thank Tomás Simón, Alberto Rodriguez, Nuria Jané and little Zoe for bridging the gap between life in Pittsburgh and in Barcelona. Tomás and Alberto, thank you for all these great years of lunches, dinners, movies, and fun.

To Georgina, thank you for always being there for me when I needed you, for so many years. I wish I could have visited you more often.

I would like to thank my family for their continued support and love. I think about you every day. Mom, Dad, I could not be writing this document without your help. To my brothers, Nacho and Alberto, the best times of my life are when we are together. There are no words to describe how grateful I am. Os quiero mucho.

Part I

Introduction

Chapter 1

Introduction

Object. [n. ob·ject. ŏb-′jĕkt]: *Something material that may be perceived by the senses.*
Merriam-Webster.com (2012)

Objects play an important role in our daily lives; we interact with hundreds of them every day—clothes, food items, keys, computers, etc. We take the concept of *object* for granted. But what, exactly, is an object? Merriam-Webster.com (2012) defines an object as "something material *that may be perceived by the senses*." Humans are particularly adroit at perceiving objects: we easily identify known and unknown objects around us, and instinctively understand how to interact with them to serve our purposes.

We want robots to interact with objects, too, and to work alongside us in human environments. We consider the case of a service robot in a home or office, operating autonomously. Most tasks in service robotics require interaction with objects, ranging from obstacle avoidance (e.g., the Roomba) to fetching drinks, unloading the dishwasher, or cooking a meal. The more complex the task, the better robots need to perceive the objects with which they interact. In this thesis, we explore how to make robots understand what objects are, so that they can perceive objects—known and unknown—as well as we do.

Object perception for robotics offers both opportunities and challenges unique to the robotics field. We can use additional sensing modalities and external information (robotic metadata) to leverage the natural constraints and structure in service robotics. Examples of metadata include robot localization, odometry, prior knowledge, and restrictions about the environment, robot, or task, among others. We use the term *robotic metadata*—or just *metadata*—very broadly in this work: visual information is the primary data source in computer vision, so we refer to *any* non-visual information as metadata. Using robotic metadata, we can tackle perception problems that would be too complex using visual information alone. After all, the data gathered by a service robot is not just an unordered collection of anonymous images: we may know where and/or when the images are captured, as well as their ordering; we may know where interesting objects usually appear for a particular environment, such as in tables or cabinets; we may have multiple sensing modalities (e.g., range measurements, localization, odometry) for some or all the images, or we may only be interested in objects of certain sizes or shapes relevant to the robot.

The extra challenges in service robotics derive from a trade-off between system latency and robustness. Service robots require fast perception, real-time or near real-time; latency of more than a few seconds in recognizing an object is rarely acceptable in on-line tasks. In addition, algorithms must be robust for safety reasons: mistakes in robotic perception have consequences ranging from broken mugs (in personal robotics) to human casualties (in autonomous cars).

In this thesis, we study the topic of Lifelong Robotic Object Perception. We propose, as a long-term goal, a framework to recognize known objects and discover unknown objects in the environment as the robot operates, for as long as the robot operates. In this framework (described in detail in Chapter 2), we recognize known objects and register their position so that the robot can interact with them. We automatically analyze sensor data to learn new information about the objects with which the robot interacts. We also automatically discover novel objects and feed their models back to the recognition component, and incorporate grasping feedback to validate and refine the discovered objects. We envision Lifelong Robotic Object Perception as a coupled process of learning, recognition and interaction with objects, in which information feeds back from each component to improve the overall system performance.

We build the foundations for Lifelong Robotic Object Perception by focusing our study on the two critical components of this framework: 1) how to recognize and register known objects for robotic manipulation, and 2) how to automatically discover novel objects in the environment so that we can recognize them in the future. We tailor our solutions to the sensing capabilities and requirements in service robotics, with the goal of enabling our service robot, HERB (Srinivasa et al., 2012), to operate autonomously in human environments.

1.1 Summary of Contributions

The main contribution in this thesis is a common formulation to represent visual information and metadata as graph constraints for object discovery, which we introduce in Section 2.3 and explore in depth in Chapter 7. In addition to this common formulation, we propose a number of key contributions for object recognition and object discovery. We provide a list of our key contributions in this section, and discuss each contribution in more detail in Chapter 2.

- A real-time object recognition system, MOPED (originally reported in Collet et al.

(2011a)), and its extension to images and depth, MOPED-RGBD (originally reported in Fouhey et al. (2012)). Our contributions in MOPED include:

   - Iterative Clustering-Estimation (ICE) to iteratively partition the scene and efficiently estimate object poses in complex scenes (Collet et al., 2011a).
   - Projection Clustering and M-estimator-based score to merge duplicate object detections (Collet et al., 2011a).
   - Scalable, low latency recognition via architectural optimizations and hybrid CPU/GPU computing to exploit parallelism (Collet et al., 2011a).
   - An image-depth fusion model based on maximum likelihood estimates for both available and missing depth data (Fouhey et al., 2012).

- A framework for Robotic Object Discovery (originally reported in Collet et al. (2012)), including:
   - An *objectness*-based scene segmentation for RGBD images, to compute object candidates for Object Discovery (Collet et al., 2011b).
   - A common formulation to represent visual information and robotic metadata as generic graph constraints for object discovery (Collet et al., 2012).
   - HerbDisc: an optimized system for Robotic Object Discovery based on graph constraints and use of robotic metadata (Collet et al., 2012).
   - A dataset of 6 h 20 min of RGBD video (and 521 234 RGBD images) of real offices and labs to evaluate HerbDisc, gathered with HERB (Collet et al., 2012).

1.2 Roadmap

Chapter 2 introduces the general framework for Lifelong Robotic Object Perception, which is our high-level guide for this thesis. We introduce the subproblems of Object Recognition and Pose Estimation in Section 2.2 and Robotic Object Discovery in Section 2.3. We study Object Recognition and Pose Estimation in Part II. In Chapter 4, we develop MOPED, a framework for Object Recognition and Pose Estimation. We extend Chapter 4 to robustly combine images and depth in Chapter 5, MOPED-RGBD. We study Robotic Object Discovery in Part III. In Chapter 6, we introduce our *objectness*-based scene segmentation algorithm to compute object candidates for Robotic Object Discovery. In Chapter 7, we introduce a unified graph-based formulation for Robotic Object Discovery and describe our optimized implementation, HerbDisc. In Part IV we conclude this thesis with a discussion of our contributions and future work.

1.3 Publication note

Earlier versions of our work on MOPED (Chapter 4) appear in Collet et al. (2009), Collet and Srinivasa (2010), Martinez et al. (2010), Collet et al. (2011a), and is joint work with Manuel Martinez. An initial version of our work on MOPED-RGBD (Chapter 5) was published in Fouhey et al. (2012) and is joint work with David Fouhey. An initial version of our work on scene segmentation (Chapter 6) was published in Collet et al. (2011b). Our work on Robotic Object Discovery and HerbDisc appears in (Collet et al., 2012) and is joint work with Bo Xiong and Corina Gurau.

Chapter 2

Lifelong Robotic Object Perception

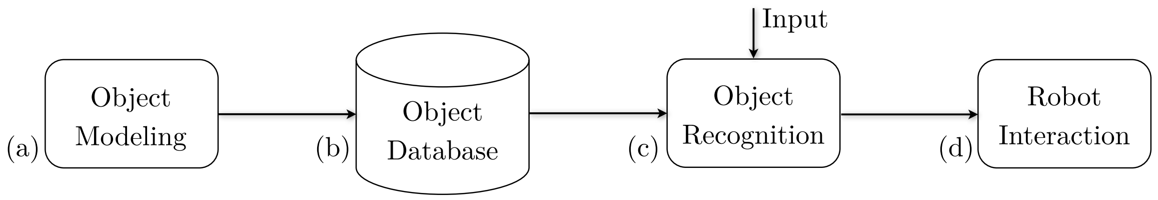
Perceive. [v. per·ceive. pər-ˈsēv]: *To become aware of through the senses.*
Merriam-Webster.com (2012)

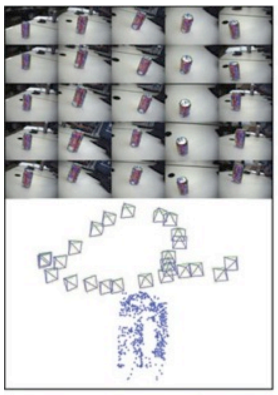
The typical object perception pipeline in a current state-of-the-art service robot (e.g., Srinivasa et al. (2012), WillowGarage (2008)) follows a similar implementation to the flowchart shown in Fig. 2.1. In Fig. 2.1, we identify four clearly separated components. First, an off-line training stage generates a set of object models, via supervised or semi-supervised methods (Fig. 2.1(a)). Each new object model created is stored in a model database (Fig. 2.1(b)). In the on-line stage, an object recognition algorithm identifies objects (from the database) in the sensor data and registers their position (Fig. 2.1(c)). Finally, the robot interacts with the identified objects if it is necessary for the task (Fig. 2.1(d)).

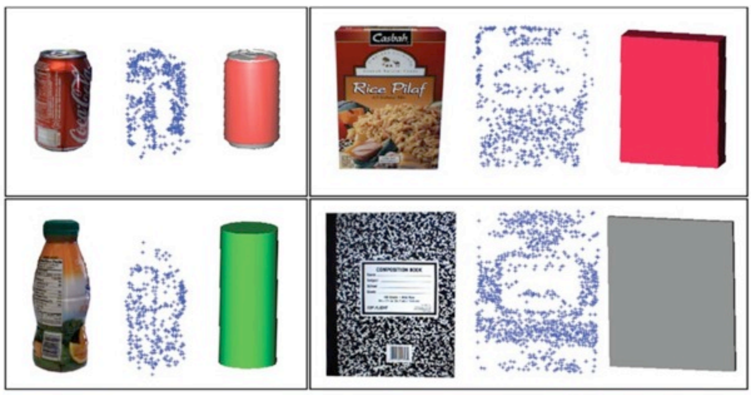
In the pipeline of Fig. 2.1, learning new objects and adding them to the database requires manual supervision. In addition, each component is isolated: there is no mechanism for object models to improve over time; the experience from robot interaction is not used for improvement; and the knowledge of the current set of objects does not help for the task of learning new objects. In contrast, the framework we propose defines a completely autonomous perception system that learns over time, both to learn new objects, and to improve the performance of each task. A global diagram of our proposed framework is given in Fig. 2.2, and discussed in Section 2.1.

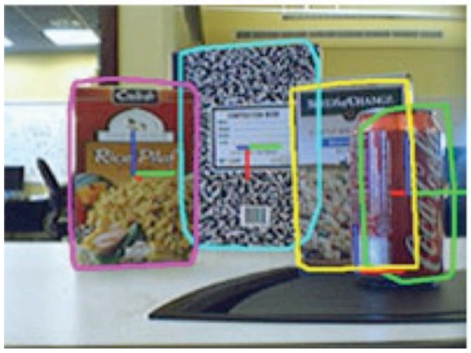
2.1 A Framework for Lifelong Robotic Object Perception

We propose the framework for Lifelong Robotic Object Perception shown in Fig. 2.2. Comparing this framework to Fig. 2.1, the main differences are the introduction of a Robotic



(a)


(b)


(c)


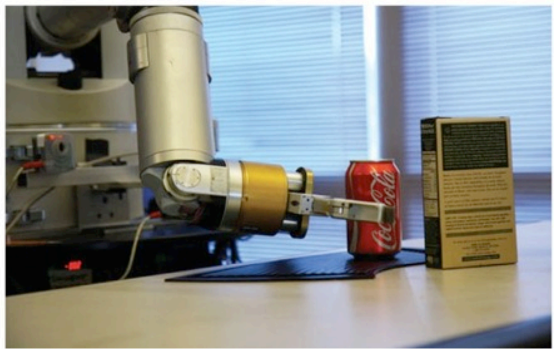
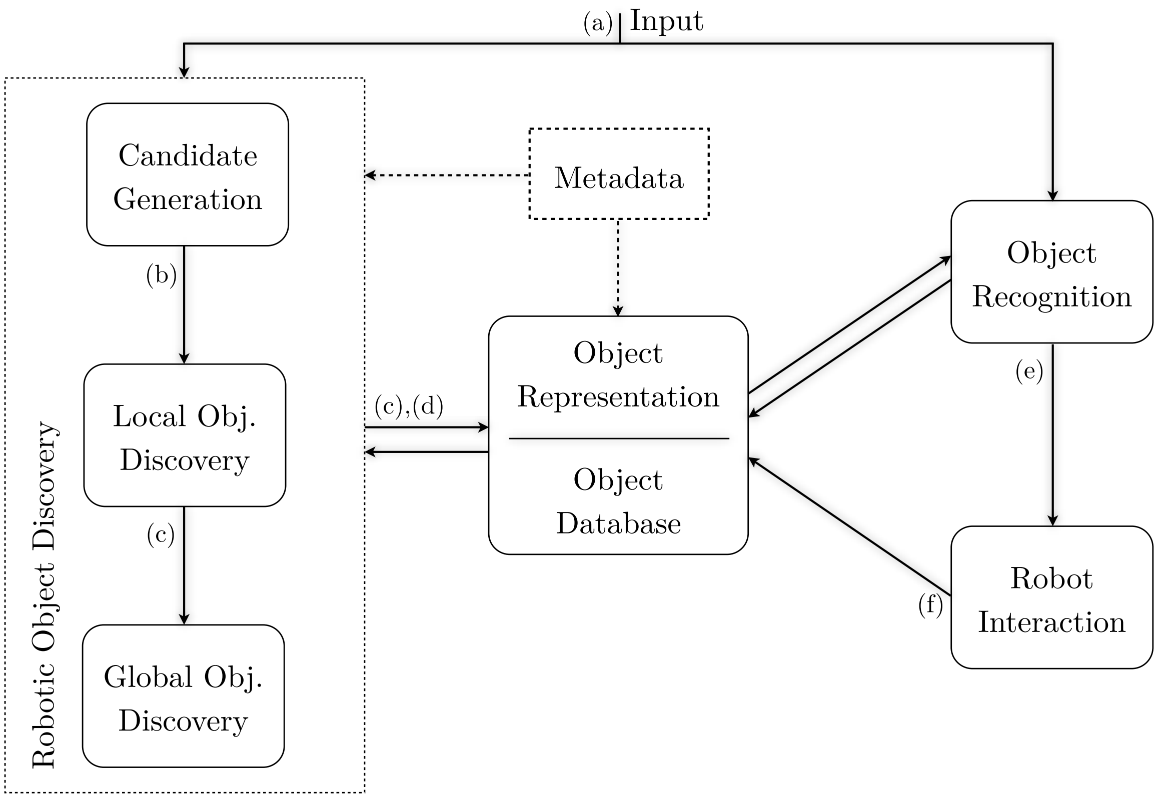
(d)


Figure 2.1: Uncoupled object perception pipeline. The training stage requires manual supervision, and each component is independent. (a) Object modeling, (b) object database, (c) object recognition, and (d) robot interaction. For each object in (b), we show (left) a sample image of the object, (center) its 3D model, and (right) its mesh-based model used for robotic manipulation.



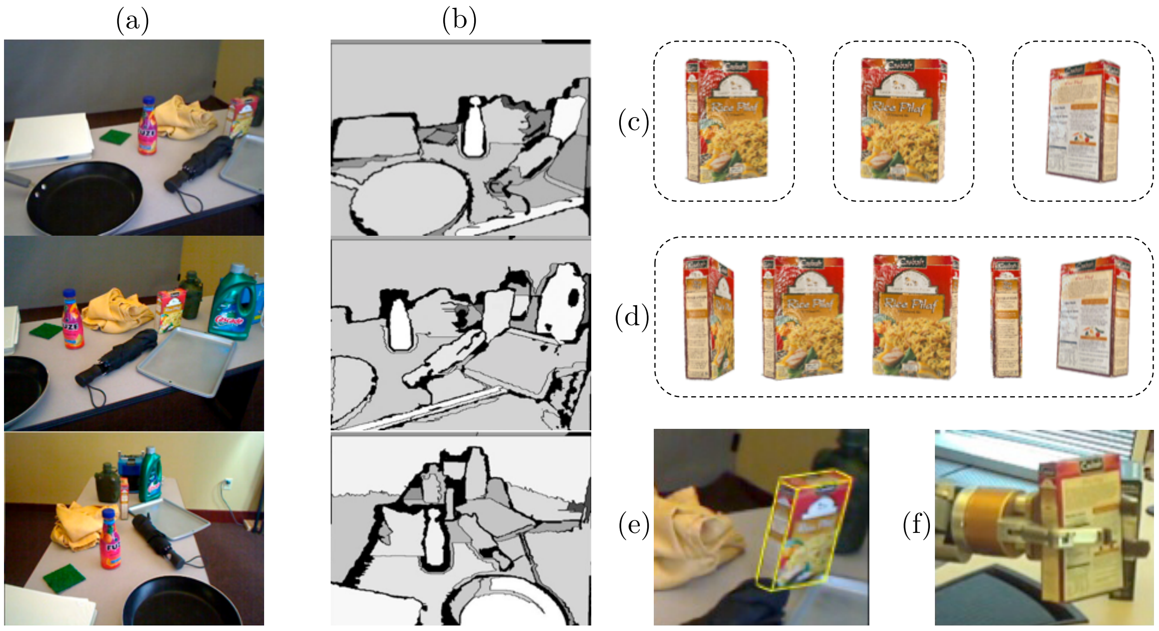


Figure 2.2: Proposed implementation of Lifelong Robotic Object Perception. (a) Input is sent to both discovery and recognition. (b) The Candidate Generation step computes *objectness*-based candidates, color coded from white (highest *objectness*) to black (lowest *objectness*). (c) Output of Local Object Discovery: partial object models. (d) Output of Global Object Discovery: groups of partial object models. (e) Recognition. (f) Robot Interaction.

Object Discovery component, and the inclusion of feedback loops among components to learn from experience.

In the **Object Recognition** step, our goal is to identify objects and estimate their poses so that a service robot can manipulate them. It is critical that the algorithm performs well in real-world data (i.e., cluttered scenes and possibly repeated objects) and with minimal latency. The object models for recognition are automatically computed by the Robotic Object Discovery and Object Representation steps. We discuss the requirements for the Object Recognition problem and our implementation in Section 2.2.

The Robotic Object Discovery (Fig. 2.2(b-d)) comprises three components: Candidate Generation, Local Object Discovery, and Global Object Discovery. The difference between the three components lies in the spatial and temporal scope that each component processes. In the **Candidate Generation** step (Fig. 2.2(b)), we process individual data samples to generate object candidates. The **Local Object Discovery** step uses robotic metadata (e.g., spatiotemporal constraints) to focus the search for objects to only a few seconds in the past. Consistent groups of candidates are used to compute partial object models (Fig. 2.2(c)). The **Global Object Discovery** step groups the partial object models from the last few Local Object Discovery steps, as well as from the current set of object models, to create full object models (Fig. 2.2(d)) that contain all the information about a given object. We discuss the requirements for the Robotic Object Discovery problem and our implementation in Section 2.3.

After Global Object Discovery, we have a 3D model for each object, alongside a set of data fragments in which each object was discovered. In the **Object Representation** step, we produce compact object models usable for object recognition. We can assess the importance of each fragment using the output of Robotic Object Discovery and the feedback from Object Recognition and Robot Interaction. In order to always keep accurate and up-to-date object models, old data fragments in object models should be replaced by new data fragments with higher importance. We discuss methods to consolidate and improve object models over time in the future work (Chapter 9.2).

The availability of **Robotic Metadata** is very helpful to make this framework feasible. We may want, for example, to only discover graspable objects, to be able to impose restrictions on the location of objects (e.g., on tables and cabinets), to use the robot's additional sensing in the search, or to use datasets of precomputed common objects.

In the **Robot Interaction** step, we manipulate some of the objects we have recognized (depending on the task). In addition, the interaction provides the necessary feedback to filter invalid discovered objects if the robot fails repeatedly to grasp them. We discuss about the interaction feedback to filter objects in the future work (Chapter 9).

2.2 Object Recognition and Pose Estimation for Manipulation

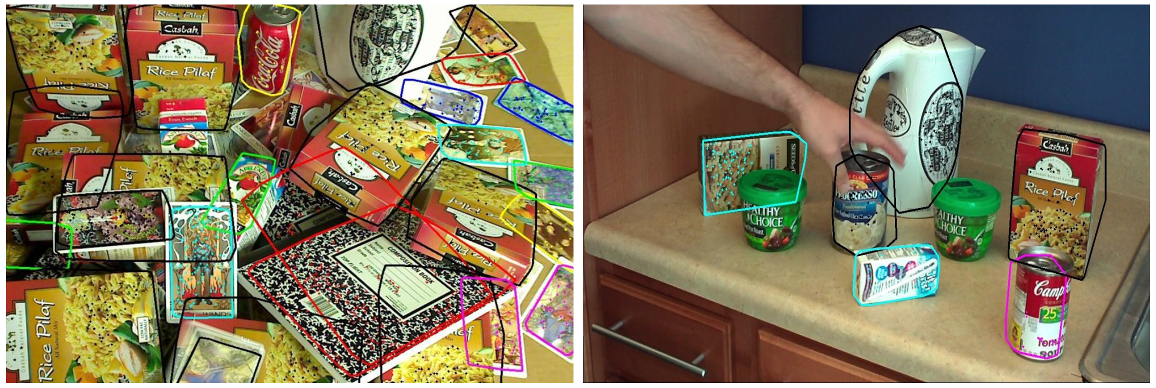


Figure 2.3: Recognition of real-world scenes. (Left) High-complexity scene. MOPED finds 27 objects, including partially-occluded, repeated and non-planar objects. Using a database of 91 models and an image resolution of 1600×1200, MOPED processes this image in 2.1 seconds. (Right) Medium complexity scene. MOPED processes this 640×360 image in 187 ms and finds all known objects (The undetected green soup can is not in the database).

The end goal of our Lifelong Robotic Object Perception framework is to recognize objects and estimate their pose for robotic manipulation. For Object Recognition, we choose the general paradigm of rigid objects and local invariant features, because these factors currently offer the best trade-off between minimal latency, reliability, and accuracy in pose estimation. We present MOPED, a framework for Multiple Object Pose Estimation and Detection that seamlessly integrates single-image and multi-image object recognition and pose estimation in one optimized, robust, and scalable framework. We address two main challenges in Object Recognition for robotics: robust performance in complex scenes, and low latency for real-time operation. We extend MOPED to robustly combine images and range data to improve its performance under heavy clutter.

Excessive scene complexity can arise due to scene clutter, with large numbers of objects in the scene, occlusions, and shadows significantly decreasing the recognition rate. Repeated objects, very common in service robotics scenarios, also contribute to increased scene complexity: the matching ambiguity introduced by repeated instances of an object presents an enormous challenge for pose estimation, because the matched features might belong to different object instances despite being correct. False positives often arise from algorithms not being able to handle unexpected scene complexity. Fig. 2.3 shows an example of an image of high complexity and multiple repeated objects correctly processed by MOPED.

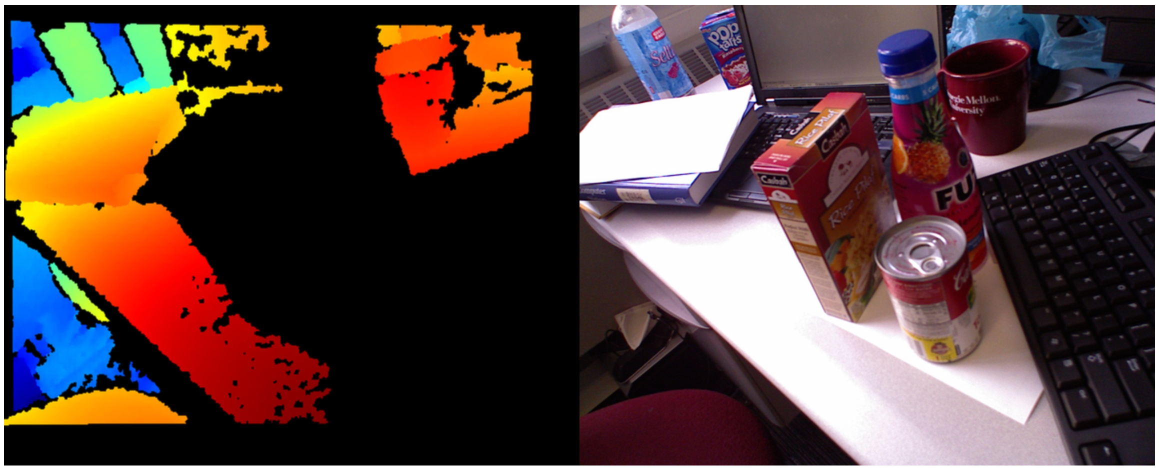


Figure 2.4: Example of scene with severe depth fading. (left) Input depth map, with black pixels indicating missing depth measurements. (right) Input image.

The second problem we analyze is that of scalability and system latency. In systems that operate online, a trade-off between recognition performance and latency must be reached, depending on the requirements for each specific task. In robotics, the reaction time of robots operating in dynamic environments is often limited by the latency of their perception. Increasing the volume of input data to process (e.g., increasing the image resolution, using multiple cameras) results in increased processing time. Yet, with cameras getting better, cheaper, and smaller, multiple high resolution views of a scene are often easily available. For example, our robot HERB has, at various times, been outfitted with cameras on its shoulder, in the palm, on its ankle-high laser, as well as with a stereo pair. Multiple views of a scene are often desirable, because they provide better depth estimation, robustness against line-of-sight occlusions, and an increased effective field of view. Higher resolution cameras can potentially improve the recognition of complicated objects and the accuracy of pose estimation algorithms. However, the penalty for using high resolution images can be steep. The increased number of features in high resolution images often degrades both the overall latency and the algorithm's precision: more features can lead to longer processing, more matching confusion, and more false positives.

A key contribution in MOPED is the Iterative Clustering-Estimation (ICE) algorithm to handle scenes with high complexity while keeping latency low. With ICE, we jointly solve the correspondence and pose estimation problems through an iterative procedure. We estimate groups of features that are likely to belong to the same object through clustering, and then search for object candidates within each of the groups. Each candidate is used to refine the feature groups that are likely to belong to the same object, which in turn helps finding more accurate candidates. The iteration of this procedure focuses the object search only in the regions with potential objects, avoiding the waste of processing power in unlikely regions. We also developed a novel object candidate scoring function based on M-estimator theory and a novel pose clustering algorithm, Projection Clustering, to detect and filter recognition outliers. We address scalability and low latency with an improved feature matching algorithm for large databases, a GPU/CPU hybrid architecture that exploits parallelism at all levels, and an optimized resource scheduler.

We extend MOPED to combine one or multiple images with depth data (from e.g., RGBD cameras) in MOPED-RGBD. Depth data can be very useful for increased precision in clutter, but we should not rely on the availability of dense depth maps for recognition. Dense depth estimation has fundamental limitations which must be addressed for robust recognition. In realistic scenes, depth sensors may fail to compute depth measurements on large portions of the associated color data (as shown in Fig. 2.4). We refer to this phenomenon as *depth fading*. Surfaces seen at oblique angles, poor lighting conditions, objects close to the camera, and reflective or specular surfaces often suffer from depth fading. These issues arise from fundamental physical limitations in depth perception, and affect all depth estimation approaches to varying degrees. In Chapter 5, we show that relying on the availability of depth data can actually *decrease* the overall recognition performance if depth fading is not considered.

2.3 Robotic Object Discovery

In the Robotic Object Discovery component of (Fig. 2.2(b-d)), our long-term goal is to develop a general solution to the problem of discovering new objects in the environment while the robot operates, for as long as the robot operates. We term this problem as the *Lifelong Robotic Object Discovery* (LROD) problem. LROD is a specialization of the generic Unsupervised Object Discovery problem that focuses on massive datasets of dynamic human environments gathered by a robotic agent. As a first step towards LROD, we aim to automatically process the raw video stream of an entire workday of a robotic agent. Considering the autonomy and charging times of current service robots, a robotic workday amounts to approximately 6-8 hours of raw sensor data (e.g., RGBD video feed) and over half a million data samples (e.g., RGBD images).

The framework shown in Fig. 2.6 summarizes our implementation for Robotic Object Discovery. We first describe the Candidate Generation step in Section 2.3.1, and the Local and Global Object Discovery steps in Section 2.3.2.

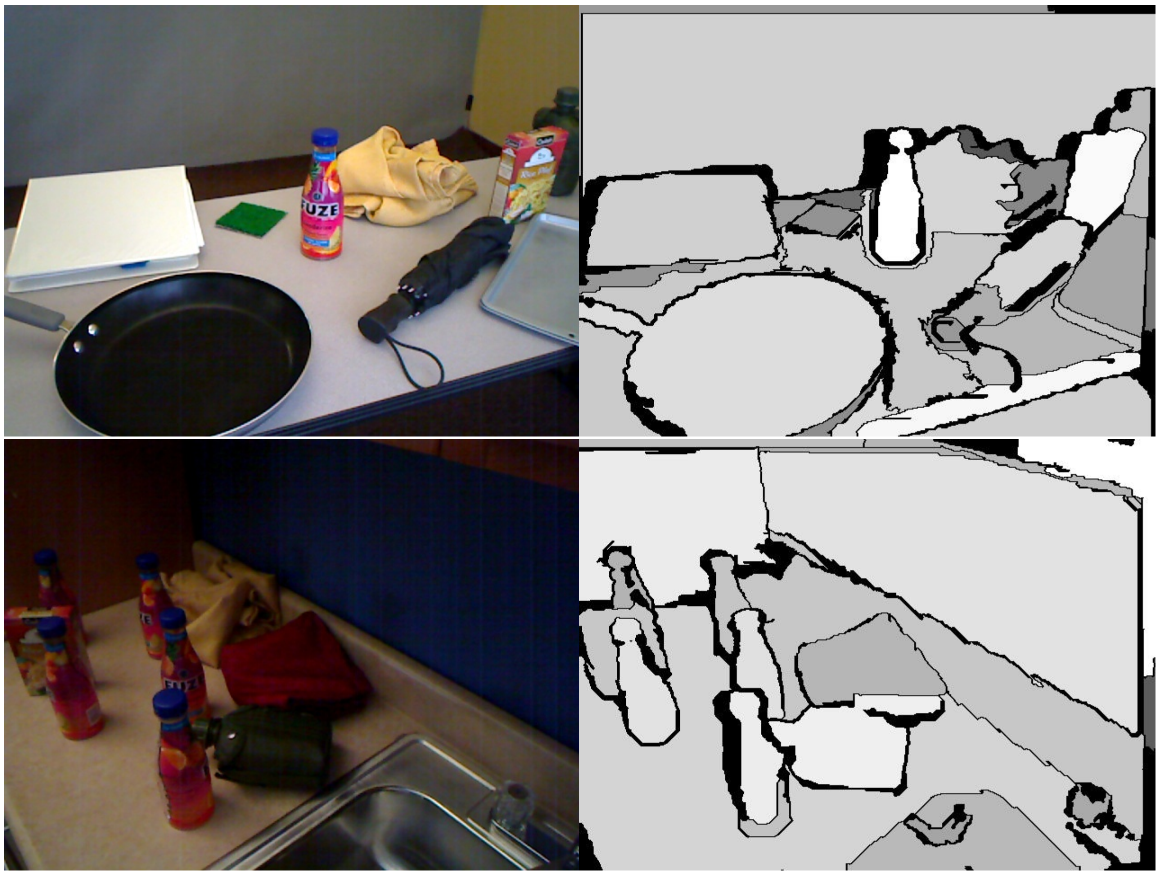


Figure 2.5: Object candidates generated for RGBD images using our *objectness*-based segmentation, Structure Discovery. (Left) Input images (range data is also an input, not shown). (Right) Ranked scene segmentation using Structure Discovery, color coded from white (highest *objectness*) to black (lowest *objectness*).

2.3.1 Candidate Generation

Our goal in the Candidate Generation step is to separate a scene into a few physically meaningful parts (objects) and discard background clutter. In particular, we aim to generate a scene segmentation, together with a ranking mechanism, such that the highest-ranking segments correspond to objects in the scene. We call this process *Structure Discovery.*

Hoiem et al. (2007) reason about occlusions as a way to segment a scene while preserving its 3D surfaces. We aim to solve a similar problem, but using a different approach. In (Hoiem et al., 2007), most of the effort is spent in reconstructing qualitative 3D interpretations of the scene. Adding range data and performing multi-modal segmentation, we can recover more information from more complex scenes, using much simpler algorithms. In particular, we combine image and range data to compute perceptual cues such as concavities and discontinuities. These cues are then used to generate scene segmentations that preserve objects.

Our main contribution in this step is a scene segmentation algorithm that exploits the availability of multi-modal (image and range) data. We generate multiple segmentations (Hoiem et al., 2005, Sivic et al., 2005) of image and range data by varying the parameters of a standard segmentation algorithm (in our case, the graph-based segmentation from Felzenszwalb and Huttenlocher (2004)). While no single segmentation is completely correct, we hope that some segments in some of the segmentations are correct and contain a whole object. We define a linkage step to relate segments in an image to the corresponding range measurements, and vice versa, to create multi-modal data regions. We term these multi-modal regions *regionlets*. We then compute region-wide features for each region, and aggregate them in a single energy function that measures the objectness of each region. We show how simple features such as color consistency, continuity, alignment, and concavity work very well to identify potential objects. We show examples of Structure Discovery candidate generation in Fig. 2.5.

We define *regionlets* as semantically equivalent regions in both image and range data. The smallest regionlet possible is a correspondence between one pixel and one 3D point, equivalent to low-level sensor fusion. By using larger image and range data regions, we can compute more powerful features for the different data sources and merge information at a higher level. An additional advantage is that we can work with the native resolution of each data source; the data sources may have different resolutions and even non-linear densities (e.g., a rotating laser range finder) with no extra overhead.

2.3.2 Local/Global Object Discovery

The framework shown in Fig. 2.6 summarizes our implementation for Robotic Object Discovery. In Fig. 2.6, a robotic agent navigates through an office environment recording an RGBD video stream (Fig. 2.6(a)). Unsupervised Object Discovery techniques (e.g., Kang et al. (2011)) create a pool of object candidates (Fig. 2.6(b)), which are represented as nodes in a pairwise graph (Fig. 2.6(c)). The graph edges are computed by comparing the visual similarity between every pair of object candidates. Then, clustering techniques are used to group similar object candidates—recurring patterns—as in Fig. 2.6(d-e). Building the pairwise graph requires $\mathcal{O}(n^2)$ similarity comparisons; as the length of the video stream grows, this cost becomes prohibitively expensive. Most of the computation time is spent comparing candidates with very low likelihood of being grouped together (the candidates in the corridor in Fig. 2.6(b)(left) and the kitchen in Fig. 2.6(b)(right)). If we analyze the input data stream based on the visual information alone, we are forced to evaluate every pair of object candidates. However, we know intuitively that objects in the kitchen and objects in the corridor have little in common. We also know that two data samples acquired within a few seconds of each other are more likely to contain the same objects than data samples acquired in different years. We can use this external information, this *metadata*, to drastically reduce the computation time and improve the quality of the discovered objects.

We claim that the key to make LROD feasible is to incorporate *robotic metadata*. Consider the example in Fig. 2.6, now using metadata. A robotic agent navigates through an office environment recording an RGBD video stream, using the robot's location and data acquisition timestamps to separate the data stream (the red-blue-green subsets in Fig. 2.6(a)). The object candidates for each subset (Fig. 2.6(b)) are compared only within the same subset. The pairwise graphs in Fig. 2.6(c) encode the visual similarity between candidates, as well as other cues such as if candidates overlap in space, or object priors based on the robot's grasping capabilities. In the clustering step (Fig. 2.6(d)), we group together object candidates with similar visual information and metadata. The metadata-augmented similarity graphs encode local information to discover individual object instances. We may discover multiple instances of the same objects in different data subsets. We perform a global clustering step (Fig. 2.6(e)) to join the multiple object instances as single object models.

The main theoretical contribution of this work is a general framework for object discovery that leverages *any* form of metadata, and in particular the natural constraints that arise in service robotics scenarios. Multiple works in the robotics literature use specific types of metadata, often by imposing restrictions on the environment, data acquisition, or agent motion, to improve performance at the cost of limited applicability when the assumptions are violated. Specific solutions could be implemented to use particular sources of metadata, but the solutions would lack adaptability, degrading with any environment changes during the lifetime of the robotic agent. For LROD, we need instead a general architecture to opportunistically leverage and adapt to the available metadata, and incorporate new metadata as it becomes available.

In our formulation, we do not distinguish between visual similarity and robotic metadata. We encode all similarities and metadata sources as an intermediate representation that we term a *constraint*. The definition of a constraint is very simple: a *measurable* yes/no question about an object candidate or a relationship between candidates, with some p—a probability of success—about the answer. For example, a visual similarity function $s(\cdot, \cdot)$ is encoded as the constraint "are candidates h_i and h_j similar in appearance?". The answer would be yes/no, with confidence $p = s(h_i, h_j)$. Metadata can be similarly encoded as constraints, as we describe in detail in Section 7.3 and Section 7.5.

With this intermediate representation of constraints, we can seamlessly combine multiple similarities and other metadata sources. We define a set of logic operations over constraints to form complex constraint expressions that encode all our knowledge relevant to discover objects. We formulate the general LROD problem as a distributed partitioning

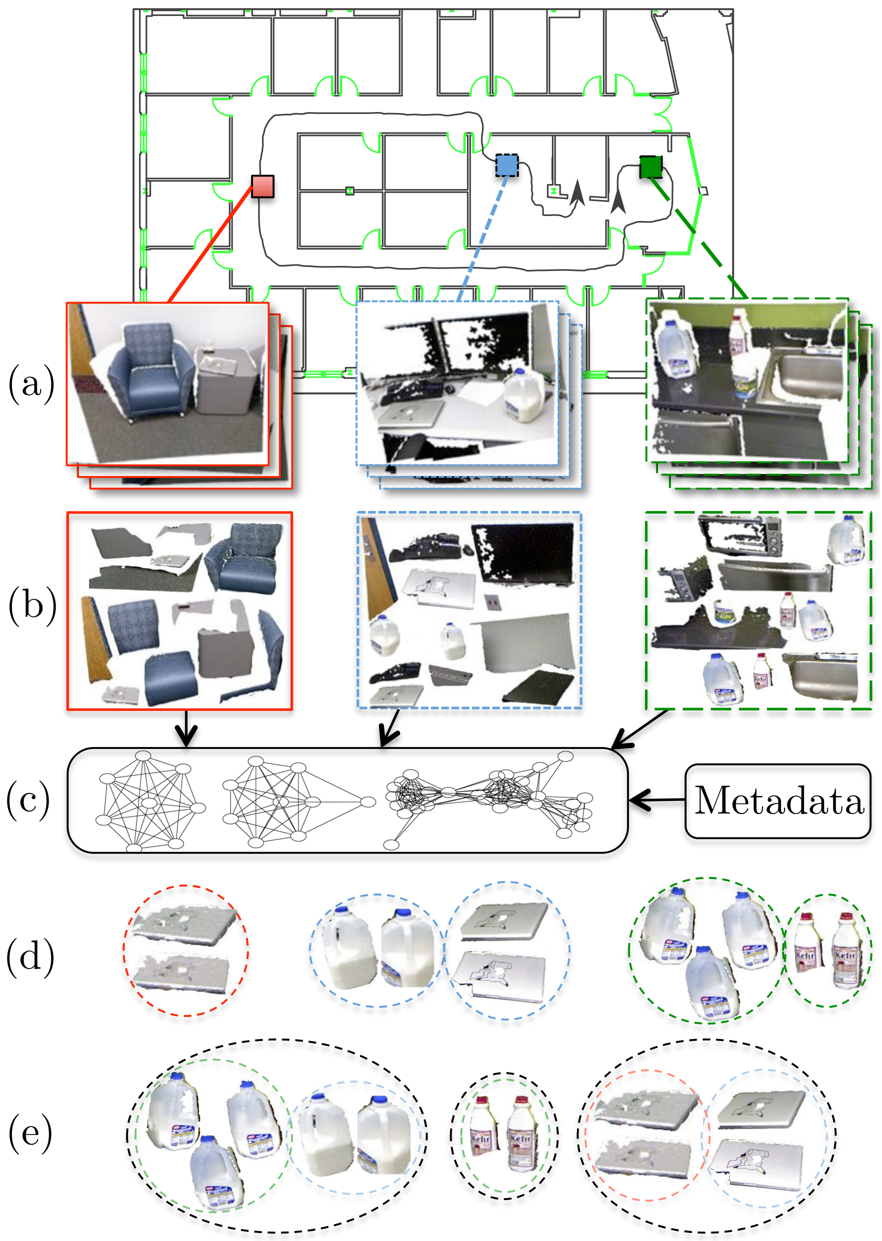


Figure 2.6: Robotic Object Discovery with Metadata (figure best viewed in color). (Top) Robotic agent navigates through office environment storing an RGBD video stream and localization information. (a) Spatial/temporal constraints separate video stream in disjoint subsets red, green and blue. (b) Images in the sequence are segmented to generate object candidates. (c) Object Discovery with Metadata: the different sequence subsets are processed independently for efficiency, using robot localization, object size/shape constraints and external knowledge to find (d) individual object instances. (e) Global Object Discovery performed on discovered object instances (d) to obtain a global representation of objects.

of graphs built over constraints, which we term Constrained Similarity Graphs (CSGs). Our distributed graph partitioning formulation is shown in Fig. 2.6, and the CSGs are illustrated in Fig. 2.6(c).

These CSGs, when coupled with natural service robotics constraints, are by construction much sparser than regular visual similarity graphs, and produce many disjoint components. This intentional graph sparsity effectively reduces the overall computational complexity for object discovery from $\mathcal{O}(n^2)$ (with respect to the number of images n) to $\mathcal{O}(n)$, as well as greatly improving the performance of the graph partitioning algorithm. In addition, our constraints-based formulation is general, in the sense that it covers both generic Unsupervised Object Discovery algorithms (e.g., Russell et al. (2006), Kang et al. (2011)) and purpose-specific algorithms (e.g., Morwald et al. (2010)).

Our main applied contribution is HerbDisc, an optimized implementation of this framework for HERB. In HerbDisc, we incorporate the natural constraints in service robotics. Our framework seamlessly integrates visual and 3D shape similarity with spatial and temporal constraints, size/shape object priors, and motion information in a flexible and extensible way. We drove HERB to over 200 offices from 4 floors of a university building, recording 6 h 20 min of continuous RGBD video of real human environments. HerbDisc processed this dataset in 18 min 34 s using a single quad-core machine and discovered 206 novel objects (44.5% precision, 28.6% recall), showcasing both the efficiency of this framework and the robustness of its results.

Chapter 3

Literature Review

Metadata. [n. meta·da·ta. me-tə-ˈdā-tə]: *Data that provides information about other data.*
Merriam-Webster.com (2012)

3.1 Object Recognition and Pose Estimation

Recognition and pose estimation of object instances in cluttered environments is a problem that has received attention from multiple research fields, and particularly Augmented Reality, Computer Vision, and Robotics. Since the literature in this area is vast, we provide only references to the research most related to our work.

3.1.1 Image-based methods

The dominant paradigm of object recognition and pose estimation in robotics is using rigid objects and local invariant features, because this combination currently offers the best trade-off between minimal latency, reliability, and accuracy in pose estimation. When using point-based features as input, the task of recognizing a single object and determining its pose from a single image requires solving two sub-problems: finding enough correct correspondences between image features and model features (data association), and estimating the model pose that best agrees with that set of correspondences (pose registration).

Estimating the pose of a rigid object model from a single image is a well studied problem in the literature. In the case of point-based features, this is known as the *Perspective-n-Point* (PnP) problem (Fischler and Bolles, 1981), for which many solutions are available, both closed-form (Lepetit et al., 2008) and iterative (Dementhon and Davis, 1995). Assuming that enough perfect correspondences between 2D image features and 3D model features are known, one only needs to use the PnP solver of choice to obtain an estimation of an object's pose. When noisy measurements are considered, non-linear least-squares minimization techniques (e.g., Levenberg-Marquardt (Marquardt, 1963)) often provide estimates of the perspective projection error. Given that such techniques require good initialization, a closed-form PnP solver is often used to initialize the non-linear minimizer.

The pose estimation problem has also been studied extensively in augmented reality research. In this area, the focus is in smooth tracking, i.e., obtaining the camera position and orientation with respect to an object or scene, and accurately registering the camera movement from one frame to the next. The well known AR toolkit from Kato and Billinghurst (1999) provides robust, accurate registration data from an object without any manual initialization. However, it requires artificial markers for tracking, which make it unsuitable for robotics applications—and particularly household robotics—as it would require placing markers on each and every object in the household. Vacchetti et al. (2004) overcome the marker limitation by using CAD models of the tracked objects/scenes, and precomputing keyframes from the most informative views of each scene. Gordon and Lowe (2006) propose a method for accurate camera tracking using a metric reconstruction of a scene from invariant local descriptors, and minimize camera jitter and drift in frame transitions with regularized non-linear minimization on the camera locations, in a work that serves as a base for our own object recognition and pose estimation system, MOPED (Chapter 4).

The data association problem, which is extremely challenging in cluttered scenes, was greatly simplified with the advent of discriminative local invariant features (Lowe, 2004, Bay et al., 2008, Mikolajczyk and Schmid, 2005). However, even with highly discriminative locally invariant features, such as SIFT (Lowe, 2004) or SURF (Bay et al., 2008), mismatched correspondences are inevitable, forcing us to utilize robust estimation techniques such as M-estimators or RANSAC (Fischler and Bolles, 1981) in most modern object recognition systems. A common problem in object recognition is degraded performance due to scene complexity. This problem can arise due to scene clutter, where large numbers of objects, occlusions, and poor lighting conditions significantly decrease the recognition rate.

Related is the issue of repeated objects: the matching ambiguity introduced by repeated instances of an object presents an enormous challenge for robust estimators, as the matched features might belong to different object instances despite being correct. Solutions such as perceptual grouping (Lowe, 1987), interpretation trees (Grimson, 1991), Hough clustering (Ballard, 1980, Grimson and Huttenlocher, 1990) or image space clustering (Collet et al., 2009) are often used to mitigate false positives due to excessive scene complexity.

Lowe (1987) propose to group line features using perceptual cues such as proximity, parallelism, and collinearity, to limit the object search space, and estimates of the object viewpoint are performed via least-squares optimization of the reprojected object model in the image.

Grimson and Lozano-Perez (1987) introduce the concept of interpretation trees for object recognition and exploit local geometric constraints to efficiently explore the trees. Interpretation trees agglomerate information from multiple image segments into plausible object candidates: the first tree level contains individual object features, the second level pairs of features, and the N-th level contains only those candidates that agree with N object features.

Viksten et al. (2009) use Hough voting for object recognition and pose estimation of repeated objects, by solving the PnP problem with triplets of point features to compute pose candidates, and filtering them with an averaging voting mechanism on the Hough pose space. The most voted triplets are then considered objects.

The work of Hinterstoisser et al. (2010) in Dominant Orientation Templates for real-time Object Recognition, and its more recent extension to RGBD images in Hinterstoisser et al. (2011), propose an alternative to the recognition and pose estimation based on point features. Hinterstoisser et al. (2010) propose to use a large set of templates and fast indexing based on a variant of HoG (Dalal and Triggs, 2005) tuned for real time operation. New images are explored with a branch-and-bound technique to match them against existing templates, which provide instant localization and visually reasonable, albeit discrete, pose estimation. However, this technique is very sensitive to partial occlusions, and it remains to be seen how accurate the pose estimation is for robotic manipulation tasks.

A comprehensive overview of single-image, model-based 3D object recognition/tracking techniques is available at Lepetit and Fua (2005).

3.1.2 3D-based methods

The use of range data for object recognition and pose estimation has also been extensively studied in the literature. The main criticism of 3D-only techniques is that geometry alone is not discriminative enough for the small, “boring” shapes often found in man-made objects, in which simple boxes and bottles are very common. Products such as refined sugar and rat poison, which come in similarly-shaped boxes, are examples of items that we certainly do not want our service robot to confuse. We refer the reader to Jain and Dorai (2000), Tangelder and Veltkamp (2004), Bimbo and Pala (2006) for comprehensive surveys on 3D-based recognition and shape retrieval, which fall outside the scope of this thesis.

3.1.3 RGBD-based methods

There has been much recent interest in the study of algorithms for RGBD cameras, such as the Microsoft Kinect. RGBD cameras compute dense depth maps registered to images, in which each depth measurement is registered to one pixel. RGBD cameras are inexpensive and retrieve dense depth maps in real time (at 30fps), which make them a very interesting alternative to alternative sources of depth measurements (e.g., laser, passive stereo).

Hinterstoisser et al. (2011) improve their earlier work in Hinterstoisser et al. (2010) to use multi-modal templates of objects from RGBD images. The addition of depth to the DOT templates filters out many spurious contours and avoids the scale ambiguity from the image-only approach, and enables recognition in cluttered scenes with moderate occlusion.

Lai et al. (2011b) propose Instance Distance Learning (IDL) to perform object category and instance classification in RGBD images. In this work, Lai et al.define a view-to-object distance where novel images are compared simultaneously to a set of templates of a previous object. The view-to-object distance is based on a weighted combination of feature differences between views. This is a classification method, i.e., it evaluates previously segmented patches to compute object identities. The authors use a sliding window approach to do for RGBD recognition.

Bo et al. (2011) introduce Hierarchical Kernel Descriptors (HKDES) to learn region-based features from point-based attributes, such as color or depth. The HKDES descriptors are applied to RGBD object recognition in Lai and Fox (2011). In Lai and Fox (2011), the authors describe a tree-based approach to perform simultaneously category, instance and pose classification from a large dataset of categories, instances and object poses. The authors also introduce a recognition system, OASIS, which uses a tabletop detector to segment a set of candidate regions that are then classified with the tree-based approach. The use of a tabletop detector limits the applicability of OASIS to uncluttered scenes.

An important question that often arises in RGBD recognition is what to do with missing data (depth fading). It is standard practice in the RGBD literature (e.g., Lai and Fox (2011), Silberman et al. (2012), Janoch et al. (2011)) to assume dense depth maps with one to one correspondences to image pixels. To address depth fading, researchers resort to interpolating depth data and then propagating the interpolated values. Common interpolation methods include the recursive median filter (Lai and Fox, 2011), inpainting (Silberman et al., 2012), or optimization techniques that minimize curvature (Janoch et al., 2011). These methods are effective when used for interpolation (e.g., the small holes in Fig. 5.1(left)), but produce severely inaccurate results when used for extrapolation (e.g., the moderate fading in Fig. 5.1(right)). We explore how to robustly address depth fading in object recognition in Chapter 5 of this thesis, MOPED-RGBD.

3.2 Robotic Object Discovery

Consider the example of Robotic Object Discovery shown in Fig. 3.1. We identify five major components. The *World* Ω represents the underlying physical phenomenon (i.e., the environment) where we discover objects. A *physical agent* $\mathcal{A}$ (e.g., a robot) gathers data through observation or interaction with the world Ω. The physical agent uses *sensors* $\mathcal{S}$ (e.g., a camera) to gather data samples I (e.g., images). A *candidate generator* $\mathcal{H}$ produces

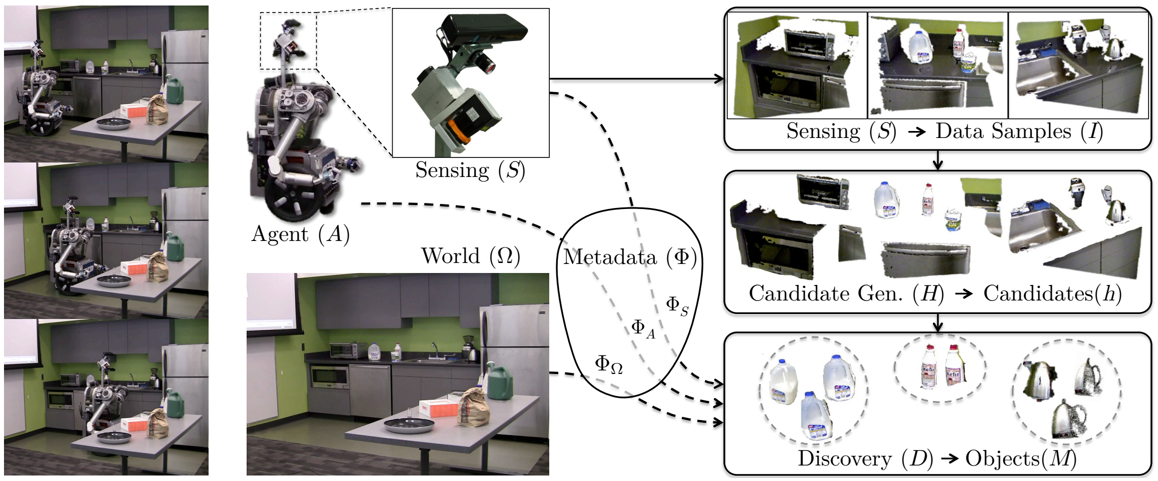

Sensing (S)
Agent (A)
World (Ω)
Metadata (Φ)
Φ_S
Φ_A
Φ_Ω
Sensing (S) → Data Samples (I)
Candidate Gen. (H) → Candidates(h)
Discovery (D) → Objects(M)

Figure 3.1: Main components in Robotic Object Discovery. (left) the robot HERB moves through a kitchen searching for novel objects. (center) The three physical components of Robotics Object Discovery are: the world Ω, the robotic agent $\mathcal{A}$, and the sensors $\mathcal{S}$. (right) The sensors capture data samples x to be processed by a candidate generator $\mathcal{H}$ to produce object candidates. The Discoverer $\mathcal{D}$ groups recurring object candidates into objects, using candidate data and metadata sources Φ_Ω (e.g., assumption "objects lie on tables"), $\Phi_\mathcal{A}$ (e.g., robot localization data), $\Phi_\mathcal{S}$ (e.g., image ordering and timestamps).

object candidates h (RGBD image regions) from data samples. Finally, the *discoverer* $\mathcal{D}$ groups recurring object candidates into objects.

In this section, we perform a literature review of techniques for Candidate Generation $\mathcal{H}$ and Object Discovery $\mathcal{D}$.

3.2.1 Candidate Generation

The search for generic objects (of unknown classes) in a single image or range scan has received recent attention from both Computer Vision (e.g., Endres and Hoiem (2010), Alexe et al. (2010), Carreira and Sminchisescu (2010)) and Robotics communities (e.g., Rusu et al. (2009b, 2010), Bjorkman and Kragic (2010)). Our area of research is most related to the field of *objectness* segmentation, in which algorithms find regions that are the most likely to be objects (in images and/or range data).

Alexe et al. (2010) propose a measure of *objectness* that attempts to quantify how likely an image window is to contain an object of any class. They use a Bayesian framework to combine features based on foreground/background comparisons, including a novel *superpixel straddling* feature that measures how many superpixels cross the estimated object boundary, which reportedly works very well on natural images. This technique assumes that objects are reasonably large (at least a few superpixels), so it is unclear how it would perform in scenes with small objects, typical of household environments.

Endres and Hoiem (2010) propose a method to produce a ranked set of regions from a single image, such that the top-ranked regions are the most likely to be good segmentations of objects. They encourage spatial diversity via structured learning to segment objects placed in different image locations, minimizing overlap. Their region ranking is based on features such as boundary probabilities (the globalPb detector of Maire et al. (2008)) and difference of color histograms between foreground and background.

Carreira and Sminchisescu (2010) generate multiple binary segmentations of objects and rank them according to their "object plausibility" using a random forest regressor. The multiple binary segmentations are generated using a min-cuts/max-flow framework, choosing foreground/background seeds from a grid. The objectness measure of Carreira and Sminchisescu (2010) learns a ranking for the different segments, using features from graph partitioning, region-based, intra-region texture/brightness comparisons, and boundary probabilities (Maire et al., 2008).

In the range sensing community, Rusu and Cousins (2011) implement in PCL an approach to find novel objects in domestic environments. The authors make the assumption that all interesting objects in their scenes lie on top of horizontal planes (e.g., tables), and proceed to search for planes prior to any object search. Detected planes are segmented out, and the 3D points on top of the planes are clustered according to their pairwise distances to find a small subset of object candidates. This approach has been very popular in the robotics community, with many authors restricting their input to tabletop scenes and implementing some form of horizontal plane segmentation (e.g., Rusu et al. (2009b), Bjorkman and Kragic (2010), Marton et al. (2010), Lai and Fox (2011), Kootstra and Kragic (2011), Mishra et al. (2012)).

Bjorkman and Kragic (2010) combine an image with stereo data to automatically detect and segment unknown objects. The authors use a probabilistic framework to jointly optimize the detection and segmentation processes. In order to do that, strong assumptions are made, particularly on the location and scale of objects (large, near the image center) and on the detection of a planar supporting surface, which limit the applicability of their approach.

Kootstra and Kragic (2011) formalize the work of Bjorkman and Kragic (2010) as a probabilistic framework based on Gestalt principles, which have been shown to be very useful in early segmentation in humans (Wertheimer, 1938). The approach of Kootstra and Kragic implements very similar cues to our own Structure Discovery algorithm, described in Chapter 6, such as concavity, continuity, contour compactness, etc.

As a key part of Structure Discovery, we develop a novel region-based image and range data fusion. Most of the literature on RGBD perception focuses on low-level fusion with one-to-one correspondences to merge image and range data (e.g., Bjorkman and Kragic

(2010), Posner et al. (2008), Gould et al. (2008)), either at the pixel level. Pixel level fusion assumes depth measurements for every pixel, that is, dense depth images (Bjorkman and Kragic, 2010, Gould et al., 2008). Dense depth estimation, however, has fundamental limitations that must be considered. As we discuss in Chapter 2.2, RGBD images may suffer from depth fading in large portions of the image. If the range data source is a laser range finder, these algorithms often require super-resolution techniques (Diebel and Thrun, 2005) because images usually have higher resolution than their corresponding point clouds.

3.2.2 Object Discovery

The aim of Unsupervised Object Discovery (Weber et al., 2000, Russell et al., 2006) is to jointly segment and learn the appearance of unknown objects in the environment. Unsupervised Object Discovery is very challenging, in part because the definition of object is subjective, as it depends on the observer. Furthermore, different works use different input sources (e.g., unorganized collections of images (Weber et al., 2000, Kang et al., 2011), image sequences (Morwald et al., 2010), images with disparity (Somanath et al., 2009), laser data (Ruhnke et al., 2009)) to produce different data outputs (e.g., clusters of images (Weber et al., 2000), clusters of bounding boxes (Lee and Grauman, 2011), clusters of image segments (Russell et al., 2006, Kang et al., 2011), 3D models (Somanath et al., 2009)), and using different assumptions (e.g., one object per image (Weber et al., 2000), only tabletop scenes (Kootstra and Kragic, 2011), multiple views of the same scene (Herbst et al., 2011)) depending on the application. Comparing the performance between methods is very challenging due to this disparity in inputs, definition of objects, assumptions, and outputs.

Methods in Unsupervised Object Discovery that assume an unorganized collection of images as input are very common in Computer Vision research (e.g., Weber et al. (2000), Russell et al. (2006), Lee and Grauman (2011), Kang et al. (2011, 2012), Philbin et al. (2010), Sivic et al. (2005), and the general survey of Tuytelaars et al. (2009)). Using an unorganized collection of images as input implies, in terms of Fig. 3.1, that we assume no knowledge about the world, the physical agent, or the sensing. Some of these methods, such as Weber et al. (2000), Tuytelaars et al. (2009), focus only on grouping entire images in categories (i.e., assuming that each image mostly contains a single, large object), which is equivalent to not using a candidate generator $\mathcal{H}$.

The key difference between Unsupervised Object Discovery and LROD is the amount and variety of information sources. Most methods in Unsupervised Object Discovery assume that no information is available about the world Ω, the physical agent $\mathcal{A}$, or the sensors $\mathcal{S}$. As datasets grow larger, visual information becomes less discriminative and recurring visual patterns appear everywhere. In addition, algorithms often require pairwise operations over all pairs of candidates, which makes them computationally expensive. In LROD, metadata—non-visual information from Ω, $\mathcal{A}$, and $\mathcal{S}$—is not only available, but necessary; we need a general architecture to leverage both visual information and metadata to discover objects and adapt as conditions change.

Prior work in robotics has widely used metadata to limit computational costs and improve robustness in perception. The metadata is mostly incorporated by imposing restrictions restrictions on the environment, data acquisition, or agent motion, which often result in single-purpose solutions of limited applicability. Common assumptions include partial knowledge of the world Ω, usually about the scene configuration or the appearance or shape of objects. Marton et al. (2010) assumes that interesting objects lie on tables to segment novel objects in 3D point clouds. A horizontal plane detector is used to pre-segment the scene and enforce the tabletop assumption. This same assumption is shared by other works in the robotics literature, such as Bjorkman and Kragic (2010), Kootstra and Kragic (2011). Mishra and Aloimonos (2011) use 3-frame sequences, motion cues, and assume that images contain a table with known color distribution to discover and accurately segment objects in cluttered scenes. Morwald et al. (2010) assume that relevant objects may be modeled by simple shapes (such as boxes or cylinders) and that images come in sequences to perform automated modeling of household objects, enforcing temporal consistency with tracking. Both Mishra and Aloimonos (2011) and Morwald et al. (2010) assume some knowledge on constraints about the sensors $\mathcal{S}$ (image ordering and sequencing). Herbst et al. (2011) use datasets consisting of multiple sequences of images collected in the same locations, in order to compute per-sequence environment maps and perform scene differencing to discover movable objects. The implicit assumptions include the knowledge of the robot location, recording time, and that the robotic agent $\mathcal{A}$ visits the same locations multiple times. Rusu et al. (2008) assume strong prior shape and location knowledge to segment cabinets, drawers and shelves in kitchen environments, which are in turn used as cues for the most likely locations of objects. Other works assume an active robotic agent $\mathcal{A}$ that interacts with Ω, $\mathcal{S}$ and $\mathcal{H}$ to modify the environment and improve the object discovery process; for example, Fitzpatrick (2003) track movable objects through random interactions with the environment.

All these works use metadata and assumptions to improve performance and efficiency for their particular setups, at the cost of underperforming (and, often, not working at all) in alternative types of scenes. Our general architecture addresses these shortcomings with a common formulation for metadata, thus allowing us to opportunistically take advantage of different sources of information as conditions change.

In our framework, we combine multiple sources of information (visual similarity and metadata) in CSGs, and cluster the CSGs to obtain groups of object candidates. In the clustering literature, this area is known as multi-similarity (or multi-source) clustering.

While multi-similarity clustering applied to Unsupervised Object Discovery is a novelty of this work, other fields (e.g., bioinformatics) commonly use multi-similarity clustering to combine multiple heterogeneous data sources. Zeng et al. (2010) combine gene expression data, text, and clustering constraints induced by the text data, to identify closely related genes. Zeng et al.use a variant of EM in which parameter estimation and cluster reassignment are performed over a single data source picked at random at each iteration. Troyanskaya et al. (2003) introduce a Bayesian framework to cluster protein-protein interaction patterns based on multiple sources of protein relations. The Bayesian network combines multiple clusterings (one for each data source) using human expert knowledge to estimate the prior probabilities of the interaction patterns.

Other fields such as machine learning and data mining have also shown interest in multi-similarity clustering. Bouvrie (2004) considers the problem of multi-similarity clustering with partially missing data, where not all data sources are available for all points. Bouvrie optimizes an information-theoretic objective function over pairs of co-occurrence matrices, which requires $\binom{n}{2}$ clustering steps (for n data sources). Tang et al. (2009) propose Link Matrix Factorization, in which multiple graphs for different data sources are approximated by a graph-specific factor and a factor common to all graphs, where the common factor is the consensus partition. Strehl and Ghosh (2002) combine multiple clusterings as a combinatorial optimization problem over the shared mutual information between clusterings. This method performs clusterings for individual data sources first, and a clustering over the co-occurrences of data labels, which the authors term a *cluster ensemble*. Hore et al. (2006) modify the cluster ensembles of Strehl and Ghosh (2002) to use clustering centroids instead of clustering labels. This change enables the combination of disjoint datasets into the same cluster ensemble, with centroids acting as representatives for the data in their clusters.

All previously mentioned methods for multi-similarity clustering except Hore et al. (2006) suffer from poor scalability, as they all require computing and storing multiple clusterings of the full dataset for each individual data source. In object discovery, some data sources (in particular, visual similarity) are very expensive to compute; therefore, clustering each individual data source can be very costly. Some cases, such as Tang et al. (2009), also require multiple full adjacency matrices in memory, which is infeasible for large datasets. In our work, we take the route of Hore et al. (2006) of computing consensus clusters over disjoint datasets. The key differences between Hore et al. (2006) and our work arise from our clustering method being tailored for object discovery. First, we compute disjoint subsets of data samples dynamically from metadata, and not random splits. Second, we use partial 3D object models as intermediate representations, and not centroids. The partial 3D models encode more information than centroids or individual candidates h_i, so our clustering method is asymmetric: the similarity functions that create the disjoint subsets (visual features and metadata) are different than the similarity functions in the consensus clustering (more complex visual and 3D features).

Part II

Object Recognition and Pose Estimation

Overview of Part II

In Lifelong Robotic Object Perception, the end goal is to perceive objects for robotic manipulation. In the Object Recognition and Pose Estimation chapters of this thesis, we pursue the goal of identifying objects and estimating their poses so that a service robot can manipulate them. A key requirement for the algorithms we present is that they must perform well in real-world data (i.e., cluttered scenes and possibly repeated objects) and with minimal latency.

We introduce two systems for Object Recognition and Pose Estimation: MOPED (Section 4), and its extension MOPED-RGBD (Section 5). Both systems use the same approach of using rigidity constraints and local invariant features, because this combination currently offers the best trade-off between minimal latency, reliability, and accuracy in pose estimation.

MOPED is a framework for Multiple Object Pose Estimation and Detection that integrates single-image and multi-image object recognition and pose estimation. In MOPED, we address two main challenges in computer vision for robotics: robust performance in complex scenes, and low latency for real-time operation. We achieve robust performance with multiple algorithmic contributions, including Iterative Clustering-Estimation (ICE), a novel object hypothesis scoring function based on M-estimator theory, and Projection Clustering a novel pose clustering algorithm that robustly handles recognition outliers. We achieve scalability and low latency with an improved feature matching algorithm for large databases, a GPU/CPU hybrid architecture that exploits parallelism at all levels, and an optimized resource scheduler.

We extend MOPED to leverage RGBD images using an adaptive image-depth fusion model based on maximum likelihood estimates. We incorporate this model to each stage of MOPED to achieve object recognition robust to imperfect depth data. The resulting system, MOPED-RGBD, leverages adaptive pose estimation, adaptive object priors, and adaptive feature matching to opportunistically use depth information when available and seamlessly transition to the performance of the image-only MOPED when depth measurements are not available.

Chapter 4

The MOPED Framework

Recognize. [v. rec·og·nize. ˈre-kəg-nīz]: *To perceive to be something or someone previously known.*

Merriam-Webster.com (2012)

In this chapter we present MOPED, a framework for Multiple Object Pose Estimation and Detection that seamlessly integrates single-image and multi-image object recognition and pose estimation in one optimized, robust, and scalable framework. We address two main challenges in computer vision for robotics: robust performance in complex scenes, and low latency for real-time operation.

We achieve robust performance with Iterative Clustering-Estimation (ICE), a novel algorithm that iteratively combines feature clustering with robust pose estimation. Feature clustering quickly partitions the scene and produces object hypotheses. The hypotheses are used to further refine the feature clusters, and the two steps iterate until convergence. ICE is easy to parallelize, and easily integrates single- and multi-camera object recognition and pose estimation. We also introduce a novel object hypothesis scoring function based on M-estimator theory, and a novel pose clustering algorithm that robustly handles recognition outliers.

We achieve scalability and low latency with an improved feature matching algorithm for large databases, a GPU/CPU hybrid architecture that exploits parallelism at all levels, and an optimized resource scheduler. We provide extensive experimental results demonstrating state-of-the-art performance in terms of recognition, scalability, and latency in real-world robotic applications.

This chapter, as well as the earlier publications of MOPED in Martinez et al. (2010), Collet et al. (2011b), are joint work with Manuel Martinez.

4.1 Problem formulation

The goal of MOPED is the recognition of objects from images given a database of object models, and the estimation of the pose of each recognized object. In this section, we formalize these inputs and outputs and introduce the relevant terminology for this chapter.

4.1.1 Input: images

The input to MOPED is a set $\mathbf{I}$ of M images

$$\mathbf{I} = \{I_1, \dots, I_m, \dots, I_M\} \qquad I_m = \{K_m, T_m, \mathbf{g}_m\}. \tag{4.1}$$

In the general case, each image is captured with a different calibrated camera. Therefore, each image I_m is defined by a 3×3 matrix of intrinsic camera parameters K_m, a 4×4 matrix of extrinsic camera parameters T_m with respect to a known world reference frame, and a matrix of pixel values $\mathbf{g}_m$.

MOPED is agnostic to the number of images M. In other words, it is equally valid in both an extrinsically calibrated multi-camera setup, and in the simplified case of a single image ($M = 1$) and a camera-centric world ($T_1 = \mathcal{I}_4$, where $\mathcal{I}_4$ is a 4×4 identity matrix).

4.1.2 Input: object models

Each object to be recognized by MOPED first goes through an off-line learning stage, in which a sparse 3D model of the object is created. First, a set of images is taken with the object in various poses. Reliable local descriptors are extracted from natural features using SIFT (Lowe, 2004), which have proven to be one of the most distinctive and robust local descriptors across a wide range of transformations (Mikolajczyk and Schmid, 2005). Alternative descriptors (e.g., SURF (Bay et al., 2008), ferns (Ozuysal et al., 2010)) can also be used. Using structure from motion (Szeliski and Kang, 1994) on the matched SIFT keypoints, we merge the information from each training image into a sparse 3D model. Each 3D point is linked to a descriptor that is produced from clustering individual matched descriptors in different views. Finally, proper alignment and scale for each model are computed to match the real object dimensions and define an appropriate coordinate frame, which for simplicity is defined at the object's center.

Let $\mathbf{O}$ be a set of object models. Each object model is defined by its object identity o and a set of features $\mathbf{F}_o$

$$O = \{o, \mathbf{F}_o\} \qquad \mathbf{F}_o = \{F_{1;o}, \dots, F_{i;o}, \dots, F_{N;o}\}. \tag{4.2}$$

Each feature is represented by a 3D point location $P = [X,\ Y,\ Z]^T$ in the object's coordinate frame and a feature descriptor D, whose dimensionality depends on the type of descriptor used, e.g., $k = 128$ if using SIFT or $k = 64$ if using SURF. That is,

$$F_{i;o} = \{P_{i;o}, D_{i;o}\} \qquad P_{i;o} \in \mathbb{R}^3,\ D_{i;o} \in \mathbb{R}^k. \tag{4.3}$$

The union of all features from all objects in $\mathbf{O}$ is defined as $\mathbf{F} = \bigcup_{o \in \mathbf{O}} \mathbf{F}_o$.

4.1.3 Output: recognized objects

The output of MOPED is a set of object hypotheses $\mathbf{H}$. Each object hypothesis $H_h = \{o, T_h\}$ is represented by an object identity o and a 4×4 matrix T_h that corresponds to the pose of the object with respect to the world reference frame.

4.2 Iterative Clustering-Estimation

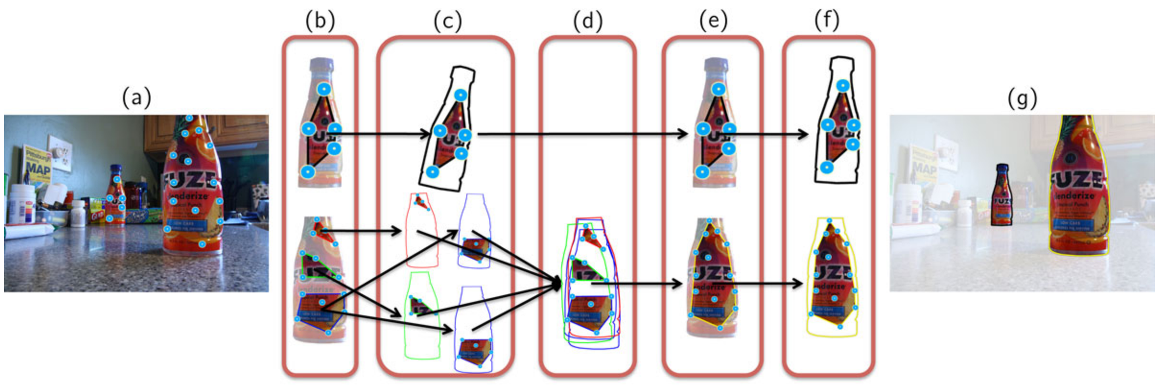


Figure 4.1: Illustration of two ICE iterations. Colored outlines represent estimated poses. (a) Feature extraction and matching. (b) Feature clustering. (c) Hypothesis generation. (d-e) Cluster Clustering. (f) Pose refinement. (g) Final result.

The task of recognizing objects from local features in images requires solving two subproblems: the *correspondence problem* and the *pose estimation problem*. The correspondence problem refers to the accurate matching of image features to features that belong to a particular object. The pose estimation problem refers to the generation of object poses that are geometrically consistent with the found correspondences.

The inevitable presence of mismatched correspondences forces us to utilize robust estimation techniques, such as M-estimators or RANSAC (Fischler and Bolles, 1981). In the presence of repeated objects in a scene, the correspondence problem cannot be solved in isolation, as even perfect image-to-model correspondences need to be linked to a particular object instance. Robust estimation techniques often fail as well in the presence of this increased complexity. Solutions such as grouping (Lowe, 1987), interpretation trees (Grimson, 1991) or image space clustering (Collet et al., 2009) alleviate the problem of repeated objects by reducing the search space for object hypotheses.

The Iterative Clustering-Estimation (ICE) algorithm at the heart of MOPED aims to jointly solve the correspondence and pose estimation problems in a principled way. Given initial image-to-model correspondences, ICE iteratively executes clustering and pose estimation to progressively refine which features belong to each object instance, and to compute the object poses that best fit each object instance. The algorithm is illustrated in Fig. 4.1.

Given a scene with a set of matched features, (Fig. 4.1(a)), the *Clustering* step generates groups of image features that are likely to belong to a single object instance (Fig. 4.1(b)). If prior object pose hypotheses are available, features consistent with each object hypothesis are used to initialize distinct clusters. Numerous object hypotheses are generated for each cluster (Fig. 4.1(c)). Then, object hypotheses are merged together if their poses are similar (Fig. 4.1(d)), thus uniting their feature clusters into larger clusters that potentially contain all information about a single object instance (Fig. 4.1(e)). With multiple images, the use of a common reference frame allows us to link object hypotheses recognized in different images, and thus create multi-image feature clusters. If prior object pose hypotheses are not available (i.e., at the first iteration of ICE), we use the density of local features matched to an object model as a prior, with the intuition that groups of features spatially close together are more likely to belong to the same object instance than features spread across all images. Thus, we initialize ICE with clusters of features in image space (x, y), as seen in Fig. 4.1(b).

The *Estimation* step computes object hypotheses given clusters of features (as shown in Fig. 4.1(c) and Fig. 4.1(f)). Each cluster can potentially generate one or multiple object hypotheses, and also contain outliers that cannot be used for any hypothesis. A common approach for hypothesis generation is the use of RANSAC along with a pose estimation algorithm, although other approaches are equally valid. In RANSAC, we choose subsets of features at random within the cluster, then hypothesize an object pose that best fits the subset of features, and finally check how many features in the cluster are consistent with the pose hypothesis. This process is repeated multiple times and a set of object hypotheses is generated for each cluster. The advantage of restricting the search space to that of feature clusters is the higher likelihood that features from only one object instance are present, or at most a very limited number of them. This process can be performed regardless of whether the features belong to one or multiple images.

The set of hypotheses from the *Estimation* step are then utilized to further refine the membership of each feature to each cluster (Fig. 4.1(d-e)). The whole process is iterated until convergence, which is reached when no features change their membership in a *Clustering* step (Fig. 4.1(f)).

In practice, ICE requires very few iterations until convergence, usually as little as 2 for setups with one or a few simultaneous images. Parallelization is easy, since the initial steps are independent for each cluster in each image and object type. Therefore, large sets of images can be potentially integrated into ICE with very little impact on overall system latency. Two ICE iterations are required for increased robustness and speed in setups ranging from one to a few simultaneous images, while further iterations of ICE might potentially be necessary if tens or hundreds of simultaneous images are to be processed. In Section 4.3, we describe how to apply ICE for robust object recognition in cluttered scenes.

4.2.1 ICE as Expectation-Maximization

It is interesting to note the conceptual similarity between ICE and the well-known Expectation-Maximization (EM) (Dempster et al., 1977) algorithm, particularly in the learning of Gaussian Mixture Models (GMM) (Redner and Walker, 1984). EM is an iterative method for finding parameter estimates in statistical models than contain unobserved latent variables, alternating between expectation (E) and maximization (M) steps. The expectation (E) step computes the expected value of the log-likelihood using the current estimate for the latent variables. The maximization (M) step computes the parameters that maximize the expected log-likelihood found on the E step. These parameter values determine the latent variable distribution in the next E step. In Gaussian Mixture Models, the EM algorithm is applied to find a set of Gaussian distributions that best fits a set of data points. The E step computes the expected membership of each data point to one of the Gaussian distributions, while the M step computes the parameters for each distribution given the memberships computed in the E step. Then, the E step is repeated with the updated parameters to recompute new membership values. The entire procedure is repeated until model parameters converge.

Despite the mathematical differences, the concept behind ICE is essentially the same. The problem of object recognition in the presence of severe clutter and/or repeated objects can be interpreted as one of estimation of model parameters—the pose of a set of objects—where the model depends on unobserved latent variables—the correspondences of image features to particular object instances. Under this perspective, the *Clustering* step of ICE computes the expected membership of each local feature to one of the object instances, while the *Estimation* step computes the best object poses given the feature memberships computed in the *Clustering* step. Then, the entire procedure is repeated until convergence. If our object models were Gaussian distributions, ICE and GMMs would be virtually equivalent.

4.3 The MOPED Framework

This section contains a brief summary of the MOPED framework and its components. Each individual component is explored in depth in subsequent sections.

The steps itemized below compose the basic MOPED framework for the typical requirements of a robotics application. In essence, MOPED is composed of a single feature extraction and matching step per image, and multiple iterations of Iterative Clustering-Estimation (ICE) that efficiently perform object recognition and pose estimation per object in a bottom-up approach. Assuming the most common setup of object recognition, utilizing a single or a small set of images (i.e., less than 10), we fix ICE to compute two full Clustering-Estimation iterations plus a final cluster merging to remove potential multiple detections that might have not yet converged. This way, we ensure a good trade-off between high recognition rate and reduced system latency, but a greater number of iterations should be considered if working with a larger set of simultaneous images. We show the effect of each MOPED step in Fig. 4.2.

1. Feature Extraction. Salient features are extracted from each image. We represent images $\mathbf{I}$ as sets of local features $\mathbf{f}_m$. Each image $I_m \in \mathbf{I}$ is processed independently, so that

$$I_m = \{K_m, T_m, \mathbf{f}_m\} \quad \mathbf{f}_m = \text{FeatExtract}(\mathbf{g}_m). \tag{4.4}$$

Each individual local feature $f_{j;m}$ from image m is defined by a 2D point location $p_{j;m} = [x,\ y]^T$ and its corresponding feature descriptor $d_{j;m}$; that is,

$$\mathbf{f}_m = \{f_{1;m}, \dots, f_{j;m}, \dots, f_{J;m}\} \quad f_{j;m} = \{p_{j;m}, d_{j;m}\}. \tag{4.5}$$

We define the union of all extracted local features from all images m as $\mathbf{f} = \bigcup_{m=1}^{M} \mathbf{f}_m$.

2. Feature Matching. One-to-one correspondences are created between extracted features in the image set and object features stored in the database. For efficiency, approximate matching techniques can be used, at the cost of a decreased recognition rate. Let C be a correspondence between an image feature $f_{j;m}$ and a model feature $F_{i;o}$, such that

$$C^o_{j,m} = \begin{cases} (f_{j;m}, F_{i;o}), & \text{if } f_{j;m} \leftrightarrow F_{i;o} \\ \emptyset, & \text{otherwise} \end{cases}. \tag{4.6}$$

The set of correspondences for a given object o and image m is represented as $\mathbf{C}^o_m = \bigcup_{\forall j} C^o_{j;m}$. The sets of correspondences $\mathbf{C}_m$ and $\mathbf{C}^o$ are defined equivalently as $\mathbf{C}_m = \bigcup_{\forall j,o} C^o_{j;m}$ and $\mathbf{C}^o = \bigcup_{\forall j,m} C^o_{j;m}$.

3. Feature Clustering. Features matched to a particular object are clustered in image space (x, y), independently for each image. Given that spatially close features are more likely to belong to the same object instance, we cluster the sets of feature locations

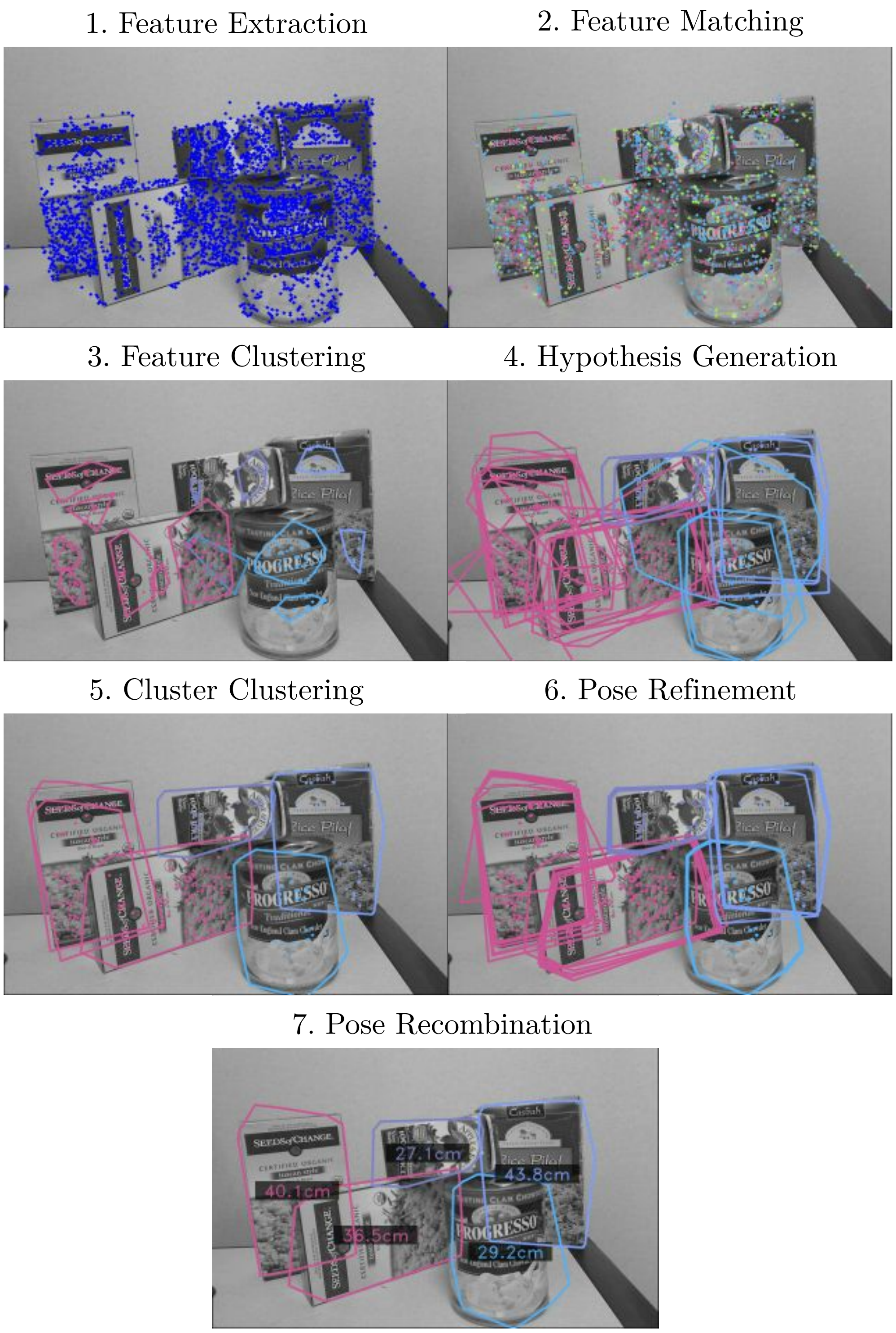


Figure 4.2: Effect of each MOPED step (best viewed in color). Each color in steps 2-7 represents a different object ID in the database.

$\mathbf{p}_m^o \in \mathbf{C}_m^o$, producing a set of clusters that group features spatially close together. We describe this step in detail in Section 4.4.1.

Each cluster $\mathcal{K}_k$ is defined by an object identity o, an image index m, and a subset of the correspondences to object O in image I_m, that is,

$$\mathcal{K}_k = \{o, m, \mathbf{C}_k \subset \mathbf{C}_m^o\}. \tag{4.7}$$

The set of all clusters is expressed as $\mathcal{K}$.

4. Estimation #1: Hypothesis Generation. Each cluster is processed in each image independently in search of objects. RANSAC and Levenberg-Marquardt (LM) are used to find object instances that are loosely consistent with each object's geometry in spite of outliers. The number of RANSAC iterations is high and the number of LM iterations is kept low, so that we discover multiple object hypotheses with coarse pose estimation. At this step, each hypothesis h consists of

$$h = \{o, k, T_h, \mathbf{C}_h \subset \mathbf{C}_k\}, \tag{4.8}$$

where o is the object identity of hypothesis h, k is a cluster index, T_h is a 4×4 transformation matrix that defines the object hypothesis pose with respect to the world reference frame, and $\mathbf{C}_h$ is the subset of correspondences that are consistent with hypothesis h. We describe this step in detail in Section 4.4.2.

5. Cluster Clustering. As the same object might be present in multiple clusters and images, poses are projected from the image set onto a common coordinate frame, and features consistent with a pose are re-clustered. New, larger clusters are created, that often contain all consistent features for a whole object across the entire image set. These new clusters contain

$$\mathcal{K}_K = \{o, \mathbf{C}_K \subset \mathbf{C}^o\}. \tag{4.9}$$

We describe this step in detail in Section 4.4.3 and Section 4.4.4.

6. Estimation #2: Pose Refinement. After Steps 4 and 5, most outliers have been removed, and each of the new clusters is very likely to contain features corresponding to only one instance of an object, spanned across multiple images. The RANSAC procedure is repeated for a low number of iterations, and poses are estimated using LM with a larger number of iterations to obtain the final poses from each cluster that are consistent with the multi-view geometry.

Each multi-view hypothesis H is defined by

$$H = \{o, T_H, \mathbf{C}_H \subset \mathbf{C}_K\}, \tag{4.10}$$

where o is the object identity of hypothesis H, T_H is a 4×4 transformation matrix that defines the object hypothesis pose with respect to the world reference frame, and $\mathbf{C}_H$ is the subset of correspondences that are consistent with hypothesis H. We describe this step in detail in Section 4.4.5.

7. Pose Recombination. A final merging step removes any multiple detections that might have survived, by merging together object instances that have similar poses. A set of hypotheses $\mathbf{H}$, with $H_h = \{o, T_H\}$, is the final output of MOPED. We describe this step in detail in Section 4.4.6.

4.4 Addressing Complexity

In this section, we provide an in-depth explanation of our contributions to address complexity that have been integrated in the MOPED object recognition framework, and how each of our contributions relate to the Iterative Clustering-Estimation procedure.

4.4.1 Image Space Clustering

The goal of *Image Space Clustering* in the context of object recognition is the creation of object priors based solely on image features. In a generic unstructured scene, it is infeasible to attempt the recognition of objects with no higher-level reasoning than the image-model correspondences $C^o_{j,m} = (f_{j;m}, F_{i;o})$. Correspondences for a single object type o may belong to different object instances, or may be matching outliers. Multi-camera setups are even more uncertain, since the amount of image features increases dramatically, and so does the probability of finding multiple repeated objects in the combined set of images. Under these circumstances, the ability to compute a prior over the image features is of utmost importance, in order to evaluate which of the features are likely to belong to the same object instance, and which of them are likely to be outliers.

RANSAC (Fischler and Bolles, 1981) and M-estimators are often the methods of choice to find models in the presence of outliers. However, both of them fail in the presence of heavy clutter and multiple objects, in which only a small percentage of the matched correspondences belong to the same object instance. To overcome this limitation, we propose the creation of object priors based on the density of correspondences across the image, by exploiting the assumption that areas with a higher concentration of correspondences for a given model are more likely to contain an object than areas with very few features. Therefore, we aim to create subsets of correspondences within each image that are reasonably close together and assume they are likely to belong to the same object instance, avoiding the waste of computation time in trying to relate features spread all across the image. We can accomplish this goal by seeking the modes of the density distribution of features in image space. A well-known technique for this task is Mean Shift clustering (Cheng, 1995), which is a particularly good choice for MOPED because no fixed number of clusters needs to be specified. Instead, a radius parameter needs to be chosen, that selects how close two features must in order to be part of the same cluster. Thus, for each object in each image independently, we cluster the set of feature locations $\mathbf{p} \in \mathbf{C}_m^o$ (i.e., pixel positions $p = (x, y)$), producing a set of clusters $\mathcal{K}$ that contain groups of features spatially close together. Clusters that contain very few features, those in which no object can be recognized, are discarded, thus considering the features as outliers and discarding them as well.

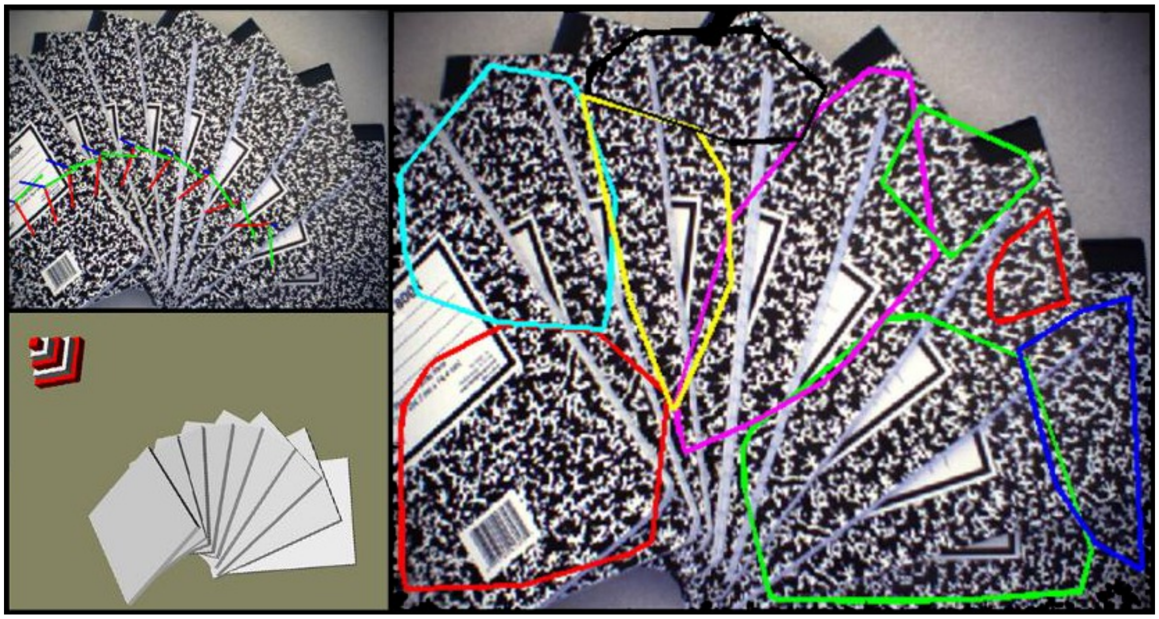


Figure 4.3: Example of highly cluttered scene and the importance of clustering. (Top-left) Scene with 9 overlapping notebooks. (Bottom-left) Recovered poses for notebooks with MOPED. (Right) Clusters of features in image space.

The advantage of using Mean Shift clusters as object priors is illustrated in Fig. 4.3. In Fig. 4.3(top-left) we see an image with 9 notebooks. As a simple example, let us imagine that all notebooks have the same number of correspondences, and that 70% of those correspondences are correct, i.e., that the global inlier ratio $w = \frac{\#\text{ inliers}}{\#\text{ points}} = 0.7$. The inlier ratio for a particular notebook is then $w_{\text{obj}} = \frac{w}{\#\text{ obj}} = 0.0778$. The number of iterations k theoretically required (Fischler and Bolles, 1981) to find one particular instance of a notebook with probability p is

$$k = \frac{log(1-p)}{log(1-(w_{\text{obj}})^n)}, \tag{4.11}$$

where n is the number of inliers for a successful detection. If we require $n = 5$ inliers and a probability $p = 0.95$ of finding a particular notebook, then we should perform $k = 1.05$ million iterations of RANSAC. On the other hand, clustering the correspondences in smaller sets as in Fig. 4.3(right) means that fewer notebooks (at most 3) are present in a given cluster. In such a scenario, finding one particular instance of a notebook with 95% probability requires 16, 586 and 4386 iterations when 1, 2 and 3 notebooks, resp., are present in a cluster, at least three orders of magnitude lower than the previous case.

4.4.2 Estimation #1: Hypothesis generation

In the first *Estimation* step of ICE, our goal is to generate coarse object hypotheses from clusters of features, so that the object poses can be used to refine the cluster associations. In general, each cluster $\mathcal{K}_k = \{o, m, \mathbf{C}_k \subset \mathbf{C}_m^o\}$ may contain features from multiple object hypotheses as well as matching outliers. In order to handle the inevitable presence of matching outliers, we use the robust estimation procedure RANSAC. For a given cluster $\mathcal{K}_k$, we choose a subset of correspondences $\mathbf{C} \subset \mathbf{C}_k$ and estimate an object hypothesis with the best pose that minimizes the sum of reprojection errors (see Eq. (A.1)). We minimize the sum of reprojection errors via a standard Levenberg-Marquardt (LM) non-linear least squares minimization. If the amount of correspondences in $\mathbf{C}_k$ consistent with the hypothesis is higher than a threshold ϵ, we create a new object instance and refine the estimated pose using all consistent correspondences in the optimization. We then repeat this procedure until the amount of unallocated points is lower than a threshold, or the maximum number of iterations has been exceeded. By repeating this procedure for all clusters in all images and objects, we produce a set of hypotheses $\mathbf{h}$, where each hypothesis $h = \{o, k, T_h, \mathbf{C}_h \subset \mathbf{C}_k\}$.

At this stage, we wish to obtain a set of coarse pose hypotheses to work with, as fast as possible. We require a large number of RANSAC iterations to detect as many object hypotheses as possible, but we can use a low maximum number of LM iterations and loose threshold ϵ when minimizing the sum of reprojection errors. Accurate pose will be achieved in later stages of MOPED. The initialization of LM for pose estimation can be implemented with either fast *PnP* solvers such as the ones proposed in Collet et al. (2009), Lepetit et al. (2008) or even completely at random within the space of possible poses (e.g., some distance in front of the camera). Random initialization is the default choice for MOPED, as we found it to be more robust to the pose ambiguities that sometimes confuse *PnP* solvers (particularly in planar objects, as described in Schweighofer and Pinz (2006)).

4.4.3 Hypothesis Quality Score

It is useful at this point to introduce a robust metric to quantitatively compare the goodness of multiple object hypotheses. The desired object hypothesis metric should favor hypotheses with:

- The most amount of consistent correspondences.
- The minimum distance error in each of the correspondences.

The sum of reprojection errors in Eq. (A.2) is not a good evaluation metric according to these requirements, as this error is bound to increase whenever an extra correspondence is added. Therefore, the sum of reprojection errors favors hypotheses with the minimum amount of correspondences, which can lead to choosing spurious hypotheses over more desirable ones.

The average reprojection error exhibits similar issues, as it does not consider the amount of consistent correspondences in the objective function. As a result, there is no incentive to detect entire objects, as local patches within an object can always be detected with lower reprojection error than the entire objects. This effect is particularly evident in large objects, or in objects in which the rigidity assumption is compromised (e.g., that have been slightly bent or deformed over time). Table 4.1 compares these error metrics in MOPED.

In contrast, we define a robust estimator based on the Cauchy distribution that balances the two criteria stated above. Consider the set of consistent correspondences $\mathbf{C}_h$ for a given object hypothesis, where each correspondence $C_j = (f_{j;m}, F_{i;o})$. Assume the corresponding features in C_j have locations in an image p_j and in an object model P_j. Let $d_j = d(p_j, T_h P_j)$ be an error metric that measures the distance between a 2D point in an image and a 3D point from hypothesis h with pose T_h (e.g., reprojection, backprojection errors). Then, the Cauchy distribution $\psi(d_j)$ is defined by

$$\psi(d_j) = \frac{1}{1 + \left(\frac{d_j}{\sigma}\right)^2}, \tag{4.12}$$

where σ^2 parameterizes the cut-off distance at which $\psi(d_j) = 0.5$. In our case, the distance metric d_j is the reprojection error measured in pixels. This distribution is maximal when $d_j = 0$, i.e., $\psi(0) = 1$, and monotonically decreases to zero when a pair of correspondences are infinitely away from each other, i.e., $\psi(\infty) = 0$. The Quality Score Q for a given object hypothesis h is then defined as a summation over the Cauchy scores $\psi(d_j)$ for all correspondences:

$$Q(h) = \sum_{\forall j : C_j \in \mathbf{C}_h} \psi(d_j) = \sum_{\forall C_j \in \mathbf{C}_h} \frac{1}{1 + \frac{d^2(p_j, T_h P_j)}{\sigma^2}}. \tag{4.13}$$

The Q-Score has a lower bound at 0, if a given hypothesis has no correspondences or if all its correspondences have infinite error, and has an upper bound at $|O|$, which is the total number of correspondences for model O. This score allows us to reliably rank our object hypotheses and evaluate their strength.

The cut-off distance σ may be either a fixed value, or adjusted at each iteration via robust estimation techniques (Zhang, 1997), depending on the application. Robust estimation techniques require a certain minimum outlier/inlier ratio to work properly ($\frac{\#\text{ outliers}}{\#\text{ inliers}} < 1$ in all cases), known as the *breaking point* of a robust estimator (Huber, 1981). In the case of MOPED, the outlier/inlier ratio is often well over the breaking point of any robust estimator, especially when multiple instances of an object are present; as a consequence, robust estimators might result in unrealistically large values of σ in complex scenes. Therefore, we choose to set a fixed value for the cut-off parameter, $\sigma = 2$ pixels, for a good balance between encouraging a large number of correspondences while keeping their reprojection error low.

4.4.4 Cluster Clustering

The disadvantage of separating the image search space into a set of clusters is that the produced pose hypotheses may be generated with only partial information from the scene, given that information from other clusters and other views is not considered in the initial *Estimation* step of ICE. However, once a rough estimate of the object poses is known, we can merge the information from multiple clusters and multiple views to obtain sets of correspondences that contain all features from a single object instance (see Fig. 4.1).

Multiple alternatives are available to group similar hypotheses into clusters. In this section, we propose a novel hypothesis clustering algorithm called *Projection Clustering*, in which we perform correspondence-level grouping from a set of object hypotheses $\mathbf{h}$ and provide a mechanism to robustly filter any pose outliers.

For comparison, we introduce a simpler Cluster Clustering scheme based on Mean Shift, and analyze the computational complexity of both schemes to conclude in which cases we might prefer one over the other.

Mean Shift clustering on pose space

A straightforward hypothesis clustering scheme is to perform Mean Shift clustering on all hypotheses $h = \{o, k, T_h, \mathbf{C}_h \subset \mathbf{C}_k\}$ for a given object type o. In particular, we cluster the pose hypotheses T_h in pose space. In order to properly measure distances between poses, it is convenient to parameterize rotations in terms of quaternions and project them in the same half of the quaternion hypersphere prior to clustering, using then Mean Shift on the resulting 7-dimensional poses. After this procedure, we merge the correspondence clusters

$\mathbf{C}_h$ of those poses that belong to the same pose cluster $\mathcal{T}_K$,

$$\mathbf{C}_K = \bigcup_{T_h \in \mathcal{T}_K} \mathbf{C}_h. \tag{4.14}$$

This produces clusters $\mathcal{K}_K = \{o, \mathbf{C}_K \subset \mathbf{C}^o\}$ whose correspondence clusters span over multiple images. In addition, the centroid of each pose cluster $\mathcal{T}_K$ can be used as initialization for the following *Estimation* iteration of ICE. At this point, we can discard all correspondences not consistent with any pose hypothesis, thus filtering many outliers and reducing the search space for future iterations of ICE. The computational complexity of Mean Shift is $\mathcal{O}(dN^2t)$, where $d = 7$ is the dimensionality of the clustered data, $N = |\mathbf{h}|$ is the total number of hypotheses to cluster, and t is the number of iterations that Mean Shift requires. In practice, the number of hypotheses is often fairly small, and $t \leq 100$ in our implementation.

Projection Clustering

Mean Shift provides basic clustering in pose space, and works well when multiple correct detections of an object are present. However, it is possible that spurious false positives are detected in the hypothesis generation step. It is important to realize that these false positives are very rarely exclusively due to random outliers in the feature matching process. To the contrary, most outlier detections are artifacts of the projection of a 3D scene into a 2D image when captured by a perspective camera. In particular, we can distinguish two different cases:

- A group of correct matches whose 3D configuration is degenerate or near-degenerate (e.g., a group of 3D points that are almost collinear), incorrectly grouped with one single matching outlier. In this case, the output pose is largely determined by the location of the matching outlier, which causes arbitrarily erroneous hypotheses to be accepted as correct.
- Pose ambiguities in objects with planar surfaces. The sum of reprojection errors in planar surfaces may contain two complementary local minima in some configurations of pose space (Schweighofer and Pinz, 2006), which often causes the appearance of two distinct and partially overlapping object hypotheses. These hypotheses are usually too distant in pose space to be grouped together.

The false positives output in the pose hypothesis generation are often too distant from any correct hypothesis in the scene, and cannot be merged using regular clustering techniques (e.g., Mean Shift). However, the point features that generated those false positives are usually correct, and they can provide valuable information to some of the correct object hypotheses. In Projection Clustering, we process each point feature individually, and assign them to the strongest pose hypothesis to which they might belong. Usually, spurious poses only contain a limited number of consistent point features, thus resulting in lower Q-scores (Section 4.4.3) than correct object hypotheses. By transferring most of the point features to the strongest object hypotheses, we not only utilize the extra information available in the scene for increased accuracy, but also filter most false positives by lowering their number of consistent points below the required minimum.

The first step in Projection Clustering is the computation of Q-Scores for all object hypotheses $\mathbf{h}$. We generate a pool of potential correspondences $\mathbf{C}_o$ that contains all correspondences for a given object type that are consistent with at least one pose hypothesis. Correspondences from all images are included in $\mathbf{C}_o$. We compute the Quality score for each hypothesis h from all correspondences $C_j = (f_j, F_j)$ such that $C_j \in \mathbf{C}_o$.

$$Q(h) = \sum_{\forall j : C_j \in \mathbf{C}_o} \psi(d_j) = \sum_{\forall C_j \in \mathbf{C}_o} \frac{1}{1 + \frac{d^2(p_j, T_h P_j)}{\sigma^2}} \tag{4.15}$$

For each potential correspondence C_j in $\mathbf{C}_o$, we define a set of likely hypotheses $\mathbf{h}_j$ as the set of those hypotheses h whose reprojection error is lower than a threshold γ. This threshold can be interpreted as an attraction coefficient; large values of γ lead to heavy transference of correspondence to strong hypotheses, while small values cause few correspondences to transfer from one hypothesis to another. In our experiments, a large threshold γ of 64 pixels is used.

$$h \in \mathbf{h}_j \Longleftrightarrow d^2(p_j, T_h P_j) < \gamma \tag{4.16}$$

At this point, the relation between correspondences C_j and hypotheses h is broken. In other words, we empty the set of correspondences $\mathbf{C}_h$ for each hypothesis h, so that $\mathbf{C}_h = \emptyset$. Then, we re-assign each correspondence $C_j \in \mathbf{C}_o$ to the pose hypothesis h within $\mathbf{h}_j$ with stronger overall Q-Score:

$$\mathbf{C}_h \leftarrow C_j : h = \arg\max_{h \in \mathbf{h}_j} Q(h) \tag{4.17}$$

Finally, it is important to check the remaining number of correspondences that each object hypothesis has after Projection Clustering. Pose hypotheses that retain less than a minimum number of consistent correspondences are considered outliers and therefore discarded.

The Projection Clustering algorithm we propose has a computational complexity $\mathcal{O}(MNI)$, where $M = |\mathbf{C}|$ is the total number of correspondences to process, $N = |\mathbf{h}|$ is the number of pose hypotheses and $I = |\mathbf{I}|$ is the total number of images. Comparing this with the complexity $\mathcal{O}(dN^2t)$ of Mean Shift, we see that the only advantage of Mean Shift in terms of computational complexity would be in the case of object models with enormous numbers of features and many high resolution images, so that $\mathcal{O}(MNI) \gg \mathcal{O}(dN^2t)$. While offering a similar computational complexity, the advantage of *Projection Clustering* with respect to Mean Shift is in terms of improved robustness and outlier detection, both of which are essential for handling increased complexity in MOPED.

Performance Comparison

In this experiment, we compare the recognition performance of MOPED when using the two Cluster-Clustering approaches explained in this section. In addition, and given that Projection Clustering depends on the behavior of a hypothesis ranking mechanism, we implement four different ranking mechanisms and compare their performance. The first ranking mechanism is our own Q-Score introduced in Section 4.4.3, while the other approaches considered are the sum of reprojection errors, the average reprojection error, and the number of consistent correspondences. The performance of MOPED when using the different Cluster-Clustering and hypothesis-ranking algorithms is shown in Table 4.1, on a subset of 100 images from the Simple Movie Benchmark (Section 4.5.2) that contain a total of 1289 object instances.

An object hypothesis is considered a true positive if its pose estimate is within 5 cm and 10 degrees of the ground truth pose. Object hypotheses whose pose is outside this error range are considered false positives. Ground truth objects without an associated true positive are considered false negatives. According to these performance metrics, the appearance of pose ambiguities and misdetections is particularly critical, since they often produce both a false positive –the rotational error is greater than the threshold– and a false negative –no pose agrees with the ground truth–.

The overall best-performer is Projection Clustering when using Q-Score as its hypothesis ranking metric, which correctly recognizes and estimates the pose of 87.6% of the objects in the dataset. Mean Shift is the second best performer, but the increased false positive rate is mainly due to pose ambiguities that cannot be resolved and produce spurious detections. The different hypothesis ranking schemes critically impact the overall performance of Projection Clustering, as multiple poses often need to be merged after the first Clustering-Estimation iteration. Ranking spurious poses over correct poses results in an increased number of false positives, as Projection Clustering is unable to merge object hypothesis properly. While *Q-Score* is able to estimate the best pose out a set of multiple detections, *Reprojection* and *Avg. Reprojection* select sub-optimal poses that contain just a few points, often including outliers. This behavior severely impacts the performance of Projection Clustering, which barely merges any pose hypotheses and produces an increased number of false positives. The *Num. Correspondences* metric prefers hypotheses

	True Pos.	False Pos.	False Neg.
Mean Shift	10.92	3.32	1.97
P.Clust. (Q-Score)	**11.3**	**2.29**	**1.59**
P.Clust. (Reproj)	11.05	7.4	1.84
P.Clust. (Avg Reproj)	10.88	7.81	2.01
P.Clust. (# Corresp)	3.6	9.82	9.29

Table 4.1: Mean Recognition per Image: Mean Shift vs Projection Clustering, in the Simple Movie Benchmark (see Section 4.5.2 for details).

with many consistent matches in the scene, and Projection Clustering merges hypotheses correctly. Unfortunately, the chosen best hypotheses are in most cases incorrect due to ambiguities, estimating only 28% of the poses correctly. It must be noted that in simple scenes with a few unoccluded objects all the evaluated metrics perform similarly well. Projection Clustering and Q-Score, however, showcase increased robustness when working with the most complex scenes.

4.4.5 Estimation #2: Pose Refinement

At the second iteration of ICE, we use the information from initial object hypotheses to obtain clusters that are most likely to belong to a single object instance, with very few outliers. For this reason, the Estimation step at the second iteration of ICE requires only a low number of RANSAC iterations to find object hypotheses. In addition, we use a high number of LM iterations to estimate object poses accurately.

This procedure is equivalent to that of the first *Estimation* step, being the objective function to minimize the only difference between the two. For a given multi-view feature cluster $\mathcal{K}_K$, we perform RANSAC on subsets of correspondences $\mathbf{C} \subset \mathbf{C}_K$ and obtain pose hypotheses $H = \{o, T_H, \mathbf{C}_H \subset \mathbf{C}_K\}$ with consistent correspondences $\mathbf{C}_H \subset \mathbf{C}_K$. The objective function to minimize can be either the sum of reprojection errors (Eq. (A.2)) or the sum of backprojection errors (Eq. (A.7)). In MOPED, we choose the backprojection error because it is simpler to extend to multiple images and depth measurements (as in Chapter 5) than the reprojection error. See Appendix A for a discussion on these two objective functions.

We initialize the non-linear minimization of the chosen objective function using the highest-ranked pose in $\mathbf{C}_K$ according to their Q-Scores. This non-linear minimization is performed, again, with a standard Levenberg-Marquardt non-linear least squares minimization algorithm.

4.4.6 Truncating ICE: Pose Recombination

An optional final step should be applied in case ICE has not converged after two iterations, in order to remove multiple detections of objects. In this case, we perform a full *Clustering* step as in Section 4.4.4, and if any hypothesis $H \in \mathbf{H}$ is updated with transferred correspondences, we perform a final LM optimization that minimizes Eq. (A.7) on all correspondences $\mathbf{C}_H$.

4.5 Addressing Scalability and Latency

In this section, we present multiple contributions to optimize MOPED in terms of scalability and latency. We first introduce a set of four Benchmarks designed to stress test every component of MOPED. Each of our contributions is evaluated and verified on this set of benchmarks.

4.5.1 Baseline system

For comparison purposes, we provide results for a *Baseline* system that implements the MOPED framework with none of the optimizations described in Section 4.5. In each case, we have tried to choose the most widespread publicly available code libraries for each task. In particular, the following configuration is used as the baseline for our performance experiments:

Feature Extraction with an OpenMP-enabled, CPU-optimized version of SIFT we have developed.

Feature matching with a publicly available implementation of ANN by Arya et al. (1998) and 2-NN Per Object (described in Section 4.5.3).

Image Space clustering with a publicly available C implementation of Mean Shift (Dollár and Rabaud, 2010).

Estimation #1 with 500 iterations of RANSAC and up to 100 iterations of the Levenberg-Marquardt (LM) implementation from Lourakis (2004), optimizing the sum of reprojection errors in Eq. (A.1).

Cluster Clustering with Mean Shift, as described in Section 4.4.4.

Estimation #2 with 24 iterations of RANSAC and up to 500 iterations of LM, optimizing the sum of backprojection errors in Eq. (A.7). The particular number of iterations of RANSAC in this step is not a critical factor, as most objects are successfully recognized in the first 10-15 iterations. It is useful, however, to use a multiple of the number of concurrent threads (see Section 4.5.4) for performance reasons.

Pose Recombination with Mean Shift and up to 500 iterations of LM.

4.5.2 Benchmarks

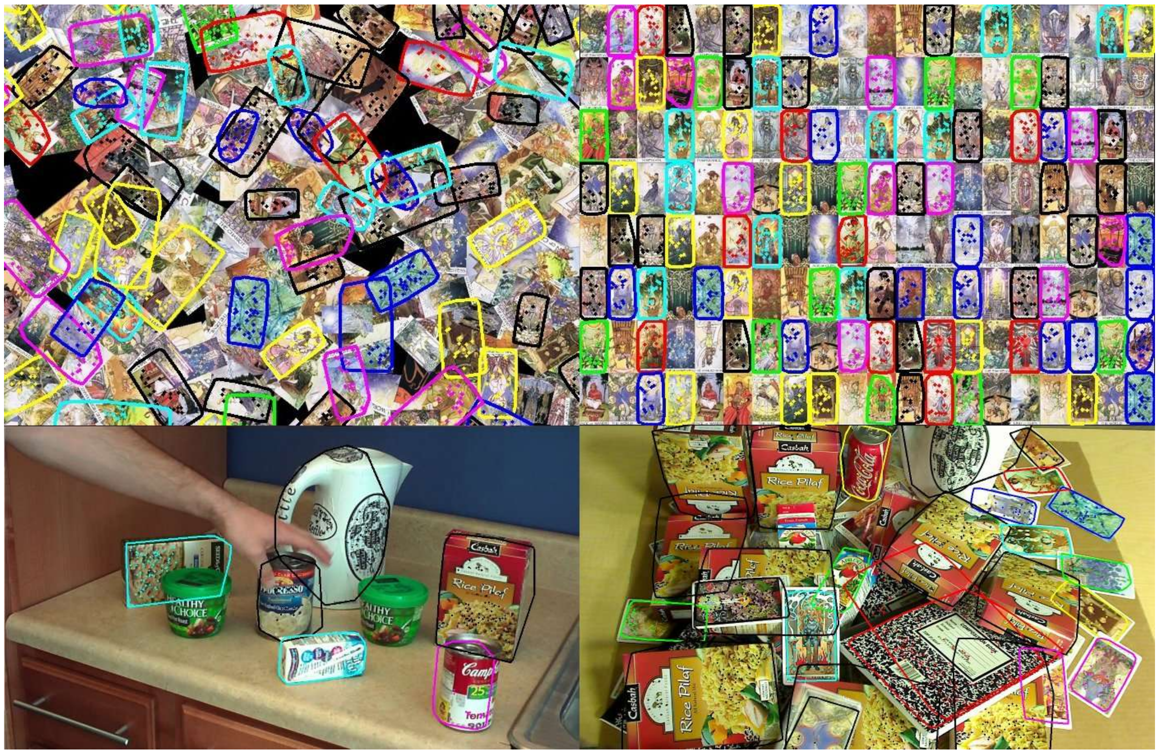


Figure 4.4: MOPED Benchmarks. For the sake of clarity, only half of the detected objects are marked. (a) The Rotation Benchmark: MOPED processes this scene 36.4x faster than the Baseline. (b) The Zoom Benchmark: MOPED processes this scene 23.4x faster than the Baseline. (c) The Simple Movie Benchmark. (d) The Complex Movie Benchmark.

We present four benchmarks (Fig. 4.4) designed to stress test every component of our system. All benchmarks, both synthetic and real-world, provide exclusively a set of images and ground truth object poses (i.e., no synthetically computed feature locations or correspondences). We performed all experiments on a 2.33GHz quad-core Intel(R) Xeon(R) E5345 CPU, 4 GB of RAM and a nVidia GeForce GTX 260 GPU running Ubuntu 8.04 (32 bits).

The Rotation Benchmark

The Rotation Benchmark is a set of synthetic images that contains highly cluttered scenes with up to 400 cards in different sizes and orientations. This benchmark is designed to test MOPED's scalability with respect to the database size, while keeping a constant number of features and objects. We have generated a total of 100 independent images for different resolutions (1400 × 1050, 1000 × 750, 700 × 525, 500 × 375 and 350 × 262). Each image contains from 5 to 80 different objects and up to 400 simultaneous object instances.

The Zoom Benchmark

The Zoom Benchmark is a set of synthetic images that progressively zooms in on 160 cards until only 12 cards are visible. This benchmark is designed to check the scalability of MOPED with respect to the total number of detected objects in a scene. We generated a total of 145 independent images for different resolutions (1400×1050, 1000×750, 700×525, 500×375 and 350×262). Each image contains from 12 to 80 different objects and up to 160 simultaneous object instances. This benchmark simulates a board with 160 cards seen by a 60° FOV camera at distances ranging from 280 mm to 1050 mm. The objects were chosen to have the same number of features at each scale. Each image has over 25000 features.

The Simple Movie Benchmark

Synthetic benchmarks are useful to test a system in controlled conditions, but are a poor estimator of the performance of a system in the real world. Therefore, we provide two real-world scenarios for algorithm comparison. The Simple Movie Benchmark consists of a 1900-frame movie at 1280 x 720 resolution, each image containing up to 18 simultaneous object instances.

The Complex Movie Benchmark

The Complex Movie Benchmark consists of a 3542-frame movie at 1600 x 1200 resolution, each image containing up to 60 simultaneous object instances. The database contains 91 models and 47342 SIFT features when running this benchmark. It is noteworthy that the scenes in this video present particularly complex situations, including: several objects of the same model contiguous to each other, which stresses the clustering step; overlapping partially-occluded objects, which stresses RANSAC; and objects in particularly ambiguous poses, which stresses both LM and the merging algorithm, that encounter difficulties determining which pose is preferable.

4.5.3 Feature Matching

Once a set of features have been extracted from input image(s), we must find correspondences between the image features and our object database. Matching is done as a nearest neighbor search in the 128-dimensional space of SIFT features. An average database of 100 objects can contain over 60,000 features. Each input image, depending on the resolution and complexity of the scene, can contain over 10,000 features.

The feature matching step aims to create one-to-one correspondences between model and image features. The feature matching step is, in general, the most important bottleneck for model-based object recognition to scale to large object databases. In this section, we propose and evaluate different alternatives to maximize scalability with respect to the number of objects in the database, without sacrificing accuracy. The extension of the matching search space is, in this case, the balancing factor between accuracy and speed when finding nearest neighbors.

There are no known exact algorithms for solving the matching problem that are faster than linear search. Approximate algorithms, on the other hand, can provide massive speedups at the cost of a decreased matching accuracy, and are often used wherever speed is an issue. Many approximate approaches for finding the nearest neighbors to a given feature are based on *kd-trees* (e.g., ANN (Arya et al., 1998), randomized kd-trees (Silpa-Anan and Hartley, 2008), FLANN (Muja and Lowe, 2009)) or hashing (e.g., LSH (Andoni and Indyk, 2006)). A ratio test between the first two nearest neighbors is often performed for outlier rejection. We analyze the different alternatives in which these techniques are often applied to feature matching, and propose an intermediate solution that achieves a good balance between recognition rate and system latency.

2-NN per Object

On the one end, we can compare the image features against each model independently. If using e.g., ANN, we build a kd-tree for each model in the database once (off-line), and we match each of them against every new image. This process entails a complexity of $\mathcal{O}(|\mathbf{f}_m||\mathbf{O}|\log(|\bar{\mathbf{F}}_o|))$, where $|\mathbf{f}_m|$ is the number of features on the image, $|\mathbf{O}|$ the number of models in the database, and $|\bar{\mathbf{F}}_o|$ the mean number of features for each model. When $|\mathbf{O}|$ is large, this approach is vastly inefficient as the cost of accessing each kd-tree dominates the overall search cost. The search space is in this case very limited, and there is no degradation in performance when new models are added to the database. We refer to it as *OBJ_MATCH*.

2-NN per Database

On the other end, we can compare the image features against the whole object database. This alternative, which we term *DB_MATCH*, builds just one kd-tree containing the features from all models. This solution has a complexity of $\mathcal{O}(|\mathbf{f}_m|\log(|\mathbf{O}||\bar{\mathbf{F}}_o|))$. The search space is in this case the whole database. While this approach is orders of magnitude faster than the previous one, every new object added to the database degrades the overall recognition performance of the system due to the presence of similar features in different objects. We provide it as a baseline because it is the fastest matching approach for large object databases.

# Corresp:	After Matching	After clustering	Final
GPU	3893.7	853.2	562.1
OBJ_MATCH	3893.6	712.0	449.2
DB_MATCH	1778.4	508.8	394.7
MOPED-90	3624.9	713.6	428.9

	Matching Time(ms)	Objects Found
GPU	253.34	8.8
OBJ_MATCH	498.586	8.0
DB_MATCH	129.85	7.5
MOPED-90	140.36	8.2

Table 4.2: Feature matching algorithms in the Simple Movie Benchmark. GPU used as performance baseline, as it computes exact nearest neighbors.

k-NN per Database

Alternatively, one can consider the closest k nearest neighbors instead (with $k > 2$). k-ANN implementations using kd-trees can provide more neighbors without significantly increasing their computational cost, as they are often a byproduct of the process of obtaining the nearest neighbor. An intermediate approach to the ones presented before is the search for k multiple neighbors in the whole database. If two neighbors from the same model are found, the distance ratio is then applied to the 2 nearest neighbors from the same model. If the nearest neighbor is the only neighbor for a given model, we apply the distance ratio with the second nearest neighbor to avoid spurious correspondences. This algorithm (with $k = 90$) is the default choice for MOPED.

Brute Force on GPU

The advent of GPUs and their many-core architecture allows the efficient implementation of an exact feature matching algorithm. The parallel nature of the brute force matching algorithm suits the GPU, and allows it to be faster than the ANN approach when $|\mathbf{O}|$ is not too large. Given that this algorithm scales linearly with the number of features instead of logarithmically, we can match each model independently without performance loss.

Performance comparison

Fig. 4.5 compares the cost of the different alternatives on the Rotation Benchmark. *OBJ_MATCH* and GPU scale linearly with respect to $|\mathbf{O}|$, while *DB_MATCH* and MOPED-k scale almost logarithmically. We show MOPED-k using $k = 90$ and $k = 30$. The value of k adjusts the speed and quality behavior of MOPED between *OBJ_MATCH* ($k = \infty$) and *DB_MATCH* ($k = 2$). The recognition performance of MOPED-k when using the different strategies is shown in Table 4.2. GPU provides an upper bound for the object recognition, as it is an exact search method. *OBJ_MATCH* comes closest in raw matching accuracy with MOPED-90 a close second. However, the number of objects detected are nearly the same. The matching speed of MOPED-90 is, however, significantly better than *OBJ_MATCH*. Feature matching in MOPED-90 thus provides a significant performance increase without sacrificing much accuracy.

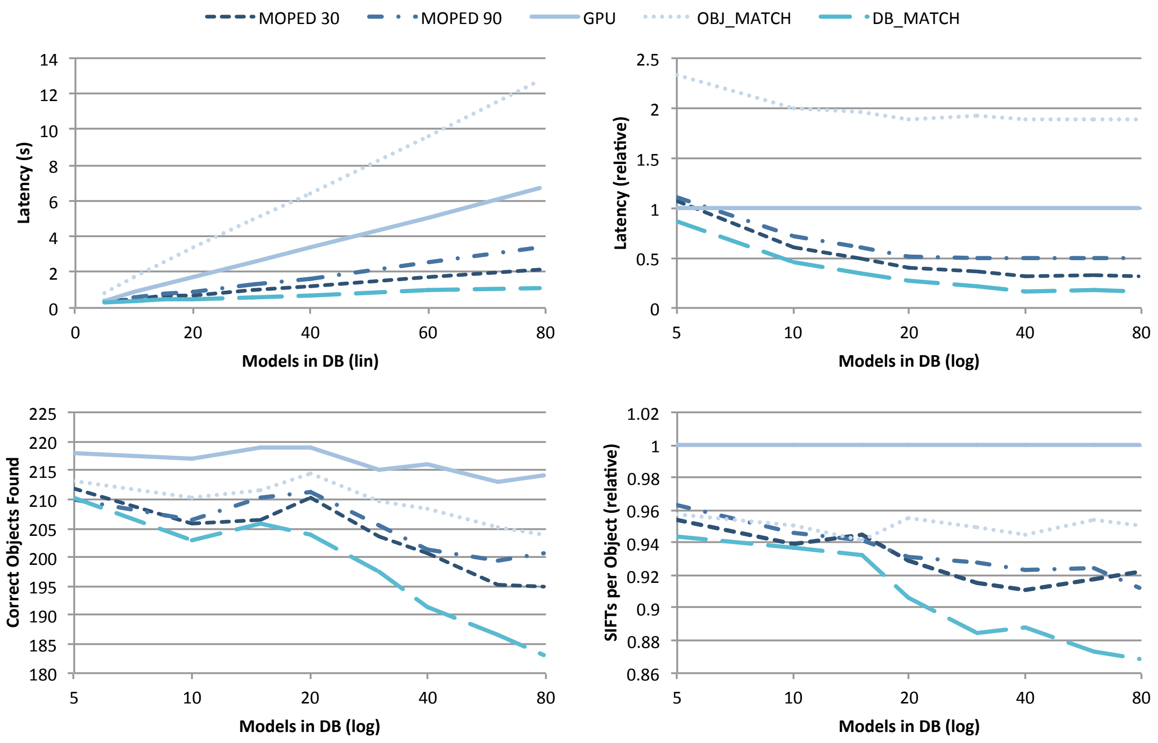


Figure 4.5: Scalability of feature matching algorithms with respect to the database size, in the Rotation Benchmark at 1400×1050 resolution.

4.5.4 Architecture Optimizations

Our algorithmic improvements were focused mainly on boosting the scalability and robustness of the system. The architectural improvements of MOPED are obtained as a result of an implementation designed to make the best use of all the processing resources of standard compute hardware. In particular, we use GPU-based processing, intra-core parallelization using SIMD instructions, and multi-core parallelization. We have also carefully optimized the memory subsystem, including bandwidth transfer and cache management.

All optimizations have been devised to reduce the latency between the acquisition of an image and the output of the pose estimates, to enable faster response times from our robotic platform.

GPU and Embarrassingly Parallel Problems

State-of-the-art CPUs, such as the Intel Core i7 975 Extreme, can achieve a peak performance of 55.36 GFLOPS, according to the manufacturer (Intel Corp., 2010). State-of-the-art GPUs, such as the ATI Radeon HD 5900, can achieve a peak performance of 4640 SP GFLOPS (AMD, 2010).

To use GPU resources efficiently, input data needs to be transferred to the GPU memory. Then, algorithms are executed simultaneously on all shaders, and finally recover the results from the GPU memory. As communication between shaders is expensive, the best GPU-performing algorithms are those that can be divided evenly into a large number of simple tasks. This class of easily separable problems is called *Embarrassingly Parallel Problems (EPP)* (Wilkinson and Allen, 2004).

GPU-Based Feature Extraction. Most feature extraction algorithms consist of an initial keypoint detection step followed by a descriptor calculation for each keypoint, both of which are EPP. Keypoint detection algorithms can process each pixel from the image independently. They may need information about neighboring pixels, but they do not typically need results from them. After obtaining the list of keypoints, the respective descriptors can also be calculated independently.

In MOPED, we consider two of the most popular locally invariant features: SIFT (Lowe, 2004) and SURF (Bay et al., 2008). SIFT features have proven to be among the best-performing invariant descriptors in the literature (Mikolajczyk and Schmid, 2005), while SURF features are considered to be a fast alternative to SIFT. MOPED uses *SIFT-GPU* (Wu, 2007) as its main feature extraction algorithm. If compatible graphics hardware is not detected MOPED automatically reverts back to performing SIFT extraction on the CPU, which is an OpenMP-enabled, CPU-optimized version of SIFT we have developed. A GPU-enabled version of SURF, *GPU-SURF* (Cornelis and Van Gool, 2008), is used for comparison purposes.

We evaluate the latency of the three implementations in Fig. 4.6. The comparison is as expected: GPU versions of both SIFT and SURF provide tremendous improvements over their non-GPU counterparts. Table 4.3 compares the object recognition performance of SIFT and SURF: SURF proves to be 2.59x faster than SIFT at the cost of detecting 54% less objects. In addition, the performance gap between both methods decreases significantly as image resolution increases, as shown in Fig. 4.6. For MOPED, we consider SIFT to be almost always the better alternative when balancing recognition performance and system latency.

	Latency (ms)	Recognized Objects
SIFT	223.072	13.83
SURF	86.136	6.27

Table 4.3: SIFT vs. SURF: Mean Recognition Performance per Image in Zoom Benchmark.

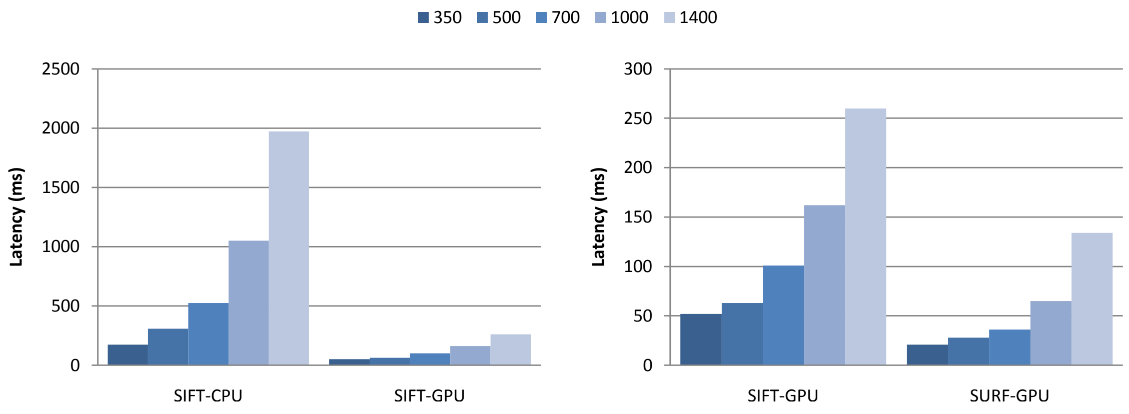


Figure 4.6: SIFT-CPU vs. SIFT-GPU vs. SURF-GPU, in the Rotation Benchmark at different resolutions. (left) SIFT-CPU vs. SIFT-GPU: 658% speed increase in SIFT extraction on GPU. (right) SIFT-GPU vs. SURF-GPU: 91% speed increase in SURF over SIFT at the cost of lower matching performance.

GPU Matching. Performing feature matching in the GPU requires a different approach than the standard Approximate Nearest Neighbor techniques. Using ANN, each match involves searching in a kd-tree, which requires fast local storage and a heavy use of branching that are not suitable for GPUs.

Instead of using ANN, Wu (2007) suggests the use of brute force nearest neighbor search on the GPU, which scales quite well as vector processing matches the GPU structure quite well. In Fig. 4.5, brute force GPU matching is shown to be faster than per-object ANN and provide better quality matches because it is not approximate. As graphics hardware becomes cheaper and more powerful, brute-force feature matching in large databases might become the most sensible choice in the near future.

Intra-core optimizations

SSE instructions allow MOPED to perform 12 floating point instructions per cycle instead of just one. The 3D to 2D projection function, critical in the pose estimation steps, is massively improved by using SSE-specific algorithms from Van Weveren (2005) and Conte et al. (2000).

The memory footprint of MOPED is very lightweight for current computers. In the case of a database of 100 models and a total of 102.400 SIFT features, the required memory is less than 13MB. Runtime memory footprint is also small: a scene with 100 different objects with 100 matched features each would require less than 10 MB of memory to be processed. This is possible thanks to using dynamic and compact structures, such as lists and sets, and removing unused data as soon as possible. In addition, SIFT descriptors are stored as integer numbers in a 128-byte array instead of a 512-byte array. Cache performance has been greatly improved due to the heavy use of memory-aligned and compact data structures (Dysart et al., 2004).

The main data structures are kept constant throughout the algorithm, so that no data needs to be copied or translated between steps. k-ANN feature matching benefits from compact structures in the kd-tree storage, as smaller structures increase the probability of staying in the cache for faster processing. In image space clustering, the performance of Mean Shift is boosted 250 times through the use of compact data structures.

The overall performance increase is over 67% in CPU processing tasks (see Fig. 4.7).

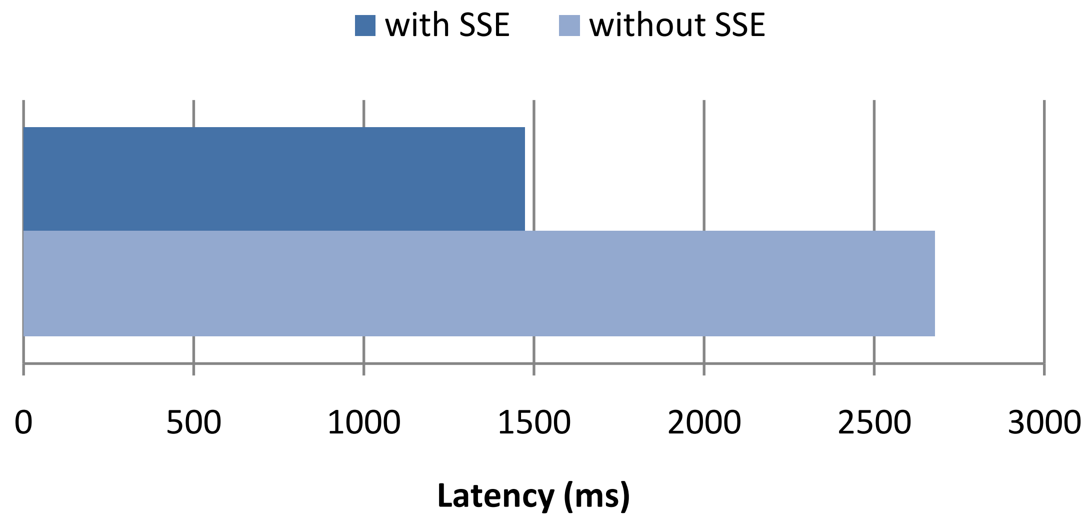


Figure 4.7: SSE performance improvement in the Complex Movie Benchmark. Time per frame without counting SIFT extraction.

Symmetric Multiprocessing

Symmetric Multiprocessing (SMP) is a multiprocessor computer architecture with identical processors and shared memory space. Most multi-core based computers are SMP systems. *OpenMP* is a standard framework for multi-processing in SMP systems that we implement in MOPED.

We use *Best Fit Decreasing* (Johnson, 1974) to balance the load between cores using the size of a cluster as an estimate of its processing time, given that each cluster of features can be processed independently. Tests on a subset of 10 images from the Complex Movie Benchmark show performance improvements of 55% and 174% on dual and quad core CPUs respectively, as seen in Fig. 4.8.

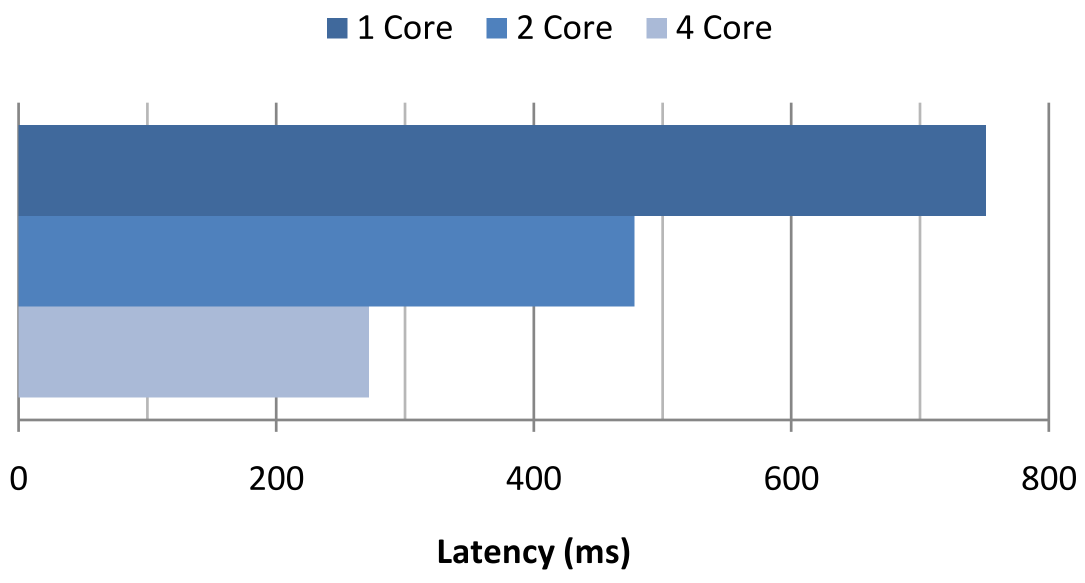


Figure 4.8: Performance improvement of *Pose Estimation* step in multi-core CPUs, in the Complex Movie Benchmark.

Multi-Frame Scheduling

In order to maximize the system throughput, MOPED can benefit from GPU-CPU pipeline scheduling (Chatha and Vemuri, 2002). In order to use all available computing resources, a second execution thread can be added, as shown in Fig. 4.9. However, the GPU and CPU execution times are not equal in real scenes, and one of the execution threads often needs to wait for the other (see Fig. 4.10). The impact of pipeline scheduling depends heavily on image resolution, as shown in Fig. 4.11, because the GPU and CPU loads do not increase at the same rate when increasing the number of features but keeping the same number of objects in each scene. In MOPED, pipeline scheduling may increase latency significantly,

	FPS	Latency	Latency Sd
Sched. MOPED	**3.49**	368.45	92.34
Non-Sched. MOPED	2.70	**303.23**	**69.26**
Baseline	0.47	2124.30	286.54

Table 4.4: Impact of pipeline scheduling, in the Simple Movie Benchmark. Latency measurements in ms.

especially if using high resolution images, but also increases throughput almost two-fold. Since GPU processing is the bottleneck on very small resolutions, these are the best scenarios for pipeline scheduling. For example, as seen in Fig. 4.11, at a lower resolution of 500×380, throughput is increased by 95.6% and latency is increased by 9%.

We further test the impact of pipeline scheduling in a real sequence in the Simple Movie Benchmark, in Table 4.4. The average throughput of the overall system is increased by 25% when using pipeline scheduling, at the cost of 21.5% more latency. In addition, we see the average system latency fluctuates 33.2% more when using pipeline scheduling. In our particular application, latency is a more critical factor than throughput, as our robot HERB (Srinivasa et al., 2010) must interact with the world in real time. Therefore, we choose not to use pipeline scheduling in our default implementation of MOPED (and in the experiments displayed in this chapter). In general, pipeline scheduling should be implemented in any kind of off-line process, or whenever latency is not a critical factor in object recognition.

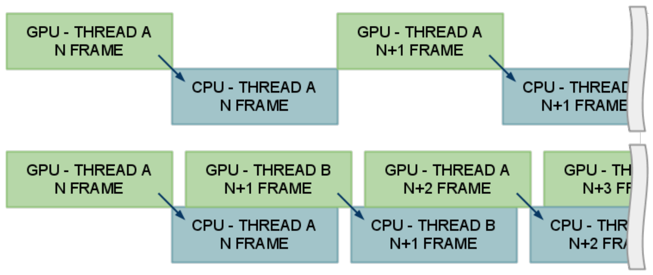


Figure 4.9: (top) Standard MOPED uses the GPU to obtain the SIFT features, then the CPU to process them. (bottom) Addition of a second execution thread does not substantially increase the system latency.

4.5.5 Performance evaluation

In this section, we evaluate the impact of our combined optimizations on the overall performance of MOPED, compared to the Baseline system in Section 4.5.1, and we analyze how

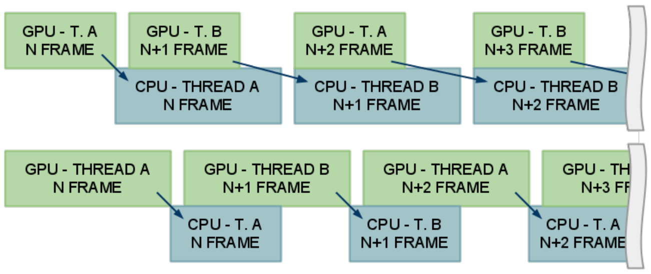


Figure 4.10: (top) Limiting factor: CPU. GPU thread processing frame N+1 must wait for CPU processing frame N to finish, increasing latency. (bottom) Limiting factor: GPU. No substantial increase in latency.

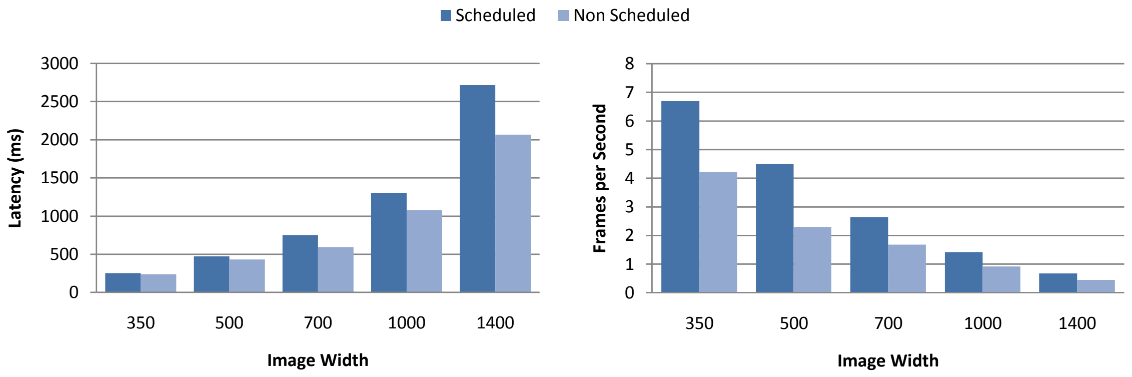


Figure 4.11: Impact of image resolution in Pipeline Scheduling, in the Rotation Benchmark. (left) Latency comparison (ms). (right) Throughput comparison (FPS).

the application of our optimizations improves system latency and scalability.

Testing both systems on the Simple Movie Benchmark (Table 4.4), MOPED outperforms the baseline with a 5.74x increase in throughput and a 7.01x decrease in latency. These improvements become more acute the greater the scene complexity is, Our architectural optimizations offer improvement even with the simple scenes, but the overhead of managing the SMP and GPU processing is large enough to limit the improvement. However, in scenes with high complexity (and, therefore, with high number of features and objects) this overhead is negligible, resulting in massive performance boosts. In the Complex Movie Benchmark, MOPED shows an average throughput of 0.44 frames per second and latency of 2173.83 ms, a 30.1x performance increase over the 0.015 frames per second and 65568.20 ms of average latency of the Baseline system.

It is also interesting to compare the Baseline and MOPED in terms of system scalability. We are most interested in the scalability with respect to image resolution, number of objects in a scene and number of objects in the database. Our synthetic benchmarks allow for a controlled comparison of these different parameters without affecting the others.

The Rotation Benchmark contains images with a constant number of object instances at 5 different resolutions. Fig. 4.12(left) shows that both MOPED and the Baseline system scale linearly in execution time with respect to image resolution, i.e., quadratically with respect to image width. To be more accurate, the implementation of ICE in both systems allows their performance to increase linearly with the number of feature clusters. The number of SIFT features and feature clusters also increase linearly with respect to image resolution in this Benchmark.

To test the scalability with respect to the number of objects in the database, images from the Rotation Benchmark are generated to have a fixed number of object instances and poses, and change the identity of the object instances from a minimum of 5 different objects to a maximum of 80. The use of a database-wide feature matching technique (k-NN Per Database, Section 4.5.3) allows MOPED to perform almost constantly with respect to the number of objects in the database. The latency of the Baseline system, which performs independent feature matching per object, increases roughly linearly. Fig. 4.12 shows the latency of each system relative to their best scores, to see how latency increments when each of the parameters change. Therefore, it is important to note that the latency of MOPED and the Baseline are in different scales in Fig. 4.12, and one should only compare the relative differences when changing the image resolution and the size of the object database.

The number of objects in a scene is another factor that can greatly affect the performance of MOPED. The Zoom Benchmark aims to show a relatively constant number of image features (Fig. 4.13), despite being only 12 (large) objects visible at 280 mm and 160 (smaller) objects visible at 1050 mm. It is interesting to notice that the required time is inversely proportional to the number of objects in the image, i.e., a small number of large objects are more demanding than large numbers of small objects. The explanation for this fact is that the smaller objects in this benchmark are more likely to fit in a single object prior in the first iteration of ICE. Clusters that converge after the first iteration of ICE (i.e., with no correspondences transferred to or from them) require very little processing time in the second iteration of ICE. On the other hand, bigger objects require more effort in the second iteration of ICE due to the cluster merging process. It is also worth noting that this experiment pushes the limits of our graphics card, causing an inevitable degradation in performance when the GPU memory limit is reached. In the 850mm-1050mm range, the number of SIFT features to compute is slightly lower than in the 280mm-850mm range. In the latter case, the memory limit of the GPU is reached, causing a two-fold increase in feature extraction latency when this happens. Despite this effect, the average latency for MOPED in the Zoom Benchmark is 2.4 seconds, compared to 65.5 seconds in the Baseline system.

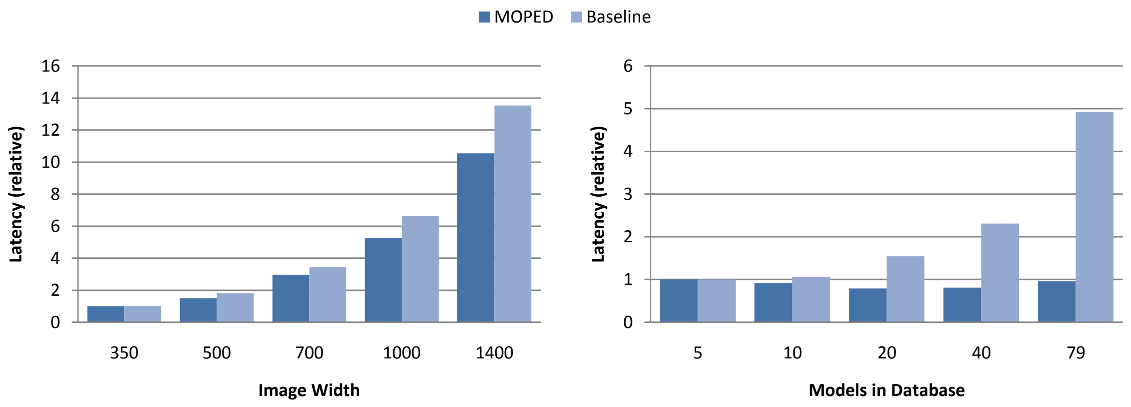


Figure 4.12: Scalability experiments in the Rotation Benchmark. (left) Latency with respect to image resolution. (right) Latency with respect to database size.

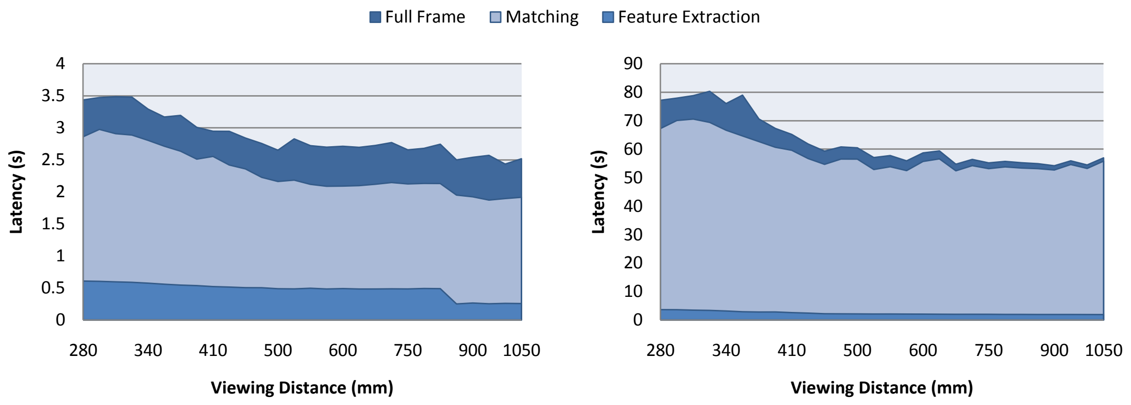


Figure 4.13: Scalability with respect to the number of objects in the scene in the Zoom Benchmark. The scale of the left chart is 22.5 times less than that of the right chart for better visibility. (left) Latency of MOPED. (right) Latency of the Baseline system.

4.6 Recognition and Accuracy

In this section, we evaluate the recognition rate, pose estimation accuracy and robustness of MOPED in the case of a single-view setup (MOPED-1V) and a three-view setup (MOPED-3V, shown in Fig. 4.15), and compare their results to other well-known multi-view object recognition strategies.

We have conducted two sets of experiments to prove MOPED's suitability for robotic manipulation. The first set evaluates the accuracy of MOPED in estimating the position and orientation of a given object in a set of images. The second set of experiments evaluates the robustness of MOPED against modeling errors, which can greatly influence the accuracy of pose estimation. In all experiments, we estimate the full 6-DOF pose of objects, and no assumptions are made on their orientation or position. In all cases, we perform the

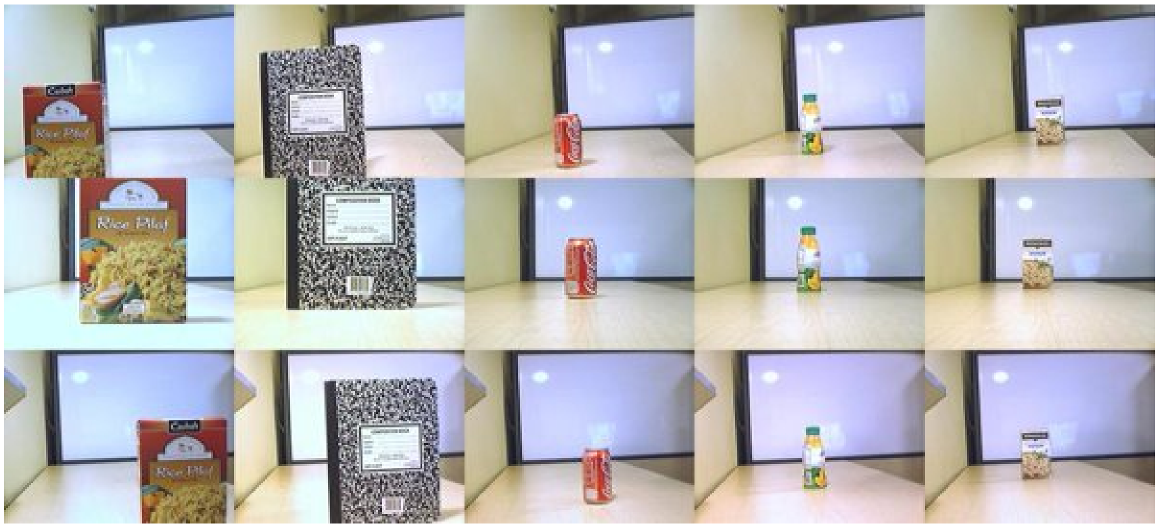


Figure 4.14: Examples scenes captured by our camera setup with Cam 1 (top), Cam 2 (middle), and Cam 3 (bottom). (Col 1) Rice box at 50 cm. (Col 2) Notebook at 60 cm. (Col 3) Coke can at 80 cm. (Col 4) Juice bottle at 1 m. (Col 5) Pasta box at 1.2 m.

image space clustering step with a Mean Shift radius of 100 pixels, and we use RANSAC with subsets of 5 correspondences to compute each hypothesis. The maximum number of RANSAC iterations is set to 500 in both Pose Estimation steps. In MOPED-3V, we enforce the requirement that a pose must be seen by at least two views, and that at least 50% of the points from the different hypotheses are consistent with the final pose. We add this requirement in order to prove that MOPED-3V takes full advantage of the multi-view geometry to improve its estimation results.

The experimental setup is a static three-camera setup with approximately 10 cm baseline between each two cameras (see Fig. 4.15). Both intrinsic and extrinsic parameters for each camera have been computed, considering camera 1 as the coordinate origin.

4.6.1 Alternatives for multi-view recognition and estimation

We consider two well-known techniques for object recognition and pose estimation in multiple simultaneous views. The techniques we consider are the Generalized Camera (Grossberg and Nayar, 2001) and the pose averaging (Viksten et al., 2006) techniques.

Generalized Camera

The Generalized Camera approach parameterizes a network of cameras as sets of rays that project from each camera center to each image pixel, thus expressing the network of cameras a single non-perspective camera with multiple projection centers and focal planes. Then, the

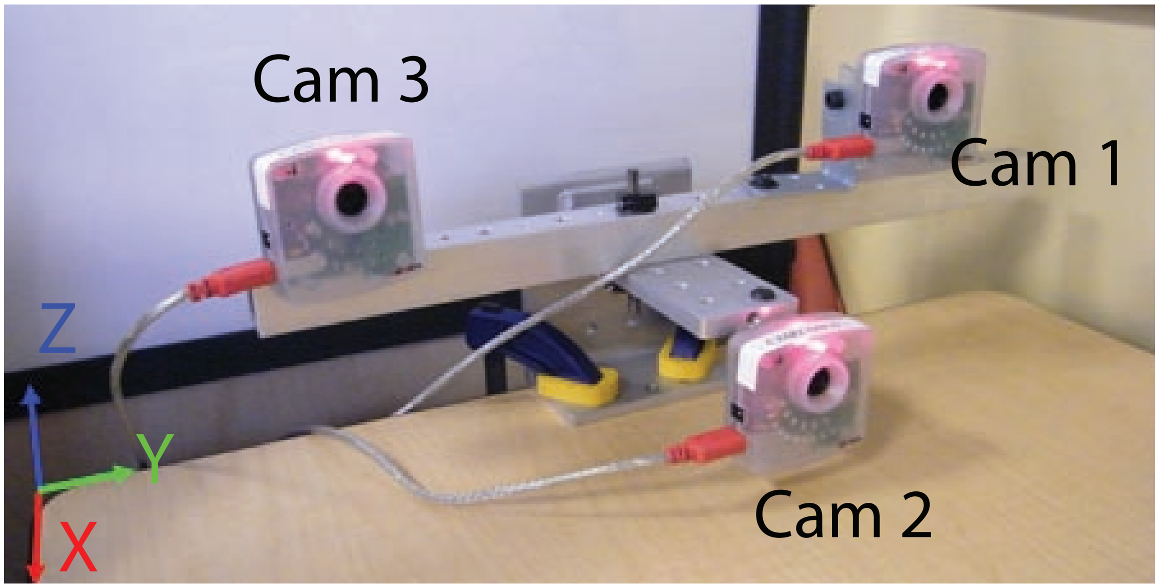


Figure 4.15: Three-camera setup used for accuracy tests with coordinate frame indicated on bottom left corner.

camera pose is optimized in this generalized space by solving the resulting non-perspective PnP (i.e., nPnP) problem (Chen and Chang, 2004). While such an approach is perfectly valid, it might not be entirely feasible in real-time if the correspondence problem needs to be addressed as well, as the search space increases dramatically with each extra image added to the system. This process takes full advantage of the multi-view geometry constraints imposed by the camera setup, and its accuracy results can be considered a theoretical limit for multi-view model-based pose estimation. In our experiments, we implement this technique and use 1000 RANSAC iterations to robustly find correspondences in the generalized space.

Pose averaging

One of the simplest and most used alternatives for multi-view recognition is to combine multiple single-image algorithms via pose verification (Selinger and Nelson, 2001), robust averaging, or weighted voting (Viksten et al., 2006). These methods avoid the larger search space that may cause difficulties in the Generalized Image approach, but they fail to extract information from the multi-view geometry to provide a globally optimized pose estimate. In our experiments, we use the output of MOPED-1V and perform pose robust averaging using the Q-Scores of the MOPED-1V hypotheses as a weighting factor.

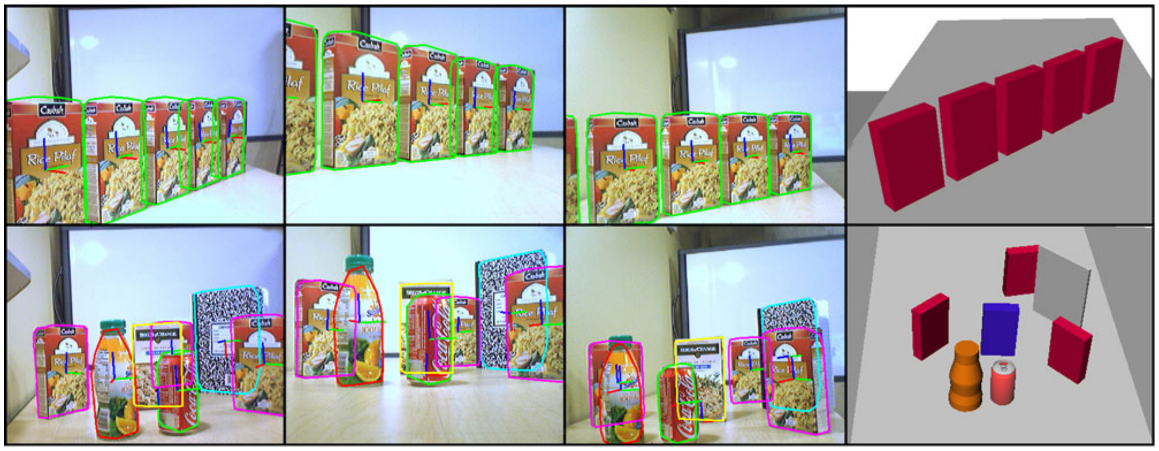


Figure 4.16: Performance of MOPED-3V in complex scenes. (Cols 1-3) depict the recognized poses overlaid on each image. (Col 4) shows a reconstruction of the given scenes in our virtual environment.

4.6.2 Pose estimation accuracy

In this set of experiments, we evaluate MOPED's accuracy over the range most useful in robotic manipulation. The three-camera setup was mounted and calibrated on a flat table (see Fig. 4.15). Our pose accuracy database is composed of five common household objects of various shapes and appearances. A set of 27 different positions and orientations for each object were gathered, with depths (i.e., distances from the central camera) ranging from 0.4 m to 1.2 m in 10 cm increments, lateral movements of up to 20 cm and out-of-plane rotations of up to 45 degrees. 10 images were taken with each camera at each position to account for possible image noise and artifacts, producing 810 images per object and a total of 4050 images. Some example images from this dataset are shown in Fig. 4.14.

It is important to mention that the choice of camera and lens can greatly affect pose estimation accuracy. The cameras we use in these experiments are low-cost cameras of 640×480 pixels with a 73° field of view. The usage of a higher resolution image and a lens with a smaller field of view would greatly improve these results.

In all these experiments, the distance-normalized translation error refers to the absolute translation error divided by the distance with respect to the closest camera. Rotation error is measured as the quaternion angle $\alpha = 2cos^{-1}(q^T q_{gt})$. The correct detection rate counts all pose hypotheses that lie within 5 cm and 10 degrees of the true pose. It is important to note that the correct detection, false positive and false negative rates do not necessarily need to add up to 100%, because an algorithm might output a correct and an incorrect pose in the same image.

Table 4.5 compares the accuracy of MOPED-1V, robust pose averaging over MOPED-

	(1)	(2)	(3)	(4)
TX error (cm)	1.45	1.36	0.47	**0.46**
TX error/dist.	1.80%	1.71%	0.61%	**0.60%**
Rot. error (deg)	6.27	8.11	5.69	**3.48**
Correct det. rate	85.0%	**88.3%**	**88.3%**	71.9%
False pos. rate	2.78%	**0%**	**0%**	**0%**
False neg. rate	13.61%	**11.67%**	**11.67%**	28.15%
Num iter./view	**96.71**	**96.71**	98.69	259.05

Table 4.5: Average accuracy test. (1) MOPED-1V (2) Pose averaging. (3) MOPED-3V (4) Generalized Image.

1V, MOPED-3V and the Generalized Image approach. MOPED-1V results show the average performance over the three cameras in our setup.

As we can see in Table 4.5, accuracy is increased threefold using MOPED-3V with respect to pose averaging, while requiring little overhead with respect to MOPED-1V. It is noteworthy that MOPED-3V and Generalized Image, considered a theoretical limit, perform very similarly in terms of accuracy, with a difference lower than 0.01%. The low detection rate of the Generalized Image approach is due to its enormous computational cost, as it often exceeds the maximum number of iterations with no correct detection. The average number of iterations required to detect a single object with a Generalized Image approach is three times greater than MOPED, and its computational complexity grows exponentially with respect to the number of objects in a scene.

4.6.3 Robustness against modeling noise

In this set of experiments, we evaluate MOPED's robustness against modeling inaccuracies. Successful pose estimation in MOPED-1V is heavily dependent on a good model calibration, especially in terms of scaling, because depth is estimated entirely based on the scale of each model. Therefore, extreme care needs to be taken when generating models to set a proper scale, and we often require several tests before a new object model can be incorporated into the robot's object database. For example, a modeling error of 1 mm in a coke can (i.e., 1 mm larger than its real size), translates into a depth estimation error of up to 3 cm at a distance of 1 m, large enough to cause problems to the robotic manipulator. On the other hand, having multiple views of the same object enables the use of further constraints in its pose. In MOPED-3V, an "implicit triangulation" takes place during the optimization, with the object drifting to its true position to minimize the global backprojection error imposed by the multi-view geometry, despite the larger error when MOPED-1V processes each view

Model scale	(1)	(2)	(3)	(4)
0.95	4.11%	4.20%	**0.81%**	**0.81%**
0.97	2.56%	2.65%	0.68%	**0.62%**
0.99	1.86%	1.76%	0.61%	**0.54%**
1.01	2.12%	1.95%	0.74%	**0.69%**
1.03	3.14%	2.90%	0.98%	**0.94%**
1.05	4.72%	4.43%	1.29%	**1.18%**

Table 4.6: Average distance-normalized translation error with varying model scale. See Table 4.5 for (1),(2),(3),(4)

Model scale	(1)	(2)	(3)	(4)
0.95	69.7%	71.7%	**80.8%**	59.3%
0.97	82.2%	85.0%	**85.8%**	66.7%
0.99	84.4%	**86.7%**	**86.7%**	71.1%
1.01	84.2%	**88.3%**	**88.3%**	70.4%
1.03	74.4%	77.5%	**87.5%**	65.2%
1.05	55.8%	58.3%	**85.0%**	54.1%

Table 4.7: Average Correct detection rate with varying model scale. See Table 4.5 for (1),(2),(3),(4)

individually.

Table 4.6 and Table 4.7 showcase the effect of scaling errors during the object modeling stage. MOPED-3V outperforms every other approach in terms of recognition rate, while achieving similar accuracy results than the Generalized Image approach. The Generalized Image approach suffers from a major performance drop when modeling errors appear, since it is often not able to find subsets of correspondences that are consistent enough to generate good hypotheses. MOPED-1V correctly finds object hypotheses in each image, but modeling noise causes a drop in the correct detection rate, as the estimated poses are often outside the 5 cm threshold. MOPED-3V, on the other hand, finds object hypotheses in each image, and then uses the inherent multi-view constraints to correctly estimate the final object poses.

4.7 Summary

We have presented and validated MOPED, an optimized framework for the recognition and registration of objects that addresses the problems of high scene complexity, scalability and latency that hamper object recognition systems when working in real-world scenes. The use of Iterative Clustering-Estimation (ICE) integrates single- and multi-view object recognition in an efficient, robust, and easy to parallelize manner. The Hypothesis Quality Score and Projection Clustering work together to minimize the number of false positives and to re-utilize all available information in the accurate pose estimation of true positives. The multiple architectural improvements in MOPED provide over 30x improvement in latency and throughput, allowing MOPED to perform in real-time robotic applications.

The different accuracy and recognition experiments we performed in this chapter gives us a quantitative evaluation of MOPED's capabilities. However, the most stringent performance test of an object recognition system for manipulation is to actually integrate it in a robotic platform and use it to interact with objects in real time. MOPED (and its extension, MOPED-RGBD) have been an active part of HERB since 2008, and the output of MOPED has been used to grasp over 2000 objects. In a controlled experiment, HERB and MOPED achieved a 91% grasping success rate using a single-image setup (MOPED-1V) and 98% success rate using a three-camera setup and MOPED-3V.

MOPED, however, is not without limitations. The recognition performance of MOPED is ultimately tied to the ability of finding enough local features in a given object. If an object is not textured enough, too far away, or has large specular reflections on its surface, the feature extraction/matching steps might not find enough correspondences in the object to perform any kind of recognition. In our experience, we have found that a minimum of 8 to 10 correspondences are necessary to successfully recognize an object and estimate its pose. Hsiao et al. (2010) showed that the ability to generate more features in a scene can result in enormous boosts in recognition rate for objects with little texture. It would be interesting to evaluate the performance of such an algorithm integrated in MOPED.

An additional issue that often arises in the model-based object recognition literature is the model building stage. The model building procedure we used in MOPED (described in Collet et al. (2009)) requires a certain amount of human supervision, and we have to manually scale our object models to achieve proper pose estimation from a single view. In Chapter 7, we describe our system HerbDisc, an integrated system that learns objects and 3D models autonomously in human environments. HerbDisc can be used as the core for Lifelong Robotic Object Perception, where the learned 3D models are used by MOPED without human supervision.

Chapter 5

An RGBD Fusion Model for MOPED-RGBD

Fade. [v. fade. ˈfād]: *To vanish, disappear; to pass completely from existence.*
Merriam-Webster.com (2012)

The use of dense depth measurements in combination with images has become popular in many areas such as object recognition (Lai et al., 2011a), scene segmentation (our own Structure Discovery in Chapter 6), SLAM (Henry et al., 2011), 3D reconstruction (Izadi et al., 2011), and Object Discovery (our own HerbDisc in Chapter 7). Dense depth measurements can be computed using a variety of sensors and techniques, such as passive stereo, active stereo (structured light), time-of-flight cameras, etc.

Despite its tremendous potential, dense depth estimation has fundamental limitations that must be addressed for robust performance. In many realistic scenes, depth sensors fail to compute depth measurements on portions of the associated color data (as shown in Fig. 5.1). We refer to this phenomenon of missing depth data as *depth fading*. Objects close to the camera, reflective or specular surfaces, poor lighting conditions, and surfaces seen at oblique angles often suffer from depth fading. These issues arise from fundamental physical limitations in depth perception, and affect all depth estimation approaches to varying degrees.

We propose to address the limitations of depth sensors at an algorithmic level instead of as a pre-processing step, and we introduce a general model to adaptively combine depth and image measurements. We derive joint image-depth measurements as the maximum likelihood estimate given the independent image and depth measurements in Section 5.1. We combine a depth-filling technique with per-pixel confidences to extend depth measurements to areas with depth fading. In this way, we adaptively combine image and depth measurements for every pixel. To demonstrate the flexibility and effectiveness of our model, we integrate it into each of the components of MOPED (Chapter 4). We use our adaptive model to derive an adaptive distance metric for feature-based pose estimation in Section 5.4; a prior generation technique based on adaptive 2D/3D similarity in Section 5.5; and a depth-adaptive feature matching scheme in Section 5.6. On our tests in Section 5.7, MOPED-RGBD yields performance gains of up to 10.4% in recall over MOPED with moderate depth fading, and a seamless transition to the image-only MOPED performance with severe depth fading. In contrast, non-adaptive data fusion models perform equally well in favorable conditions, but can detect up to 15.3% fewer objects than the image-only MOPED in the presence of depth fading.

The research described in this chapter and its original publication (Fouhey et al., 2012) are joint work with David Fouhey.

5.1 Adaptive 2D/3D Measurement Model

We propose a general model to adaptively combine image and depth measurements into a joint image-depth function. Given partial observations $x_{2D} \in \mathcal{X}_{2D}$ (image only) and $x_{3D} \in \mathcal{X}_{3D}$ (depth only), let $\phi_{2D} : \mathcal{X}_{2D} \to \mathbb{R}$ be an image-only function, and $\phi_{3D} : \mathcal{X}_{3D} \to \mathbb{R}$ a depth-only function. Defining the full observation $x = \{x_{2D}, x_{3D}\} \in \mathcal{X} = \{\mathcal{X}_{2D}, \mathcal{X}_{3D}\}$, the joint image-depth function $\phi : \mathcal{X} \to \mathbb{R}$ is a combination of the partial functions ϕ_{2D}, ϕ_{3D}.

The functions ϕ, ϕ_{2D} and ϕ_{3D} are general functions which can model a wide array of processes, as we show in Sections 5.4, 5.5, and 5.6. For the remainder of this chapter, we assume that $\phi_{2D}(x_{2D})$ and $\phi_{3D}(x_{3D})$ are measured in the same units. We also assume that $\phi_{2D}(x_{2D})$ and $\phi_{3D}(x_{3D})$ are noisy observations of the true values $\bar{\phi}_{2D}(x_{2D})$, $\bar{\phi}_{3D}(x_{3D})$, corrupted with i.i.d. noise with distributions $\mathcal{N}(0, \sigma_{2D}(x_{2D}))$ and $\mathcal{N}(0, \sigma_{3D}(x_{3D}))$ respectively.

Our goal is to find $\phi(x)$ that maximizes the probability $P(\phi(x)|\phi_{2D}(x_{2D}), \phi_{3D}(x_{3D}))$. Using Bayes' Rule, and assuming no prior knowledge of the function distributions, the optimal image-depth function $\phi^*(x)$ corresponds to the Maximum Likelihood Estimate (MLE)

$$\phi^*(x) = \arg\max_{\phi(x)} \quad P(\phi_{2D}(x_{2D}), \phi_{3D}(x_{3D})|\phi(x)). \tag{5.1}$$

Following the work of Hackett and Shah (1990), the MLE $\phi^*(x)$ is computed as

$$\phi^*(x) = \frac{\sigma^2_{2D}(x_{2D})}{\sigma^2_{2D}(x_{2D}) + \sigma^2_{3D}(x_{3D})}\phi_{3D}(x_{3D}) + \frac{\sigma^2_{3D}(x_{3D})}{\sigma^2_{2D}(x_{2D}) + \sigma^2_{3D}(x_{3D})}\phi_{2D}(x_{2D}). \tag{5.2}$$

We parameterize Eq. (5.2) in terms of a depth confidence function

$$c(x) = \frac{1}{1 + \gamma^2(x)} \tag{5.3}$$

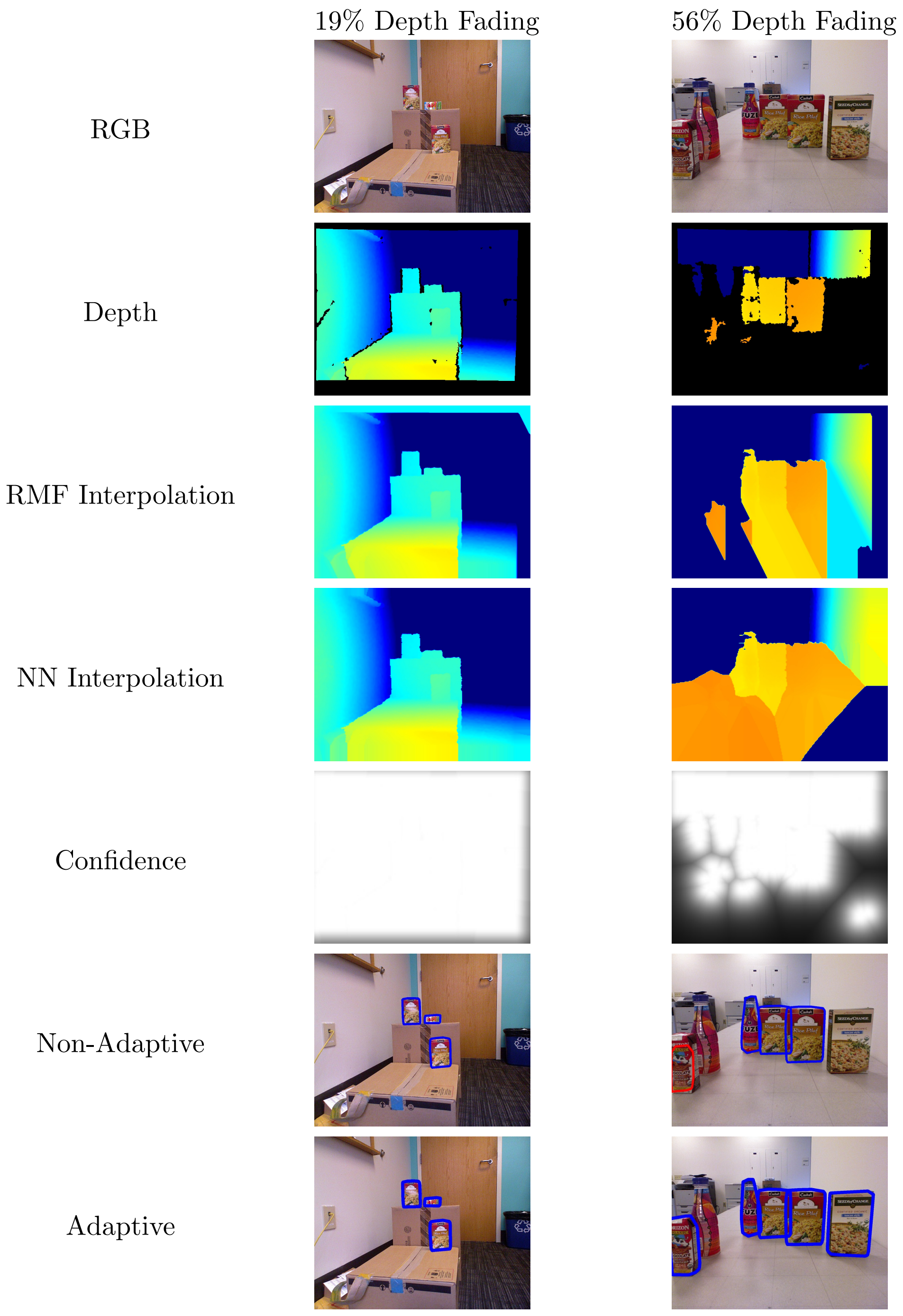


Figure 5.1: Recursive median filter and nearest neighbor succeed at removing small depth fading (left), but fail in scenes with stronger depth fading (right). Our adaptive model yields robust performance under both conditions.

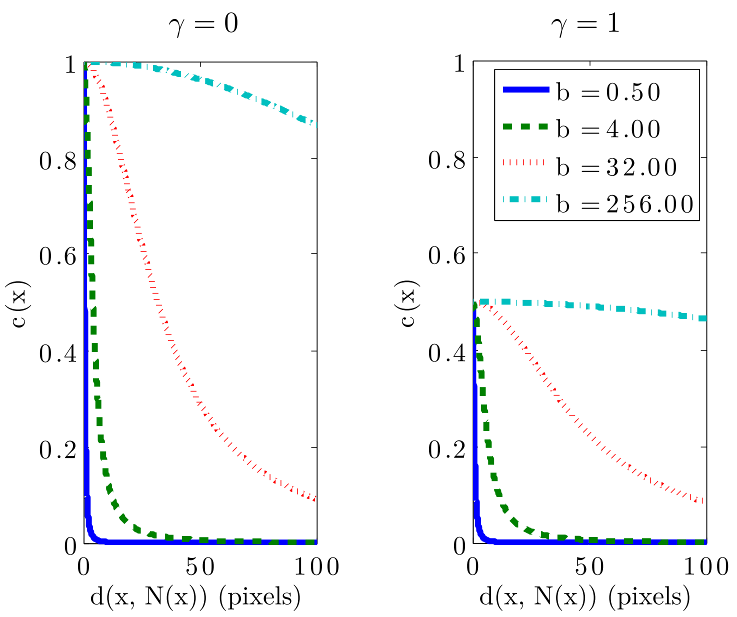


Figure 5.2: The role of confidence parameter γ. $c(x)$ vs $d(x, N(x))$ for two values of γ and varying b.

dependent on the ratio of variances

$$\gamma^2(x) = \frac{\sigma_{3D}^2(x_{3D})}{\sigma_{2D}^2(x_{2D})}, \tag{5.4}$$

such that

$$\hat{\phi}(x) = c(x)\phi_{3D}(x_{3D}) + [1 - c(x)]\phi_{2D}(x_{2D}). \tag{5.5}$$

In this formulation, $c(x)$ reflects the confidence of the depth function relative to the image function. The value of $c(x)$, and thus $\phi^*(x)$, depends only on the ratio of variances γ^2 of the noise distributions of $\phi_{2D}(x_{2D})$, $\phi_{3D}(x_{3D})$.

We extend the model in Eq. (5.5) to include depth-filling methods (e.g., nearest neighbor, recursive median filter) by estimating the ratio of variances $\gamma^2(x)$ in areas with depth fading. Let INT be an interpolation method which computes depth z_x for a point x with missing depth from a set of support datapoints $N(x)$ with depth measurements, such that $z_x = \text{INT}(N(x))$. We assume that the variance $\sigma_{3D}^2(x)$ for an interpolated datapoint x is higher than for its support datapoints $N(x)$, i.e., $\sigma_{3D}^2(x) > \sigma_{3D}^2(N(x))$. Then, given that the variance $\sigma_{2D}^2(x)$ remains constant, the ratio of variances $\gamma^2(x) > \gamma^2(N(x))$, and thus $c(x) < c(N(x))$. The resulting dense depth map contains depth values and confidences for every datapoint, with decreasing confidences $c(x)$ for areas with depth fading.

We estimate the ratio of variances γ^2 of interpolated datapoints from the ratio of vari-

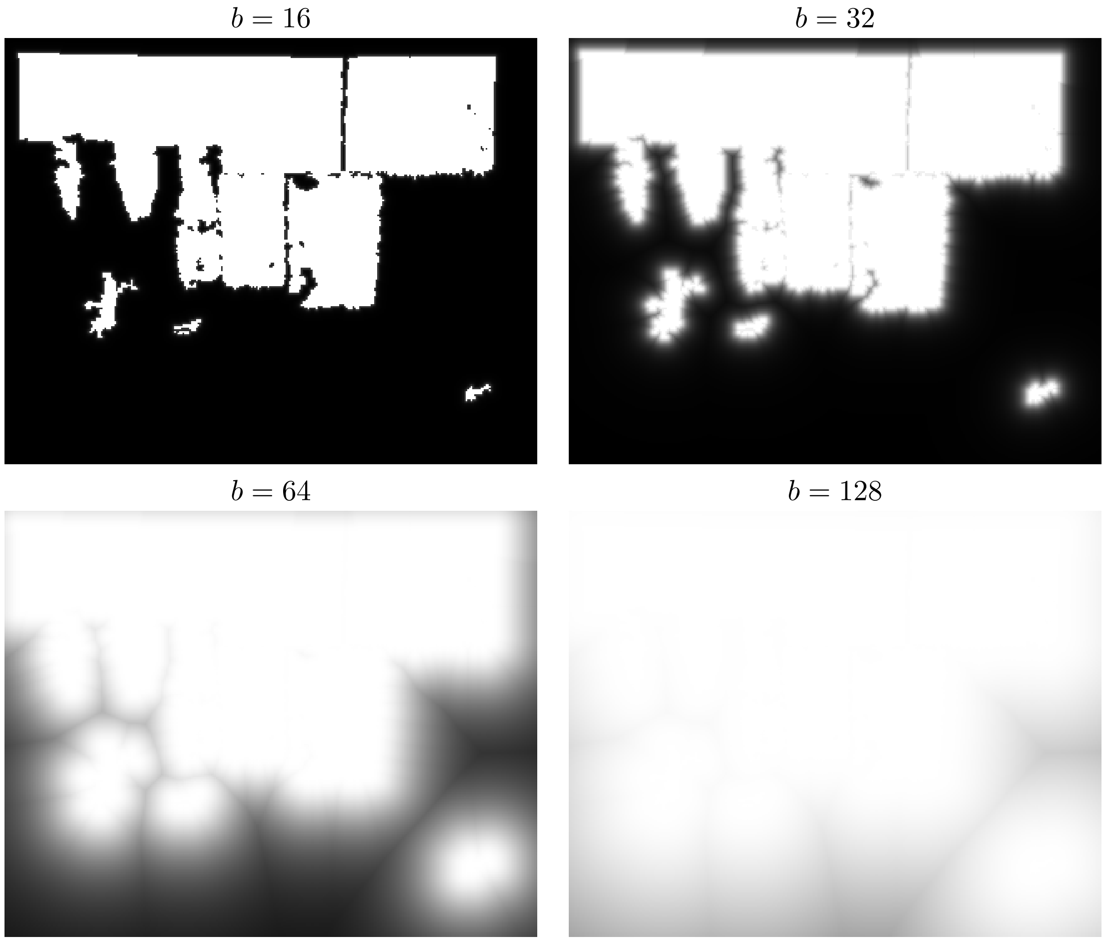


Figure 5.3: The role of confidence parapmeter b. $c(x)$ plotted for Fig. 5.1(bottom) with $\gamma = 1$ and varying b.

ances of datapoints with measured depth, $\bar{\gamma}^2$. For an interpolated datapoint x,

$$\gamma^2(x) \approx \bar{\gamma}^2(N(x)) + \frac{d(x, N(x))^2}{b^2}, \tag{5.6}$$

where $d(x, N(x))$ is a distance function between the interpolated datapoint x and its datapoints of support $N(x)$ in image space (i.e., pixel positions). The scalar parameter b encodes the trust in interpolated values with respect to measured values.

The confidence model $c(x)$ for dense depth depends exclusively on the interpolator INT, on the scalar b, and on $\gamma^2(x)$ for known datapoints. For the remainder of this chapter, we use a simple Nearest Neighbor as our interpolator $\text{INT}(x) = NN(x)$, but other alternatives (e.g. recursive median filter) are also possible. The ratio of variances γ^2 depends on the particular task and sensor. γ^2 can be estimated empirically in some cases, but it is hard in the general case. The alternative, when there is no further information, is to set a global prior on γ^2. A trivial, yet useful, prior is to assume that both noise distributions have equal variances, so $\gamma^2 = 1$ for all known datapoints. In this case, $c(x) = 0.5$ for all datapoints with known depth, weighing equally both image and depth functions. An alternative prior, useful when the depth function is more informative than the image function, is $\gamma^2 = 0$. In this case, $c(x) = 1$ for all datapoints with known depth, thus using exclusively the depth function when depth is available, and adaptively reverting to image when depth is not available.

We illustrate common values of parameters γ^2 and b in Fig. 5.2 and Fig. 5.3. In Fig. 5.2, we show $c(x)$ for a range of values of b as the interpolation distance increases. The scalar b parameterizes the confidence decrease rate as a function of interpolation distance; analytically, for a fixed γ^2, a point b pixels away from the known value has confidence $(\gamma^2+1)/(\gamma^2+2)$. Fig. 5.3 shows maps of $c(x)$ plotted for Fig. 5.1(right) for b ranging from 0.5 to 128 pixels.

Setting γ and b for a new algorithm is straightforward. Sensible values of γ are $\gamma = 1$ (for balanced image and depth measurements), $\gamma = 0$ (for less informative image measurements), or values from a depth sensor model. To find a suitable b we perform a logarithmic grid search over the validation set for a fixed γ. We establish the stability of b in extensive experiments detailed in Section 5.7.

5.2 Problem Formulation

We demonstrate the application of our model to each step of MOPED (see Section 4.1 for details). The system input is a calibrated RGBD image $I = \{\text{Rgb}, \boldsymbol{z}\}$ such that each color pixel value has a corresponding depth z_i. The output of the system is a set of object hypotheses $\mathbf{H}$ represented by an object identity and the pose of the object in the camera's reference frame. Each object model to be recognized is represented as a set of features $\mathbf{F}_o$; each feature is represented by a 3D point location $P = [X,\ Y,\ Z]^T$ in the object's reference frame and a feature descriptor D, (e.g., SIFT (Lowe, 2004)).

In the image-only system, matching is performed with the standard 2-nearest neighbors distance ratio test (Lowe, 2004), clustering with mean-shift in the image plane, and the pose estimation optimizes the reprojection error (Szeliski, 2011) with Levenberg-Marquardt minimization. We extend these techniques to adaptively use depth: we propose a depth-aware matching technique that adapts the match acceptance threshold; a prior generation approach that exploits depth for generating highly effective object proposals and fast outlier rejection; and a pose estimation technique that adaptively incorporates depth evidence to improve its objective function.

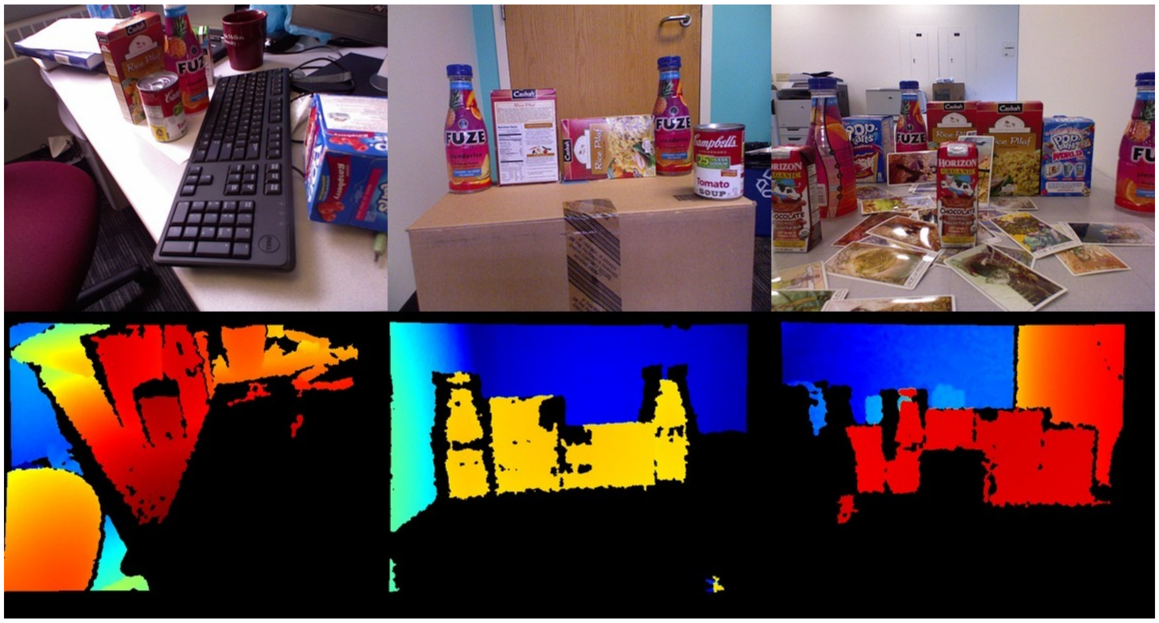


Figure 5.4: Example RGB (top) and depth images (bottom) with depth fading, from the Offices (left), validation (center) and Tables (right) datasets.

5.3 Datasets

We aim to enable perception robust to imperfect depth data. As has been articulated in Chiu et al. (2011), data in many applications (e.g., service robotics) naturally spans a wide spectrum of depth quality, including severe depth fading. We replicate this fact with two datasets with varying depth quality: the *Offices Dataset* and the *Tables Dataset*. We captured all scenes using a Microsoft Kinect RGBD sensor with registered 1280×1024 RGB data and 640×480 depth data which we upsample. Additionally, we gathered a smaller validation set that contains the same objects in 79 scenes of various household environments. Fig. 5.4 shows examples from the Offices and Tables Datasets, as well as from our validation set.

Offices Dataset: In this first set, we aim to represent optimal operating conditions for RGBD sensors; most scenes show little or no depth fading. These scenes only depict small gaps due to partial occlusion or shadowing, with an average depth coverage of 66% of the image; most fading is due to registering the depth and color images and does not fall on the objects. We captured 350 scenes with an average of 4.4 objects per scene.

Tables Dataset: In the second set, we captured 200 scenes with large sections of depth fading. These scenes show, among other anomalies, missing surfaces due to steep viewing angles and objects at short distances away from the sensor which cause heavy depth fading.

	Translation Error (%)		Rotation Error (°)	
	Offices	Tables	Offices	Tables
Reproj. Error	3.699	3.253	7.547	**6.575**
Adaptive Backproj.	**1.460**	**2.314**	**5.577**	6.919

Table 5.1: Avg. errors with adaptive backprojection and reprojection errors.

These 200 scenes contain an average of 5.2 objects and have 35% average depth coverage.

5.4 Depth-Adaptive Pose Estimation

In the first demonstration of the model, we derive an adaptive algorithm for feature-alignment-based pose estimation. In general, the estimation of an object pose given a set of 2D/3D correspondences between image features and a known model is the well-known PnP problem. The most accurate solutions are usually found by non-linear least squares minimization of pose parameters using the reprojection or backprojection errors. To adaptively use depth information, we introduce a 2D/3D distance metric, or ϕ.

Given a set of features $\boldsymbol{F}$ with 2D positions $\boldsymbol{p}_i$ and 3D positions $\boldsymbol{P}_i$, we parameterize P_i as a line L_i through p_i and the camera center, as well as a depth z_i. Given a pose hypothesis with transformation T, we define $P_{T;i}$ as the position of the corresponding feature of the hypothesis that matches P_i. Let $\hat{P}_{T;i}$ be the projection of $P_{T;i}$ onto L. Using $\hat{P}_{T;i}$ we derive two common image-only errors: the backprojection error is the 3D distance $||\hat{P}_{T;i} - P_{T;i}||^2$ and the reprojection error is the distance when projected on the image plane.

To formulate our adaptive model ϕ, we select the backprojection error as ϕ_{2D}, and introduce an orthogonal penalty in depth $||P_i - \hat{P}_{T;i}||^2$ to serve as ϕ_{3D}. We minimize the objective function

$$\arg\min_T \sum_i \phi^*(i) = \arg\min_T \sum_i c(i)\phi_{3D}(i) + [1 - c(i)]\phi_{2D}(i). \tag{5.7}$$

We set $\gamma(i) = 1$ and use our validation set to determine $b = 0.1$, which is held constant throughout the experiments.

We compare the effectiveness of our adaptive backprojection objective function with the 2D reprojection error. We present the relative translation and rotation errors in Table 5.1. Compared to the baseline, the recovered translation error is over 2.5× more accurate when using the adaptive backprojection error on images with high depth quality, and 28.8% more accurate even in scenes with low depth quality. The rotational error also improves when using data with high depth quality. The higher rotational error on low quality depth quality is due to the increased recall in detections of small, cylindrical objects. These additional detected objects have correct translation, but less accurate rotation.

5.5 Depth-Adaptive Priors for Object Recognition

Model-based object recognition and pose estimation from local features requires solving two sub-problems: data association and pose optimization. In simple scenes, RANSAC and a PnP solver are sufficient to address these two sub-problems. However, realistic scenarios may contain large numbers of outliers and multiple instances of the same object. In this case, it is vital to compute object priors to limit the otherwise overwhelming search space of potential hypotheses. A number of approaches have been used to generate priors, both from depth and image data. Collet et al. (2009) propose to estimate spatial feature density with Mean Shift to find plausible regions for objects, which is also the technique used in MOPED. Other approaches, such as the methods of Lai and Fox (2011), Kootstra and Kragic (2011), use horizontal plane information to generate priors.

In MOPED-RGBD, we use clustering for prior generation, as we do in MOPED. We partition the set of matches for a object into clusters and search only within the poses supported by these clusters. An ideal cluster contains only matches supporting one object instance and no outliers.

To provide a prior generation algorithm robust to depth fading, we propose an agglomerative clustering scheme based on 2D/3D feature similarity. Here, we extend our model to pairs of measurements and fuse a 2D-similarity function $\phi_{2D} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with a 3D-similarity function $\phi_{3D} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. To model the confidence of a pair of points, we assume their independence and set $c(\{i,j\}) = c(i)c(j)$. Our similarity function is then:

$$\phi^*(i,j) = c(i)c(j)\phi_{\text{3D}}(i,j) + [1 - c(i)c(j)]\phi_{2D}(i,j) \tag{5.8}$$

We build each term of our similarity function using simple features. W denote p_i and P_i as the 2D and 3D positions of image feature F_i, respectively; $\hat{P}_i$ is the 3D position of the corresponding feature in the model; and $d(\cdot,\cdot)$ is the Euclidean distance between two vectors.

Spatial Proximity. Objects are generally continuous; we model this assumption using the feature

$$S_{\text{2E}}(i,j) = \exp\left(\frac{-d(p_i,p_j)^2}{\sigma_{\text{2D}}}\right), \tag{5.9}$$

and equivalently S_{3E} in 3D using P_i and P_j.

Depth Discontinuity. Two matches are unlikely to belong to the same object if there is a significant depth discontinuity between them. We formalize this intuition by sampling N points along the line through p_i and p_j in the image plane, and measuring the change in

	MS 2D	Ag-2D	MS 3D	A-C	Opt. 2D
Cluster Precision (%)	27.7	60.8	17.9	**81.6**	80.0
Cluster Recall (%)	99.9	81.5	**100.0**	97.3	**100.0**
End-to-End Recall @0.9P (%)	53.3	55.0	46.0	**62.3**	56.9

Table 5.2: Comparing the effectiveness of Adaptive Clustering (A-C) to other approaches.

depth as an angle over each segment compared to the global change in depth θ of line $\overline{p_i p_j}$. This yields a penalty on strong changes in depth:

$$S_{\mathrm{D}}(i,j) = \exp\left(\frac{-\max\limits_{1\le k\le N}\{(\theta_k-\theta)\}}{\sigma_{\mathrm{disc}}}\right). \tag{5.10}$$

Distance Consistency. The availability of depth measurements enables us to check the consistency of the distance between points in the world and their locations in the model coordinate frame: we can prevent the clustering of two matches if they are at inconsistent distances and thus cannot support the same pose. The consistency similarity function is defined as:

$$S_{\mathrm{C}}(i,j) = \exp\left(\frac{-|d(P_i,P_j)-d(\hat{P}_i,\hat{P}_j)|}{d(\hat{P}_i,\hat{P}_j)\sigma_{\mathrm{cons}}}\right). \tag{5.11}$$

To combine these features into similarity functions, we choose $\phi_{2D}(i,j) = S_{2E}(i,j)$ and $\phi_{3D}(i,j) = \frac{1}{2}S_C(i,j)(S_{\mathrm{D}}(i,j) + S_{3\mathrm{E}}(i,j))$. We enforce the S_C feature more strongly as it is a hard requirement for clustering, rather than a preference (such as spatial proximity). We set $\gamma(i) = 1$ and use our validation set to determine $b = 25$, which is held constant throughout for all experiments. Our formulation has no predefined number of clusters, and similarity is defined with a general function. We use Agglomerative Clustering (Hastie et al., 2003) with group-average linkage as our clustering function.

We evaluate the output of a number of clustering approaches. We define inlier matches as matches for which there is a correctly detected pose under which the match has reprojection error 2 pixels or less. We define *cluster precision* as the fraction of matches in any clusters that are inliers and *cluster recall* is the fraction of inlier matches appearing in any cluster.

In Table 5.2, we evaluate a number of clustering approaches on the Offices 66% data set: Mean Shift in 2D (MS 2D); agglomerative clustering using 2D proximity (Ag-2D); Mean Shift in 3D (MS 3D); and our proposed approach, Adaptive Clustering, (A-C). We additionally manually label the outlines of the objects in the scene to produce the optimal clustering possible using only 2D information (Opt. 2D). Our results demonstrate the effectiveness of our proposed approach. While retaining almost all inlier matches, we achieve slightly higher precision than the 2D optimal clustering; this is possible since S_C filters out feature mismatches, even on ground-truth outline models. Additionally, we obtain a 53%

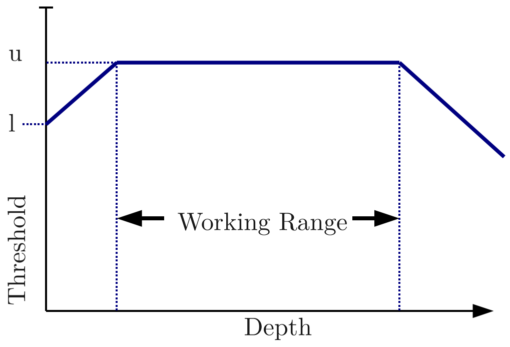
gain compared to MS-2D. Further, in comparison to Ag-2D, which uses the same features as Mean Shift and the same clustering algorithm as our approach, we achieve gains in both precision (20.8%) and recall (15.8%). These performance gains have meaningful impact on the object recognition performance of the system: at a 90% precision level, we achieve between a 5.3% (vs. Opt. 2D) and 16.2% (vs. MS 3D) object recall gain.

5.6 Adaptive Feature Matching

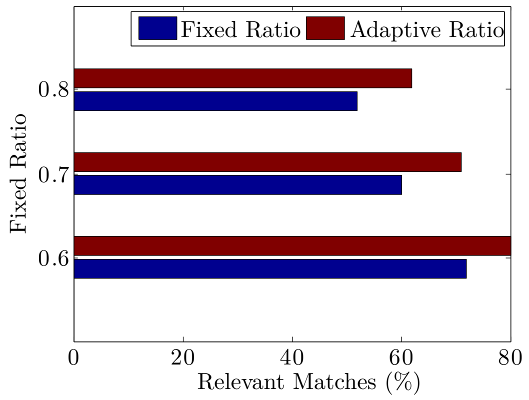
In this section, we demonstrate the application of our method to situations in which the function depends only on 3D. For instance, depth can constrain the scale at which one searches for objects (e.g., Helmer and Lowe (2010)). However, such techniques cannot be used when depth data is entirely or largely absent. In contrast, our model transitions between aggressive scale constraining during optimal conditions and more cautious behavior when depth data is unavailable. To achieve this, we let ϕ_{2D} be a constant function and set $\gamma(i) = 0$.

One way to constrain the scale of the search for objects is to adjust the number of feature matches that are accepted by adjusting the threshold used in the ratio test. The ratio test (Lowe, 2004) is a common criterion to evaluate whether a pair of local features are sufficiently similar to be considered a match. A match between a local feature F_i and its nearest neighbor $N(F_i)$ in a database is only made if the ratio between the distance $d(F_i, N(F_i))$ and the distance to the second nearest neighbor $d(F_i, N_2(F_i))$ is less than a threshold τ. The only parameter in the ratio test is the threshold τ. In the absence of *a priori* information, an educated guess is used. With RGBD sensors, however, we have depth measurements for each local feature, and we know the scale of the objects in the database. Thus, we can use the working ranges at which we expect to detect our objects based on the object's size and density of features.

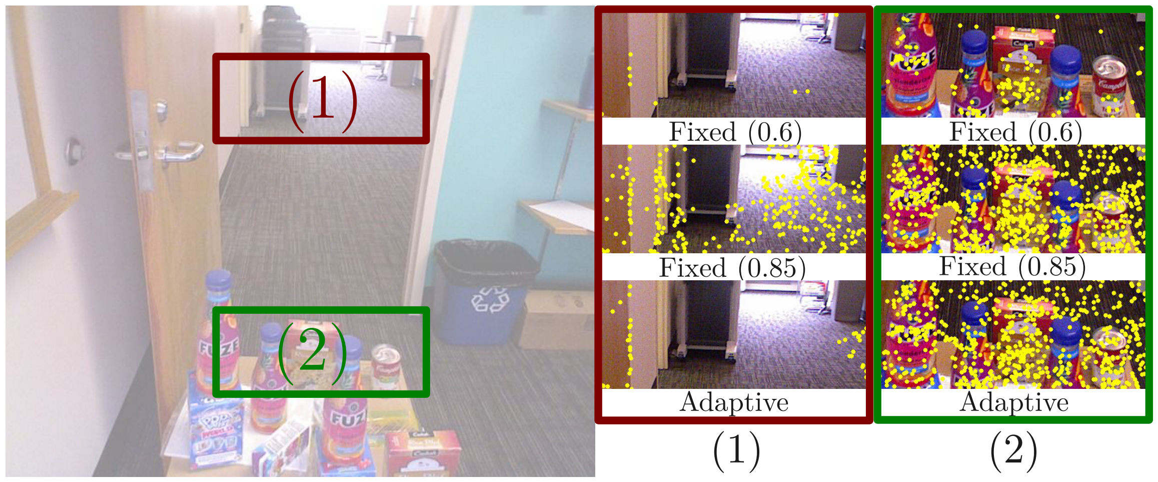
To maximize recognition and speed performance, our goal is to find just enough matches for an object to be detected throughout the object's entire working range (determined by physical size), and no matches outside this range. We approximate this behavior by replacing the fixed τ with a function ϕ^* that depends on the depth measurement z_i and the confidence score $c(i)$. When depth information is present, a function $\theta(z)$ maps depth z to a threshold. The form of $\theta(z)$ is illustrated in Fig. 5.5(a). To make this approach robust to imperfect depth data, we set $\phi_{3D}(i)$ to $\theta(z_i)$ and $\phi_{2D}(i)$ to a default ratio $\theta(z_0)$, where z_0 is a default depth. In our experiments, z_0 is fixed at $1m$. Since ϕ_{3D} completely determines ϕ with known depth, we set $\gamma(i) = 0$; we use our validation set to determine $b = 75$, which is held constant throughout the experiments. Parameters l, u are fixed using grid search to maximize recall on a validation set; they are not sensitive to particular values, and are in the vicinity of usually used values.



((a)) The threshold function $\theta(z)$ for a given object as a function of depth. The feature density and size of each object determine each threshold function.



((b)) For a given total number of matches inside objects (relevant matches), Adaptive matching achieves significantly higher percentage of relevant matches (i.e., lower error rate) than fixed thresholds.



((c)) Qualitative illustration of Adaptive Matching: more matches in relevant regions and limited matches in the background.

Figure 5.5: Illustration of Adaptive Matching. (i) Using Adaptive Matching instead of a fixed threshold, we (ii) achieve a higher percentage of matches inside objects. (iii) Qualitative example of Adaptive Matching.

We evaluate the matching performance by counting the fraction of relevant matches; we define a feature as *relevant* if it falls on any ground-truth object. Higher thresholds yield more relevant features at the cost of more irrelevant ones. Since our approach reverts to a per-object fixed threshold in the absence of depth data, we only consider matches with sufficiently high confidence (above 0.5) to focus the evaluation on the depth-dependent scheme. Fig. 5.5(b) demonstrates that we can replace a fixed ratio with an adaptive ratio that yields the same number of relevant matches, but fewer irrelevant matches. We show a qualitative example of this behavior in Fig. 5.5(c).

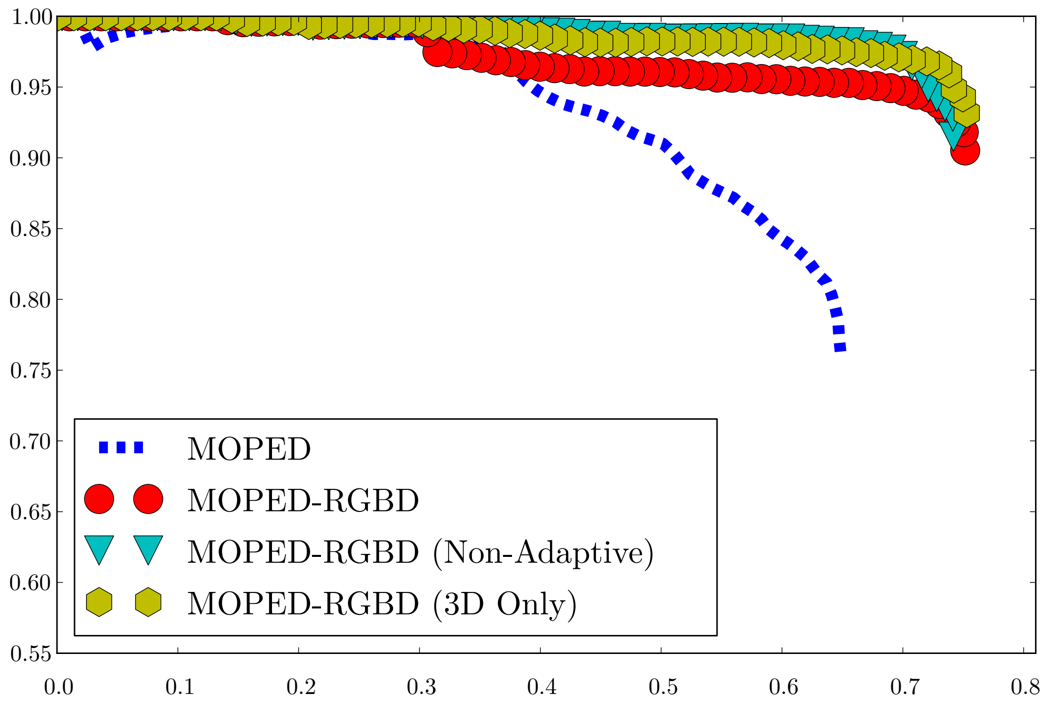
5.7 Integrated Testing

In this section, we evaluate the impact of our adaptive model for the integrated system **MOPED-RGBD**, which uses all of the proposed algorithms. To evaluate the adaptive model, we compare performance with two non-adaptive baselines as well as the image-only **MOPED**: the first, **3D Only**, uses only data points for which there is depth data; the second, **Non-Adaptive**, trusts all depth values (interpolated and measured) equally. In addition, we evaluate the effect of introducing individual components to MOPED.

We process each image 5 times to account for the non-determinisc nature of RANSAC. We consider an object hypothesis as correct if its translational distance to the ground truth is less than 10 cm and its rotational distance is less than 20°. *Precision* is the fraction of correct object hypotheses among the detected objects, and *recall* is the fraction of ground-truth objects with a correctly detected hypothesis.

The precision-recall curves in Fig. 5.6 demonstrate the importance of an adaptive approach to address depth fading in object recognition. With small depth fading (Fig. 5.6(top)), the adaptive and non-adaptive models perform similarly well, and all approaches outperform the baseline (MOPED). However, with moderate to severe depth fading, *both non-adaptive models perform significantly worse than using image data alone* (Fig. 5.6(bottom)). In contrast, the adaptive model leverages depth when available to boost performance, while seamlessly transitioning to image-only performance when depth fading is present. Further analysis shows that non-adaptive approaches yield 10.3% and 15.3% decreases in recall under moderate depth fading compared to using image data alone, as shown in Fig. 5.7. An illustration of a non-adaptive failure mode is presented in Fig. 5.9.

To distinguish the changes in performance due to depth quality from those due to scene composition, we perform additional experiments with synthetically degraded data. We remove data within circles with varying radii (5-30 pixels) to produce scenes with 35% depth coverage (the average depth coverage of the Tables data set) and 15% depth coverage (the lowest naturally-occurring depth coverage in our dataset) for each RGBD image in each dataset. In Fig. 5.7, we present the area under the precision-recall curve (AUPRC) for



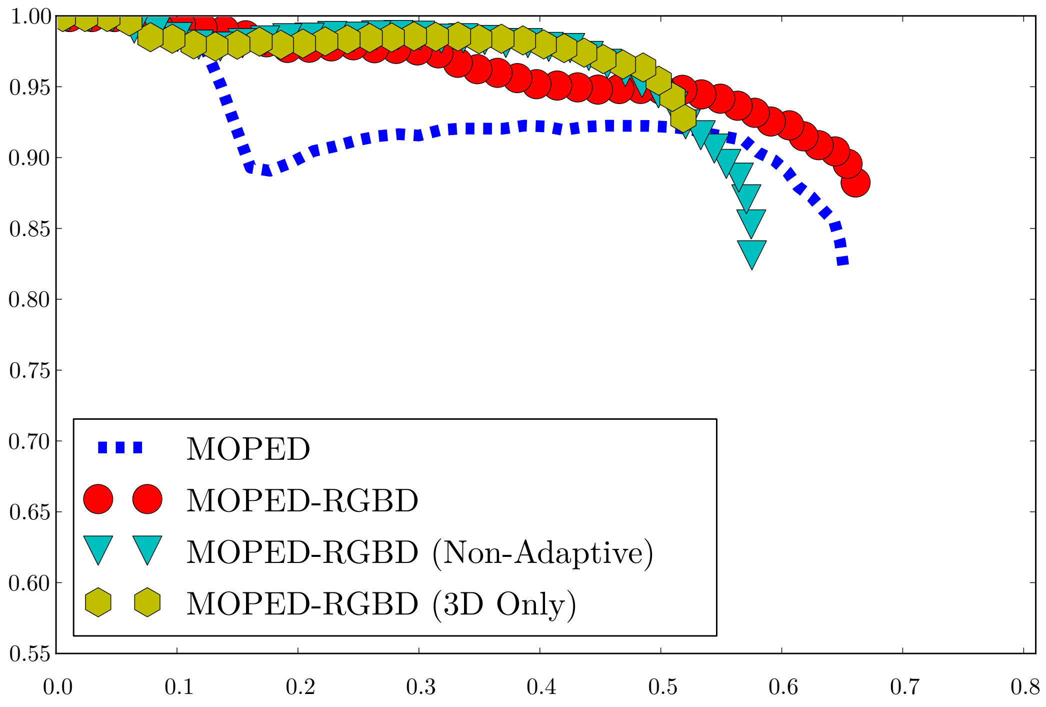


Figure 5.6: Precision-recall curves comparing adaptive vs. non-adaptive approaches on the Offices (top) and Tables (bottom) dataset.

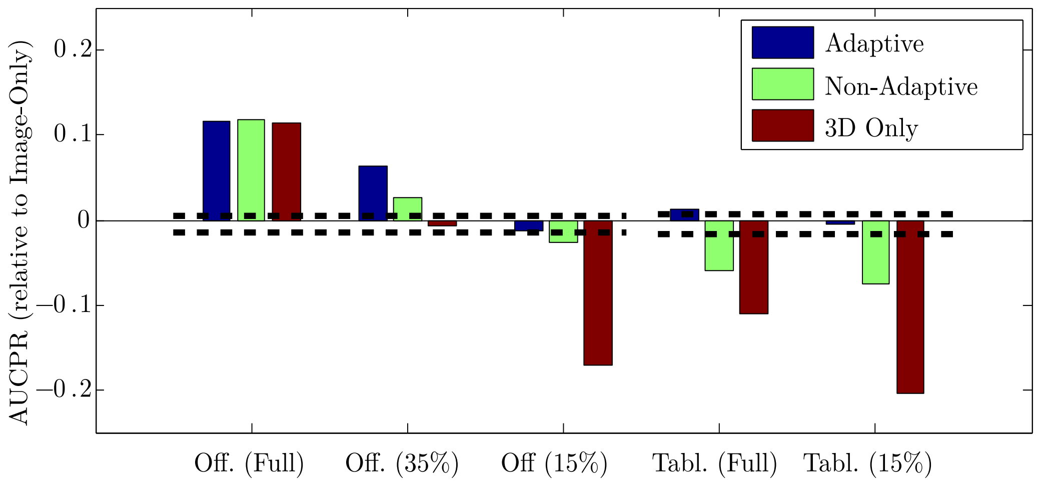


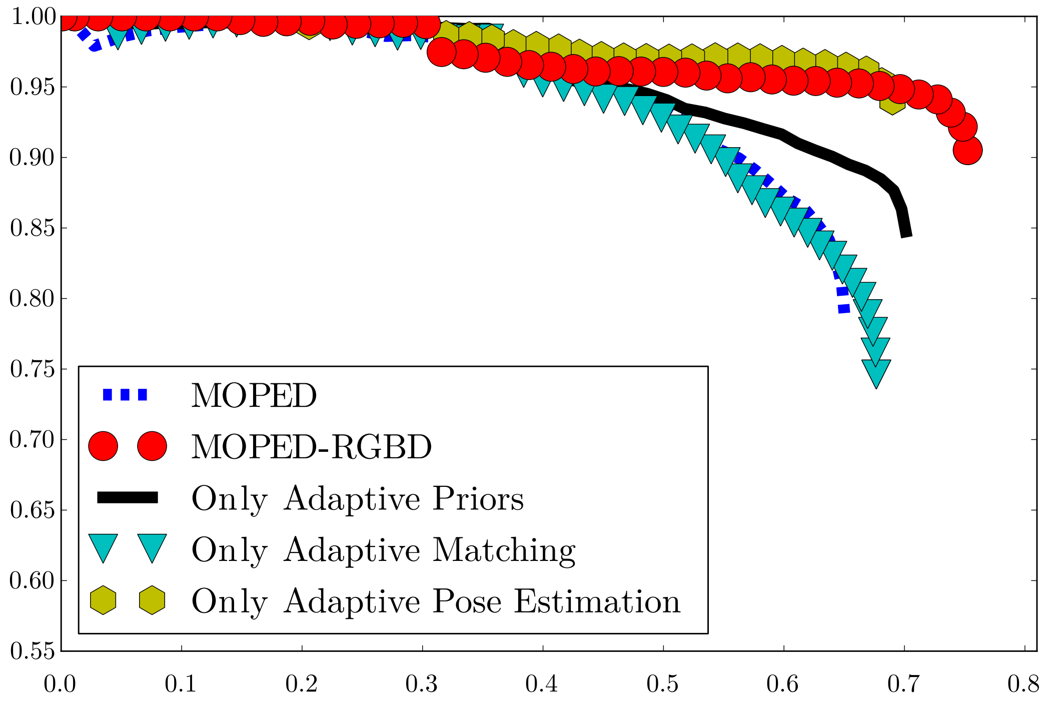
Figure 5.7: Average area under the precision-recall curve (AUPRC) for adaptive vs non-adaptive approaches, relative to the image-only. The Adaptive approach boosts performance when range data is present, and seamlessly transitions to image-only performance (black dashed lines) as range data degrades. The black dashed lines show the performance range of the image-only approach over 10 runs.

each approach. Again, only the adaptive approach performs similarly or better than the image-only approach across all data sets.

We evaluate the impact of each individual component (Adaptive Priors, Adaptive Matching, Adaptive Pose Estimation) in MOPED-RGBD in Fig. 5.8(top). The best performing individual algorithm is the Adaptive Pose Estimation, which achieves similar precision as MOPED-RGBD throughout the P-R curve. MOPED-RGBD leverages the increased percentage of relevant matches from Adaptive Matching, and the more accurate object priors from Adaptive Priors, to boost the maximum recall 9% higher than any of the independent algorithms at similar precision.

The timing comparison in Fig. 5.8(bottom) shows that we do not incur in any extra latency in MOPED-RGBD with respect to MOPED. The increased time spent computing more accurate priors (Step 3 in Fig. 5.8(bottom)) is compensated by the time saved in estimating poses (Step 5 in Fig. 5.8(bottom)) for only the most likely priors.

We show the stability of the parameter b of our model for each individual algorithm in Fig. 5.10. In particular, we show the AUPRC for increasing values of b with fixed $\gamma(i)$. We see that b is not very sensitive to specific values, and it is only the order of magnitude that is relevant. For tasks which require high precision in depth measurements (such as pose estimation) the best-performing values of b are small, $b < 1$. In contrast, tasks requiring only rough depth measurements (such as clustering) perform better with larger values of b.



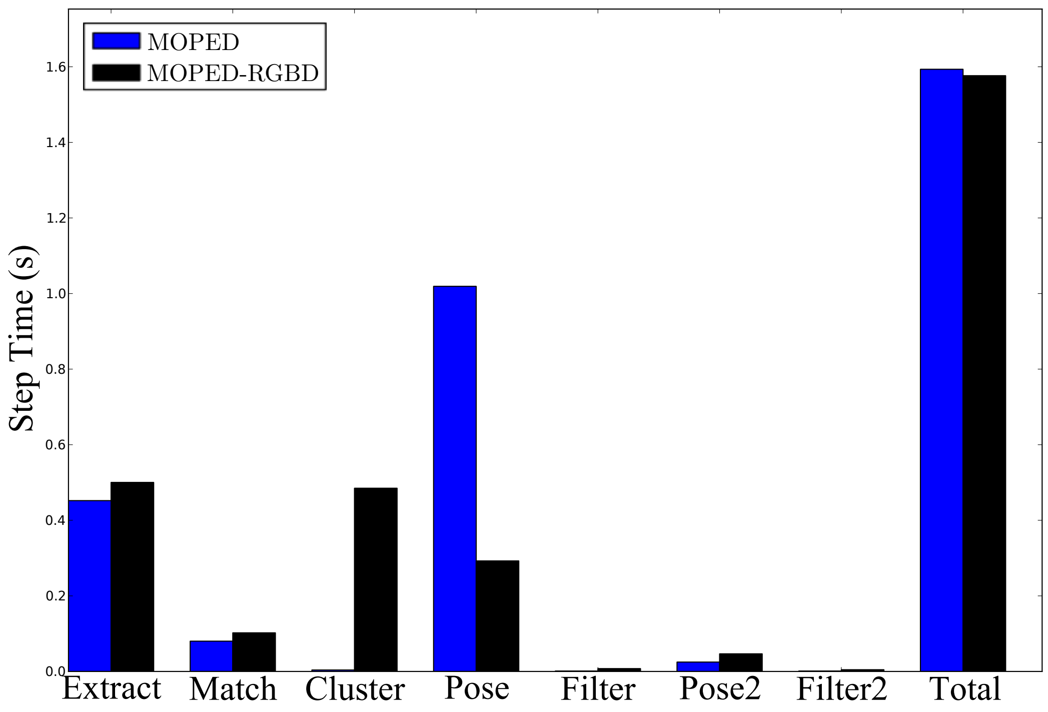


Figure 5.8: (top) Precision-recall curves evaluating the individual contributions of Adaptive Priors, Adaptive Matching, and Adaptive Pose Estimation, on the Offices Dataset.

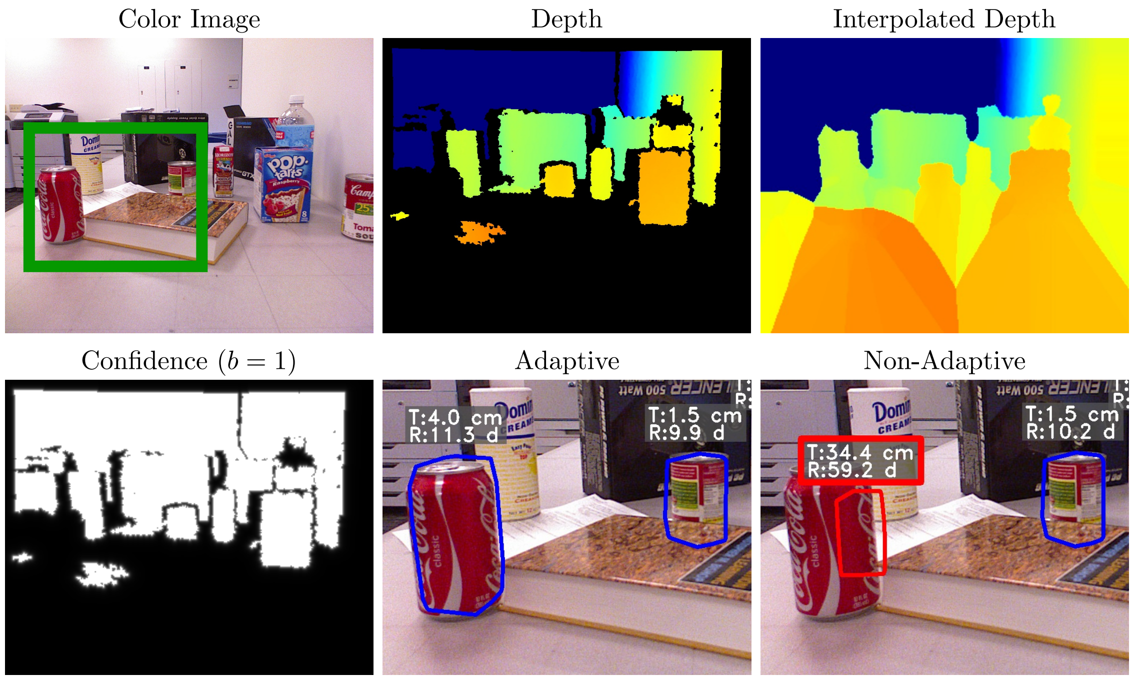


Figure 5.9: The adaptive model is able to correctly estimate object poses, even with inaccurate interpolated depth; in contrast, relying on depth filling methods results in both poorly localized objects and missed detections.

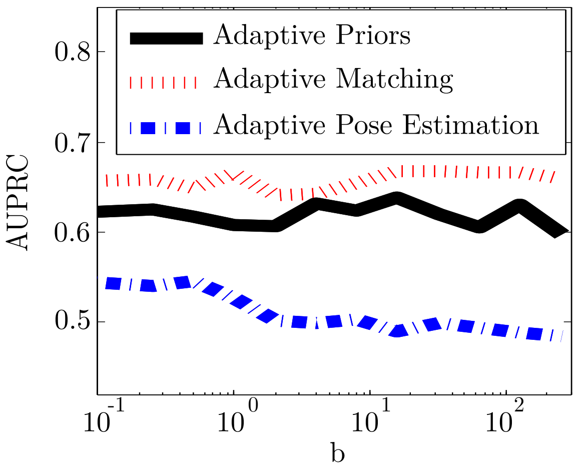


Figure 5.10: Average AUPRC for the individual algorithms, for fixed γ^2 and varying b (log scale) on the validation set.

5.8 Summary

We have introduced an adaptive model to robustly integrate depth data into image-only preception and we have applied it to each stage of MOPED. The resulting system, MOPED-RGBD, yields significant performance gains over MOPED with moderate depth fading, and a seamless transition to the performance of the image-only MOPED in the presence of severe depth fading. MOPED-RGBD is the object recognition system used in HERB since September 2011.

The limitations of MOPED-RGBD are the essentially the same as those from MOPED. Namely, the problem of finding enough correspondences in a given object. If the object contains enough texture, then MOPED-RGBD will most likely identify the object and estimate its pose. If the object is untextured, it will not be recognized with any local feature-based approach, including MOPED. For such cases, there exist promising approaches such as Hinterstoisser et al. (2011), but the pose estimation capabilities for such algorithms are not yet nearly as accurate and robust as those from MOPED or MOPED-RGBD. A line of research for our future work is to add 3D-based features to MOPED-RGBD for recognition of textured and untextured objects based on combined appearance and geometry.

Part III

Robotic Object Discovery

Overview of Part III

A critical part of our Lifelong Robotic Object Perception framework is Robotic Object Discovery. In the following chapters of this thesis, we study the problem of discovering novel objects in the environment while the robot operates, for as long as the robot operates. We term this problem as the *Lifelong Robotic Object Discovery* (LROD) problem. As a first step towards LROD, we aim to automatically discover objects during one workday of a service robot, which amounts to 6-8 hours of raw RGBD video.

Our work in Robotic Object Discovery is separated in two chapters: first, we describe how to compute generic object candidates from single RGBD images in Chapter 6, using our Structure Discovery algorithm. Then, we describe how to combine the generic object candidates with graph-based clustering methods to discover objects in Chapter 7.

We argue that leveraging the non-visual information (*metadata*) from natural robotics constraints is crucial for speed and robustness. We introduce the concept of *constraint* as an intermediate representation to encode information, both metadata and visual similarity. We use these constraints to compute Constrained Similarity Graphs (CSGs). To evaluate our Robotic Object Discovery implementation, HerbDisc, we gathered 6 h 20 min of raw RGBD video stream from HERB exploring an university building. Our Object Discovery system, HerbDisc, discovers 206 novel objects in this dataset in 18 min 34 s.

Chapter 6

RGBD Scene Segmentation: Structure Discovery

Structure. [n. struc·ture. strək-chər]: *Something arranged in a definite pattern of organization.*
Merriam-Webster.com (2012)

In this chapter, our goal is to compute object candidates for Robotic Object Discovery (Section 7). In particular, we aim to generate a scene segmentation, together with a ranking mechanism, such that the highest-ranking segments correspond to objects in the scene (e.g., Fig. 6.1). We call this process *Structure Discovery.* We combine range and image data to compute perceptual cues such as concavities and discontinuities. These cues are then used to generate scene segmentations that preserve objects.

Our main contribution in this work is a structure discovery algorithm that exploits the availability of RGBD data, as illustrated in Fig. 6.1 and Fig. 6.2. We generate multiple segmentations, as in Hoiem et al. (2005), Sivic et al. (2005), of image and range data by varying the parameters of a standard segmentation algorithm (in our case, the FH graph-based segmentation of Felzenszwalb and Huttenlocher (2004)). While no single segmentation is completely correct, we assume that some segments in some of the segmentations are correct and contain a whole object. We relate segments in an image to the corresponding range measurements, and vice versa, to create RGBD data regions. We term these RGBD regions *regionlets.* We then compute region-wide features for each regionlet, and aggregate them in a single objective function that measures the structure of each region. We show how simple features such as color consistency, continuity, alignment, and concavity work very well to identify potential structures.

We apply our Structure Discovery algorithm to discover objects in indoor environments. The current state of the art in this area is to use range data, tuned to exploit domain knowl-

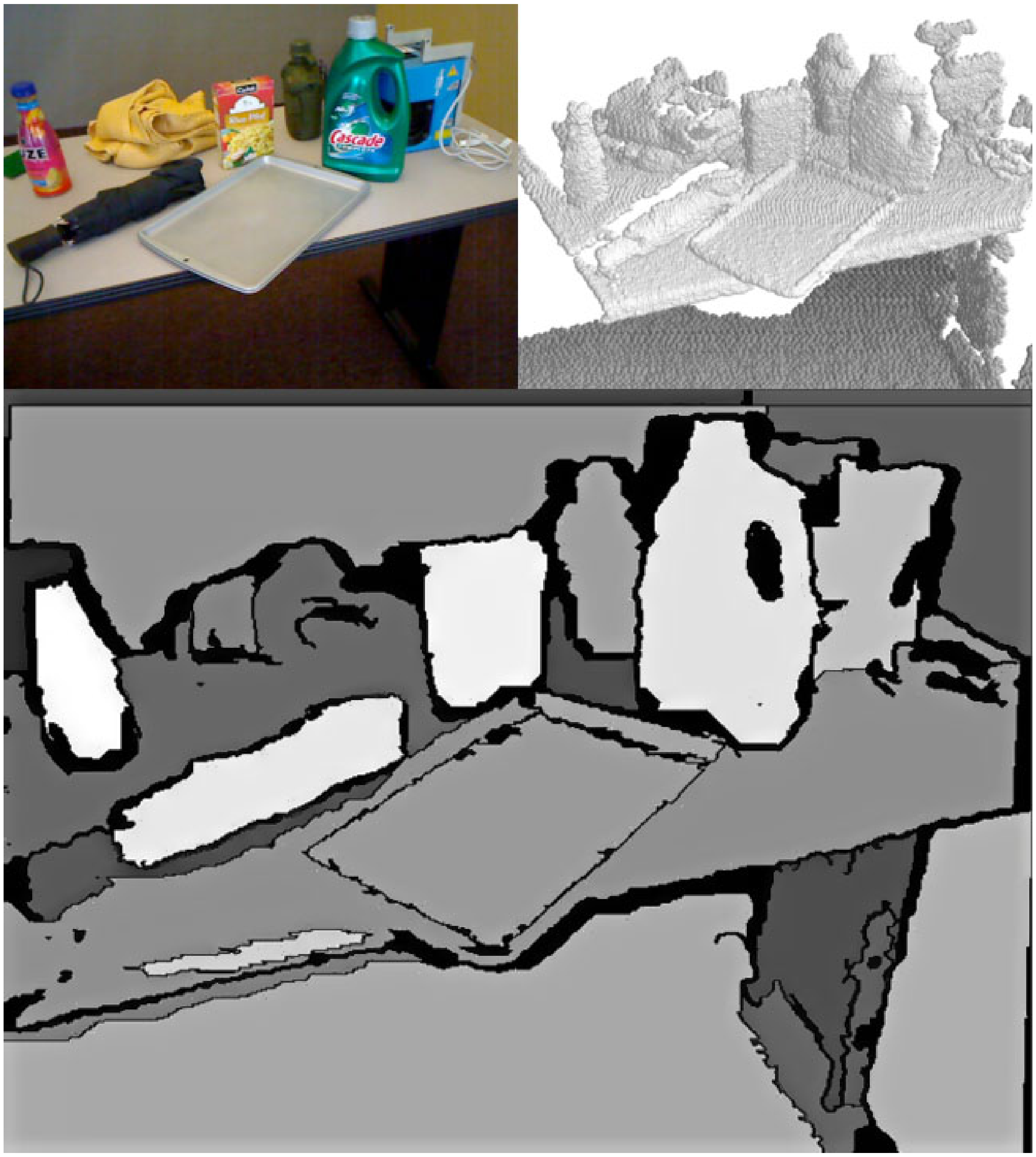


Figure 6.1: Results of our algorithm for structure discovery applied to household items. (Top-left) Input image. (Top-right) Input depth image. (Bottom) Objectness RGBD Segmentation, color-coded to show the highest-ranked segments in white and the lowest-ranked segments in black.

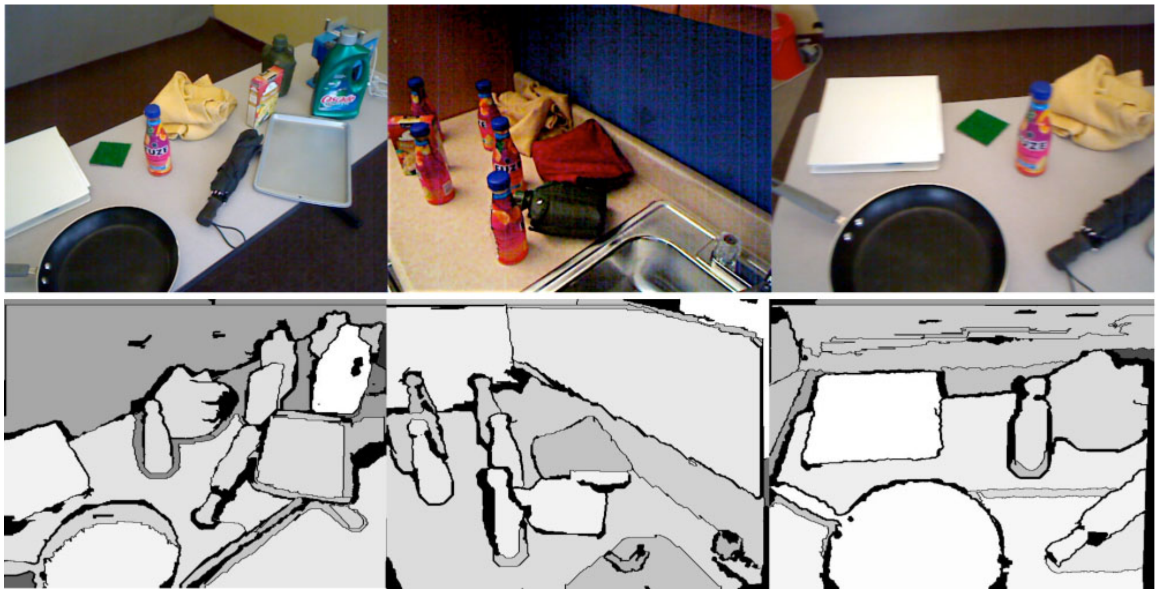


Figure 6.2: Example scene segmentations with Structure Discovery, color-coded as in Fig. 6.1. Note that the whiter sections of the images correspond to objects.

edge of the geometry of the scene, such as the plane-based 3D segmentation of Rusu and Cousins (2011). The use of multiple assumptions simplifies the solution, but also limits its application to finding objects on top of a nearby table. Image-based object discovery techniques are more focused on larger objects and natural scenes (Endres and Hoiem, 2010, Alexe et al., 2010), and are outperformed by the range-based approach in indoor environments. In the experiments section, we show that Structure Discovery performs as well as the state of the art in finding objects in tabletop scenes, without any limiting assumptions about the scene. Algorithms optimized for particular scene geometries work poorly when their assumptions about the scene are violated (e.g., Fig. 6.1, in which the table is not visible enough). The greater generality of our algorithm is showcased in a second set of experiments, in which we discover objects in generic indoor scenes that the current state of the art is unable to process. We provide further experiments in Chapter 7 showing that using Structure Discovery for candidate generation in Robotic Object Discovery, we achieve 13% higher recall at the same precision than using the tabletop segmentation of Rusu and Cousins (2011).

6.1 Method overview

Given a single image and a single range scan of a scene, our goal is to automatically discover objects, mainly in terms of appearance, shape smoothness and continuity. Our algorithm is

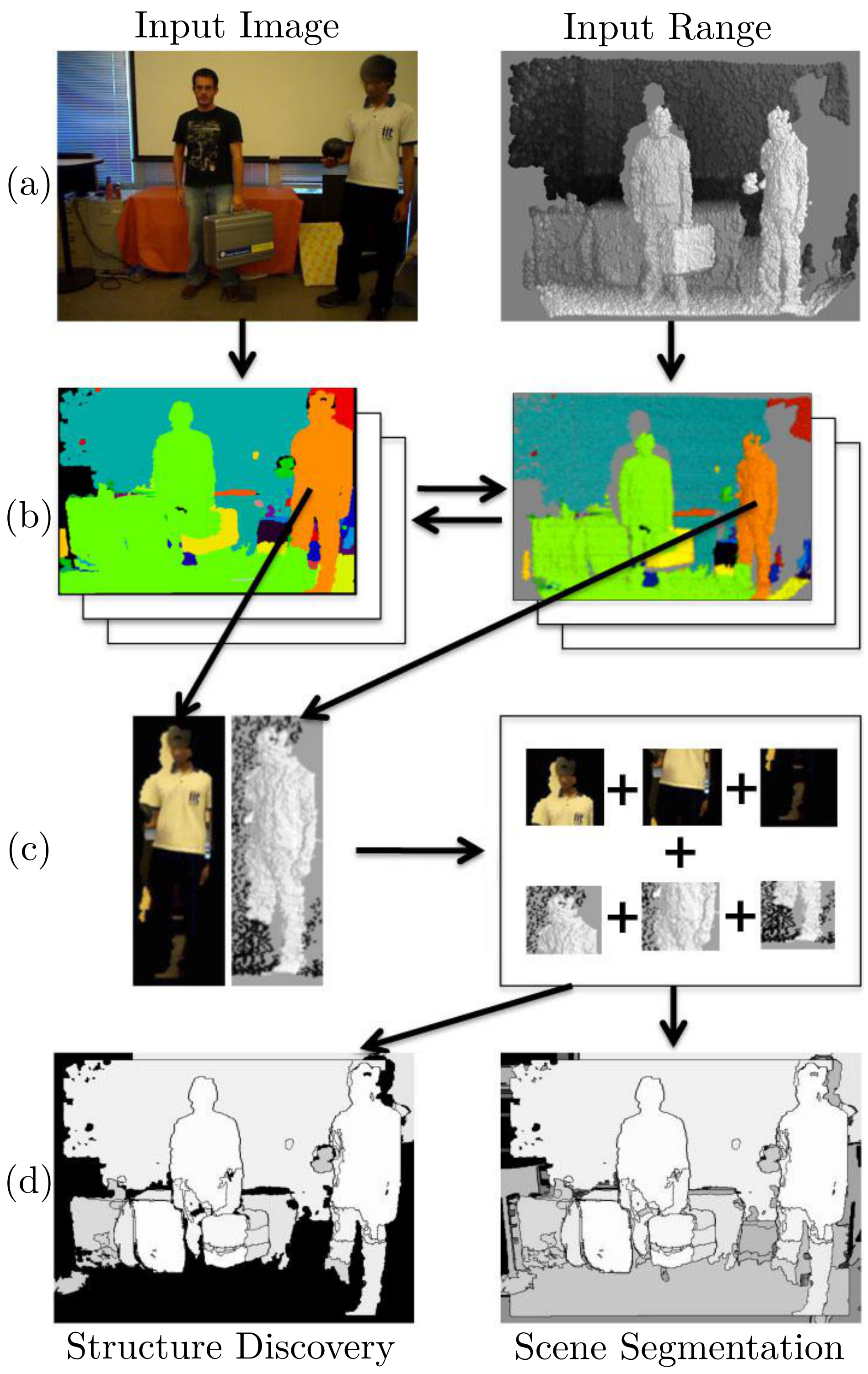


Figure 6.3: Method overview. (a) Input data, image + range data. (b) Initial segmentations on each data source. Segmentations are projected to second data source. (c) Regionlets and sub-regionlets are generated from image, range data segmentations. Structure-ness score is computed from features on regionlets. (d) Output: structure discovery and scene segmentation.

summarized in Fig. 6.3. The result is a scene segmentation of the different structures that compose a scene, ranked from most to least structured.

We organize our method in four different steps, which we describe in detail in the following sections. Fig. 6.3(a) shows an example of the input we receive from our sensors for a particular scene: an image, and a point cloud or depth image. Fig. 6.3(b) shows example segmentations from each data source. Each of the segments in each of the segmentations is considered a candidate for a potential structure; our work, therefore, is to analyze each of the segments and rank them in terms of their structure-ness, according to the features defined below.

Once the initial segmentations in each data source are generated, we need to compute a RGBD region, a regionlet, from each segment in each segmentation. Segments from each data source are associated with data points from the other data source through a projection scheme. Each resulting regionlet contains a set of pixels and a set of 3D points, but no individual pixel to 3D point association is made. In order to rank the different candidates, we develop a hierarchical scheme: each regionlet is split into sub-regionlets to evaluate both local (at the sub-regionlet level) and global (at the regionlet level) consistency, in a multi-resolution grid similar to segmentation trees of Borenstein et al. (2004), Sharon et al. (2001). Example sub-regionlets are shown in Fig. 6.3(c). We then calculate each regionlet score by evaluating image features, range features and mixed features at each level, and produce a ranking of regionlets based on their score . Finally, we assign each pixel and 3D point to the highest-ranked regionlet that contains it, thus producing a segmentation of the scene based on structure-ness, as shown in Fig. 6.3(d). If we wish to retrieve only the most structured regionlets, we only need to choose the few highest ranked regionlets.

6.2 Candidate generation

Generating likely candidates to evaluate their structure-ness is a hard problem. Ideally, we would like to try all possible segmentations from a scene and rank them, keeping only those with highest score. Unfortunately, such a procedure is infeasible. Following Hoiem et al. (2005), Sivic et al. (2005), we generate a small number of segmentations with the well-known segmentation algorithm of Felzenszwalb and Huttenlocher (2004) as a representative sample of the set of all possible segmentations. Our assumption is that none of the segmentations are correct as a whole, but that *some* segments in *some* segmentations will be correct and contain objects. It is important to note that, since we do not rely on the full segmentation to be correct, the particular choice of a segmentation algorithm is not that critical. We choose the method of Felzenszwalb and Huttenlocher (2004) because it is fast and produces reasonable results. We perform this procedure independently for the image and range data. For the image segmentations, we use the difference between pixel colors in RGB space as a similarity measure, and generate multiple segmentations by progressively increasing the threshold k, typically from $k = 50$ to $k = 500$ in increments of 50. For the range data segmentations, we define similarity as the Euclidean distance between the eigenvalue-based features in Lalonde et al. (2006), computed in a neighborhood of up to 20 nearest neighbors. To generate multiple segmentations, we progressively increase the threshold k from $k = 1$ to $k = 10$ in increments of 1.

6.3 Generating Regionlets

One of the main contributions of our work is the mid-level fusion of image and range data. We use regionlets as our elementary processing units, which we create from image and range segments. Each regionlet

$$R_i = \{\mathbf{I}_i, \mathbf{p}_i, \mathbf{P}_i, \mathcal{A}(R_i)\} \tag{6.1}$$

is defined by a set of N image pixels $\mathbf{I}_i$ and their corresponding pixel positions $\mathbf{p}_i$, a set of M 3D points $\mathbf{P}_i$, and a set of adjacencies $\mathcal{A}(R_i)$. Hierarchies of regionlets are denoted by $R_i^{(k)}$ for regionlet R_i in the k-th level of the hierarchy. In this case, the adjacencies of a regionlet $R^{(k)}$ can be parents, neighbors and children of $R^{(k)}$ depending on whether they are from level $k-1$, k or $k+1$ in the hierarchy, respectively. In this work, we use a two-layer hierarchical scheme. For simplicity, we refer to the top-level regionlets $R^{(0)}$ as simply *regionlets* R, and the bottom-level regionlets $R^{(1)}$ as *sub-regionlets* r.

6.3.1 Associating 3D to image segments

Given an image segmentation, we must compute the 3D range measurements associated with each segment in the image. Assuming that camera and 3D sensor are calibrated, the 3D data are projected into the image and associated with the image segments into which they are projected.

6.3.2 Associating image pixels to 3D segments

To generate a regionlet given 3D segments, we must associate a set of pixels in the image with each 3D segment. We compute which image pixels correspond to a range data segment via Z-buffering (Wand et al., 2001). We sort all range measurements according to their depth with respect to the camera. Starting with the points that are furthest away, each 3D point P_j paints a circle of pixels around its image projection $\hat{p}_j$ with the regionlet ID that corresponds to P_j. 3D points closer to the camera paint over 3D points further away. As a result, we obtain a set of image pixels associated with a range data segment. In order

	Unary	Pairwise	Regionlet	Sub-regionlet
Appearance Model	✓	✗	✓	✗
Shape Model	✓	✗	✓	✗
Self-Continuity	✓	✗	✓	✓
Contour Compactness	✓	✗	✓	✓
Pair-Continuity	✗	✓	✗	✓
Verticality	✓	✗	✓	✓
Concavity	✗	✓	✗	✓
Projection	✗	✓	✗	✓
Alignment	✗	✓	✗	✓
Color histogram	✗	✓	✗	✓
Surface compatibility	✗	✓	✗	✓

Table 6.1: Unary and pairwise terms, and application to regionlets and sub-regionlets.

to account for differences in range data density, we adapt the radii of the circles painted according to the density of each region.

6.3.3 Generating sub-regionlets

In order to calculate a score for each regionlet and keep computations tractable, we separate each top-level regionlet into sub-regionlets, akin to the use of super-pixels from an image over-segmentation as elementary units of processing. In our work, we consider three choices to generate sub-regionlets: super-pixels (over-segmentation in image space), super-points (over-segmentation in 3D space), and fixed shape and size sub-regionlets. One important drawback of both super-pixels and super-points is that their shapes are often elongated and contain many twists and turns. This effect often results in very narrow sub-regionlets, with a width of only a few data points. Computing any kind of 3D features (e.g., surface normals) is unreliable in such data, so we choose instead the safer approach of sub-regionlets of non-overlapping shapes that maximize the visible surface area, in order to compute both 2D and 3D features reliably. In particular, we use squares of side 1/12 of the vertical resolution of the image (e.g., 40×40 pixels in a 640×480 pixels image).

To generate sub-regionlets r_{ij}, we first compute a bounding box around a given regionlet R_i in the image domain. We then separate the bounding box in squares and compute the range measurements associated with them using the method described in Section 6.3.2. Sub-regionlets with less than a certain number of pixels and range measurements are considered insufficient and discarded.

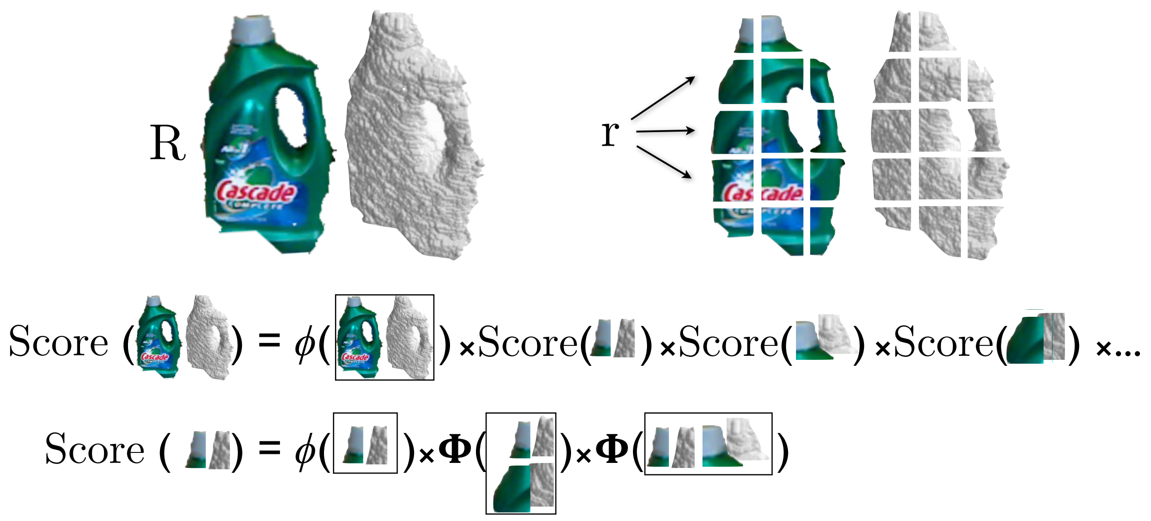

R
r
Score ( ) = ϕ( ) ×Score( ) ×Score( ) ×Score( ) ×...
Score ( ) = ϕ( )×Φ( )×Φ( )

Figure 6.4: Illustration of Regionlet and sub-regionlet scoring. (Top) Example Regionlet (R) and its division in sub-regionlets (r). (Bottom) Computing St scores for R given its sub-regionlets, and for a sub-regionlet r given its neighbors.

6.4 Structure Discovery as Regionlet Scoring

Once the multiple segmentations are computed and regionlets generated, we assume we have populated our candidate space with all potential objects we want to discover. We need now to score all regionlets in terms of their structure-ness and rank them from best to worst. In this section, we develop a framework to evaluate the structure-ness of regionlets, and define the different features we use in each data modality.

We evaluate regionlets at both the global level, i.e., considering each region as a whole, and at the local level via sub-regionlets. Therefore, the objective function we define is composed of unary terms at both the local and global level, and of pairwise terms at the local level.

Each regionlet R_i is composed of sub-regionlets r_{ij}. The set $\mathbf{r}$ of all sub-regionlets within a regionlet R is expressed as the *children* $\mathcal{C}$ of R, i.e., $\mathbf{r} = \mathcal{C}(R)$. Each sub-regionlet r_{ij} is characterized by a set of RGB values $\mathbf{I}_{ij}$, a set of 3D points $\mathbf{P}_{ij}$ and a set of neighbors $\mathbf{N}(r_{ij})$. In particular, each sub-regionlet r_{ij} is 4-connected to the adjacent sub-regionlets.

The final structure-ness score $\mathrm{St}(\cdot)$ for regionlet R is

$$\mathrm{St}(R;\Theta) = \phi(R;\Theta) \prod_{r\in\mathcal{C}(R)} \mathrm{St}(r;\Theta), \tag{6.2}$$

where

$$\mathrm{St}(r;\Theta) = \phi(r;\Theta) \prod_{r_j\in\mathbf{N}(r)} \Phi(r, r_i;\Theta). \tag{6.3}$$

The structure of this objective function is similar to a MRF (Huang and Ogata, 2002), although for the task of region scoring we do not need to perform any inference on this function. It is important to mention that, for stability purposes, the log-linear version $\log(\mathrm{St}(\cdot))$ is preferred over the score $\mathrm{St}(\cdot)$. Fig. 6.4 illustrates the process of scoring a regionlet and a sub-regionlet.

The features we explain in this section are inspired from previous work from Chu and Aggarwal (1993), Katz and Tal (2003), Lalonde et al. (2006), Hoiem et al. (2007), Tu and Zhu (2002). It is important to mention that, while the features we present work well in practice, they are just an example for the implementation of our structure discovery framework. Any other image-based, 3D-based, or mixed feature f to be computed in a group of data points such that $f(R_i) \in [0, 1]$ is a potential feature to be used in our framework.

6.4.1 Unary terms

The unary terms of regionlets $\phi(R)$ and sub-regionlets $\phi(r)$ are computed as the interaction between the different features described in Table 6.1. For a given set of features $\mathbf{f}$ and parameters Θ, the unary term

$$\phi(\cdot;\Theta) = \frac{1}{\sum_i w_i} \sum_{f_i \in \mathbf{f}} w_i f_i(\cdot;\Theta), \tag{6.4}$$

where the weights w_i account for the different importance of individual features when searching for particular types of structure. In order to be able to compare the different terms, we normalize all features (unless otherwise noted) to have a range $[0, 1]$, where 0 is the worst and 1 the best score a feature can achieve.

Appearance model

An appearance model is used to search for structure with particular visual properties. The appearance model $\mathrm{App}(\cdot)$ can be any image-based likelihood function that returns the confidence value in the range $[0, 1]$ of an image segment given model parameters Θ. For an extensive discussion and more details on appearance models for color and grayscale segmentation, see Tu and Zhu (2002). Examples of useful appearance models are:

- **Single color distribution**, to model non-textured, smooth regions.
- **Mixture of gaussians**, useful to model textured color regions.
- **Global color or intensity histograms**.

3D shape model

A 3D shape model is defined equivalently to the image-based appearance model in Section 6.4.1. The shape model $\mathrm{Sh}(\cdot)$ specifies which types of structures are desired. Useful shape priors include:

- **Planar shape**. A planar approximation $\hat{\mathbf{P}}$ of a set of 3D points $\mathbf{P}$ can be easily computed via PCA analysis or RANSAC. Then $\mathrm{Sh}(R;\Theta) \in [0,1]$ is a confidence measure of how planar R is.
- **Size**. It is not uncommon for structures within the same scene to have vastly different scales, and we can specify surface or volume priors on structures to prioritize which ones should be preferred.
- **Scale**. An alternative to a fixed size prior is to define a size dependent on distance. This way, we can discover small structures near the camera, and larger structures further away from the camera.

Self-continuity

The self-continuity feature measures abrupt changes in depth within a regionlet or sub-regionlet, which are often representative of discontinuities and boundaries between objects. Finding discontinuities in unstructured point clouds is a hard problem and multiple algorithms have been developed for this purpose, e.g., Tang et al. (2007). In our framework, we simplify this problem by accounting for the implicit ordering given by the image data. We construct a grid of control points in the image domain, and these control points are associated with the 3D points in the regionlet with minimal reprojection error.

In a sub-regionlet, the control points are equally spaced to form a grid of 64 control points. In a regionlet R_i, the control points are the centers of each sub-regionlet r_{ij}, and their connectivity maps that of the sub-regionlets. Once the Euclidean distances d_k between connected 3D control points have been computed, we define the continuity score $\mathrm{Cont}(\cdot;\Theta)$ as

$$\mathrm{Cont}(\cdot;\Theta) = \exp\left(-\frac{1}{w_{cont}^2}\frac{1}{|\bar{P}|}\max_k d_k\right), \tag{6.5}$$

where $|\bar{P}|$ is the average distance from regionlet R to the camera, which is necessary in order to reliably compare regionlet scores from different depths.

Verticality

Structures facing the camera and range sensor, i.e., parallel to the image plane, are more desirable than structures almost perpendicular to the image plane, as it is less reliable to estimate shape and appearance parameters on structures under heavy projective distortion. In addition, it is more complicated to use discovered structures under heavy projective distortion in further tasks, such as object modeling or object recognition. Therefore, we implement verticality as a feature in our regionlet-scoring framework.

We compute the verticality score as a ratio between the area projected in the image and the maximum area spanned by the set of 3D points $\mathbf{P}$ of regionlet R. We approximate the computation of the area of a regionlet by the area of its bounding box along the directions of maximum variation.

Contour compactness

Object boundaries in the real world are usually smooth and contain few jagged edges. This fact has been used in the image segmentation and sensor fusion literature to produce smoother segmentations of objects (e.g., Chu and Aggarwal (1993)). We use the same definition as Chu and Aggarwal (1993) for a Contour Compactness $\mathrm{CC}(\cdot)$ feature to encourage smooth edges. In particular, we measure the ratio of regionlet area to perimeter length, both in the image domain, normalized so that $\mathrm{CC}(\cdot) \in [0, 1]$.

6.4.2 Pairwise terms

Pairwise terms $\Phi(\cdot, \cdot)$ measure the interactions between neighboring sub-regionlets, in order to compute the likelihood that two sub-regionlets belong to the same structure.

For a given set of features $\mathbf{f}$ and parameters Θ, the pairwise term

$$\Phi(r_j, r_k; \Theta) = \frac{1}{\sum_i w_i} \sum_{f_i \in \mathbf{f}} w_i f_i(r_j, r_k; \Theta), \tag{6.6}$$

where the weights w_i account for the different importance of individual features when searching for particular types of structure. As with the Unary terms, we normalize all features (unless otherwise noted) to $\Phi(r_j, r_k; \Theta) \in [0, 1]$.

Pairwise continuity

The pairwise continuity feature measures abrupt changes in depth and discontinuities between two sub-regionlets. We follow a similar approach to Section 6.4.1, and reuse the same set of control points $\mathbf{P}_i^c \in r_i$ and $\mathbf{P}_j^c \in r_j$. We compute the pairwise continuity score as the average between the M minimal distances between $\mathbf{P}_i^c$ and $\mathbf{P}_j^c$, normalized as in Eq. (6.5).

Concavity

Studies in human perception have shown that concavities are one of the major cues in the human visual system to segment a scene into parts (Hoffman and Richards, 1984), and have been used successfully in 3D segmentation and mesh decomposition, as in Katz and Tal (2003). In our work, we compute the concavity between two sub-regionlets as the difference in orientation α_{ij} between their surface normals pointing towards the camera.

Given the importance of concavities in 3D segmentation, we can enforce a strong penalty on concave unions and reward convex unions by using

$$\mathrm{Cv}(r_i, r_j) = \sin \alpha_{ij}. \tag{6.7}$$

In this case, $\mathrm{Cv}(\cdot,\cdot) \in [-1, 1]$.

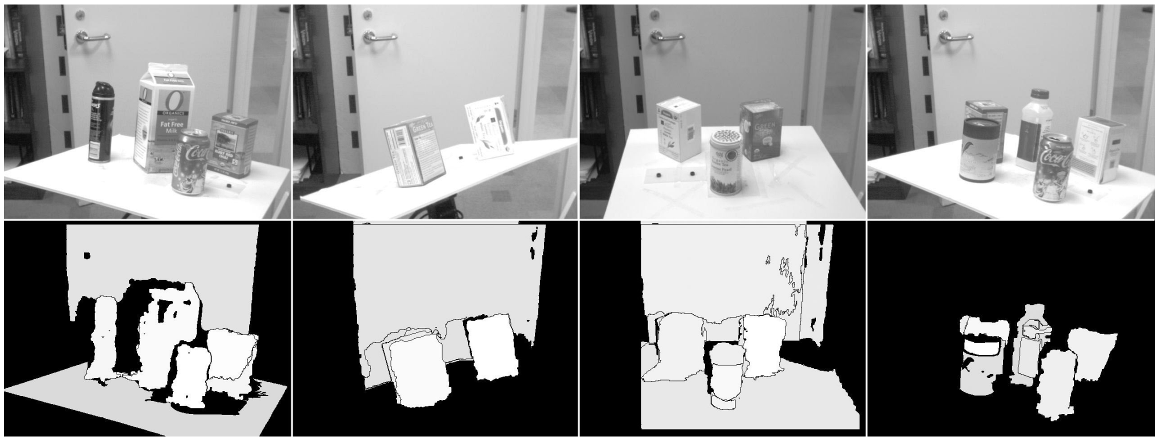


Figure 6.5: Examples of our structure discovery algorithm applied to household objects, in the Objects Pan-tilt Database. (Top row) Input images. (Bottom row) Top 10% ranked regionlets for each scene, color-coded from white to black, being white the highest ranked regionlet.

Projection

The Projection feature captures information about the smoothness of a surface, by computing the projection error of a planar approximation of a sub-regionlet onto its neighbor. This way, surfaces with small variation score high, since both sub-regionlets have similar global properties, while different shapes and orientations achieve a low score. Let $\hat{\mathbf{P}}_i^j$ be the projection of the 3D points $\mathbf{P}_i \in r_i$ onto r_j. The projection score $\mathrm{Proj}(\cdot,\cdot;\Theta)$ is then

$$\mathrm{Proj}(r_i, r_j; \Theta) = \exp\left(-\frac{1}{w_{\text{proj}}}\frac{1}{N}\sum_{k=1}^{N}||E_k||_2\right) \tag{6.8}$$

$$\mathbf{E} = \min(\mathbf{P}_i - \hat{\mathbf{P}}_i^j, \mathbf{P}_j - \hat{\mathbf{P}}_j^i) \tag{6.9}$$

Alignment

The Alignment feature grades the depth alignment of parallel surfaces. Despite this fact being partially captured by the pairwise continuity feature, we enforce the alignment of

	Single	Single, Non-transp.	Multiple	Total
Structure Discovery	52.7%	76%	61.2%	59.9%
Rusu and Cousins (2011)	38.8%	56%	66.8%	62.5%

Table 6.2: Object Discovery and number of objects per scene.

surfaces with a more robust feature that operates on average range measurements from regionlets, and not individual distances between points. For this task we re-use the Projection score from Section 6.4.2, but we measure the projection error of the vector difference of means, i.e., $E = \mu_i - \mu_j$, where μ_i, μ_j are the average of the sets of 3D points $\mathbf{P}_i \in r_i, \mathbf{P}_j \in r_j$.

Color histogram

This image-only feature captures the similarity in terms of appearance between sub-regionlets. Following Hoiem et al. (2007), we compute the distance between the color histogram (in $8 \times 8 \times 8$ bins) of each individual sub-regionlet compared to the histogram union of the two sub-regionlets, normalized so that $\text{ColorHist}(r_i, r_j; \Theta) \in [0, 1]$.

Surface compatibility

The surface compatibility feature is based on the 3D features described in Lalonde et al. (2006) of linear-ness l, planar-ness p and scatter-ness s, computed from the relative weights of the eigenvalues of the range data. We hypothesize that physical objects do not have abrupt changes in their surface properties. In other words, we assume that two planar surfaces are more likely to be parts of the same object than a planar and a spherical surface, or that two curvy surfaces are more likely to be the same object than a curvy surface and a plane. A simple way of encode this information is to compute the Euclidean distance between the two vectors $V_i = [l_i, p_i, s_i]$ and $V_j = [l_j, p_j, s_j]$ from r_i and r_j.

6.5 Experiments

Our goal in the experiments section is to compare our algorithm to the state of the art in object discovery in indoor scenes. We want to show that our algorithm performs as well as a specialized algorithm carefully optimized for the particular scene geometry of objects on top of a table. We also want to show that that our algorithm generalizes better to generic indoor scenes, because we do not enforce any limiting assumptions on the scene geometry. To that end, the first set of experiments focuses on the discovery of common household objects on top of a table, while the second one focuses on the discovery of larger structures such as people, tables or walls. Each dataset has been gathered with a different source of depth information (the Projected Textured Stereo of Konolige (2010), and an RGBD camera) to test the performance of our algorithm with different data sources.

6.5.1 Our implementation

In our implementation of the Structure Discovery algorithm, we use the following constants:

- Appearance/shape model: we use a simple maximality prior instead of a full appearance/shape model, in which we encourage the creation of large regionlets. In particular, we use $App(R;\Theta) = \log(|\mathbf{I}|)$, where $|\mathbf{I}|$ is the total number of pixels of regionlet R. We use a logarithm to avoid this feature from overpowering all others.

- Normalization weights: all weights used for normalization purposes, i.e., inside the exponentials, are set according to the uncertainty/noise characteristics of each sensor. For the stereo data, these are set so an average error of 1.5 cm at 1 m outputs a feature score of 0.5. The RGBD camera has an average uncertainty of 3%, so we set the weights so that an average error of 3% outputs a feature score of 0.5. This normalization scheme is described in more detail in Section 4.4.3.

- Feature weights: We learn a set of weights for each feature using grid search on a 100-image training dataset.

All our parameters are kept constant throughout our experiments to demonstrate the generality of our algorithm in discovering structure.

6.5.2 Baseline algorithm

The baseline algorithm we compare against is the plane-based 3D object segmentation algorithm from Radu Rusu's Point Cloud Library (Rusu and Cousins, 2011). This algorithm exploits domain knowledge of the geometry of common household environments; it is specialized in finding objects on planar surfaces, by first performing a plane-fitting procedure and then clustering groups of 3D points that lie on top of the plane. It is a simple method that performs remarkably well as long as the plane-fitting procedure is able to find valid planes, and it has been showcased on multiple occasions in most demonstrations of the PR2 Personal Robot.

6.5.3 Results

In this first experiment, we use a subset of 90 scenes extracted from Willow Garage's "Objects Pan-tilt Database" (WillowGarage, 2010) (some examples shown in Fig. 6.5).

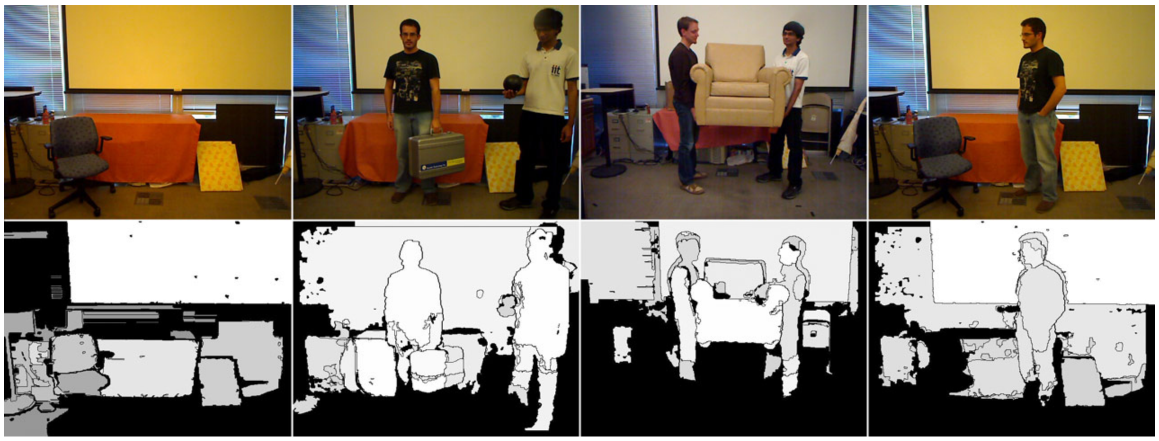


Figure 6.6: Examples of our structure discovery algorithm applied to household scenes, in the Household Dataset. (Top row) Input images. (Bottom row) Top 10% ranked regionlets for each scene.

This dataset contains different sets of household objects attached to a pan-tilt unit that is moved around, so that multiple views of the objects are available. Images from this dataset are grayscale, with a resolution of 640×480 pixels. The depth information is computed using the Projected Textured Stereo of Konolige (2010), a technique that extracts dense depth maps from images through the projection of a structured texture on the scene. The objects used in our evaluation have different shapes and appearances, and include a Gillette Shaving Cream, a Kleenex Cube, Mop'n'Glow floor cleaner, a soda can and a milk carton, among others. Some scenes contain multiple objects (up to 5) in different levels of occlusion and some contain single objects, for a total of 223 object instances in 90 scenes. Some examples of scenes and our segmentations are shown in Fig. 6.5.

In our evaluation, we ground truth each scene with a bounding box around each object, and use the PASCAL criteria (Everingham et al., 2010) bounding box evaluation to identify which objects are correctly discovered.

Our results are shown in Table 6.2. Both our algorithm (Structure Discovery, in Table 6.2) and the baseline from Rusu and Cousins (2011) perform similarly well, despite using vastly different approaches. It is interesting to note that our algorithm tends to discover objects more reliably in scenes with little clutter. The explanation for this tendency is that our algorithm is designed to detect prominent structures in the scene, which is very often correlated to the size of the structures in the image. In some of the highly cluttered scenes, the objects seldom span more than a single sub-regionlet, and contain little 3D information. Under these circumstances, the algorithm of Rusu and Cousins (2011) is a more convenient choice.

For the single object experiments, we split our results between "Single" and "Single,

	Walls	Person	Person (torso)	Furniture	Other
Total Present	12	14	14	11	13
Found (%)	100%	35.7%	78.5%	63.6%	61.5%

Table 6.3: Structure Discovery on indoors scenes. Results from Rusu *et al.* show no detections and are omitted from the table.

non-transparent" because neither algorithm is able to discover a transparent wine glass in the dataset, mainly because of a lack of consistent range data in its surface, as the Projected Texture Stereo algorithm fails to recover stereo data from it. In the single object experiments, our algorithm outperforms the baseline by 20%. This difference, however, is a bit misleading, since both algorithms perform equally well on all objects but one (the Mop'n'Glow bottle). In a general setting, we believe that these two algorithms perform very similarly, as long as there is a planar surface for the baseline algorithm to detect.

The second experiment we conduct is the discovery of larger objects in indoor environments, such as walls, furniture and people. In order to do so, we gathered a dataset of 15 indoor scenes, which we call the Household Dataset. The image/range sensor is an RGBD camera that outputs 640×480 resolution images with associated depth for every pixel. We downsample the range data to one third of the original resolution to verify that we do not require a pixel-level fusion of data sources. We have annotated and produced bounding boxes for four types of structure: wall, person, furniture, and "other". The label "other" is for other prominent objects in the scene, such as a backpack, a painting, a cardboard box or a suitcase.

For this experiment we are unfortunately unable to provide any results from the baseline system, as it is optimized for a much shorter range than these scenes, and does not return any detections in these scenarios.

Results from this dataset are shown on Table 6.3, and example scenes and their highest ranked regionlets are shown in Fig. 6.6. Analyzing the results, we see that large, simple structures such as walls are discovered very reliably, while more complex objects such as people are often missed, in particular their legs. If we focus on the upper body of a person, we find a twofold increase in performance, as a person's upper body is often larger and has less variability.

6.6 Summary

We have presented and validated an algorithm to perform structure discovery from RGBD data using a novel region-based approach. We have demonstrated that our algorithm is able to discover common household objects with similar accuracy than specialized 3D object segmentation algorithms, without the need to rely on any rigid assumptions about the scene structure, such as the presence of a visible planar surface in which objects are placed.

Interestingly, the results from both algorithms are almost complementary in the kind of scenes they perform best. Our algorithm discovers objects best when the objects' largest faces are parallel to the image plane, as the image and range sensors capture more information about them. On the other hand, the plane-based 3D segmentation of Rusu and Cousins (2011) performs best when the table is the most prominent part of the scene, e.g., seen from a high viewpoint.

An interesting follow-up work to this algorithm would be a higher-level reasoning about the interpretation of a scene; in the case of an overhead picture, the detection of a planar surface could lead to a closer inspection of objects on top of it.

A close analysis on the limitations of our algorithm shows some important conclusions and areas of improvement. We have found that a bad performance of our algorithm is often tied to the bad performance of the initial segmentations. On multiple occasions where an object is missed, the reason is that none of the initial segmentations was able to capture that object in its entirety in a single segment. When this happens, our assumptions do not hold and thus our algorithm cannot generate the expected results. In addition, while we discover structures such as walls with high accuracy, our structure model is sometimes not flexible enough to handle the large variations of more complex structures such as people or some furniture, as they are seldom segmented in their entirety.

We are confident that the introduction of regionlets as the minimal processing units is an important step in image-range data sensor fusion. Regionlets supersede and integrate low- and mid-level fusion in one framework, extracting the advantages from both approaches. Regionlets are compatible with existing algorithms that require one-to-one correspondences, while adding an extra layer of abstraction that may lead to more sophisticated perception tasks using mixed image and range data.

In the next chapter, Section 7, we describe how the Structure Discovery algorithm we propose can be used for Robotic Object Discovery.

Chapter 7

Robotic Object Discovery

Discover. [v. dis·cov·er. dis-ˈkə-vər]: *To obtain sight or knowledge of for the first time.*
Merriam-Webster Dictionary

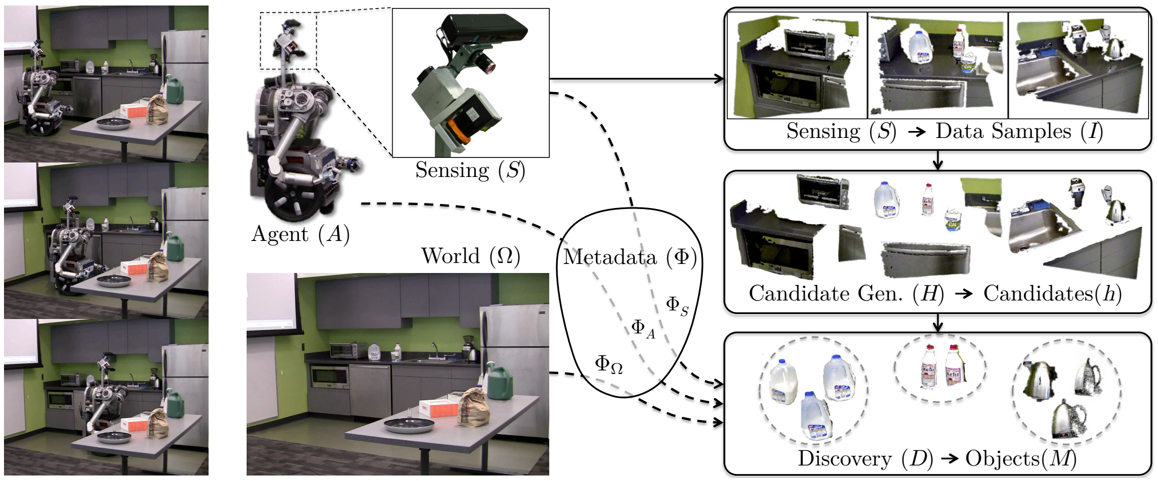
In this chapter, we consider the problem of *Lifelong Robotic Object Discovery* (LROD) as the long-term goal of discovering novel objects in the environment while the robot operates, for as long as the robot operates. As a first step towards LROD, we automatically process the raw video stream of an entire workday of a robotic agent and discover hundreds of objects.

We claim that the key to achieve this goal is to incorporate metadata whenever available, in order to detect and adapt to changes in the environment. We propose a general graph-based formulation for LROD in which generic metadata is encoded as constraints. Our formulation enables the introduction of new sources of metadata to be added dynamically to the system, as they become available or as conditions change. We describe an optimized implementation of this framework in HERB, which we term HerbDisc, to discover objects in a raw RGBD video stream of over 6 hours of duration. With HerbDisc, we process 6 h 20 min of RGBD video of real human environments in 18 min 34 s, and discover 206 novel objects with their 3D models.

The research presented in this chapter is joint work with Bo Xiong and Corina Gurau.

7.1 Problem Formulation

Consider the example of Robotic Object Discovery shown in Fig. 7.1. We identify five major components. The *World* Ω represents the underlying physical phenomenon (i.e., the environment) in which we discover objects. A *physical agent* $\mathcal{A}$ (e.g., a robot) gathers data through observation of the world Ω. The physical agent uses *sensors* $\mathcal{S}$ (e.g., a camera) to



Sensing (S)
Agent (A)
World (Ω)
Metadata (Φ)
Φ_S
Φ_A
Φ_Ω
Sensing (S) → Data Samples (I)
Candidate Gen. (H) → Candidates(h)
Discovery (D) → Objects(M)

Figure 7.1: Main components in Robotic Object Discovery. (left) the robot HERB moves through a kitchen searching for novel objects. (center) The three physical components of Robotics Object Discovery are: the world Ω, the robotic agent $\mathcal{A}$, and the sensors $\mathcal{S}$. (right) The sensors capture data samples x to be processed by a candidate generator $\mathcal{H}$ to produce object candidates. The Discoverer $\mathcal{D}$ groups recurring object candidates into objects, using candidate data and metadata sources Φ_Ω (e.g., assumption "objects lie on tables"), $\Phi_\mathcal{A}$ (e.g., robot localization data), $\Phi_\mathcal{S}$ (e.g., image ordering and timestamps).

gather data samples I (e.g., images). A *candidate generator* $\mathcal{H}$ produces object candidates h from data samples. Finally, the *discoverer* $\mathcal{D}$ groups recurring object candidates into objects.

In this chapter, we describe a general architecture for an object discoverer $\mathcal{D}$ that uses metadata from the world Ω, the physical agent $\mathcal{A}$ and the sensors $\mathcal{S}$, alongside visual information from the object candidates h, to discover objects robustly and efficiently.

7.1.1 Inputs and Outputs

The visual input to HerbDisc is a set $\boldsymbol{I}$ of N images with associated range data:

$$\mathbf{I} = \{I_1, \ldots, I_n, \ldots, I_N\} \qquad I_n = \{I_n^{\boldsymbol{rgb}}, I_n^{\boldsymbol{P}}\}, \tag{7.1}$$

where $I_n^{\boldsymbol{rgb}}$ is the set of color RGB values in image n, and $I_n^{\boldsymbol{P}}$ is the set of 3D points available from the viewpoint of image n.

A candidate generator $\mathcal{H}$ generates a set of data fragments $\boldsymbol{h}$ from image and range data in $\boldsymbol{I}$, which we consider the object candidates. Each object candidate

$$h_i = \{h_i^{\boldsymbol{rgb}}, h_i^{\boldsymbol{P}}, h_i^{\boldsymbol{\Phi}}\} \tag{7.2}$$

is defined by a set of image pixels $h_i^{\boldsymbol{rgb}}$, a set of 3D points $h_i^{\boldsymbol{P}}$, and a set of metadata attributes $h_i^{\boldsymbol{\Phi}}$.

The output of this framework is a set of 3D object models $\boldsymbol{M}$. Each object $M_k = \{M_k^{\boldsymbol{rgb}}, M_k^{\boldsymbol{P}}, M_k^{\boldsymbol{h}}\}$ is defined by a set of 3D points $M_k^{\boldsymbol{P}}$ with associated color $M_k^{\boldsymbol{rgb}}$ and the set of object candidates $M_k^{\boldsymbol{h}} = \{h_{1,k}, \ldots, h_{i,k}, \ldots\}$ used to create object M_k.

7.1.2 Constraints

Constraints encode generic information about an object candidate h_i or a relationship between candidates h_i, h_j. There are two types of constraints: node constraints Θ^{n} (which encode information about a single candidate) and edge constraints Θ^{e} (which encode information about the relationship between a pair of candidates). Table 7.1 shows a list of the constraints we use in this work. We model each constraint Θ as a Bernoulli distribution with probability of success p (and, conversely, a probability of failure $q = 1 - p$). Node constraints Θ^{n} modify a single object candidate h_i,

$$\Theta^{\mathrm{n}} : h_i \mapsto \{0, 1\} \tag{7.3}$$

$$P(\Theta^{\mathrm{n}}(h_i) = 1 | h_i) = p. \tag{7.4}$$

Analogously, edge constraints Θ^{e} modify the edge between a pair of object candidates h_i, h_j, such that

$$\Theta^{\mathrm{e}} : h_i, h_j \mapsto \{0, 1\} \tag{7.5}$$

$$P(\Theta^{\mathrm{e}}(h_i, h_j) = 1 | h_i, h_j) = p. \tag{7.6}$$

In a slight abuse of notation, we use the forms $P_{\Theta^{\mathrm{n}}}(h) \equiv P(\Theta^{\mathrm{n}}(h) = 1|h)$ and $P_{\Theta^{\mathrm{e}}}(h_i, h_j) \equiv P(\Theta^{\mathrm{e}}(h_i, h_j) = 1|h_i, h_j)$ in the remainder of this chapter.

7.2 Framework overview

This section contains a brief summary of the discovery framework and its components, alongside a description of how each component is implemented in HerbDisc. We explore each component in detail in the following sections.

We describe the general flowchart of our framework in the following itemized list. In the following sections, we focus on the novel elements of this work: defining constraints (Section 7.3), generating CSGs (Section 7.3.3), and the implementation of constraints and CSGs in HerbDisc (Section 7.5). We provide a list of the constraints implemented in HerbDisc in Table 7.1.

1. Candidate Generation. We compute object candidates h_i from each data sample $I_n \in \boldsymbol{I}$. We use the *objectness*-based segmentation algorithm described in Chapter 6 (Structure Discovery).

Constraint	Type	Information	Source	Description	Section
Θ_{motion}	node	Relative camera motion	$\Phi_{\mathcal{A}}$	Acquire data samples only if there is motion (no repeated frames).	7.5.2
Θ_{seq}	edge	"data comes in sequences"	$\Phi_{\mathcal{S}}$	Split data stream in short sequences based on camera motion and maximum sequence length.	7.5.2
Θ_{support}	node	"objects have surfaces of support"	Φ_{Ω}	Reject candidates not supported by horizontal or vertical planes (tables or walls).	7.5.1
Θ_{static}	edge	"scene is static for a few seconds"	Φ_{Ω}	Measure 3D overlap between candidates.	7.5.3
Θ_{size}	node	Object size	Φ_{Ω}	Compare candidate's size with object prior.	7.5.4
Θ_{shape}	node	Object shape	Φ_{Ω}	Compare candidate's shape with object prior.	7.5.4
Θ_{app}	edge	Visual Similarity	V	Compare visual similarity between candidates using color histograms.	7.5.5
Θ_{3D}	edge	Shape Similarity	V	Compare shape similarity between candidates using FPFH features.	7.5.5

Table 7.1: Constraints used in HerbDisc. For each constraint Θ_i, we provide: the type of information encoded in Θ_i; whether Θ_i is applied on a single object candidate (node) or a relation between a pair of candidates (edge); the information source(s) encoded in Θ_i; a short description of the meaning of Θ_i; and the section in which Θ_i is described in detail. The possible sources of information are: Φ_{Ω} (metadata about the environment), $\Phi_{\mathcal{A}}$ (metadata about the robot), $\Phi_{\mathcal{S}}$ (metadata about the sensors), or V (visual information).

2. CSG Generation. We create a graph of relationships between object candidates using constraints Θ. We define the CSG built by constraint Θ as $G^{\Theta} = (E^{\Theta}, V^{\Theta})$ (Section 7.3.3).

If the constraint Θ encodes visual similarity, then the CSG G^{Θ} is equivalent to regular pairwise similarity graphs in Unsupervised Object Discovery (e.g., Kang et al. (2011)). Applying the constraints in Table 7.1 to create G^{Θ} produces multiple connected components G^{Θ}_g.

3. CSG Clustering. We compute groups of candidates for each $G^{\Theta}_g \in G^{\Theta}$ with the graph partitioning algorithm of Brandes (2001). This algorithm is a greedy community discovery method based on the Betweenness Centrality metric, which is very efficient for sparse graphs and works well for our problem.

Each cluster C_i contains a set of object candidates $\boldsymbol{h}_i$, which are registered together and merged to compute partial 3D models m_i. The set of all partial models discovered is

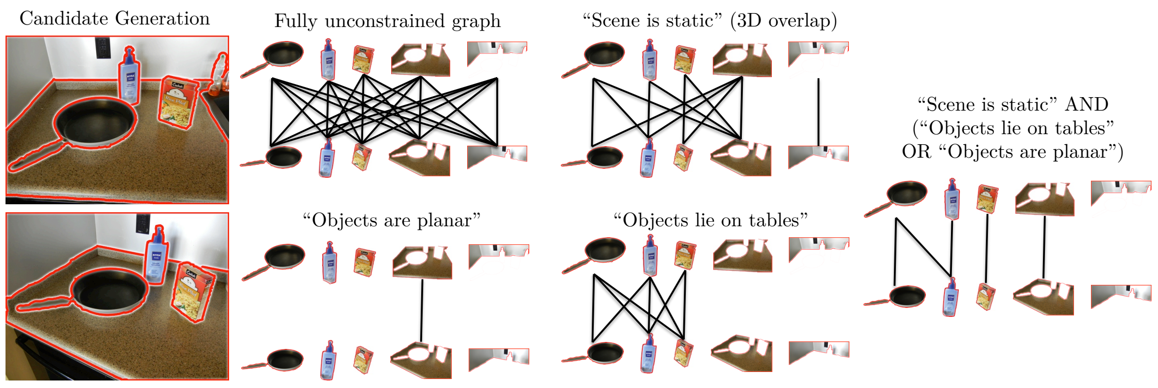

Candidate Generation
Fully unconstrained graph
"Scene is static" (3D overlap)
"Scene is static" AND ("Objects lie on tables" OR "Objects are planar")
"Objects are planar"
"Objects lie on tables"

Figure 7.2: Metadata induces constraints on pairwise similarity graphs. We illustrate this effect on a pair of manually segmented images for simplicity. The fully unconstrained graph is seldom computed in practice, as techniques such as inverted indexes are used to preselect potential matches (Philbin et al., 2010). Our formulation generalizes such techniques, constraining a graph based on any source of metadata (columns 2-3). Most importantly, our formulation facilitates the creation of complex rules from the combination of multiple sources of metadata (column 4).

denoted as $\boldsymbol{m}$.

Each object $m_i = \{m_i^{\boldsymbol{rgb}}, m_i^{\boldsymbol{P}}, m_i^{\boldsymbol{h}}\}$ is defined by a set of 3D points $m_i^{\boldsymbol{P}}$ with associated color $m_i^{\boldsymbol{rgb}}$ and the set of object candidates $m_i^{\boldsymbol{h}}$ used to create object m_i.

4. Object CSG Graph Generation. We compute a CSG graph $G^m = (E^m, V^m)$ over partial object models $m_i \in \boldsymbol{m}$. The number of nodes in this graph is orders of magnitude smaller than G^Θ, so we can afford to compute more complex constraints if needed. Only a subset of the constraints from Table 7.1 are available for partial object models m_i. In particular, we use Θ_{size}, Θ_{shape}, Θ_{app}, and Θ_{3D}, as the others require local information that is not relevant for the partial objects.

5. Object Clustering. We compute clusters of partial 3D models using the graph partitioning algorithm of Brandes (2001) on the graph G^m. Each cluster C_i contains a set of partial object models m_i.

6. 3D model generation. We generate full object models M_i from clusters of partial object models C_i. We globally register the partial models with the Global Alignment algorithm of Borrmann et al. (2008) to produce full 3D models.

7.3 Information as Constraints

In the introduction, we defined a constraint Θ as a *measurable* yes/no question about a node or edge, with probability of success p about the answer. In Section 7.1, we modeled each constraint Θ as a Bernoulli distribution. In this section, we describe how to encode information as constraints; we define logic operations of constraints that allow us to create complex constraint expressions; and how to compute CSGs from constraints.

7.3.1 Defining Constraints

Consider the pair of scenes illustrated in Fig. 7.2, in which we encode as constraints the assumptions $\Theta^{\mathrm{n}}_{\mathrm{planar}}$ = "objects are planar", $\Theta^{\mathrm{e}}_{\mathrm{static}}$ = "scene is static", and $\Theta^{\mathrm{n}}_{\mathrm{tables}}$ = "objects lie on tables". Encoding Θ_{planar} requires answering the question "is candidate h_i planar?". If we can measure whether an object is planar or not (e.g., by computing the reconstruction error of a planar approximation of h_i's 3D points), then we can encode the assumption as a constraint, with the result shown in Fig. 7.2(row 2, col 2). Similarly, to encode Θ_{tables} we must answer the question "is candidate h_i on a table?" for which we need to 1) detect a table, and 2) determine if candidate h_i is on it. If we can measure these two factors, then the assumption can be encoded as a node constraint, with the result shown in Fig. 7.2(row 2, col 3). Finally, the assumption "the scene is static" implies to answer affirmatively that "Do candidates h_i at time t and h_j at time $t+1$ occupy the same location in space?" If we can register the two scenes and h_i and h_j occupy the same 3D location, then Θ_{static} would be satisfied with p proportional to the overlap between h_i and h_j. The result of Θ_{static} is shown in Fig. 7.2(row 1, col 3).

Some sources of metadata may also operate over both nodes and edges. For example, object tracking can be encoded as a union of an edge constraint Θ^{e} = "are candidates h_i at time t and h_j at time $t+1$ the same object?," and a node constraint Θ^{n} = "is candidate h_i being tracked?" To incorporate such constraints, we redefine the constraint Θ as a pair $\Theta \equiv (\Theta^{\mathrm{n}}, \Theta^{\mathrm{e}})$. Constraints that operate only on nodes or edges should implement a default operator for nodes ($\Theta^{\mathrm{n}} = 1$) or edges ($\Theta^{\mathrm{e}} = 1$) which satisfies the constraint with $p = 1$ for any input.

Pairwise similarity functions also induce constraints Θ. In particular, a normalized similarity function $s(h_i, h_j) \in [0, 1]$ induces an edge constraint Θ^{e} with $P_{\Theta^{\mathrm{e}}}(h_i, h_j) = s(h_i, h_j)$. In HerbDisc, we do not distinguish between visual similarity and metadata: they are all encoded as constraints Θ_i. This unification is very useful to combine multiple constraints (Section 7.3.2) and build CSGs (Section 7.3.3).

7.3.2 The Logic of Constraints

A key consequence of our generic constraint formulation is that we can seamlessly combine multiple sources of metadata using logic statements. In order to take full advantage of Boolean algebra, we define the logic operations of conjunction $\wedge$, disjunction $\vee$ and negation $\neg$ over node and edge constraints induced by metadata. Let Θ^{n}_i, Θ^{n}_j be independent node constraints induced by metadata, and $P_\Theta(h)$ the probability of candidate h satisfying constraint Θ^{n}. Then, the negation operator $\neg\Theta_i^{\mathrm{n}}$ is computed as

$$P_{\neg\Theta_i^{\mathrm{n}}}(h) = 1 - P_{\Theta_i^{\mathrm{n}}}(h), \tag{7.7}$$

which represents the probability of h *not* satisfying constraint Θ^{n}. The conjunction operator $\Theta_i^{\mathrm{n}} \wedge \Theta_j^{\mathrm{n}}$ is then computed as

$$P_{\Theta_i^{\mathrm{n}} \wedge \Theta_j^{\mathrm{n}}}(h) = P_{\Theta_i^{\mathrm{n}}}(h) P_{\Theta_j^{\mathrm{n}}}(h). \tag{7.8}$$

Finally, the disjunction operator $\Theta_i^{\mathrm{n}} \vee \Theta_j^{\mathrm{n}}$ is computed as

$$P_{\Theta_i^{\mathrm{n}} \vee \Theta_j^{\mathrm{n}}}(h) = 1 - P_{\neg\Theta_i^{\mathrm{n}} \wedge \neg\Theta_j^{\mathrm{n}}}(h). \tag{7.9}$$

We analogously define the conjunction $\wedge$, disjunction $\vee$ and negation $\neg$ operators for edge constraints, by substituting $P_{\Theta^{\mathrm{n}}}(\cdot)$ for $P_{\Theta^{\mathrm{e}}}(\cdot,\cdot)$ in Eq. (7.7), Eq. (7.8) and Eq. (7.9).

Logic operations over constraint pairs $\Theta = (\Theta^{\mathrm{n}}, \Theta^{\mathrm{e}})$ operate on Θ^{n} and Θ^{e} independently, so that

$$\begin{aligned} \neg\Theta_i &= (\neg\Theta_i^{\mathrm{n}}, \neg\Theta_i^{\mathrm{e}}) & (7.10)\\ \Theta_i \vee \Theta_j &= (\Theta_i^{\mathrm{n}} \vee \Theta_j^{\mathrm{n}}, \Theta_i^{\mathrm{e}} \vee \Theta_j^{\mathrm{e}}) & (7.11)\\ \Theta_i \wedge \Theta_j &= (\Theta_i^{\mathrm{n}} \wedge \Theta_j^{\mathrm{n}}, \Theta_i^{\mathrm{e}} \wedge \Theta_j^{\mathrm{e}}) & (7.12) \end{aligned}$$

Any logic operation can be derived from the conjunction, disjunction and negation operators. We can now define arbitrarily complex constraint expressions based on logic operations over primitive constraints. In Fig. 7.2(row 1, col 4), we illustrate this behavior with the three hard constraints: Θ_{static}, Θ_{tables}, Θ_{planar}. To search for objects assuming that “the scene is static” AND that “objects that lie on tables” OR “objects are planar”, we simply define the constraint

$$\Theta = \Theta_{\text{static}} \wedge (\Theta_{\text{tables}} \vee \Theta_{\text{planar}}). \tag{7.13}$$

A generic constraint Θ can be composed of multiple constraints Θ_i using the logic operators defined above,

$$\Theta = \Theta_1 \circ \Theta_2 \circ \ldots \circ \Theta_i \circ \ldots, \tag{7.14}$$

where the composition operator $\circ$ denotes any logic operation using Boolean algebra.

7.3.3 Constrained Similarity Graphs

CSGs are undirected graphs which encode information from constraints into nodes, edges, node weights and edge weights. Let $G^\Theta = (E^\Theta, V^\Theta)$ be an undirected pairwise graph. G^Θ is a CSG of constraint Θ if and only if:

1. Every node $h_i \in V^{\Theta}$ satisfies Θ^{n},
2. every edge $e_{i,j} \in E^{\Theta}$ satisfies Θ^{e}, and
3. has node weights $w(h_i) = P_{\Theta^{\mathrm{n}}}(h_i)$, and edge weights $w(h_i, h_j) = P_{\Theta^{\mathrm{e}}}(h_i, h_j)$.

We generate the CSG G^{Θ} for constraint Θ following Algorithm 1.

Algorithm 1 Building a Constrained Similarity Graph

```
 1: $V^{\Theta} = \varnothing$
 2: $E^{\Theta} = \varnothing$
 3: for $h_i$ in $\boldsymbol{h}$ do                                ▷ Add nodes that satisfy $\Theta$
 4:     if $P_{\Theta^{\mathrm{n}}}(h_i) > p_{\min}$ then
 5:         $V^{\Theta} \leftarrow V^{\Theta} \bigcup \{h_i\}$
 6:         $w(h_i) \leftarrow P_{\Theta^{\mathrm{n}}}(h_i)$
 7: for $h_i$ in $\boldsymbol{V^{\Theta}}$ do                         ▷ Add edges that satisfy $\Theta$
 8:     for $h_j$ in $\boldsymbol{h}$ with $j > i$ do
 9:         if $P_{\Theta^{\mathrm{e}}}(h_i, h_j) > p_{\min}$ then
10:             $E^{\Theta} \leftarrow E^{\Theta} \bigcup \{e_{i,j}\}$
11:             $w(e_{i,j}) \leftarrow P_{\Theta^{\mathrm{e}}}(h_i, h_j)$
```

In Algorithm 1, $p_{\min}$ denotes the threshold probability for nodes and edges (in normal conditions, $p_{\min} = 0.5$). The CSG construction and the entire framework are independent of the particular choice of Θ. Θ can be any arbitrarily complex constraint expression, ranging from visual similarity only (in which case, the CSG becomes a regular pairwise similarity graph) to multiple sources of metadata and visual similarity, as we implement in HerbDisc.

Building a generic CSG has necessarily a worst-case complexity of $\mathcal{O}(n^2)$, where $n = |\boldsymbol{h}|$, since the CSG must be able to build any graph including pairwise similarity graphs, or even complete graphs, which are $\mathcal{O}(n^2)$. In addition, evaluating a constraint expression for a node or edge can be expensive, especially if computing complex visual similarities.

In practice, we can simplify the construction of a CSG by using conjunctive constraint expressions (as in Eq. (7.8)), and positioning the most restrictive constraints first. Evaluating a conjunctive constraint expression is much faster than evaluating generic constraint expressions, as we only need to evaluate a constraint in the constraint expression if all previous constraints are successful.

Consider a constraint Θ_0 that generates the CSG $G^{\Theta_0} = (E^{\Theta_0}, V^{\Theta_0})$. We can compute the CSG G^{Θ} from G^{Θ_0} using conjunctive constraints as in Algorithm 2.

The complexity of Algorithm 2 for a given Θ and G^{Θ_0} is $\mathcal{O}(|E^{\Theta_0}|)$. The size of the graph G^{Θ_0} determines the complexity of building G^{Θ}. Therefore, an appropriate choice of Θ_0 to build a sparse CSG very quickly can greatly improve the performance of the overall

Algorithm 2 Building a CSG with conjunctive constraints

```
1:  $V^{\Theta} = V^{\Theta_0}$
2:  $E^{\Theta} = E^{\Theta_0}$
3:  for $h_i$ in $V^{\Theta}$ do
4:      for $\Theta_k$ in $\mathbf{\Theta}$ do                    ▷ Erase nodes that do not satisfy $\Theta_k$
5:          if $P_{\Theta_k^{\mathrm{n}}}(h_i) < p_{\min}$ then
6:              $V^{\Theta} \leftarrow V^{\Theta} - \{h_i\}$
7:              break
8:          else
9:              $w(h_i) \leftarrow w(h_i) P_{\Theta_k^{\mathrm{n}}}(h_i)$
10: for $h_i$ in $\boldsymbol{V^{\Theta}}$ do
11:     for $\Theta_k$ in $\mathbf{\Theta}$ do                    ▷ Erase edges that do not satisfy $\Theta_k$
12:         for $h_j$ in $\mathcal{N}^{\Theta}(h_i)$ do
13:             if $P_{\Theta_k^{\mathrm{e}}}(h_i, h_j) < p_{\min}$ then
14:                 $E^{\Theta} \leftarrow E^{\Theta} - \{e_{i,j}\}$
15:                 break
16:             else
17:                 $w(e_{i,j}) \leftarrow w(e_{i,j}) P_{\Theta_k^{\mathrm{e}}}(h_i, h_j)$
```

algorithm. Some of the natural constraints in service robotics are excellent for this purpose, such as spatiotemporal constraints. The motion and sequencing constraints $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ from Table 7.1 that we define in HerbDisc (see Section 7.5.2 for details) split the data stream into subsets of samples with limited motion and at most m samples per subset. Using $\Theta_0 = \Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ yields a CSG G^{Θ_0} with $|E^{\Theta_0}| = \mathcal{O}(nm) \approx \mathcal{O}(n)$ edges, considering that m is fixed and $n \ll m$ in realistic situations (in the NSH Dataset, $m = 50$ and $n = 521234$). Under these conditions, the CSG construction, given G^{Θ_0}, has a complexity of $\mathcal{O}(n)$ for the remaining constraints $\Theta_k \in \mathbf{\Theta}$. Given that the visual similarities are the most expensive constraints, it is crucial to perform this optimization to only compute $\mathcal{O}(n)$ similarities. See Table 7.2 for a quantitative evaluation of the reduced complexity of this method.

The constraints Θ and the generic CSG construction of Algorithm 1 are designed for both soft constraints (i.e., Θ such that $P_{\Theta} \in [0,1]$) and hard constraints (i.e., Θ such that $P_{\Theta} \in 0, 1)$. In HerbDisc, we use Algorithm 2 with a mix of soft and hard constraints. The hard constraints are positioned first in the constraint expression to purposefully split the CSG into many small connected components as quickly as possible. We then use soft constraints to better evaluate the nuances of appearance and shape similarity for those candidates with real potential of being part of the same object.

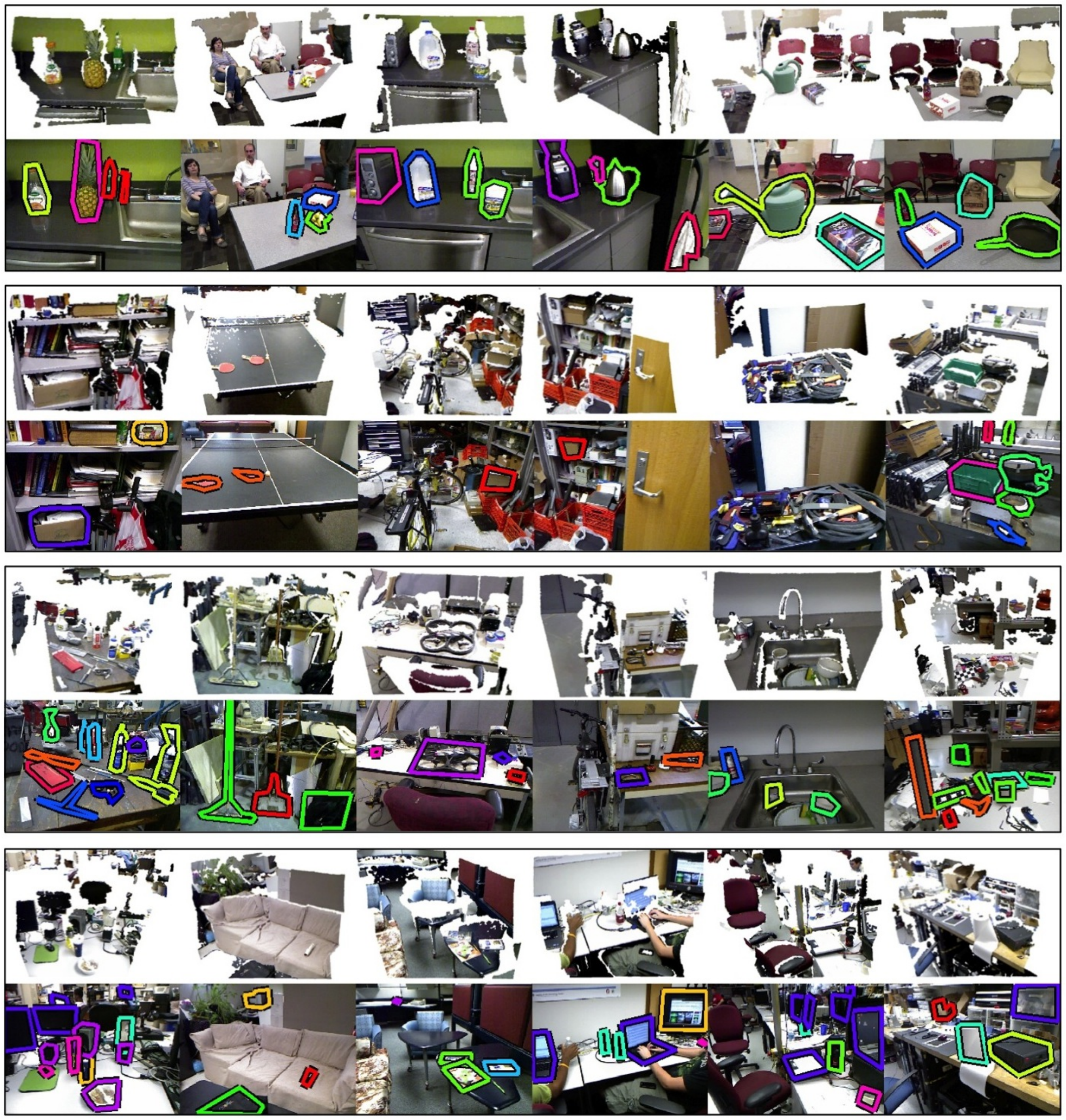


Figure 7.3: The Kitchen Dataset (top row) and the NSH Dataset (bottom three rows). Each row depicts the Kinect 3D point clouds (top) and their corresponding images with ground truth annotations (bottom) for some of the environments we visited. The Kitchen Dataset captures a low-clutter environment with 20 objects of interest. The NSH Dataset captures office and lab environments, ranging from moderate to extreme clutter. Some scenes were so challenging (e.g., row 2, col 3-5) that the annotators could not separate the objects in the scene.

7.4 Datasets

We present two datasets of real human environments in which we evaluate HerbDisc: the Kitchen Dataset and the NSH Dataset (see Fig. 7.3). We recorded both datasets by driving HERB around the environment and capturing data with a pair of extrinsically calibrated cameras: a Kinect RGBD camera at 640×480 resolution and a Point Grey Flea2 at 1024×768 resolution. The framerate is set to 30 fps on both cameras, but due to throughput limitations the effective framerate is approximately 22 fps. For the remainder of this chapter, we refer to the synchronized Kinect image, depth image, and Flea2 image, as a *data sample*.

We manually annotated both datasets to obtain ground truth, with the following labeling procedure. Our goal is to obtain the list of objects that HERB could potentially grasp. Since it is infeasible to annotate every single data sample—there are over half a million—we process each data stream with a motion filter to eliminate redundant samples (the same motion filter used in HerbDisc, described in Section 7.5.2). Then, we select between 1 and 10 samples from each office, lab, kitchen, etc., we visited, showing the maximum amount of different objects, and label all objects with the LabelMe tool Russell et al. (2008). As a rough estimate of the objects that HERB can grasp, we consider valid any object that:

- is at least 10×5 cm in its two largest dimensions (e.g., a smartphone),
- is at most 60 cm long in its longest dimension (e.g., a monitor),
- appears unoccluded in at least one data sample, and
- is movable, with free space around it to be grasped (e.g., a stack of books in a bookshelf is not labeled).

Fig. 7.3 shows examples of data samples from the Kitchen (top row) and NSH dataset (bottom 3 rows), alongside the annotated data.

The Kitchen Dataset captures four 3-minute recordings of HERB in a kitchen environment, with relatively clean scenarios and 20 ground truth objects that HERB must discover. We refer to the four individual recordings as Kitchen-1 to Kitchen-4, and their union (a 12-minute recording with 14282 data samples) as the Kitchen Dataset.

The NSH Dataset is a stream of 6 hours and 20 minutes of HERB exploring the NSH building of Carnegie Mellon University, comprising 521234 data samples. We divided the recording in four fragments lasting between 1 h and 1 h 50 min each, one per building floor. We refer to the four individual recordings as NSH-1 to NSH-4, and the full-length stream as the NSH Dataset. For this dataset, we visited over 200 real offices and laboratories to capture the real conditions in which people work, with scenes ranging from moderate to extreme clutter. This dataset also captures the wide differences in lighting conditions in human environments (from dim light to bright sunlight), which degrade the data acquisition and to which a lifelong agent must be robust. We labeled a total of 423 unique ground truth objects.

7.5 Implementation of HerbDisc

In this section, we describe the novel components of HerbDisc, focusing on how to formulate similarities, assumptions, and other metadata from Table 7.1 as constraints. The advantage of formulating the different components as constraints is the adaptability of the system. We can completely control and modify the behavior of HerbDisc (e.g., to adapt it a particular task) without modifying a single line of code, as HerbDisc only depends on the constraint expression Θ to construct the CSGs. For example, we could measure if the assumptions for specific algorithms hold before using them, and revert to safer algorithms if they do not, modifying only the constraint expression. By modifying Θ when environmental conditions change, we can adapt and opportunistically select the best constraints for each task.

We show experimental results on the impact of each component in the different subsections. See Section 7.6 for a description of the baseline and the evaluation procedure.

7.5.1 Constrained Candidate Generation

The candidates h produced by a candidate generator $\mathcal{H}$ can be refined with constraints to adapt to the particular algorithm assumptions, either by entirely enabling/disabling a candidate generator based on metadata, or by rejecting unnecessary candidates for the particular task. An example of such a constraint would be the requirement that "objects lie on tables".

Candidate generators that rely on metadata are common in the robotics literature. For example, algorithms that track objects (Morwald et al., 2010), that assume tabletop scenes (Bjorkman and Kragic, 2010), or that perform scene differencing (Herbst et al., 2011) usually compute better candidates than generic objectness segmentation algorithms. These "specialized" candidate generators all have one thing in common: they impose restrictions on the environment to simplify the task and improve performance, at the cost of limited applicability in alternative types of scenes. In our framework, we can include multiple candidate generators and use them when their assumptions are met, and revert to more generic candidate generators otherwise.

In HerbDisc, we combine our generic objectness segmentation algorithm (Structure Discovery) from Chapter 6 with the assumption that objects have surfaces of support in floors, tables and walls. The constraint $\Theta_{\text{support}} = (\Theta^{\text{n}}_{\text{support}}, 1)$ is defined as

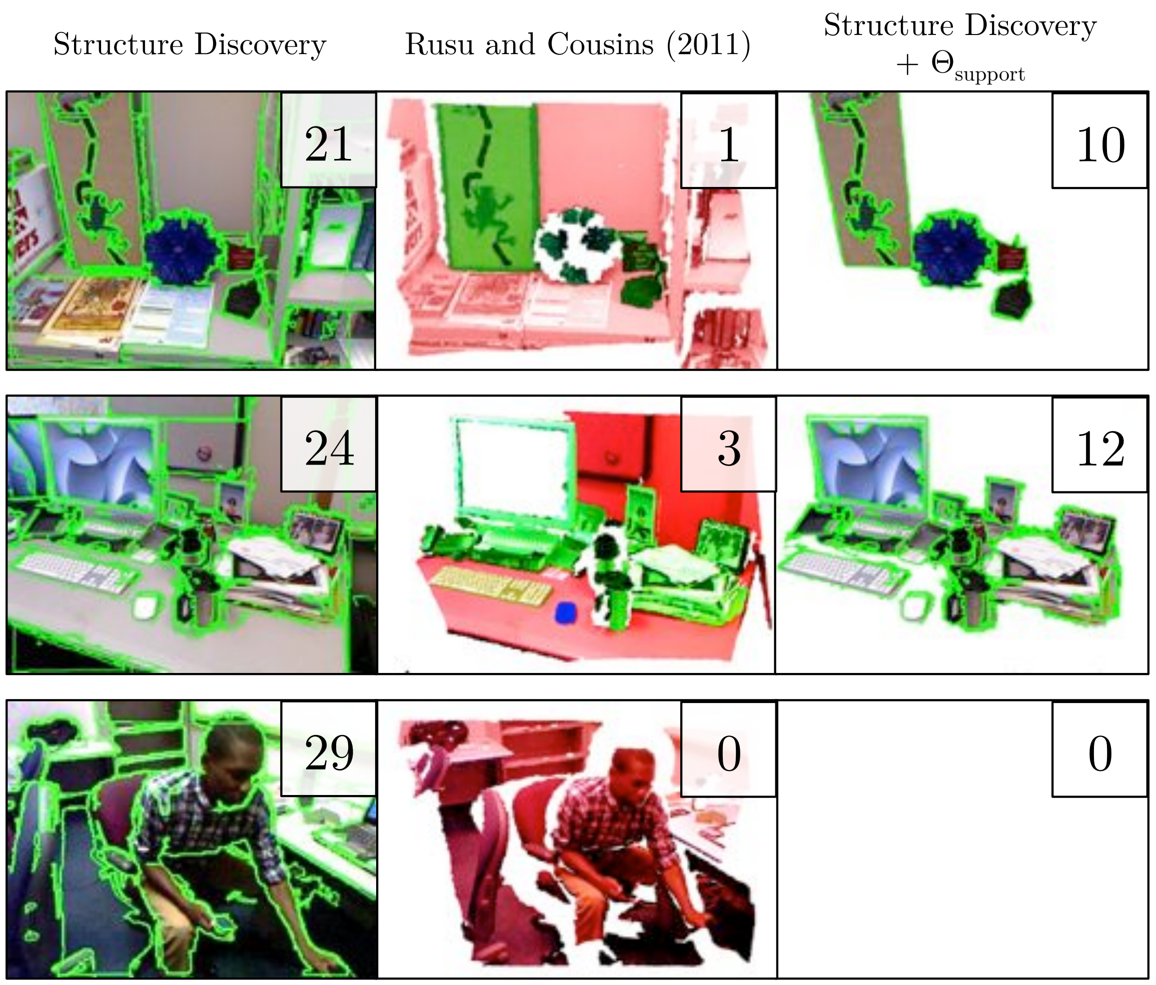


Figure 7.4: Examples of Constrained Candidate Generation in the NSH-1 Dataset (figure best viewed in color). The number of candidates in each data sample is shown at the top right corner of each image. (left) Objectness-based segmentation (Structure Discovery, Chapter 6). (center) Rejected areas according to Θ_{support} are shown in red; the connected components of accepted 3D points are shown in green/yellow/blue. In cluttered scenes, multiple objects are sometimes grouped together. Scenes with no visible support are rejected (e.g., row 3). (right) Combining Structure Discovery and Θ_{support} limits the number of candidates but does not result in undersegmentation.

$$\Theta^{\text{n}}_{\text{support}}(h_i) = \begin{cases} 1, \text{with } p = 1 & \text{if supported}(h_i, I_j) \\ 0, \text{with } q = 1 & \text{otherwise,} \end{cases} \tag{7.15}$$

where $q = 1 - p$ is the probability of failure of $\Theta^{\text{n}}_{\text{support}}$, the supported$(\cdot)$ function searches for large planes in the data sample I_j that generated candidate h_i, and accepts h_i if it lies within a certain distance above the planes found. In simple scenes, Θ_{support} can be used as a standalone candidate generator, by clustering the point clouds above the detected planes

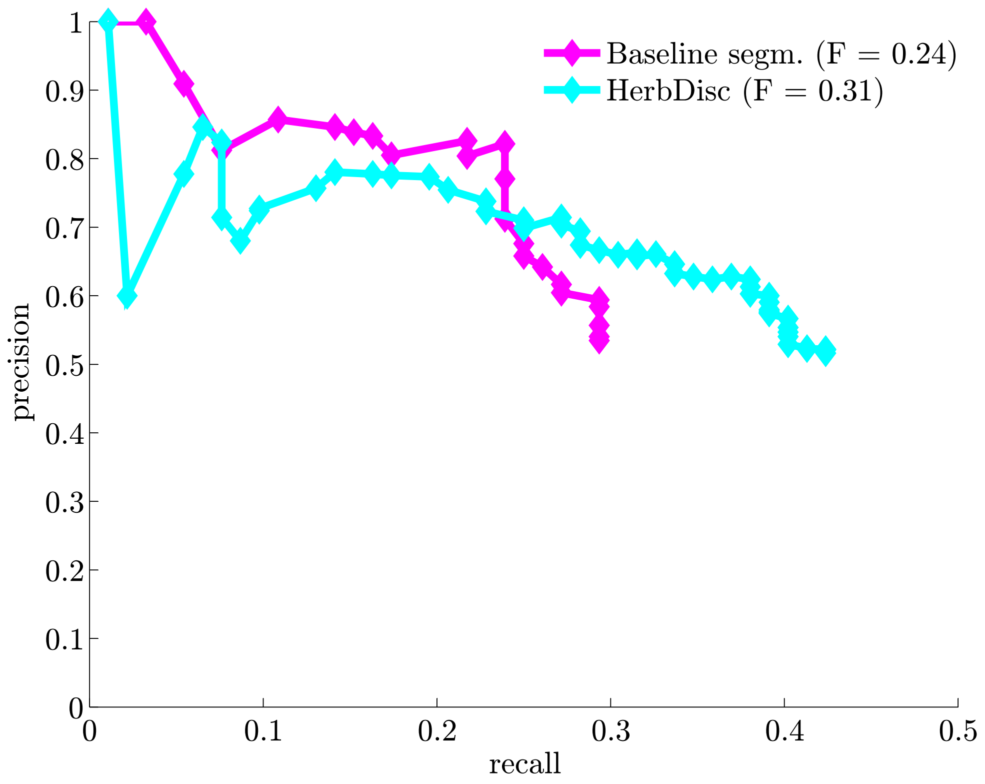


Figure 7.5: Impact of Θ_{support} (Rusu and Cousins (2011), Baseline Segm.) vs. HerbDisc's Structure Discovery + Θ_{support} in the NSH-1 Dataset. Rusu and Cousins (2011) achieves higher precision (80% precision at 20% recall, compared to 78% precision of HerbDisc) at the cost of 14% lower maximum recall.

into a few connected components. For the standalone Θ_{support}, we use the implementation of Rusu and Cousins (2011).

In Fig. 7.5, we compare the performance of Rusu and Cousins (2011) and Structure Discovery with Θ_{support} used in HerbDisc. We see in Fig. 7.5 that the standalone Θ_{support} achieves better precision as it accurately segments simple scenes better. However, the performance degrades in complex scenes (see Fig. 7.4 for examples), as the connected components may include large groups of objects. Combining Structure Discovery with Θ_{support} yields a good trade-off between generating enough candidates for complex scenes, and filtering unlikely candidates for efficiency. In Section 7.4, we show examples of our Structure Discovery applied to office scenes, and compare it with the standalone candidate generator from Rusu and Cousins (2011) and the combination of Structure Discovery and Θ_{support}.

Time (min)	$\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$	Θ_{motion}	Raw data
58.0	0.7M	29.1M	2.9B
102.7	1.2M	83.9M	10.4B
186.9	2.5M	263M	30.4B
262.2	3.3M	517M	59.9B
319.9	4.0M	803M	89.2B
380.6	4.9M	1.2B	126.0B

Table 7.2: Effect of motion and sequencing in computational cost, for the NSH Dataset. Number of edges to evaluate if using 1) the motion and sequencing constraints, 2) only the motion constraint, and 3) the raw data stream.

7.5.2 Motion and Sequencing

In LROD, we receive a never-ending data stream of information from the robot sensing. We assume that the data stream is:

1. an ordered sequence of data samples, and
2. recorded at a frame rate high enough so that there is spatial overlap between data samples.

During the data acquisition, the motion of HERB influences the amount of spatial overlap between data samples. In particular, HERB may a) not be in motion and acquiring repeated data samples, b) be in motion and fulfilling assumption 2, or c) be in motion and violating assumption 2 (i.e., moving too fast). We address these issues with constraints Θ_{motion} and Θ_{seq}.

In particular, we sample the input data stream at a dynamic framerate depending on HERB's motion, and split the subsampled data stream into small subsets that we term *sequences*. Using Θ_{motion} and Θ_{seq}, we do not process repeated samples, and we do not consider any edges between data samples that violate assumption 2. We enforce a maximum sequence length m to limit the order $|V^{\Theta_{\text{seq}}}|$ of any connected component in the CSG.

Let $T_{k,k-1} \in \mathbb{R}^{4\times4}$ be the transformation between sample I_k and the previous sample in the data stream I_{k-1}, and $M : T \mapsto \mathbb{R}$ the magnitude of the motion T. We model the motion constraint $\Theta_{\text{motion}} = (\Theta^{\text{n}}_{\text{motion}}, 1)$ for $h_i \in I_k$, as

$$\Theta^{\text{n}}_{\text{motion}}(h_i) = \begin{cases} 1, \text{with } p = 1 & \text{if } M(T_{k,k-1}) > \gamma_{\min} \\ 0, \text{with } q = 1 & \text{otherwise.} \end{cases} \tag{7.16}$$

Θ_{motion} only samples the data stream when there is enough motion $\gamma_{\min}$ between data samples.

The sequencing constraint $\Theta_{\text{seq}} = (1, \Theta^{\text{e}}_{\text{seq}})$, where

$$\Theta^{\text{e}}_{\text{seq}}(h_i, h_j) = \begin{cases} 1, \text{with } p = 1 & \text{if } \operatorname{seq}(h_i) = \operatorname{seq}(h_j) \\ 0, \text{with } q = 1 & \text{otherwise,} \end{cases} \tag{7.17}$$

limits the potential edges to candidates $h_i \in I_k$, $h_j \in I_l$ which belong to the same sequence. We compute $\operatorname{seq}(\cdot)$ during the data acquisition. For data sample I_k, the sequence identifier

$$\operatorname{seq}(I_k) = \begin{cases} \operatorname{seq}(I_{k-1}) + 1 & \text{if } M(T_{n,n-1}) > \gamma_{\max} \\ \operatorname{seq}(I_{k-1}) & \text{otherwise} \end{cases} \tag{7.18}$$

is incremented if there is too much motion ($\gamma_{\max}$) between the current sample I_k and I_{k-1} (or if we reach the maximum sequence length m).

We use $M(T) = \|T\|_F$ as an estimate of the relative motion T. $\gamma_{\min}$ and $\gamma_{\max}$ are calibrated so that we capture m data samples in 20 seconds moving in a straight line at HERB's slowest and fastest speed. In practice, sequences are finished because $\gamma_{\max}$ is exceeded in approximately 73% of the sequences (often due to sharp turns), and reaching our limit of $m = 50$ in 27% of the cases, mainly in long, straight corridors. HerbDisc is not very sensitive to particular choices of the maximum sequence length; halving or doubling the maximum sequence length ($m = 25$ and $m = 100$, respectively) yields a decrease of less than 3% in maximum recall with respect to our default choice of $m = 50$.

Table 7.2 shows the effect of using the motion and sequencing constraints Θ_{seq} to computational complexity for the NSH dataset. We calculate the total number of potential edges remaining in the CSG, which is a measure of the computational cost, in the cases of 1) Using $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ to generate connected components; 2) Only using Θ_{motion} to downsample the input data stream; and 3) the raw data stream. Our implementation in HerbDisc, which uses $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ as the initial constraint (using Algorithm 2), yields equivalent computational cost after processing 6 h 20 min as Θ_{motion} after approximately 18 min, or as the raw data stream after 2 min 24 s. Fig. 7.6 compares the trend in computational cost with respect to the data stream length. While using Θ_{motion} is two orders of magnitude more efficient than the raw data stream, it still yields a squared cost with respect to the data stream length, compared to the linear cost of $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$.

For the actual implementation, we considered two alternatives: 1) to track the robot motion using the robot's odometry, or 2) to track the camera motion using real-time techniques such as Kinect Fusion (Izadi et al., 2011). We decided to implement the Kinect Fusion approach, because the odometry does not track the camera tilt and shake while HERB drives. These artifacts can be pretty significant depending on the surface (e.g., camera shake on floor tiles, and tilt on thick carpeting). To implement this motion filter, we modify the Kinect Fusion 6DoF tracker available in PCL (Rusu and Cousins, 2011). Our implementation of Θ_{motion} and Θ_{seq} in HerbDisc, including the PCL Kinect Fusion tracking,

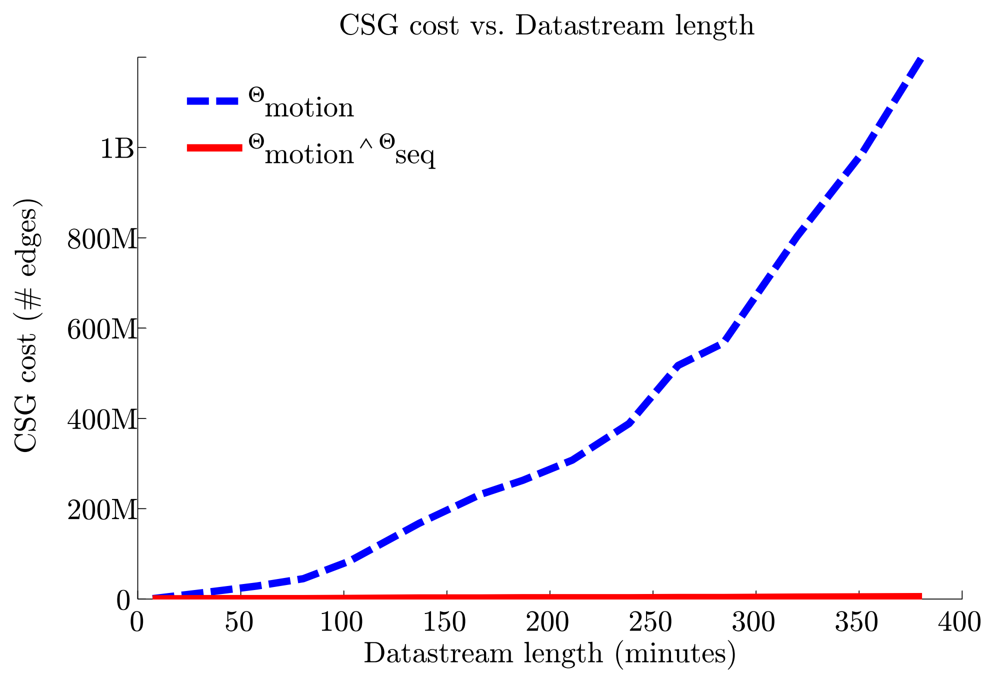


Figure 7.6: Comparing the computational cost of HerbDisc when using $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ and Θ_{motion} for the NSH Dataset, with respect to the data stream length. Using $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ results in linear cost in the number of samples, compared to the squared cost of Θ_{motion}.

runs in real time (up to 30 fps) during the data acquisition process to compute the initial CSG $G^{\Theta_0} = G^{\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}}$ from Algorithm 2.

7.5.3 Spatial Overlap

Many objects in human environments are only moved occasionally, and remain static across most observations. The constraint Θ_{static} encodes our assumption that objects remain static for at least a few seconds at a time. To encode this assumption in our framework, we consider the question "do candidates h_i and h_j overlap in space at a within a sequence?" Relationships between candidates that do not overlap in space should not be considered any further, as they most likely belong to different objects.

The constraint $\Theta_{\text{static}} = (1, \Theta^{\text{e}}_{\text{static}})$, where

$$\Theta^{\text{e}}_{\text{static}}(h_i, h_j) = \begin{cases} 1, \text{with } p = s^{\text{overlap}}_{i,j} & \text{if } s^{\text{overlap}}_{i,j} > s^{\text{overlap}}_{\text{min}} \\ 0, \text{with } q = 1 & \text{otherwise,} \end{cases} \tag{7.19}$$

is a soft constraint that measures the amount of 3D overlap $s^{\text{overlap}}_{i,j} = s^{\text{overlap}}(h_i, h_j)$ between candidates h_i, h_j, and returns true with probability $s^{\text{overlap}}_{i,j}$ if the overlap is above a threshold $s^{\text{overlap}}_{\text{min}}$.

This constraint is designed to operate in unison with the sequencing constraint Θ_{seq}. Θ_{seq} splits the data stream into small subsets of samples close in time (sequences), and Θ_{static} ensures that, within the same sequence, we only evaluate groups of candidates in a similar position in space.

To measure the overlap between hypotheses, we use the incremental registration provided by PCL Kinect Fusion to register all data samples within a sequence with respect to some common coordinate frame T^s (the first sample in that sequence). We transform all object candidates $\boldsymbol{h}$ to the common coordinate frame, and measure the 3D overlap $s_{i,j}^{\text{overlap}}$ between 3D point clouds $h_i^{\boldsymbol{P}}, h_j^{\boldsymbol{P}}$ by comparing their voxel grids.

In Fig. 7.7, we compare the impact of using the 3D overlap constraint Θ_{static} in HerbDisc. We see that Θ_{static} is a crucial metadata constraint in HerbDisc, as disabling Θ_{static} yields a maximum recall of 8% at 47% precision in the NSH-1 Dataset, a difference of 27% recall at the same precision when enabled. Furthermore, disabling the visual similarity features and using only Θ_{static} as an edge constraint results in a drop of only 7% recall and 12% in precision (at maximum recall). These results reinforce our claim that visual features alone are not descriptive enough for large-scale datasets, and that metadata plays a key role in LROD.

7.5.4 Size/shape priors

Part of the reason why there is no clear definition of *object* is because its meaning is subjective: it depends on the observer. In service robotics, different robots might have different definitions of objects depending on their capabilities. For HerbDisc, we consider a definition of object based on the manipulation capabilities of HERB. In particular, we define a prior based on the sizes and shapes of known objects that HERB can grasp.

In order to build an object prior for our framework, we define it as a constraint $\Theta_{\text{prior}} = \Theta_{\text{size}} \wedge \Theta_{\text{shape}}$ composed of size and shape components. Let $\Theta_{\text{size}} = (\Theta^{\text{n}}_{\text{size}}, 1)$ be a constraint on an object candidate's size, such that

$$\Theta^{\text{n}}_{\text{size}}(h_i) = \begin{cases} 1, \text{with } p = s_i^{\text{size}} & \text{if } s_i^{\text{size}} > s_{\text{min}}^{\text{size}} \\ 0, \text{with } q = 1 & \text{otherwise.} \end{cases} \tag{7.20}$$

The function $s_i^{\text{size}} = s^{\text{size}}(h_i, h_{\text{prior}})$ estimates the likelihood that the longest dimension of h_i could be sampled from a Gaussian distribution centered at the size given by h_{prior}.

Analogously, $\Theta_{\text{shape}} = (\Theta^{\text{n}}_{\text{shape}}, 1)$ is a constraint on the candidate's shape, such that

$$\Theta^{\text{n}}_{\text{shape}}(h_i) = \begin{cases} 1, \text{with } p = s_i^{\text{shape}} & \text{if } s_i^{\text{shape}} > s_{\text{min}}^{\text{shape}} \\ 0, \text{with } q = 1 & \text{otherwise.} \end{cases} \tag{7.21}$$

The measure $s_i^{\text{shape}} = s^{\text{shape}}(h_i, h_{\text{prior}})$ estimates the similarity between h_i and h_{prior} according to the PCA-based shape features of Lalonde et al. (2006) (linearity, planarity, and

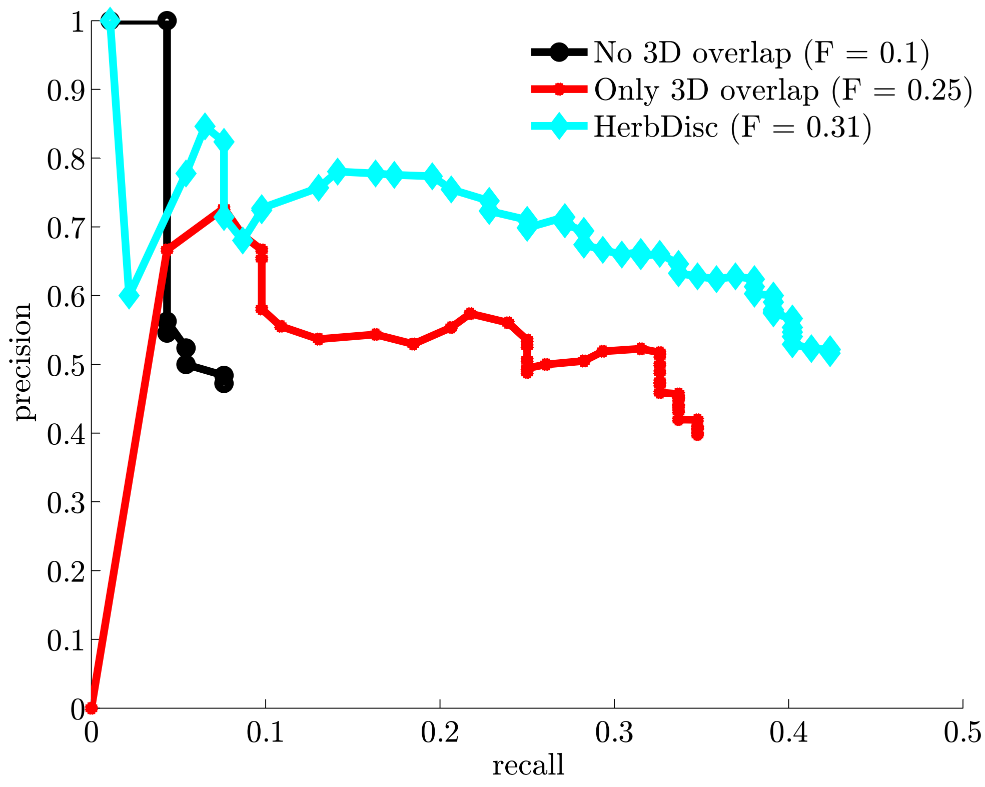


Figure 7.7: Impact of Θ_{static} in HerbDisc for the NSH-1 Dataset. Not using the 3D overlap similarity of Θ_{static} yields a 27% drop in recall compared to HerbDisc. Comparatively, using the 3D overlap similarity Θ_{static} alone with no visual features in HerbDisc only results in a decrease of 7% recall and 12% precision at maximum recall with respect to HerbDisc.

scatterness). The effect of this constraint is to essentially require that object candidates have some volume and are not purely planes or lines.

In Fig. 7.8, we evaluate the impact of size and shape priors in HerbDisc for the NSH-1 Dataset. The main effect of Θ_{prior} is to limit the amount of candidates to cluster, with Θ_{prior} rejecting 63% of the original pool of candidates. The increased number of candidates when Θ_{prior} is disabled yields a 301% increase in the number of objects discovered, most of which are just repetitions due to cluster fragmentation. The final output without Θ_{prior} yields a decrease of 7% recall and 10% in precision (at maximum recall), compared to HerbDisc.

7.5.5 Visual and 3D shape similarity

We describe and compare candidates with features based on 3D shape and appearance. Using these features alone to compute a CSG would result in a pairwise similarity graph as in Kang et al. (2011). For appearance features, we compute the color histogram of each candidate in LAB color space, as in Hoiem et al. (2007), and compare a pair of candidates

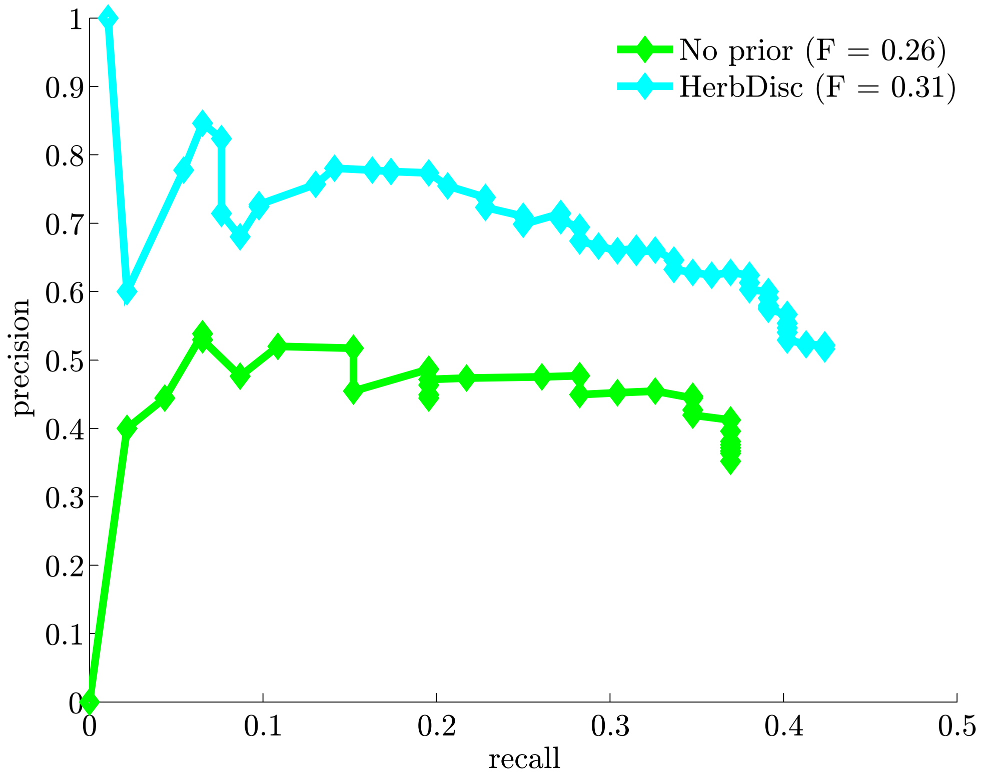


Figure 7.8: Impact of Θ_{prior} in HerbDisc for the NSH-1 Dataset. Not using Θ_{prior} yields a decrease of 7% recall and 10% in precision (at maximum recall).

h_i, h_j with the χ^2 distance between normalized color histograms. For 3D shape, we use the FPFH features of Rusu et al. (2009a), which compute a histogram of the local geometry around each 3D point. We compare the FPFH features of a pair of candidates h_i, h_j by estimating the average χ^2 distance among the nearest neighbor 3D points between h_i, h_j. Both similarity metrics $s^{\text{app}}(\cdot,\cdot)$ and $s^{\text{3D}}(\cdot,\cdot)$ are normalized so that $s(\cdot,\cdot) \in [0,1]$.

In order to use these similarities in our framework, we reformulate them as constraints Θ_{app} and Θ_{3D}. In particular, we define $\Theta_{\text{app}} = (1, \Theta^{\text{e}}_{\text{app}})$ as a soft constraint such that

$$\Theta^{\text{e}}_{\text{app}}(h_i, h_j) = \begin{cases} 1, \text{with } p = s^{\text{app}}_{i,j} & \text{if } s^{\text{app}}_{i,j} > s^{\text{app}}_{\text{min}} \\ 0, \text{with } q = 1 & \text{otherwise,} \end{cases} \tag{7.22}$$

where $s^{\text{app}}_{i,j} = s^{\text{app}}(h_i, h_j)$. Analogously, we define $\Theta_{\text{3D}} = (1, \Theta^{\text{e}}_{\text{3D}})$ as a soft constraint such that

$$\Theta^{\text{e}}_{\text{3D}}(h_i, h_j) = \begin{cases} 1, \text{with } p = s^{\text{3D}}_{i,j} & \text{if } s^{\text{3D}}_{i,j} > s^{\text{3D}}_{\text{min}} \\ 0, \text{with } q = 1 & \text{otherwise,} \end{cases} \tag{7.23}$$

where $s^{\text{3D}}_{i,j} = s_{\text{3D}}(h_i, h_j)$.

In Fig. 7.9, we compare the impact of Θ_{app} and Θ_{3D} in HerbDisc. Disabling the 3D shape similarity Θ_{3D} yields a decrease of 7% recall and 15% precision at maximum recall, compared to HerbDisc, as well as a more significant drop in precision at low recalls (e.g., a 28% decrease in precision at 20% recall). The contribution of Θ_{app} is less noticeable: disabling Θ_{app} results in an decrease of 1% in maximum recall at only 3% lower precision, although it is significant at lower recall (e.g., disabling Θ_{app} yields a 19% decrease in precision at 20% recall).

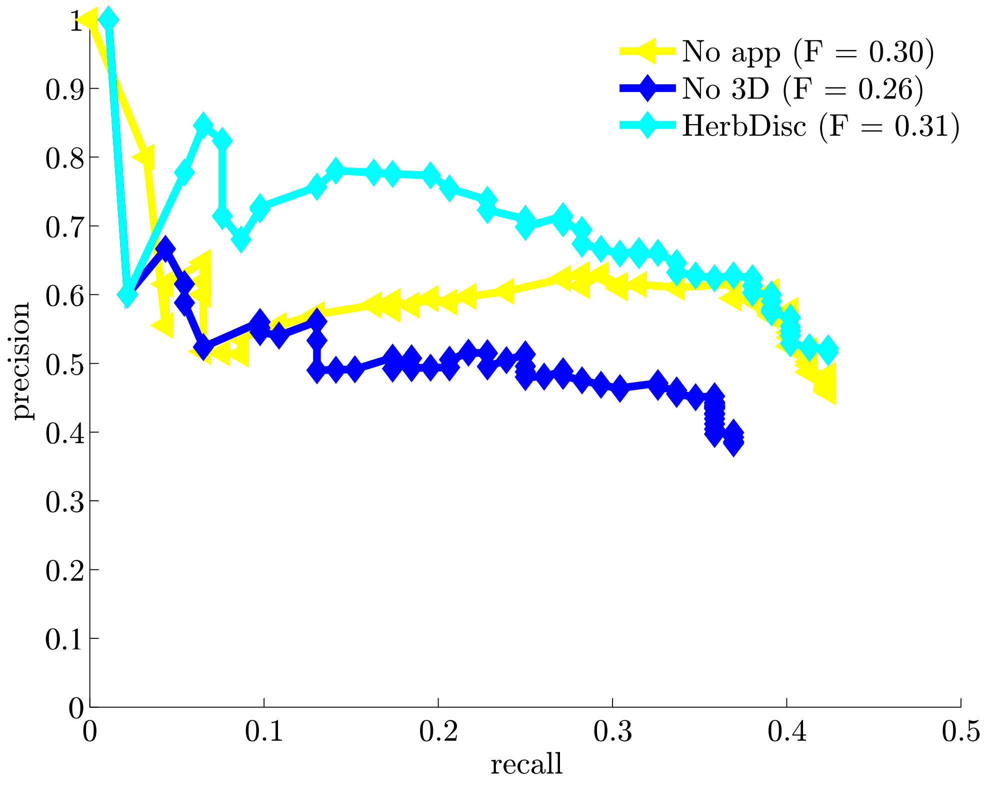


Figure 7.9: Impact of Θ_{3D} and Θ_{app} in HerbDisc for the NSH-1 Dataset. Disabling Θ_{3D} in HerbDisc decreases 7% recall and 15% precision at maximum recall, as well as 20% lower precision at 20% recall. Disabling Θ_{app} yields a decrease of 1% recall and 3% precision at maximum recall, and 19% lower precision at 20% recall.

7.6 Experiments

In this section, we evaluate the impact of using metadata to discover objects. We first compare the performance of HerbDisc with and without any metadata on the Kitchen Dataset in Section 7.6.3. Using metadata, we evaluate the ability of HerbDisc to discover novel objects during a whole workday of operating in challenging human environments. We perform an ablative analysis to assess the impact of each constraint in the constraint expression Θ. Thanks to our framework, performing such an analysis only requires modifying the definition of the constraint expression Θ, but not any change in the source code. This feature is critical for our goal to develop a system that can adapt its behavior as conditions change, using metadata opportunistically.

Our main testbed is the NSH Dataset (Section 7.4), with 6 h 20 min of HERB driving into over 200 offices and engineering labs, and containing 423 annotated ground truth objects. We use the smaller Kitchen Dataset in Section 7.6.3 to evaluate the visual similarity-only baseline, as it is too computationally expensive to execute in the NSH Dataset.

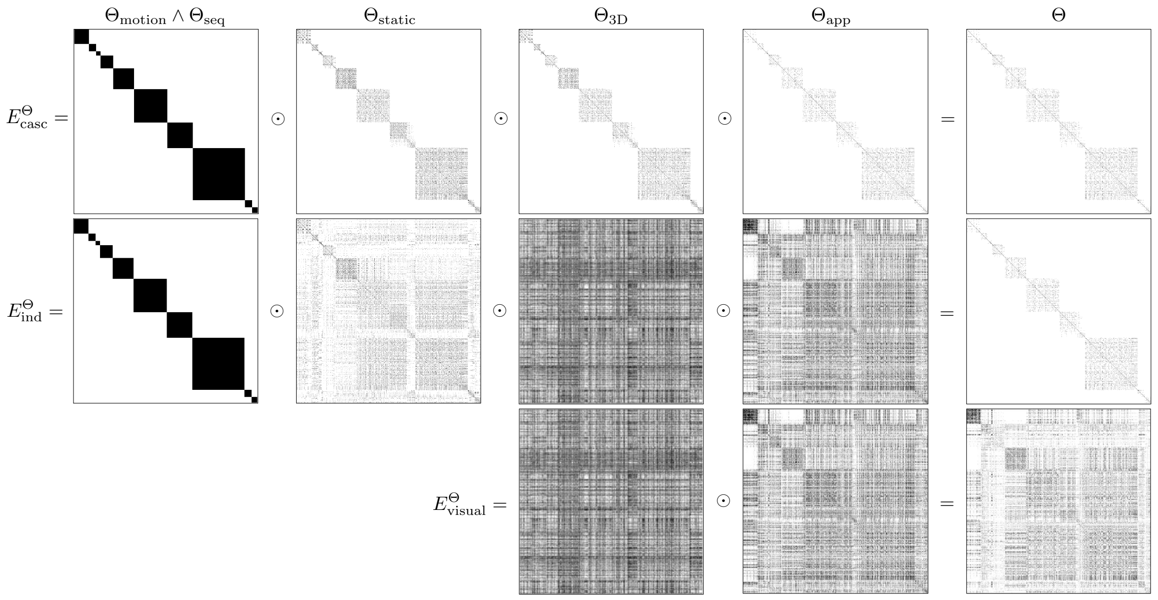


Figure 7.10: CSG graphs for the edge constraints in HerbDisc, displayed as adjacency matrices (where a black dot indicates an edge between candidates), in the Kitchen Dataset. The overall graph E^{Θ} (rightmost column) is the product of each adjacency matrix. (top) Cascaded CSGs using conjunctive constraints, as implemented in HerbDisc. (center) CSGs computed for each constraint independently. The overall CSG E^{Θ} is the same for the cascaded and independent CSGs. (bottom) CSGs for the visual similarity constraints Θ_{app} and Θ_{3D}. The overall CSG E^{Θ} for this case is a regular pairwise similarity graph. The CSGs using metadata (top/center cols) are much more discriminative than the CSG for visual similarity only.

7.6.1 Baseline and training

The baseline for all our experiments is the full system HerbDisc, with all constraints enabled. The default candidate generator is the Structure Discovery from Chapter 6 with Θ_{support}. In each experiment, we enable/disable individual components (through the constraint expression) and analyze the resulting behavior.

The constraint expression Θ_{local} we use in the CSG construction step of HerbDisc is

$$\begin{aligned}\Theta_{\text{local}} =& \Theta_{\text{motion}} \wedge \Theta_{\text{seq}} \quad \wedge \Theta_{\text{support}} \wedge \Theta_{\text{static}} \wedge \\ & \Theta_{\text{size}} \quad \wedge \Theta_{\text{shape}} \wedge \Theta_{\text{app}} \quad \wedge \Theta_{\text{3D}}.\end{aligned} \tag{7.24}$$

In the Object CSG Clustering, we cluster the CSG built with

$$\Theta_{\text{global}} = \Theta_{\text{size}} \wedge \Theta_{\text{shape}} \wedge \Theta_{\text{app}} \wedge \Theta_{\text{3D}}. \tag{7.25}$$

We design the constraints Θ_{app} and Θ_{3D} to be more exhaustive for Θ_{global} than Θ_{local}. In Θ_{local}, we compute the histograms in Θ_{app} with 6 bins per channel, and compute the FPFH features of Θ_{3D} only for the centers of a 1 cm voxel grid. In Θ_{global}, the partial objects contain significantly more information than individual hypotheses. We use 10 bins for the histograms in Θ_{app} and compute FPFH features for Θ_{3D} for the centers of a 3 mm voxel grid. In our experience, the choice of Θ_{local} has significantly more impact in the overall performance than Θ_{global} for Object CSG Clustering. We therefore focus our experiments on the local step and modify *only* Θ_{local}, while we keep Θ_{global} constant throughout the experiments.

We use the first 20% of the NSH-1 Dataset (not included in the evaluation) to train the parameters and thresholds in HerbDisc, by maximizing the average F_1-measure (defined in Section 7.6.2). To do so, we discretize each parameter in 5 settings in the range $[0, 1]$ and choose the best-performer configuration according to a grid search. We do not modify any parameter in any experiment after the initial training phase. All experiments were performed on a computer with an Intel Core i7-920 CPU, 16GB of RAM, a nVidia GTX 580 GPU, and runninng 64-bit Ubuntu Linux 10.04.

7.6.2 Evaluation Procedure

In this section, we describe the metrics to evaluate the ability of HerbDisc to discover objects during HERB's workday. For a given object model M_k, we define the metrics of *Candidate purity*, *Group purity*, and *3D purity*, as:

Candidate purity. We describe an object candidate h_i as *pure* if over 80% of the area in $h_{i,k}$ overlaps with a ground truth object.

Group purity. Following Tuytelaars et al. (2009), we measure the group purity of model M as the largest percentage of *pure* object candidates in $M_k^{\boldsymbol{h}} = \{h_{1,k}, \ldots, h_{i,k}\}$ that belong to the same object.

3D purity. We require that the 3D models reconstruct the partial viewpoints seen by HERB. Therefore, we define an object's 3D point cloud $M_k^{\boldsymbol{P}}$ as *pure* if the 3D points in $M_k^{\boldsymbol{P}}$ cover over 80% of the area visible in the data samples for that particular object.

Given the open and unsupervised nature of LROD, we often discover objects that do not appear in the ground truth set, despite being real objects. Following Kang et al. (2011), we distinguish between three categories of objects: *correct*, *valid*, and *invalid*.

We define an object model M_k as ***correct*** if 1) it is an object annotated in the ground truth, 2) its 3D point cloud is pure, and 3) every object candidate $h_{i,k}$ associated to M_k is pure, i.e., if the set $M_k^{\boldsymbol{h}}$ is 100% pure. Other works in the literature commonly define correct objects as clusters with some minimum percentage of purity (e.g., 80% in Kang et al. (2011)), but we believe that object models need to be 100% correct to be of any use for robotics. Fig. 7.14 shows multiple examples of *correct* objects.

We define an object model M_k as ***valid*** if 1) its 3D point cloud is pure, 2) the set of candidates $M_k^{\boldsymbol{h}}$ is 100% pure (as with *correct* objects), but 3) it has not been labeled as a ground truth object. We rely on an "oracle" evaluation, as in Tuytelaars et al. (2009). The "oracle" evaluation is a human annotator who answers the questions "Is M_k an object?", and "Does M_k have a name?" when faced with the set of candidates for object model M_k. The category *valid* mainly contains objects too big or too small to be grasped (e.g., chairs), immovable objects (e.g., attached to a wall), or parts of complex objects (which could be objects themselves, such as a bicycle's seat). Fig. 7.15(top) shows multiple examples of *valid* objects.

We define an object model M_k as ***invalid*** if it is neither *correct* nor *valid*. The category *invalid* mainly includes models M_k of groups of objects erroneously segmented, of a single object but $<$ 100% group purity, or a mix of multiple objects erroneously clustered together. Fig. 7.15(bottom) shows multiple examples of *invalid* objects.

We define Precision and Recall as in Tuytelaars et al. (2009) and Kang et al. (2011). In Kang et al. (2011), Precision is the ratio between the number of *correct+valid* object models and the total number of discovered models:

$$\text{Precision} = \frac{\#\text{correct} + \#\text{valid}}{\#\text{correct} + \#\text{valid} + \#\text{invalid}} \tag{7.26}$$

We measure Recall as the ratio between the number of unique *correct* objects and the total number of ground truth objects.

$$\text{Recall} = \frac{\#\text{unique correct obj.}}{\#\text{unique ground truth obj.}} \tag{7.27}$$

We use the cluster size to estimate the quality of an object, and use it as the variable to threshold to compute the P-R curves. To summarize the P-R curves in a single number, we use the average F_1-measure, which balances Precision and Recall for each sample i in the P-R curve:

$$F_1 = \frac{1}{N}\sum_i^N \frac{2\text{Precision}_i\text{Recall}_i}{\text{Precision}_i + \text{Recall}_i} \tag{7.28}$$

7.6.3 Results

In this section, we evaluate the impact of metadata to discover objects. We evaluate the use of metadata compared to using visual similarity alone in Section 7.6.3, and then we show

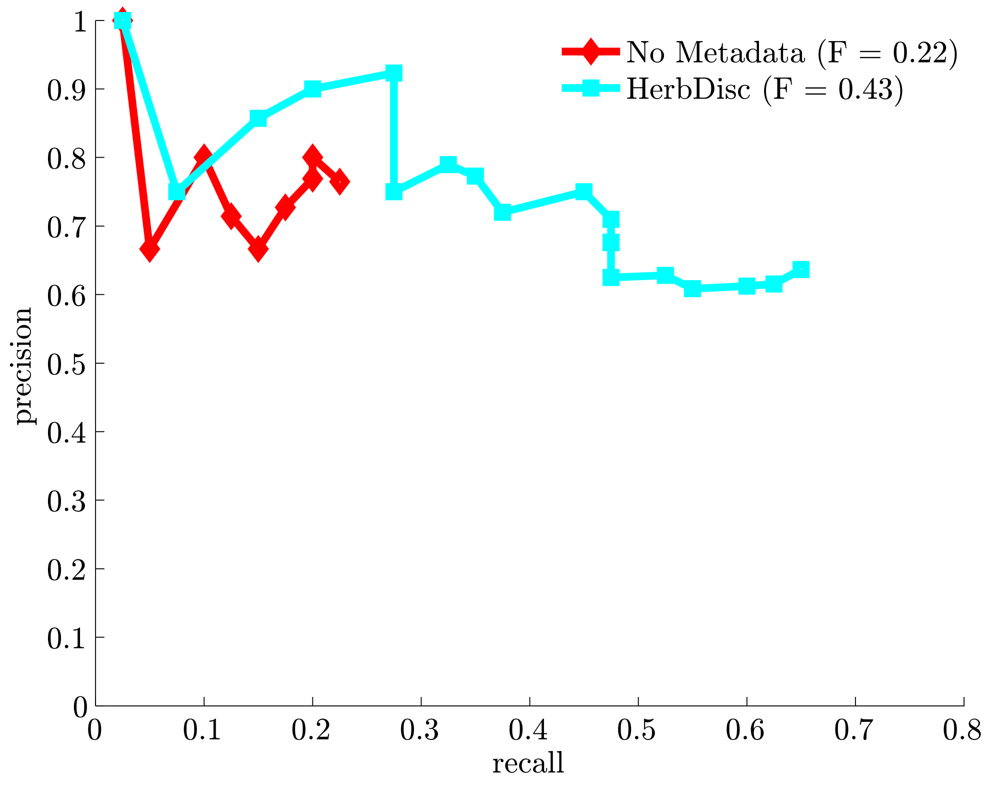


Figure 7.11: Impact of using metadata in HerbDisc for the Kitchen Dataset, compared to visual and 3D similarity alone. HerbDisc achieves with a maximum recall of 65% at 62% precision, compared to 24% maximum recall at 77% precision. For the same recall of 24%, HerbDisc achieves 90% precision (13% higher than the visual similarity Θ_{visual} alone).

Component	HerbDisc	Θ_{visual}
CSG Construction	35.9 s	18981.8 s
CSG Clustering	61.3 s	394.0 s
Object CSG Clustering	4.4 s	2.8 s
Total processing time	101.6 s	19378.6 s

Table 7.3: Running times of HerbDisc vs. Θ_{visual} in the Kitchen Dataset. Using no metadata (Θ_{visual}) is 190× slower than using metadata in this dataset, mainly due to the extra cost of constructing the graph. The Θ_{visual} needs to evaluate $1.6M$ pairwise visual similarities from 1806 object candidates, compared to the 16271 pairwise visual similarities to evaluate when using metadata in HerbDisc.

that we can leverage metadata to process very large datasets such as the NSH Dataset in Section 7.6.3.

HerbDisc vs. Visual Similarity

Fig. 7.11 shows the performance of using a CSG with visual similarity only ($\Theta_{\text{visual}} = \Theta_{\text{motion}}\Theta_{\text{3D}} \wedge \Theta_{\text{app}}$), compared to the full HerbDisc, in the Kitchen Dataset. We include the motion filter Θ_{motion} in the evaluation of Θ_{visual} so that both systems have the same initial pool of object candidates.

HerbDisc is the clear winner in the Kitchen Dataset, with a maximum recall of 65% at 62% precision, compared to 24% maximum recall at 77% precision. For the same recall of 24%, HerbDisc achieves 90% precision (13% higher than the visual similarity Θ_{visual} alone). The additional constraints provided by the metadata (and especially Θ_{seq}) allow HerbDisc to process the Kitchen Dataset 190× faster than if using visual similarity alone, as shown in Table 7.3. The main reason for this speedup is the limited number of pairwise similarities to evaluate in the CSG (mainly due to Θ_{seq}) compared to the regular pairwise similarity graph from Θ_{visual}. Namely, HerbDisc evaluates 16271 pairwise visual similarities, compared to 1.6M in Θ_{visual}.

To illustrate the impact of different constraints on the CSG, we show in Fig. 7.10 the graphs (displayed as adjacency matrices) generated by each edge constraint $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$, Θ_{static}, Θ_{3D}, and Θ_{app}, for the Kitchen Dataset. Fig. 7.10(top) displays the CSG after each constraint as evaluated in HerbDisc, cascading the multiple conjunctive constraints for efficiency. Fig. 7.10(middle) shows the CSG for each constraint independently. The product of all adjacency matrices (rightmost column) is the same for both approaches, but HerbDisc is more efficient. The metadata-based constraints Θ_{seq}, Θ_{static} are significantly more discriminative than the visual features Θ_{3D} and Θ_{app}. The adjacency matrix for $\Theta_{\text{motion}} \wedge \Theta_{\text{seq}}$ also illustrates the behavior of the dynamic motion filter, generating sequences of different length (i.e., squares of different size) depending on HERB's motion. Fig. 7.10(bottom) shows the result of using visual similarity constraints with no metadata. In this case, the product of all adjacency matrices (rightmost column) is significantly denser than in HerbDisc, which accounts for the increased computation time shown in Table 7.3.

HerbDisc in the NSH Dataset

In Section 7.5, we explored the impact of each individual component of HerbDisc. We provide a summary plot in Fig. 7.13 that combines the P-R curves of all components.

The attempt to evaluate Θ_{visual} on the NSH Dataset was unsuccessful, after the testing machine had made barely any progress after a week of processing. HerbDisc processes the NSH Dataset in 18 min 34 s. We show an itemized list of running times for the different steps

Component	time (s)
Data acquisition	22836
Read sequence/candidate data Θ_{seq}	25.9
CSG Construction	710.1
CSG Clustering	211.9
Object CSG Clustering	54.6
Model Registration	111.4
Total processing time	1113.9

Table 7.4: Running times of HerbDisc in the NSH Dataset. The motion and sequencing constraint and the candidate generation are executed in parallel with the data acquisition and are not included. The overall running time is 1113.9 seconds (18 min 34 s), to discover 121 correct and 85 valid objects from an RGBD video feed of 6 h 20 min (521234 samples).

Component	Output	Quantity
$\mathcal{S}$	Input samples I	521234
Θ_{motion}	Samples I	19614
Θ_{seq}	Disjoint Sequences I^s	732
$\mathcal{H} \wedge \Theta_{\text{support}}$	Object Candidates h	58682
$\Theta_{\text{size}} \wedge \Theta_{\text{shape}}$	CSG nodes V^Θ	49230
$\Theta_{\text{static}} \wedge \Theta_{\text{3D}} \wedge \Theta_{\text{app}}$	CSG edges E^Θ	431121
CSG Clustering	Partial objects m_k	2215
Object CSG Clustering	Full Objects M_k	464

Table 7.5: Impact of each component and quantities generated for the NSH Dataset, from 521234 input images to 464 output models (with 121 *correct* and 85 *valid* objects).

in Table 7.4, and the statistics for images, candidates, edges, etc., in Table 7.5. The overall running time does not include data acquisition time (and motion filtering and candidate generation, which we execute in parallel with the data acquisition). The most expensive step is the CSG construction, which processes 732 connected components in the CSG, for a total of 49230 nodes and 4.9M edges—with 431121 edges satisfying all constraints—in 11 min 49 s. The CSG Clustering step is the second most expensive step, separating 2215 clusters (i.e., partial objects) in 3 min 31 s. The Object CSG Clustering and model registration are the most expensive per-unit steps. However, they leverage the filtered information from the CSGs to cluster and register 464 objects in 2 min 45 s.

We discover a total of 464 object models in the NSH Dataset, where 121 unique objects are *correct* (28.6% recall) and 85 are *valid* (44.4% precision). In Fig. 7.12, we show the P-R curves for the NSH Dataset, as well as a floor-by-floor analysis (NSH-1 to NSH-4). We

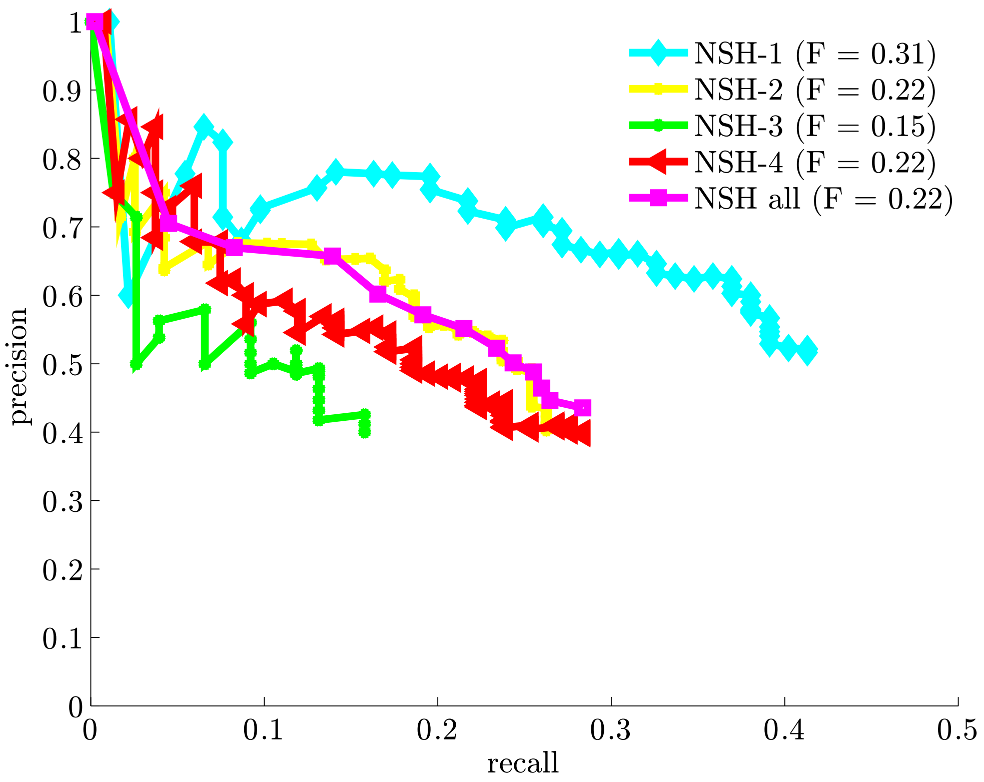


Figure 7.12: Floor-by-floor evaluation of HerbDisc on the NSH Dataset. HerbDisc achieves a 28% higher recall in regular office environments (NSH-1) compared to laboratory and machine shop environments (NSH-3). Mixed environments containing both laboratories and offices (NSH-2 and NSH-4) achieve similar recall. HerbDisc achieves a maximum recall of 28.6% in the overall NSH Dataset at 44.4% precision, compared to 43.9% maximum recall in office environments (NSH-1) and 15% in laboratories and machine shops (NSH-3).

see a clear difference in performance as we move from regular office environments (NSH-1) to laboratories and machine shops (NSH-3). In office environments, HerbDisc displays a maximum recall of 43.9% at 52% precision, and 78% precision at 20% recall. In contrast, we only achieve a maximum recall of 15% at 41% precision in the laboratories of NSH-3 (e.g., Fig. 7.3(2, 3-5)), which include multiple shiny metallic objects, specular reflections, and extreme clutter.

We can also modify the configuration of HerbDisc on the fly to achieve different behaviors. For example, if we are more interested in precision than recall, we can use Θ_{support} as a standalone candidate generator and achieve 82% precision at 25% recall (on NSH-1), or reject the lowest-ranked models and achieve 60% precision at 40% recall. We show examples of *correct* objects in Fig. 7.14 and of *valid* and *invalid* objects in Fig. 7.15.

The *correct* objects discovered by HerbDisc are predominantly objects we would expect in an office environment, such as laptops, books, phones, monitors, keyboards, and mice.

Other objects, such as basketball balls, watering cans, plants, and food items, showcase the object diversity—and therefore difficulty—from the NSH Dataset. We require objects to be 100% pure to be considered *correct*, which assures a high quality for potential robotics applications.

In an open task such as object discovery, it is nearly impossible to obtain comprehensive ground truth. HerbDisc discovers objects that the annotators considered outside the guidelines for ground truth in Section 7.4, such as chairs, trashcans, or wall-mounted paper holders (see Fig. 7.15). The discovery of such objects can be due to several reasons. First, the object priors specified in HerbDisc may not be specific enough, accepting objects that HERB cannot manipulate (e.g., chairs and people). Other objects are not considered correct due to semantic meaning (e.g., the object is a part of a more complex object, such as a bike seat or a chair's armrest), because the object is immovable (e.g., a wall-mounted paper holder), or because the annotators did not notice or recognize the object (e.g., paper folders, cables). We believe that the only way to disambiguate between theses cases is to interact with the objects during the discovery process, which is a future direction. Our framework can be used to leverage interaction information if available, as well as any other source of metadata, when formulated as constraints.

Among the *invalid* objects, we identify three main categories: 1) correct but impure objects; 2) groups of objects; and 3) mixtures of fragments. The first case refers to correctly discovered objects that contain a few (or sometimes only one) misplaced candidates, such as the red cup or the plastic bag in Fig. 7.15. Objects in the second case are usually compound of multiple objects very close to each other or touching each other, such as groups monitor-keyboard-mouse or stapler-stapler-table. The third case comprises unrecognizable groups of object candidates from multiple objects. Invalid objects in cases 1) and 3) are mostly due to clustering errors, which improperly unite candidates from different objects. Objects in case 2) are mostly due to candidate generation/segmentation errors, failing to separate the individual objects in complex scenes.

7.7 Summary

In this chapter we have introduced Lifelong Robotic Object Discovery, the problem of discovering new objects in the environment during an entire robot's lifetime: while the robot operates, for as long as the robot operates. As a first step towards LROD, we have proposed a solution to process the raw video stream of an entire workday of a robotic agent.

We claim that the key to make LROD feasible is to incorporate metadata. We have described a graph-based framework to integrate any source of metadata and similarity functions as graph constraints, and to combine multiple constraints using boolean logic expressions. With these graph constraints, we provide a common formulation to encode any

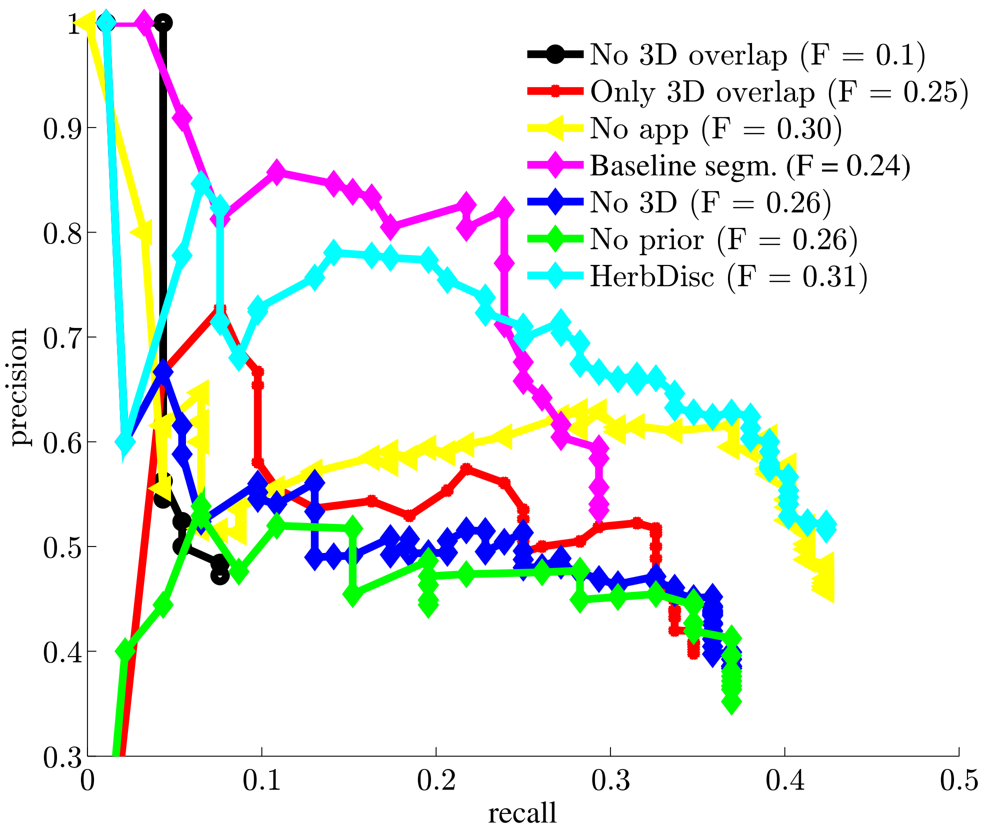


Figure 7.13: Summary of P-R curves for the ablative analysis. HerbDisc is the best-performing method, combining the results of all constraints for improved object discovery in the NSH-1 Dataset.

source of information, both visual data and metadata, as metadata-augmented graphs—Constrained Similarity Graphs (CSGs)—for object discovery.

We have introduced HerbDisc, an optimized implementation of our framework which leverages metadata about the environment, the robotic agent, and the sensors, as well as visual information, to efficiently discover objects in large datasets. To evaluate the performance of HerbDisc, we have gathered a dataset of over half a million RGBD images (6 h 20 min of raw RGBD video) of office and lab environments, ranging from moderately to extremely cluttered, and containing 423 ground truth objects. HerbDisc processed this dataset in under 19 minutes and discovered 206 novel objects, such as monitors, keyboards, plants, and food items, with a maximum recall of 28.6% at 44.4% precision, and 68% precision at 15% recall (and, for regular office environments, maximum recall of 43.9% at 52% precision, and 78% precision at 20% recall). More importantly, we showed that our framework can opportunistically leverage different sources of information adaptively, as conditions change, which is a necessary feature to make LROD feasible in the long term.

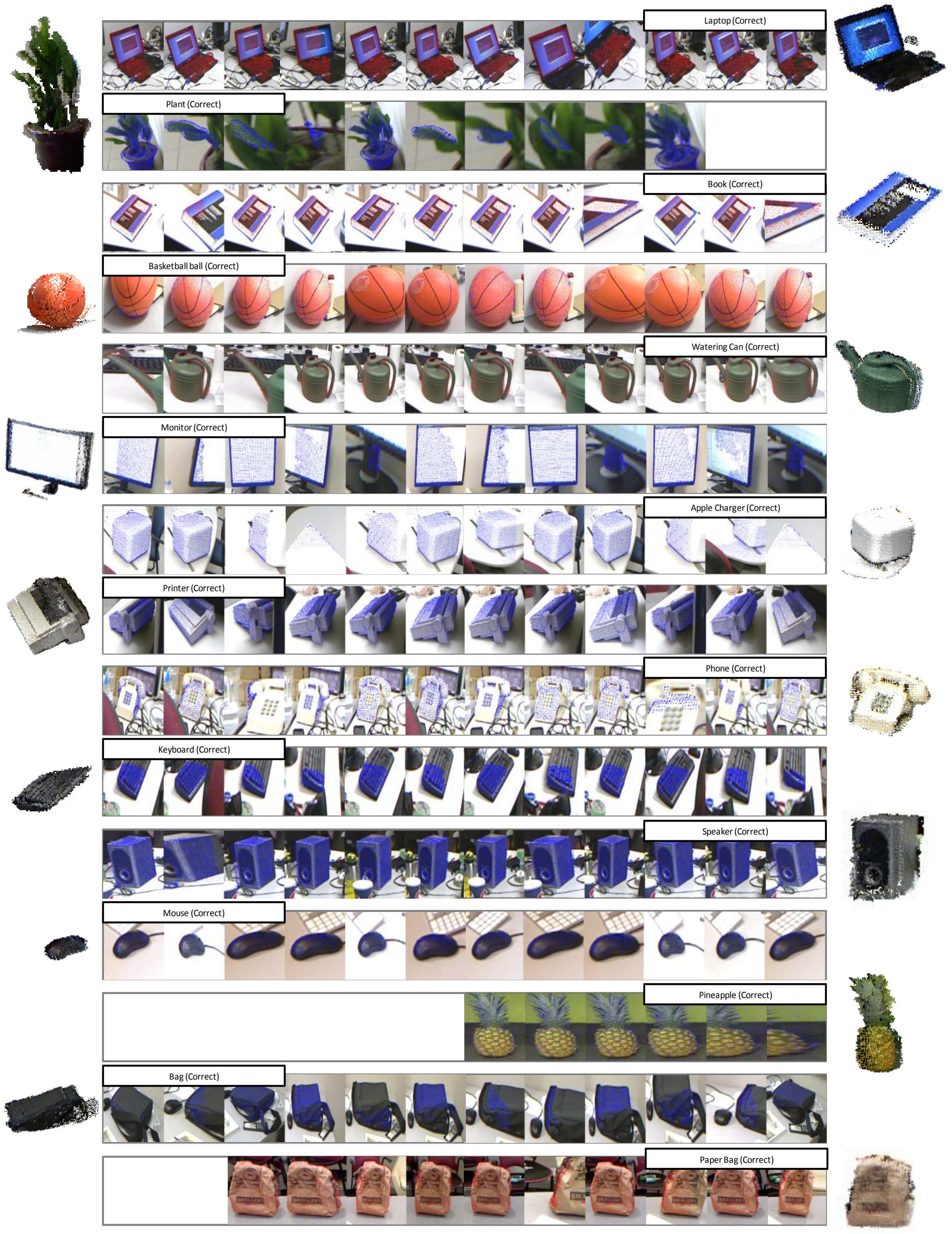


Figure 7.14: Examples of *Correct* objects. For each object, we display its object label (text box); its 3D model (left/right); and 10 randomly selected images from the set of object candidates h_i (center), with the 3D point clouds $h_i^{\boldsymbol{P}}$ overlaid in red or blue over each image.

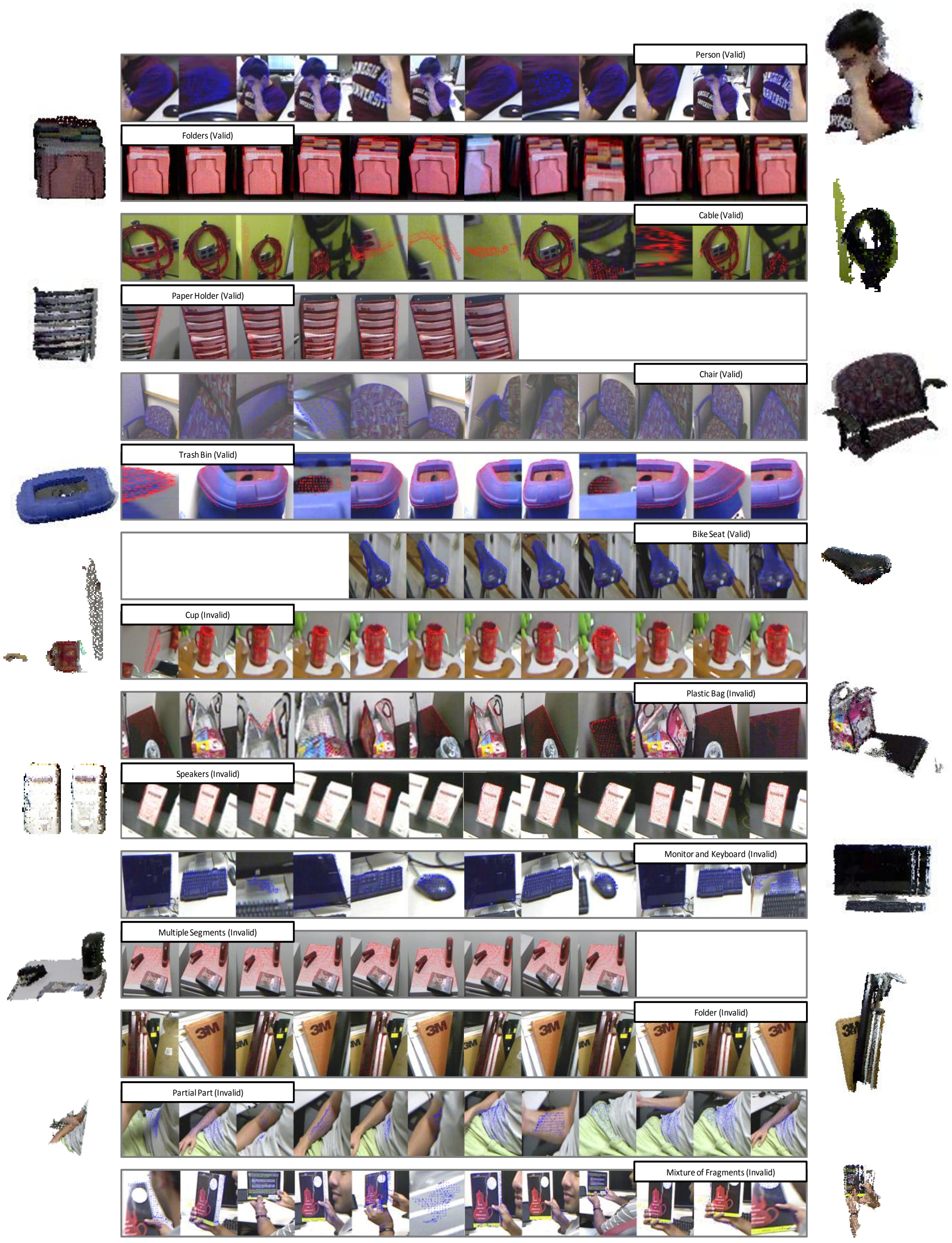


Figure 7.15: Examples of *Valid* and *Invalid* objects. For each object, we display its object label (text box); its 3D model (left/right); and 10 randomly selected images from the set of object candidates h_i (center), with the 3D point clouds $h_i^{\boldsymbol{P}}$ overlaid in red or blue over each image.

And yet, despite discovering hundreds of novel objects, HerbDisc failed to discover over half of the total number of objects. We believe that, in order to truly solve the LROD problem, it will be necessary to transform the robot from an observer to an active agent, interacting with objects and leveraging that information to discover and validate discovered objects. With our framework, we can encode the information coming from interaction as more effective graph constraints, to discover objects resulting from that interaction. A future direction for our research is to develop effective interaction strategies to discover novel objects, to disambiguate when uncertain, and to validate the discovered objects by interacting with them.

Another related future direction is to explore online techniques for object discovery. The framework described here is essentially a batch process, so that it can be processed during the robot's downtime. However, online processing could be performed using the sequences provided by the motion filter. Once the motion filter generates a new sequence—of up to 20 seconds—we can cluster and generate partial objects for that sequence. We would perform an Object CSG Clustering step every few hours, to join the most recent partial objects with the full objects found during previous Object CSG Clusterings.

Part IV

Conclusion

Chapter 8

Contributions

Similarity. [n. sim·i·lar·i·ty. si-mə-ˈla-rə-tē]:
Correspondence in appearance or superficial qualities.
Merriam-Webster Dictionary

In this thesis, we have explored the problem of Lifelong Robotic Object Perception. Our work has focused on the two core components of this problem: Object Recognition, and Robotic Object Discovery. We have presented two systems, MOPED and HerbDisc, to perceive known and unknown objects in challenging human environments, respectively. In this chapter, we summarize our findings in these two areas, and we describe interesting future directions that we need to address to solve the Lifelong Object Perception problem.

Our main contributions are as follows:

- *MOPED:* We address the problems of high scene complexity, scalability, and latency that hamper object recognition systems in service robotics. With Iterative Clustering-Estimation (ICE), we solve the pose estimation and data association problems, and integrate single- and multi-view object recognition. The Projection Clustering method and the Q-Scores detect false positives and reuse that information to accurately estimate the poses of true positives. The multiple architectural improvements in MOPED provide over 30x improvement in latency and throughput, allowing MOPED to perform in real-time robotic applications.

- *MOPED-RGBD:* We derive a data fusion model based on maximum likelihood estimates to integrate RGBD and image-only object recognition. We use depth opportunistically, when it is available, and seamlessly transition to image-only performance in the presence of depth fading. We incorporate our data fusion model to each stage of MOPED to create an object recognition system, MOPED-RGBD, robust to imperfect depth data.

- *Structure Discovery:* We introduce an *objectness*-based scene segmentation algorithm for RGBD images to generate object candidates for Object Discovery. We exploit the availability of RGBD data using a novel region-based approach. We use Structure Discovery to discover objects in human environments, and we yield a 13% increase in recall over specialized 3D object segmentation algorithms.

- *Robotic Object Discovery with Metadata:* We introduce a common formulation to represent visual information (RGBD images) and robotic metadata as constraints. We compute Constrained Similarity Graphs (CSGs) from these constraints. The use of CSGs, coupled with the natural constraints in service robotics, enables efficient object discovery in large-scale datasets. We introduce an optimized implementation, HerbDisc, that processes an entire workday of HERB (6 h 20 min of raw RGBD video) in under 19 min, to discover 206 novel objects. We show that, by using robotic metadata in object discovery, we discover over 2.7$\times$ more correct objects, and we process the data 190$\times$ faster than using visual information alone.

Chapter 9

Future Directions

Manipulate. [v. ma·nip·u·late. ma-ˈni-pyə-vlāt]:
To operate with the hands or by mechanical means especially in a skillful manner.
Merriam-Webster Dictionary

Despite our contributions to Object Recognition and Robotic Object Discovery, we cannot say that the problem of Lifelong Robotic Object Perception is solved. Neither MOPED nor HerbDisc are perfect; both systems make mistakes, and their performance is still not comparable to human perception.

There are multiple areas that should be explored as follow-up works to this thesis, which we briefly describe in the following sections.

9.1 Model improvement through robot interaction

The performance of HerbDisc exemplifies the formidable challenge of Lifelong Robotic Object Perception in human environments. Despite efficiently processing an entire robotic workday and discovering hundreds of novel objects, HerbDisc still failed to discover over half of the total number of objects. Object Discovery in human environments is so challenging that sometimes even our annotators failed to provide accurate ground truth due to extreme scene clutter. We believe that it is necessary for a robot to interact with objects as part of the learning process, if we want to solve this problem. An important future direction is to implement the interaction-perception feedback loop discussed in Section 2.1, to transform the robot from a passive observer to an active agent that can interact with objects. We can leverage the interaction information to validate discovered objects and to improve the segmentation of movable objects.

It is possible to use the recognition output from MOPED and the successful/unsuccessful robot interactions with objects as a feedback loop to validate and improve the quality of object models. We can execute MOPED and HerbDisc simultaneously to associate recognized objects with partial object models. A recognized object associated repeatedly with the same partial object model is a strong indication that the association is correct. We can use this cue to refine the recognition models with the additional data from partial object models.

An object recognized by MOPED and successfully grasped by the robot is an even stronger cue that the recognition was successful, and that the partial object model from the data stream is valid. Such examples should therefore be used to validate and improve the recognized object models. Similarly, repeatedly failing to grasp an object should be a cue that the object model is invalid. Ultimately, we should analyze which information within each model is more prone to result in an unsuccessful grasp, and weight down the relative importance of different features to prevent errors from happening again. Objects that are never successfully recognized or grasped should be forgotten.

9.2 Scalable object representation

Gathering sensor data—and discovering objects—every time the robot is in operation is important to always maintain up-to-date knowledge. However, common everyday objects are seen in thousands of images since their discovery, which can unnecessary burden the object modeling. We want to generate models *for recognition*, not any other task. Unlike model building for accurate 3D reconstruction (e.g., Snavely et al. (2006)), we do *not* need every single image and every single feature available in the input data. Detailed models with hundreds of thousands of features are in general not that useful for recognition, as the overabundance of features makes recognition both slower (there are more features to match) and potentially less precise (increased false positives, due to more confusion between features), which are also critical factors in robotics applications. In addition, the model building stage alone might take several days when thousands of images are present (Snavely et al., 2008).

Considering these constraints, an important follow-up work to this thesis is to develop a strategy to generate compact object models and maximize recognition performance, given a very large number of observations from an object. A proper object representation should consider, among others, the following aspects:

- Efficient processing of potentially large amounts of observations
- Easy to merge multiple partial models of an object into a single structure
- Easy to update a given model with new observations
- Optimize models for recognition

Recent works in Structure from Motion, such as Snavely et al. (2008), Havlena et al. (2010), sample a set of observations using a minimum Connected Dominating Set (CDS) algorithm to speed up model reconstruction. In Snavely et al. (2008), CDS is used to find a suitable subgraph of a pairwise similarity graph. The reconstruction of a CDS subgraph instead of the whole graph results in speedups of over an order of magnitude for large image sets. Given the additional constraints in service robotics, we could consider to compute a CDS for Constrained Similarity Graphs. As we showed in Chapter 7, CSGs can encode not only visual similarity but also other generic information. In particular, information about image sequencing and similar constraints, such as frame decimation (Nister, 2000) or keyframe extraction (Thormahlen et al., 2004), could be very useful to reduce the number of input data samples. Alternatively, feature resampling techniques (e.g., Fang and Quan (2010)) can also be considered to produce more compact object models for recognition.

Another important consideration to develop a scalable object representation is how to feed back information from recognition and robot interaction. A potential solution would be to incorporate the information as constraints in a CSG as well. Given that CSGs encode node and edge weights, we can exploit similar tools to those in Structure from Motion, such as CDS subgraphs for node- and edge-weighted graphs. In particular, Guha and Khuller (1998) developed a Weighted CDS (WCDS) technique to compute CDS subgraphs on graphs with node and edge weights, which we could use for our problem.

To update object models with new partial object models, we can devise a training method that uses cross-validation to maximize the recognition performance given the observed data. Every new observation of an object should be used to improve the object model. A potential solution would be to build several potential object models containing the new observation and a subset of previous observations. We would evaluate the models on multiple validation sets, randomly chosen among the images not used in the model building, and choose the best-performing one for recognition.

9.3 Databases of common objects

A key feature of our framework is its ability to discover objects in a completely unsupervised fashion, based on the output of Structure Discovery and the available robotic metadata. The availability of a large-scale database of common objects poses interesting challenges for a number of areas, both for Object Recognition and for Object Discovery.

In particular, we can assume that every object is similar to some object we already know. A similar approach was shown in the work of Kang et al. (2012). Using this rationale, we can combine our current Object Discovery with an exemplar-based search.

This exemplar-based approach can be integrated into Structure Discovery. In the candidate generation step, we are not interested in perfectly identifying every single object, but rather in finding which parts of the scene are the most likely to be objects. Therefore, the distance to a nearest neighbor may be used as a way of ranking a scene in terms of structure. The more similar a candidate is to an object in the database, the higher the confidence that the candidate is a real object. This approach, using image data only, is explored in Hongwen Kang's forthcoming doctoral dissertation. We can exploit the additional information from range data, and use 3D shape as well as appearance to search for the most likely objects in a scene.

An additional benefit of a large-scale database of objects is the ability to generalize robotic grasping to virtually any object in the environment. Assuming that we know a set of possible actions and interactions for all objects in the database, we can potentially transfer the set of actions to the novel objects in a scene based on their similarity to known objects. For example, if we know how to grasp boxes, and we see something whose nearest neighbor is a box, we can try to generalize the current grasping knowledge to this new object, without needing to accurately know the identity of that object. In this way, we may potentially transfer knowledge about more complex interactions for specific types of objects, e.g., with handles, dangerous, or fragile, which we might want our robot to manipulate in specific ways. This transfer of knowledge could be extended to use affordances. We could exploit the metadata provided by the database about the potential uses of a given type of object to empower service robots with more complex manipulation capabilities inferred autonomously from object perception.

9.4 Physics and Context Information

An interesting follow-up to this work is to extend our constraints to incorporate new information, especially physics laws and context information to interpret the scenes and provide better discovery. Our additional constraints should encode both generic reasoning (e.g., occlusions, object stability, or volumetric constraints) and domain knowledge (e.g., configuration of indoor scenes, prior locations of objects). In essence, we want to 1) enhance the HerbDisc with a generic set of physics rules, and 2) use specific domain knowledge to infer objects that may not be visible but that are necessary to fully explain a scene.

The outcome of this process should be a physically-plausible scene interpretation that segments the visual input into whole, stable objects, alongside some relational information about how the objects interact with each other. Gupta et al. (2010) combine volumetric and stability constraints to compute high-level qualitative interpretations of natural scenes from image data, by reasoning which of the possible segmentations are physically plausible. We hypothesize that, by combining visual information and robotic metadata in the reasoning process, we can compute high-quality, detailed scene segmentations that preserve individual objects in cluttered indoors scenes, rather than the simpler natural scenes analyzed by

Gupta et al. (2010). In the case of human environments, surfaces of support play a crucial role in scene interpretation. Learning the location of commonly used surfaces of support in an environment allows us to focus future object discovery searches in the more relevant areas of the scene. These high-level scene interpretations could be used alongside an active robotic agent to guide the search for objects to the areas with higher likelihood of containing objects. This information should also be given to the object recognition step in subsequent visits to this scene, to provide faster recognition in the areas where objects are expected.

Chapter 10

Closing thoughts

Autonomous operation in the real world is *the* challenge for service robotics. In order to succeed in that challenge, robots need to perceive and understand their surroundings. The work I have presented in this thesis is a step in the right direction for object perception, but the overall problem is still formidable.

After five years of research in this area, I strongly believe that we cannot teach robots to see using visual information alone; robotic metadata and interactive perception are the key for robust perception in human environments. Robotic metadata grounds the visual information in the context of the real world, while interactive perception modifies the environment to disambiguate between perceptual cues. Every area of Object Perception should benefit from such additional information. In HerbDisc, we have shown that robotic metadata makes feasible the problem of Robotic Object Discovery. Our intermediate representation in terms of constraints offers an extensible method to encode visual information and robotic metadata in the same framework. The next step is to incorporate robotic interaction strategies to provide additional cues for long-term autonomy in perception, with the hope that one day service robots reach the necessary levels of reliability to operate in our homes.

Appendices

Appendix A

Pose Estimation from Point Correspondences

There are two main error metrics to recover the pose of a 3D model from point correspondences, the reprojection and backprojection errors. Both error functions perform equivalently when estimating object poses in Euclidean space, so one may choose either one. The reprojection error is usually preferred in the computer vision community because it is invariant to projective transformations, while the backprojection error is meaningless in projective space (Hartley and Sturm, 1997). In our particular case, working with calibrated cameras in an Euclidean space, we have chosen the backprojection error because it makes our framework more easily extensible to other types of multi-modal data, such as 3D point clouds or RGBD cameras, which we plan to incorporate in the near future. This section contains a brief derivation of the error functions we use in MOPED, both for the reprojection and backprojection errors.

A.1 Reprojection Error

Consider a set of correspondences $\mathbf{C}_m$ in image m, where each correspondence $C^o_{j;m} = (f_{j;m}, F_{i;o})$. Assume the corresponding features in $C^o_{j;m}$ have locations $p_{j;m}$ in an image and $P_{i;o}$ in an object model. For a given transformation T and an image m with extrinsic parameters T_m[1], the sum of reprojection errors is defined by

$$\text{ReprojectionErr}(T, m) = \sum_{C^o_{j;m} \in \mathbf{C}_m} \left[p_{j;m} - \text{proj}\left(T^m T P_{i;o}\right)\right]^2 \qquad \text{(A.1)}$$

[1]Reminder: in tensor notation, $T^m = (T_m)^{-1}$

The optimal single hypothesis h^* for a given set of correspondences $\mathbf{C}$ is one that minimizes the sum of reprojection errors of the correspondences across all images. The pose T_h^* for h^* is then defined as:

$$T_h^* = \arg\min_T \sum_{m=1}^{M} \text{ReprojectionErr}(T, m) \tag{A.2}$$

A.2 Backprojection Error

Alternatively, one can define an analogous optimization in terms of the backprojection error, by tracing the line $L_{j;m}$ from the camera center to each 2D point $p_{j;m}$, and computing the distance from $L_{j;m}$ to the corresponding 3D point $P_{i;o}$. We parameterize a line as $L = (c, v)$, where v is a unit vector indicating the line direction and c is an arbitrary point on that line, (e.g., the camera center). Using projective geometry, we obtain

$$\bar{v}_{j;m} = \frac{K_m^{-1} p_{j;m}}{\|K_m^{-1} p_{j;m}\|} \tag{A.3}$$

where K_m is a 3×3 intrinsic camera matrix for image m. Each line $L_{j;m}$ in the world reference frame is then given by

$$v_{j;m} = (R_m)^T \bar{v}_{j;m} \qquad c_{j;m} = -(R_m)^T t_m \tag{A.4}$$

The distance between a point $P_{i;o}$ and $L_{j;m}$ is given by

$$d(P_{i;o}, L_{j;m}) = \|\left(\mathcal{I}_{3\times3} - v_{j;m} {v_{j;m}}^T\right)(P_{i;o} - c_{j;m})\| \tag{A.5}$$

The analogous equation to Eq. (A.2) that minimizes the sum of backprojection errors of a set of correspondences $\mathbf{C}$ with $C_j = (P_{i;o}, L_{j;m})$ is given by

$$T_h^* = \arg\min_T \sum_{i=1}^{M} \sum_{C_j \in \mathbf{C}} \left[d\left(T^m T P_{i;o}, L_{j;m}\right)\right]^2 \tag{A.6}$$

Additionally, we found it useful to constrain the objects to lie in front of the cameras. Given that $v_{j;m}$ are vectors from the camera center pointing towards the image plane, ${v_{j;m}}^T (P_{i;o} - c_{j;m}) > 0$ for all points $P_{i;o}$ in front of camera m. We incorporate this constraint as a regularizer (with weight $\xi > 0$) in the minimization

$$T_h^* = \arg\min_T \sum_{i=1}^{M} \sum_{C_j \in \mathbf{C}} \left[d\left(T^m T P_{i;o}, L_{j;m}\right) + \xi \left(1 - {v_{j;m}}^T \frac{(P_{i;o} - c_{j;m})}{\|P_{i;o} - c_{j;m}\|}\right)\right]^2 \tag{A.7}$$

References

Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. What is an object? In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 73–80. 2010.

AMD. ATI Radeon HD 5970 Graphics Feature Summary, 2010. URL http://www.amd.com/us/products/desktop/graphics/ati-radeon-hd-5000/hd-5970/Pages/ati-radeon-hd-5970-specifications.aspx.

Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *International Symposium on Foundations of Computer Science*, pages 459–468. 2006.

Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

D.H. Ballard. Generalizing the Hough Transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1980.

Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

Alberto Del Bimbo and Pietro Pala. Content-based retrieval of 3D models. *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2(1):20–43, 2006.

Marten Bjorkman and Danica Kragic. Active 3D scene segmentation and detection of unknown objects. In *IEEE International Conference on Robotics and Automation*, pages 3114–3120. 2010.

Liefeng Bo, Kevin Lai, Xiaofeng Ren, and Dieter Fox. Object Recognition with Hierarchical Kernel Descriptors. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

E. Borenstein, E. Sharon, and S. Ullman. Combining Top-Down and Bottom-Up Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2004.

Dorit Borrmann, Jan Elseberg, Kai Lingermann, Andreas Nuchter, and Joachim Hertzberg. The Efficient Extension of Globally Consistent Scan Matching to 6 DOF. In *International Symposium on 3D Data Processing, Visualization and Transmission*, 2008.

Jacob V Bouvrie. Multi-Source Contingency Clustering. *Master's Thesis*, 2004.

Ulrik Brandes. A Faster Algorithm for Betweenness Centrality. *Journal of Mathematical Sociology*, 25(2):163–177, 2001.

Joao Carreira and Cristian Sminchisescu. Constrained Parametric Min-Cuts for Automatic Object Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

K.S. Chatha and R. Vemuri. Hardware-software partitioning and pipelined scheduling of transformative applications. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 10(3):193–208, 2002.

Chu-Song Chen and Wen-Yan Chang. On Pose Recovery For Generalized Visual Sensors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(7):848–61, 2004.

Yizong Cheng. Mean Shift, Mode Seeking, and Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):790, 1995.

Wei-Chen Chiu, Ulf Blanke, and Mario Fritz. Improving the Kinect by Cross-Modal Stereo. In *British Machine Vision Conference*, 2011.

C. C. Chu and J. K. Aggarwal. The Integration of Image Segmentation Maps using Region and Edge Information. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(12):1241, 1993.

Alvaro Collet and Siddhartha S. Srinivasa. Efficient Multi-View Object Recognition and Full Pose Estimation. In *IEEE International Conference on Robotics and Automation*, 2010.

Alvaro Collet, Dmitry Berenson, Siddhartha S. Srinivasa, and Dave Ferguson. Object recognition and full pose registration from a single image for robotic manipulation. In *IEEE International Conference on Robotics and Automation*, pages 48–55, 2009.

Alvaro Collet, Manuel Martinez, and Siddhartha S. Srinivasa. The MOPED framework: Object Recognition and Pose Estimation for Manipulation. *The International Journal of Robotics Research*, 30(10):1284–1306, 2011a.

Alvaro Collet, Siddhartha S Srinivasa, and Martial Hebert. Structure Discovery in Multimodal Data : a Region-based Approach. In *IEEE International Conference on Robotics and Automation*, 2011b.

Alvaro Collet, Martial Hebert, and Siddhartha S. Srinivasa. HerbDisc: Towards Lifelong Robotic Object Discovery. *International Journal of Robotics Research [In Press]*, 2012.

G. Conte, S. Tommesani, and F. Zanichelli. The Long And Winding Road to High-Performance Image Processing with MMX/SSE. In *IEEE International Workshop on Computer Architectures for Machine Perception*, page 302. 2000.

Nico Cornelis and Luc Van Gool. Fast scale invariant feature detection and matching on programmable graphics hardware. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2008.

Navneet Dalal and Bill Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 886–893. 2005.

Daniel F. Dementhon and Larry S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1-2):123–141, 1995.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

James Diebel and Sebastian Thrun. An application of markov random fields to range sensing. In *Advances in neural information processing systems*, volume 18, page 291, 2005.

Piotr Dollár and Vincent Rabaud. Piotr Dollar's Image & Video Toolbox for Matlab, 2010. URL http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html.

T.J. Dysart, B.J. Moore, L. Schaelicke, and P.M. Kogge. Cache implications of aggressively pipelined high performance microprocessors. In *IEEE International Symposium on Performance Analysis of Systems and Software*, pages 123–132, 2004.

Ian Endres and Derek Hoiem. Category Independent Object Proposals. *European Conference on Computer Vision*, 2010.

Mark Everingham, Luc Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

Tian Fang and Long Quan. Resampling structure from motion. In *European Conference on Computer Vision*, pages 1–14, 2010.

Pedro Felzenszwalb and Daniel Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2), 2004.

Martin Fischler and Robert Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

Paul Fitzpatrick. First Contact: an active vision approach to segmentation. In *International Conference on Intelligent Robots and Systems*, 2003.

David Fouhey, Alvaro Collet, Martial Hebert, and Siddhartha S. Srinivasa. Object Recognition Robust to Imperfect Depth Data. In *European Conference on Computer Vision Workshops: Consumer Depth Cameras for Computer Vision*, 2012.

Iryna Gordon and David Lowe. What and Where: 3D Object Recognition with Accurate Pose. *Toward Category-Level Object Recognition*, 4170:67–82, 2006.

Stephen Gould, Paul Baumstarck, Morgan Quigley, Andrew Y Ng, and Daphne Koller. Integrating Visual and Range Data for Robotic Object Detection. In *ECCV Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications*, 2008.

W. Eric L. Grimson. *Object Recognition by Computer*. MIT Press, 1991. ISBN 0-262-07130-4.

W Eric L Grimson and Daniel P. Huttenlocher. On the sensitivity of the Hough transform for object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(3):255–274, 1990.

W Eric L Grimson and Tomas Lozano-Perez. Localizing Overlapping Parts by Searching the Interpretation Tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):469–482, 1987.

Michael D. Grossberg and Shree K. Nayar. A general imaging model and a method for finding its parameters. In *IEEE International Conference on Computer Vision*, pages 108–115. 2001.

S. Guha and S. Khuller. Approximation Algorithms for Connected Dominating Sets. *Algorithmica*, 20(4):374–387, 1998.

Abhinav Gupta, Alexei A Efros, and Martial Hebert. Blocks World Revisited: Image Understanding Using Qualitative Geometry and Mechanics. In *European Conference on Computer Vision*, 2010.

Jay Hackett and Mubarak Shah. Multi-sensor fusion: a perspective. In *International Conference on Robotics and Automation*, 1990.

Richard Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

Trevor. Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer, 2003. ISBN 0387952845.

Michal Havlena, Akihiko Torii, and T. Pajdla. Efficient structure from motion by graph optimization. In *European Conference on Computer Vision*, pages 100–113. 2010.

Scott Helmer and David Lowe. Using Stereo for Object Recognition. In *IEEE International Conference on Robotics and Automation*, pages 3121–3127, 2010.

Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. RGB-D Mapping : Using Depth Cameras for Dense 3D Modeling of Indoor Environments. In *International Symposium on Experimental Robotics*, 2011.

Evan Herbst, Peter Henry, Xiaofeng Ren, and Dieter Fox. Toward Object Discovery and Modeling via 3-D Scene Comparison. In *IEEE International Conference on Robotics and Automation*. 2011.

Stefan Hinterstoisser, Vincent Lepetit, Slobodan Ilic, Pascal Fua, and Nassir Navab. Dominant Orientation Templates for Real-Time Detection of Texture-Less Objects. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

Stefan Hinterstoisser, Stefan Holzer, Cedric Cagniart, Slobodan Ilic, Kurt Konolige, Nassir Navab, and Vincent Lepetit. Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes. In *IEEE International Conference on Computer Vision*, pages 858–865, 2011.

D. Hoffman and Whitman Richards. Parts of recognition. *Cognition*, 18(1-3):65–96, 1984.

Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric Context from a Single Image. In *IEEE International Conference on Computer Vision*, pages 654–661. 2005.

Derek Hoiem, Andrew N. Stein, Alexei A. Efros, and Martial Hebert. Recovering Occlusion Boundaries from a Single Image. In *IEEE International Conference on Computer Vision*, pages 1–8. 2007.

Prodip Hore, Lawrence Hall, and Dmitry Goldgof. A Cluster Ensemble Framework for Large Data sets. *IEEE International Conference on Systems, Man and Cybernetics*, pages 3342–3347, 2006.

Edward Hsiao, Alvaro Collet, and Martial Hebert. Making specific features less discriminative to improve point-based 3D object recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

Fuchun Huang and Yosihiko Ogata. Generalized Pseudo-Likelihood Estimates for Markov Random Fields on Lattice. *Annals of the Institute of Statistical Mathematics*, 54(1):1–18, 2002.

Peter J. Huber. *Robust Statistics*. Wiley, 1981.

Intel Corp. Core i7 Performance, 2010. URL http://www.intel.com/support/processors/sb/cs-023143.htm.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Steve Hodges, Pushmeet Kohli, Jamie Shotton, Andrew Davison, Andrew Fitzgibbon, and Dustin Freeman. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *ACM Symposium on User Interface Software and Technology*, 2011.

Anil K. Jain and Chitra Dorai. 3D object recognition: Representation and matching. *Statistics and Computing*, 10(2):167–182, 2000.

A Janoch, S Karayev, Y Jia, J T Barron, M Fritz, K Saenko, and T Darrell. A Category-Level 3-D Object Dataset: Putting the Kinect to Work. In *Workshop on Consumer Depth Cameras in Computer Vision (in conjunction with ICCV)*, 2011.

David S. Johnson. Fast algorithms for bin packing. *Journal of Computer and System Sciences*, 8(3):42, 1974.

Hongwen Kang, Martial Hebert, and Takeo Kanade. Discovering Object Instances from Scenes of Daily Living. In *International Conference on Computer Vision*, 2011.

Hongwen Kang, Martial Hebert, Alexei A Efros, and Takeo Kanade. Connecting Missing Links : Object Discovery from Sparse Observations Using 5 Million Product Images. In *European Conference on Computer Vision*, 2012.

H. Kato and Mark Billinghurst. Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System. In *International Workshop on Augmented Reality*, 1999.

Sagi Katz and Ayellet Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954, 2003.

Kurt Konolige. Projected Texture Stereo. In *IEEE International Conference on Robotics and Automation*, pages 148–155, 2010.

Gert Kootstra and Danica Kragic. Fast and Bottom-Up Object Detection, Segmentation, and Evaluation using Gestalt Principles. In *IEEE International Conference on Robotics and Automation*, pages 3423–3428, 2011.

Kevin Lai and Dieter Fox. A Scalable Tree-based Approach for Joint Object and Pose Recognition. In *Conference on Artificial Intelligence*, 2011.

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A Large-Scale Hierarchical Multi-View RGB-D Object Dataset. In *IEEE International Conference on Robotics and Automation*, 2011a.

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Sparse Distance Learning for Object Recognition Combining RGB and Depth Information. In *IEEE International Conference on Robotics and Automation*, pages 4007–4013, 2011b.

Jean-François Lalonde, Nicolas Vandapel, Daniel F. Huber, and Martial Hebert. Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics*, 23(10):839–861, 2006.

Yong Jae Lee and Kristen Grauman. Learning the Easy Things First : Self-Paced Visual Category Discovery. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1721–1728, 2011.

Vincent Lepetit and Pascal Fua. Monocular Model-Based 3D Tracking of Rigid Objects: A Survey. *Foundations and Trends in Computer Graphics and Vision*, 1(1):1–89, 2005.

Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal of Computer Vision*, 81(2):155–166, 2008.

Manolis Lourakis. levmar: Levenberg-Marquardt nonlinear least squares algorithms in C/C++, 2004. URL http://www.ics.forth.gr/~lourakis/levmar.

David Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, 31(3):355–395, 1987.

David Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Michael Maire, Pedro Arbelaez, Charless C Fowlkes, and Jitendra Malik. Using Contours to Detect and Localize Junctions in Natural Images. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

Donald W Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parmaters. *SIAM Journal on Applied Mathematics*, 11(2):431–441, 1963.

Manuel Martinez, Alvaro Collet, and Siddhartha S. Srinivasa. MOPED: A Scalable and low Latency Object Recognition and Pose Estimation System. In *IEEE International Conference on Robotics and Automation*, 2010.

Zoltan-csaba Marton, Dejan Pangercic, Nico Blodow, Jonathan Kleinehellefort, and Michael Beetz. General 3D Modelling of Novel Objects from a Single View. In *IEEE International Conference on Intelligent Robots and Systems*, pages 3700–3705, 2010.

Merriam-Webster.com. Merriam-Webster Dictionary, 2012. URL http://www.merriam-webseter.com/dictionary.

Krystian Mikolajczyk and Cordelia Schmid. A Performance Evaluation of Local Descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

Ajay K. Mishra and Yiannis Aloimonos. Visual Segmentation of Simple Objects for Robots. In *Robotics: Science and Systems*, 2011.

Ajay K. Mishra, Ashish Shrivastava, and Yiannis Aloimonos. Segmenting simple objects using RGB-D. In *IEEE International Conference on Robotics and Automation*, pages 4406–4413. 2012.

T. Morwald, J. Prankl, A. Richtsfeld, M. Zillich, and M. Vincze. BLORT - The Blocks World Robotic Vision Toolbox. In *IEEE International Conference on Robotics and Automation Workshops*, 2010.

Marius Muja and David Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Applications*, pages 331–340, 2009.

David Nister. Frame Decimation for Structure and Motion. In *Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, pages 17–34, 2000.

Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(3):448–61, 2010.

James Philbin, Josef Sivic, and Andrew Zisserman. Geometric Latent Dirichlet Allocation on a Matching Graph for Large-scale Image Datasets. *International Journal of Computer Vision*, 95(2):138–153, 2010.

Ingmar Posner, Derik Schroeter, and Paul Newman. Online generation of scene descriptions in urban environments. *Robotics and Autonomous Systems*, 56(11):901–914, 2008.

Richard A. Redner and Homer F. Walker. Mixture Densities, Maximum Likelihood and the EM Algorithm. *SIAM Review*, 26(2):195–239, 1984.

M. Ruhnke, B. Steder, G. Grisetti, and Wolfram Burgard. Unsupervised learning of 3D object models from partial views. *IEEE International Conference on Robotics and Automation*, pages 801–806, 2009.

Bryan C. Russell, William T. Freeman, Alexei A. Efros, Josef Sivic, and Andrew Zisserman. Using Multiple Segmentations to Discover Objects and their Extent in Image Collections. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2006.

Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. LabelMe: A Database and Web-Based Tool for Image Annotation. *International Journal of Computer Vision*, 77(1-3):157–173, 2008.

R B Rusu, G Bradski, R Thibaux, and J Hsu. Fast 3D recognition and pose using the Viewpoint Feature Histogram. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2155–2162, 2010.

Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation*, 2011.

Radu Bogdan Rusu, Zoltan Csaba Marton, Nico Blodow, M Dolha, and Michael Beetz. Towards 3D Point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008.

Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. *IEEE International Conference on Robotics and Automation*, pages 3212–3217, 2009a.

Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments. In *IEEE International Conference on Intelligent Robots and Systems*, pages 1–6. 2009b.

Gerald Schweighofer and Axel Pinz. Robust pose estimation from a planar target. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(12):2024–30, 2006.

Andrea Selinger and Randal C. Nelson. Appearance-based object recognition using multiple views. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 905–911. 2001.

E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 469–476. 2001.

N Silberman, D Hoiem, P Kohli, and R Fergus. Indoor Segmentation and Support Inference from {RGBD} Images. In *European Conference on Computer Vision*, 2012.

Chanop Silpa-Anan and Richard Hartley. Optimised KD-trees for fast image descriptor matching. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. 2008.

Josef Sivic, Bryan C. Russell, Alexei A. Efros, Andrew Zisserman, and William T. Freeman. Discovering Objects and their Location in Images. In *IEEE International Conference on Computer Vision*, pages 370–377. 2005.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. Photo tourism: Exploring photo collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, 2006.

Noah Snavely, Steven M. Seitz, and Richard Szeliski. Skeletal graphs for efficient structure from motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. 2008.

Gowri Somanath, Rohith MV, Dmitris Metaxas, and Chandra Kambhamettu. D - Clutter: Building object model library from unsupervised segmentation of cluttered scenes. In *IEEE Conference on Computer Vision and Pattern Recognition*. 2009.

Siddhartha S. Srinivasa, Dave Ferguson, Casey J. Helfrich, Dmitry Berenson, Alvaro Collet, Rosen Diankov, Garratt Gallagher, Geoffrey Hollinger, James Kuffner, and Michael Vande Weghe. HERB: a home exploring robotic butler. *Autonomous Robots*, 28 (1):5–20, 2010.

Siddhartha S Srinivasa, Dmitry Berenson, Maya Cakmak, Alvaro Collet, Mehmet R Dogar, Anca D Dragan, Ross A Knepper, Tim Niemueller, Kyle Strabala, Mike Vande Weghe, and Julius Ziegler. HERB 2.0: Lessons Learned from Developing a Mobile Manipulator for the Home. 100(8):1–17, 2012.

Alexander Strehl and Joydeep Ghosh. Cluster Ensembles - A Knowledge Reuse Framework for Combining Multiple Partitions. *Journal of Machine Learning Research*, 3(1):583–617, 2002.

Richard Szeliski. *Computer Vision: Algorithms and Applications.* Springer, 2011.

Richard Szeliski and Sing Bing Kang. Recovering 3D shape and motion from image streams using nonlinear least squares. *Journal of Visual Communication and Image Representation*, 5(1):10–28, 1994.

Pingbo Tang, Daniel Huber, and Burku Akinci. A Comparative Analysis of Depth-Discontinuity and Mixed-Pixel Detection Algorithms. In *International Conference on 3-D Digital Imaging and Modeling*, pages 29–38. 2007.

Wei Tang, Zhengdong Lu, and Inderjit S Dhillon. Clustering with Multiple Graphs. In *IEEE International Conference on Data Mining*, 2009.

J.W.H. Tangelder and R.C. Veltkamp. A survey of content based 3D shape retrieval methods. In *Shape Modeling Applications.* 2004.

Thorsten Thormahlen, Hellward Broszio, and Axel Weissenfeld. Keyframe Selection for Camera Motion and Structure Estimation from Multiple Views. In *European Conference on Computer Vision*, pages 523–535, 2004.

Olga G Troyanskaya, Kara Dolinski, Art B Owen, Russ B Altman, and David Botstein. A Bayesian framework for combining heterogeneous data sources for gene function prediction (in Saccharomyces cerevisiae). *Proceedings of the National Academy of Sciences*, 100 (14):8348–53, 2003.

Zhuowen Tu and Song-Chun Zhu. Image Segmentation by Data-Driven Markov Chain Monte Carlo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5): 657–673, 2002.

Tinne Tuytelaars, Christoph H. Lampert, Matthew B. Blaschko, and Wray Buntine. Unsupervised Object Discovery: A Comparison. *International Journal of Computer Vision*, 88(2):284–302, 2009.

Luca Vacchetti, Vincent Lepetit, and Pascal Fua. Stable real-time 3D tracking using online and offline information. *IEEE transactions on pattern analysis and machine intelligence*, 26(10):1385–91, 2004.

J. Van Weveren. From Quaternion to Matrix and Back. Technical report, Id Software, Inc., 2005.

Fredrik Viksten, Robert Soderberg, Klas Nordberg, and Christian Perwass. Increasing pose estimation performance using multi-cue integration. In *IEEE International Conference on Robotics and Automation*, pages 3760–3767. 2006.

Fredrik Viksten, Per-Erik Forssén, Björn Johansson, and Anders Moe. Comparison of local image descriptors for full 6 degree-of-freedom pose estimation. In *IEEE International Conference on Robotics and Automation*, pages 1139–1146, 2009.

Michael Wand, Matthias Fischer, Ingmar Peter, Friedhelm Meyer, and Wolfgang Straser. The randomized z-buffer algorithm: interactive rendering of highly complex scenes. In *SIGGRAPH*, pages 361–370, 2001.

M. Weber, M. Welling, and Pietro Perona. Towards automatic discovery of object categories. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 101–108. 2000.

Max Wertheimer. Laws of organization in perceptual forms. In W. B. Ellis, editor, *A Sourcebook of Gestalt Psychology*, pages 71–88. Harcourt, Brace and Company, 1938.

Barry Wilkinson and Michael Allen. Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers. chapter 3. Prentice Hall, 2 edition, 2004.

WillowGarage. The Personal Robot Project, 2008. URL http://www.willowgarage.com.

WillowGarage. Objects Pan-tilt Database, 2010. URL http://vault.willowgarage.com/wgdata1/vol1/objects_pantilt_database/.

Changchang Wu. SiftGPU: A GPU Implementation of Scale Invariant Feature Transform (SIFT), 2007. URL http://cs.unc.edu/~ccwu/siftgpu.

Erliang Zeng, Chengyong Yang, Tao Li, Giri Narasimhan, Fitzpatrick Hall, Notre Dame, I N Life, Centre Drive, Foster City, and C A Bioinformatics. Clustering Genes using Heterogeneous Data Sources. *International Journal of Knowledge Discovery in Bioinformatics*, 1(2):12–28, 2010.

Zhengyou Zhang. Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. *Image and Vision Computing*, 15(1):59–76, 1997.