Three-Dimensional Object Search, Understanding, and Pose Estimation with Low-Cost Sensors

Yan Wang

Submitted in partial fulfillment of the
requirements for the degree
of Doctor of Philosophy
in the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2015

ABSTRACT

Three-Dimensional Object Search, Understanding, and Pose Estimation with Low-Cost Sensors

Yan Wang

With the recent development of low-cost depth sensors, an entirely new type of 3D data is being generated rapidly by regular consumers. Traditionally, 3D data is produced by a small number of professional designers (i.e., the Computer Aided Design (CAD) model); however, 3D data from massive consumer-level sensors has the potential of introducing many new applications, such as user-captured 3D warehouse and search engines, robots with 3D sensing capability, and customized 3D printing. Nevertheless, the low-cost sensors used by general consumers also pose new technological challenges. First, they have relatively high levels of sensor noise. Second, the use of such consumer devices is often in uncontrolled settings, resulting in challenging conditions, such as poor lighting, cluttered scenes, and object occlusion. To address such emerging opportunities and associated challenges, this dissertation is dedicated to the development of novel algorithms and systems for 3D data understanding and processing, using input from a consumer-level 3D sensor.

In particular, the key problems of 3D shape retrieval, scene understanding, and pose recognition are explored in order to present a comprehensive coverage of the key aspects of content-based 3D shape analysis. To resolve the aforementioned challenges, we propose a flexible Markov Random Field (MRF) framework that uses local information to allow partial matching, and thus address the model incompleteness problem; the framework also uses higher-order correlation to provide additional robustness against sensor noise. With the MRF framework, these 3D analysis problems can be transformed into a unified potential energy minimization problem, while preserving the flexibility to adapt to different settings and resolve the unique challenges of each problem. The contributions of the dissertation include:

a. **Cross-Domain 3D Retrieval**: First we tackle the problem of searching 3D noise-free models using noisy data captured by low-cost 3D sensors – a unique cross-domain setting. To manage the challenges of sensor noise and model incompleteness from consumer-level sensors, we propose a novel MRF formulation for the retrieval problem. The potential function of the random field is designed to capture both the local shape and global spatial consistency in order to preserve the local matching capability, while offering robustness against the sensor noise. The specific form of the potential functions is determined efficiently by a series of weak classifiers, thus forming a variant of the Regression Tree Field (RTF). We achieve better retrieval precision and recall in the cross-domain settings with a consumer-level depth sensor compared with state-of-the-art approaches.

b. **3D Scene Understanding**: We develop a scene understanding system based on input from consumer-level depth sensors. To resolve the key challenge of the lack of annotated 3D training data, we construct an MRF that connects the input 3D point cloud and the associated 2D reference images, based on which the 3D point cloud is stitched. A series of weak classifiers are trained to obtain an approximate semantic segmentation result from the reference images. The potential function of the field is designed to integrate the results from the classifiers, while taking advantage of the 3D spatial consistency in order to output a comprehensive scene understanding result. We achieve comparable accuracy and much faster speed compared with state-of-the-art 3D scene understanding systems, with the difference that we do not require annotated 3D training data.

c. **Pose Recognition of Deformable Objects**: We develop a method for supporting a robotics system to recognize pose and manipulate deformable objects. More specifically, garment pose is recognized with the help of an offline simulated database and the proposed retrieval approach. We use a novel binary feature representation extracted from the reconstructed 3D surfaces in order to allow efficient matching, thus achieving real-time performance. A spatial weight is further learned in order to integrate the local matching result. The system shows superior recognition accuracy

and faster speed than the state-of-the-art approaches.

d. Application with 2D Data: In addition to the traditional 3D applications, we explore the possibility of extending MRF formulation to 2D data, especially those used in classical low-level 2D vision problems, such as image deblurring and denoising. One well-known technique that uses image prior, the probabilistic patched-based prior, is known to have bottlenecks in finding the most similar model from a model set, which can be posed as a retrieval problem. Therefore, we apply the MRF formulation originally developed for 3D shape retrieval, and extend it to this 2D problem by introducing a grid-like random field structure. We can achieve 40x acceleration compared with the state-of-the-art algorithm, while preserving quality.

We organize the dissertation as follows. First, the core problems of 3D shape retrieval, scene understanding, and pose recognition, and with the proposed solutions that use MRF and RTF are explored in Part I. In Part II, the extension to 2D data is discussed. Extensive evaluation is performed in each specific task in order to compare the proposed approaches with state-of-the-art algorithms and systems, and also to justify the components of the proposed methods. Finally, in Part III, we include the conclusion remarks and discussion of open issues and future work.

Table of Contents

List of Figures

List of Tables

Acknowledgment

The five-year Ph.D. life at Columbia University has made me a more mature person, both in terms of technical skills and personality. This process would not occur without the guidance, help, and company of my mentors, friends, and family.

First, I would like to express my immense gratitude to my Ph.D. advisor, Professor Shih-Fu Chang, who led me through the Ph.D. training and taught me the critical skills necessary to survive in the professional world. I am sincerely grateful for Prof. Chang persistently offering guidance, encouragement, and support to my study, and to the use of his vision, which led me to the fascinating field of 3D computer vision and machine learning. Prof. Chang taught me many valuable skills, especially to emphasize the easy intuition behind complicated problems, and systematic analysis skills to master common human mistakes ; such skills benefitted me not only in my research, but also in many other aspects. In addition, his supportive attitude in allowing me to perform multiple internships permitted me to gather comprehensive skills from various industries, which shaped my skillset along with the academic guidance obtained from him.

I also received generous help from my intern managers, Omid Rouhani-Kalleh, Dr. Jonathan Chang, Dr. Jue Wang, and Dr. Eyal Ofek. They not only taught me how to find important research problems from real products, but also guided me through a systematic approach to manage both the project and the related resources.

It has been a great fortune and pleasure to work with friends from the Digital Video and Multimedia Lab. Here, I would like to thank Dr. Jun Wang, Dr. Rongrong Ji, Dr. Wei Liu, Dr. Zhenguo Li, Dr. Yu-Gang Jiang, Dr. Yadong Mu, Dr. Tao Chen, Dr. Dong Liu, Dr. I-Hong Juo, Dr. Junfeng He, Guangnan Ye, Felix Yu, Xiao-Ming Wu, Hongzhi Li, Brendan Jou, Joe Ellis, Jie Feng, and Zheng Shou for their insightful discussion and company, which brought much inspiration and support. I would also like to thank Jim

Simon, Matthew Finley, and Kai Wang, who helped reshape my personality to become a more calm, patient, and rigorous individual through pilot training. During these five years, it was my parents who shared happiness and sorrow with me, and supported me under all circumstances. I would not have been able to make such fruitful journey without their support and encouragement.

Last, I sincerely thank the dissertation committee members, Prof. Zoran Kostic, Prof. Xiaodong Wang, Prof. Matei Ciocarlie, and Dr. Jue Wang, for attending my Ph.D. defense and offering precious feedback on the dissertation.

To my parents.

Chapter 1

Introduction

1.1 Motivation

With the continuing development of 3D design tools and sensors, and the recently emerged consumer-level 3D sensors, we face explosive growth of 3D data, both from manual design and user capture. This calls for novel scalable 3D search, understanding, and registration techniques, with the potential of not only pushing the frontier of computer vision research, but also igniting novel applications and markets.

Traditional 3D acquisition approaches, such as manual design (Computer Aided Design–CAD), structure from motion, and Light Detection and Ranging (LiDAR) sensors, have been developing rapidly in recent years, and have contributed large amounts of 3D data to 3D warehouses, either for public access or commercial licensing. For example, Sketchup 3D Warehouse already has millions of 3D models[1], and this amount continues to grow. Google Maps launched its 3D version in 2012, and it has collected the 3D models of hundreds of cities[2] using vehicle-carried LiDAR sensors, as well as satellite and aircraft images. The 3D models of hundreds of museums have also provided virtual tours to the Google Art Project. In addition, Microsoft offers a popular tool to users, PhotoSynth, to allow them to synthesize and share 3D models through a dense collection of photos captured from diverse views.

[1] As of Jan. 2015.

[2] As of Jan. 2015.

Meanwhile, low-cost handheld 3D sensors, such as Microsoft Kinect, Apple PrimeSense, and Google Project Tango [Google Inc.], allow inexperienced consumers to capture their own 3D models for the first time. Some low-end sensors, such as first generation Kinect, can capture depth typically by projecting and measuring random infrared patterns [Zhang, 2012] at a cost of approximately 100 US dollars. Some higher-end models, such as Microsoft Kinect 2, use Time-of-Flight cameras to obtain higher-precision 3D scans; however, their cost continues to be well below 300 US dollars, whereas traditional 3D sensors, such as LiDAR, cost tens of thousands of dollars. Although such sensors have the limitation of only being able to work indoors in a limited range (usually less than 15 feet), they allow users to create, store, review, and share 3D scans of objects, and have stimulated rapid development of new consumer markets.

One market that has changed dramatically is 3D printing. Before low-cost 3D sensors, the main users of 3D printing were professional manufacturers who needed to print human-designed models. Recently, however, certain startup companies use multiple Kinect sensors (rather than million-dollar professional equipment) to capture the entire body of a user, and then print the captured 3D model after simple editing [Shapify], thus resulting in an end-to-end product. In other fields, the sensors also bring new interaction methods to online conferencing, gaming, arts, and design.

Given the large amount of accumulated 3D data, there is a growing need to develop automatic techniques that facilitate efficient search, manipulation, and recognition of 3D data. Without losing the depth dimension, 3D information from sensors has much less ambiguity than 2D photos, such as preserving physical size and distance from the camera. This helps bring new opportunities to the computer vision field. For example, face [Bowyer *et al.*, 2006] and human pose recognition [Shotton *et al.*, 2013] are easier with 3D data than 2D data. In addition, industrial prototypes for self-driving vehicles based on 3D sensors have been manufactured, which is extremely challenging, if not impossible, with 2D vision.

3D data pose new challenges, especially for those captured from low-cost sensors. One such challenge is that the output from these consumer level sensors is extremely noisy and unreliable. This causes significant performance degeneration of existing 3D indexing and understanding algorithms [Machado *et al.*, 2013], and thus requires additional robustness

in the algorithm design. The second challenge is that the captured model from 3D sensors is typically incomplete, in contrast with the assumption of traditional 3D content analysis, which is that the model is manually designed and thus usually complete. Such incompleteness usually originates from the object's self-occlusion. To alleviate this problem, various 3D registration algorithms have been proposed to combine multiple depth input to one 3D model [Izadi *et al.*, 2011]. However, it is still not practical to conduct a 360-degree scan of certain objects – for instance, the bottom of a vehicle. Hence, to properly take advantage of the user-captured 3D data, the processing algorithm has to offer the capability of managing incomplete input models, which, unfortunately, is barely addressed by existing approaches.

Another challenge originates from the unique structure of the 3D data, which is essentially different from 2D images. In 2D images, the sensing units are structured as grids on the sensors, thus producing *dense* data; this means that for every captured pixel, it can be guaranteed that it has neighbors above, below, and to its left and right (with the exception of boundary pixels). This property allows many effective and efficient processing techniques, among which convolution and integral images are especially important. However, 3D data are structured as *sparse* vertices and edges, and there is no guarantee of neighborhood in the 3D space. On one hand, this resolves some ambiguity and makes segmentation much easier; on the other hand, this makes techniques that rely on a dense structure, such as convolution and Scale-Invariant Feature Transform (SIFT) [Lowe, 1999], invalid. Therefore, new techniques that fit such different sparse data structures are required.

1.2 Content-based 3D Analysis

In this dissertation, we develop comprehensive novel techniques to solve the aforementioned critical problems associated with 3D search, scene understanding, and object pose estimation, in order to present a coherent coverage of problems related to different aspects of 3D content analysis. In particular, we target the low-cost sensors, and provide a unified framework to resolve the model incompleteness and sensor noise problems in different applications.

1.2.1 3D Shape Retrieval

Content-based shape retrieval is defined as obtaining 3D models that are visually similar to the *query* from a given *database*, and "reranking" the 3D models according to the similarity. Such retrieval plays a fundamental role in 3D applications. First, shape retrieval is an important application for users. The capability to search for 3D models similar to a scanned model can dramatically improve the efficiency of 3D modeling, industrial design, and game design. Second, given a sufficiently large database, 3D retrieval can also benefit other research fields via metadata transfer [Malisiewicz *et al.*, 2011a,b; Kuettel *et al.*, 2012]. For example, one state-of-the-art solution to the problem of 3D object detection is to search for the most similar model in a large database in a sliding-window manner [Song and Xiao, 2014]. As is shown later, in addition to the semantic category, other metadata can be transferred via retrieval. Examples include object pose, geometry shape, and even patterns (e.g., edges and peaks) of 2D image data.

However, the 3D shape retrieval problem is especially difficult in our setting, where the database consists of complete and noise-free models from human designers, and the query of incomplete and noisy 3D models from low-cost sensors. To address these challenges, we employ a variant of the Regression Tree Fields (RTFs). In particular, a Markov Random Field (MRF) is built on the query 3D model, and the retrieval process is formulated as a minimization problem of its *potential function*. The potential function is determined by two factors: 1) an approximate retrieval result based merely on a local region, and 2) the consistency of retrieval results across different local regions. The first factor provides the capability of local matching, and thus addresses the challenge of model incompleteness, and the second factor integrates information across different regions to obtain additional robustness against sensor noise. In order to expedite the optimization process, we also use random forests as the indexing structure.

1.2.2 3D Scene Understanding

The problem of 3D scene understanding is to provide point-wise reasoning of semantic categories (e.g., *table* and *floor*) given a 3D scene as input, typically in the form of a point cloud or mesh model. This is the fundamental for many applications, such as self-driving

cars and semantic-aware augmented reality. In addition to the challenges from the low-cost sensors, 3D scene understanding faces the problem of lack of training data. Because the semantic category classification problem is essentially a supervised learning problem, it requires correspondences of 3D regions and category annotations as the training data. Unfortunately, we still do not have sufficient annotated data to support a reasonably good classifier. Whereas the computer vision community dedicated a decade building a large-scale annotated database [Deng *et al.*, 2009], it is much more difficult for us to repeat this endeavor in 3D, given that considerably more effort is required to label a boundary in 3D with the current technology.

From another perspective, there is hope in the fact that capture of 3D scenes generally requires registering and stitching multiple RGB-Depth (RGBD) inputs [Endres *et al.*, 2012; Izadi *et al.*, 2011], which indicates that the input of scene understanding also contains the underlying RGB *reference images* and the correspondences between each pixel and 3D vertex. This inspires us to utilize large-scale annotated 2D datasets, such as ImageNet [Deng *et al.*, 2009] and LabelMe [Russell *et al.*, 2008], in order to improve the 3D scene understanding, by using the RGB reference images from the stitching process. To address the unique challenge of 2D-3D cross-domain knowledge transfer, an MRF is constructed to associate the underlying 2D images with the 3D input. The potential function of the MRF is designed not only to fuse the annotation from the 2D images, but also to encourage the spatial consistency of the inferred categories. Linear Support Vector Machines (SVMs) are chosen to provide an approximate classification result based on local regions in order to provide partial matching capability, and a ranking SVM is further trained on top of them to make the output score comparable even across different SVMs.

1.2.3 Pose Estimation of 3D Objects

Another important aspect of content-based 3D analysis is to infer the pose of a 3D object given its geometry, for instance, given a garment grasped by a robotic arm at some point, reasoning the relative position of the grasping point on the garment (e.g., at the left shoulder). It is a critical component in building automatic industrial robots, and has attracted a lot of attention. Motivated by the vast application potential in industry, we also explore the

problem of pose estimation of garments based on a robot platform. We do not follow the existing approach of training a universal classifier to consider all possible variances [Li *et al.*, 2014b] because the dramatic variance in garment type, material, shape, and illumination is too difficult to be captured by a single classifier. Instead, we choose to first perform offline simulations for multiple variables, such as types, materials, etc. with industry level physics engines, store the resulting 3D model in a database, and search for similar instances from the input. This way, the complexity of classifier learning is transferred to efficient database indexing, for which we already have solutions from the previous two problems.

Because we face a well-controlled working environment in robotics applications, we address the model incompleteness challenge by allowing the robot arm to grasp the garment in the air, and rotate the garment by 360 degrees. By employing real-time RGBD stitching algorithms [Izadi *et al.*, 2011], we ensure that the processed input encapsulates the complete shape of the garment. In addition, in order to satisfy the expectation of real-time performance in industrial application, a compact binary feature is proposed to save space and time in the robotic platform, and a ranking SVM are used to learn a weighted Hamming Distance to further enhance the pose estimation accuracy.

1.2.4 Extension to 2D Applications

Although not directly related, 2D applications are also discussed to demonstrate the potential of the aforementioned techniques being applied to 2D data, especially for low-level vision problems, such as image deblurring and denoising. Given a noisy or blurred image, recovering a clear image is generally an ill-posed problem. Therefore, image prior models learned from natural images are used to constrain the optimization problem. Recently, promising performance is found in a new branch of patch-based image priors, such as Expected Patch Log-Likelihood (EPLL), which first inspects every image patch to identify its pattern, and then applies a corresponding optimization model to recover the clear image. However, this exhaustive inspection process of patterns is time consuming. We extend the MRF formulation for 3D shape retrieval to this setting, and build a retrieval engine for the patterns. In particular, we build an MRF on the query image with each overlapped image patch as a node, and design the potential function required to encapsulate the EPLL

formulation and spatial consistency. A random forest is also built as the indexing structure to expedite the optimization process.

1.3 A Unified Framework

We select the topics of shape retrieval, scene understanding, and pose estimation of 3D objects to cover the key aspects of 3D content analysis. Although the definitions and settings for each topic are different, the proposed approaches actually share a unified framework whose eventual goal is to address the challenges illustrated in Section 1.1. In the following sections, we illustrate the common themes shared by all the proposed approaches, and provide justification for why the approaches help solve the challenges.

1.3.1 MRF Formulation for Non-Grid Data

As mentioned above, the key difference between 2D and 3D data is that 3D data has a unique sparse structure that does not ensure spatial neighborhood relationships. This is especially troublesome in shape retrieval. A typical (2D) search engine contains one or multiple feature representations, an indexing structure, and generally, a spatial consistency-based reranking module. However, the special mesh structure of 3D models, usually organized as a set of vertices and edges (with optional texture), breaks multiple components of the existing retrieval pipeline. For feature representations, the lacking of dense pixel structures invalidates important 2D processing techniques, such as convolution and gradient, on which many popular 2D features rely, such as DoG, SIFT, and SURF, and thus immediately prevents the extension of major 2D corner detectors and descriptors to 3D. Therefore, 3D models cannot use simple extension of 2D features, but have to develop their own representation.

Meanwhile, traditional 2D spatial verification approaches, such as RANSAC on SIFT, are also invalid. Although we could use traditional 3D spatial verification approaches, such as Iterated Closest Point (ICP), they are actually designed for registration and do not fit the retrieval scenario, where efficiency is emphasized. Therefore, the traditional 2D search pipeline has quite a few components broken in 3D, and we need to propose new approaches

to resolve the issue.

In 2D recognition or scene understanding approaches, MRFs are extensively used because their inter-node potential can effectively encapsulate the expected label smoothness in a neighborhood. This property makes MRF formulation look promising because of the robustness it provides to resolve sensor noise. From another perspective, MRFs are also sufficiently flexible to apply to non-grid structures, with various definitions of cliques and corresponding potential functions. Therefore, although not typically used in retrieval, MRFs can actually be suitable models for retrieval applications. In particular, in our approaches of shape retrieval, scene understanding, and low-level vision, an MRF is built on the query with the unary potential that captures local similarity, and higher-order potential that captures spatial consistency. The flexibility of potential function design and fast inference also allows the formulation to be adapted to the specific settings of different applications.

1.3.2 Extension Beyond Local Regions to Fill the Model Gap

With the low-cost sensors used to generate input for the analysis or retrieval system, such 3D input usually contains severe sensor noise, and it is incomplete because of self-occlusion. On the contrary, the majority of the 3D databases consist of complete and noise-free CAD models. This gap between the input and database models requires special data management, and it is called the *model gap*.

Model incompleteness requires the system to use local features to allow partial matching. Local feature extraction and partial matching are not challenging. However, they are much more troublesome when the input is also noisy. When the neighborhood from which features are extracted becomes smaller, this also brings less discriminative ability and more ambiguity. Combined with the considerable noise from the sensor, this significantly reduces the Signal-to-Noise Ratio in the feature representation, and may make the retrieval process unreliable.

Fortunately, we have the mesh structure from the 3D data that can possibly help us combine the information from nearby 3D neighborhoods to filter noise. More specifically, if some nearby 3D regions have consistent retrieval results, we can be more confident about these results; otherwise, we downgrade their confidence score. This idea can easily be

integrated as a high-order potential function of MRFs. In addition, we use *random forests* to better divide and index the feature space compared with traditional KDTrees, with the idea of searching with an ensemble of classifiers, and ending with the unary potential function for the MRF. In summary, the incompleteness problem is solved by local feature representation instead of global shapes, and various learning techniques, including random forests and MRFs, are used to enhance robustness against sensor noise.

1.3.3 Summary

In summary, we propose a novel and unified framework for different key problems of content-based 3D analysis: shape retrieval, scene understanding, and pose estimation. An MRF formulation is proposed to address the unique mesh structure of the 3D models, with the unary potential that describes local similarity, and higher-order potential for the spatial consistency. On one hand, this solves the challenges of incompleteness and noise from low-cost sensors by fusing the noisy local matching scores, and on the other hand, this takes advantage of large-scale databases in other domains, such as annotated 2D datasets. Efficient indexing and approximate optimization techniques are also used to accelerate MRF inference.

1.4 Related Work

In this section, we provide a high-level literature review, providing readers with an overview of the related fields, and thus the technical contribution of the dissertation. More detailed prior art review can be found in each individual chapter.

1.4.1 2D Image Retrieval Systems

A typical 2D image retrieval system involves one or more *feature representations* to describe global or local properties of the images, an *indexing structure* to expedite the retrieval process by filtering unpromising candidates, and a *reranking module* to rerank the candidates. In this subsection, we focus mainly on the feature representations and reranking module, and leave the review of indexing structures to Section 1.4.4.

Feature Representations. Visual features have long history in computer vision given their fundamental position in many applications. The features can generally be divided into two categories, *global* and *local*. Global features capture information from the entire image, varying from the simple color histogram to more complicated gist features [Oliva and Torralba, 2006], and are more compact compared to local features. Local features only describe a local region in the image, and subsequently, they are capable of partial matching. Examples include SIFT [Lowe, 1999], SURF [Bay *et al.*, 2006], and MSER [Donoser and Bischof, 2006].

Local features are capable of providing partial matching capability; however, they have the problem of redundancy and require an additional step of image-level *pooling* in order to obtain image-wise representation, which inspires different *mid-level representations*. Popular approaches include Bag-of-Visual-Words [Sivic and Zisserman, 2003], Spatial Pyramid Matching [Lazebnik *et al.*, 2006], and Pyramid Matching Kernel [Grauman and Darrell, 2005].

Reranking. The most usual reranking method is geometric verification [Sivic and Zisserman, 2003; Philbin *et al.*, 2007b]; however, other reranking approaches are sometimes also used, such as graph-based [Wang *et al.*, 2012c], information-theory-based [Hsu *et al.*, 2006], and user-click-based reranking [Agichtein *et al.*, 2006].

1.4.2 3D Model Retrieval Systems

General Framework. As illustrated previously, the 3D model retrieval systems face several unique challenges compared to 2D image retrieval systems, especially in handling the mesh representation. To manage this problem, the community has established two directions. One direction is to project 3D models to 2D imaging planes so that the 3D retrieval problem is converted into a view retrieval problem based on 2D images [Chen *et al.*, 2003; Daras and Axenopoulos, 2010]. The advantage of this approach is that it directly benefits from the mature features of 2D image analysis, such as SIFT [Lowe, 1999], which provides more robustness against noise and appearance variance. However, the view projection process inevitably produces large amount of redundant data, posting challenges to the speed and scalability of the system. Another direction uses a similar framework with 2D retrieval

systems [Dutagaci *et al.*, 2009; Li *et al.*, 2012; Pickup *et al.*, 2014; Wang *et al.*, 2014b], which contains a feature extractor, indexing structure, and spatial consistency checker/reranker, but it requires development of new features to fit the new type of 3D data.

Feature Representations. There are also different trends for 3D feature design. One popular direction is to use a heat transfer process to embed 3D spatial information, for example, the Heat Kernel Signature (HKS) [Sun *et al.*, 2009; Bronstein and Kokkinos, 2010],and Intrinsic Shape Context (ISC) [Kokkinos *et al.*, 2012]. Local geometric properties have also been used to describe 3D shapes, such as Persistent Feature Histograms (PFH) [Rusu *et al.*, 2008] and Fast Point Feature Histograms (FPFH) [Rusu *et al.*, 2009]. The unique topological structure of 3D models is also used to construct descriptive features, resulting in topological dictionaries [Tung and Matsuyama, 2012; Huang *et al.*, 2010]. Inspired by the tremendous success of the Difference of Gaussian (DoG) detector and Histogram of Gradient (HoG) descriptor, some researchers have also extended such features to 3D mesh models by treating a 3D model as a graph, resulting in MeshHoG [Zaharescu *et al.*, 2009] as a 3D extension of the SIFT feature. Similar ideas for extending 2D mid-level representations to 3D cases can also be found in Bag-of-Visual-Words [Bronstein *et al.*, 2011; Pickup *et al.*, 2014].

1.4.3 Probabilistic Graphical Model

General Formulation. A Probabilistic Graphical Model (PGM) [Koller and Friedman, 2009], such as MRF, is a powerful statistical model capable of encapsulating prior knowledge and variable inter-correlation. A PGM is essentially an undirectional graph that consists of a set of vertices and a set of edges. Each vertex is associated with a variable. A function called *potential function* is defined on each clique (i.e., fully connected component). The output of the graphical model is obtained by minimizing the potential function with regard to the variables.

There are two views to interpret the probabilistic graphical model. First, we can interpret it from a probabilistic view. The potential function is often defined as the negative log-likelihood of the joint distribution of the variables. Therefore, minimizing the potential is equivalent to optimizing the variables for the maximum likelihood. Another possible

interpretation is from a pure optimization perspective, and does not assume any additional properties on the potential.

Potential Design and Applications. In computer vision applications, PGM is widely used in scenarios where the inter-correlation of the variables needs to be captured. In these application scenarios, the potential usually has the form of the sum of a unary potential and pairwise potentials. The unary potential is typically defined on a single vertex/variable (or a small region), thus encapsulating information from a local area. The pairwise potential is defined on a pair of vertices (or a larger clique), thus capturing the cross-vertex correlation.

For example, in binary image segmentation [Rother *et al.*, 2004], the unary potential describes the probability of each pixel to be the foreground using a Gaussian Mixture Model (GMM), and the pairwise potential enforces the smoothness of labels, which can be either foreground or background. Similar settings can also be found in scene understanding [Gould *et al.*, 2009b] and image denoising [Beck and Teboulle, 2009].

Inference. Depending on the formulation of the PGM, different inference approaches can be used to find the optimal or an optima point of the potential function. When the PGM variables are continuous, continuous optimization solvers can be used, among which L-BFGS [Liu and Nocedal, 1989] is especially popular when the objective function is convex because of its fast convergence speed and efficient memory space utilization. When the variables are discrete, which is especially common in applications such as image segmentation and scene understanding, Loopy Belief Propagation [Murphy *et al.*, 1999] and GraphCut [Kolmogorov and Zabih, 2001] are usually employed. When the graphical model has a tree structure, Belief Propagation algorithms can obtain a global optimal, and when the variables are binary, GraphCut can obtain a global optimal. For other cases, the solutions are usually local optima. When efficiency, rather than accuracy, is emphasized, approximate algorithms such as Guided Filter [He *et al.*, 2010] can be used.

1.4.4 Data-Driven Approaches in Computer Vision

The main idea of data-driven approaches is that, given a query with some information missing (e.g., segmentation mask and semantic labels), propagating metadata from its Nearest Neighbors (NNs) in a pre-labeled dataset can be an effective way for reasoning such missing

information.

Application Scenarios. Depending on the type of metadata to be propagated, data-driven approaches have diverse applications in both fields of computer vision and computer graphics. For example, they benefit the traditional branches of computer vision, including segmentation [Kuettel *et al.*, 2012], object detection [Malisiewicz *et al.*, 2011a; Patterson *et al.*, 2008], and scene understanding [Liu *et al.*, 2011], based on the rapidly growing labeled dataset backed by crowd-sourcing. Data-driven approaches are also widely used in 3D applications, for example, 3D object detection and recognition [Song and Xiao, 2014], scene understanding [Satkin and Hebert, 2013; Satkin *et al.*, 2012; Wang *et al.*, 2013], action detection [Yuan *et al.*, 2009], and pose estimation [Shao *et al.*, 2012; Li *et al.*, 2014c, 2015].

Data-driven approaches also have well-known applications in low-level vision, albeit sometimes without a large-scale database. Some examples include the example-based super-resolution [Freeman *et al.*, 2002] searches for similar patches on an external database, which is among the best performed early work of super-resolution. The state-of-the-art image denoising algorithm BM3D [Dabov *et al.*, 2007], which uses multiple similar patches within an image to help clean a target patch, also uses the idea of data-driven approach to search within the query image. This usage of NN search within the query image is also extended to super-resolution [Glasner *et al.*, 2009].

Distance Metric and Index. Because visual features usually lie in a low-dimensional manifold, NN search based on Euclidean distance often results in suboptimal performance. Therefore, much attention has been drawn on learning a distance metric for NN search. Typical settings of distance metric learning involve learning a Mahalanobis Distance with supervision, such as class labels or pairwise similar/dissimilar constraints. However, the optimization objective often involves small distance of similar instances under the new metric [Xing *et al.*, 2002] and large margin between different classes [Weinberger and Saul, 2009], to sparsity [Lai *et al.*, 2011]. Other settings have also been explored, for instance, weighted Hamming Distance learning when combined with hashing for large-scale NN retrieval [Zhang *et al.*, 2013], and GMM as a weighted sum of Mahalanobis Distances [Zoran and Weiss, 2011; Wang *et al.*, 2014a].

To further improve speed when the database is large, various indexing methods have

been developed. One typical indexing structure to accelerate the NN search is tree structures, some examples of which are KDTrees [Bentley, 1975] and their variances [Philbin *et al.*, 2007b; Nister and Stewenius, 2006]. When the database scale further increases, hash tables [Datar *et al.*, 2004; Weiss *et al.*, 2009] or hamming distance reranking [Gong and Lazebnik, 2011] are often employed to accommodate the index for large amounts of data. Supervision can be optionally introduced to further improve the indexing quality [Wang *et al.*, 2012b; Liu *et al.*, 2012].

1.5 Dissertation Overview

This dissertation is organized as follows: Chapter 2 to 4 are devoted to the three key topics, shape retrieval, scene understanding, and pose estimation, respectively. In Chapter 5, we demonstrate the potential of the proposed framework in analyzing 2D data, showing how the MRF formulation also benefits low-level vision applications, such as deblurring and denoising. Finally, in Chapter 6, we discuss conclusions and future works.

Part I

3D Search, Understanding and Pose Recognition

Chapter 2

Cross-Domain 3D Search via Ensemble of Classifiers

2.1 Introduction

3D shape analysis and retrieval has been an important research topic in computer vision, graphics, and computational geometry. In the past two decades, extensive efforts have been made to design effective 3D shape retrieval algorithms [Tangelder and Veltkamp, 2008]. The existing work is mainly focused on two search scenarios, i.e., search by sketch [Funkhouser *et al.*, 2003; Zeleznik *et al.*, 2007] (Figure 2.1(a)) and search with CAD models as query input [Tangelder and Veltkamp, 2008] (Figure 2.1(b)). On the other hand, our target scenarios are based on the 3D models reconstructed from the newly emerged low-cost depth sensors, and thus bring a different setting of retrieval problems, i.e., *search with user captured models*, as illustrated in Figure 2.1(c).

Challenges. On one hand, this new setting promotes new applications, such as high-quality 3D scanning, manipulation, and printing, and on the other hand, has several unique properties. First, the user-captured models often contain a significant level of noise generated in either the capturing or reconstruction process. Second, the model generated in the uncontrolled environment is often incomplete because of occlusions or partial views. Hence, this new retrieval scenario with user-captured models results in significant challenges in various aspects of shape analysis and retrieval, which unfortunately, cannot be solved by

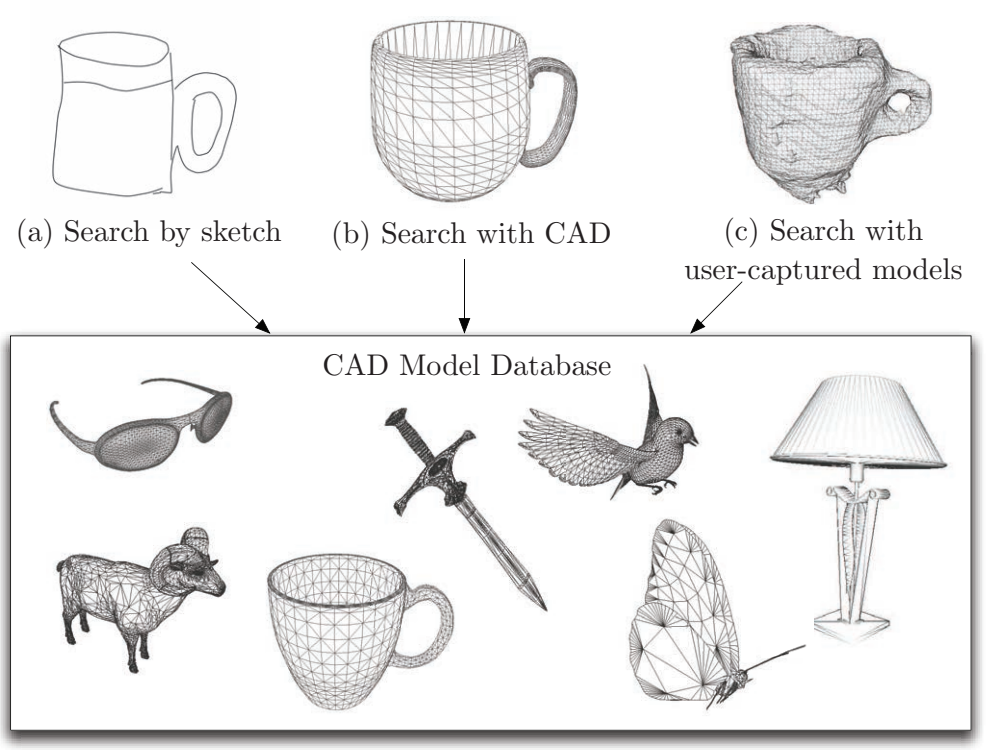


Figure 2.1: Different 3D shape retrieval scenarios: (a) search by sketch; (b) search with CAD; and (c) cross-domain search with user-captured models from low-cost sensors.

existing approaches or their simple extension.

More specifically, existing 3D shape retrieval approaches generally follow two popular frameworks, local feature matching with optional spatial verification [Funkhouser *et al.*, 2004; Johnson and Hebert, 1999; Zaharescu *et al.*, 2009; Sun *et al.*, 2009; Kokkinos *et al.*, 2012], and the Bag-of-Feature scheme [Bronstein *et al.*, 2011; Bronstein and Kokkinos, 2010], both of which require effective 3D local features. Although great progress has been made in 3D feature design, such as spin-image based descriptor [Johnson and Hebert, 1999], Mesh-DOG/MeshHOG [Zaharescu *et al.*, 2009], HKS [Sun *et al.*, 2009; Bronstein and Kokkinos, 2010], and ISC descriptor [Kokkinos *et al.*, 2012], these low-level shape features rely highly on the quality of the 3D models and tend to be sensitive to the model noise often encountered with low-cost depth sensors. Furthermore, neither of these two frameworks explicitly addresses the challenge of model incompleteness, resulting in degenerated performance in cross-domain 3D retrieval. For instance, a previous study shows that the Scale-Invariant

HKS (SIHKS) achieves a high retrieval accuracy with CAD model queries [Bronstein and Kokkinos, 2010], but degrades significantly for user-captured model queries [Machado *et al.*, 2013]. To address these issues, spatial consistency checking has been used in both 2D [Philbin *et al.*, 2007a] and 3D [Bronstein *et al.*, 2011] cases. However, the existing spatial consistency checking approaches, such as pairwise feature quantization [Bronstein *et al.*, 2011] and RANSAC [Philbin *et al.*, 2007a], are still insufficient for managing the severe challenges associated with user-generated low-quality partial models, as observed in [Machado *et al.*, 2013]. This is because spatial consistency checking is often heuristic, and merely acts as preprocessing or postprocessing without principled optimization that considers both feature similarity and spatial constraints.

Solutions. To address the above two challenges, we propose a robust and effective cross-domain shape retrieval approach by encoding local geometric structures in an MRF with a learned similarity measurement for robust feature matching. In particular, we build an MRF on the 3D points of the query model. Random forests are exploited to estimate approximate similarity efficiently, thus determining the unary potential. The geometric structures around each 3D point are embedded in the pairwise potential in a novel way, thus formulating the overall framework as an *RTF* [Jancsary *et al.*, 2012a] variant, as shown in Figure 2.2. Compared with earlier approaches, such as the Bag-of-Feature scheme and the existing partial matching algorithms, the proposed RTF approach utilizes rich geometric information (instead of traditional pairwise spatial relationship checking) to compensate ill effects from model noise and incompleteness. We evaluate our approach using two empirical study cases for cross-domain shape retrieval: a) the *Querying with Partial Models* dataset from the SHape REtrieval Contest (SHREC) '09 [Dutagaci *et al.*, 2009]; and b) the *Low-Cost Depth Sensing Camera* data from SHREC '13 [Machado *et al.*, 2013], both of which contain noisy 3D models reconstructed from low-cost depth sensors. The experimental results clearly demonstrate the superior performance of the proposed method, compared with several state-of-the-art 3D shape retrieval approaches.

This chapter is organized as follows. Section 2.2 presents a brief review of the related work. In Section 2.3, we describe the proposed *RTF*-based cross-domain shape retrieval method. More details on the potential design and efficient determination are illustrated in

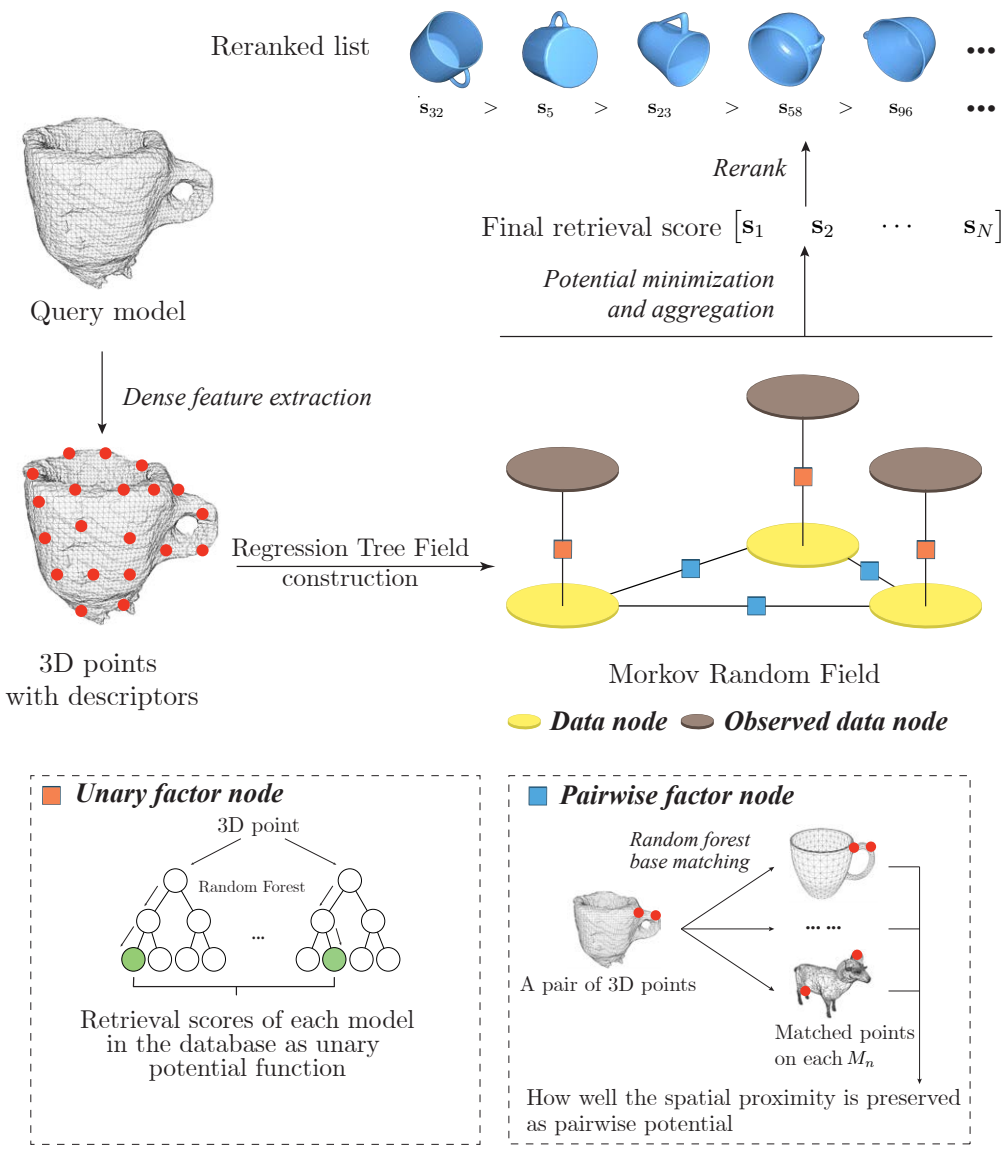


Figure 2.2: Framework of our cross-domain 3D shape retrieval based on RTFs.

Section 2.4 and Section 2.5. The experiment results and comparison studies are reported in Section 2.6, followed by our conclusions and discussions in Section 2.7.

2.2 Related Work

As discussed earlier, most 3D shape retrieval and search methods can be grouped into two major categories: a) search by sketch; b) search with CAD models. In the following paragraphs, we briefly review the representative approaches in each category. Detailed survey papers for shape retrial methods can be found in [Tangelder and Veltkamp, 2008; Li *et al.*, 2014a].

Search by Sketch. As shown in Figure 2.1 (a), first, a 2D projection of a 3D object is sketched, and then the sketch is used as the query example to find similar 3D objects in a shape database that often contains CAD 3D models. Because of the simplicity, various techniques have been developed to retrieve 3D models whose 2D images match the query sketch. For instance, Funkhouser *et al.* used a variant of the 3D sphere harmonics to develop a shape search engine that accepts sketches as queries [Funkhouser *et al.*, 2003]. Yoon *et al.* [2010] employed suggestive contours and diffusion tensor fields to improve robustness against the shape and pose variance that often occurs in user-sketched images. More recently, Shao *et al.* utilized a combination of contour-based representation and dense 2D matching to develop a robust approach that can perform partial matching between a query sketch and 3D models [Shao *et al.*, 2011]. In summary, the sketch-based framework is still a popular choice for 3D shape retrieval, and the influential SHREC specifically has a sketch-based contest track. A comprehensive review on this topic is available in [Li *et al.*, 2014a].

Search with CAD. The setting for searching with CAD often requires the query sample to be a complete or partial CAD model. There have been two popular directions with regard to this task. One is to design powerful 3D shape signatures that can capture the intrinsic geometric information of the CAD models, with the motivation that the query and database samples are essentially the same type of 3D models. To this end, various local features have been developed to describe the local geometry of 3D models, including MeshHoG as a 3D extension of the SIFT feature [Zaharescu *et al.*, 2009], HKS [Sun *et al.*, 2009; Bronstein and Kokkinos, 2010], ISC [Kokkinos *et al.*, 2012], and PFH [Rusu *et al.*, 2008, 2009]. Realizing the sensitivity to model noise for those local descriptors [Dutagaci *et al.*, 2009], researchers have proposed using high-level topological features [Tung and Matsuyama, 2012; Huang *et al.*, 2010], or aggregating low-level features to mid-level representations, such

as the extended Bag-of-Words model [Bronstein *et al.*, 2011; Pickup *et al.*, 2014] and graph correspondences [Wang *et al.*, 2012a]. Another direction is to map 3D models to a set of views, each of which can be represented using 2D descriptors [Chen *et al.*, 2003; Daras and Axenopoulos, 2010]. Sometimes Fourier-Mellin Transform is also used to provide scale invariance [Li *et al.*, 2008]. Although such multi-view shape descriptors can benefit from the discriminative power of mature 2D features, such as SIFT, they often overlook the important spatial information and suffer from expensive computational costs caused by matching a large number of views.

Finally, the recent rapid growth of consumer 3D models promotes the study of a new shape search scheme, i.e., search with consumer models, which explores cross-domain shape retrieval using models generated from low-cost depth sensors to query CAD model databases. Representative efforts include the "Querying with Partial Models" track in SHREC '09 [Dutagaci *et al.*, 2009] and "Low-Cost Depth Sensing Camera" track in SHREC '13 [Machado *et al.*, 2013]. The robotics community also had similar trials to utilize partial models to benefit grapsing [Goldfeder *et al.*, 2009]. However, the evaluation of existing shape retrieval methods on these two test benchmarks shows unsatisfactory performance because of the challenging issues of model noise and incompleteness. Therefore, we are motivated to design robust and accurate cross-domain shape retrieval techniques that can compensate the low quality of consumer models.

2.3 Framework

To address the partial matching problem for 3D shape retrieval using noisy models captured by low-cost depth sensors, we propose using a potential minimization formulation on MRF defined on the query model, where the potential functions are efficiently estimated through random forest prediction. This forms an RTF [Jancsary *et al.*, 2012a] variant, where the difference is that the potential is not learned fully jointly, resulting in more affordable training and testing time for larger-scale shape retrieval. In the following sections, we first introduce notations, and then illustrate the potential function design, followed by our efficient method to determine potential functions.

2.3.1 Background and Notations

Assume we are given a database that consists of N 3D mesh models $\{M_n\}_{n=1}^{N}$ with n as the model index, and a possibly incomplete and noisy user-captured model M_q as the query. The goal of a cross-domain shape retrieval engine is to return a ranked list of the 3D models in the database, such that the models ranked higher are more *similar* to the query.

In our formulation, we first construct MRF on M_q with an undirected graph representation $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Here, we specifically use the 3D points in M_q as the vertices $\mathcal{V} = \{v_i\}_{i=1}^{|\mathcal{V}|}$ with $|\mathcal{V}|$ being the cardinality, and the edges $\mathcal{E}$ are the edges of the query model M_q. For a 3D point v_i in M_q, we compute the Scale-Invariant Spin Image (SISI) [Darom and Keller, 2012] to represent the local geometry of a *3D patch* centered at v_i. Note that the size of the 3D patch (the scale) is determined by the SISI detector, which is scale invariant. The calculated 128-dimensional descriptor is used as the *observation* $\boldsymbol{x}_i$ of the MRF. In addition to the observation $\boldsymbol{x}_i$, each vertex is associated with a continuous vector $\boldsymbol{y}_i \in \mathbb{R}^N$ as the output variable conditioned on $\boldsymbol{x}$, where the n-th dimension $(\boldsymbol{y}_i)_n$ denotes the partial matching score between the i-th patch of the query model M_q and the n-th CAD model in the database. Compared with the standard MRF setting that often has a scalar as the output variable, in our MRF construction process, we have the output variable as an N-dimensional vector that indicates the partial similarity between the 3D patch and each CAD model in the database.

2.3.2 Formulation

Joint Distribution. With the undirected graph model $(\mathcal{V}, \mathcal{E})$ and the associated random variables $\boldsymbol{x} = \{\boldsymbol{x}_i\}_{i=1}^{N}$ and $\mathbf{y} = \{\boldsymbol{y}_i\}_{i=1}^{N}$, a probabilistic graphical model generally aims to maximize the joint distribution $\text{prob}(\boldsymbol{y} \mid \boldsymbol{x})$. With the assumption that the distribution obeys the Markov property with respect to the graph, the distribution can be further decomposed as the product of *unary* terms prob_u defined on each vertex and *pairwise* terms prob_p defined on each pair of connected vertices,

$$\text{prob}(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{v_i \in \mathcal{V}} \text{prob}_u(\boldsymbol{y}_i \mid \boldsymbol{x}_i)^{\lambda} \prod_{(v_i, v_j) \in \mathcal{E}} \text{prob}_p(\boldsymbol{y}_i, \boldsymbol{y}_j \mid \boldsymbol{x}_i, \boldsymbol{x}_j)^{1-\lambda}, \tag{2.1}$$

where λ is a parameter that balances the weights of the two terms. A graphical illustration of the nodes and potential functions is shown in Figure 2.2, where the red nodes show the unary terms, and the blue nodes show the pairwise terms.

With distribution, the goal is to find the retrieval scores $\boldsymbol{y}$ given the observation $\boldsymbol{x}$, such that the joint probability of prob($\boldsymbol{y} \mid \boldsymbol{x}$) is maximized. By designing the objective potential function to encode both the *shape similarity* and *geometric consistency*, we expect the inferred $\boldsymbol{y}$ to be a discriminant indicator for measuring the partial similarity between 3D patches and CAD models, while being robust to model noise and model incompleteness in the cross-domain shape retrieval task,

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmax}} \operatorname{prob}(\boldsymbol{y} \mid \boldsymbol{x}). \tag{2.2}$$

Potential Function. To avoid possible numerical problems in the optimization process, we introduce a potential function $\Psi\big(\boldsymbol{y} \mid \boldsymbol{x}\big)$ that is the negative logarithm of the joint probability, thus to be minimized,

$$\begin{aligned}
\boldsymbol{y}^* &= \underset{\boldsymbol{y}}{\operatorname{argmax}} \operatorname{prob}(\boldsymbol{y} \mid \boldsymbol{x}) \\
&= \underset{\boldsymbol{y}}{\operatorname{argmin}} \big(- \log \operatorname{prob}(\boldsymbol{y} \mid \boldsymbol{x})\big) \\
&= \underset{\boldsymbol{y}}{\operatorname{argmin}} \, \Psi\big(\boldsymbol{y} \mid \boldsymbol{x}\big) \\
&= \underset{\boldsymbol{y}}{\operatorname{argmin}} \Big(\lambda \sum_{v_i \in \mathcal{V}} \Psi_u(\boldsymbol{y}_i \mid \boldsymbol{x}_i) + (1-\lambda) \sum_{(v_i, v_j) \in \mathcal{E}} \Psi_p(\boldsymbol{y}_i, \boldsymbol{y}_j \mid \boldsymbol{x}_i, \boldsymbol{x}_j)\Big).
\end{aligned} \tag{2.3}$$

Similarly, the potential function is decomposed to two terms. The unary term $\Psi_u(\cdot) = -\log \operatorname{prob}_u(\cdot)$ provides a robust estimation of similarity scores that solely consider the local shape of the individual 3D patches, namely, *shape similarity*. The pairwise term $\Psi_p(\cdot) = -\log \operatorname{prob}_p(\cdot)$ aims to further refine the scores by enforcing *geometric consistency* among neighbor patches. Through combining these two terms, our method can manage cross-domain partial matching with the unary term, while being less sensitive to model noise and incompleteness because of the embedded geometric consistency in the pairwise term.

Inference. With our potential function design, which is described in the following two sections, the final objective $\Psi(\boldsymbol{y} \mid \boldsymbol{x})$ has the form of a quadratic function (but does

not necessarily have a positive semi-definite quadratic coefficient). Then, we use gradient descent to efficiently optimize for the local minima.

A natural concern of this formulation is the scalability, especially given that the optimization in Equation (2.3) may involve hundreds of variables with thousands of dimensions. However, as we show shortly, by exploring the sparsity of the problem and using discriminative random forests, inference on such MRFs can be extremely efficient and scalable to large-scale datasets.

2.4 Unary Potential: Search via Ensemble of Classifiers

2.4.1 Potential Design

In an MRF, the unary term of the joint distribution typically encourages the variable $\boldsymbol{y}_i$ to be consistent with the local observation $\boldsymbol{x}_i$. In particular, we use a Gaussian distribution to describe the probability of having a $\boldsymbol{y}_i$ conditioned on observing $\boldsymbol{x}_i$,

$$\text{prob}(\boldsymbol{y}_i \mid \boldsymbol{x}_i) \sim \mathcal{N}\big(\boldsymbol{y}_i \mid \boldsymbol{\mu}(\boldsymbol{x}_i), \Sigma\big), \tag{2.4}$$

where $\mathcal{N}(\cdot \mid \boldsymbol{\mu}, \Sigma)$ is a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance Σ, and $\boldsymbol{\mu}(\boldsymbol{x}_i)$ is a possibly noisy estimation of $\boldsymbol{y}_i$ purely from local observation $\boldsymbol{x}_i$. To simplify the problem, we assume all $\boldsymbol{y}_i$s share a diagonal covariance matrix $\Sigma = \sigma^2 I$. σ is a parameter of the model. Note that $\mathcal{N}$ is in an N dimensional space given $\boldsymbol{y}_i \in \mathbb{R}^N$.

Therefore, the unary potential, as the negative logarithm of the probability, penalizes the variable $\boldsymbol{y}_i$ that strays far from a local estimation, and has the form of quadratic loss,

$$\begin{aligned} \Psi_u(\boldsymbol{y}_i \mid \boldsymbol{x}_i) &= -\log \text{prob}(\boldsymbol{y}_i \mid \boldsymbol{x}_i) \\ &= \frac{1}{2\sigma^2}(\boldsymbol{y}_i - \boldsymbol{\mu}(\boldsymbol{x}_i))^T(\boldsymbol{y}_i - \boldsymbol{\mu}(\boldsymbol{x}_i)). \end{aligned} \tag{2.5}$$

Here, $\boldsymbol{\mu} : \mathbb{R}^{128} \to \mathbb{R}^N$, determining the mean of the Gaussian, is a function that estimates the similarity scores between a 3D patch in M_q and all the database models $M_n, n = 1, \cdots, N$. Traditional search engines usually employ a handcrafted (i.e., manual-designed) similarity measurement (e.g., Histogram Intersection Kernel) in addition to an optional indexing structure to compute such similarity scores; however, we employ random forests as an en-

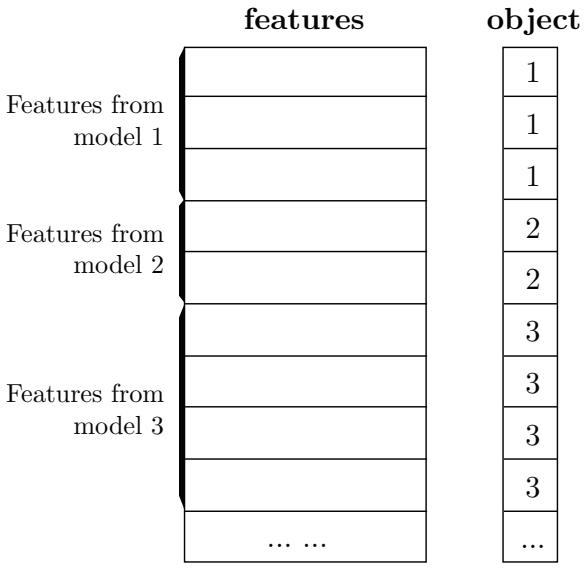


Figure 2.3: Collection of the training data for the random forest in the unary potential.

semble learning method in order to build an efficient indexing structure and simultaneously learn a similarity measurement.

2.4.2 Random Forest-based Mean Estimation

To achieve fast estimation of the similarity score $\{\boldsymbol{\mu}(\boldsymbol{x}_i)\}_{i=1}^{|\mathcal{V}|}$, the random forest method is used to perform a regression process divided into two stages, training and testing.

Training Phase. The training data contain all the extracted features of 3D patches from the database models as inputs, and the indices of the associated model as discrete responses, as shown in Figure 2.3. For the random forest, each decision tree is trained recursively using the standard information gain algorithm with the linear classifiers for data division. Finally, each leaf node in a decision tree receives a score vector $\boldsymbol{p}_l = [p_{l1}, \cdots, p_{ln}, \cdots, p_{lN}]$ that measures the frequencies of the patches of the specific 3D model that falls in that leaf, with each dimension computed as

$$p_{ln} = \frac{\#\text{ of training examples from model } n}{\#\text{ of training examples}}, n = 1, \cdots, N. \tag{2.6}$$

Here, $l = 1, \cdots, L$ is the index of the decision tree with L being the number of decision trees in the random forest.

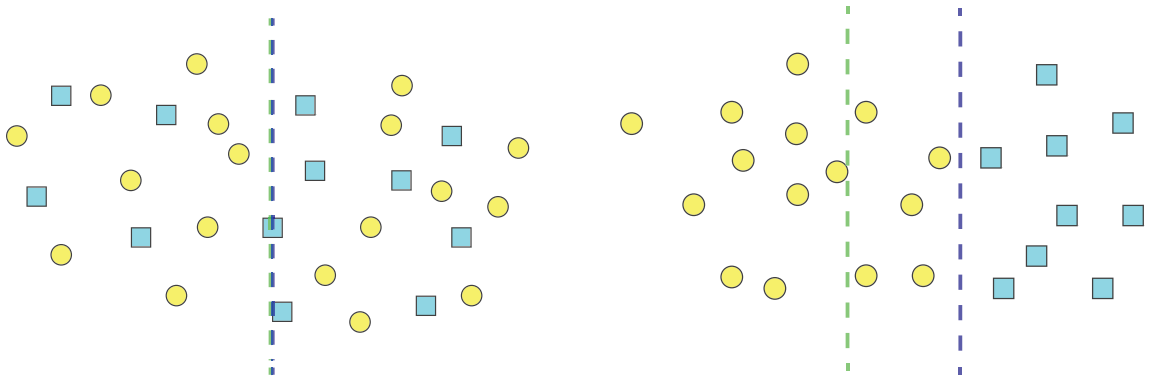
Testing Phase. Given a feature vector $\boldsymbol{x}_i$ from a 3D patch in the query model, we first

conduct an examination from root nodes to leaf nodes through all the decision trees in the trained random forest. The approximate estimation of the similarity scores between a patch in the query model $\boldsymbol{x}_i$ and the CAD models are computed via averaging the recomputed score $\boldsymbol{p}_l$ on the retrieved leaf nodes as $\boldsymbol{\mu}_i = \frac{1}{L}\sum_{l=1}^{L}\boldsymbol{p}_l$. Compared with the traditional method for computing the similarity score by performing exhaustive matching between features, the regression method utilizes a discriminative decision model that can capture the underlying distributions of the features, resulting in more robust estimation against model noise. In addition, the random forest method also benefits from the computational efficiency with a sub-linear time complexity that can be accelerated further for managing large-scale applications through easy parallel implementations.

2.4.3 Discussion: a Retrieval Perspective

Although the random forest design is inspired from a supervised learning perspective, it can also be justified from a unsupervised retrieval perspective. On one hand, it is an indexing structure. On the other hand, it learns a similarity measurement to some extent. In this subsection, we first compare the random forest with traditional indexing approaches, such as KDTrees and content-based hashing techniques, and then discuss its functions in indexing and similarity measurement learning.

Relationship with KDTrees. KDTree [Bentley, 1975] is a data structure that supports efficient approximate NN(s) search in a vector space. Because KDTrees are capable of effectively reducing the number of candidates to feed into the rerank module, they are widely used as indexing structures in search engines [Nister and Stewenius, 2006; Philbin *et al.*, 2007b]. The framework of our approach is similar to KDTree, where both approaches involve a tree structure and use hyperplanes to conduct subspace division. However, the division criteria are different. KDTree considers every feature point equally and only relies on the point density to determine the division hyperplane. On the contrary, random forests take advantage of the additional information that indicates the model from where each feature point originates. That is, random forests always attempt to divide the feature space such that feature points from different 3D models are separated. When the feature point distribution is completely independent from its originating 3D model, random forests de-



(a) (b)

Figure 2.4: Behavior comparison between a KDTree and the proposal random forest-based approach. The circles and rectangles show the feature points in a 2D vector space, different colors/shapes of dots are from different 3D models. The green line is the expected division hyperplane from KDTree, and the purple line shows the expected division hyperplane from the random forest approach. (a) shows one extreme case where the feature points are completely random, when two approaches obtain the same result; and (b) shows another extreme where the feature points from different models can be separated perfectly, when the random forest approach achieves devision with better discriminative ability.

generate to KDTrees, and can only divide subspaces by density of feature points. However, when feature points contain meaningful information of the model from where they originate, which is usually the case in reality because this is the design goal of features, random forests generate better space division than KDTrees in terms of estimating the similarity between the input feature point and each CAD model, and this is the goal for $\boldsymbol{\mu}(\boldsymbol{x}_i)$. Figure 2.4 shows the behavior comparison between KDTrees and the proposed random forest in two extreme cases.

From another perspective, KDTree is designed to find the NNs in terms of point-to-point distances. However, random forests can capture point-to-3D-model distances because they have additional supervision as to which feature points are from which model.

Relationship with Content-based Hashing. Similar to KDTrees, content-based hashing is another method for performing NN search in a vector space. Given the feature point $\boldsymbol{x} \in \mathbb{R}^d$, n hash functions $\{f_i : \mathbb{R}^d \rightarrow \mathbb{B}\}_{i=1}^n$ are applied to $\boldsymbol{x}$, resulting in a binary string

$\boldsymbol{y} \in \mathbb{B}^n$ that can be interpreted as an integer. Therefore, this hashing process essentially maps all the feature points into many buckets. Given a query feature point $\boldsymbol{x}$, the instances (feature points in our setting) that fall into the same bucket, or within a certain Hamming distance, are treated as the approximate NNs. The advantages of content-based hashing are in both space and time efficiency. Because of the high expectation of speed, the hashing functions are typically linear projections of the form $f(\boldsymbol{x}) = \mathbf{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b)$.

This is similar to the our random forest approach, especially considering the process of traversing from the root to the leaf in each decision tree, where we are also perform a series of binary testing $\mathbf{sgn}(\boldsymbol{w}^T\boldsymbol{x} + b)$, and use the results to determine whether to proceed to the left or right child. From this perspective, each decision tree essentially divides the feature space into many buckets using a series of linear testing. Moreover, similar to content-based hashing, the actual binary functions are also learned from the distribution of the feature points with supervision of the model from where they originate. The difference is that the hash functions used in content-based hashing are usually independent of each other, whereas in decision trees, one has to know the testing results of one hash function before determining the next hash function to use. Considering that a random forest generally contains multiple decision trees, if there are n decision trees with d levels of nodes in the forest, the process for random forest testing can also be viewed as d stages of binary testing, each of which uses n hash functions determined from the results of the previous test.

Random Forest as an Index. An important component of a visual search engine is the index that maps the input model or features to a limited set of candidate models; and therefore, the time-consuming reranking does not need to be performed on the entire database. Although we do not have an explicit indexing module in the pipeline, the proposed random forest acts as the index, given that forest training aims to maximize the purity of the children nodes (i.e., minimize the entropy of the distribution over the source models). This optimization objective effectively makes the distribution in Equation (2.6) become more sparse with a depth increase in the training process, making the leaf nodes contain only the features from a few 3D models. Therefore, during the testing stage, when we go from the root to the leaf to obtain the similarity scores, most vector entries are actually zero, thus filtering most models and allowing us only to do reranking based on the few

non-zero entries.

A Similarity Measurement. During the process of parameter tuning (detailed later), we observed that when there is one or extremely few trees in the random forest, the CAD model that shows the largest similarity score is usually the most similar to the query, but for other CAD models, a larger similarity score does not necessarily mean it is more similar to the query. This is reasonable because a single decision tree as a classifier only attempts to ensure that the most similar class has the largest score, with no guarantee that the second most similar class will have the second largest score, or similar orders. Therefore, it does not fit the retrieval scenario well. However, this is improved when the number of trees in the forest increases, i.e., with more trees, the returned similarity scores become a better indicator of the human perceived similarity.

This can be interpreted as the ensemble of a set of decision trees. Whereas each decision tree performs a "hard" classification, when multiple, and even a large number of, trees trained from different portions of the data are combined, the comprehensive probability can provide a "soft" version of the similarity measurement.

2.5 Pairwise Potential: Spatial Verification

2.5.1 Potential Design

Although local matching, as introduced in the last section, provides capabilities for managing partial matching, purely relying on local matching to determine the similarity score may be problematic. This is because the local 3D patches are sometimes ambiguous and confusing, resulting in uninformative similarity scores. For example, for a local surface that is simply planar, it is difficult to determine whether it is from a table or bookshelf, despite the fact that a table is fairly different from a bookshelf. Therefore, purely relying on local patch matching does not take full advantage of the information available and may result in unnecessarily ambiguous similarity scores. This problem is even more severe given that low-cost depth sensors also bring considerable noise.

2D search systems also have this problem, and research has shown that spatial consistency is critical to help resolve such ambiguity [Philbin *et al.*, 2007b; Zhang *et al.*, 2011;

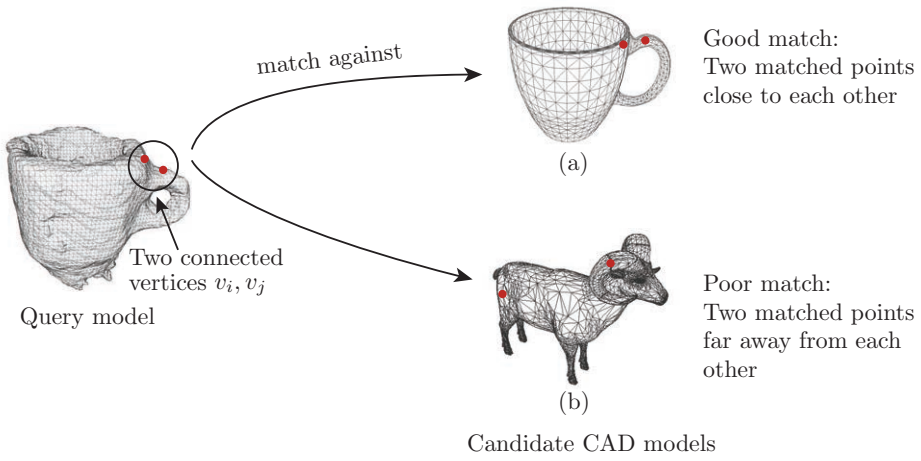


Figure 2.5: Intuition of the pairwise potential design.

Sadeghi and Farhadi, 2011]. This is especially useful when two models are locally similar, but actually different in a global view, as demonstrated by our previous table and bookshelf example. The key to taking advantage of spatial consistency is to find some indicator of matching quality across different local regions. Whereas traditional 2D search engines use random sampling across local regions to check whether they share consistent geometric transforms (e.g., RANSAC), we take advantage of the nice properties of MRFs to embed such global check in the pairwise potential. More specifically, in the potential minimization process of MRFs, although every single pairwise term in the potential only involves two adjacent nodes (or a local clique), the optimizer can find a joint minimal of all pairwise terms, given that they are combined in the objective. On one hand, this simplifies the problem from global consistency checking to merely pairwise consistency checking; on the other hand, this is more efficient, backed by the decade-long development of the probabilistic graphical model. Here, we start from some intuitive examples to derive the actual pairwise potential design.

Intuition. As a key difference from standard MRF formulation, the pairwise potential in our approach utilizes all the models in the database to help embed the local geometric structures. In Figure 2.5, two examples are demonstrated to illustrated the intuition. Assume the query is a mug, and note the two close-by vertices on the mug body and handle, shown as two red dots in the figure. For some well matched models, for example, another mug as shown in Figure 2.5(a), if we attempt to find the NN of the two vertices in the CAD

model, it is likely for them to be matched to vertices also on the body and handle respectively, and therefore, the two matched points in the CAD model are also close by. However, if the CAD model is not similar to the query, as shown in Figure 2.5(b), when we attempt to find the most similar vertices of the two red points, they are likely to lie in random parts of the CAD model, and thus are far from each other. In this specific example, one is on the head, and one is on the leg of the sheep. Subsequently, this can act as an indicator of whether two nearby matches are consistent for a model in the database. Furthermore, by expanding this intuition to all CAD models, we can use all the dimensions of the inferred similarity scores $\boldsymbol{y}_i, \boldsymbol{y}_j$ to obtain an evaluation for the consistency.

Potential Design. Extending the example above to multiple CAD models, for a pair of neighbor vertices $(v_i, v_j) \in \mathcal{E}$ from the query model M_q, the distance of their matched (NN) vertices $v_i^{n\prime}, v_j^{n\prime}$ in a CAD model M_n is an indicator of the spatial consistency between M_q and M_n. A small distance $\|v_i^{n\prime} - v_j^{n\prime}\|_2$ indicates better spatial consistency between M_q and M_n, which means the corresponding retrieval scores $(\boldsymbol{y}_i)_n$ and $(\boldsymbol{y}_j)_n$ should be higher. Moreover, a larger distance indicates that the spatial proximity of the neighbor vertices (v_i, v_j) is violated in the process of matching against the model M_n, and therefore, M_n is not a spatially consistent candidate to the query, which would penalize $(\boldsymbol{y}_i)_n$ and $(\boldsymbol{y}_j)_n$. Therefore, we define the pairwise term as

$$\Psi_p(\boldsymbol{y}_i, \boldsymbol{y}_j | \boldsymbol{x}_i, \boldsymbol{x}_j) = \sum_{n=1}^{N} \|\boldsymbol{v}_i^{n\prime}(\boldsymbol{x}_i) - \boldsymbol{v}_j^{n\prime}(\boldsymbol{x}_j)\|_2 \cdot (\boldsymbol{y}_i)_n (\boldsymbol{y}_j)_n. \tag{2.7}$$

Recall that $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are the retrieval scores of the 3D patches $\boldsymbol{x}_i, \boldsymbol{x}_j$ in the query against all database models, and therefore, lie in an N dimensional space. Furthermore, $\boldsymbol{v}_i^{n\prime}(\boldsymbol{x}_i)$ and $\boldsymbol{v}_j^{n\prime}(\boldsymbol{x}_j)$ are the 3D coordinates of the matched vertices in the model M_n that corresponds to the 3D patches $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively. Thus, $\|\boldsymbol{v}_i^{n\prime}(\boldsymbol{x}_i) - \boldsymbol{v}_j^{n\prime}(\boldsymbol{x}_j)\|_2$ measures the Euclidean distance between two matched vertices in the model M_n. Recall that the potential function is to be minimized, and therefore, this design places less penalty on the retrieval scores for the vertex pairs $\boldsymbol{v}_i^{n\prime}(\boldsymbol{x}_i)$ and $\boldsymbol{v}_j^{n\prime}(\boldsymbol{x}_j)$ with a small distance. On the contrary, if the matched vertices $\boldsymbol{v}_i^{n\prime}(\boldsymbol{x}_i)$ and $\boldsymbol{v}_j^{n\prime}(\boldsymbol{x}_j)$ are not spatially close to each other, their similarity scores $(\boldsymbol{y}_i)_n, (\boldsymbol{y}_j)_n$ to the query patches are suppressed with a larger coefficient. It is also worth noting that each model in the database is checked separately in the pairwise term

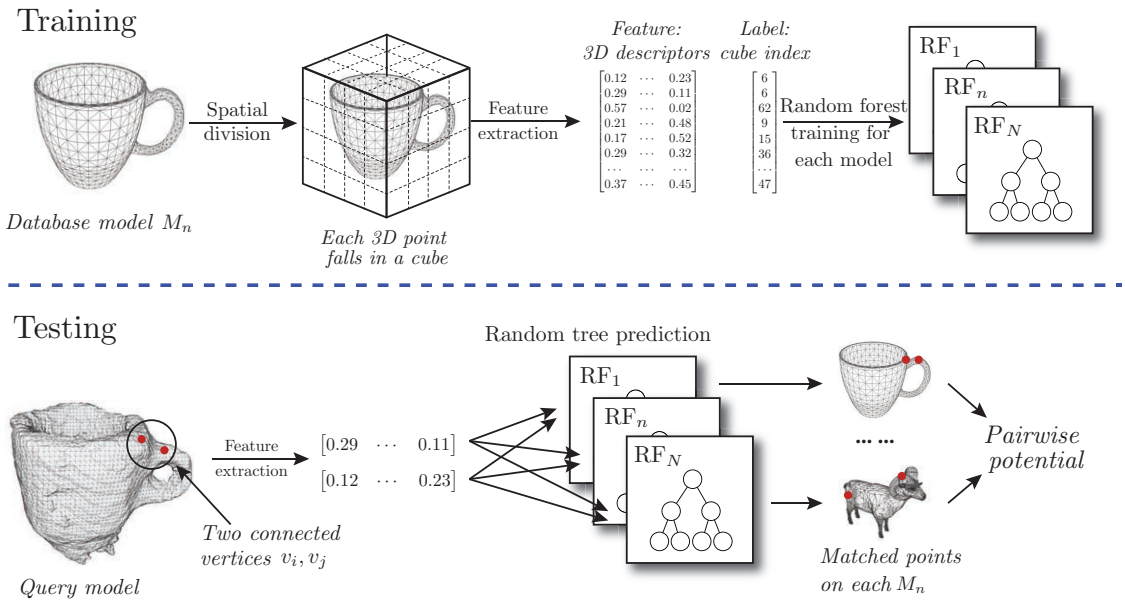


Figure 2.6: Illustration of the efficient estimation of the pairwise potential term using random forests.

computation, which does not require any pose estimation or calibration, and thus it is more reliable against sensor noise and incomplete models.

In practice, the straightforward local feature that matches to find the corresponding vertices $\boldsymbol{v}_i^n\prime, \boldsymbol{v}_j^n\prime$ is unreliable under sensor noise. Therefore, we further use random forests to robustly determine the vertex correspondences, as introduced below.

2.5.2 Random Forest-based Efficient Matching

To estimate the pairwise potential term, it is necessary to find the best matched patch in a CAD model M_n for a query patch $\boldsymbol{x}_i$ to derive the corresponding vertex $v_i^n\prime$. Here, we propose to again employ random forests to perform fast matching in a classification manner, with the framework shown in Figure 2.6. In particular, we look at the 3D bounding box on each CAD model and partition the model into $d \times d \times d$ voxels, each of which contains a set of 3D vertices. Here, d is often set as a small value, such as $d = 4$ in our experiments. Then, we use those partitioned vertices as training data to build a random forest *for each model* with the leaf node generating the prediction of the voxel into which the query patch will fall. The random forests are trained in the same manner using the information-gain based

algorithm. Then, for a given query patch $\boldsymbol{x}_i$, we can quickly retrieve a small voxel in M_n that could contain a similar patch, and adopt the center of that voxel as the matched vertex $v_i^{n\prime}$. Provided that random forests empirically contribute a testing time of $\mathcal{O}(\log C)$, where C is the class number, the total time cost for matching a query patch with all the CAD models is $\mathcal{O}(N \log d)$, which is significantly faster than exhaustive matching with the time cost at $\mathcal{O}(N|\mathcal{V}|)$. In our experiments, we observe that such random forest-based matching achieves fast, yet accurate, matching results in practice. For instance, for a database with 720 models and a query with 500 points, less than 0.2 seconds on a modern i7 CPU is required to accomplish the matching procedure, where 80% of the matched results are consistent with the nearest vertices.

2.5.3 Inference of the MRF

Given the exact forms of the unary potential from Equation (2.5) and pairwise potential from Equation (2.7), we can rewrite the objective function into a compact matrix form. Define a matrix $\mathbf{V} \in \mathbb{R}^{N\times N}$ with its element V_{ij} calculated as $V_{ij} = \sum_{n=1}^{N} \left(\|\boldsymbol{v}_i^{n\prime} - \boldsymbol{v}_j^{n\prime}\|_2\right)$. Then, the pairwise potential can be written as

$$\Psi_p(\boldsymbol{y}_i, \boldsymbol{y}_j) = \boldsymbol{y}_i^T V_{ij} \boldsymbol{y}_j. \tag{2.8}$$

Hence, the overall pairwise potential is represented as

$$\sum_{(v_i,v_j)\in\mathcal{E}} \Psi_p(\boldsymbol{y}_i, \boldsymbol{y}_j) = \boldsymbol{y}^T V \boldsymbol{y}, \tag{2.9}$$

where $\boldsymbol{y} \in \mathbb{R}^{N|\mathcal{V}|}$ is the concatenation of all the column vectors $\boldsymbol{y}_i$, and V is a blockwise matrix with $|\mathcal{V}| \times |\mathcal{V}|$ blocks, each as V_{ij}. Substituting Ψ_u and Ψ_p in Equation (2.3) by the above derivations, we can derive the objective potential function in quadratic form as

$$\boldsymbol{y}^* = \underset{\boldsymbol{y}}{\operatorname{argmin}} \Psi(\boldsymbol{y} \mid \boldsymbol{x}) = \underset{\boldsymbol{y}}{\operatorname{argmin}} \quad \left(\frac{1}{2}\boldsymbol{y}^T H \boldsymbol{y} - \boldsymbol{c}^T \boldsymbol{y}\right), \tag{2.10}$$

where we have

$$\begin{aligned} H &= \lambda I + (1-\lambda)V \\ \boldsymbol{c} &= \lambda \boldsymbol{\mu} = \lambda f(\boldsymbol{x}). \end{aligned}$$

$\boldsymbol{y}$ and $\boldsymbol{\mu}$ are the column concatenation of $\boldsymbol{y}_i$ and $\boldsymbol{\mu}_i$ (c.f. Unary Potential above), respectively. However, the above quadratic problem does not have to be convex because H might not be positive semi-definite in practice. Therefore, we use the stationary point that provides the solution to the linear system $H\boldsymbol{y} = \boldsymbol{c}$ as an approximate solution. Because H is high dimensional, it is computationally prohibitive to directly compute the analytical solution to the linear system. Following RTFs [Jancsary *et al.*, 2012a], we use the conjugate gradient descent to obtain the solution efficiently in an iterative manner. In addition, because H is often sparse, the inference procedure is fairly efficient, which usually ends in ten iterations within 0.1 seconds on a desktop i7 CPU.

After computing the locally optimal solution $\boldsymbol{y}^* = \{(\boldsymbol{y}_i^*)_n\}$ $(1 \leq i \leq |\mathcal{V}|, 1 \leq n \leq N)$, we can derive the final ranking score to a query model as $\boldsymbol{s}_n = \sum_{i=1}^{|\mathcal{V}|}(\boldsymbol{y}_i^*)_n$, which is used for reranking.

In summary, we formulate the cross-domain search as a potential minimization problem on an MRF, whose potential functions are dynamically determined from random forests, thus forming an RTF variant. The two challenges of sensor noise and model incompleteness are resolved with the random forest-based similarity computation and pairwise geometric consistency checking, which is demonstrated quantitatively and qualitatively with experiments on real consumer models.

2.6 Experiments

In order to provide a quantitative performance evaluation of the proposed cross-domain shape retrieval approach, we conduct experiments on two benchmarks from the well-known SHape REtrieval Contest (SHREC). The first dataset is from Querying with the Partial Models track in SHREC '09 [Dutagaci *et al.*, 2009], which consists of incomplete and noisy models captured from desktop 3D scanners. The second dataset contains query 3D models generated by the Microsoft Kinect sensors used in the SHREC '13 [Machado *et al.*, 2013]. Below, we describe the details of the datasets, experiment settings, and evaluation results.

(a) (b) (c)

Figure 2.7: Illustration of the physical objects and the user-captured 3D models from the benchmark dataset: (a) physical objects used to generate the 3D models for the SHREC '09 dataset (figure cited from [Dutagaci *et al.*, 2009]); (b) an incomplete query model of the SHREC '09 dataset captured by a 3D desktop scanner; and (c) a noisy and low-resolution query model of the SHREC '13 dataset captured by a Microsoft Kinect sensor.

2.6.1 Datasets

The dataset from Querying with Partial Models track in SHREC '09 is specifically designed to explore the frontier of 3D shape retrieval techniques in managing incomplete and possibly noisy query samples. Such dataset consists of a set of 720 high-quality CAD models as the database for querying. The CAD models are from 40 categories, such as *bird, fish, mug*, and *car* with 18 models for each category. In addition, it has two query sets, including a set of high-quality incomplete samples cropped from CAD models, and a set of user-captured models obtained with a desktop 3D scanner. Here, we use the user-captured query set because it represents well the common challenges of cross-domain shape retrieval, such as surface noise and model incompleteness caused by self-occlusion. Examples of the physical objects used to capture the models are shown in Figure 2.7 (a), with the user-captured models shown in Figure 2.7 (b).

Another popular low-cost depth sensor is Microsoft Kinect, which is used to build 3D models by employing multiple range images [Izadi *et al.*, 2011]. Compared with single range image-based 3D models, such as the SHREC '09 dataset, the Kinect-captured models tend to be noisier because of non-smooth surfaces, and also have lower resolutions. In our experiments, we adopt the dataset from the Low-Cost Depth Sensing Camera track of the

SHREC '13 [Machado *et al.*, 2013], which contains a total of 192 Kinect models. Note that the original test in the SHREC '13 is designed for 3D retrieval with both queries and database containing Kinect models. To test cross-domain performance, here we use the CAD models from the SHREC '09 dataset as the database, and use the 192 Kinect models from the SHREC '13 as the query set. Figure 2.7 (c) demonstrates an example of the used Kinect models.

2.6.2 Experiment settings

We conduct two types of empirical studies. On the SHREC '09 dataset, we provide quantitative performance evaluations and compare several representative 3D shape retrieval methods. Because the query dataset from the SHREC '13 has no ground truth category information, we simply design qualitative evaluation by demonstrating the retrieval results.

For the quantitative comparison, we compare popular methods on CAD model retrieval and several approaches that achieve state-of-the-art performance in the cross-domain contest track, including one 3D feature-based approach [Bronstein *et al.*, 2011] and two 2D view-based approaches [Dutagaci *et al.*, 2009]. For our method, we also evaluate a variant that only uses the unary term without the pair-wise term of spatial consistency. Below, we briefly describe the settings for each compared method.

- **Shape Google** [Bronstein *et al.*, 2011]: we implement the Shape Google approach [Bronstein *et al.*, 2011], a shape retrieval approach for CAD models. For fair comparison, we use the same SISI feature [Darom and Keller, 2012] as in our approach. A codebook of size $10,000$ is built using the Approximate KMeans method [Philbin *et al.*, 2007a].

- **CMVD-Depth** [Dutagaci *et al.*, 2009]: achieving the best precision-recall in the SHREC '09 contest, the Compact Multi-View Descriptor (CMVD) extracts global 2D descriptors from the depth maps rendered from different views. The retrieval ranking is derived based on the minimum ℓ_1 distances between the signatures of the query and that of the database model.

- **CMVD-Binary** [Dutagaci *et al.*, 2009]: **CMVD-Binary** is another approach with strong performance on the consumer model retrieval task in the SHREC '09 [Dutagaci

et al., 2009]. Unlike the **CMVD-Depth** method that renders depth images, **CMVD-Binary** renders binary masks of the model in order to achieve computational efficiency and robustness against model noise.

- **BF-GridSIFT** [Ohbuchi *et al.*, 2008]: as a state-of-the-art approach for both generic and user-captured model 3D shape retrieval, **BF-GridSIFT** first performs pose normalization of the models, and then renders depth maps from uniformly distributed views. Then, the Bag-of-Feature scheme is employed to aggregate the extracted 2D dense SIFT descriptors. In the retrieval stage, KL-Divergence is used to compute a non-symmetric distance between the query sample and a database model.

- **RTF-Unary**: this is a simplified version of the proposed RTF-based approach that only considers the unary term by setting $\lambda = 1$ in Equation (2.3) and Equation (2.10). Note that the **RTF-Unary** approach is equivalent to only using the computed similarity score from partial matching with random forests to perform ranking.

- **RTF**: The proposed RTF approach. In the implementation of both RTF-based methods, i.e., **RTF-Unary** and **RTF**, we use 128 trees with depth 12 in the unary term. For the pairwise term in the **RTF** method, we apply bounding boxes to partition each model into 64 voxels ($d = 4$), and build a random forest with four trees of height 6. The coefficients that balance the two potential terms are set as $\lambda = 0.9$ uniformly across all the experiments.

To measure performance, we adopt the semantic category information to evaluate the retrieved results. In particular, we treat the models from the same category as *relevant* and the models from different categories as *irrelevant* in order to compute two quantitative measurements as the evaluation protocols. First, we compute the Mean Average Precision (MeanAP) that measures the average precision scores across all queries [Yilmaz and Aslam, 2006]. Second, we employ the popular evaluation criteria, the Normalized Discounted Cumulative Gain (NDCG) defined as

$$\mathrm{NDCG} = \frac{\sum_{n=1}^{N} \frac{\mathrm{Relevant}_n}{\log_2(n+1)}}{\sum_{n=1}^{N} \frac{1}{\log_2(n+1)}},$$

Approach	MeanAP	NDCG
Shape Google	0.188	0.506
CMVD-Depth	0.193	0.521
CMVD-Binary	0.203	0.511
BF-GridSIFT	0.219	0.532
RTF-Unary	0.281	0.591
RTF	**0.315**	**0.611**

Table 2.1: The computed MeanAP and NDCG on SHREC '09.

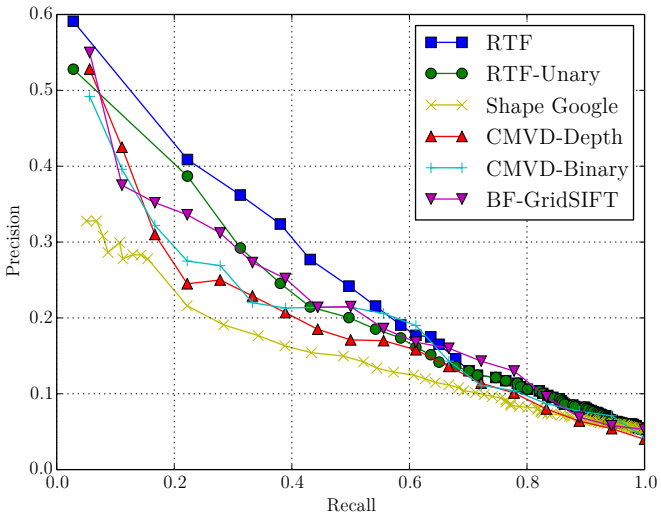


Figure 2.8: Precision-recall curves for the approaches evaluated on the SHREC '09 dataset.

where Relevant_n is one when the nth sample is relevant to the query; otherwise, it is zero. By assigning larger weights to the results ranked higher, NDCG favors high-ranked relevant instances because they are more important for user experience. Below we report the results for both quantitative and qualitative evaluations.

2.6.3 Results

For the results on the SHREC '09 dataset, we report MeanAP and NDCG for all the methods compared in Table 2.1, with the performance for **CMVD-Depth**, **CMVD-Binary**, and **BF-GridSIFT** cited from [Dutagaci *et al.*, 2009].

It is clear that the proposed **RTF** method achieves the highest performance among all the compared methods. Note that the pairwise term results in significant performance improvement compared with **RTF-Unary** – a 12% gain in MeanAP. Although only exploring a single unary potential term, the **RTF-Unary** method achieves the second best performance in the SHREC '09 dataset. This is because the unary potential term derives cross-domain partial matching-based similarity retrieval, which is suitable for addressing the model incompleteness and noise issues on this dataset. Note that the methods that adopt multiple views, such as **BF-GridSIFT** and **CMVD-Depth**, perform stronger than

the single-view method **Shape Google**, which might also be the result of the model incompleteness issue on this data. In addition, we plot the precision-recall curves for all the methods in Figure 2.8, which further confirms the clear performance gain of the proposed method. Finally, in terms of computational cost, on a desktop PC with an i7 3.0 GHz CPU, the proposed method requires less than 1 second to perform the retrieval process in a database that contains 720 objects, which is significantly faster than other compared methods.

On the SHREC '13 dataset, we present the qualitative evaluation by demonstrating the top retrieved 3D models in Figure 2.9. In particular, we compare the results of the two variants of our method, i.e., **RTF** and **RTF-Unary**, and a strong competitor method, **BF-GridSIFT**. From Figure 2.9, it is clear that the **RTF** method outperforms the other two methods by generating semantically consistent 3D models both for simple objects, such as *mugs*, and complicated objects, such as *planes*.

2.7 Summary and Future Work

This chapter illustrated the problem of 3D shape retrieval under the setting of queries captured using low-cost depth sensors and a database that contains conventional high-quality CAD models. To resolve challenging issues such as noise and incompleteness of the user-captured models, we presented an approach for performing retrieval with a principled optimization framework. Intuition is embedded in a formulation of an RTF, resulting in a comprehensive minimization problem of the MRF potential function, which contains a unary term that measures the similarity of cross-domain partial matching and a pairwise term with embedded geometric consistency. Both of these two terms were determined using efficient random forest algorithms. We conducted extensive empirical studies on two benchmark datasets from the well-known SHREC. The results clearly corroborated the superior performance of the proposed method, compared with other representative shape retrieval algorithms. One of our future directions is to introduce online random forest training algorithms [Ben-Haim and Tom-Tov, 2010] in order to avoid the necessity of retraining when adding new models, and also to extend the proposed method to explore

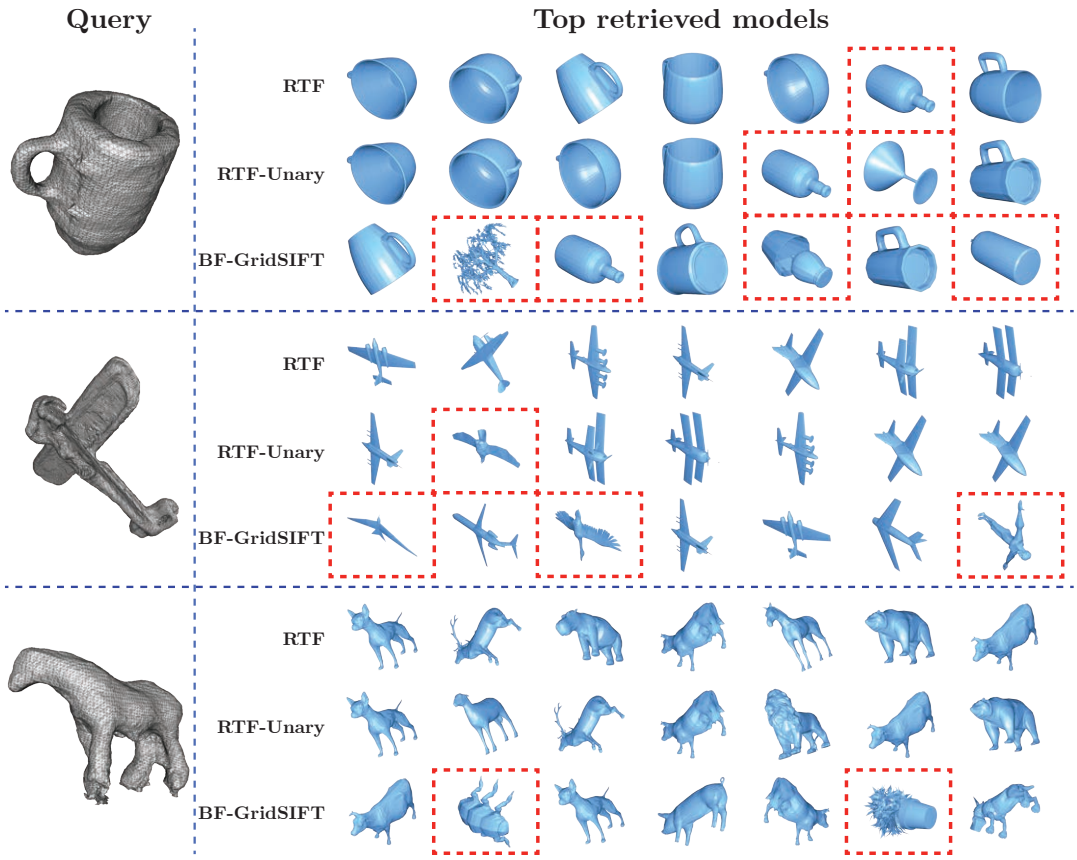


Figure 2.9: Examples of the top results for cross-domain shape retrieval, where the database contains CAD models from the SHREC '09 dataset, and the query models are the user-captured models from Microsoft Kinect. From top to bottom, the query models are *Mug*, *Airplane*, and *Quadruped*. For each query, the three rows show the results from **RTF**, **RTF-Unary**, and **BF-GridSIFT**, respectively. The results highlighted by red bounding boxes indicate the irrelevant 3D models.

cross-domain 3D shape recognition and classification.

Chapter 3

3D Scene Understanding

3.1 Introduction

In addition to content based shape retrieval, another important problem of 3D shape analysis is to infer point-wise semantic labels with an input 3D point cloud, i.e., the problem of semantic 3D scene understanding. Here, by a *point cloud*, we specifically refer to a set of 3D points, each of which carries its 3D coordinates and RGB colors. Previously, point clouds could only be generated using expensive LiDAR sensors; however, the increasing popularity of low-cost 3D sensors has made point clouds accessible to massive consumers supported with RGBD registration algorithms, such as RGBD Simultaneous Localization and Mapping (SLAM) [Endres *et al.*, 2012]. Given the ease of collecting 3D point clouds, a solution to scene understanding based on point clouds, especially those collected with low-cost sensors, may result in a breakthrough in a wide variety of computer vision and robotics applications, with great potential in human-computer interface, object manipulation in robotics, and even exciting applications such as self-driving cars and semantic-aware augmented reality.

Challenges. Although important, semantic labeling of 3D point clouds is not an easy task. Following 2D semantic labeling, the state-of-the-art solutions [Koppula *et al.*, 2011; Anand *et al.*, 2012; Xiong *et al.*, 2011; Nan *et al.*, 2012; Kalogerakis *et al.*, 2010; Lai and Fox, 2010] train point-wise label classifiers based on visual and 3D geometric features, and optionally refine them with spatial contexts. However, it is difficult to directly extend

such approaches to 3D data for several reasons. First, the scalability to large amount of training data is limited because of the saturation of the classifier's capability of digesting training data. Second, it is difficult to design effective 3D features while a 3D feature variant to rotation, translation, scaling, and illumination, such as those performed by 2D SIFT, continues to be missing. Meanwhile, it is also difficult to extend 2D features to 3D given that the critical operations for 2D feature extraction, such as convolution, are no longer valid for point clouds.

Another challenge is the fact that semantic labeling is a supervised problem that requires human labeled databases. Labeled training data has been shown to be a key factor towards successful 2D image labeling [Russell *et al.*, 2008; Deng *et al.*, 2009; Xiao *et al.*, 2010]. This factor, which is well known in the computer vision community, has led to a decade-long effort on building large-scale annotated datasets that show large benefits for 2D image segmentation, labeling, classification, and object detection [Russell *et al.*, 2008; Deng *et al.*, 2009; Xiao *et al.*, 2010; Kuettel *et al.*, 2012]. However, limited efforts are conducted for point cloud labeling benchmarks. To the best of the our knowledge, the existing annotated point cloud or RGBD datasets [Silberman, 2012; Hema *et al.*, 2009] are incomparable to the 2D ones in terms of either scale or coverage. This causes even state-of-the-art point cloud labeling algorithms to only consider well controlled environments with similar training and testing conditions [Koppula *et al.*, 2011; Anand *et al.*, 2012; Xiong *et al.*, 2011].

Inspirations. Manual point cloud labeling is certainly one solution to the lack of sufficient training data. However, it requires intensive human labor, which costs the community years for 2D ImageNet, even with the help of crowdsourcing, let alone the fact that labeling 3D points is more user-unfriendly and time-consuming than 2D images. Even given sufficient point cloud labels, effective 3D feature design continues to remain open. However, turning to the 2D side, with such massive pixel-wise image labels already at hand, is it possible to do a cross-domain search and "propagate" or "transfer" such labels from images to point clouds? This approach, if possible, would solve the training data insufficiency, while not requiring intensive point cloud labeling, and also resolves the open problem of designing effective 3D feature and geometric representation.

It is worth noting that 3D models or point clouds from either Structure from Mo-

tion [Montemerlo *et al.*, 2002; Furukawa *et al.*, 2010] or low-cost RGBD sensors are usually accompanied by RGB *reference images*. Subsequently, we propose exploiting the *reference images* required for point cloud constructions as a bridge. This idea is also inspired from the recent endeavors in search-based mask transfer learning, which has shown great potential to manage the "cross-domain" issue in both object detection and image segmentation [Kuettel *et al.*, 2012; Shrivastava *et al.*, 2011; Malisiewicz *et al.*, 2011a]. While requiring good data coverage, this is progressively practical with the increasingly "dense" sampling of our world as images, for instance, there are over 10M images in ImageNet [Deng *et al.*, 2009], over 100K segments in LabelMe [Russell *et al.*, 2008], and over 500K segmented images in ImageNet-Segment[Kuettel *et al.*, 2012]. Furthermore, such search-based propagation can be performed in parallel by nature, with high scalability towards big data.

Approach. To utilize the large-scale annotated 2D datasets, we formulate the scene understanding problem as a potential minimization problem on an MRF. This is similar to the MRF formulation from the previous chapter, but the differences are in the variable definition (as semantic labels instead of similarity scores), and cross-domain structure, which naturally also results in different potential function designs. In order to propagate semantic labels from external images to the query point cloud, the potential function uses two terms to capture two necessary operations, namely, *search based superpixel labeling* as the unary potential, and *3D contextual refinement* as the pairwise potential. The overall framework is outlined in Figure 3.1.

Search based Superpixel Labeling. Given the massive pixel-wise image labels from external sources, such as ImageNet [Deng *et al.*, 2009] or LabelMe [Russell *et al.*, 2008], we first use MeanShift to over-segment individual images into *superpixels*, and then propagate their labels onto the visually similar superpixels in the reference images of point clouds via an MRF. From an MRF perspective, these local search results based on 2D superpixels act as the unary term, and the final cross-domain fusion process is described with the pairwise term. Inspired by the recent success in 2D [Malisiewicz *et al.*, 2011b] and 3D detection [Song and Xiao, 2014], we accomplish the search process using *Exemplar SVMs* (ESVMs), rather than the naïve NN search, because the latter is not sufficiently robust against the "data bias" issue, e.g., the photometric condition changes between training and testing sets. More

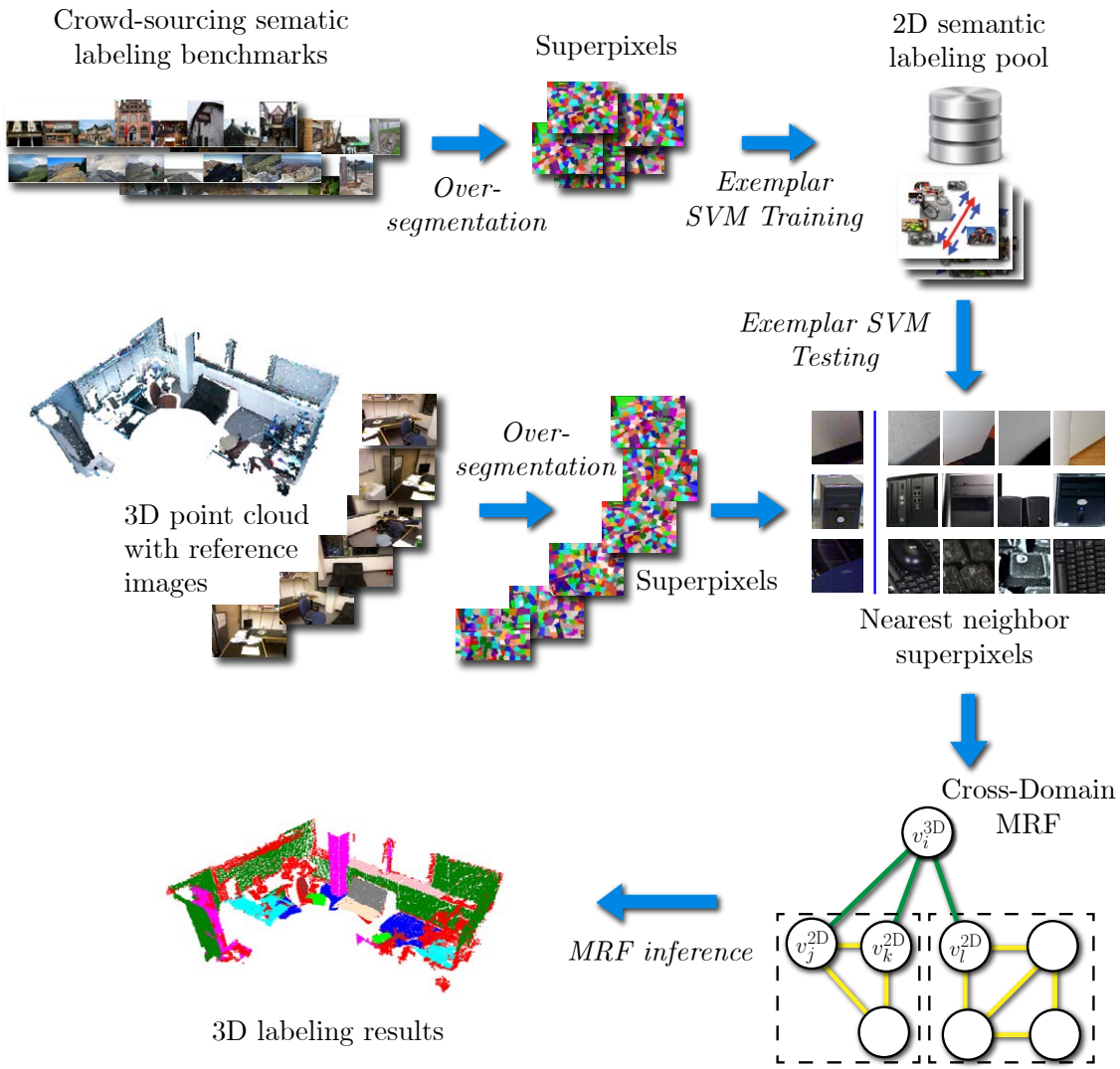


Figure 3.1: The framework of 3D scene understanding based on cross-domain search on 2D annotated datasets.

specifically, we first train linear SVMs for individual "exemplar" superpixels in the external image collection, use them to retrieve the robust k Nearest Neighbors (kNN) for each superpixel from the reference images, and then collect their labels for future fusion.

3D Contextual Refinement. We then aggregate superpixel label candidates to jointly infer the point cloud labels. Similar to the existing works in image labeling, we exploit the intra-image spatial consistency to boost labeling accuracy. In addition, and more importantly, 3D contexts are further modeled to capture the inter-image superpixel consistency.

Both contexts are integrated into the pairwise potential in the graphical model to seek a joint optimal among the superpixel outputs with Loopy Belief Propagation.

In summary, a cross-domain MRF is constructed on the supervoxels of the input point cloud and the superpixels of 2D input reference images. The MRF vertices are connected based on 2D-3D correspondence and 2D neighboring relationships. By minimizing the potential function as a weighted sum of the unary and pairwise terms, the semantic labels of each supervoxel, and thus each 3D point, are obtained.

The rest of this chapter is organized as follows: First, the related literature is reviewed in Section 3.2. Then, the general MRF framework is introduced in Section 3.3. Section 3.4 introduces our search-based superpixel labeling as the unary potential. Section 3.5 introduces our 3D contextual refinement as the pairwise potential. We detail the experiment comparisons in Section 3.6, followed by the summary and future work in Section 3.7.

3.2 Related Work

Semantic Labeling with Contextual Optimization. Semantic labeling of 2D images is a long-standing problem in computer vision. State-of-the-art approaches usually incorporate context with independently predicted labels (for each pixel or region) to obtain spatially consistent results [Munoz *et al.*, 2010; Murphy *et al.*, 2003; Fulton and Koller, 2009]. When integrating the contextual information, MRFs are often used [Murphy *et al.*, 2003; Fulton and Koller, 2009], whereas some work also makes context a feature, and encodes it in the independent classifiers [Munoz *et al.*, 2010].

Some of the recent works in 3D semantic labeling also follow this scheme, either under a structured SVM framework [Koppula *et al.*, 2011; Anand *et al.*, 2012], or using MRFs [Kalogerakis *et al.*, 2010]. Similarly, there are also work use features that incorporate the spatial context into classifiers [Xiong *et al.*, 2011]. Although good performance is reported, such approaches, regardless of whether they are 2D or 3D, require the training and testing data to be from similar collection settings, thus preventing its practical applications on 3D point clouds, where large-scale training data is not available and is difficult to label. We address this problem by seeking help from existing massive 2D datasets, with a novel

labeling approach inspired from mask transfer.

Semantic Labeling with Mask Transfer. Another branch for labeling work originates from the rising endeavors in transfer learning, i.e., to intelligently obtain certain knowledge from different, yet related, sources with metadata propagation [Pan and Yang, 2010] that show promising performance in various tasks, such as scene understanding [Liu *et al.*, 2011], segmentation [Kuettel *et al.*, 2012], and 3D object detection [Patterson *et al.*, 2008]. In 3D semantic labeling, there is also work that adopts online synthesized data for label transfer [Nan *et al.*, 2012; Lai and Fox, 2010]. Its principle lies in identifying NNs in the reference data collection, followed by transferring the corresponding metadata from the neighbors to the query target. However, traditional search-based mask transfer is typically deployed between datasets within the same domain (e.g., from 2D images to 2D images), which does not fit our scenario that involves domain changes. We address this with robust search using ESVMs and incorporating 3D context to ensure a robust fusion from 2D superpixels to point clouds.

Search by ESVMs. The goal of ESVMs [Shrivastava *et al.*, 2011; Malisiewicz *et al.*, 2011a] is to combine the previous parametric classifiers with the non-parametric, search-based model. To this end, an SVM is trained for each instance, e.g., image or superpixel, the ensemble of which is then used to identify NNs of the target instance. Because the discriminatively trained classifier can detect the most unique features for each instance, ESVM has shown promising performance in object detection [Malisiewicz *et al.*, 2011a] and cross-domain retrieval [Shrivastava *et al.*, 2011]. However, it is not easy to directly extend the ESVMs trained on 2D images/superpixels to 3D points, which demands a comprehensive distance metric, rather than binary decisions (is/is not the given instance). Our approach, as detailed in Section 3.4, manages this with a jointly optimized reranking step that uses structured prediction [Joachims.T., 2002].

3.3 Framework

3.3.1 Problem Formulation

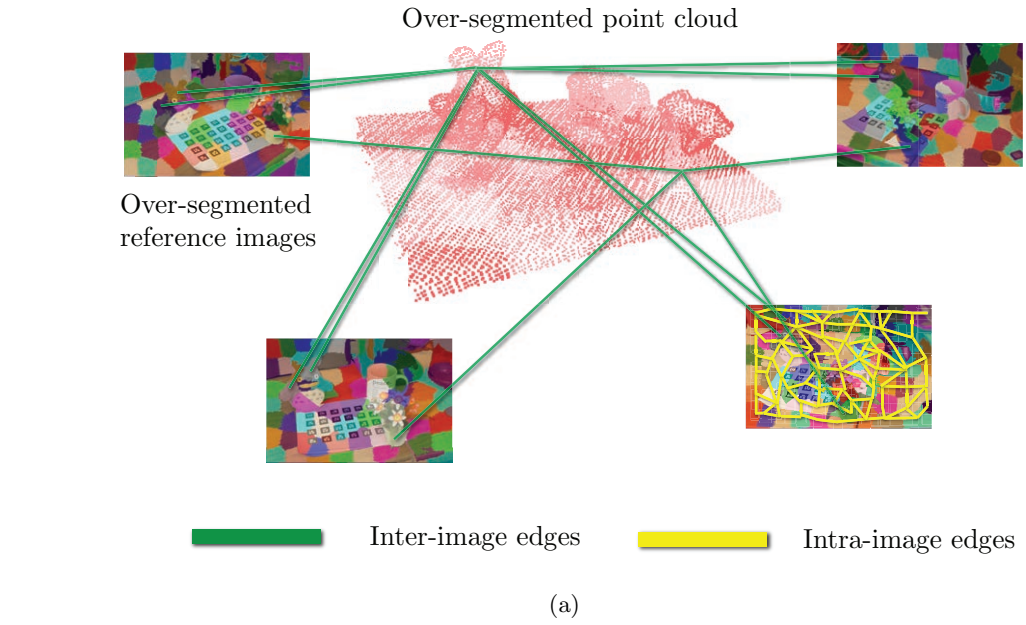
We denote a point cloud as a set of 3D points $\mathcal{P} = \{\boldsymbol{p}_i\}$, each of which is described with its 3D coordinates and RGB colors $(x_i, y_i, z_i, R_i, G_i, B_i)$. $\mathcal{P}$ is built from R reference images $\mathcal{I}_R = \{I_r\}_{r=1}^{R}$ that use methods such as Structure from Motion [Snavely *et al.*, 2006] or SLAM [Montemerlo *et al.*, 2002; Endres *et al.*, 2012]. We also have an external superpixel labeling pool that consists of superpixels with ground truth labels $\mathcal{S} = \{S_i, l_i\}_{i=1}^{N}$. Moreover, the target of the 3D scene understanding problem is to infer the semantic label associated with each point $\boldsymbol{p}_i$.

3.3.2 MRF Construction

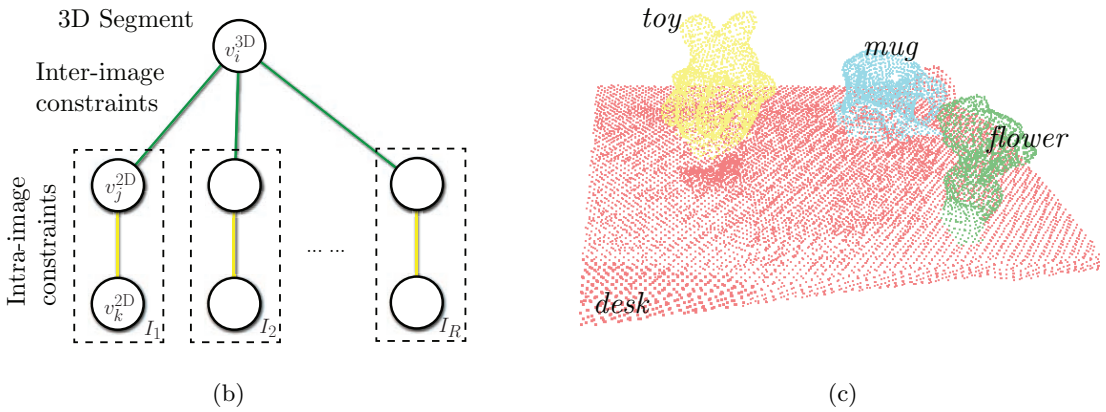
Following the practice of contextual refinement, we formulate a cross-domain MRF to utilize the information from both the 2D reference images and the 3D point cloud. The MRF consists of a vertex set $\mathcal{V} = \{v_i\}_{i=1}^{|\mathcal{V}|}$ and an edge set $\mathcal{E}$. Each vertex v_i is associated with a semantic label y_i (e.g., *wall*, *monitor*) that is the expected output. Note that the vertex in the MRF here is not a 3D point – it can be a set of 3D points, or a set of 2D pixels, as detailed below. A potential function is defined on the vertices $\Psi(\{v_i, y_i\}_{i=1}^{|\mathcal{V}|})$, and we expect to properly design the potential function such that the most likely semantic labels $\{y_i\}_{i}^{|\mathcal{V}|}$ appear when the potential function is minimized.

The vertex set $\mathcal{V}$ consists of two types of nodes, 3D $\mathcal{V}^{3D}$ and 2D $\mathcal{V}^{2D}$, from the over-segment of the 3D point cloud and the 2D reference images, respectively. The 3D point cloud is over-segmented based on smoothness and continuity [Koppula *et al.*, 2011], thus producing a set of 3D segments $\mathcal{V}^{3D} = \{v_i^{3D}\}$, as shown in Figure 3.2(a). Moreover, MeanShift [Achanta *et al.*, 2012] is used to cluster the reference images by dividing them into superpixels $\mathcal{V}^{2D} = \{v_i^{2D}\}$. Note that we run MeanShift on each reference image independently, and collect all the superpixels to form $\mathcal{V}^{2D}$. The vertex set is defined as the union of 3D supervoxels and 2D superpixels $\mathcal{V} = \{\mathcal{V}^{3D}, \mathcal{V}^{2D}\}$.

The edges also have two categories, inter-image, and intra-image. Because the transform matrices $\{M_i\}$ from the global 3D coordinates to the local 2D coordinates in the reference



(a)



(b) (c)

Figure 3.2: Construction of the cross-domain MRF: (a) shows how to construct nodes and edges from an example point cloud; (b) shows the abstract representation of the MRF; and (c) shows the expected output after optimizing the model. For clarity, only part of the connections is plotted in (a).

images $\{I_i\}_{i=1}^{R}$ are known from the 3D reconstruction or registration process, the segments in $\{v_i^{3D}\}$ can be projected to the reference images, each resulting in a 2D region $S_{M_j}(v_i^{3D})$. If this projected region shares a sufficient portion with a given superpixel v_k^{2D} from this reference image, we connect an edge between v_i^{3D} and v_k^{2D}, shown as green links in Figure 3.2(a). This type of edges $\mathcal{E}^{3D\text{-}2D}$ are called inter-image edges, and utilize the spatial

consistency across different images. Note that MRFs do not require a 3D supervoxel v_i^{3D} to correspond to a single superpixel v_j^{3D} in a reference image. This gives us more flexibility, and allows segmentation of the 3D point cloud and 2D reference images to be performed independently.

In addition, spatially adjacent superpixels within one reference image are also connected, as shown by the yellow links in Figure 3.2(a). This is called inter-image edges $\mathcal{E}^{2D\text{-}2D}$, and utilize the spatial consistency within one reference image. Combining the two types of edges, the edge set of the MRF is $\mathcal{E} = \{\mathcal{E}^{3D\text{-}2D}$ and $\mathcal{E}^{2D\text{-}2D}\}$. Figure 3.2(b) shows the corresponding MRF.

3.3.3 Potential Function

Following the MRF standard practice, the potential function contains a unary term to capture the local recognition result, and a pairwise term to embed the spatial consistency. Given the lack of 3D training data, as indicated in the previous section, the unary potential is only defined on the 2D nodes $\mathcal{V}^{2D}$, whereas the pairwise potential are defined on all the edges $\mathcal{E}$.

$$\Psi(\boldsymbol{v}, \boldsymbol{y}) = \sum_{v_i^{2D} \in \mathcal{V}^{2D}} \Psi_u(v_i^{2D}, y_i) + \lambda \sum_{(v_i, v_j) \in \mathcal{E}} \Psi_p(y_i, y_j). \tag{3.1}$$

Here, λ is a parameter for weighing the two terms. The unary term $\Psi_u(v_i^{2D})$ uses an offline database to estimate the semantic label of a local 2D superpixel v_i^{2D}, which is detailed in Section 3.4. The pairwise potential introduces both the intra-image constraint defined by $\mathcal{E}^{2D\text{-}2D}$, and the inter-image constraint defined by $\mathcal{E}^{3D\text{-}2D}$ to enforce spatial consistency, which is illustrated in Section 3.5. Finally, we use Loopy Belief Propagation to find a maximum likelihood solution of the MRF, which is also introduced in Section 3.5.

3.4 Unary Potential: Search via Ensemble of Classifiers

3.4.1 Search-based Label Propagation

As mentioned previously, because of the lack of 3D training data, the unary potential is only defined on the 2D vertices, i.e., the superpixels from MeanShift [Achanta *et al.*, 2012]. Note that we do not leverage the randomly sampled rectangles used in recent works in search-based segmentation [Kuettel *et al.*, 2012; Deselaers *et al.*, 2012] or object detection [Malisiewicz *et al.*, 2011a] in order to ensure label consistency among the pixels in each region, as widely assumed for superpixels [Gould *et al.*, 2009a][1].

For every superpixel v_i^{2D} to be labeled in the reference images, we first obtain an estimation of the semantic labels from this local region, and use the unary potential to penalize the final inferred label that is far from the estimation. More specifically, the unary potential is defined as

$$\Psi_p(v_i^{\text{2D}}, y_i) = 1 - f(\boldsymbol{x}_i^{2D}, y_i). \tag{3.2}$$

Here, $\boldsymbol{x}^{\text{2D}}$ represents the features extracted from v^{2D}, and $f : \mathbb{R}^d \times \mathbb{N} \rightarrow [0, 1]$ is a function that maps the visual feature to a prediction score for a specific semantic class, or in other words, the likelihood of the superpixel belonging to the class given the visual features. Because the potential is to be minimized, the $(1 - f)$ design ensures that the class with a smaller prediction score will be penalized. Whereas traditional approaches train a classifier to perform this estimation, in order to achieve better cross-domain performance and scalability, we obtain $f(\cdot, \cdot)$ by finding the most visually similar superpixels in the external labeled superpixel pool $\mathcal{S}$, whose label is then propagated and fused to v_i^{3D} via the MRF. This is less sensitive to the training data, and thus fits our specific scenario of cross-dataset semantic labeling propagation better, with the benefits of less generalization error and more scalability to the training data amount.

To achieve this goal, a straightforward solution is to directly find the kNN in $\mathcal{S}$ that

[1] Techniques such as objectness detectors [Alexe *et al.*, 2010] can be further integrated to boost accuracy and efficiency.

result in the following objective function:

$$k\text{NN}(v_i^{\text{2D}}) = \arg\min_{S_i \in \mathcal{S}}^{k} D(S_i, v_i^{\text{2D}}), \tag{3.3}$$

where $\arg\min_{S_i}^{k}$ denotes the top k superpixels with the least distance D from v_i^{2D}, and $D(\cdot,\cdot)$ is the metric used to measure the distance among superpixels. In our approach, the predictor f in Equation (3.2) is then defined as the label distribution in $k\text{NN}(v_i^{\text{2D}})$.

3.4.2 Label Propagation with ESVM

As indicated in [Malisiewicz *et al.*, 2011a], the NN search with Euclidean distance cannot capture the intrinsic visual similarity between superpixels. While it is possible to learn a distance metric for the NN search [Yang and Jin, 2006; Weinberger and Saul, 2009], such approaches still suffer from the lack of scalability to large scale of data. Similar to the previous chapter, we divide the overall feature space into a large number of small regions, each with a local similarity measurement[2]. More specifically, ESVM [Malisiewicz *et al.*, 2011a] is introduced to build such a robust measurement. For *every* superpixel extracted from the labeling pool $S_i \in \mathcal{S}$, which contains the 2D image, mask, and corresponding semantic information, we train a linear SVM to identify its visually similar superpixels.

For any superpixel S_i, the training process starts with the generation of training examples. To improve robustness in the testing stage, S_i is translated and rotated to expand to more positive examples for training, and the negative examples are subsampled from other superpixels. One observation worth noting is that, if a superpixel that has similar visual appearance to the positive example appears in the negative set, this will significantly degenerate performance with an ill-trained SVM; this is not studied in object detection [Malisiewicz *et al.*, 2011a]. To address this issue, we add an extra constraint where only superpixels from a semantic category different from S_i can be chosen as negative examples, given that we have the semantic information of each superpixel in the database $\mathcal{S}$.

Subsequently, assuming that M training examples are collected, the ESVM for S_i is

[2]It is not necessarily a distance metric because we do not enforce triangle inequality in the optimization.

trained to optimize the margin of the classification boundaries:

$$\begin{aligned} \arg\min_{\boldsymbol{w}_i, b_i} \frac{1}{2}\|\boldsymbol{w}_i\|_2^2 + C^+ \sum_{\{j|y_j>0\}} \xi_j + C^- \sum_{\{k|y_k<0\}} \xi_k, \\ \text{s.t. } y_j(\boldsymbol{w}_i^T \boldsymbol{x}_j + b_i) \geq 1 - \xi_j, \quad j = 1, 2, \cdots, M. \end{aligned} \tag{3.4}$$

This is derived from the optimization problem of regular linear SVMs. Here, $\boldsymbol{w}_i$ and b_i are the projection direction and constant offset of the decision plane, respectively. Note that in one optimization problem, we only have one $\boldsymbol{w}_i$ and b_i, whereas i remains constant in the training process. y_i is the ground truth training labels of the data, which is $+1$ for positive examples, and -1 for negative examples. The second term in the objective penalizes false negatives, and the third term penalizes false positives. Given that we have fewer positive than negative examples, a larger C^+ is applied to penalize the decision boundary that contracts against the positive examples. $\boldsymbol{x}_j$ represents the visual features extracted from training example S_j.

3.4.3 Hard Negative Mining

Unlike regular ESVMs that can find nearly identical instances, we set a small C^+ and C^- in the training process for more generality, thus allowing the superpixels that are not exactly the same as S_i to also have positive scores. However, this may increase the number of false positives with different labels. To address this problem, given that the decision boundary is only determined by the "hard" examples (the support vectors), we introduce hard negative mining to constrain the decision boundary. More specifically, the hard negative mining process contains three steps:

1. Apply the ESVM trained from S_i on the training data, thus collecting the prediction scores $\{s_j\}$
2. Add the false positives $\{S_j | s_j > 0, l(S_j) \neq l(S_i)\}$ into the negative examples and launch another round of SVM training
3. Repeat the first two steps until no new hard examples are found, or a preset iteration number is reached.

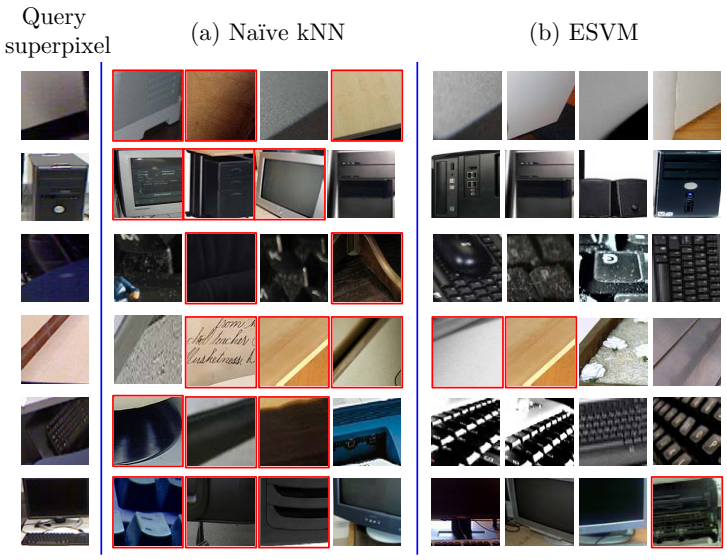


Figure 3.3: Examples of search-based label propagation from ImageNet [Deng *et al.*, 2009] to Cornell Point Cloud Dataset [Anand *et al.*, 2012]: (a) outputs from Naïve kNN; and (b) outputs from ESVMs. We can see that Naïve kNN has many false positives (with red borders), e.g., it outputs *printer* and *table* for the input *wall*. However, the performance for ESVM is much more robust. For each result, we show not only the superpixel, but also its surroundings for clarity.

By combining a small C and supervised hard negative mining with labels, we achieve a balance of generality and sensitivity. Figure 3.3 shows a comparison between ESVM and naïve kNN search-based on Euclidean Distance. We can see that, although naïve kNN usually outputs visually similar instances, it is not robust against false positives, whereas ESVM performs better in filtering superpixels with different semantic labels, although they may be visually similar to the query.

Using Equation (3.4), we can train an ESVM for each superpixel $S_i \in \mathcal{S}$, resulting in the SVM weights and offsets $\{\boldsymbol{w}_i, b_i\}$. In order to label the superpixel S_q in the reference images, we find the superpixels with the k strongest responses from their ESVMs as their kNN in $\mathcal{S}$, i.e.,

$$k\text{NN}(S_q) = \arg\max_{S_i \in \mathcal{S}}^{k} F_i(\boldsymbol{w}_i^T \boldsymbol{x}(S_q) + b_i). \tag{3.5}$$

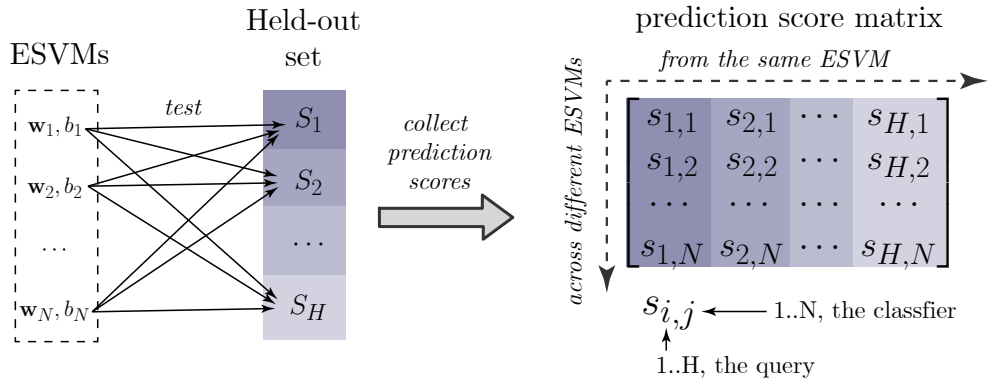


Figure 3.4: Settings for the ESVM calibration step. The independently trained ESVMs are applied on the held-out set $\mathcal{S}_H$, and the prediction scores are collected as a matrix. Color coding is used to distinguish among scores from different superpixels.

Here, F_i is the ranking function introduced below.

3.4.4 SVM Calibration

With the constraint for Equation (3.4), for one single ESVM, a higher response $s_i = \boldsymbol{w}_i^T \boldsymbol{x}(S_q) + b_i$ generally means higher similarity between query S_q and superpixel S_i. However, such responses $\{s_i\}$ are not comparable across different SVMs (i.e., different is) because each ESVM is trained independently without global or pairwise constraints. We address this issue by learning a reranking function $F_i(\cdot)$ for each ESVM, thus making $\{F_i(s_i)\}$ comparable by calibrating each ESVM score with scaling and offsetting. The calibration function in our method has a linear form,

$$F_i(x) = w_i^{(r)} \cdot x + b_i^{(r)}. \tag{3.6}$$

This linear mapping does not modify the relative order in each ESVM, but rescales and pushes the ESVMs jointly in order to make their prediction scores comparable. This differs from the calibration step in [Malisiewicz *et al.*, 2011a] that transforms the decision boundary of each SVM *independently* for object detection.

The basic setting of the calibration step is shown in Figure 3.4. We have a held-out set $\mathcal{S}_H = \{S_h\}_{h=1}^H$ that consists of superpixels with human-labeled semantic labels. After applying N ESVMs to the superpixels in $\mathcal{S}_H$, the prediction scores $\{S_{i,j}\}$ are collected,

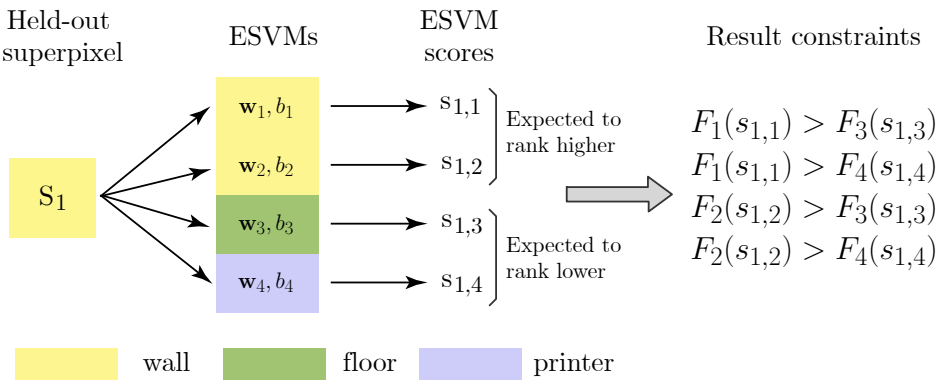


Figure 3.5: Intuition of how to generate optimization constraints for ESVM calibration. Color coding shows the semantic labels of the held-out superpixel and ESVMs.

where i indicates the superpixel ID, and j indicates the ESVM ID, shown as the matrix to the right of Figure 3.4. As color coding shows, the scores in each column of this matrix are from the same superpixel, but different ESVMs, and thus need to be calibrated.

In addition, given that we have the ground truth semantic labels for both the $\mathcal{S}_H$ and EVSMs, intuition lies in the fact that, for every held-out superpixels $S_h \in \mathcal{S}_H$, ESVMs with the same semantic label should rank higher than those with different semantic labels. A toy example is shown in Figure 3.5, where only one superpixel, which is *wall*, is present in the held-out set, and four ESVMs are trained independently with semantic labels *wall*, *floor*, and *printer*, respectively. Because the first two ESVMs have the same categories as the superpixel S_1, they are expected to have higher calibrated scores $F_1(s_{1,1}), F_2(s_{1,2})$ compared with those from the bottom two ESVMs $F_3(s_{1,3}), F_4(s_{1,4})$, as shown to the right of the figure.

Finding $\boldsymbol{w}$s and bs that satisfy such ranking constraints exactly fits the settings for the

Algorithm 1: Training algorithm of the 3D scene understanding based on ensemble of weak classifiers.

Input: A set of superpixels with ground truth labels $\mathcal{S} = \{S_i, y_i\}_{i=1}^{N}$

for Superpixel $S_i \in \mathcal{S}$ **do**

  Train an ESVM for each superpixel with Equation 3.4 and hard negative mining

end

Learn a reranking function by solving Equation 3.7

Output: A set of ESVMs with reranking weights and associated labels $\{\boldsymbol{w}_i, b_i, w_i^{(r)}, b_i^{(r)}, y_i\}$

structured learning-to-rank problem [Joachims.T., 2002].

$$\begin{aligned} &\min \sum_i \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i,j,k}\xi_{i,j,k} \\ &\text{s.t. for every query } q_i, \\ &\quad \boldsymbol{w}^T\boldsymbol{\Phi}(q_i, s_j) > \boldsymbol{w}^T\boldsymbol{\Phi}(q_i, s_k) + 1 - \xi_{i,j,k} \\ &\quad \forall l(S_j) = l(S_{q_i}), l(S_k) \neq l(S_{q_i}) \\ &\quad \xi_{i,j,k} \geq 0. \end{aligned} \tag{3.7}$$

Here, $\boldsymbol{w} = \{(w_i^{(r)}, b_i^{(r)})\}_{i=1}^{n}$. The $\boldsymbol{\Phi}(q_i, s_j)$'s $(2j-1)$th and $2j$th dimensions are s_j and 1, respectively, for encoding the weights and scores into a single vector. This problem is not convex or differentiable, and therefore, we optimize its upper bound with a cutting plane algorithm [Tsochantaridis *et al.*, 2005]. Algorithm 1 shows the training procedures of our label propagation.

3.5 Pairwise Potential: Spatial Verification

3.5.1 Intra-image and Inter-image Terms

The unary potential defined in Equation (3.2) captures the local recognition result; however, spatial consistency is also utilized to filter noise in the local recognition. Therefore, pairwise potential is introduced, which can be interpreted as "fusing" the approximate results of the

Algorithm 2: Algorithm of the testing stage of the 3D scene understanding based on ensemble of weak classifiers.

Input: Point cloud $\mathcal{P}$ with reference images $\mathcal{I}_R$, and superpixel propagation pool $\mathcal{S}$
Do over-segmentation on both $\mathcal{P}$ and $\mathcal{I}_R$, and construct the MRF
for Superpixel $S_q \in \mathcal{I}_R$ **do**
    Use Equation 3.5 to retrieve kNN from $\mathcal{S}$
end
Use the over-segmentation structure of $\mathcal{P}$ and $\mathcal{I}_R$, and kNN of S_q to construct the MRF as introduced in Section 3.5
Do MRF inference by minimizing Equation 3.11 with Loopy Belief Propagation
Output: 3D-segment-wise semantic labels of $\mathcal{P}$

ESVMs from 2D reference images to the 3D point cloud. Note that, unlike traditional contextual refinement approaches, [Koppula *et al.*, 2011; Anand *et al.*, 2012; Munoz *et al.*, 2010], our approach does not require labeled 3D training data.

As mentioned in the construction of the MRF, the pairwise potential consists of two parts, the intra-image potential defined between two spatially adjacent superpixels within the same reference image, and inter-image potential defined between a 3D supervoxel and 2D superpixel.

$$\sum_{(v_i,v_j)\in\mathcal{E}} \Psi_p(y_i,y_j) = \lambda_1 \sum_{(v_i,v_j)\in\mathcal{E}^{2D-2D}} \Psi_{p2}(y_i,y_j) + \lambda_2 \sum_{(v_i,v_j)\in\mathcal{E}^{3D-2D}} \Psi_{p3}(y_i,y_j). \tag{3.8}$$

Similar with the above, λ_1 and λ_2 denote the weights for different potential parts.

3.5.2 Potential Design

Intra-Image Consistency. To encode the intra-image consistency in the reference images, neighboring superpixels are encouraged to have related labels, defined by the intra-image smoothing term $\psi_{s,2\mathrm{D}}$.

$$\Psi_{p2}(y_i,y_j) = p(y_i,y_j), \tag{3.9}$$

where $p(\cdot,\cdot)$ is the co-occurrence probability learned from the superpixel labeling pool $\mathcal{S}$.

Inter-Image Consistency. To make the 3D labeling results consistent among reference images, we further define the inter-image smoothing term as

$$\Psi_{p3}(y_i, y_j) = \begin{cases} 1 & y_i = y_j \\ c & y_i \neq y_j \end{cases} \tag{3.10}$$

where $c > 1$ is a constant to penalize inconsistent labels between a 3D supervoxel and the corresponding 3D superpixel.

Integrating Other Context. If 3D training data with ground truth label is also available, we can further integrate stronger context into our MRF. Such context may include 3D relative positions (e.g., a *table* may appear under a *book*, but should not under the *floor*), and normal vector (e.g., a *wall* must be vertical and *floor* must be horizontal), etc., as investigated in [Koppula *et al.*, 2011]. However, this requirement places stronger dependence on training data, thus decreasing generality.

3.5.3 Inference

Overall, we have a more specific potential function design.

$$\begin{aligned} \Psi(\boldsymbol{v}, \boldsymbol{y}) = & \sum_{v_i^{2\mathrm{D}} \in \mathcal{V}^{2\mathrm{D}}} \Psi_u(v_i^{2\mathrm{D}}, y_i) \\ & + \lambda_1 \sum_{(v_i, v_j) \in \mathcal{E}^{2D-2D}} \Psi_{p2}(y_i, y_j) \\ & + \lambda_2 \sum_{(v_i, v_j) \in \mathcal{E}^{3D-2D}} \Psi_{p3}(y_i, y_j). \end{aligned} \tag{3.11}$$

We use Loopy Belief Propagation [Murphy *et al.*, 1999] to find a local minima of Ψ. Algorithm 2 outlines the overall procedure based on the potential design.

3.6 Experimental Validations

3.6.1 Data Collection

To build the superpixel labeling pool, we collect superpixels from ImageNet [Deng *et al.*, 2009], which provides object detection ground truth as bounding boxes[3]. First, we over-segment the image using MeanShift [Comaniciu and Meer, 2002], and then select the superpixels that share sufficient area with the bounding boxes of some object of interest (e.g., *wall* or *floor*). These superpixels are then added to the superpixel labeling pool with their corresponding labels.

To evaluate our algorithm, the Cornell indoor dataset [Hema *et al.*, 2009] is adopted, which contains 24 and 28 office and home scenes, respectively, all constructed from a Kinect sensor and RGBD SLAM [Endres *et al.*, 2012]. Figure 3.6 shows an example of one office scene, including the textured point cloud and stitched point cloud. Each scene consists of 3D points with 3D coordinates, RGB values, semantic labels, and reference images used for the RGBD SLAM construction. Following [Koppula *et al.*, 2011], we take the labels present in ≥ 5 scenes (point clouds) for evaluation. Given that some semantic labels are too specific for real applications (e.g., *chairBackRest*, *chairBase*, and *chairBack*) and are not present in the labeling pool $\mathcal{S}$, we merge those labels into more general labels, for example, *chair*. As a result, in the office scene, we use the labels {*wall, floor, table, chair, monitor, printer, keyboard, cpu, book, and paper*}, and in the home scene, we use the labels {*wall, floor, table, chair, sofa, bed, quilt, pillow, shelfRack, laptop, and book*}[4].

3.6.2 Experiment Settings

We use average classification accuracy, that is, the average percentage of the correctly classified points among all the point clouds, as our protocol for evaluating both the 2D superpixel and 3D point labeling. We compare our approach with the following:

[3]Although the general ImageNet does not have an object bounding box and semantic label available, there is a specific subset of ImageNet that has such information.

[4]Unlike [Koppula *et al.*, 2011], the data are only used for testing, and are *not* involved in the training process.

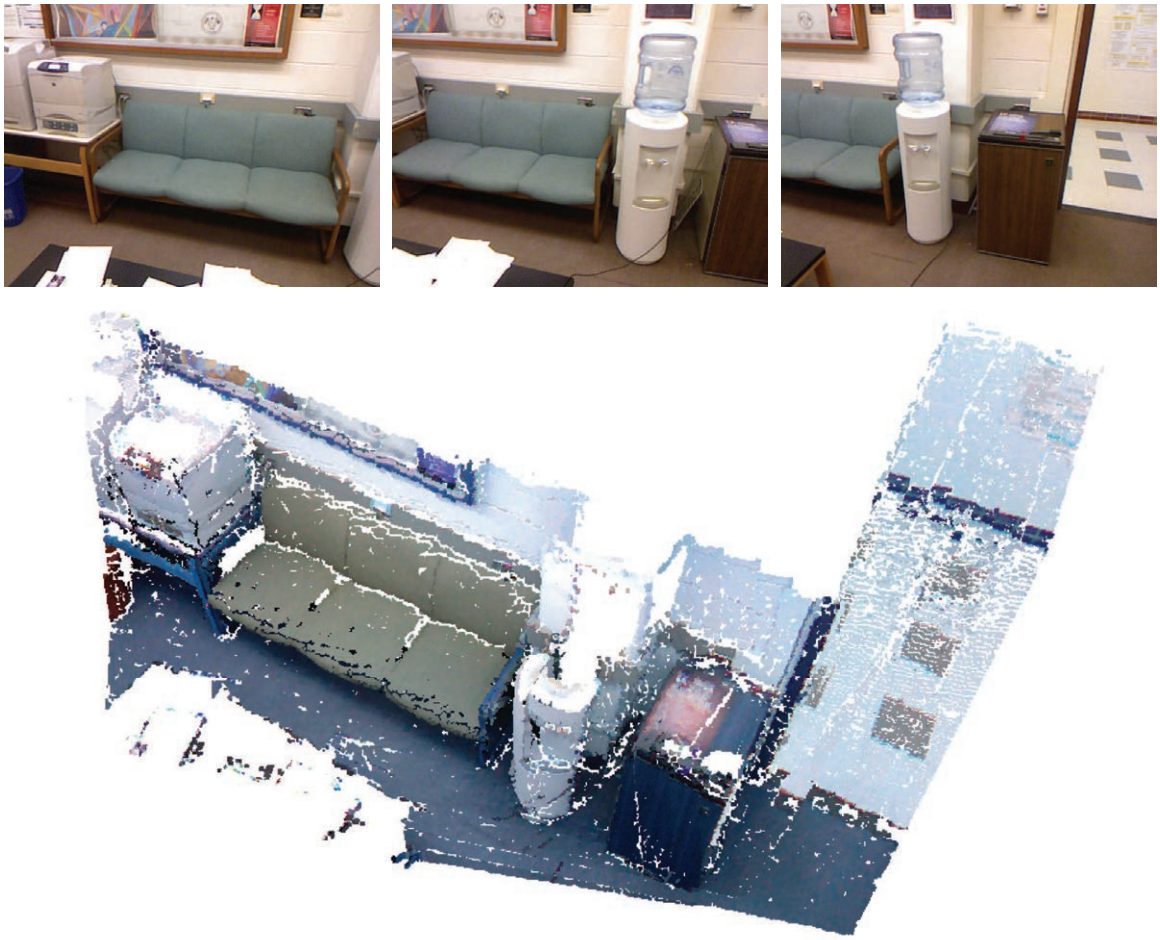


Figure 3.6: A sample scene of the Cornell Point Cloud Dataset used in our evaluation, including both the stitched point cloud and reference images.

(1) Naïve kNN search-based propagation;

(2) ESVM-based propagation;

(3) ESVM-based propagation with contextual refinement;

(4) The state-of-the-art work by Anand et al. [Koppula *et al.*, 2011] that uses 3D training data known to be similar with the test data.

For Approaches (1) and (2), because of the lack of inter-image optimization, we use majority voting to obtain the 3D labels. With regard to Approach (4), given that our approach does

Superpixel Feature	Dimension
HoG	9×12
Average HSL values	3×12
Difference of Gaussian	1×12
Laplacian	1×12
Edge detector (with different angles)	5×12
4×4 Walsh Hadamard kernels	16

Table 3.1: Visual features for label propagation.

not use local 3D shapes or geometry, for comparison fairness, we adopt accuracy using only the visual appearance reported in [Anand *et al.*, 2012]. However, note that our approach is complementary to 3D geometry-based methods; therefore, it is easy to add more features and contexts as indicated in Section 3.5. Approaches (1) to (3) use our superpixel label propagation pool for ESVM training, and all the Cornell Point Cloud Dataset for testing.

3.6.3 Implementation Details

In terms of visual features, we use HoG feature [Dalal and Triggs, 2005] (with 4×3 grids), average HSL values, texture features, and 4×4 Walsh Hadamard kernels [Ben-Artzi *et al.*, 2007] to represent each superpixel. A complete list of features is listed in Table 3.1.

We adopt LibSVM[5] for ESVM training and testing with $C^- = 0.01$ and $C^+ = 0.05$. The training examples are generated from the original superpixel with five levels of translation and rotation. For every superpixel, we collect 10,000 negative examples and perform five rounds of hard negative mining. For reranking function learning, we use SVM$^{\text{rank}}$[6] with $C = 200$ and ten-fold cross-validation. In terms of the superpixel labeling pool, approximately 28 K superpixels for each type of scenes (office or home) are collected.

[5] http://csie.ntu.edu.tw/~cjlin/libsvm/

[6] http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

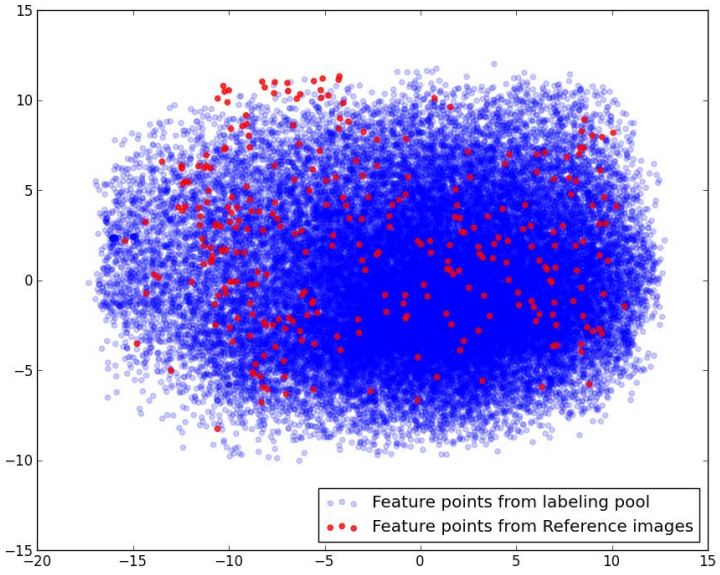


Figure 3.7: Superpixel distribution of the labeling pool (blue) and reference images (red) in the (dimension reduced) feature space.

3.6.4 Quantitative Results

Rationality Checking. Before discussing the details of the quantitative evaluation results, we first illustrate the rationality of our approach by visualizing the superpixel labeling pool (blue dots) and the superpixels from the reference images (red dots) in a 2D space mapped from the feature space with Principal Component Analysis, as Figure 3.7 shows. We can observe good coverage in the feature space, although the collection conditions of the Cornell Point Cloud Dataset are certainly different from ImageNet. On the other hand, the coverage is still not perfect, and requires more advanced techniques than naïve kNN search, such as ESVMs.

Quantitative Results. Figure 3.8 shows the average accuracy of 3D labeling in both office and home scenes, with comparison among different approaches. We can see that ESVMs, even without contextual refinement from the pairwise potential, performs well and outperforms naïve kNN with a large gap. In addition, there is a performance gain after incorporating context, thus making our method (Approach (3)) comparable with the state-

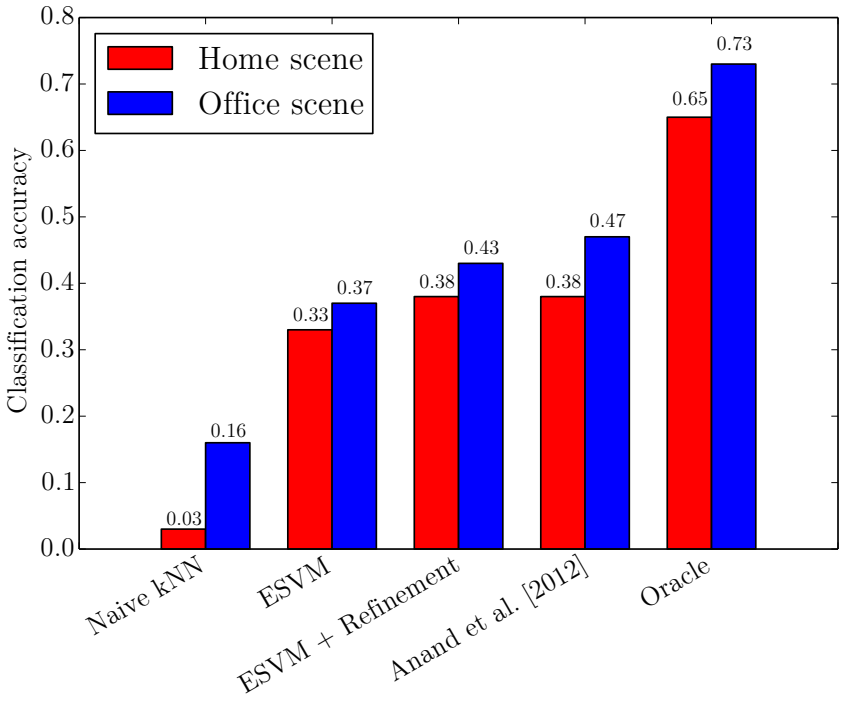


Figure 3.8: Point cloud labeling accuracy of different approaches in both office and home scenes.

Training	**Training** (parallel)	**Testing** (parallel)
43.2 s/superpixel	3.6 s/superpixel	18.2 s/point cloud

Table 3.2: Efficiency of our approach (ESVM + Context)

of-the-art method [Anand *et al.*, 2012] in classification accuracy, without requiring specific knowledge of the target scene. (Note that it is also easy to integrate geometric features in our approach.) In terms of efficiency, our methods requires less than 20 seconds for one point cloud, whereas the approach in [Anand *et al.*, 2012] requires 18 minutes to finish, on average. In the training stage, training one ESVM on a 3.0 GHz desktop CPU requires 3 to 4 seconds. More details on efficiency are listed in Table 3.2. Figure 3.9 shows several examples of results from different approaches with ground truth labels.

Oracle rate. We also test our oracle rate, i.e., ESVM with contextual refinement trained with labeled reference images in the Cornell Point Cloud Dataset. The accuracy of

Figure 3.9: Sample results of point cloud labeling on Cornell dataset [Hema *et al.*, 2009]. To demonstrate labeling results with more details, reference images from multiple views are provided. The rows are reference images, ground truth, and labeling result from Naïve kNN, ESVM, ESVM with refinement, and our oracle performance.

the oracle rate is also shown in Figure 3.8. The performance is evaluated with a four-fold cross-validation setting with Cornell Point Cloud Dataset for training and testing. We can see with a fair experiment setting, i.e., using the same training data and type of features (visual feature), that our approach can outperform the state-of-the-art method with only the visual appearance features. Figure 3.10 shows the confusion matrices for the office and home scenes. Because some classes are affected by extreme shooting conditions, for example, *paper* is often over-exposed, we cannot extract meaningful features from these classes, and therefore, we cannot distinguish them well from other classes, such as *wall*. This indicates the limits of pure visual approaches for 3D point cloud labeling. However, we do not propose substituting geometric features and context with our approach. On the contrary, this observation denotes that our approach can provide complementary information with shape-based methods, considering that easy 3D features, such as normal vectors, can distinguish *paper* from *wall*. We can expect even higher performance by integrating geometric features and context, as mentioned in Section 3.5.2.

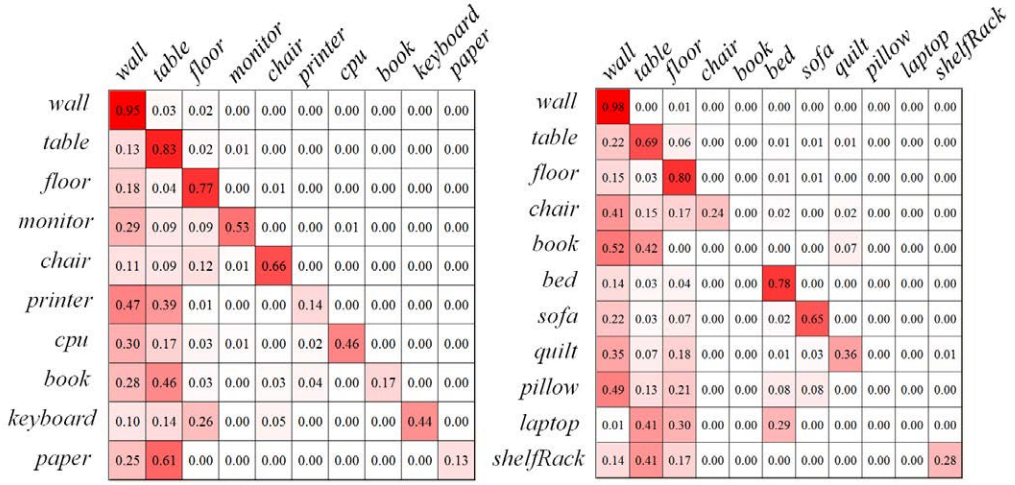


Figure 3.10: Confusion matrix for both office and home scenes if we have training data from the target scenes.

3.7 Summary and Future Work

This chapter investigated another important problem in content-based 3D shape analysis, scene understanding based on point clouds. Because of the RGBD SLAM algorithms, the input for this problem was relatively complete, but sensor noise was still present. There is another unique challenge to the lack of annotated training data for 3D applications. A novel solution for using existing large-scale 2D labeled data was discussed, and this is the first work in this direction, to the best of our knowledge. Similar to the previous chapter, MRF was constructed on the query point cloud. Unlike the general 3D MRF formulation, our MRF was built not only on the 3D supervoxels, but also on the superpixels of the reference images, which act as a bridge to introduce the 2D training data. The potential function of the MRF was then defined as the weighted sum of a unary term and a pairwise term. The unary term encapsulated the local recognition scores from an ensemble of classifiers (ESVM in our case). In the case of the pairwise terms, on one hand, they fused the local recognition 2D results to the 3D space, and on the other hand, they enforced intra and inter-image spatial consistency. Loopy Belief Propagation was then used to obtain a local minima of the potential function. The experiments we conducted over popular datasets validated our advantages with comparable accuracy and superior efficiency to the direct

and fully supervised 3D point labeling state-of-the-arts method, even *without* any point cloud labeling ground truth.

Several interesting directions can be explored to further expand the 2D-benefiting-3D idea. In terms of the unary potential design, currently all ESVMs have to be linearly scanned in order to determine the specific form of the unary potential function. Building an indexing structure to expedite the unary potential computation would be helpful for making the proposed approach more efficient. In terms of the pairwise potential design, how to incorporate the geometric information learned from the 3D data, as mentioned in Section 3.5.2, could also be investigated to further boost performance.

Chapter 4

Pose Recognition of Deformable Objects

4.1 Introduction

In this chapter, we discuss another important problem in content-based 3D shape analysis, pose recognition of deformable objects. More specifically, garment pose recognition in robotics is explored, the settings for which is shown in Figure 4.1. This is motived by the increasing interest of various industries, e.g., fabric and food industry, on the manipulation of deformable objects, such as garments, fruit, and soft containers. In the automation of manufacturing processes, certain deformation or configuration of the target object may be the goal of the manipulation, such as folding garments and putting them into containers. Regardless of the rigidity of the target object, such manipulation systems are usually built as close-loop controlling pipelines. As Figure 4.2 shows, there is a visual recognition module that identifies the current state or *pose* of the target from the sensor input. Based on the recognition result, the robot arm can then manipulate a given object to a predefined configuration (e.g., placing the garment flat). Because of possible error in the recognition or manipulation, this process may be repeated multiple times until the target configuration is reached and recognized.

In the manipulation process, one critical step is for the robot to grasp the object with a gripper and attempt to determine how to transform it into the target configuration. At this

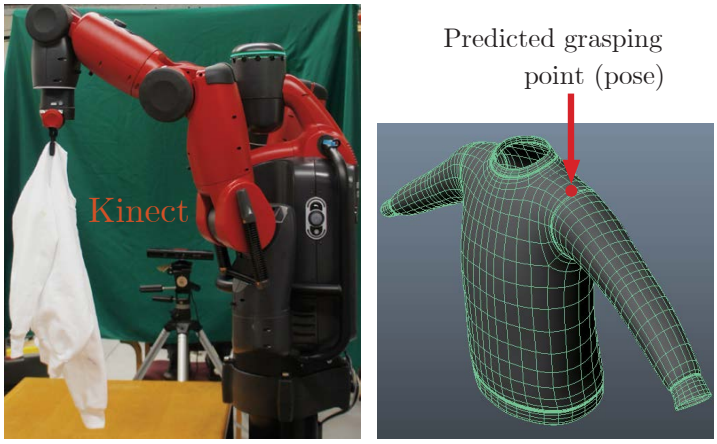


Figure 4.1: Our application scenario: a Baxter robot grasps a sweater, and a Kinect device captures depth images to recognize the pose of the sweater. The recognition result is shown on the right.

stage, the relative position of the grasping point (e.g., grasping at a sleeve, or 1 inch to the left of the collar) is defined as the *pose* of the garment. Because further manipulation relies on the estimation of the current target pose, a pose recognition component is critical for successful manipulation. Therefore, in this chapter, we focus mainly on the *pose recognition* of the pipeline, as shown in the purple rectangle in Figure 4.2, and utilize garments as an example of a deformable object.

Although pose recognition and manipulation of *rigid* objects already have reliable solutions in the robotics community [Brogrdh, 2007], pose recognition for deformable objects remains open because of several challenges. In particular, the existing rigid object manipulation algorithms are not ready to be extended to deformable objects because of two limits. On one hand, these algorithms generally rely on a global rigid geometric transform to describe the final target, which is not applicable to object deformation. On the other hand, because of the many more possible deformation states of the object, the variance of visual appearance is also dramatically enlarged. In addition, considering that real-time or nearly real-time algorithms are generally expected on robotics platforms, these unique challenges make it difficult to extend the pose recognition algorithms from rigid objects to deformable objects.

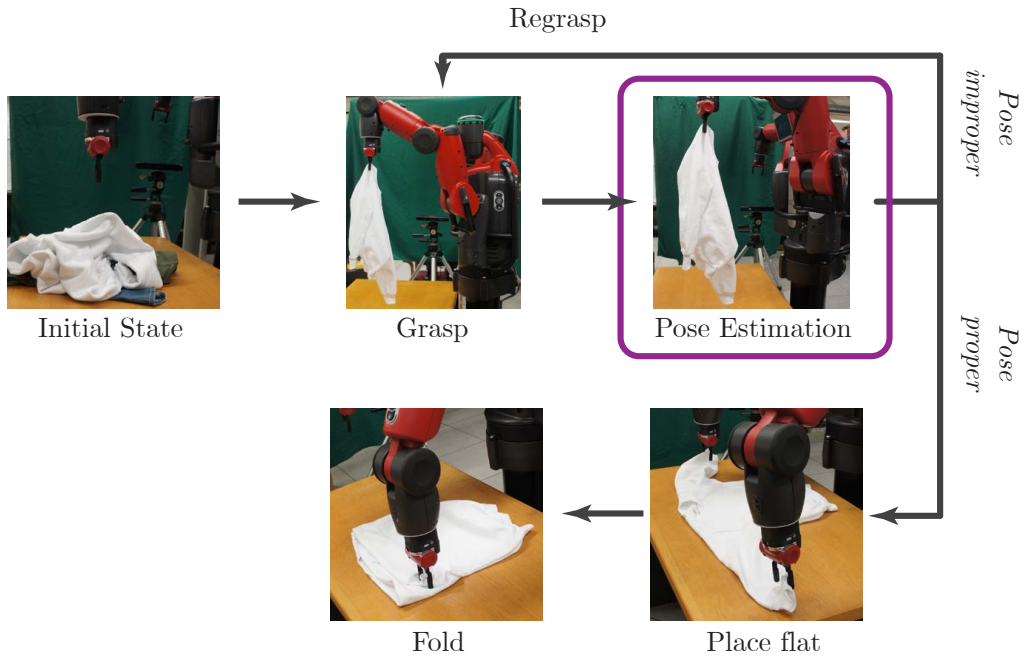


Figure 4.2: The entire pipeline of dexterous manipulation of deformable objects. Our major focus in this chapter is on the phase of pose estimation, as highlighted by the purple rectangle.

However, thanks to the recent development of consumer-level depth sensors, depth information, which has been demonstrated as helpful in resolving visual ambiguity compared with optical cameras, has become available. Therefore, if the challenges mentioned above can be addressed with the newly available depth information, the solution can also benefit more general robotics and computer vision research. Subsequently in this chapter, we focus particularly on the pose recognition problem of a deformable garment, given depth information. We use a Baxter robot as the target platform and equip it with two arms, each with a gripper. Furthermore, a Microsoft Kinect sensor is added to the robot platform to provide depth input. The application setting and expected output are illustrated in Figure 4.1.

Although the problem can be treated as a continuous regression problem, in practice, we quantize the possible grasping points to a finite set. The reason is that from an application perspective, treating the problem as completely continuous does not benefit further motion planning and manipulation algorithms, but dramatically increases the solution space, thus

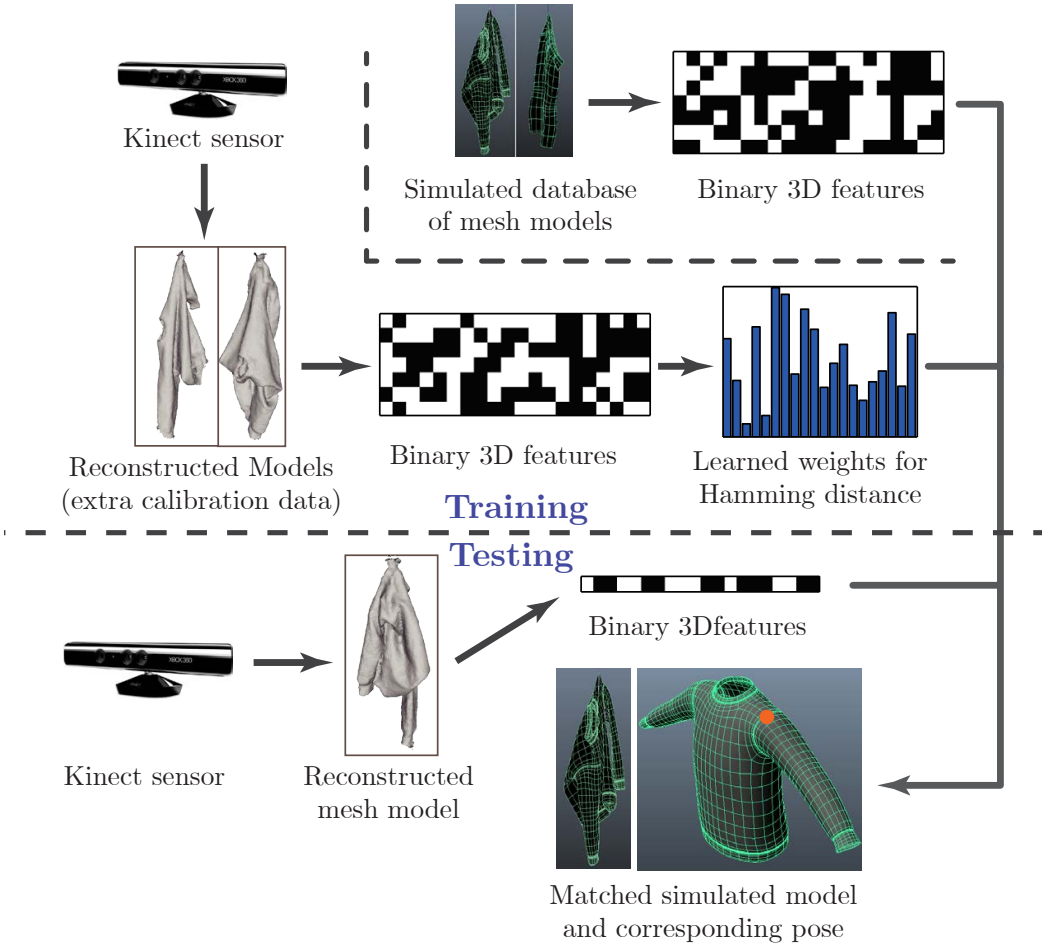


Figure 4.3: Framework of the proposed pose recognition approach. In the offline training stage (in the upper part), we simulate mesh models of different types of garments in different poses, and learn a weighted Hamming distance from additional calibrated data collected from the Kinect device. In the online testing stage (in the lower part), we reconstruct a 3D model from the depth input, find NN from the simulated database with the learned distance metric, and then adopt the pose of the matched model as the output.

increasing difficulty. Subsequently, as argued in the previous chapters, we have two possible directions, directly training a classifier, or performing NN search within a pre-labeled database. Following the advantages of better scalability to the ever-increasing training data, and better generality to cross-domain data, we also adopt the NN search direction of first synthesizing a depth image database offline, and then conduct online search in the testing

stage.

The main idea is that given a commercial quality 3D model of the garments, such as jeans, shorts, and sweaters, we can use physics engines to simulate the 3D model under different conditions. By "different conditions," we refer particularly to different grasping points (i.e., poses), garment categories, specific model design, and material properties (e.g., frictions and hardness). In the testing stage, we first reconstruct a 3D model from the garment, and then search NNs in the database. Because we already know the poses of the models when simulating the database, these ground truth poses of the NNs are then fused as the final output.

This chapter is organized as follows. First, we review the related work in Section 4.2. Because our method contains an online prediction stage and offline simulation and training stage, we introduce the prediction stage in Section 4.3, and then illustrate the training stage in Section 4.4. Section 4.5 compares the proposed approach with state-of-the-art methods both qualitatively and quantitatively. In the end, we discuss the contributions and future work in Section 4.6.

4.2 Related Work

In this section, we first introduce the background of the recognition and manipulation of deformable objects. Because the proposed method is also related to shape matching, we review shape matching and 3D reconstruction from RGBD input.

4.2.1 Recognition and Manipulation of Deformable Objects

Recognition and manipulation of deformable objects has attracted extensive attention from the community, both because of the high demand from industry, and its fundamental position in vision and robotics research. Some initial attempts to classify garment categories can be found in [Willimon *et al.*, 2010, 2011, 2013] with human interactions. Intuition lies in using color segmentation to obtain the boundary shape feature, which is then input into a classifier to predict the clothing category. However, the method is sensitive to texture variance, and thus becomes limited in its applicability.

Since 2011, exploration on the manipulation of deformable objects can be seen on the PR2 robot platform, especially on the clothes-folding task, which requires recognition [Wang *et al.*, 2011; Miller *et al.*, 2012; Schulman *et al.*, 2013; J. Maitin-Shepard and Abbeel, 2010; Cusumano-Towner *et al.*, 2011]. A particularly popular direction is matching the observed shape to a known template. Although these methods have shown good performance in certain cases, they rely on traditional optical cameras, and thus are limited by the natural loss of depth information and subsequent ambiguity. Therefore, these methods may suffer from self-occlusion and texture-less input, which unfortunately, are common in practice. For example, work that relies on corner detection [Cusumano-Towner *et al.*, 2011; J. Maitin-Shepard and Abbeel, 2010] could fail on soft garments whose deformations are usually complicated, and could produce many misleading corner-like features; therefore, they could not reliably match the template against the input. Similarly, the garment pose recognition approaches proposed in [Kita and Kita, 2002; Kita *et al.*, 2009, 2011] show impressive performance when the extent of deformation is limited, but lack further exploration on the practical cases where deformation is reasonably complex.

Whereas most of the methods above are based on optical sensors that include single or stereo cameras, recently, there have been works on pose recognition based on depth information [Li *et al.*, 2014b]. The approach in [Li *et al.*, 2014b] performs recognition directly on individual raw depth images based on a pre-trained classifier, and uses majority voting to obtain a comprehensive result. When the deformation is complicated, the inherent noise of the Kinect sensor and self-occlusions may confuse the algorithm. Furthermore, the method's reliance on the linear scan of every input depth image makes it slow, thus limiting its application. Therefore, in this chapter, the depth images are first fused into a 3D model, rather than search through a large number of individual depth images, in order to make the result more compact and reliable.

4.2.2 Shape Matching

Shape matching is another related and long-standing topic in robotics and computer vision. On the 2D side, various local features have been developed for image matching and recognition [Huttenlocher *et al.*, 1993; Latecki *et al.*, 2000; Lowe, 1999] that have shown good

performance on textured images. Another direction is shape context-based recognition [Belongie *et al.*, 2002; Toshev *et al.*, 2010; Tu and Yuille, 2004], which is better for handwriting and character matching. On the 3D side, Wu *et al.* [Wu *et al.*, 2008] and Wang *et al.* [Wang *et al.*, 2006] proposed methods to match patches based on 3D local features. They extracted Viewpoint-Invariant Patches or the distribution of geometry primitives as features, based on the matching performed. Osada and Funkhouser [Osada and Funkhouser, 2001], Thayananthan *et al.* [Thayananthan *et al.*, 2003], and Frome *et al.* [Frome *et al.*, 2004] applied 3D shape-context as a metric for computing the similarities of the 3D layout for recognition. However, most of the methods are designed for noise-free human-designed models, without the capability of matching the relatively noisy and incomplete mesh model produced by Kinect to the human-designed models. Our method is inspired by the 3D shape context [Frome *et al.*, 2004], but provides the capability of cross-domain matching with a learned distance metric, and also utilizes a volumetric data representation to efficiently extract the features.

Shape matching also plays an important role in 3D reconstruction, which is also related to the proposed approach. With the increasing popularity of the Kinect sensor, various methods are emerging in computer graphics, such as KinectFusion and its variants [Izadi *et al.*, 2011; Chen *et al.*, 2013; Li *et al.*, 2013]. Although these methods have shown success in reconstructing static scenes, they do not fit our scenario where a robotic arm rotates the target garment about a grasping point. Therefore, we first perform 3D segmentation to obtain the garment masks on the depth images, and then use a volumetric-based representation of 3D data [Curless and Levoy, 1996] to perform registration and model refinement.

4.3 Online Reconstruction and Pose Recognition

As Figure 4.3 shows, our method consists of two stages, offline model simulation and online recognition. In the offline training stage, three procedures are accomplished in order. First, we use a physics engine to simulate the stationary state of the mesh models of different types of garments in different poses. Second, a novel binary feature is extracted from the models in order to provide a compact and descriptive representation. Finally, a distance

metric is learned from an extra calibration model set with a supervised setting.

In the online testing stage, a Kinect sensor captures (noisy) depth images of the garment from different views, and we reconstruct a smooth 3D model from the depth input. The same binary feature is extracted from the reconstructed model, and then matched against the features in the database according to the learned distance metric to obtain NN. Because we know the ground truth pose used to simulate the 3D model NN, it is then adopted as the recognition result. The two stages are connected by sharing the features extracted from the database and the learned distance metric.

In this section, we mainly discuss the online testing stage, i.e., what happens after the robot actually picks up the garment. From a high level perspective, this involves three technical components, 3D segmentation and reconstruction, feature extraction, and matching based on a learned distance metric. In the following subsections, we illustrate each technical component.

4.3.1 3D Reconstruction

As mentioned in Section 4.2, direct recognition from depth images suffers from the problems of self-occlusion and sensor noise. Naturally, this leads to our new method of first building a smooth 3D model from the noisy input, and then performing recognition in 3D. There are existing approaches for obtaining high-quality models from noisy depth inputs, such as KinectFusion [Izadi *et al.*, 2011], which uses depth images from multiple views to filter noise. However, although it is possible to move the Kinect sensor while maintaining the garment static, it is more practical to let the garment be rotated by the robot arm, and thus preserve the Kinect sensor fixed in our settings. Unfortunately, this invalidates the KinectFusion's assumption that the scene is static. Therefore, in order to reconstruct the 3D model, we propose to first segment the garment from its background, and then use the truncated *Signed Distance Function (SDF)* [Curless and Levoy, 1996] to obtain a smooth 3D model, assuming that the rotation is sufficiently slow and steady such that the garment will not deform in the process. The details are illustrated below.

Segmentation. Before discussing the reconstruction algorithm, let us first define some notations. Given the intrinsic matrix F_d of the depth camera and the ith depth image I_i,

based on the pinhole camera model, we can compute the 3D coordinates of all the pixels in the camera coordinate system:

$$\begin{bmatrix} x_{ci} \\ y_{ci} \\ z_{ci} \end{bmatrix} = F^{-1} d_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}, \tag{4.1}$$

where (u_i, v_i) is the coordinate of a pixel in I_i, d_i as the corresponding depth, and (x_{ci}, y_{ci}, z_{ci}) is the corresponding 3D coordinate in the camera coordinate system.

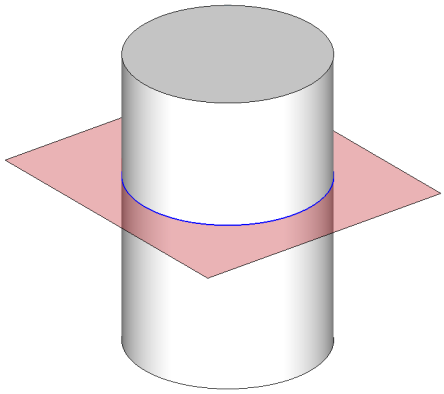
Therefore, for each depth input, we first back-project the depth input (u_i, v_i, d_i) into the 3D space with Equation (4.1), and perform segmentation in this 3D space. We ask the user to specify a 2D bounding box on the depth image $(u_{\min}, u_{\max}, v_{\min}, v_{\max})$ with estimation of the garment depth at $(d_{\min}, d_{\max})$. Given that the data collection environment is reasonably constrained, we find that even one predefined bounding box works well. Note that this bounding box is defined with regard to the Kinect sensor, which is fixed during the entire process. Then, we adopt all the pixels that have 3D coordinates within the bounding box as the foreground, resulting in a series of masked depth images $\{I_i\}$ and their corresponding 3D points, which are input in the reconstruction module.

SDF. Similar to KinectFusion [Izadi *et al.*, 2011], our 3D reconstruction algorithm also uses a volumetric representation with SDF to describe the location of a surface. We divide the given 3D bounding box into $d \times d \times d$ voxels, and calculate and maintain SDF for each voxel. Because the 3D bounding box is fixed in the entire process, the voxels are also static.

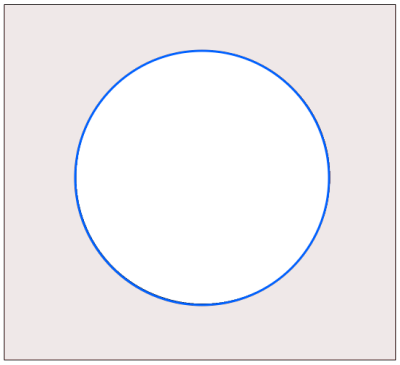
In the 3D space, for point $\boldsymbol{p}$ and surface $\mathcal{S}$, a Distance Function $\mathrm{DF}(\boldsymbol{p}, \mathcal{S})$ is defined as the distance between $\boldsymbol{p}$ and $\mathcal{S}$. By the definition of distance, $\mathrm{DF}(\boldsymbol{p}, \mathcal{S})$ is always non-negative. Assuming that a surface is closed (not necessarily watertight, but capable of distinguishing between "inside" and "outside"), $\mathrm{SDF}(\boldsymbol{p}, \mathcal{S})$ is defined as

$$\mathrm{SDF}(\boldsymbol{p}, \mathcal{S}) = \begin{cases} -\mathrm{DF}(\boldsymbol{p}, \mathcal{S}) & \boldsymbol{p} \text{ is inside } \mathcal{S}, \\ 0 & \boldsymbol{p} \text{ is on } \mathcal{S}, \\ \mathrm{DF}(\boldsymbol{p}, \mathcal{S}) & \boldsymbol{p} \text{ is outside } \mathcal{S}, \end{cases} \tag{4.2}$$

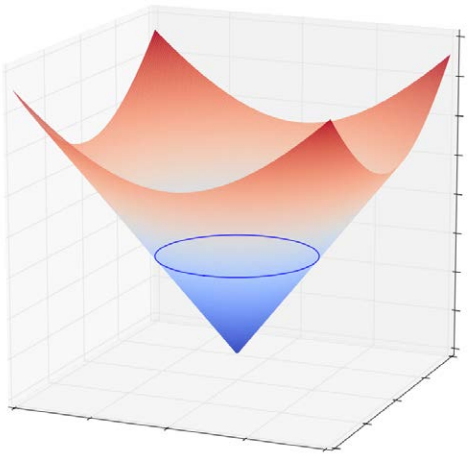
as shown in Figure 4.4. From the definition, we can see that a surface's location can be computed easily by searching for the zero points in the SDFs. On the other hand, given



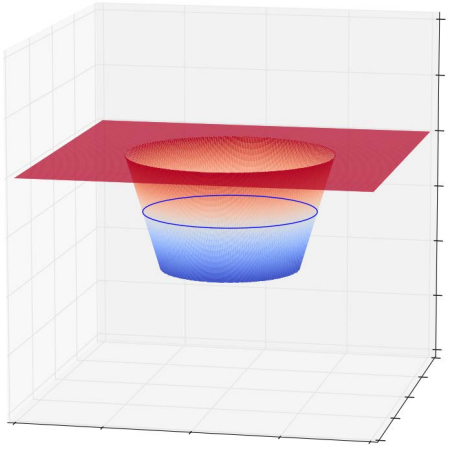
(a) A sample cylinder model with a conceptual crossing plane.



(b) The cross section of the 3D model at the plane.



(c) SDF at the crossing plane, appearing as the z axis.



(d) The Truncated SDF.

Figure 4.4: Illustration of the SDF definition. The blue circle shows the surface of the cylinder model at the crossing plane, where SDF is zero.

surface $\mathcal{S}$ and point $\boldsymbol{p}$, in order to compute SDF, we perform an approximation by not strictly computing the distance between $\boldsymbol{p}$ and $\mathcal{S}$, but imagining a virtual beam ℓ from the camera to point p, and directly using the Euclidean distance between $\boldsymbol{p}$ and the intersection of ℓ and $\mathcal{S}$, as Figure 4.5 shows. This introduces some error to the SDF values, but does not empirically affect the location of the SDF zero points in which we are essentially interested, and can accelerate the calculation significantly. At the same time, such raycasting approach

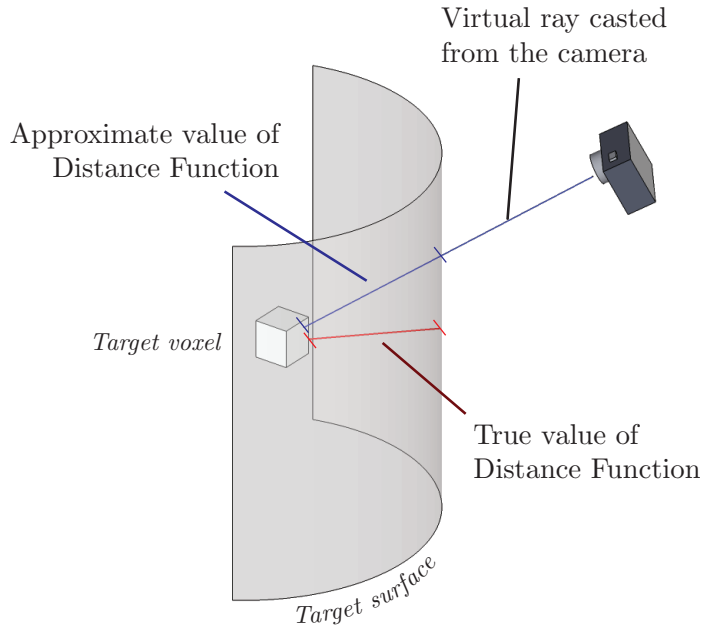


Figure 4.5: Illustration of the approximate SDF calculation. Note that the SDF sign changes from positive to negative when the virtual ray intersects the target surface.

can also determine the SDF sign with the prior knowledge that the camera must be outside the surface in our settings. Because a part of the garment is always occluded by itself, we discard negative SDFs with too large absolute values in order to prevent them from affecting the occluded parts. This results in a design of truncated SDF, as introduced in [Izadi *et al.*, 2011].

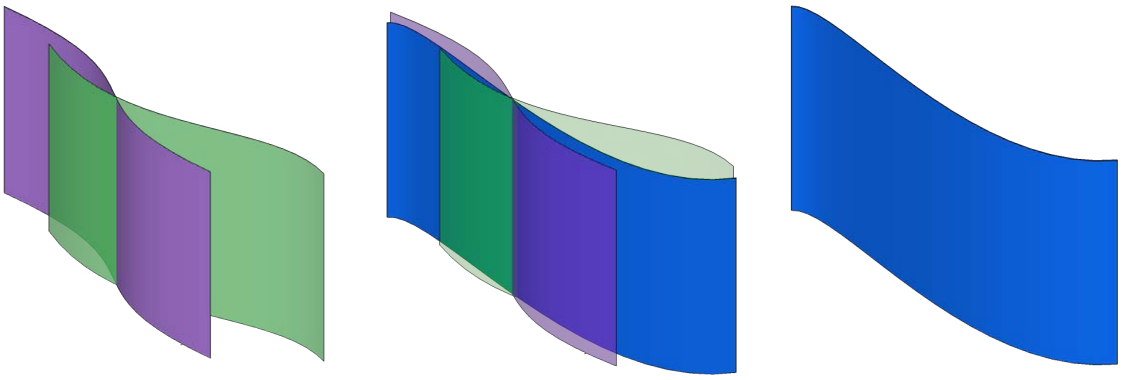
3D Reconstruction. Intuition of the 3D reconstruction algorithm is to calculate the average on the SDFs in order to smooth the noise from the Kinect sensor. For every depth image I_i, we can compute the SDF for each voxel and the garment surface by simple raycasting, as mentioned above. More specifically, because the locations or coordinates in the global 3D system of each voxel never change, for every voxel, we update the SDF per change of the captured garment surface, which is caused by the rotation of the robot arm. Because of the sensor noise, although the shape of the garment does not change in this gentle rotation, the captured shape of the surface still changes slightly, in addition to the rotation, as illustrated in Figure 4.6. If we make proper adjustments to the captured surface in order to compensate for rotation, the two shapes can be "averaged" to make their overlapped area smoother. Such averaging process is conducted by averaging the SDF

(a) (b)

Figure 4.6: The input and expected output of the 3D reconstruction. The yellow surface is the true surface of the garment. However, because of sensor noise, in (a), Kinect sees the purple surface. A moment later, when the garment is rotated, Kinect sees the green surface in (b). The goal of 3D reconstruction is to recover the true yellow surface from the noisy and inconsistently perceived purple and green surfaces.

value in each voxel, and can effectively smooth the surface, especially given that the sensor noise from Kinect is random in nature, and we can easily obtain a 360-degree view of the garment to make every region appear in multiple depth frames. Figure 4.7 illustrates this process.

In order to compensate for the rotation, 3D rigid registration is performed. As mentioned above, in any moment of the algorithm, we maintain a smoothed surface in the voxel representation, defined with the SDF zero points. A simple 3D rigid registration (e.g., Iterated Closest Point algorithm) between this smoothed surface and the captured noisy surface provides us with a rotation matrix. By multiplying the inverse of the matrix by the captured surface, we can align it to the smooth surface in memory, and then perform the SDF averaging steps. Note that every step described above, including registration, raycasting-based SDF calculation, and averaging, are parallel in nature, and thus they can be performed in real time. As shown in the following sections, the calculated SDF can also benefit feature extraction by dramatically expediting the process.



(a) (b) (c)

Figure 4.7: Concept illustration of the 3D reconstruction process: (a) shows the two captured surfaces from Figure 4.6 after registration. Because of sensor noise, although they are from the same garment, the recovered surfaces are not perfectly aligned; (b) shows SDF averaging that essentially calculates a surface average that results in the blue surface; and (c) shows the reconstructed surface only. Note that the data is simplified in this figure for clarity. The captured surfaces are much noisier and more complicated in shape; otherwise, 3D registration could not work.

4.3.2 Feature Extraction

Because of high expectation on speed, and for higher database scalability, we design a binary feature to describe the 3D models. Our feature design is inspired by the 2D Shape Context. In our method, the features are defined on a cylindrical coordinate system that fits to the hanging garment, unlike the traditional Shape Context, which uses a spherical coordinate system [Frome *et al.*, 2004].

For each layer, as shown in the top right of Figure 4.8, we *uniformly* divide the world space into (R rings) $\times$ (Φ sectors) in a polar coordinate system, with the largest ring covering the largest radius among all layers. The center of the polar coordinate system is determined as the mean of all points in the highest layer, which generally only contains the robot grasper. Note that because of physics, the gravity center of the garment must be on the same vertical line as the grasping point, which is also the center of the first layer. Therefore, this coordinate system can ensure the 3D points to be distributed in balance about the

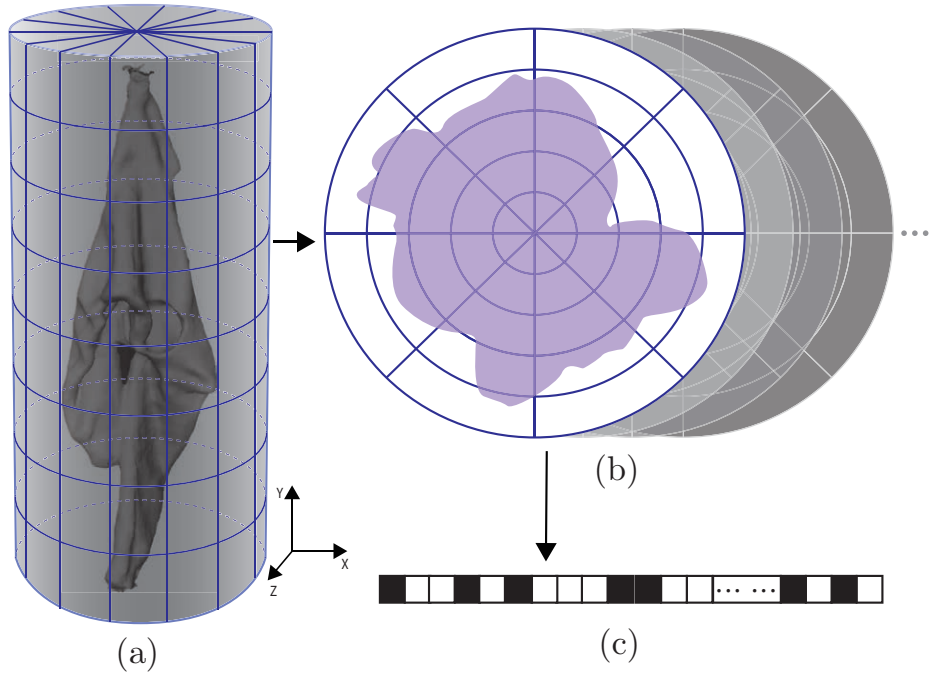


Figure 4.8: Feature extraction from a reconstructed mesh model: (a) indicates that a garment bounding cylinder is cut into several layers; (b) shows a set of layers (sections). We divide each layer into cells via rings and sectors; and (c) shows a binary feature vector collected from each cell. The details are described in section 4.3.2.

cylinder center.

It is worth noting that we perform uniform division, rather than the logarithmic division of r performed by Shape Context. The reason for Shape Context using the logarithm division of r is that the cells farther from the center are less important, which is not the case in our settings. For each layer, rather than performing a point count as in the original Shape Context method, we check the SDF voxel to which the center of the polar cell belongs, and enter one (1) in the cell if the SDF is zero or negative (i.e., the cell is inside the voxel); otherwise, we enter zero (0). Finally, all the binary numbers in each cell are collected in order (e.g., with ϕ increasing, and then r increasing), and concatenated as the final feature vector.

The insight behind this design is that in order to improve robustness against local surface disturbance because of friction, we include the 3D voxels *inside* the surface in the features. Note we do not need to perform time-consuming classification (e.g., ray tracing)

Algorithm 3: Feature extraction for pose estimation of deformable objects.

Input: Vertices of the mesh model $\Omega = \{v_i\}$, precomputed SDF(x, y, z),

Parameters: N = #layers, R = #rings, Φ = #sectors

Output: Corresponding feature vector $\boldsymbol{x} \in \mathbb{B}^{\Phi RN}$

$\boldsymbol{x} = \mathbf{0} \in \mathbb{B}^{\Phi RN}$;
Divide mesh Ω into l layers $\Omega_1, \Omega_2 \cdots$ in a top-down manner;
Origin = Mean$(\Omega_{1x}, \Omega_{1z})$;
$[\boldsymbol{r}, \Phi]$ = Polar (Origin, Ω_x, Ω_z) ;
$r_m = \max \boldsymbol{r}$;
for *each layer* Ω_i **do**
    for *each cell (ring, sector)* $\in \Omega_i$ **do**
        (x, y, z) = center of the cell ;
        if *SDF*$(x, y, z) \leq 0$ **then**
            $\boldsymbol{x}\left[\frac{rR\Phi}{r_m} + \frac{\phi\Phi}{2\pi}\right] = 1$;
        else
            $\boldsymbol{x}\left[\frac{rR\Phi}{r_m} + \frac{\phi\Phi}{2\pi}\right] = 0$;
        end
    end
end
return $\boldsymbol{x}$.

to determine whether each cell is inside the surface, but only need to search their SDFs, thus dramatically accelerating feature extraction. However, similar to Shape Context, the proposed feature is not rotation invariant, which should be address with a special matching scheme.

4.3.3 Matching Scheme

When matching against two shapes, we conceptually rotate one and adopt the minimum distance as the matching cost in order to provide rotation invariance. That is,

$$\text{Distance}(\boldsymbol{x}_1, \boldsymbol{x}_2) = \min_i \|R_i \boldsymbol{x}_1 \oplus \boldsymbol{x}_2\|_1, \tag{4.3}$$

where $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{B}^{\Phi RN}$ are the features to be matched, $\oplus$ is the binary XOR operation, and R_i is the transform matrix to rotate the feature of each layer by $2\pi/\Phi$. Recall that both features to be matched are compact binary codes. Thus, such conceptual rotation and hamming distance computation can be efficiently implemented by integer shifting and XOR operations; consequently, matching is even faster than the Euclidean Distance given reasonable Φs (e.g., $\Phi = 10$). A complete illustration of the feature extraction algorithm can be found in Algorithm 3.

Although this scheme works well among models from the same domain (e.g., matching simulated models against other simulated models), we still require more a sophisticated distance metric to manage cross-domain matching (i.e., matching reconstructed noisy models against noise-free simulated models). Similar to the previous chapter, we introduce a rank SVM to learn a weighted Hamming distance to address this issue. In particular, we place different weights for different bits of the binary feature. For cross-domain matching with $\boldsymbol{q}$ as the query feature, the best match is

$$\text{BestMatch}_{\boldsymbol{w}}(\boldsymbol{q}) = \arg\min_i \boldsymbol{w}^T \left(\hat{\boldsymbol{x}}_i \oplus \boldsymbol{q}\right), \tag{4.4}$$

where $\boldsymbol{q}$ is the feature vector of the query, i is the model index in the simulated database, and $\oplus$ is the binary XOR operation. $\hat{\boldsymbol{x}}_i = \hat{R}_i \boldsymbol{x}_i$ indicates the feature vector of the ith model, with $\hat{R}_i$ as the optimal R in Equation (4.3).

The insight here is granting our distance metric more robustness against material properties by assigning larger weights to the regions that are invariant to the material differences. In other words, this amplifies the features that are more intrinsic for the recognition task.

After illustrating the online testing stage, the remaining problems turn to constructing the database, and learning such a weight $\boldsymbol{w}$ in the distance, which are introduced in the next section.

4.4 Offline Database Simulation and Domain Adaptation

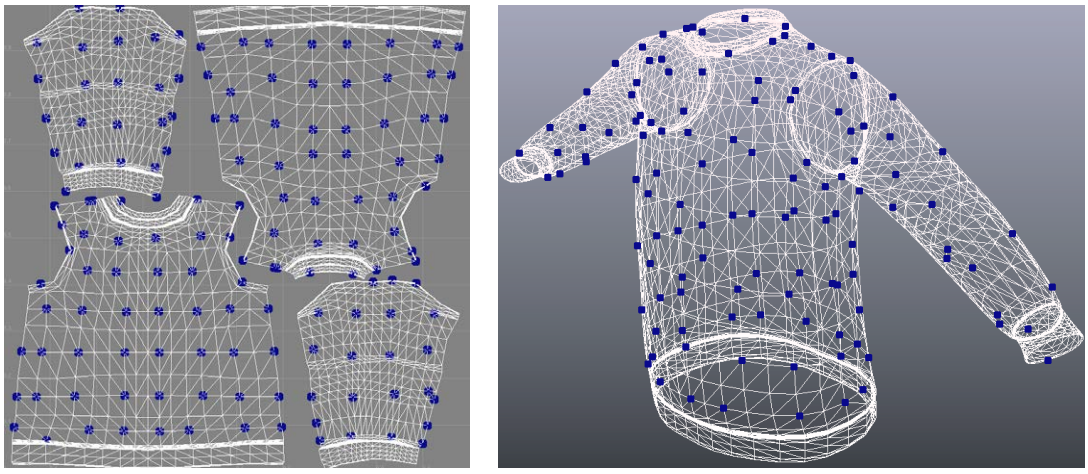
4.4.1 Model Simulation

Back to the overall problem of pose recognition of deformable objects, the main challenge lies in the enormous number of possible configurations. As illustrated in the introduction, we use a "data-driven" approach to resolve this issue. A large database is simulated offline to include different appearances of the models under different conditions. Then, the query is matched against the database, and the pose of the most similar database instances is adopted as the final output.

Therefore, the way in which this database is generated is critical to the quality of pose recognition. The expectation lies in two aspects. First, we expect it to be comprehensive so that reasonable variance in poses, garment categories, and materials can be covered. At the same time, it needs to be compact without much redundancy in order to avoid wasting time searching for identical models. Based on the expectation, we generate the database by varying the aforementioned variables based on commercial 3D models from garment manufacturers.

In particular, for each commercial garment model, we first select a set of points from the vertices of the model, and then use a physics engine to compute the mesh model in the stationary state as though a robotic arm were grasping each selected point. As mentioned in the introduction, because we use a data-driven approach and thus we do not have a concept of "class," it is not mandatory for the grasping points to generate consistent models among different garment instances. Therefore, we use a uniform sampling scheme. To select such grasping points, we first "unwrap" the mesh of the garment to a planar UV map, and then perform uniform sampling on it, as Figure 4.9 shows. The intuition of UV mapping is to obtain piecewise linear mapping (rotating and minimal stretching in our case) on the vertices such that the resulting planar faces can preserve garment size, with the final goal of making uniform sampling on the 2D map result in evenly distributed points on the garment surface in 3D space. We use the Global/Local Mapping proposed in [Liu *et al.*, 2008].

After the UV mapping step, we perform uniform sampling in terms of physical size on the mapped plane. The grasping points are selected as the closest vertices to the sampled



(a) (b)

Figure 4.9: An example of generating a set of grasping points for offline simulation: (a) shows the UV map of a sweater mesh model. Grasping points (the blue dots) are selected by finding the closest vertices to the uniformly sampled points; and (b) shows the original sweater model with the selected grasping points mapped back.

points, with one example shown in Figure 4.9. From the figure, we can see that the sampled grasping points are generally uniformly distributed. We employ a similar physics simulation method as described in our previous work [Li *et al.*, 2014b], with the difference that the final outputs are mesh models instead of rendered depth maps. This simulation stage results in a set of mesh models with different garment types, material properties, and grasping points to be matched against a reconstructed model from a Kinect sensor.

4.4.2 Domain Adaptation

It is worth noting that even for the same garment and grasping point by the robot, the ways in which different garments deform may be slightly different because of friction. In addition, considering that the models captured from 3D construction and those from simulation have inherently different noise levels, we need to learn a distance metric in order to provide additional robustness against such variances. Naïve Euclidean or Hamming distances are notorious for managing such challenges. Therefore, we adopt the "calibration" step to adapt knowledge from one domain (simulated models) to another (reconstructed models),

as commonly done in cross-domain retrieval applications.

Distance Metric Learning. To robustly learn the weighted Hamming distance, we use an additional set of mesh models collected from Kinect as *calibration data*. The collection settings are the same as described in Section 4.3.1, and only a small amount of calibration data is required for each category (e.g., 5 models in 28 poses for *sweater* models). To determine the weight vector $\boldsymbol{w}$, we formulate the learning process as an optimization problem of minimizing the empirical error with a large-margin regularizer:

$$\begin{aligned} &\min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_j \xi_j \\ &\text{s.t. } \boldsymbol{w}^T(\hat{\boldsymbol{x}}_i \oplus \boldsymbol{q}_j) < \boldsymbol{w}^T(\hat{\boldsymbol{x}}_k \oplus \boldsymbol{q}_j) + \xi_j, \\ &\quad \forall j, \forall y_i = l_j, y_k \neq l_j, \\ &\quad \xi_j \geq 0, \end{aligned} \tag{4.5}$$

where $\hat{\boldsymbol{x}}_i$ is the orientation-calibrated feature of the ith model (from the database), with y_i as the corresponding ground truth label (i.e., the pose index). $\boldsymbol{q}_j$ is the extracted feature of the jth training model (from Kinect), with l_i as the ground truth label. We want to minimize $\sum_i \xi_i$, which indicates how many wrong results are provided by the learned metric $\boldsymbol{w}$, with a quadratic regularizer. C controls how much penalty is given to wrong predictions. Note that this is slightly different from regular SVM formulation in that we adopt the class with the smallest score, instead of the largest score, as the prediction result. This is reflected in the $<$ sign in the constraint, rather than the $>$ for regular SVM formulation.

This is a non-convex and even non-differentiable problem. Therefore, we employ SVM$^{\text{rank}}$ [Joachims.T., 2002] to obtain an approximate solution using the cutting-plane method.

Knowledge Transfer. Given the learned $\boldsymbol{w}$, we use Equation (4.4) in the testing stage to obtain the query model NN. We directly adopt the NN grasping point, which is known from the simulation process, as the final prediction.

In summary, the contributions of the proposed approach lie in three aspects. First, we use a data-driven approach to address the pose recognition problem of deformable objects. Second, a variant of KinectFusion is employed to address the problem of dynamic scene reconstruction in a robotics setting. Third, a novel compact feature is proposed, with a matching scheme based on a learned weighted Hamming Distance.

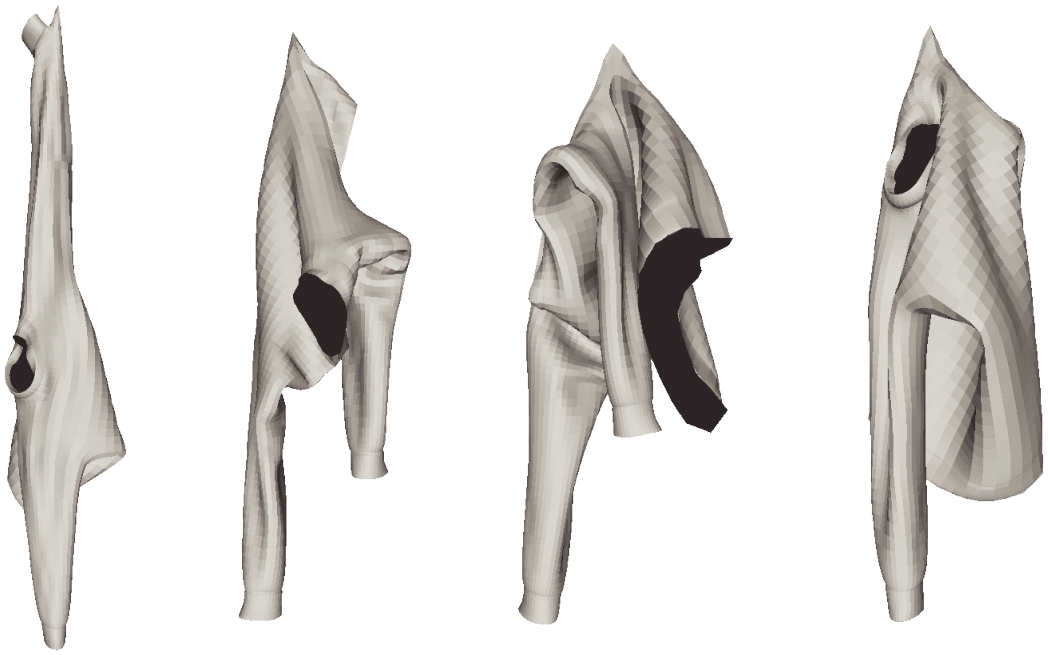


Figure 4.10: Example of the 3D models simulated from the commercial garment models and a physics engine. A sweater in different poses is shown here.

4.5 Experiment Results

To test the effectiveness of the proposed method and justify the components, a series of experiments are conducted. We collected a dataset of various garment categories from practical settings, and then quantitatively compared the results with the state-of-the-art approach for depth-based garment pose estimation described in [Li *et al.*, 2014b]. To evaluate our results, the geodesic distance on the garment between the predicted grasping point and the ground truth is computed, along with the running time. Experiment results demonstrate that our method can achieve both higher accuracy and acceleration with orders of magnitude. A demonstration video is provided at https://www.youtube.com/watch?v=awhaJ_jcjnE.

4.5.1 Dataset

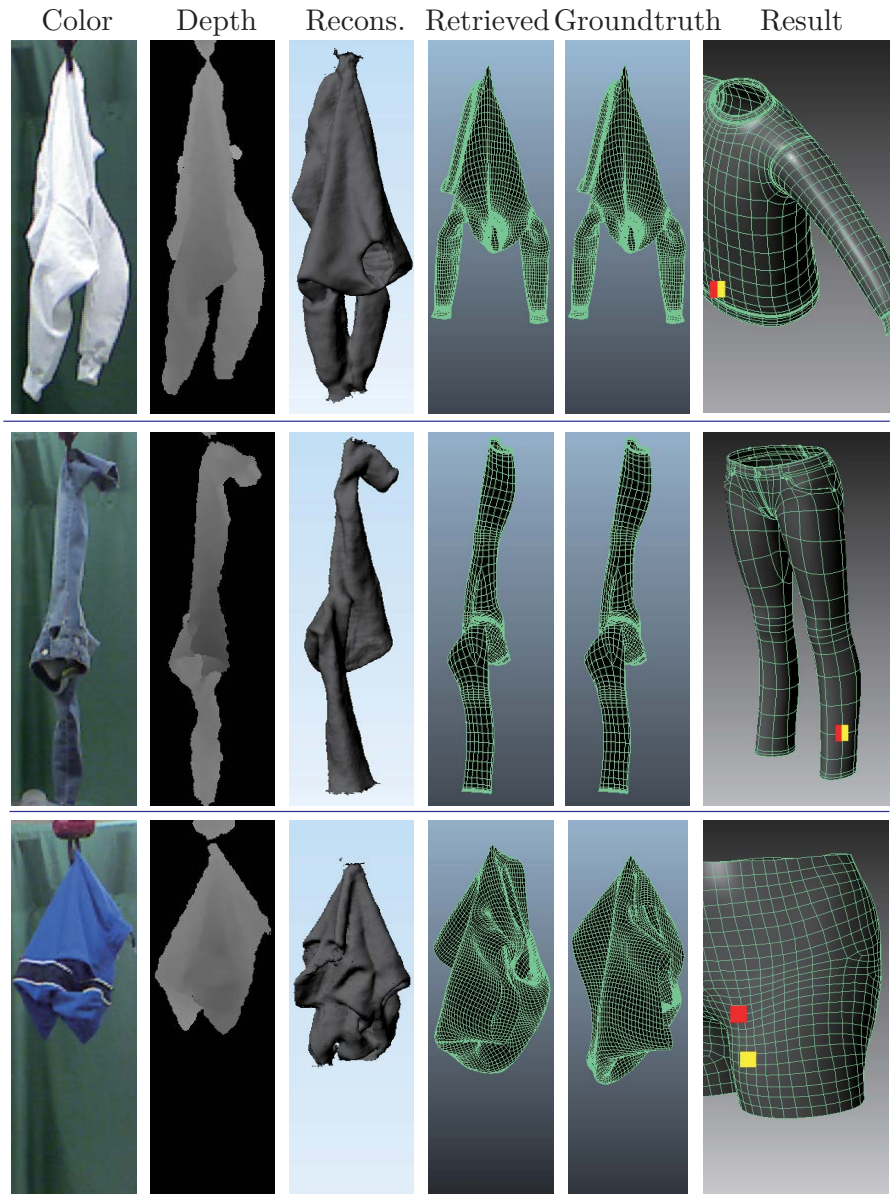
We collect a dataset for general evaluation of pose recognition of deformable objects based on depth image as input. The dataset consists of three parts: training, testing, and calibration.

The training set is the simulated mesh models of different types of garments in different poses, as introduced in Section 4.4.1. We purchased three categories of commercial-quality mesh models – *sweaters*, *jeans*, and *shorts*, and simulate each model with $200-240$ grasping points (different garment types have different numbers of grasping points depending on the surface area and model complexity). An example of the simulated 3D model is shown in Figure 4.10. Because all of our garment candidates are symmetric front and back, left and right, we only adopt those grasping points on 1/4 of the surface over the entire garment in order to remove duplicates; this results in 68 grasping points, each with a corresponding simulated mesh model. To collect the testing set, we use a Baxter robot equipped with two arms with seven degrees of freedom. A Kinect sensor is mounted on a horizontal platform at a height of 1.2 meters to capture the depth images, as shown in Figure 4.1. We purchased one real garment for each category for the testing set collection. For each garment, we collect data at the same grasping points as the training set, and then use our 3D reconstruction algorithm as introduced in Section 4.3.1 to obtain their mesh models. For each grasping point of each garment, the robot rotates the garment 360 degrees for approximately 10 seconds, while the Kinect captures at 30fps, which provides us approximately 300 depth images for each garment/pose. This results in a test set of 68 mesh models with raw depth images.

Given that we also need to learn/calibrate a distance metric from additional Kinect data, we collect an additional small amount of data with the same settings as the calibration data, thus only collecting 5 poses for each garment, whose positions are randomly selected from the 68 given poses. A weight vector $\boldsymbol{w}$ is then learned from this calibration data for each type of garment as introduced in Section 4.4.2.

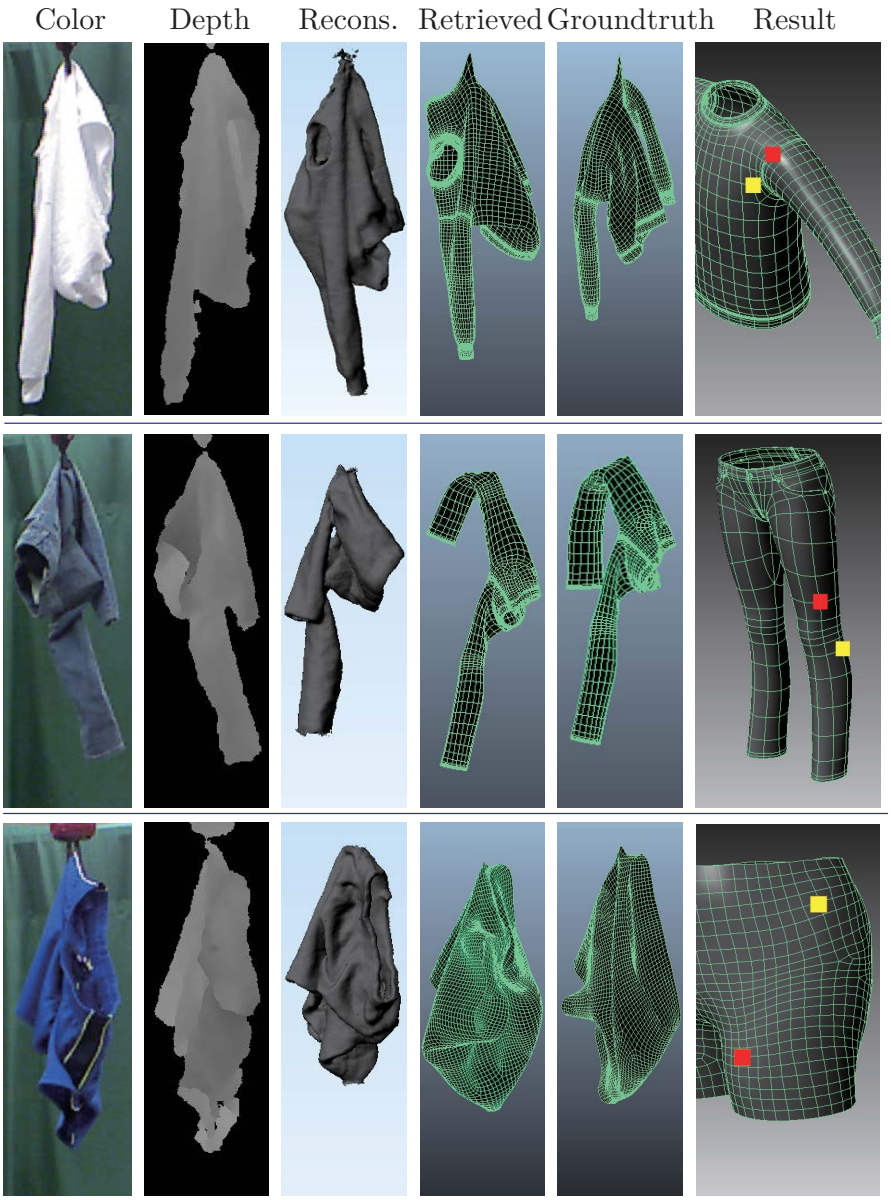
4.5.2 Qualitative Evaluation

We demonstrate some of the recognition results in Figure 4.11 in the order of color image, depth image, reconstructed model, predicted model, ground truth model, predicted grasping point (red), and ground truth grasping point (yellow) on the garment. From the figure, we can see that our 3D reconstruction can provide us with good-quality models for a fixed camera that captures a dynamic scene. Furthermore, our shape retrieval scheme with



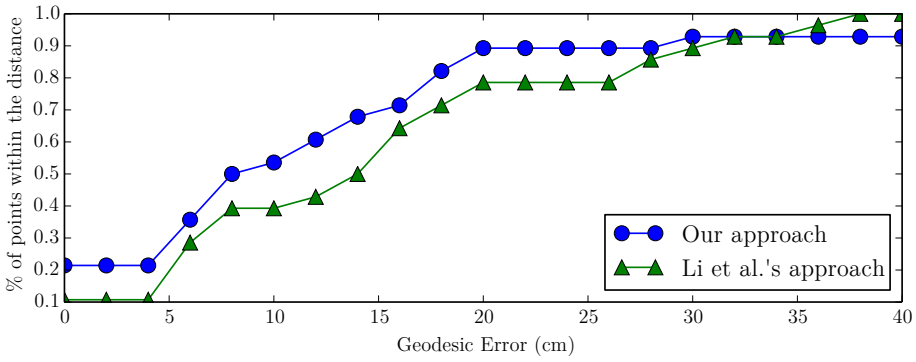
(a) Visual examples of the pose recognition results of our method.

learned distance metrics can also provide reasonable matches for the grasping points. Note that our method can output a mesh model of the target garment. This is critical for subsequent operations, such as path planning and object manipulation.

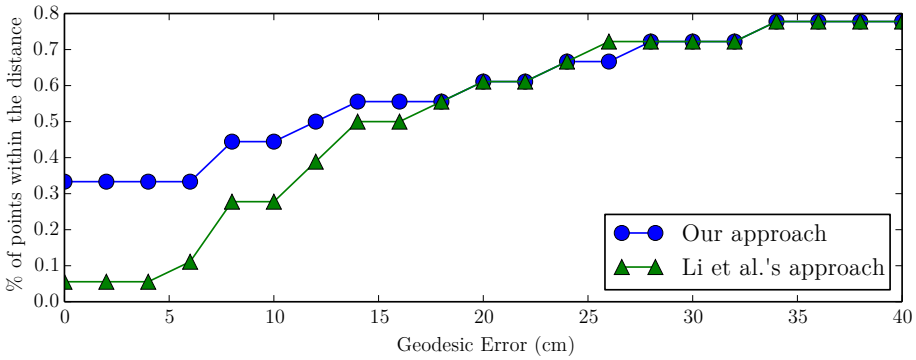


(b) (Continued) Visual examples of the pose recognition results of our method.

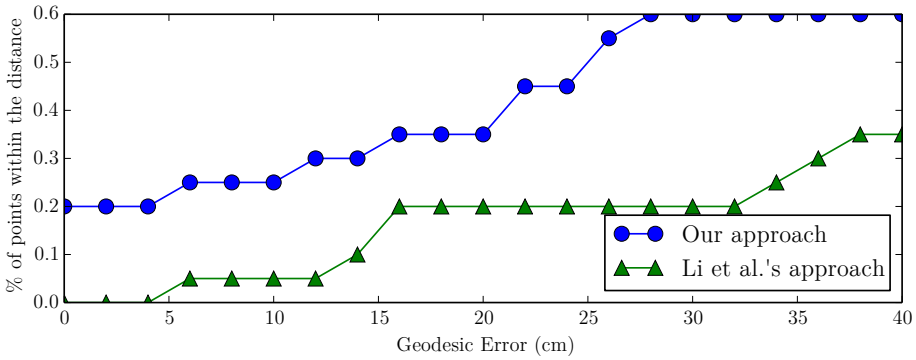
Figure 4.11: The garment is lifted via a gripper of the Baxter robot. From left to right, each example shows the color image, input depth image, reconstructed model, matched simulated model, ground truth simulated model, and the predicted grasping points (red) marked on the model with the ground truth (yellow). The example shown in the bottom right is considered a failure example, which may have occurred because of the uninformative deformation shape. Note that our method does not use any color information.



(a) Sweaters



(b) Jeans



(c) Shorts

Figure 4.12: Quantitative comparison of the proposed method and state-of-the-art algorithm [Li *et al.*, 2014b] on different garment categories.

4.5.3 Quantitative Evaluation

First, we introduce some implementation details of our method, and then provide the quantitative evaluations.

Implementation Details. In the 3D reconstruction, we set $X = 384$, $Y = Z = 768$ voxels, and the resolution of the voxels at 384 voxels per meter in order to obtain a trade-off between resolution and robustness against sensor noise. In the feature extraction, our implementation adopts $R = 16, \Phi = 16, N = 16$ in the feature extraction as an empirically good configuration. That is, each mesh model provides a $16 \times 16 \times 16 = 4,096$ dimensional binary feature. We set the penalty at $C = 10$ in Equation (4.5).

Geodesic Error. As mentioned above, we purchased three real garments and collected data on different poses using the Kinect sensor and a Baxter robot. That is, for each garment, we have a set of 3D models, each of which is collected under a certain pose. For each 3D model, we have a predicted grasping point as our algorithm's output, and a ground truth grasping point that is known from the data collection process. For evaluation, we compute the geodesic distance of the predicted point and the ground truth for a specific 3D model, which we refer to as *Geodesic Error* in the following text. The distribution and mean of the Geodesic Error of all tested models from a test garment are used as the evaluation protocol.

We also compare our method with state-of-the-art algorithms in depth-based pose recognition of deformable objects [Li *et al.*, 2014b]. Li's approach performs independent pose recognition on each depth input, and then uses majority voting to obtain the final result. A comprehensive global classifier is trained to process each depth image. Because this method uses depth images instead of a 3D model as input, for fair comparison, we input all test depth images on which we reconstructed the 3D model to Li's method, for each pose of each garment, and measure both the Geodesic Error and running time.

A comparison of the distribution of the Geodesic Error is plotted in Figure 4.12. The x-axis shows the Geodesic Error, and the y-axis shows the accumulative percentage of the test cases (i.e., the garmentpose combination). The top row shows the result of a *sweater* as input, with maximum distance between any two grasping points at 75 cm. The middle and bottom rows show a pair of *jeans* and *shorts*, with maximum distance between

Garment	State-of-the-art Method [Li *et al.*, 2014b]	Our Method (No DA)	Our Method
Sweater	16.05	18.64	**13.61**
Jeans	10.89	14.31	**9.70**
Shorts	22.44	25.78	**17.82**

Table 4.1: Comparison on average Geodesic Error for different garment types. The unit is cm. Our Method (No DA) means our method without domain adaptation.

any two grasping points at 65 cm and 51 cm, respectively. We can clearly see that our method outperforms the state-of-the-art method [Li *et al.*, 2014b] in all the tested garment types. Our method benefits from the 3D reconstruction step, which reduces sensor noise and integrates the information of each frame to a comprehensive model, and thus leads to better decisions. Among the three types of garments, *shorts* recognition is not as accurate as the other two. One possible reason is that shorts are more symmetric than the other garment categories, and thus introduce more ambiguity in terms of shapes from different grasping points. Even for human observers, it is difficult to determine the correct pose by observing the reconstructed model or depth images.

To justify the component of domain adaptation (i.e., the $\boldsymbol{w}$ in the weighted Hamming distance), we also test our approach using the naïve Hamming Distance in Equation (4.3) as the distance metric. The mean of the Geodesic errors of different methods on different garments is listed in Table 4.1. We can see that the lack of domain adaptation degenerates performance and makes it worse than the state-of-the-art approach, which verifies our motivation for introducing cross-domain learning. When combined with the learned distance metric, our method can achieve lower Geodesic Error than the results from [Li *et al.*, 2014b].

Running Time. In addition, we compare the processing time of our method with the state-of-the-art method [Li *et al.*, 2014b] that uses individual depth images. The time is measured on a PC with Intel i7 3.0 GHz CPU, and listed in Table 4.2. We can see that our method demonstrates acceleration in orders of magnitude against the depth image-based

Garment	Li's Method [Li *et al.*, 2014b]	Our Method
sweater	46	**0.30**
jeans	42	**0.20**
shorts	71	**0.22**

Table 4.2: Average running time in seconds of our method and state-of-the-art method, with the input of different garment types.

	Sweater	Jeans	Shorts
Accuracy	85.7%	70.0%	90.0%

Table 4.3: Classification accuracy of our method on the task of garment categorization.

method, which verifies our advantages from efficient 3D reconstruction, feature extraction, and matching. The main bottleneck of Li's method is SIFT extraction and a sparse coding process, which require solving a series of linear programming problems. Our method also shows less variance in running time, especially on the *shorts* input, whereas the running time of Li's method heavily depends on the number of extracted SIFT points, especially when the depth input has rich textures or noise.

4.5.4 Generality to Unseen Garments

Though we use a relatively small garment database for our experiments, we notice that our simulated models can be generalized to recognize similar, but unseen, garments. For example, *long-sleeve shirts* and *jackets* can be considered similar garments to our *sweater* model. In addition, *knit pants* and *suit pants* are similar to our *jeans* model. Although they are made of different materials, the manner in which they deform is similar to our training models in some poses. Figure 4.13 shows some additional examples of recognizing poses of unseen garments using the same weight $\boldsymbol{w}$ learned on our original dataset. We also notice that some decorations exist on those garments, such as pockets or shoulder boards; however, our method is sufficiently robust to ignore these subtler features.

Figure 4.13: Sample results of applying our method to unseen garments. Each group of results shows the color image, reconstructed model, predicted grasping points (red), and the ground truth (yellow) marked on the model from left to right.

4.5.5 Application on Category Classification

A basic assumption of our method is that the garment category is known in advance, so that we only need to search within the training data of the same category. However, our method for NN search can also be used to predict garment category. Therefore, we also test the performance of our method on this task by searching NN within the entire training set instead of only the part with the same category. By adopting NN's category as the prediction, we can compute the classification accuracy for evaluation, as listed in Table 4.3. We can see that our method can produce reasonable categorization results, even without special optimization on the task.

4.6 Conclusion and Future Work

In this chapter, we discussed the problem of pose recognition of deformable objects on a robotic platform. In particular, we explored pose recognition of garments using a low-cost depth sensor. First, we commanded the robotic arm to rotate the garment 360 degrees, and then reconstructed a 3D model from the captured depth images in order to resolve the incomplete model problem from low-cost depth sensors. To address the unique challenges of noisy input and constricted time budget, we extracted a compact binary 3D features derived from the 3D shape context of the model. The feature representation was then matched against a simulated database with a learned distance metric to find NN, whose grasping point was adopted as the prediction. Experiments demonstrated the superior effectiveness and efficiency of our approach against state-of-the-art methods. These experiments assumed that each garment category was already known. We believe that we can learn the category and the pose using an extension of this method.

This work can be extended in several directions. First, it is possible to add garment color and texture as features to further improve recognition accuracy. Second, an accurate and fast method for pose estimation of deformable objects may benefit a variety of practical applications, such as clothes folding, which would be easier once the robot has an accurate mesh model and knows the point that it is grasping. Last, while this chapter is on the vision techniques that benefit from robotics application, it is also possible to use robotics to simplify vision tasks. For example, simple actions, such as shaking robotic arms, may help resolve visual variance from friction.

Part II

Extension to 2D Data

Chapter 5

Extension to 2D Applications

5.1 Introduction

In the previous chapters, we introduced a comprehensive framework that may benefit the key problems in content-based 3D shape analysis. Examples include cross-domain retrieval, scene understanding, and pose recognition. Although the approaches show promising performance in the 3D applications, there is actually no factor that prevents them from being applied to traditional 2D vision problems. Therefore, inspired by success in the 3D field, we explore the possibility of extending the approach to 2D applications in this chapter, and compare it with traditional and state-of-the-art approaches.

In particular, we are interested in low-level vision applications, such as *image deblurring* and *image denoising*. With the rapid development of camera phones, it is extremely common for regular consumers to use camera phones rather than professional cameras (either DSLR or point-and-shoot) to capture daily photos. The small size of the sensor makes such on-phone cameras easy to carry; however, it suffers from the problem of high noise levels and low sensitivity. This problem is especially severe in dark environments, where short exposure time and limited ISO cause motion blur. For this reason, in an attempt to solve this problem from a post-processing perspective, image deblurring has attracted much attention in computer vision and computational photography. However, state-of-the-art methods that can achieve impressive deblurring effects are usually extremely slow. In this chapter, we propose using the MRF formulation with approximate inference to accelerate existing image

deblurring algorithms in order to make such algorithms practical for consumer applications.

Given that low-level vision is less related to the 3D retrieval and understand field, we first make a brief introduction of its background for the convenience of readers. Then we illustrate the current challenges in this field, based on which we argue that the bottleneck of the problem can be addressed by efficient retrieval techniques.

5.1.1 Background of Image Deblurring

Image Deblurring. As the name states, given a blurry image, the act of recovering a clear image from the input is called image deblurring. From the perspective of the physics model, the blurry image can be viewed as the result of adding noise to the convolution of the clear image and a *motion kernel*, which is essentially the motion of the camera. Depending on whether the motion kernel is given, image deblurring is divided into two categories, blind and non-blind. When the motion kernel is unknown, the deblurring problem is treated as "blind." When the motion kernel is given, either from an estimation algorithm or from other inertia sensor, the problem is considered "non-blind." Whereas non-blind deblurring is often a basic unit in blind deblurring, we focus mainly on the non-blind deblurring problem in this chapter.

Given the presence of noise, as shown in the next section in mathematical form, the non-blind deblurring problem is *ill-posed*, which means it is an under-constrained system, and there are multiple solutions that satisfy the convolution model. To further constrain the problem and make the resulting solution consistent with human perception, introducing an *image prior* is standard practice. For any image, an image prior describes the probability of the image being from a *natural* photo. For example, an image with pure white noise has extremely low probability under a good image prior, and a photo captured from the real world should have high probability. From a mathematics perspective, an image can be viewed as a vector that lies in an extremely high dimensional space. However, the images we see in the real world are not scattered everywhere in this space, but distributed in a small subset that satisfies some special properties.

In the settings for image deblurring, an image prior penalizes the solutions that are not likely to be seen in the real world, regardless of whether the solutions satisfy the convolution

model. Then, the final deblurring model attempts to seek an image that satisfies both "naturalness" and consistency between the convolved result and the observation. From a Bayesian perspective, the convolution model describes the *likelihood* given the clear image and the observed output. Combined with the *priori* described with the image prior, we attempt to recover the clear image with a Maximize-A-Posteriori (MAP) problem. Given that the likelihood model is based on physics principles, the image prior is the major factor that distinguishes one deblurring model from another. Now let us see some typical image priors that have been proven effective in image deblurring.

Image Priors. One classic family of image priors is defined on image gradients. These priors assume that the magnitude of image gradients follows certain distributions. Examples include exponential distributions [Yang *et al.*, 2009], hyper Laplacian distributions [Krishnan and Fergus, 2009], or a mixture of Gaussians [Fergus *et al.*, 2006]. These priors are computationally efficient because both the value (for evaluation) and the gradient (for MAP optimization) can be computed using simple gradient filters. The Half-Quadratic Splitting optimization framework [Krishnan and Fergus, 2009], to be introduced in the next section, is also used to further decouple the variables, and thus accelerate the optimization process. However, because of the extremely small spatial support of gradient filters, gradient priors have severe limitations in their description capabilities, and cannot faithfully capture image structures.

To address this issue, image priors with larger spatial support have been proposed. One popular direction is to formulate the image restoration problem within a Conditional Random Field (CRF) framework, and associate nonadjacent pixels by connecting them in the field. Field of Experts (FoE) [Roth and Blacky, 2009] is a typical example that constructs a CRF on all pixels of the input image, and defines priors on the cliques. Whereas gradient priors can be viewed as special cases where a clique is only a pair of adjacent pixels, FoEs have significantly larger cliques than the pixel pairs. This design provides much larger spatial support than the gradient priors. However, it also considerably increases the complexity of the field structure, and thus suffers from the optimization tractability problem. To this end, approximate inference is often adopted in practice, but continues to result in slow speed.

Patch-based Image Priors. Other challenges, such as pixel saturation [Whyte *et al.*, 2011] and outliers [Cho *et al.*, 2011], exist in the image restoration field; however, the critical problem of image deblurring continues to be how to model the image prior. An emerging trend is defining probabilistic priors on image patches (i.e., *probabilistic patch-based prior*) without explicit connections among pixels, despite natural pixel-sharing between adjacent patches. An example is EPLL [Zoran and Weiss, 2011] that shows state-of-the-art performance on image deblurring and competitive results on denoising and inpainting. This type of priors can preserve the efficient Half-Quadratic Splitting optimization framework, thus avoiding the slow and approximate inference of CRFs. Unfortunately, they still require an excessive amount of computation, which severely limits their practical usage. For example, the non-blind deconvolution method of Zoran and Weiss requires tens of minutes for one megapixel image [Zoran and Weiss, 2011] on a workstation. This becomes even worse for blind deconvolution, where the non-blind deconvolution component needs to be applied repeatedly, and typically requires hours to finish [Sun *et al.*, 2013]. In the following text, we show that the bottleneck in evaluating and optimizing such patch-based priors actually has exact settings of a retrieval problem. Furthermore, building an index on the priors can dramatically improve optimization speed with unnoticeable quality compromise.

5.1.2 Image Prior Indexing

The speed issue of the non-gradient priors has already attracted much attention, and various approaches have been proposed to address it. For the random-field-based priors, Jancsary *et al.* [Jancsary *et al.*, 2012b] restricted the potential functions of the CRF to Gaussians, so that much more efficient inference algorithms could be used. To compensate for the performance drop in limiting forms of potential functions, regressors are trained to discriminatively determine the mean and covariance of the potential functions. This results in an RTF formulation [Jancsary *et al.*, 2012a] that provides state-of-the-art performance for denoising and inpainting. This method can be interpreted as using random forests to pre-index a flexible prior defined on cliques in the random field. Recently, a similar idea of using pre-trained tree structures to efficiently construct an RTF has been used in deblurring [Schmidt *et al.*, 2013].

However, for the newly emerged probabilistic patch-based prior direction, scant exploration has been performed in expediting the associated optimization process, albeit such expedition can potentially benefit a series of practical applications and possibly reveal more insights on patch-based priors. Inspired by the pre-indexing view of RTFs, we propose to pre-index the probabilistic patch-based priors to accelerate their optimization. Furthermore, we adopt EPLL as an example to demonstrate the novel prior indexing approach.

Challenges. However, indexing the patch-based priors is fairly challenging, and the existing approaches are not readily extended to their unique settings. First, the image patches lie in a relatively high dimensional space. This makes straightforward lookup tables, as used in the hyper Laplacian prior [Krishnan and Fergus, 2009], not capable of working properly because of significant memory consumption. Content-based hashing is known to be compact and fast, especially for high-dimensional data, but its accuracy is insufficient for image restoration tasks. Second, from the motivation of acceleration, we have a constricted budget in the tree depth, and natural image patches have special structures that are different from common distributions. Therefore, as shown shortly, existing tree indexing structures, as used in [Jancsary *et al.*, 2012b; Schmidt *et al.*, 2013], also suffer from the computational cost problem in our settings.

To address these challenges, we adopt the formulation from Chapter 2 to search for the proper prior to use. In particular, we observe that the EPLL prior, which has the form of mixture of Gaussians, can be well approximated with one single Gaussian in each optimization step, although the selected Gaussian may be different in different steps. Therefore, we use the MRF formulation to efficiently infer the Gaussian to use for each optimization step. A unary potential is discriminatively determined with a tree indexing structure, whose training algorithm is specifically tailored for the patch distributions of natural images for efficient computation. Moreover, a pairwise potential is designed to further take advantage of the spatial consistency. In the end, an approximate MRF inference method [He *et al.*, 2010] is used to maintain high speed. We take image deblurring as the primary application because of the state-of-the-art performance EPLL appears on it. Complexity analysis and experiment results show that our indexing approach leads to significant acceleration, while preserving the power of patch-based priors. Qualitative experiments also demonstrate

the potential of the proposed indexing approach in deblurring real-life photos and image denoising.

Our main technical contributions include:

1. A novel framework for indexing patch-based natural image priors using decision trees (Section 5.3).

2. An efficient way for constructing the indexing tree by exploring the special structure of the parametric patch prior components (Section 5.4).

Section 5.2 introduces the mathematical formulation of non-blind image deblurring and EPLL with our general framework. Section 5.5 discusses the experiment, and Section 5.6 summarizes the chapter with discussions on future work.

5.2 Observations and Our General Framework

5.2.1 Background and Notations

Before introducing our approach in more detail, we first provide a formal description of the problem. Image degradation is typically modeled as

$$\boldsymbol{y} = A\boldsymbol{x} + \boldsymbol{n}, \tag{5.1}$$

where $\boldsymbol{y}$, $\boldsymbol{x}$, and $\boldsymbol{n}$ are vectors that represent an observed blurry image, its latent image to be recovered, and noise, respectively. For deconvolution, A is a convolution matrix, and for denoising, $A = I$, an identity matrix.

The restored image $\hat{\boldsymbol{x}}$ can be estimated using MAP estimation.

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, p(\boldsymbol{x}|\boldsymbol{y}) = \underset{\boldsymbol{x}}{\operatorname{argmax}}\, p(\boldsymbol{y}|\boldsymbol{x})p(\boldsymbol{x}), \tag{5.2}$$

where $p(\boldsymbol{y}|\boldsymbol{x})$ is the likelihood, and $p(\boldsymbol{x})$ is the image prior introduced in the previous section. Assuming that the noise is a Gaussian noise, $\hat{\boldsymbol{x}}$ can be estimated by minimizing the energy, which is the negative log posterior,

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\{ \frac{\lambda}{2} \|\boldsymbol{y} - A\boldsymbol{x}\|^2 - \log p(\boldsymbol{x}) \right\}, \tag{5.3}$$

where λ is a parameter to control the restoration strength.

Gaussian Mixture Model (GMM)-based Patch Prior proposed by Zoran and Weiss [Zoran and Weiss, 2011] is defined as:

$$p(\boldsymbol{x}) \propto \prod_i p(\boldsymbol{x}_i) = \prod_i \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_i | \boldsymbol{\mu}_k, \Sigma_k), \tag{5.4}$$

where i is a pixel index, and $\boldsymbol{x}_i$ is a patch centered at the i-th pixel. A GMM $\{\boldsymbol{\mu}_k, \Sigma_k, \pi_k\}_{k=1}^{K}$ is learned from a large collection of natural image patches, with k as the index of the Gaussian components, and $\boldsymbol{\mu}_k$, Σ_k, and π_k as the mean, covariance, and weights of the Gaussians, respectively.

Directly optimizing Equation (5.3) is difficult because of the coupling of the two terms. For efficient optimization, auxiliary variables $\{\boldsymbol{z}_i\}$ can be introduced as in [Zoran and Weiss, 2011], reformulating Equation (5.3) with a popular half-quadratic scheme as

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmin}} \Big\{ \frac{\lambda}{2} \|\boldsymbol{y} - A\boldsymbol{x}\|^2 + \frac{\beta}{2} \sum_i \|\boldsymbol{z}_i - \boldsymbol{x}_i\|^2 - \sum_i \log p(\boldsymbol{z}_i) \Big\}, \tag{5.5}$$

Optimization starts from a small value of β and develops by fixing $\boldsymbol{x}$ to solve for $\boldsymbol{z}$ (the z-step), and fixing $\boldsymbol{z}$ to solve for $\boldsymbol{x}$ (the x-step) alternatingly, with increasing β values. When β becomes sufficiently large, the optimal $\hat{x}$ and $\hat{z}$ are nearly the same with negligible difference.

Bottleneck. When $\boldsymbol{z}$ is fixed in an iteration, the problem turns to

$$\hat{\boldsymbol{x}}_i = \underset{\boldsymbol{x}_i}{\operatorname{argmin}} \Big\{ \frac{\lambda}{2} \|A\boldsymbol{x}_i - \boldsymbol{y}\|^2 + \frac{\beta}{2} \sum_i \|\boldsymbol{x}_i - \boldsymbol{z}_i\|^2 \Big\}. \tag{5.6}$$

This is essentially a quadratic programming problem with a close-form solution, but the z-step is a much slower optimization process. With a fixed $\boldsymbol{x}$, the z-step attempts to solve the following problems for all is:

$$\hat{\boldsymbol{z}}_i = \underset{\boldsymbol{z}_i}{\operatorname{argmin}} \left\{ \frac{\beta}{2} \|\boldsymbol{z}_i - \boldsymbol{x}_i\|^2 - \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{z}_i | \boldsymbol{\mu}_k, \Sigma_k) \right\}, \tag{5.7}$$

which is a complex and expensive non-linear optimization problem that involves many matrix multiplications. To alleviate the optimization difficulty, Zoran and Weiss [Zoran

and Weiss, 2011] only used the Gaussian component with the largest conditional likelihood $p(k|P_i\boldsymbol{x})$, instead of all the components, to perform optimization. More specifically, with the chosen Gaussian $\hat{k}_i$, Zoran and Weiss solved the following simplified problem:

$$\hat{\boldsymbol{z}}_i = \underset{\boldsymbol{z}_i}{\operatorname{argmin}} \left\{ \frac{\beta}{2} \|\boldsymbol{z}_i - \boldsymbol{x}_i\|^2 - \log \mathcal{N}(\boldsymbol{z}_i | \boldsymbol{\mu}_{\hat{k}_i}, \Sigma_{\hat{k}_i}) \right\}. \tag{5.8}$$

However, this approximation still requires a significant amount of computations. In particular, to find $\hat{k}_i$ for the i-th patch, we need to compute

$$\begin{aligned} \arg\max_k p(k|\boldsymbol{x}_i) &\propto p(\boldsymbol{x}_i|k)p(k) \\ &= \int_{\boldsymbol{z}_i} p(\boldsymbol{x}_i|\boldsymbol{z}_i)p(\boldsymbol{z}_i|k)p(k) \\ &= \int_{\boldsymbol{z}_i} \mathcal{N}(\boldsymbol{x}_i|\boldsymbol{z}_i, \beta^{-1}I)\mathcal{N}(\boldsymbol{z}_i|\mu_k, \Sigma_k)\pi_k \\ &= \pi_k \mathcal{N}(\boldsymbol{x}_i|\mu_k, \beta^{-1}I + \Sigma_k) \end{aligned} \tag{5.9}$$

for *all the K Gaussian components*, resulting in $2K$ expensive matrix multiplication operations *for every patch*.

That is, the EPLL bottleneck lies in using the naïve linear scan to solve the optimization problem in Equation (5.9). Because the goal of this problem is to identify the dominant Gaussian component in GMM, we call it the *dominant Gaussian identification* problem in the following text.

5.2.2 Observations

If we take another look at the brute-force optimization process of Equation (5.9), this is extremely similar to searching NNs in a database, despite the fact that deblurring traditionally has scant connection with image retrieval. Therefore, this problem can be viewed as a "Gaussian retrieval" problem, i.e., with an image patch as query, a set of Gaussians $\{\mu_k, \Sigma_k,$ and $\pi_k\}$ can be used to search for the patch with the highest (conditional) likelihood.

NN Search. Although appearing closer, this is still different from the traditional settings for image retrieval, where the query and database instances are in the same form. However, if we take a closer look at the inside representation of the likelihood, it is easy to notice that this is a Mahalanobis distance metric $D(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T M \boldsymbol{y}$ with $M = \beta^{-1}I + \Sigma_k$.

Therefore, if we treat the centers of the Gaussian components $\{\boldsymbol{\mu}_k\}$ as the instances in the database, Equation (5.9) turns into a slight variant of the NN search with respect to a given Mahalanobis distance metric[1]. This is a well studied problem in the field of image retrieval, and various indexing structures can be built to eliminate the brute-force linear scan, including the core approach proposed in this dissertation. That is, the bottleneck of a powerful image deblurring method is fairly similar to an image retrieval problem.

The previous analysis is for each patch as a query. However, if we view the problem one step back from an image's perspective, the spatial consistency could also help. This is because the image patches from natural images have spatial smoothness by themselves. Moreover, this property causes the nearest Gaussian center of each patch to be generally smooth (i.e., the same) with relatively few jumps, which is verified in the experiments described later in this chapter. Therefore, the core technique proposed in this dissertation is a good match for the problem, with the capability of efficiently indexing the Gaussian components, and utilizing the spatial context to refine the result.

Balance between Effect and Efficiency. An immediate concern is that this is only an approximated solution that may affect the quality of the restored image, especially considering that low-level applications are sensitive to even pixel-level error. As shown shortly, for the patches for which it is more difficult to find the corrected NN, i.e., the patches on which error is more likely to be introduced, the value of the final $\hat{k}_i$ actually has less effect on the quality of the restored patch $\hat{\boldsymbol{z}}_i$. In addition, with the refinement from the spatial consistency, as demonstrated in the evaluation, it is possible to achieve high-quality NN search for the Gaussian components.

If we assume that high-accuracy retrieval is achievable, another question is at what cost. In order to achieve high NN retrieval accuracy, traditional indexing structures usually require much more storage and speed degeneration even with moderate accuracy expectation. For example, KDTree [Bentley, 1975] would require much more back tracking on

[1] The difference from traditional NN search is mainly that every instance has its own distance metric, similar to the setting in Chapter 3. As discussed later, the distance metrics play a more important role than the center positions in the retrieval process. However, it is safe to view the problem from a retrieval perspective now.

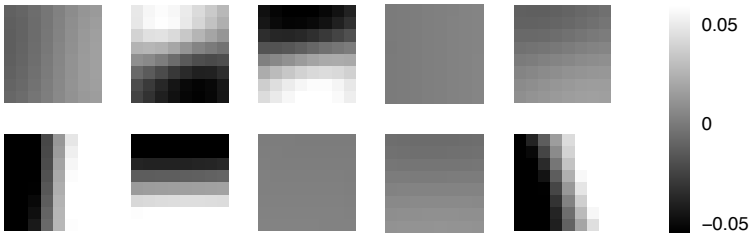


Figure 5.1: Visualization of the most significant eigenvectors of each Gaussian component in an EPLL prior with 10 components.

the neighbor nodes, and may even degenerate to linear scan in the worse case. Moreover, LSH [Datar *et al.*, 2004] requires much longer code and more hash tables, thus making the running time possibly even worse than brute-force search. However, it is worth noting that there is one key difference between the settings here and the general NN search. That is, the Mahalanobis distance is not specified arbitrarily, but reflects the distribution of the natural image patches. These metrics have their own properties that can help us construct shallow tree indexing structures, thus achieving high NN search accuracy with no backtracking.

Special Property of the Gaussians. More specifically, if we closely observe the GMM learned from natural image patches, we can see that most Gaussian components have very elongated shapes, i.e. Σ_k has only a few significant eigenvectors (that is, the eigenvectors with large eigenvalues), and they do not overlap each other, with the exception of the small parts [Zoran and Weiss, 2012]. For example, the strongest unnormalized eigenvectors of a learned ten-component GMM are generally extremely different from each other, as shown in Figure 5.1, indicating that the Gaussian components point to different directions, and thus are relatively easy to distinguish from each other. This leads us to believe that it may be possible to use a hierarchy of simple classifiers, *e.g.* linear classifiers, to separate the high-dimensional space into different subspaces that belong to different Gaussians. One subsequent concern is that, because all the mixture components share the same center, which is the origin observed in [Zoran and Weiss, 2011], such overlap may confuse the linear classifiers. We found that this does not affect performance significantly because the optimizer in Equation (5.8) pushes the patch slightly to the center along the path determined by the dominant Gaussian, and this is hardly significant when the patch is already close to

the shared center, even if a wrong Gaussian is identified and used.

5.2.3 Framework for Our Approach

Given the analogy of the core problem Equation (5.9) and the NN search problem we discussed in the previous chapters, we propose applying the core technique proposed in this dissertation to accelerate the probabilistic patch-based prior. In particular, an MRF is constructed on the query image with each overlapped patch as a vertex, and spatially close-by patches are connected with edges. Then, the dominant Gaussian identification problem in Equation (5.9) is transformed to a potential minimization problem on an MRF. The potential function consists of a unary term and a pairwise term. The unary term encapsulates the local patch information to obtain an estimation of the retrieval scores of the candidates, and the pairwise term utilizes spatial consistency to further refine the retrieval scores in order to obtain the final NN. Then, we continue EPLL optimization with other existing components in Equation (5.6) and Equation (5.8) to obtain the final deblurring result. To ensure high efficiency, approximate inference methods, such as Guided Filter [He *et al.*, 2010], are used.

However, given that the settings are different from Chapter 2 in that there is more than one distance metric, we have to adopt some adaptations in the index construction algorithm. In particular, traditional decision tree algorithms cannot be applied directly because their random generation schemes of classifier candidates are extremely inefficient in a high-dimensional space. To make the training process more stable and efficient, we utilize the GMM structure and overcome the challenges of candidate classifier generation with a Gibbs sampling approach.

First, we introduce how to search the dominant Gaussian component given by minimizing the MRF potential in the next section, and then illustrate the training process of the indexing structures in Section 5.4.

5.3 Index-Assisted Patch Prior Optimization

Similar to Chapter 3, we construct MRF $(\mathcal{V}, \mathcal{E})$ on the input (i.e., the noisy image in the ith iteration). The vertex set $\mathcal{V}$ contains all the patches extracted from the image (in an overlapped style), with each patch as a vertex. In addition, spatially close patches are connected, thus forming the edges in $\mathcal{E}$. Each vertex v_i is associated with a variable k_i, which is the ID of the dominant Gaussian in the GMM. This is our expected (approximate) solution for Equation (5.9); note that each k_i is a scalar. Unlike the 3D cases, MRF is built on an image, and thus has a grid structure with the potential function defined as

$$\Psi(\boldsymbol{k}, \boldsymbol{v}) = \sum_{v_i \in \mathcal{V}} \Psi_u(k_i, \boldsymbol{v}_i) + \lambda \sum_{(v_i, v_j) \in \mathcal{E}} \Psi_p(k_i, k_j, \boldsymbol{v}_i, \boldsymbol{v}_j). \tag{5.10}$$

Here, v_i indicates the ith vertex, and with some abuse of notation, we use $\boldsymbol{v}_i$ to indicate the vectorized raw pixels of the vertex (i.e., the noisy patch). We assume that the input images are grayscale, and it is trivial to accept multi-channel images by simply repeating the optimization process multiple times independently. λ is a parameter that balances the two terms in the potential function. As explained in the previous section, we expect that with proper design of the potential function Ψ, the solution to $\arg\min_{\boldsymbol{k}} \Psi(\boldsymbol{v}, \boldsymbol{k})$ would be a good approximation to the dominant Gaussian identification problem in Equation (5.9). As shown in the equation, there are two terms in the potential, the unary Ψ_u and pairwise Ψ_p, which are introduced in the following sections.

5.3.1 Unary Potential: Search via Ensemble of Classifiers

Similar to Chapter 2, the unary potential $\Psi_u(\cdot)$ uses a decision tree to seek local retrieval candidates[2]. With the assumption that the dominant Gaussian IDs satisfy the Markov property, Ψ_u are evaluated on each individual noisy patch at this stage. In particular, from a given noisy patch $\boldsymbol{v}_i$, the search process goes from the root of the tree to one of the leaves. Each non-leaf node in the tree has a linear classifier $\mathbf{sgn}(\boldsymbol{w}^T \boldsymbol{v}_i + b)$ that determines where

[2]We also tested random forests, and obtained a similar accuracy with decision trees.

$\boldsymbol{v}_i$ goes in the next level as follows:

$$\text{Next}(\boldsymbol{v}_i|\boldsymbol{w}, b) = \begin{cases} \text{Left child} & \boldsymbol{w}^T\boldsymbol{v}_i + b \geq 0, \\ \text{Right child} & \boldsymbol{w}^T\boldsymbol{v}_i + b < 0. \end{cases} \tag{5.11}$$

While traversing the tree from its root to a leaf node, the space of patches is recursively bisected by the linear classifiers, ending with a polyhedron $L_i = \{\boldsymbol{v}|W_i\boldsymbol{v} + B_i \leq 0\}$, with W_i and B_i determined by the traversal path of $\boldsymbol{v}_i$. From the leaf node, we obtain the pre-stored amount of average probabilities of each Gaussian component that dominates GMM in the polyhedron:

$$\phi_{ik} = \mathbb{E}_{\boldsymbol{v}\in L_i}[\text{Prob}(\hat{\mathcal{N}}(\boldsymbol{x}) = \mathcal{N}_k)]. \tag{5.12}$$

Then, the unary potential is built upon ϕ_{ik}, penalizing the variable k_i for straying far from the average case in the polyhedron,

$$\Psi_u(k) = -\phi_{ik}. \tag{5.13}$$

This has a similar probabilistic interpretation if treating the potential as the negative likelihood. (Given its resemblance to the model in Chapter 2, see Section 2.4 for a more detailed derivation.) This potential design uses local statistics in a polyhedron to describe each individual instance, which is the core idea of approximate indexing approaches, such as Bag-Of-Visual-Words [Sivic and Zisserman, 2003]. Combined with the fact that all ϕ_{ik}s are pre-computed/estimated from the database (i.e., the GMM), this is why we call it "indexing the image prior."

5.3.2 Pairwise Potential: Spatial Verification

The pairwise potential utilizes the spatial consistency of variable k_i to refine the result. This spatial consistency is essentially from the smoothness of the patch appearance. Similar to Chapter 3, we use a Potts model,

$$\Psi_p(k_i, k_j) = \begin{cases} 0 & \text{if } k_i = k_j, \\ 1 - |I_i - I_j|_2 & \text{otherwise.} \end{cases} \tag{5.14}$$

I_i and I_j are the average intensities of the patches $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$, respectively. The intuition of this design is to penalize the cases where adjacent patches have different dominant Gaussian

components, and the penalty amount is determined by the difference of two patches. For efficiency consideration, we use the difference in average intensity as a quick indicator for the difference in the patches. For two patches with a large intensity difference, we apply less penalty; otherwise, a larger penalty is specified. Note that all the intensity is between 0 and 1 in our images. Therefore, when $k_i \neq k_j$, the pairwise potential is always greater than zero.

5.3.3 Fast Approximate Inference

Substituting the definition of the unary and the pairwise term, the overall potential $\Psi(\boldsymbol{k}, \boldsymbol{v})$ has the form of

$$\Psi(\boldsymbol{k}, \boldsymbol{v}) = \sum_{v_i \in \mathcal{V}} -\phi_{ik_i} + \lambda \sum_{(v_i, v_j) \in \mathcal{E}} \delta(k_i, k_j)(1 - |I_i - I_j|_2). \tag{5.15}$$

Here, $\delta(k_j, k_j)$ is an indicator function that is equal to 0 when $k_i \neq k_j$, and 1 when $k_i = k_j$. The specific form of $\Psi(\boldsymbol{k}, \boldsymbol{v})$ is determined by first running the decision tree on the patches to obtain ϕ_{ik_i}, and then collecting the average intensity of each patch with integral images. Given that the decision tree testing process is completely independent on different patches, thus trivial to perform in parallel, and integral images are efficient to compute, the determination process of the potential function is fast.

This MRF has two special properties. One is that it has a grid structure, and the other is that it uses a Potts model in the pairwise potential design. This allows us to use faster approximate inference algorithms to optimize Equation (5.15). More specifically, we adopt cost-volume filtering [Rhemann *et al.*, 2011]. Similarly to Loopy Belief Propagation, the cost-volume filters update the marginal distribution stored in each node. However, instead of message collection and passing, such updates are performed with the guided filter [He *et al.*, 2010], which is accelerated by the integral images. Therefore, the optimization process of the potential function is also fast.

5.3.4 x-step and z-step

z-step. Once the dominant mixture component $\hat{y}_i$ for each patch $\boldsymbol{x}_i$ is found, it is input to the optimizer for Equation (5.8) as $\hat{k}$. This has a close form solution as the Wiener filter:

$$\hat{\boldsymbol{z}}_i = (\Sigma_{\hat{k}_i} + \sigma^2 I)^{-1}(\Sigma_{\hat{k}_i}\boldsymbol{x}_i + \sigma^2 I\mu_{\hat{k}_i}). \tag{5.16}$$

x-step. By setting the derivative as zero, we obtain the stationary point of Equation (5.6) as the solution for the following linear system:

$$(\lambda A^T A + \beta)\boldsymbol{x}_i = \lambda A^T \boldsymbol{y} + \beta \boldsymbol{z}_i. \tag{5.17}$$

Note that $A \in \mathbb{R}^{n^2 \times n^2}$, and n is the width/height of the image patches. Therefore, a naïve Gaussian elimination would cost $O(n^4)$. In practice, we use conjugate gradient descent to obtain a numerical solution.

FFT Acceleration. After implementing the entire pipeline, an observation is that the x-step optimization becomes the new bottleneck. Therefore, we follow [Yang *et al.*, 2009] to use Fast Fourier Transform (FFT) to further accelerate the optimization process.

Given that A is the convolution matrix for the motion kernel k, if we view the system from the original perspective of convolution, the system becomes:

$$(\lambda k^T \otimes k + \beta) \otimes \hat{\boldsymbol{x}}_i = \lambda k^T \otimes \hat{\boldsymbol{y}} + \beta \hat{\boldsymbol{x}}_i. \tag{5.18}$$

Here, $\hat{\boldsymbol{x}}_i, \hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}_i$ are the original matrix form of the patch (in contrast with the vectorized form $\boldsymbol{x}_i, \boldsymbol{y}, and \boldsymbol{z}_i$). $\otimes$ denotes convolution, and k is the motion kernel.

If we apply Fourier Transform to both ends of the system, the Fourier Transform of $\boldsymbol{x}_i$ has the form,

$$\mathcal{F}(\hat{\boldsymbol{x}}_i) = \frac{\lambda\hat{\mathcal{F}}(k) \circ \mathcal{F}(k) + \mathcal{F}(\beta)}{\lambda\hat{\mathcal{F}}(k) \circ \mathcal{F}(\hat{\boldsymbol{y}}) + \beta\mathcal{F}(\hat{\boldsymbol{z}}_i)}, \tag{5.19}$$

where $\mathcal{F}(\cdot)$ is the 2D Fourier Transform of the given signal, and $\circ$ indicates element-wise multiplication. The division is also element-wise. The final solution of the x-step is calculated by performing inverse 2D Fourier Transform to the result of Equation (5.19). Because the input signals are not strictly circular, image padding from [Liu and Jia, 2008] is used to reduce possible ringing effects.

FFT is used to expedite the $\mathcal{F}, \mathcal{F}^{-1}$ operations, thus reducing the time cost from $O(n^4)$ to $O(n^2 \log n)$. Because all multiplications and divisions in Equation (5.19) are element-wise, the overall complexity of the x-step becomes $O(n^2 \log n)$. This also accelerates x-step by orders of magnitudes in our experiments, with a patch size of 8 pixels.

5.3.5 Discussion

Interpretation of Model Selection. As revealed in [Zoran and Weiss, 2012], the learned GMM from natural image patches has some special properties. In addition to that mentioned previously, where the Gaussians have elongated shape, another such property is that the mean of the Gaussians are extremely close to zero. This brings us another interpretation of the model selection to our proposed approach.

Because the matrix Σ is a covariance matrix of a Gaussian, it has Eigen-decomposition $\Sigma = Q\Lambda Q^T$, where Q is an orthogonal matrix, by stacking the eigenvectors of Σ, and Λ is a diagonal matrix that contains the eigenvalues of Σ. If we set $L = Q\Lambda^{1/2}$, Σ can also be represented by LL^T. Therefore, in Equation (5.9), if we place a logarithm ahead of the objective and set $\beta^{-1}I + \Sigma_k = \Sigma'_k = L_k L_k^T$, the problem can be rewritten as

$$\begin{aligned}
\hat{k} &= \operatorname*{argmax}_k \pi_k \mathcal{N}\left(\boldsymbol{x}_i \mid \mu_k, \beta^{-1}I + \Sigma_k\right) \\
&= \operatorname*{argmax}_k \pi_k \mathcal{N}\left(\boldsymbol{x}_i \mid \mathbf{0}, \Sigma'_k\right) \\
&= \operatorname*{argmax}_k \left(\log \pi_k + \|L_k \boldsymbol{x}_i\|^2\right).
\end{aligned} \tag{5.20}$$

Here, $L_k \in \mathbb{R}^{n \times n}$ is from the Eigen-decomposition, and $\boldsymbol{x}_i \in \mathbb{R}^{n^2 \times 1}$ is the vectorized form of the noisy patch. Therefore, L_k can be viewed as a set of filters, where each row is a filter, and $\|L_k \boldsymbol{x}_i\|^2$ can be interpreted as the norm of the responses of the input patch and the given filters. The eigenvectors visualized in Figure 5.1 are actually the normalized version of each row L_k (from different ks in that specific figure).

Therefore, when the noisy patch $\boldsymbol{x}_i$ is given, selecting $\hat{k}$ is equivalent to finding the filter set, which provides the strongest response to $\boldsymbol{x}_i$. Then, this gives us a perspective of the model/filter selection to the problem, and our tree indexing structure gives an efficient way for retrieving the proper model(s).

This view is actually closely related to the retrieval interpretation illustrated in the introduction. From the commutative property of the dot product, the fact that $\boldsymbol{x}_i$ can be treated as an instance also means that it can be treated as a filter. In this setting, the rows L_ks can be treated as instances in the dataset. Then, the problem turns to finding the NN set in a database under the distance metric of the inner product (or linear kernel from a kernel's view), which is the retrieval interpretation.

Time Complexity. As explained before, the proposed algorithm is extremely efficient given the determination and optimization of the potential function. Here, we provide some quantitative computations.

For the testing stage of the decision tree, each linear classifier only requires a dot product operation. By taking advantage of the special properties of the Gaussians in EPLL, even a few levels of tree nodes (e.g., 12 levels) are sufficient for reasonable accuracy in practice – this already makes it even faster than the hashing-based approaches, which typically require more than 20 bits for reasonable accuracy.

Therefore, given an index tree with depth D for a K-component GMM defined on $n \times n$ patches, because we only need to apply one dot product on each level, the tree traversal for each patch requires $O(n^2 D)$ operations. The cost-volume filtering requires $O(K)$ time for each patch. Therefore, the overall time complexity is $O(mn^2 D + mK)$ for an image with m patches, with an extremely small coefficient for $O(K)$, which is from the guided filter. In contrast, the original EPLL requires $O(mn^4 K)$ time.

5.4 Prior Index Construction

As shown in Equation (5.12), the final goal of the decision tree is to estimate the probability of each Gaussian component to be the dominant one (which is a discrete distribution). The unary term design essentially assumes that all the points within the polyhedron L_i have the same distribution, i.e., a winner-take-all-scheme is used. This results in error in the process, and degenerates the deblurring quality.

Because the misclassification only occurs when the actual dominant Gaussian falls outside the winner in the distribution, in order to minimize the error introduced by indexing,

we expect such distribution within each polyhedron to be as "pure" as possible, or in other words, with as low entropy as possible. For example, with the distribution having only a single obvious peak with 90% probability, when the winner-take-all scheme is used, the misidentification only occurs with 10% probability. This entropy minimization expectation makes using a decision tree as the indexing structure appealing, especially considering that a tree index only requires logarithmic time at the testing stage.

However, a single decision tree may not be sufficient for the task. This is because the conditional likelihood $p(k|\boldsymbol{x})$ depends on β, which is the pre-defined parameter for the alternating optimization, as shown in Equation (5.6). Therefore, we build different index trees with respect to different values of β with the algorithm introduced in this section.

5.4.1 Problems of Traditional Training Algorithms

Although the testing phase of our index tree is similar to a decision tree, the training algorithm of decision trees cannot be applied directly here. In the decision tree training algorithm, a set of training examples with ground truth class labels are given as input, and the tree is trained in a recursive manner. For each splitting node (i.e., the non-leaf node), many *classifier candidates* are generated *randomly*, each of which can divide the training examples into two partitions. Then, the best classifier with the largest *information gain* computed from the partitions is selected and stored in the node.

This can be viewed as a naïve optimizer that searches the classifier randomly with the largest information gain. When incorporated with the linear classifiers, this works well on low-dimension data. However, with an increase in dimensionality, the feasible space to search expands much faster than the small space where the good solutions lie. This leads to the failure of the naïve random search optimizer when the dimensionality of $\boldsymbol{x}$ is not trivially small. Or in other words, a significant number of trials are required before it can reach the optimal or even near-optimal solutions.

To demonstrate such inefficiency of the random search scheme, we collect two million 8×8 patches as training data from natural images. The ground truth labels of the dominant Gaussian are then computed with brute force. The traditional decision tree training algorithm is applied on the dataset, with $1,000$ candidates generated randomly for every

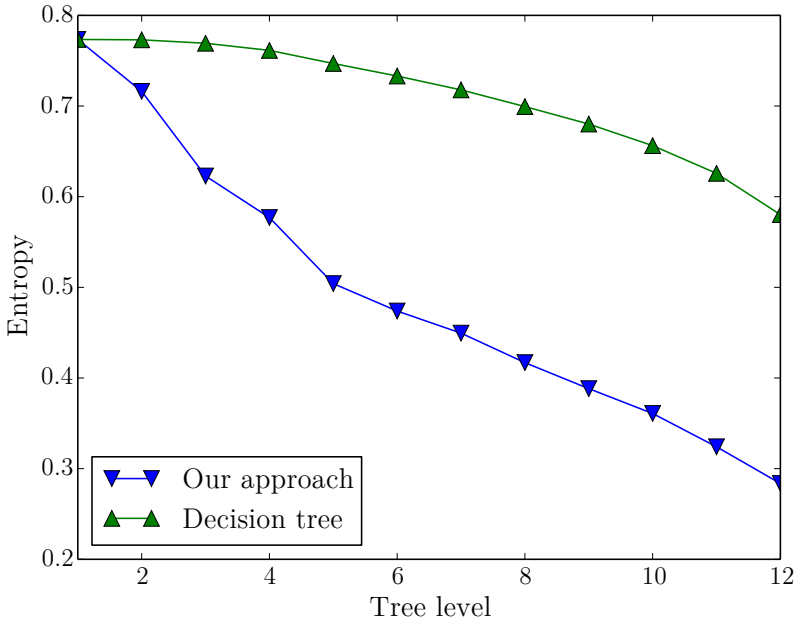


Figure 5.2: Comparison of how the average entropy decreases in different levels of the index tree with different training schemes. The traditional decision tree training algorithm is plotted in green, with the proposed approach in blue.

node. A total of 48 hours is required to train a 12-level tree on a Core i7 3.0 GHz desktop computer with MATLAB implementation, and we plot the average entropy of each level as shown by the green curve in Figure 5.2. From the figure, we can see that even after 12 levels, the average entropy continues to be close to 0.6, indicating that the distributions in the leaf nodes are still not far from uniform, and do not contain much information. As analyzed above, this also results in large indexing error in our unary term. Given that we have a high expectation on the testing speed, and thus a constrained budget on the tree depth, the decision tree training algorithm actually does not fit our problem settings.

To mitigate this challenge, we exploit the special structure of the GMM learned from natural image patches, and formulate the candidate classifier generation as an optimization problem coupled with random sampling. The recursive greedy training framework of the decision tree is still used in our approach because of its simplicity and robustness. In the following text, we discuss each step of our training process in more detail.

5.4.2 Ensemble Learning for Prior Indexing

Training Data Generation. To train an index tree for a given β, we collect a set of noisy patches $\{\mathbf{x}\}$ from the output of the x-steps of EPLL [Zoran and Weiss, 2011] as the X for training because that is the input faced by our index in real applications. The ground truth labels Y are then determined with Equation (5.9). To the best of our knowledge, there is no theoretical evidence as to how many training examples are "sufficient;" therefore, we revisit this step in Section 5.5 for the practical concern of the size of the training dataset.

Candidate Classifier Generation. Given a set of noisy patches X and the ground truth labels Y, the problem we face is to find a linear classifier $\mathbf{sgn}(\hat{\boldsymbol{w}}^T\boldsymbol{x} + \hat{b})$ so that the information gain is maximized:

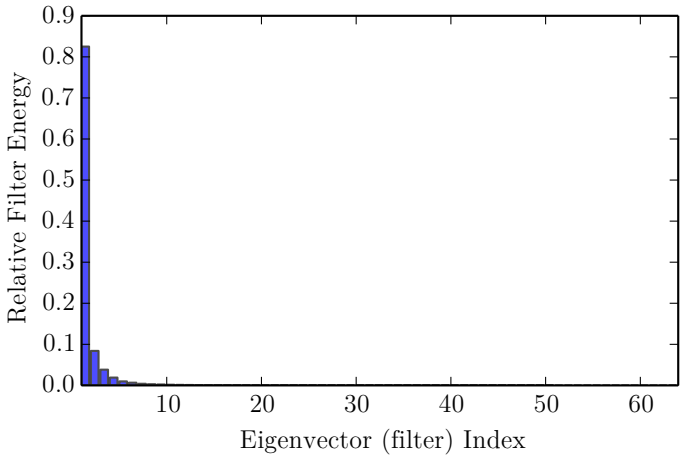
$$\hat{\boldsymbol{w}}, \hat{b} = \underset{\boldsymbol{w},b}{\operatorname{argmax}}\, E(Y) - \frac{|Y_+|}{|Y|}E(Y_+) - \frac{|Y_-|}{|Y|}E(Y_-), \tag{5.21}$$

where $E(\cdot)$ is the entropy function, and Y_+ and Y_- are, respectively, the positive and negative partitions divided by the classifier:

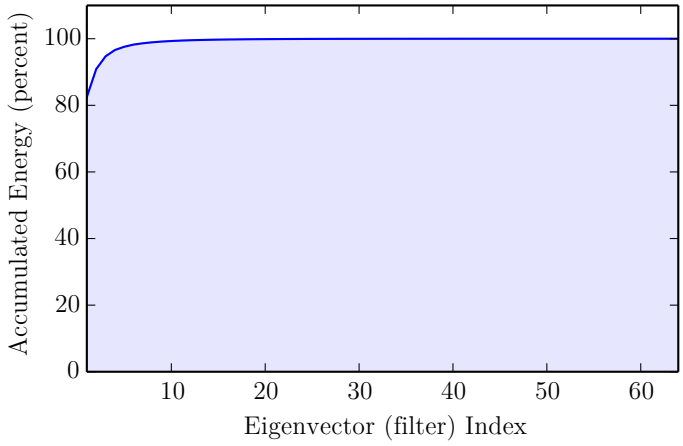
$$\begin{aligned} Y_+ &= \{y_i | \boldsymbol{w}^T\boldsymbol{x}_i + b \geq 0,\ \forall i\}, \quad \text{and} \\ Y_- &= \{y_i | \boldsymbol{w}^T\boldsymbol{x}_i + b < 0,\ \forall i\}. \end{aligned} \tag{5.22}$$

It has been shown that naïve random search does not work for high dimensional $\boldsymbol{x}$. Given that this problem is non-differentiable with the discrete training examples, we do not use classical continuous optimization methods, such as gradient descent or BFGS. Instead, we adopt a Gibbs sampling approach, while some heuristics are introduced to restrict the space from which the candidates are generated.

There are two important observations of the learned GMM. First, for most components, only a few strongest eigenvectors of the covariance matrix take the most energy of the Gaussian, as shown in Figure 5.3. More specifically, only the strongest three to four eigenvectors take more than 90% of the energy of the Gaussians learned in 8×8 patches in average. This indicates that it is possible to dramatically reduce the computation complexity by only performing sampling based on these few strong *principal directions*, which are the eigenvectors of the Gaussians' covariance matrices. The second observation is that all the Gaussian



(a) Average energy of the principal directions of the learned Gaussians from EPLL.



(b) Accumulated average energy of the principal directions of the learned Gaussians from EPLL.

Figure 5.3: Energy distribution of the principal directions of Gaussians in EPLL. The energy of the principal directions of each Gaussian is computed as the square of the corresponding eigenvalues. Then, the energy of each principal direction, sorted according to the energy, is averaged across all the Gaussians. (a) shows the distribution; and (b) shows the accumulative sum of the energy.

components share the same center, which is the origin, as observed in [Zoran and Weiss, 2011]. This can be explained with the inherent symmetry of the natural image patches.

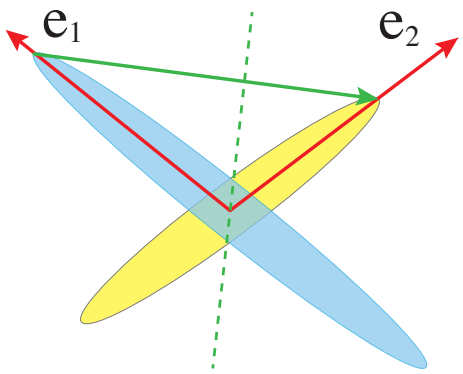


Figure 5.4: A toy sample of the proposed heuristic used to generate a candidate classifier from two given principal directions $\boldsymbol{e}_1, \boldsymbol{e}_2$. The dashed green line shows the classifier generated when $b = 0$, with the green arrow as its normal vector.

These two properties inspire a simple heuristic to generate a classifier candidate for two principal directions from two Gaussians. Take Figure 5.4 as an example: if two 2D Gaussians are given with two principal directions $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ marked as red, a reasonable guess of the decision (hyper)plane is

$$\boldsymbol{w} = \lambda_1 \boldsymbol{e}_1 - \lambda_2 \boldsymbol{e}_2, \tag{5.23}$$

where λ_1, λ_2 are the corresponding eigenvalues of $\boldsymbol{e}_1, \boldsymbol{e}_2$, as the green arrow shows. Note that $-\boldsymbol{e}_1$ and $-\boldsymbol{e}_2$ are also the principal directions. Therefore, this scheme actually generates four $\boldsymbol{w}$s.[3]

With this candidate generation scheme, the problem becomes how to sample the principal directions such that we can partition the training data "effectively." With the expectation of minimizing the tree depth with a target accuracy, we add a balance factor to the objective function in Equation (5.21). More specifically, we expect the positive and negative examples predicted by the classifier $\mathbf{sgn}(\hat{\boldsymbol{w}}^T \boldsymbol{x} + \hat{b})$ to be approximately the same in number. Or in other words, the average projection values are expected to be as small as

[3]One classifier may not be able to distinguish the two Gaussians shown in Figure 5.4. However, a simple two-level decision trump from the four candidate classifiers would have sufficient discrimination power.

possible. Then, the objective function becomes,

$$E(Y) - \frac{|Y_+|}{|Y|}E(Y_+) - \frac{|Y_-|}{|Y|}E(Y_-) - \gamma \left| \sum_{\boldsymbol{x}} \left(\boldsymbol{w}^T \boldsymbol{x} + b \right) \right|, \quad \text{s.t. } \|\boldsymbol{w}\|^2 = 1 \text{ generated from Equation (5.23).} \tag{5.24}$$

Here, $\gamma = 0.5$ is a parameter that controls the strength of the balance factor.

Note that both terms in Equation (5.24), information gain and balance factor, would only change when some example $\boldsymbol{x}_i$ changes its predicted label. That is, the terms change faster if the $\boldsymbol{w}$ swipes along some high-density area with more training examples, and slower in the low-density areas. Therefore, it is reasonable to sample more $\boldsymbol{w}$-s from the regions with low GMM probabilistic densities, which are analogous to the stationary points in the continuous case. More specifically, we place more priority in sampling the decision boundaries between two principal directions with large eigenvalues. That is, given the weights of the Gaussians $\{\pi_k\}$, we first sample two Gaussians with probability $p(k) = \pi_k$, and then sample one principal direction from the eigenvectors of the covariance matrix of each Gaussian $\{\boldsymbol{e}_{ki}\}$ with corresponding eigenvalues as the probability $p(i|k) = \lambda_{ki}$. Subsequently, Equation (5.23) is applied on the principal directions to obtain the final $\boldsymbol{w}$-s, which forms a Gibbs sampling process. Because all the Gaussians share the same center as the origin, we use $\mathcal{N}(0, 1)$ to sample bs.

A complete algorithm is illustrated in Algorithm 4. We apply the proposed approach to the same data in the experiment shown in Figure 5.2, and obtain much better training efficiency with average entropy below 0.3 in the 12th level, which is plotted as the blue curve in Figure 5.2. This proves the effectiveness of our training scheme; in the next section, we conduct more justifications on our approach, followed by evaluations on actual applications.

5.5 Experiments

We conducted a series of experiments to quantitatively verify

(1) how well the proposed MRF formulation performs for the dominant Gaussian identification task; and

Algorithm 4: Index construction for patch priors.

Input: the patch prior $\{\pi_k, \boldsymbol{\mu}_k, \Sigma_k\}_{k=1}^{K}$, training examples X, the ground truth labels Y, and the max tree depth D

Output: a decision tree T based on linear classifiers

if $D = 0$ **then**
  return a leaf node with label distribution Y.
end
foreach $1 \leq I \leq I_{max}$ **do**
  Sample two Gaussians k_1, k_2 without replacement with probability $p(k) = \pi_k$.
  Given each Gaussian k from k_1, k_2, sample one eigenvector from the eigenvectors of the covariance matrix $\{\boldsymbol{e}_{ki}\}$ with probability $p(i|k) = \lambda_{ki}$, where λ_{ki} are the corresponding eigenvalues.
  Use Equation (5.23) to generate $\boldsymbol{w}$-s given the two eigenvectors $\boldsymbol{e}_1, and \boldsymbol{e}_2$.
  Sample b from $\mathcal{N}(0, 1)$.
end
Collect all the candidate $\boldsymbol{w}$ and b; store the one that maximizes Equation (5.24) in the tree node T.
Train the left and right child of T with $(D-1)$ tree depth and Y_+, Y_- as training data, which are defined in Equation (5.22).
return T.

(2) how much the prior indexing idea may benefit real applications, such as deblurring, and even denoising.

In this section, we first quantitatively evaluate the proposed method in non-blind image deblurring, and then justify its components, especially on the performance of dominant Gaussian identification. Other applications, including deblurring real-life photos and denoising, are also demonstrated.

5.5.1 Evaluation on Non-Blind Image Deblurring

Dataset and Evaluation Protocol. We use the standard benchmark [Köhler *et al.*, 2012] that contains 48 blurry photos and 12 motion kernels collected from real life for the evaluation. Different deblurring approaches are applied to the input images, and average Peak Signal-to-Noise Ratio (PSNR) among all the kernels on each image are reported as quantitative measurements. We compare our deblurring approach with tree-based indexing with several state-of-the-art algorithms, including Discriminative non-blind deblurring [Schmidt *et al.*, 2013] (referred as Schmidt), ℓ_0-based deblurring [Xu *et al.*, 2013] (referred as Xu), and Cho's fast deblurring [Cho and Lee, 2009] (referred as Cho).

Implementation Details. We collect two million patches from 100 training images from the Berkeley Segmentation Dataset [Martin *et al.*, 2001], convolve them with one blur kernel *different* from all the testing kernels, and add Gaussian noise to obtain the training data. Then, an index tree with 12 levels is trained for each β in Half-Quadratic Splitting using Algorithm 4 for our deblurring approach. As observed in [Zoran and Weiss, 2012], increasing the GMM component number hardly improves EPLL performance after reaching 10. Subsequently, we adopt a 10-component GMM as the prior for *both* our approach and the original EPLL.

We implement our algorithm in MATLAB with the core components, such as the index tree testing, written in C++. Because the author's implementation of EPLL [Zoran and Weiss, 2011] is fully written in MATLAB, in order to make a fair comparison in speed, we attempted to implement the EPLL bottleneck (*i.e.* the dominant Gaussian identification) with C++, but we only found that it became slower. This is because MATLAB's matrix library is highly optimized. Subsequently, we determined to continue using the MATLAB implementation for EPLL. Note that aside from the newly introduced indexing approach, there is another difference compared with the original EPLL implementation, which is that we also used FFT to accelerate the x-step. All the running time is measured on a desktop computer with a Core i7 3.0 GHz CPU.

Results and Discussions. The average PSNRs of all approaches are listed in Table 5.1. Ours$_C$ shows our PSNRs based on the kernels estimated from Cho's approach [Cho and Lee, 2009], and Ours$_X$ is based on Xu's kernel [Xu *et al.*, 2013]. We can see that our non-

Img	1	2	3	4
Cho [Cho and Lee, 2009]	30.61	26.03	31.32	27.98
Xu [Xu *et al.*, 2013]	31.64	26.64	31.45	28.42
Schmidt [Schmidt *et al.*, 2013]	32.05	26.99	32.13	28.90
Ours$_C$	30.75	26.12	32.28	28.00
Ours$_X$	31.69	26.68	32.31	28.65

Table 5.1: Average PSNRs of each testing image on non-blind deblurring. Ours$_C$ and Ours$_X$ indicate our non-blind deblurring approach based on kernels estimated from Cho and Xu, respectively.

blind deblurring component improves the performance of both Cho's and Xu's approaches in most cases. Although our PSNR is slightly worse than Schmidt's approach [Schmidt *et al.*, 2013], the running time per RGB image is 2 minutes on average, which is approximately 20 times faster than [Schmidt *et al.*, 2013][4], and 40 times faster than EPLL. Also note that this is achieved when the blur kernel for index construction is dramatically different from the blur kernels used in the test images. This suggests that our index construction is not sensitive to the blur kernel used for training.

5.5.2 Evaluation on Prior Indexing and Parameter Tuning

We also evaluate the performance of our prior indexing in terms of component identification accuracy. With the same training data and training algorithm in Section 5.5.1, we vary the depth of the decision trees to explore how it affects indexing performance. The classification accuracy of the dominant Gaussian is calculated with ground truth from brute-force search, and it is averaged on all the stages and test images as the evaluation protocol.

The results with different tree depth settings are plotted in Figure 5.5. To justify the effect of the unary and pairwise potential terms, we show the identification accuracy with only the unary term and with both terms. First, it shows that the identification accuracy

[4]The authors of [Schmidt *et al.*, 2013] did not report the running time on [Köhler *et al.*, 2012], but on smaller images. We project their running time to [Köhler *et al.*, 2012] based on the (linear) time complexity on resolution.

σ	0.1	0.25	0.5	1.0	Time
BM3D [Dabov *et al.*, 2007]	30.33	26.92	23.91	17.85	4.4
BM3D$_S$ [Dabov *et al.*, 2009]	30.46	26.62	23.22	19.73	782
K-SVD$_G$ [Elad and Aharon, 2006]	29.39	25.57	22.68	19.31	60.1
K-SVD$_I$ [Elad and Aharon, 2006]	29.76	25.68	22.70	19.38	177.7
EPLL [Zoran and Weiss, 2011]	29.57	26.13	23.44	20.62	61.7
Our approach	29.47	26.08	23.49	20.62	4.5

Table 5.2: Quantitative evaluation results on image denoising. The PSNR in dB is shown for each baseline and noise level (σ) setting. The average running time (in seconds) is shown in the rightmost column.

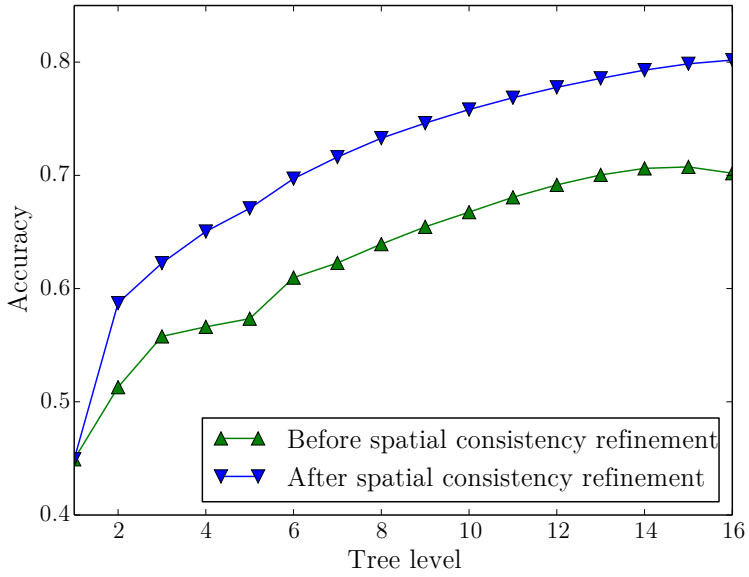


Figure 5.5: Component identification accuracy along with different tree depth.

reaches 80% with 16 levels of tree nodes. Given that we have 10 components in the GMM, this proves that the tree index performs a reasonable job in approximating the brute-force search with merely a few dot product operations. Considering the trade-off between quality and efficiency, we use 12 level trees in all other experiments. In addition, this also suggests that the pairwise term improves identification accuracy by 10% consistently over the raw identification results. This verifies our observation on the spatial coherence of the distributions of dominant Gaussians.

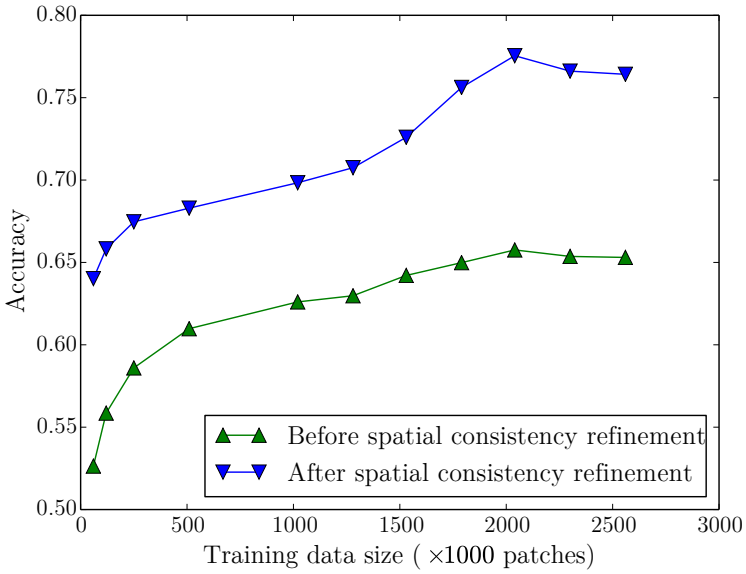


Figure 5.6: Component identification accuracy along with different training data size.

Training Data Collection. With the same training and testing image sets, we also explore the number of training patches that are required for achieving reasonable quality of dominant Gaussian identification. Figure 5.6 plots the identification accuracy against different training data sizes. This suggests that with a 12-level tree, accuracy saturates after the training dataset reaches two million patches. Therefore, we use this setting for all the experiments, including the non-blind image deblurring, deblurring high-resolution photos, and image deblurring.

5.5.3 Deblurring High-Resolution Photos from Real Life

In order to demonstrate the capability of our deblurring approach for managing real-life data, we collect some blurred photos captured from real life, run [Cho and Lee, 2009]'s approach to estimate a blur kernel, and then apply the proposed algorithm on the R, G, and B channels separately. All the collected photos have resolution greater than 800×800, and are color photos. With such input scale, EPLL [Zoran and Weiss, 2011] requires more than 0.5 hour to deblur one image, and therefore, it is not practical for deblurring applications in real life. On the other hand, our algorithm generally outputs the result within 3 minutes. Figure 5.7 shows a comparison between the results from our approach and EPLL. From the

figure, we can see that the proposed approach can achieve deblurring results with nearly unnoticeable difference from the original patch-based approaches.

5.5.4 Evaluation on Image Denoising

To demonstrate the potential of the proposed approach in other low-level vision applications, we also report the performance on denoising. We use the standard benchmark in denoising, eight 512×512 grayscale standard test images *Barbara, Boat, Cameraman, Hill, House, Lena, Man*, and *Peppers* for this evaluation. Gaussian noise with standard variance of 0.1, 0.25, 0.5, and 1 is added to the original images, respectively, as the noisy inputs. Average PSNR and the running time for all images are measured for different noise levels. We compare our approach with the state-of-the-art denoising algorithms BM3D [Dabov *et al.*, 2007], BM3D-SAPCA [Dabov *et al.*, 2009] (referred as BM3D$_S$), K-SVD [Elad and Aharon, 2006] with global dictionary (referred as K-SVD$_G$) and learned dictionary from the noisy image (referred as K-SVD$_I$), and EPLL [Zoran and Weiss, 2011] with the authors' implementations and recommended parameters. The quantitative results are reported in Table 5.2. The results show that EPLL performance is slightly worse than BM3D and BM3D-SAPCA, which is reasonable given that the latter two are designed specially for denoising. Our approach achieves extremely similar performance to EPLL, with < 0.1 dB PSNR drop on average, but it is much faster than EPLL and other denoising methods, with the exception of BM3D. We further confirmed that there are no noticeable differences between our and EPLL's results. Some sample results are shown in Figure 5.8. To demonstrate the details of the output from different approaches, only a part of the images is shown.

5.6 Conclusion and Future Work

Whereas the capability of the proposed MRF formulation was demonstrated in 3D applications in the previous chapters, thus showing promising performance, we explored the application of these techniques in 2D applications in this chapter, or more specifically, on low-level vision applications that include image deblurring and denoising.

Analysis on a state-of-the-art probabilistic patch-based prior EPLL showed that its bot-

tleneck is on an exhaustive scan of all possible Gaussian models, which can be alleviated with the proposed MRF approach in this dissertation. Similar to the previous chapters, an MRF was built on the query image, and a potential function that consists of unary and pairwise terms was minimized, thus resulting in the identified dominant Gaussian component of each patch. The training algorithm was adapted to the special distributions of patches from natural images, thus dramatically improving training efficiency. Experiment results showed that our approach achieves up to 40 times acceleration, and at the same time, provides comparable high quality results with the original EPLL approach. The performance is also competitive with other state-of-the-art deconvolution algorithms. The same acceleration effect and negligible quality loss was also observed on denoising applications.

There are several interesting directions for future exploration. One is to explore the possibility of analytically constructing the index solely from the prior model. This might even shorten the training time, and could help reveal intrinsic properties of natural image priors. Another possible direction is to extend the prior indexing idea to other patched-based approaches without a MAP framework, such as BM3D. Although the index tree is designed for probabilistic patch priors, we believe that the idea of discriminative indexing has the potential of benefiting other approaches in speed, and possibly even accuracy.

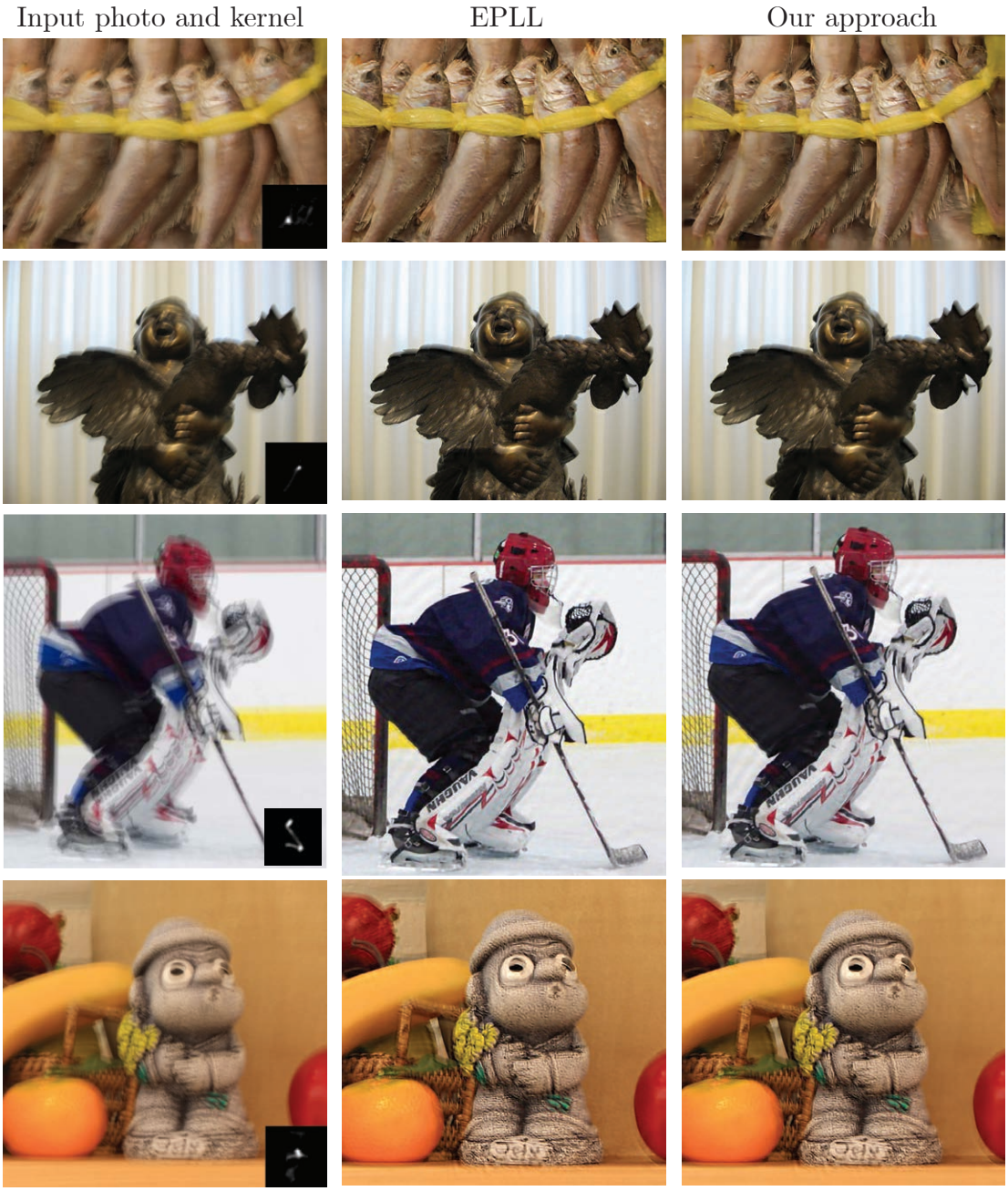


Figure 5.7: Qualitative evaluation on deblurring high-resolution photos from real life. The input with the motion kernel estimated with [Cho and Lee, 2009], the deblurring results of EPLL and the proposed approach are shown from left to right.

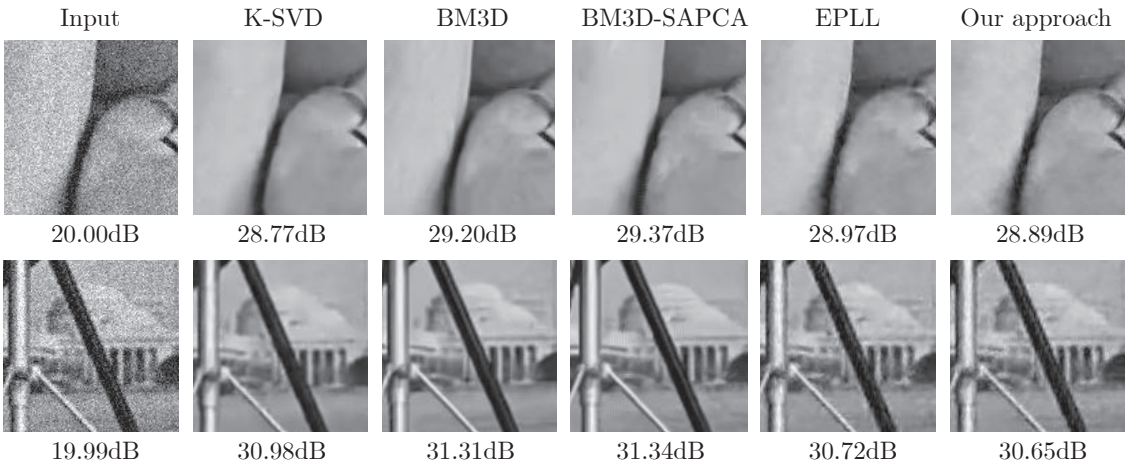


Figure 5.8: Comparison of different denoising approaches. The left-most image is the input. Only parts of the images are shown to demonstrate the details.

Part III

Conclusions

Chapter 6

Conclusions and Future Work

This dissertation was dedicated to content-based 3D shape analysis, especially on the consumer-captured data from low-cost sensors. In particular, we covered the key problems of shape retrieval, scene understanding, and pose recognition. The low-level 2D vision was also tested to demonstrate the potential of the proposed core technique to 2D data. To address the challenges of sensor noise and model incompleteness, a novel framework of using MRF was proposed, thus transforming the problems into a potential energy minimization problem. Potential functions have different specific forms for different applications, but all contain a unary term that allows partial matching, and thus resolves the model incompleteness challenge, and a pairwise term that utilizes the spatial information to provide additional robustness against the sensor noise. Different indexing structures were provided to efficiently determine the specific form of the potential functions. In this section, we first summarize our contributions, and then discuss future works for the proposed approaches.

6.1 Summary of Contributions

a. **3D Shape Retrieval.** We proposed a novel approach of 3D shape retrieval. With RFT variants as the formulation, we addressed the challenges of sensor noise and incomplete input models. This formulation uses local information to retrieve similar 3D models, formulated as the *unary potential*, and therefore preserves the capability of partial matching. It also provides better robustness against sensor noise by

checking the spatial consistency among different local parts, which is encapsulated in the *pairwise potential*. A series of weak (linear) classifiers were trained such that the specific parameters of the potential functions could be determined efficiently conditioned on the input. Such formulation form is the first work in the 3D retrieval field to the best of our knowledge. Experiment results demonstrated superior retrieval precision and recall for cross-domain 3D search with low-cost sensors, compared with state-of-the-art approaches.

b. 3D Scene Understanding. We developed a system to perform 3D scene understanding. The system uses 2D-3D MRF formulation to address the unique challenge of lacking 3D annotated training data by allowing 2D training data to be introduced in order to benefit 3D recognition. An ensemble of linear classifiers was trained on the 2D images to take advantage of the readily available 2D annotation. MRF used the 3D structure information to combine the classifiers and obtain a comprehensive scene understanding result. Experiments showed that even with no 3D training data, the proposed approach can achieve comparable results with state-of-the-art 3D scene understanding approaches trained on 3D annotated data, with orders-of-magnitude acceleration.

c. 3D Pose Recognition. A pose recognition system was developed for a robotic system that performs deformable objects manipulation. Garment pose was recognized by searching the nearest 3D model in a database simulated offline that contains the 3D models of different garments in different poses. To accelerate the search process, a novel binary 3D feature was developed. In addition, a spatial-varying weight was learned in order to integrate the local binary codes to the final retrieval score. Experiments demonstrated better recognition accuracy and faster speed compared with state-of-the-art pose recognition algorithms on deformable objects. An end-to-end system to recognize and manipulate deformable garments was also demonstrated in the form of a video.

d. Extension on 2D Data. We also explored the potential of the formulation on 2D data. An image deblurring system was built based on the same formulation in

(**a**). By constructing an MRF on an input image with a series of linear classifiers to determine the potential functions, proper deblurring models were retrieved efficiently by performing approximate inference. The retrieved models were then applied to the blurry input, resulting in a complete and efficient optimization framework. This resolved the bottleneck of the popular and powerful probabilistic patch-based priors, and accelerated the deblurring process up to 40 times without noticeable quality difference.

6.2 Future Work

Despite the recent advances in 3D content analysis, this problem still remains open and has the potential to dramatically change humans' life. In this section, we first discuss the limitations of the proposed MRF formulation, and then provide two directions to outline exciting opportunities and possible extensions as the future work.

Our MRF formulation has several limitations.

a1. Trade-off between Potential Function Complexity and Optimization Tractability. Although MRF has advantages of flexibility, robustness against noise, and capability to combine higher-order inter-correlation, it also has drawbacks, which may limit its application in practice. For example, the current optimization algorithms can only achieve global optimal under certain circumstances (binary variables for graph-cut, and tree structures for LBP). This requires more sophisticated potential function designs that work well even when only local optima can be achieved, and these designs often result in slower optimization speed. An interesting direction is to explore novel formulations that preserve the flexibility and robustness of MRFs, but easier to optimize. Promising directions include Fields of Experts [Roth and Black, 2005], and fully connected PGMs [Koltun, 2011], which have a surprisingly fast solver available, despite the fact that the potential function is extremely complicated.

a2. CAD Database Update for Object Search. Our approach for 3D object search uses a random forest to determine the specific form of the unary term of the potential function. When the CAD database needs to be updated, such as a new model needs

to be inserted, the aforementioned random forest has to be retrained from scratch. Therefore its applicability is limited when the database is expected to have frequent updates. A possible solution is to introduce online random forest training algorithms such as [Ben-Haim and Tom-Tov, 2010] to avoid the retraining, but use an efficient updating process to accommodate the database updates.

Besides the three discussed problems of object search, scene understanding, and pose estimation, there are many interesting directions worth exploring.

b1. 3D Feature Learning. In the proposed approaches to all the problems in this dissertation, we rely on existing 3D features, including Scale-Invariant Spin-Image and a 3D extension of shape context. Although these features have good empirical performance, there is still not any 3D features that provide multiple invariant properties, comparable to SIFT in 2D computer vision. It also lacks theoretical justification about why the chosen 3D features work well. Therefore one important and fundamental topic in 3D shape analysis is to design an effective 3D feature to describe both the complete, noise-free CAD models and the possibly incomplete, noisy user-captured models. An annotated dataset with model-level or point-wise correspondences may be helpful, to allow supervised techniques to learn an optimal feature representation in certain scenarios.

b2. Single-View based 3D Content Analysis. It is not always feasible to conduct a 360-degree scan for a target object, e.g. when it is large or in the corner of a room. In these scenarios, the capability of effectively utilizing a single-view capture from a low-cost sensor becomes critical. While it is possible to utilize the input data from a pure 3D perspective, a 2.5D perspective that views the depth input as an image allows more sophisticated operations such as convolution and integral images. In addition, this 2.5D perspective also enables recent advances in deep-learning-based representation learning to be easily integrated, which may dramatically boost the performance. Therefore how to properly manage a single-view capture, especially from a 2.5D point of view, is another direction worth exploring.

If an effective approach can be developed for the single-view 3D content analysis, more

interesting research topics can be derived. For example, it is obvious that different views of the same object may provide different amount of information to the analysis algorithms – a view of a coffee mug with its handle visible will likely obtain higher-quality search results compared to that with the handle occluded. Subsequently it would be meaningful to predict and guide users to the optimal angle to achieve high-quality results for object search, scene understanding, and/or pose estimation. It becomes even more interesting when the "user" is a robot, leading to counter-intuitive conclusions. For example, a robot with feet may see better, because by moving to a proper location/orientation to help the underlying scene understanding algorithm achieve better results, it may understand the surroundings better and make more reasonable decisions.

Part IV

Bibliography

Bibliography

Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(11):2274–2282, 2012.

Eugene Agichtein, Eric Brill, and Susan Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 19–26. ACM, 2006.

B. Alexe, T. Deselaers, and V. Ferrari. What is an object. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

A. Anand, H. Koppula, T. Joachims, and A. Saxena. Contextually guided semantic labeling and search for 3d point clouds. *International Journal of Robotics Research*, 2012.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 404–417. Springer, 2006.

Amir Beck and Marc Teboulle. Fast gradient-based algorithms for constrained total variation image denoising and deblurring problems. *Image Processing, IEEE Transactions on*, 18(11):2419–2434, 2009.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(24), April 2002.

G. Ben-Artzi, H. Hel-Or, and Y. Hel-Or. The gray-code filter kernels. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2007.

Y. Ben-Haim and E. Tom-Tov. A streaming parallel decision tree algorithm. *Journal of Machine Learning Research*, 11:849–872, March 2010.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

Kevin W Bowyer, Kyong Chang, and Patrick Flynn. A survey of approaches and challenges in 3d and multi-modal 3d+ 2d face recognition. *Computer Vision and Image Understanding*, 101(1):1–15, 2006.

T. Brogrdh. Present and future robot control development – an industrial perspective. *Annual Reviews in Control*, 31(1):69 – 79, 2007.

M. Bronstein and I. Kokkinos. Scale-Invariant Heat Kernel Signatures for Non-Rigid Shape Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1704–1711, 2010.

A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov. Shape Google: Geometric Words and Expressions for Invariant Shape Retrieval. *ACM Transaction on Graphics*, 30(1), February 2011.

Ding-Yun Chen, Xiao-Pei Tian, Yu-Te Shen, and Ming Ouhyoung. On Visual Similarity Based 3D Model Retrieval. *Computer Graphics Forum (EuroGraphics)*, 22(3):223–232, 2003.

J. Chen, D. Bautembach, and S. Izadi. Scalable real-time volumetric surface reconstruction. *ACM Transaction on Graphics (Proceedings of SIGGRAPH)*, 32(4):113:1–113:16, July 2013.

S. Cho and S. Lee. Fast motion deblurring. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)*, 28(5), 2009.

Sunghyun Cho, Jue Wang, and Seungyong Lee. Handling outliers in non-blind image deconvolution. In *Proceedings of the International Conference on Computer Vision (ICCV)*, Nov 2011.

D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

Brian Curless and Marc Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM, 1996.

M. Cusumano-Towner, A. Singh, S. Miller, J. F. O'Brien, and P. Abbeel. Bringing clothing into desired configurations with limited perception. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. Image denoising by sparse 3D transform-domain collaborative filtering. *IEEE Transactions on Image Processing*, 16(8), Aug 2007.

K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. BM3D image denoising with shape-adaptive principal component analysis. In *SPARS*, 2009.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

Petros Daras and Apostolos Axenopoulos. A 3d shape retrieval framework supporting multimodal queries. *International Journal of Computer Vision*, 89(2-3):229–247, 2010.

T. Darom and Y. Keller. Scale-Invariant Features for 3-D Mesh Models. *IEEE Transactions on Image Processing*, 21(5):2758–2769, May 2012.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262. ACM, 2004.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 248–255. IEEE, 2009.

T. Deselaers, B. Alexe, and V. Ferrari. Weakly supervised localization and learning with generic knowledge. *International Journal of Computer Vision*, 2012.

Michael Donoser and Horst Bischof. Efficient maximally stable extremal region (mser) tracking. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 553–560. IEEE, 2006.

Helin Dutagaci, Afzal Godil, Apostolos Axenopoulos, Petros Daras, Takahiko Furuya, and Ryutarou Ohbuchi. SHREC'09 Track: Querying with Partial Models. In *Proceedings of Eurographics 3DOR*, pages 69–76, 2009.

M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Transactions on Image Processing*, 15(12):3736–3745, December 2006.

Felix Endres, Jürgen Hess, Nikolas Engelhard, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. An evaluation of the rgb-d slam system. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 1691–1696. IEEE, 2012.

R. Fergus, B. Singh, A. Hertzmann, S. Roweis, and W. T. Freeman. Removing camera shake from a single photograph. *ACM Transaction on Graphics (Proceedings of SIGGRAPH)*, 2006.

William T Freeman, Thouis R Jones, and Egon C Pasztor. Example-based super-resolution. *Computer Graphics and Applications, IEEE*, 22(2):56–65, 2002.

A. Frome, D. Huber, R. Kolluri, T. Blow, and J. Malik. Recognizing objects in range data using regional point descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 224–237, 2004.

R. Fulton and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009.

Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A Search Engine for 3D Models. *ACM Transaction on Graphics*, 22(1):83–105, January 2003.

Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. Modeling by Example. *ACM Transaction on Graphics*, 23(3):652–663, 2004.

Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

Daniel Glasner, Shai Bagon, and Michal Irani. Super-resolution from a single image. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 349–356. IEEE, 2009.

Corey Goldfeder, Matei Ciocarlie, Jaime Peretzman, Hao Dang, and Peter K Allen. Data-driven grasping with partial sensor data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1278–1283. IEEE, 2009.

Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 817–824. IEEE, 2011.

Google Inc. Project Tango. http://www.google.com/atap/projecttango/.

S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *International Journal of Computer Vision*, 2009.

Stephen Gould, Richard Fulton, and Daphne Koller. Decomposing a scene into geometric and semantically consistent regions. In *Computer Vision, 2009 IEEE 12th International Conference on*, 2009.

Kristen Grauman and Trevor Darrell. The pyramid match kernel: Discriminative classification with sets of image features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2, pages 1458–1465. IEEE, 2005.

K. He, J. Sun, and X. Tang. Guided image filtering. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010.

Koppula Hema, Abhishek Anand, Thorsten Joachims, and Ashutosh Saxena. Cornell point cloud dataset. http://pr.cs.cornell.edu/sceneunderstanding/data/data.php, 2009.

Winston H Hsu, Lyndon S Kennedy, and Shih-Fu Chang. Video search reranking via information bottleneck principle. In *Proceedings of the 14th annual ACM international conference on Multimedia*, pages 35–44. ACM, 2006.

Peng Huang, Adrian Hilton, and Jonathan Starck. Shape similarity for 3d video sequences of people. *International Journal of Computer Vision*, 89(2-3):362–381, 2010.

D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge. Comparing images using the hausdorff distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1993.

Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. KinectFusion: Real-time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, pages 559–568, 2011.

J. Lei J. Maitin-Shepard, M. Cusumano-Towner and P. Abbeel. Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. Regression Tree Fields - an Efficient, Non-

Parametric Approach to Image Labeling Problems. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2376–2383, 2012.

Jeremy Jancsary, Sebastian Nowozin, and Carsten Rother. Loss-specific training of non-parametric image restoration models: A new state of the art. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

Joachims.T. Optimizing search engines using clickthrough data. In *KDD*, 2002.

A.E. Johnson and M. Hebert. Using Spin Images for Efficient Object Recognition in Cluttered 3D Scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, May 1999.

E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. *ACM Transaction on Graphics*, 2010.

Y. Kita and N. Kita. A model-driven method of estimating the state of clothes for manipulating it. In *Proceedings WACV*, 2002.

Y. Kita, T. Ueshiba, E-S Neo, and N. Kita. Clothes state recognition using 3d observed data. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2009.

Y. Kita, F. Kanehiro, T. Ueshiba, and N. Kita. Clothes handling based on recognition by strategic observation. In *Humanoid Robots*, 2011.

R. Köhler, M. Hirsch, B. Mohler, B. Schölkopf, and S. Harmeling. Recording and playback of camera shake: benchmarking blind deconvolution with a real-world database. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

I. Kokkinos, M. M. Bronstein, R. Litman, and A. M. Bronstein. Intrinsic Shape Context Descriptors for Deformable Shapes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 159–166, 2012.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press, 2009.

Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2, pages 508–515. IEEE, 2001.

Vladlen Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Advances in Neural Information Processing Systems (NIPS)*, 2011.

H. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3d point clouds for indoor scenes. In *Advances in Neural Information Processing Systems (NIPS)*, 2011.

D. Krishnan and R. Fergus. Fast image deconvolution using hyper-laplacian priors. In *Advances in Neural Information Processing Systems (NIPS)*, 2009.

D. Kuettel, M. Guillaumin, and V. Ferrari. Segmentation propagation in imagenet. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012.

K. Lai and D. Fox. Object recognition in 3d point clouds using web data and domain adaptation. *International Journal of Robotics Research*, 2010.

Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Sparse distance learning for object recognition combining rgb and depth information. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, pages 4007–4013. IEEE, 2011.

L. Latecki, R. Lakamper, and T. Eckhardt. Shape descriptors for non-rigid shapes with a single closed contour. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2169–2178. IEEE, 2006.

Xiaolan Li, Afzal Godil, and Asim Wagan. Shrec08 entry: Visual based 3d cad retrieval using fourier mellin transform. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications*. IEEE, 2008.

Bo Li, Afzal Godil, Masaki Aono, X. Bai, Takahiko Furuya, L. Li, Roberto Javier Lpez-Sastre, Henry Johan, Ryutarou Ohbuchi, Carolina Redondo-Cabrera, Atsushi Tatsuma, Tomohiro Yanagimachi, and S. Zhang. SHREC'12 Track: Generic 3D Shape Retrieval. In *Proceedings of EuroGraphics 3DOR*, pages 119–126, 2012.

H. Li, E. Vouga, A. Gudym, L. Luo, J. T. Barron, and G. Gusev. 3d self-portraits. *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)*, 32(6), November 2013.

Bo Li, Yijuan Lu, Afzal Godil, Tobias Schreck, Benjamin Bustos, Alfredo Ferreira, Takahiko Furuya, Manuel J. Fonseca, Henry Johan, Takahiro Matsuda, Ryutarou Ohbuchi, Pedro B. Pascoal, and Jose M. Saavedra. A Comparison of Methods for Sketch-Based 3D Shape Retrieval. *CVIU*, 119:57 – 80, 2014.

Y. Li, C-F Chen, and P. K. Allen. Recognition of deformable object category and pose. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2014.

Yinxiao Li, Yan Wang, Michael Case, Shih-Fu Chang, and Peter K. Allen. Real-time pose estimation of deformable objects using a volumetric approach. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, September 2014.

Yinxiao Li, Danfei Xu, Yonghao Yue, Yan Wang, Shih-Fu Chang, Eitan Grinspun, and Peter K. Allen. Regrasping and unfolding of garments using predictive thin shell modeling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2015.

Renting Liu and Jiaya Jia. Reducing boundary artifacts in image deconvolution. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, pages 505–508, Oct 2008.

Dong C Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

L. Liu, L. Zhang, Y. Xu, C. Gotsman, and S. J. Gortler. A local/global approach to mesh parameterization. In *Proceedings of the Symposium Geometry Processing (SGP)*, 2008.

C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2074–2081, 2012.

David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the International Conference on Computer Vision (ICCV)*, volume 2, pages 1150–1157. Ieee, 1999.

J. Machado, A. Ferreira, P. B. Pascoal, M. Abdelrahman, M. Aono, M. T. El-Melegy, A. A. Farag, H. Johan, B. Li, Y. Lu, and A. Tatsuma. Shrec'13 track: Retrieval of objects captured with low-cost depth-sensing cameras. In *Proceedings of EuroGraphics 3DOR*, pages 65–71, 2013.

T. Malisiewicz, A. Gupta, and A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Ensemble of exemplar-svms for object detection and beyond. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 89–96. IEEE, 2011.

D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2001.

S. Miller, J. Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. *International Journal of Robotics Research*, 2012.

M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problems. In *AAAI*, 2002.

D. Munoz, J. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2010.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 467–475. Morgan Kaufmann Publishers Inc., 1999.

K. Murphy, A. Torralba, and W. Freeman. Using the forest to see the trees: a graphical model relating features, objects and scenes. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

L. Nan, K. Xie, and A. Sharf. A search-classify approach for cluttered indoor scene understandins. In *ACM Transaction on Graphics (Proceedings of SIGGRAPH Asia)*, 2012.

David Nister and Henrik Stewenius. Scalable recognition with a vocabulary tree. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2161–2168. IEEE, 2006.

R. Ohbuchi, K. Osada, T. Furuya, and T. Banno. Salient Local Visual Features for Shape-Based 3D Model Retrieval. In *Proceedings of Shape Modeling and Applications*, pages 93–102, June 2008.

Aude Oliva and Antonio Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in brain research*, 155:23–36, 2006.

R. Osada and T. Funkhouser. Matching 3d models with shape distributions. In *Proceedings of the IEEE International Conference on Shape Modeling and Applications*, 2001.

S. Pan and Q. Yang. A survey on transfer learning. *TKDE*, 2010.

A. Patterson, P. Mordohai, and K. Daniilidis. Object detection from large-scale 3-d datasets using bottom-up and top-down descriptors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008.

J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object Retrieval with Large Vocabularies and Fast Spatial Matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.

James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2007.

D. Pickup, X. Sun, P. L. Rosin, R. R. Martin, Z. Cheng, Z. Lian, M. Aono, A. Ben Hamza, A. Bronstein, M. Bronstein, S. Bu, U. Castellani, S. Cheng, V. Garro, A. Giachetti, A. Godil, J. Han, H. Johan, L. Lai, B. Li, C. Li, H. Li, R. Litman, X. Liu, Z. Liu, Y. Lu, A. Tatsuma, and J. Ye. SHREC'14 track: Shape retrieval of non-rigid 3d human models. In *EG 3DOR*, 2014.

C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz. Fast cost-volume filtering for visual correspondence and beyond. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011.

Stefan Roth and Michael J Black. Fields of experts: A framework for learning image priors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 860–867, 2005.

S. Roth and M. Blacky. Fields of experts. *International Journal of Computer Vision*, 82(2):205 – 229, 2009.

Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics (TOG)*, 23(3):309–314, 2004.

B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: A database and web-based tool for image annotation. *International Journal of Computer Vision*, 2008.

Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Aligning point cloud views using persistent feature histograms. In *Proceedings of the IEEE/RSJ*

International Conference on Intelligent Robots and Systems (IROS), pages 3384–3391. IEEE, 2008.

Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (FPFH) for 3d registration. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, pages 3212–3217. IEEE, 2009.

Mohammad Amin Sadeghi and Ali Farhadi. Recognition using visual phrases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1745–1752. IEEE, 2011.

Scott Satkin and Martial Hebert. 3dnn: Viewpoint invariant 3d geometry matching for scene understanding. In *Proceedings of the International Conference on Computer Vision (ICCV)*, pages 1873–1880. IEEE, 2013.

Scott Satkin, Jason Lin, and Martial Hebert. Data-driven scene understanding from 3d models. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2012.

Uwe Schmidt, Carsten Rother, Sebastian Nowozin, Jeremy Jancsary, and Stefan Roth. Discriminative Non-blind Deblurring. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 604–611, 2013.

J. Schulman, A. Lee, J. Ho, and P. Abbeel. Tracking deformable objects with point clouds. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

Tianjia Shao, Weiwei Xu, KangKang Yin, Jingdong Wang, Kun Zhou, and Baining Guo. Discriminative Sketch-based 3D Model Retrieval via Robust Shape Matching. *Computer Graphics Forum (PG)*, pages 2011–2020, 2011.

Tianjia Shao, Weiwei Xu, Kun Zhou, Jingdong Wang, Dongping Li, and Baining Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG)*, 31(6):136, 2012.

Shapify.me. https://shapify.me/.

Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

A. Shrivastava, T. Malisiewicz, A. Gupta, and A. Efros. Data-driven visual similarity for cross-domain image matching. In *ACM Transaction on Graphics (Proceedings of SIGGRAPH)*, 2011.

Nathan Silberman. Nyu indoor depth dataset. http://cs.nyu.edu/~silberman/datasets/, 2012.

Josef Sivic and Andrew Zisserman. Video google: A text retrieval approach to object matching in videos. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1470–1477. IEEE, 2003.

N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring image collections in 3d. In *ACM Transaction on Graphics (Proceedings of SIGGRAPH)*, 2006.

Shuran Song and Jianxiong Xiao. Sliding shapes for 3d object detection in depth images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 634–651. 2014.

Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A Concise and Provably Informative Multi-Scale Signature Based on Heat Diffusion. *Computer Graphics Forum*, 28(5):1383–1392, 2009.

L. Sun, S. Cho, J. Wang, and J. Hays. Edge-based blur kernel estimation using patch priors. In *ICCP*, 2013.

Johan W.H. Tangelder and Remco C. Veltkamp. A Survey of Content based 3D Shape Retrieval Methods. *Multimedia Tools and Applications*, 39(3):441–471, 2008.

A. Thayananthan, B. Stenger, P. H. S. Torr, and R. Cipolla. Shape context and chamfer matching in cluttered scenes. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003.

A. Toshev, B. Taskar, and K. Daniilidis. Object detection via boundary structure segmentation. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variable. *Journal of Machine Learning Research*, 2005.

Z. Tu and A. Yuille. Shape matching and recognition: Using generative models and informative features. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2004.

T Tung and T. Matsuyama. Topology dictionary for 3d video understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34:1645 – 1657, August 2012.

J. Wang, L. Yin, X. Wei, and Y. Sun. 3d facial expression recognition based on primitive surface feature distribution. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

P-C Wang, S. Miller, M. Fritz, T. Darrell, and P. Abbbeel. Perception for the manipulation of socks. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011.

C. Wang, M. Bronstein, A. Bronstein, and N. Paragios. Discrete minimum distortion correspondence problems for non-rigid shape matching. In *Proceedings of International Conference on Scale Space and Variational Methods in Computer Vision*, pages 580–591, 2012.

Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2393–2406, 2012.

Meng Wang, Hao Li, Dacheng Tao, Ke Lu, and Xindong Wu. Multimodal graph-based reranking for web image search. *Image Processing, IEEE Transactions on*, 21(11):4649–4661, 2012.

Yan Wang, Rongrong Ji, and Shih-Fu Chang. Label propagation from imagenet to 3d point clouds. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

Yan Wang, Sunghyun Cho, Jue Wang, and Shih-Fu Chang. Discriminative indexing for probabilistic image patch priors. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2014.

Yan Wang, Jie Feng, Zhixiang Wu, Jun Wang, and Shih-Fu Chang. From low-cost depth sensors to cad: Cross-domain 3d shape retrieval via regression tree fields. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2014.

Kilian Q Weinberger and Lawrence K Saul. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, 10:207–244, 2009.

Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

Oliver Whyte, Josef Sivic, and Andrew Zisserman. Deblurring shaken and partially saturated images. In *Proceedings of the IEEE Workshop on Color and Photometry in Computer Vision, with ICCV 2011*, 2011.

B. Willimon, S. Birchfield, and I. Walker. Rigid and non-rigid classification using interactive perception. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2010.

B. Willimon, S. Birchfield, and I. Walker. Classification of clothing using interactive perception. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

B. Willimon, I. Walker, and S. Birchfield. A new approach to clothing classification using mid-level layers. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2013.

C. Wu, B. Clipp, X. Li, J-M Frahm, and M. Pollefeys. 3d model matching with viewpoint-invariant patches. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

Eric P Xing, Michael I Jordan, Stuart Russell, and Andrew Y Ng. Distance metric learning with application to clustering with side-information. In *Advances in neural information processing systems*, pages 505–512, 2002.

X. Xiong, D. Munoz, J. Bagnell, and M. Hebert. 3-d scene analysis via sequenced predictions over points and regions. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

Li Xu, Shicheng Zheng, and Jiaya Jia. Unnatural l0 sparse representation for natural image deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

Liu Yang and Rong Jin. Distance metric learning: A comprehensive survey. *Michigan State Universiy*, 2, 2006.

J. Yang, Y. Zhang, and W. Yin. An efficient TVL1 algorithm for deblurring multichannel images corrupted by impulsive noise. *Journal on Scientific Computing*, 31(4), 2009.

Emine Yilmaz and Javed A Aslam. Estimating average precision with incomplete and imperfect judgments. In *Proceedings of ACM International Conference on Information and Knowledge Management*, 2006.

Sang Min Yoon, Maximilian Scherer, Tobias Schreck, and Arjan Kuijper. Sketch-based 3D Model Retrieval Using Diffusion Tensor Fields of Suggestive Contours. In *Proceedings of ACM Multimedia*, pages 193–200, 2010.

Junsong Yuan, Zicheng Liu, and Ying Wu. Discriminative subvolume search for efficient

action detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2442–2449. IEEE, 2009.

A. Zaharescu, E. Boyer, K. Varanasi, and R. Horaud. Surface Feature Detection and Description with Applications to Mesh Matching. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 373–380, 2009.

Robert C. Zeleznik, Kenneth P. Herndon, and John F. Hughes. SKETCH: An Interface for Sketching 3D Scenes. In *ACM SIGGRAPH 2007 Courses*, 2007.

Yimeng Zhang, Zhaoyin Jia, and Tsuhan Chen. Image retrieval with geometry-preserving visual phrases. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 809–816. IEEE, 2011.

Lei Zhang, Yongdong Zhang, Jinhu Tang, Ke Lu, and Qi Tian. Binary code ranking with weighted hamming distance. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1586–1593. IEEE, 2013.

Zhengyou Zhang. Microsoft kinect sensor and its effect. *MultiMedia, IEEE*, 19(2):4–10, 2012.

D. Zoran and Y. Weiss. From learning models of natural image patches to whole image restoration. In *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011.

D. Zoran and Y. Weiss. Natural images, gaussian mixtures and dead leaves. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.