博士論文

Object-level visual mining

（オブジェクトレベルのビジュアルマイニング）

by

QUANSHI ZHANG

（張拳石）

THESIS
Presented to the Faculty of the Graduate School of
Engineering, The University of Tokyo
in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR of PHILOSOPHY

Department of Civil Engineering
THE UNIVERSITY OF TOKYO
August 2014

Abstract

In this thesis, the author focuses on one of ultimate dreams and biggest challenges in the field of computer vision, *i.e.*, object-level precise visual mining. The author aims to propose a general platform that can directly mine category models from large, unaligned, and cluttered images without labeling "what is where" for many visual tasks, rather than a technique for a specific visual application.

This will be the main breakthrough of computer vision in the era of big data. Based on the proposed method, people can abandon the conventional style of training category models from a large number of well prepared of training samples. Instead, the category model will be automatically mined from big visual data without much cost of human labeling. In addition, this visual mining idea can be extended from mining category models for object detection to mining category knowledge for many other visual applications, such as object tracking, recognition, segmentation, and 3D reconstruction, Therefore, the proposed object-level visual mining can be also understood as a plausible way efficiently build a comprehensive knowledge base that contains a huge number of object categories and can provide the object-level knowledge to guide different kinds of visual tasks.

Conventional studies of model training usually developed algorithms considering the terms of improving performance or increasing accuracy, and ignored problems with the choice of the training data. By contrast, the object-level visual mining pays more attention to loosening the requirement for training data, as without sufficient human labeling, the big visual data in the internet contains all typical kinds of challenges in the field of computer vision. For example, the mining process would probably simultaneously suffer from the intra-category variations in texture, rotation, scale, illumination, pose, and structure. To some extent, this is the biggest challenge in the field of computer vision, and to the best of the knowledge, no approaches can automatically overcome all of these intra-category variations without sufficient human labeling. Some methods related to the concept of visual

mining escape from this challenge by ignoring some of these variations, *e.g.*, the techniques of object discovery and co-segmentation mainly focus on textural knowledge and ignore the information of structure and scale. However, in this research, the author aims to develop visual-mining methods that can make a full use of these kinds of information, rather than simply ignore some of them. It is because that the author refers to achieve high-level robustness in model mining, *i.e.*, the mined model should have an ability of dealing with different kinds of variations, although most model learning methods only seek for low-level model robustness by applying X-independent methods or X-invariant features.

Therefore, the author has proposed a bunch of algorithms for visual mining, including "attributed graph mining" and "unsupervised learning for graph matching". In particular, attributed graph mining is the core technology. The author first defines the problem of "maximal-size graph mining" in graph domain of attributed relational graphs (ARGs), as the attributed graph mining. This extends the boundary of graph theory from previous graph domain of "labeled graphs". Attributed graph mining bridges the two main fields of artificial intelligence, *i.e.*, graph theory and computer vision. The author uses the ARG to represent an image, and in this case, the model for small common objects inside a number of unaligned images corresponds to the common subgraph pattern among a set of ARGs. The typical variations in texture, rotation, scale, illumination, pose, and structure can be modeled as the attribute variations of the subgraph pattern. In addition, the task of identifying target objects in unlabeled images can be formulated as a graph matching problem. In this way, the attributed graph mining can be regarded a general tool for category discovery and modeling from ubiquitous images

In addition, the proposed technique has a large number of extended applications besides the basic application of category modeling. Because attributed graph mining can automatically discover the target object in unlabeled images, label the object parts, and determine inter-image part correspondences, the author uses the part correspondences as input for further model training for other visual tasks. For example, this method has been successfully applied to training models for object tracking and single-view 3D reconstruction. The proposed method can also be used to recover the model for the whole object from object fragments.

Furthermore, theoretically speaking, the applications of proposed techniques in this thesis are not limited to mining from ordinary images. They can be applied to other kinds of visual data, such as RGB-D images, videos, 3D point clouds, and so on. Their applications can be even extended to the other fields, for example, mining structural patterns of protein. These techniques can be used, as long as the people want to mine knowledge from fuzzy data that can be represented using ARGs.

In this thesis, the author introduces the algorithms of attributed graph mining and unsupervised learning for graph matching, as well as some technical extensions and applications of them.

Table of Contents

Page

Abstract iii

Table of Contents vi

Chapter

1 Introduction 1

2 A General Platform for Visual Mining: Attributed Graph Mining 5

- 2.1 Introduction 6
  - 2.1.1 Views of graph matching & task introduction 6
  - 2.1.2 Views of graph mining & the proposed method 8
  - 2.1.3 Platform for model mining from big visual data 10
  - 2.1.4 Summary 12
- 2.2 Related work 12
- 2.3 Problem formulation 14
- 2.4 Algorithm 18
- 2.5 Experiments 22
  - 2.5.1 Experiment 1: mining edge-based models from cluttered RGB-D images 23
  - 2.5.2 Experiment 2: mining edge-based models from cluttered RGB images 26
  - 2.5.3 Experiment 3: mining general models from web images 27
  - 2.5.4 Quantitative analysis and evaluations 29
- 2.6 Discussion and conclusions 33
- 2.7 APPENDIX: Demonstration of Equation (2.8) 33

3 Secondary Platform: Unsupervised Learning of Graph Matching 37

- 3.1 Introduction 38
- 3.2 Related work 40
- 3.3 Preliminary: graph matching problem 42

3.4 Learning of graph matching 43
3.4.1 Parameter and attribute estimation 44
3.4.2 Structural modification 46
3.5 Experiments 48
3.5.1 Category modeling from RGB & RGB-D images 49
3.5.2 Experiments and quantitative evaluations 49
3.6 Conclusions 55
4 Visual Mining Beyond RGB Images: Use of RGB-D Images 57
4.1 Introduction 58
4.2 Related Work 62
4.3 Graphical model of object edge segments and graph matching 65
4.3.1 Model for object collection from RGB-D images 68
4.3.2 Category model for ordinary RGB images 70
4.3.3 HoG feature extraction 71
4.4 Model learning algorithms 71
4.4.1 Basic framework: model learning from RGB-D images 72
4.4.2 Object recognition based on category models 75
4.4.3 Technical extension 1: model learning via knowledge transfer 77
4.4.4 Technical extension 2: initial labeling refinement 79
4.5 Experiments 82
4.5.1 Data 83
4.5.2 Competing methods 83
4.5.3 Experiment 1: learning from a single labeled object 88
4.5.4 Experiment 2: Learning from refined object labeling 93
4.6 Conclusions 95
5 Extended Application 1: Model Mining for Single-View 3D Reconstruction . . . 97
5.1 Introduction 98
5.2 Related work 102

5.3 Training of category detectors 103

5.4 Learning 3D reconstruction 106

5.4.1 Object sample collection & pose normalization 107

5.4.2 Model learning 108

5.4.3 3D reconstruction based on category models 112

5.5 Experiments 113

5.5.1 Data 113

5.5.2 Implementation details 113

5.5.3 Quantitative comparison 115

5.6 Conclusions 120

6 Extended Application 2: Mining Deformable Models of Animals 121

6.1 ARG-based deformable model for animals 121

6.2 Experiments 124

7 Kinect RGB-D image dataset 125

8 Conclusion and future work 128

Acknowledgements 143

Curriculum Vitae 144

Chapter 1

Introduction

My search direction is “visual mining”. Actually, the concept of visual mining is newly defined by this thesis, and there is no clear definition of it presented in previous literatures. Generally speaking, visual mining can be understood as a new academic subfield of artificial intelligence that ranges across both field of computer vision and the field of data mining. In other words, visual mining aims to directly mine visual knowledge from ubiquitous and unlabeled visual data, instead of conventional style of learning from well prepared training samples.

Visual mining & big data: The concept of visual mining is of great value in the era of big data, because it provides a plausible solution to one of ultimate dreams in the field of computer vision, *i.e.* building a comprehensive model base that can provide object-level prior knowledge of all the daily-use categories for various visual tasks. Compared to conventional model training, the idea of mining does not require or just requires a small amount of human labeling to learn a model for each category. This is very important for the processing of big visual data that contains a vast number of categories.

However, on the other hand, such visual mining is one of the biggest bottlenecks in the field of computer vision. The most common case of big visual data is the ubiquitous images and videos that are casually captured in people daily life. Thus, the visual mining involves almost all the typical challenges in image processing, such as the unalignment of objects and the intra-category variations in texture, rotation, scale, and illumination. These propose continuous challenges to the state-of-the-art algorithms.

Therefore, in this thesis, I propose two visual-mining platforms, each of which simultaneously models these challenges using a single algorithm. To the best of my knowledge, my methods are the first techniques that can deal with all of these challenges, which breaks though conventional “model training” to the level of “model mining”.

Precise object-level visual mining: Considering that the concept of visual mining is very extensive, in this thesis, I just limit the discussion to the most typical case of visual mining, *i.e.* “precise object-level visual mining”. I want to mine model at the object (or category) level, rather than mine visual knowledge at the image level (*e.g.* the knowledge for image retrieval based on the whole scene in the image and that for separating the texture, shade, and light in an image). Therefore, my study can be regarded as building an “object-level visual dictionary” that can identify “what is where” in any arbitrary image. To some extent, people can regard the object-level symbolism based on the visual dictionary as the first step for high-level artificial intelligence of computer vision.

Actually, some directions of computer vision are close to the idea of visual mining, but there is still a distance between them and the concept of “precise object-level visual mining”. I need to note that the phrases of “precise” and “object-level” indicate the detailed knowledge for objects in each category, for example, abilities of accurately localizing target objects in cluttered environment and providing clear object boundaries. For commercial purposes, some daily-use objects are designed with various textures and structures. Therefore, the robustness to structure deformation and intra-category variations in texture and rotation should also be considered.

Deep learning is a new and hot direction for artificial intelligence and has made a significant influence on computer vision. However, it mainly learns an optimal set of features to represent the whole image or some typical kinds of scenes, rather than extract knowledge of specific small-size objects in large and unlabeled images with considerable intra-category variations. As one of main applications of deep learning, image recognition, *i.e.* providing a set of tag candidates to describe the content of each image, does not contain the knowledge as detailed as that at the level of “exact” objects, either. Then, topics of object discovery and co-segmentation can be understood, to some extent, as the mining of object-level knowledge. However, for the state-of-the-art algorithms, they mainly focus on the textural knowledge, *i.e.* techniques based on “bag-of-words” models, and it is still a challenge for them to encode detailed structural knowledge of objects. Therefore, perhaps, I can understand them as a kind of “fuzzy” object models, rather than “precise” object models encoding part-level object information with robustness to texture variations.

This is just a general discussion of related work close to visual mining. Please see the related-work sections of following chapters for detailed analysis.

General platform & extended applications: In spite of the discussion above, the concept of precise object-level visual mining is still too extensive, because the mining task can be oriented to many different visual applications, such as object-level tracking, segmentation, pose estimation, 3D reconstruction, and etc. Each of these applications has its own algorithms and challenges.

Therefore, I mainly focus on developing a general platform for visual mining, based on which mining tasks for other visual applications can be performed. In my opinion, the general platform should mine category models for object detection. It is because that target objects of different categories can be automatically discovered and labeled using the mined models, and the labeled objects can be applied to further model training for all the potential extended applications.

In this thesis, I propose general platforms for visual mining and present two examples of their extended applications. The rest of this thesis is organized as follows. Chapter 2 and Chapter 3 present two general platforms for visual mining that are based on attributed graph mining and unsupervised learning of graph matching, respectively. In Chapter 4, I introduce a visual-mining strategy that uses depth information in RGB-D images to improve the accuracy of model mining. The mined category models can be applied back to ordinary RGB images. Then, I introduce the extended applications of the proposed visual-mining platforms. Chapter 5 focuses on mining category models from ubiquitous RGB-D images for single-view 3D reconstruction, while Chapter 6 aims to mine deformable animal models from unlabeled videos. In addition, I have published a Kinect RGB-D image dataset oriented to challenges in graph matching, which has been used in my previous studies. Chapter 7 presents details of the dataset. Finally, the whole thesis if summarized in Chapter 8.

Chapter 2

A General Platform for Visual Mining: Attributed Graph Mining

The concept of "precise object-level visual mining" is extensive and has a wide range of applications, *e.g.* mining category models for tracking, segmentation, detection, and 3D reconstruction.

In this chapter, I design a general platform to guide the model mining from big visual data for all of these applications, although each of them has its own challenges. I propose "attributed graph mining" to directly discover a set of key object parts for a category from unaligned images to construct the category model, simultaneously determining the corresponding parts in the unaligned images. Actually, the estimated part correspondences between different images can be regarded as automatic object alignments at the part level in ubiquitous images. Thus, the models for other visual tasks, such as object tracking, segmentation, and 3D reconstruction, can be further trained using the part correspondences.

I categorize the attributed graph mining from both the view of graph mining and the view of learning graph matching. I can understand this study as both the first attempt to formulate the idea of mining maximal frequent subgraphs (MFSs) in challenging graph domain of attributed relational graphs (ARGs) and a concept extension for unsupervised learning of graph matching. I define the soft attributed pattern (SAP) to represent the common subgraph pattern among a set of ARGs, considering both the graphical structure and graph attributes. Regarding the differences between graph domain of ARGs and conventional labeled graphs, I propose a new mining strategy that directly extracts the SAP with the maximal graph size without applying node enumeration. Given an initial graph template and a number of ARGs, I modify the graph template into the maximal-size SAP with good matches to the ARGs in an unsupervised fashion.

The rest of this chapter is organized as follows. The introduction and discussion of related work are presented in Sections 2.1 and 2.2. Section 2.3 defines the target problem of attributed graph mining, and Section 2.4 presents the detailed algorithm. In Section 2.5, I introduce the design of experiments and evaluate the algorithm performance. Finally, the overall chapter is summarized in Section 2.6.

2.1 Introduction

In the era of big data, do you still plan to label a set of object samples for each category and train category models one-by-one? In this section, let me locate this research in both the areas of graph matching and graph mining, and introduce its contributions to ubiquitous learning from big visual data.

2.1.1 Views of graph matching & task introduction

Attributed relational graphs (ARGs) are widely used. For example, in computer vision, ARGs can represent either scenes or objects, using the local and pairwise attributes to describe part features and the spatial relationship between the parts, respectively. The goal attributed graph matching is to estimate node correspondences between two ARGs, *e.g.* a small ARG template of an object and a large ARG of an image, based on the similarity of local and pairwise attributes.

In the general case[1] of ARGs, the graph matching is typically formulated as a quadratic assignment problem (QAP), which requires global optimization.

Recently, a number of approaches to learning graph matching have been proposed. They

---

[1] Unlike ARGs in [1], local attributes in ARGs may not, in general, be sufficiently distinguished to independently provide matching correspondences or just matching candidates between ARGs without global optimization.

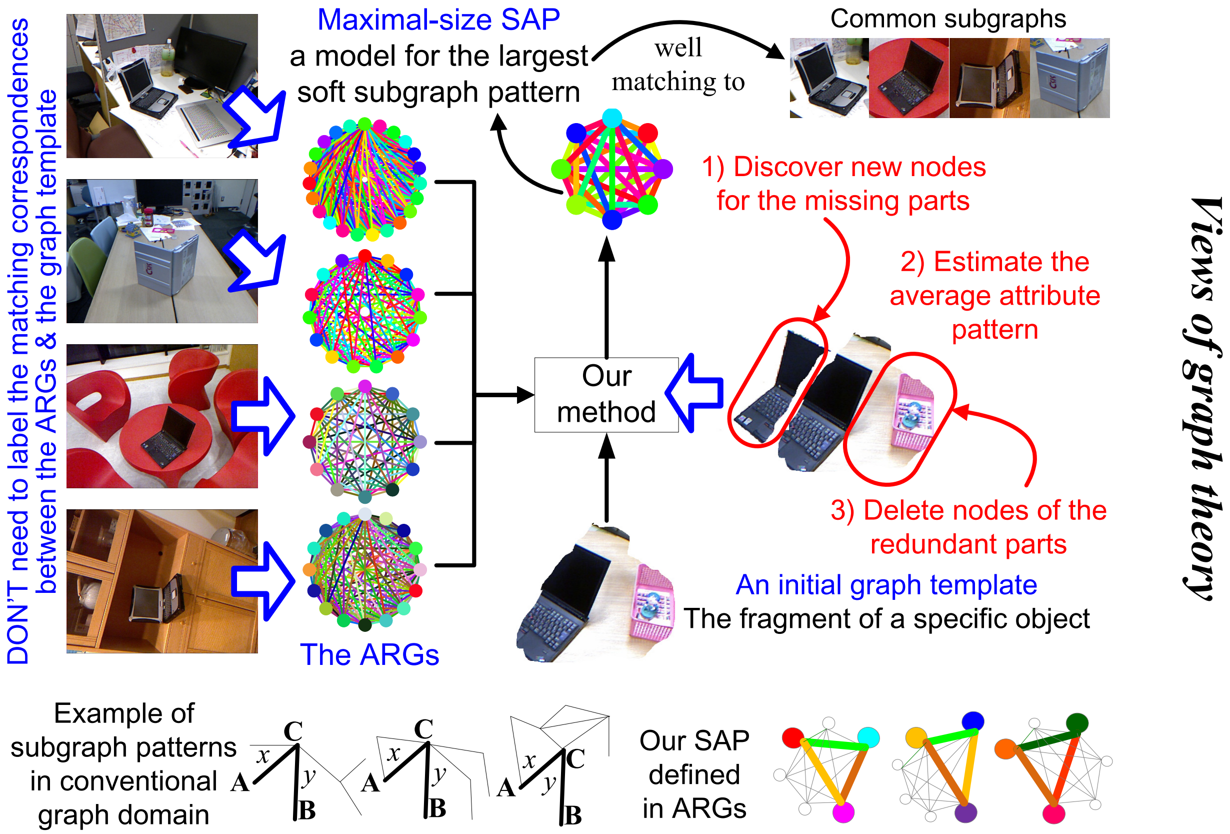

DON'T need to label the matching correspondences between the ARGs & the graph template
Maximal-size SAP
a model for the largest soft subgraph pattern
well matching to
Common subgraphs
1) Discover new nodes for the missing parts
2) Estimate the average attribute pattern
Our method
3) Delete nodes of the redundant parts
An initial graph template
The fragment of a specific object
The ARGs
Views of graph theory
Example of subgraph patterns in conventional graph domain
C
x
y
A
B
C
x
y
A
B
C
x
y
A
B
Our SAP defined in ARGs

Figure 2.1: Overview from views of the graph theory. I define and extract the soft attributed pattern (SAP) from ARGs, and the size of the SAP is maximized. This study overcomes a key challenge in graph mining, as it formulates the idea of mining maximal-size common subgraphs in challenging graph domain of ARGs. This method also extends the concept of unsupervised learning for graph matching. Given an initial graph template and a set of large ARGs, I simultaneously discover the missing nodes, delete redundant nodes, and train attributes, so as to obtain a graphical model with good matching performance.

train models or matching parameters to achieve good graph-matching performance, and their superior performance in terms of improving matching accuracy has been demonstrated. Indeed, the concept of learning graph matching has been extended. Generally speaking, I can categorize these approaches to supervised methods [2, 3, 4, 5, 6] and unsupervised methods [5, 7]. The unsupervised methods do not require people to manually label the matching correspondences of target objects in the ARGs for training, whereas the supervised methods do. In this study, I focus on unsupervised approaches, which are analogous to automatic category modeling from big visual data.

In this chapter, I propose a new concept of learning graph matching that focuses on the discovery of the missing[2] *graph parts (nodes) of the common subgraph pattern. Given a graph template and a number of ARGs, my method simultaneously 1) discovers missing parts of the template, 2) eliminates redundant parts, and 3) adjusts its attributes in an unsupervised manner, so as to grow the initial template into the common subgraph pattern among these ARGs, and achieve good matching performance (Fig. 2.1).* In other words, I focus on the ability to recover a full-size graphical pattern from a fragmentary graph template, as shown in Fig.2.5. Obviously, this is orthogonal to conventional unsupervised approaches that learn attribute weights [5] and refine the template structure [7, 8].

2.1.2 Views of graph mining & the proposed method

From another perspective, I can understand the proposed method as the mining of maximal-size[3] subgraph patterns. In fact, this is one of the core branches of graph mining, and many related techniques have been extensively investigated and developed in the past, including maximal frequent subgraph (MFS) extraction and maximal clique mining.

However, a bottleneck for mining maximal subgraph patterns lies in the strict constraints of the target graphs. Pioneering studies are mainly applied to “labeled graphs” (the graphs that have distinct node labels or edge labels) and the graphs with a list of pre-determined potential node correspondence candidates. Such graphs are usually generated from tabular data and have distinguishing structures.

---

[2] The missing part discovery is the key issue for growing the current fragmentary subgraph pattern into the maximal-size subgraph pattern. The missing pattern parts' the corresponding nodes in different ARGs should have similar unary and pairwise attributes, so as to maintain the fuzziness of the subgraph pattern.

[3] The word “maximal” indicates that I should grow the target subgraph pattern until the graph size of the pattern is maximized.

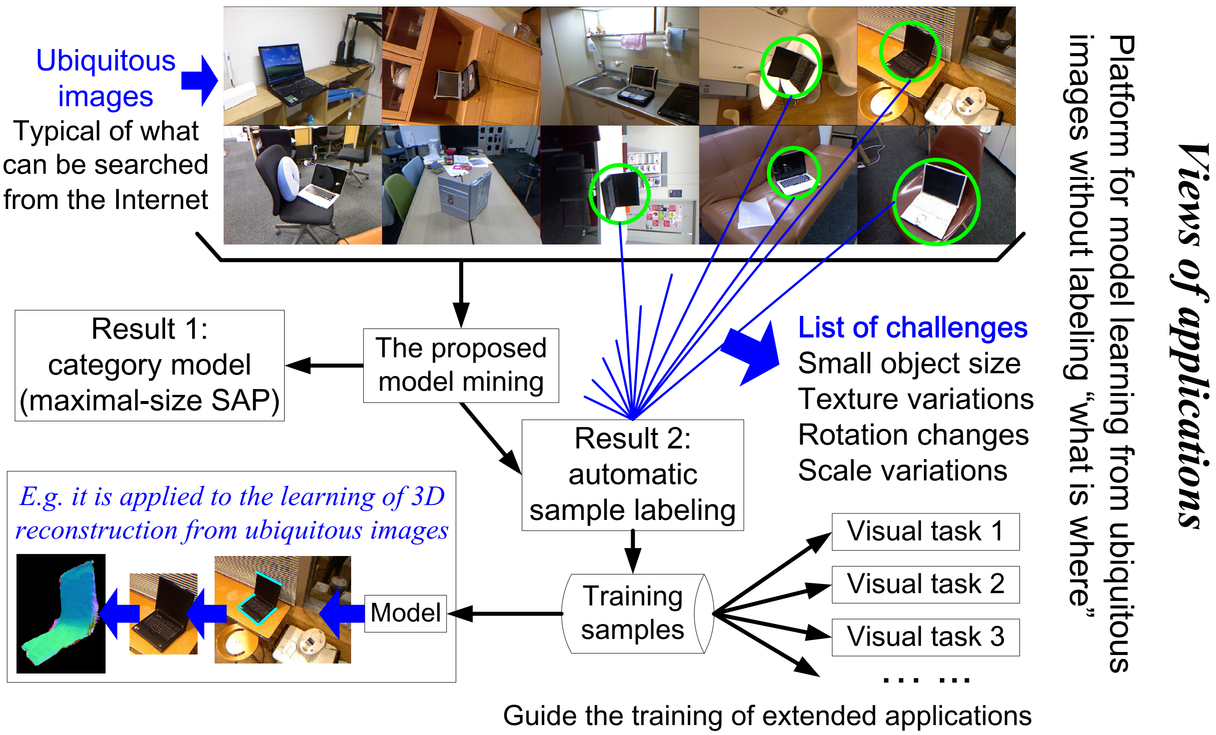

Ubiquitous images
Typical of what can be searched from the Internet
Result 1: category model (maximal-size SAP)
The proposed model mining
Result 2: automatic sample labeling
List of challenges
Small object size
Texture variations
Rotation changes
Scale variations
E.g. it is applied to the learning of 3D reconstruction from ubiquitous images
Model
Training samples
Visual task 1
Visual task 2
Visual task 3
… …
Guide the training of extended applications
Views of applications
Platform for model learning from ubiquitous images without labeling "what is where"

Figure 2.2: Overview from views of applications. The study proposes a platform for model learning and sample labeling in big visual data, which has a plenty of potential applications.

By contrast, when I extend this topic to "fuzzier[1]*" graph domain of ARGs, both the definition of the subgraph pattern and the mining method are much more challenging.* First, the correspondences between the pattern and the target subgraphs embedded in the ARGs can only be formulated as a QAP based global optimization. Second, conventional judgment of graph isomorphism between the pattern and these target subgraphs can no longer be applied to ARGs that have neither distinguishing structures nor node and edge labels, as shown in Fig. 2.1. *Alternatively, I redefine the subgraph pattern as a "soft" attributed pattern (SAP), which uses a threshold to limit its attribute differences from the target subgraphs and thereby maintain the pattern's significance.* Consequently, the mining process is actually to extract the SAP with the maximal graph size among the ARGs.

Moreover, the fuzzy definition of the SAP requests a new graph-mining methodology, as conventional approaches based on node enumeration (or node search) are all hampered in graph domain of ARGs. *Therefore, in this chapter, I design a mining strategy that grows the pattern size by directly discovering new pattern nodes from ARGs without any node enumeration.* I demonstrate the existence of an approximate solution to this strategy, when I use the typical squared differences to define the dissimilarity between the SAP and its target subgraphs.

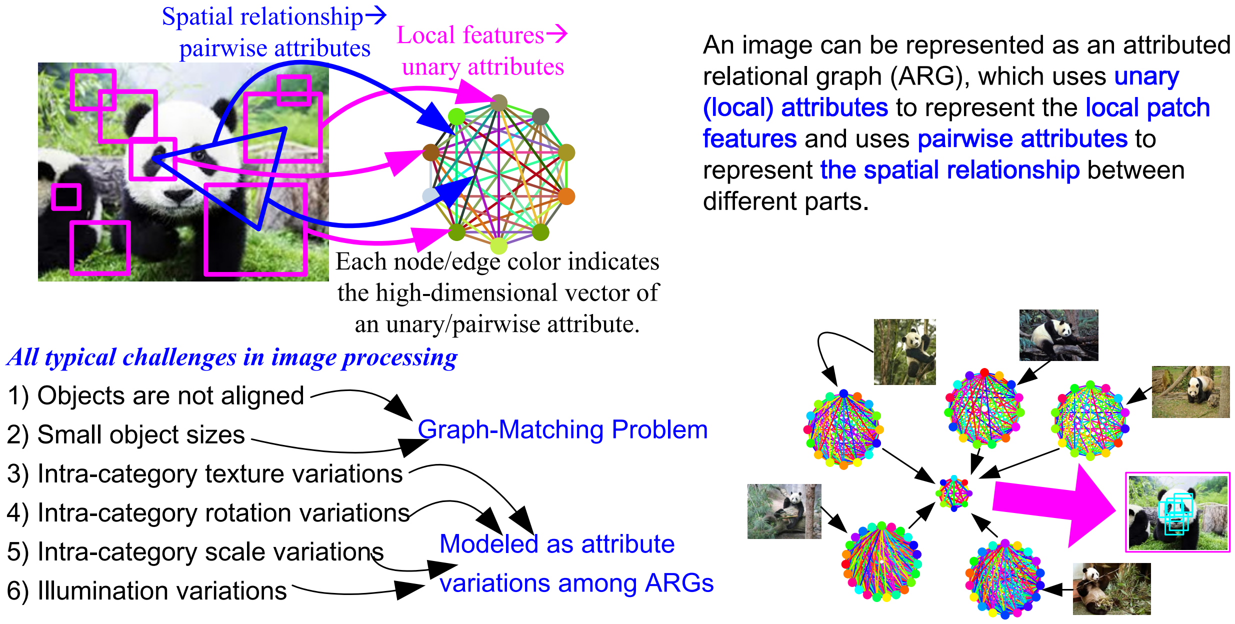
2.1.3 Platform for model mining from big visual data

I would rather regard this study as a method of "model mining" to differentiate from conventional "model learning". Perhaps, the boundary between learning and mining is just the watershed for the era of big visual data.

A main bottleneck for the whole computer vision field facing big visual data is the difficulty of model learning from ubiquitous images. If people have to prepare a set of training images to train a model for each category, the great cost of human labeling may hamper the ultimate goal of building an all-embracing knowledge base to provide object-level understanding of images. Therefore, it is of great value to mine category models from ubiquitous images without labeling "what is where" as an efficient way of category modeling.

However, learning from ubiquitous images involves almost all the typical challenges in computer vision. Such images are usually casually captured in people's daily life, and they are typical of what can be collected from the Internet using search engines. Target objects are usually small and randomly located in cluttered scenes with considerable intra-category variations in texture, rotation, and scale.

Therefore, I propose this graph-mining method as a general platform of learning from ubiquitous images. As shown in Fig. 2.3, the ubiquitous images can be represented by ARGs, and thus, objects in the target category within these images correspond to the common subgraphs embedded in the ARGs. Consequently, the maximal-size SAP can be considered as the category model. A good design of the unary and pairwise attributes in


Spatial relationship→ pairwise attributes
Local features→ unary attributes
Graph-Matching Problem
Modeled as attribute variations among ARGs

An image can be represented as an attributed relational graph (ARG), which uses unary (local) attributes to represent the local patch features and uses pairwise attributes to represent the spatial relationship between different parts.

Each node/edge color indicates the high-dimensional vector of an unary/pairwise attribute.

All typical challenges in image processing

1) Objects are not aligned
2) Small object sizes
3) Intra-category texture variations
4) Intra-category rotation variations
5) Intra-category scale variations
6) Illumination variations

Figure 2.3: ARGs for image representation and common subgraph patterns for model representation

the ARGs can ensure, to some extent, the robustness to both the feature variations of the object parts and their spatial relationship changes that are caused by variations in rotation and scale. In this case, the model-learning problem is equivalent to the mining of the maximal-size SAP.

Extended applications: Defining graph mining in ARGs has a vast range of applications. This approach can be applied to many academic fields, when the problems can be formulated using ARGs. However, in this chapter, I just limit my discussion within the range of potential extended applications in computer vision.

My method not only learns category models from cluttered scenes, but also simultaneously detects object parts. Such part-level object detection can be regarded as the automatic labeling of training samples in ubiquitous images, thereby being able to guide the training of many visual tasks, such as object recognition, tracking, segmentation, and etc. For example, based on this method, I can use the part-level labeling to learn 3D-reconstruction knowledge of each object category from informally collected RGB-D images, which has be achieved in [9]. I can use this method to recover global object shapes from fragments, as shown Fig. 2.5.

2.1.4 Summary

I summarize contributions of this chapter as follows. First, I redefine the concept of unsupervised learning for graph matching in order to idealize the spirit of training graphical structures. To the best of my knowledge, this study is the first attempt to encode the discovery of missing parts into the learning of graph matching. Second, in terms of graph mining, this research extends the mining of maximal-size subgraph patterns to challenging graph domain of ARGs. I propose the maximal-size SAP to define the subgraph pattern, and demonstrate the existence of a direct solution that does not require computationally intensive node enumeration. Both the pattern definition and the mining strategy are totally different from the mining approaches defined in conventional graph domain. Third, the proposed technique can be understood as a platform for model learning from big visual data, which automatically labels common objects in ubiquitous images. It can be used to guide many extended visual tasks.

A preliminary version of this chapter appeared in [10].

2.2 Related work

Views of graph matching: Given a graph template and a number of ARGs, conventional algorithms for learning graph matching [3, 4, 5, 6] mainly train matching parameters, and Cho *et al.* [2] proposed to learn a model for matching. Most of them are supervised methods that require the target subgraphs in ARGs to be labeled for training. Leordeanu *et al.* [5] proposed the first unsupervised method that did not require such manual labeling.

[7] considered the structural refinement as a part of the unsupervised learning for graph matching. [11] utilized a similar idea to mine spatial patterns from ARGs. [8] aimed to learn the node linkage, and this can also be regarded as structural refinement. However,

they only consider the matching between two ARGs.

Essentially, structural refinement just deletes “bad” nodes from the graph template, rather than recovering the prototype graphical patterns. Therefore, to perfect the learning of a graph structure, I encode the challenging task, *i.e.* the discovery of missing parts from large ARGs, into my definition of learning graph matching.

Views of graph mining: The discovery of missing parts relates this study to the mining of maximal-size[3] subgraph patterns, as it is usually meaningful to mine the pattern with the maximal graph size. This idea has been realized by MFS extraction [12] and maximal clique mining [13, 14] in the field of graph mining (reviewed in [15]).

As shown in Fig. 2.1(bottom), MFS extraction [12, 1, 16, 17] is based on graph isomorphisms and usually require 1) the distinguishing (topological) structure of the subgraph pattern, and 2) pre-defined distinct node/edge labels or potential inter-graph node correspondences determined by local consistency. Thus, node numeration is used to mine the MFS. The distinct graph structure and labels are used to prune the enumeration range, thereby avoiding possible NP-hard computation. Similarly, maximal clique mining [13, 14, 18, 19] mainly extracts a dense graph clique to maintain geometric consistency during matching. This method also requires distinguishing local features to pre-determine local matching correspondence candidates among graphs (as discussed in “Views of applications”).

In contrast, fuzzily defined ARGs usually have neither distinguishing structures nor distinct node labels. In many applications, nodes are connected in a uniform style and have not-so-strong local attributes. Thus, it requires a new mining strategy (without node enumeration) to deal with the ARGs. Considering the fuzzy condition[1], the matching between ARGs can only be solved by global optimization (*i.e.* a QAP). Thus, I redefine the common subgraph pattern as the SAP based on the attributes’ consistency, rather than a graph isomorphism *w.r.t.* the structure and labels.

Views of applications: For some certain graph-matching applications, the iterative methods for estimating common graph structures have been proposed [20, 21], which is a pioneer that discovered common structural patterns of edges from images. In fact, many visual applications involve the detection of common objects (or co-appearing parts) in a set of images. However, they are mainly designed with some data-driven techniques oriented to their own applications. For example, many studies of common object extraction from images [22, 23, 24, 25, 26, 27] use techniques related to maximal clique mining [13, 14] to some extent. They thereby require target objects to have high-quality patch features (with little texture variations), thus pre-determining a set of potential inter-image matching correspondences using local features. In contrast, my approach is formulated in the theory system of attributed graph matching. Thus, it is not limited to some specific CV applications, although I use certain ARGs generated from RGB and RGB-D images for testing.

2.3 Problem formulation

Definition 1 (ARG) *An ARG G is a three element tuple $G = (V, \mathbf{F}_V, \mathbf{F}_{V\times V})$, where V is the node set. Undirected edges connect each pair of nodes to form a completed graph. G contains N_P types of local attributes for each node and N_Q types of pairwise attributes for each edge. $\mathbf{F}_V = \{\mathcal{F}_i^s | s \in V, i = 1, 2, ..., N_P\}$ and $\mathbf{F}_{V\times V} = \{\mathcal{F}_j^{st} | s, t \in V, s \neq t, j = 1, 2, ..., N_Q\}$ denote the local and pairwise attribute sets, respectively. Each attribute corresponds to a feature vector.*

Actually, this definition can be extended to incomplete graphs with the form $G^* = (V, E, \mathbf{F}_V, \mathbf{F}_E)$. I can transform G^* to my ARG by setting a pairwise attribute $\mathcal{F}_j^{st} = 1$ if edge $(s,t) \in E$, and 0 otherwise.

Attributed graph matching: Given a set of ARGs $GS = \{G'_k | k = 1, 2, ..., N\}$, $G'_k = (V_k, \mathbf{F}_{V_k}, \mathbf{F}_{V_k\times V_k})$, the graph template $G = (V, \mathbf{F}_V, \mathbf{F}_{V\times V})$ represents an attribute pattern among the ARGs in GS, and is not exactly embedded in any G'_k. The matching between G and G'_k aims to compute a set of matching assignments between G and G'_k, denoted by $\mathbf{x}^k = \{x_s^k | s \in V\}$. Each matching assignment $x_s^k \in V_k \cup \{none\}$ maps node s

in G to a node in G'_k or a dummy choice *none*. *none* is used when some nodes in G do not exist in G'_k. The attributed graph matching is formulated as a typical QAP with the following energy function:

$$\mathcal{E}(\mathbf{x}^k|G,G'_k)=\sum_{s\in V}P_s(x^k_s|G,G'_k)+\sum_{(s,t)\in V,s\neq t}Q_{st}(x^k_s,x^k_t|G,G'_k) \tag{2.1}$$

Function $\mathcal{E}(\mathbf{x}^k|G,G'_k)$ indicates the total matching energy. The functions $P_s(\cdot)$ and $Q_{st}(\cdot,\cdot)$ denote matching penalties for local and pairwise attributes. Various graph matching optimization techniques can solve the energy minimization of $\mathcal{E}(\mathbf{x}^k|G,G'_k)$, and I choose TRW-S [28]. In this study, matching penalties are defined using squared differences.

$$P_s(x^k_s|G,G'_k)=\begin{cases}\sum_{i=1}^{N_P} w^P_i\|\mathcal{F}^s_i-\mathcal{F}^{x^k_s}_i\|^2, & x^k_s\in V_k\\ P_{none}, & x^k_s=none\end{cases} \tag{2.2a}$$

$$Q_{st}(x^k_s,x^k_t|G,G'_k)=\begin{cases}\frac{\sum_{j=1}^{N_Q}w^Q_j\|\mathcal{F}^{st}_j-\mathcal{F}^{x^k_s x^k_t}_j\|^2}{\|V\|-1}, & x^k_s\neq x^k_t\in V_k\\ +\infty, & x^k_s=x^k_t\in V_k\\ \frac{Q_{none}}{\|V\|-1}, & x^k_s \text{ or } x^k_t=none\end{cases} \tag{2.2b}$$

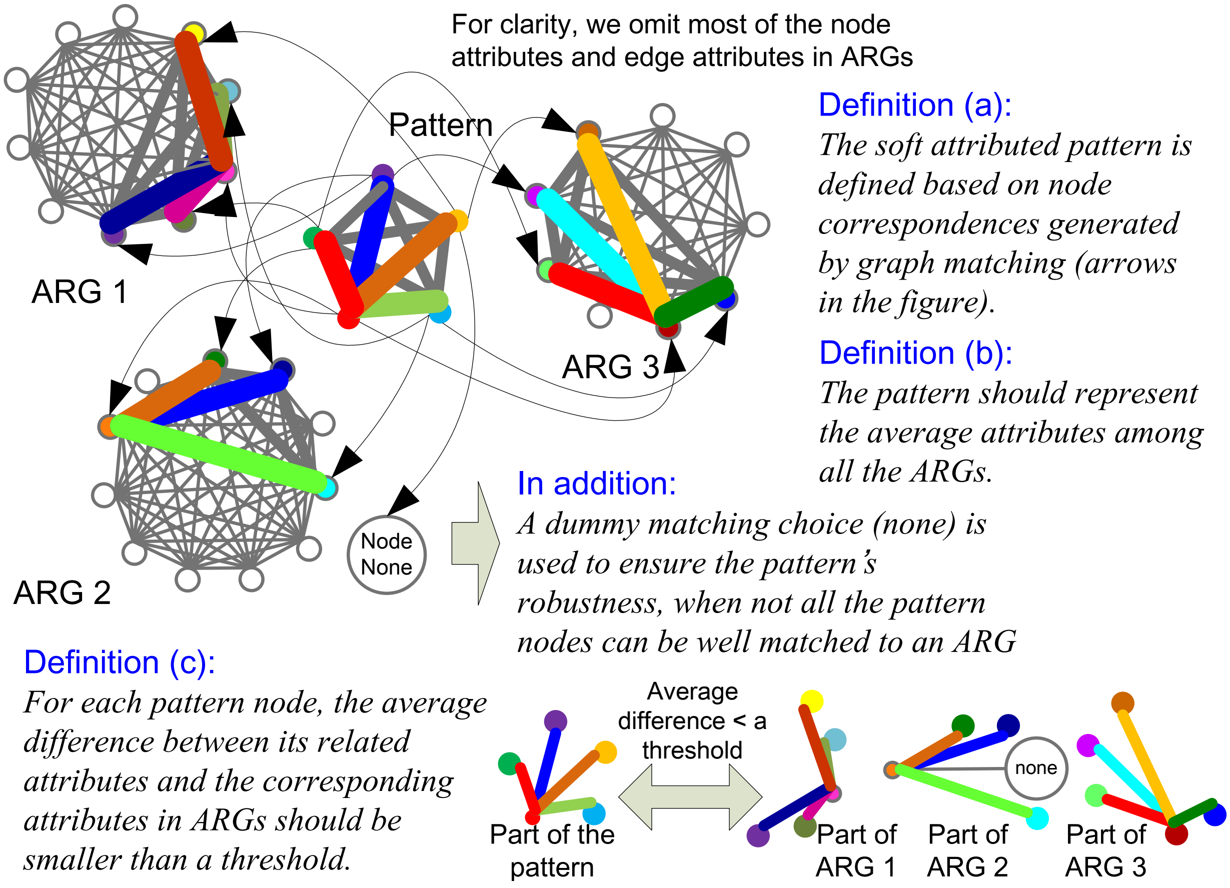
where P_{none} and Q_{none} are relatively large constant penalties for matching to *none*. $\|\cdot\|$ is the Euclidean norm. I use infinite penalties to avoid many-to-one matching assignments. w^P_i and w^Q_j denote the weights for local and pairwise attribute differences. I require the pairwise penalty to be symmetric, *i.e.* $Q_{st}(x_s,x_t|G,G'_k)=Q_{ts}(x_t,x_s|G,G'_k)$, and to be normalized[4] by $(\|V\|-1)$.

Definition 2 (SAP) *Given a set of ARGs* $GS=\{G'_k|k=1,2,...,N\}$ *and a threshold* τ, *a graph template* $G=(V,\mathbf{F}_V,\mathbf{F}_{V\times V})$ *is an SAP among the ARGs in* GS, *iff*

(a) $\hat{\mathbf{x}}^k=\mathrm{argmin}_{\mathbf{x}^k}\,\mathcal{E}(\mathbf{x}^k|G,G'_k)$; *I set* $\hat{\mathbf{x}}^k=\{x^k_s|s\in V,k=1,2,...,N\}$;

---

[4] During the learning process, I insert missing nodes into G and simultaneously delete redundant nodes to obtain the maximal-size SAP. However, the operation of node insertion (or delete) will increase (or decrease) the overall weights for pairwise attributes in both the graph matching (2.1) and the calculation of $E_s(\{\hat{\mathbf{x}}^k\}|G,GS)$, causing an unstable performance. Therefore, I normalize $Q_{st}(x_s,x_t|G,G'_k)$ using $(\|V\|-1)$ to prevent such effects.


For clarity, we omit most of the node attributes and edge attributes in ARGs
Pattern
ARG 1
ARG 2
ARG 3
Node None
Average difference < a threshold
Part of the pattern
none
Part of ARG 1
Part of ARG 2
Part of ARG 3

Definition (a):
The soft attributed pattern is defined based on node correspondences generated by graph matching (arrows in the figure).

Definition (b):
The pattern should represent the average attributes among all the ARGs.

In addition:
A dummy matching choice (none) is used to ensure the pattern's robustness, when not all the pattern nodes can be well matched to an ARG

Definition (c):
For each pattern node, the average difference between its related attributes and the corresponding attributes in ARGs should be smaller than a threshold.

Figure 2.4: Visualization of the SAP in Definition 2. Colors in ARGs denote different local and pairwise attributes. Note that in graph matching, I use pairwise attributes (edge colors), rather than only geometric distances between nodes (although such distances can be used as one of the N_Q types of pairwise attributes).

(b) $(\mathbf{F}_V,\mathbf{F}_{V\times V}) \leftarrow \mathrm{argmin}_{\mathbf{F}_V,\mathbf{F}_{V\times V}} \sum_{k=1}^{N} \mathcal{E}(\hat{\mathbf{x}}^k|G,G'_k)$;

(c) $\forall s \in V,\ E_s(\{\hat{\mathbf{x}}^k\}|G,GS) \leq \tau$;

where $E_s(\{\hat{\mathbf{x}}^k\}|G,GS)$ *is defined as the average matching penalty related to node* s *in* G *among all the ARGs in* GS.

$$E_s(\{\hat{\mathbf{x}}^k\}|G,GS)=\frac{1}{N}\sum_{k=1}^{N}\Big[P_s(\hat{x}_s^k|G,G'_k)+\sum_{t\in V,t\neq s}Q_{st}(\hat{x}_s^k,\hat{x}_t^k|G,G'_k)\Big]$$

Maximal SAP: The definition of the SAP can be visualized in Fig. 2.4, and I introduce the physical meaning of each item in Definition 2, as follows.

Condition (a) directly matches the SAP G to each large ARG G'_k in GS to determine the SAP's corresponding subgraphs embedded in these ARGs.

Condition (b) trains the local and pairwise attributes of the SAP G. G should represent the average attribute pattern among all its corresponding subgraphs determined by ***Condition (a)***. In other words, the SAP's attributes $(\mathbf{F}_V, \mathbf{F}_{V\times V})$ should minimize the total matching energy, given all the matches between G and the ARGs in GS.

Condition (c) sets a threshold τ to control the fuzziness of G. I require each node s in the SAP to have a low average matching penalty among all the matches to ensure that all the SAP's nodes represent the common parts in the ARGs.

With these preliminaries, my goal is to mine the SAP G with maximal graph size $\|V\|$, *i.e.* the largest common subgraph pattern among the ARGs.

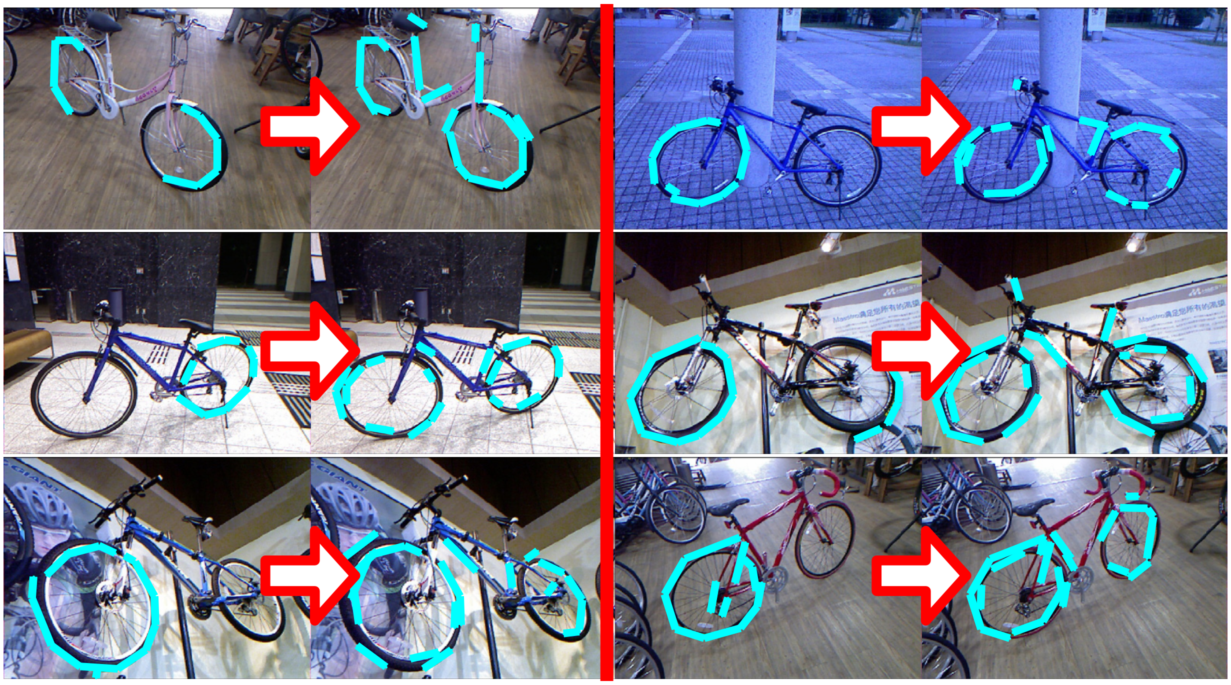


Figure 2.5: Structure modification from different graph templates (object fragments) to SAPs (fuzziness $\tau = 0.4$).

Algorithm 1 Maximal SAP extraction

```
Input: The initial graph template $G = (V, \mathbf{F}_V, \mathbf{F}_{V\times V})$; a set of ARGs $GS = \{G'_k | k = 1, 2, ..., N\}$, where $G'_k = (V_k, \mathbf{F}_{V_k}, \mathbf{F}_{V_k\times V_k})$; a threshold $\tau$ controlling the SAP's fuzziness; the maximum iteration number $M$.
for $iteration = 1$ to $M$ do
    1. Use the current $G$ to estimate matching assignments in all the $N$ ARGs as $\{\hat{\mathbf{x}}^k\}$ (see Definition 2(a)).
    2. Given $\{\hat{\mathbf{x}}^k\}$, update the attribute sets $\mathbf{F}_V$ and $\mathbf{F}_{V\times V}$ of $G$ (see (2.3) and Definition 2(b)).
    3. With the updated attributes, compute the local matching penalty $E_s(\{\hat{\mathbf{x}}^k\}|G, GS)$ of each node $s$ in $G$ matching the ARGs in $GS$. Select the worst node $\hat{s} = \arg\max_{s\in V} E_s(\{\hat{\mathbf{x}}^k\}|G, GS)$, and if $E_{\hat{s}}(\{\hat{\mathbf{x}}^k\}|G, GS) > \tau$, then delete $\hat{s}$ from $G$ (see Definition 2(c)).
    4. Create a new node $y$ as the potential missing node of $G$, and thus construct $G^{new}$. Estimate the optimal attributes and matching correspondences for $y$ (see (2.6) and (2.8)). If $E_y(\{\mathbf{x}^k_{new}\}|G^{new}, GS) \leq \tau$, then insert node $y$ into $G$. (see Definition 2(c))
end for
```

2.4 Algorithm

To extract a maximal SAP, the initial graph template G is modified in the following EM framework. In each iteration, I use the current G to estimate the matching assignments in the ARGs in GS, $\{\hat{\mathbf{x}}^k\}$, and then use $\{\hat{\mathbf{x}}^k\}$ to update the attribute sets of $\mathbf{F}_V$ and $\mathbf{F}_{V\times V}$ of G. The new $\mathbf{F}_V$ and $\mathbf{F}_{V\times V}$ are finally used as feedback to modify the structure of G by (probably) discovering a missing node from the ARGs, or deleting a redundant one. Thus, the initial graph template G is iteratively modified to the maximal SAP (see Fig. 2.5).

Attribute estimation: According to Definition 2(a), matching assignments $\{\hat{\mathbf{x}}^k\}$ are first estimated based on the current G. I then use Definition 2(b) to estimate the local and pairwise attributes of G, given $\{\hat{\mathbf{x}}^k\}$. As $\mathcal{E}(\hat{\mathbf{x}}^k|G, G'_k)$ is a convex function with respect to $\mathbf{F}_V$ and $\mathbf{F}_{V\times V}$ (see (2.1) and (2.2)), the minimization problem can be directly solved by $\frac{\partial \sum_{k=1}^{N} \mathcal{E}(\hat{\mathbf{x}}^k|G,G'_k)}{\partial \mathcal{F}_i^s} = 0$ and $\frac{\partial \sum_{k=1}^{N} \mathcal{E}(\hat{\mathbf{x}}^k|G,G'_k)}{\partial \mathcal{F}_i^{st}} = 0$. I thus have that G's attributes are equal to the average attributes among all subgraphs matched to G in the ARGs.

$$\begin{aligned} \mathcal{F}_i^s &= \underset{k:\delta(\hat{x}_s^k)=1}{\text{average}} \mathcal{F}_i^{\hat{x}_s^k} \\ \mathcal{F}_i^{st} &= \underset{k:\delta(\hat{x}_s^k)\delta(\hat{x}_t^k)=1}{\text{average}} \mathcal{F}_i^{\hat{x}_s^k \hat{x}_t^k} \end{aligned} \tag{2.3}$$

where $\delta(\cdot)$ indicates whether a node in G is matched to *none*. If $\hat{x}_s^k = none$, $\delta(\hat{x}_s^k)$ is set to 0; otherwise 1.

Structure modification: I grow the initial G into the maximal SAP using a greedy strategy (see Algorithm 1). In each iteration, I delete the "worst" (not well matched to the ARGs) node from G, and insert the "most probable" missing node. Both the insertion and elimination depend on the unified requirement for the local matching quality in Definition 2(c). I choose node $\hat{s} = \arg\max_{s\in V} E_s(\{\hat{\mathbf{x}}^k\}|G,GS)$ in G. If $E_{\hat{s}}(\{\hat{\mathbf{x}}^k\}|G,GS) > \tau$, I delete $\hat{s}$ from G; otherwise, this node is retained.

The key part is the node insertion. This involves two issues, i.e. *the attribute estimation of the new node and the determination of its matching assignments to the ARGs. However, this has the appearance of a chicken-and-egg problem.* On the one hand, the local and pairwise attributes related to the new node represent the pattern of their corresponding nodes and edges in ARGs[5]. They are thus determined by the matching assignments of the new node (see Definition 2(b)). On the other hand, the matching of the new node cannot be applied without knowing its attributes.

Fortunately, I have developed an efficient solution that simultaneously determines the attributes and matching assignments of the missing node, thus overcoming the chicken-and-egg problem. Let y be the missing node of G, and let $\mathbf{F}_y = \{\mathcal{F}_i^y | 1 \le i \le N_P\}$ and

---

[5] Conventional enumeration of the new nodes in the graphs cannot ensure the algorithm's stability, as the attributes of the enumerated nodes may be greatly biased. Moreover, owing to the existence of the dummy matching choice ("none"), I cannot limit the node enumeration within any single ARG to reduce computation.

$\mathbf{F}_{\{y\}\times V} = \{\mathcal{F}_j^{yt}, \mathcal{F}_j^{ty} | t \in V, 1 \leq j \leq N_Q\}$ denote the local and pairwise attribute sets related to y. Consequently, in ARG G'_k, the node matched by y can be denoted by $x_y^k \in V_k \setminus \hat{\mathbf{x}}^k$ ($\hat{\mathbf{x}}^k = \{\hat{x}_s^k | s \in V\}$). Thus, y's matching assignments in all the ARGs are denoted by $\{x_y^k | k = 1, 2, .., N\}$.

I use $G^{new}=(V^{new}, \mathbf{F}_{V^{new}}, \mathbf{F}_{V^{new}\times V^{new}})$ to denote the dummy enlarged model after node insertion. I define the notation for G^{new} in the same way as that for G: $V^{new} = V \cup \{y\}$, $\mathbf{F}_{V^{new}} = \mathbf{F}_V \cup \mathbf{F}_y$, $\mathbf{F}_{V^{new}\times V^{new}} = \mathbf{F}_{V\times V} \cup \mathbf{F}_{\{y\}\times V}$, $\mathbf{x}_{new}^k = \hat{\mathbf{x}}^k \cup \{x_y^k\}$.

Thus, the local matching penalty of y is transformed to

$$\begin{aligned} &E_y(\{\mathbf{x}_{new}^k\} | G^{new}, GS) = \mathbf{P}_y + \sum\nolimits_{t\in V} \mathbf{Q}_{yt} \\ &\mathbf{P}_y = \sum\nolimits_{k=1}^{N} P_y(x_y^k | G^{new}, G'_k)/N \\ &\mathbf{Q}_{yt} = \sum\nolimits_{k=1}^{N} Q_{yt}(x_y^k, \hat{x}_t^k | G^{new}, G'_k)/N \end{aligned} \tag{2.4}$$

The goal of node insertion is transformed to

$$\underset{\mathbf{F}_y, \mathbf{F}_{\{y\}\times V}}{\arg\min} \sum\nolimits_{k=1}^{N} \mathcal{E}(\mathbf{x}_{new}^k | G^{new}, G'_k) \tag{2.5a}$$

$$\underset{\{x_y^k\}}{\text{argmin}} E_y(\{\mathbf{x}_{new}^k\} | G^{new}, GS) \tag{2.5b}$$

Equation 2.5a corresponds to Definition 2(b). Given the matching assignments of y in the ARGs ($\{x_y^k\}$), y's attributes ($\mathbf{F}_y, \mathbf{F}_{\{y\}\times V}$) are trained to minimize the matching energy, *i.e.* representing the attributed pattern among the ARGs.

Equation 2.5b estimates y's matching assignments ($\{x_y^k\}$) based on Definition 2(c). Given the attributes of y, the nodes in ARGs matched by y ($\{x_y^k\}$) should have similar attributes to y. In other words, they should minimize the local matching penalty ($E_y(\{\mathbf{x}_{new}^k\} | G^{new}, GS)$). If $E_y(\{\mathbf{x}_{new}^k\} | G^{new}, GS) < \tau$, then y satisfies Definition 2(c).

Similar to (2.3), attributes in $\mathbf{F}_y$ and $\mathbf{F}_{\{y\}\times V}$ are represented by x_y^k as the solution to (2.5a).

$$\mathcal{F}_i^y = \underset{k:\delta(x_y^k)=1}{\text{average}} \mathcal{F}_i^{x_y^k} = \sum\nolimits_{k=1}^{N} \mathcal{F}_i^{x_y^k}/N$$

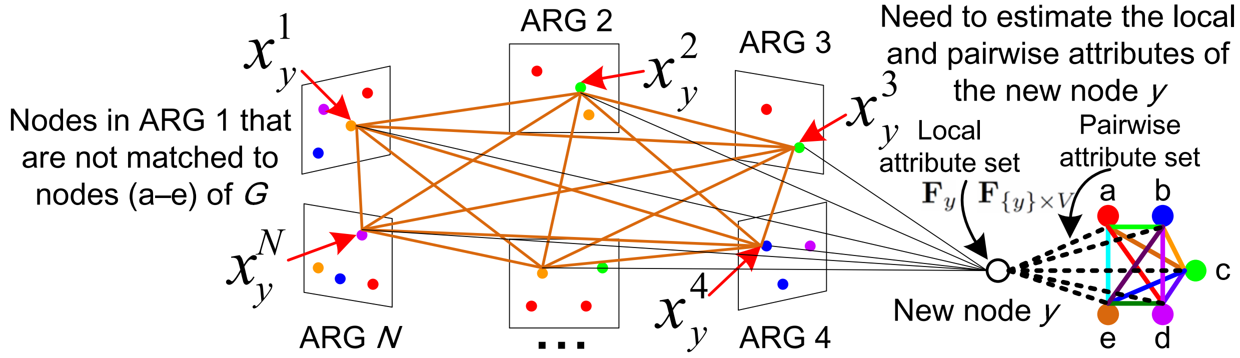

x_y^1
ARG 2
x_y^2
ARG 3
x_y^3
Need to estimate the local and pairwise attributes of the new node *y*
Nodes in ARG 1 that are not matched to nodes (a–e) of *G*
Local attribute set
Pairwise attribute set
$\mathbf{F}_y$
$\mathbf{F}_{\{y\}\times V}$
a
b
c
x_y^N
New node *y*
e
d
ARG *N*
...
x_y^4
ARG 4

Figure 2.6: Discovery of the missing node y in G. I have demonstrated a direct solution to the determination of y's matching assignments $\{x_y^k\}$ in the N ARGs that minimize $E_y(\{\mathbf{x}_{new}^k\}|G^{new}, GS)$, without requiring any prior knowledge of y's attributes. The ARGs are connected to each other to construct a Markov random field that solves this problem.

$$\mathcal{F}_i^{yt} = \underset{k:\delta(x_y^k)\delta(\hat{x}_t^k)=1}{\text{average}} \mathcal{F}_i^{x_y^k \hat{x}_t^k} = \underset{k:\delta(\hat{x}_t^k)=1}{\text{average}} \mathcal{F}_i^{x_y^k \hat{x}_t^k} \tag{2.6}$$

$$\mathcal{F}_i^{ty} = \underset{k:\delta(\hat{x}_t^k)\delta(x_y^k)=1}{\text{average}} \mathcal{F}_i^{\hat{x}_t^k x_y^k} = \underset{k:\delta(\hat{x}_t^k)=1}{\text{average}} \mathcal{F}_i^{\hat{x}_t^k x_y^k}$$

where $\delta(x_y^k)$=1, k=1,2,...,N. This is because, as the new node y should be well matched to most of the ARGs, I approximate $\mathcal{F}_i^y$, $\mathcal{F}_i^{yt}$, and $\mathcal{F}_i^{ty}$ by ignoring the possibility of matching y to *none*, so as to simplify the calculation. I substitute $\mathcal{F}_i^y$ and $\mathcal{F}_i^{yt}$ into $\mathbf{P}_y$ and $\mathbf{Q}_{yt}$ in (2.4). In the appendix, I demonstrate that

$$\mathbf{P}_y = \frac{1}{2N^2} \sum_{i=1}^{N_P} w_i^P \sum_{1\le k,l \le N} \|\mathcal{F}_i^{x_y^k} - \mathcal{F}_i^{x_y^l}\|^2 \tag{2.7}$$

$$\mathbf{Q}_{yt} = \frac{\sum_{i=1}^{N_Q} w_i^Q \sum_{k,l:\delta(\hat{x}_t^k)\delta(\hat{x}_t^l)=1} \|\mathcal{F}_i^{x_y^k \hat{x}_t^k} - \mathcal{F}_i^{x_y^l \hat{x}_t^l}\|^2}{2\|V\|N \sum_j \delta(\hat{x}_t^j)} + C_t$$

where $C_t = \sum_{k:\delta(\hat{x}_t^k)=0} \frac{Q_{none}}{\|V\|N} = \frac{Q_{none}\sum_k(1-\delta(\hat{x}_t^k))}{\|V\|N}$ is a constant *w.r.t.* x_y^k, given $\{\hat{\mathbf{x}}^k\}$. Because

C_t is a constant *w.r.t.* $\{x_y^k\}$, I insert $\mathbf{P}_y$ and $\mathbf{Q}_{yt}$ into (2.4) and (2.5b), and obtain

$$\begin{aligned}
&\underset{\{x_y^k\}}{\operatorname{argmin}} E_y(\{\mathbf{x}_{new}^k\}|G^{new}, GS)\\
=&\underset{\{x_y^k\}}{\operatorname{argmin}} \big\{\mathbf{P}_y + \textstyle\sum_{t\in V}\mathbf{Q}_{yt}\big\}\\
=&\underset{\{x_y^k\}}{\operatorname{argmin}} \sum_{1\leq k,l\leq N} M_{kl}(x_y^k, x_y^l),\\
&\text{where } M_{kl}(x_y^k, x_y^l) = \frac{1}{2N^2}\sum_{i=1}^{N_P} w_i^P \|\mathcal{F}_i^{x_y^k} - \mathcal{F}_i^{x_y^l}\|^2\\
&\qquad + \sum_{t\in V:\delta(\hat{x}_t^k)\delta(\hat{x}_t^l)=1} \frac{\sum_{i=1}^{N_Q} w_i^Q \|\mathcal{F}_i^{x_y^k\hat{x}_t^k} - \mathcal{F}_i^{x_y^l\hat{x}_t^l}\|^2}{2\|V\|N\sum_j \delta(\hat{x}_t^j)}
\end{aligned} \tag{2.8}$$

Thus, the problem of (2.5b) is transformed to a QAP, which can be directly solved using a Markov random field (MRF). As shown in Fig. 2.6, the ARGs are connected to each other to construct the MRF and determine y's matching assignments $\{x_y^k\}$. In this study, I use TRW-S [28] to solve the energy minimization of the MRF. y's attributes $\mathbf{F}_y$ and $\mathbf{F}_{\{y\}\times V}$ are computed by (2.6). If $E_y(\{\mathbf{x}_{new}^k\}|G^{new}, GS) \leq \tau$, I replace G by the enlarged graph template G^{new}.

2.5 Experiments

The proposed method is meaningful in the field of computer vision, enabling the discovery of a general category model for image matching when the target objects are randomly placed in large and cluttered scenes. In particular, my technique satisfies the condition of relatively weak local attributes for matching. I have designed two experiments to validate the proposed method on the ARGs generated from RGB and RGB-D images. I compare the proposed method with unsupervised approaches to learning graph matching, although the discovery of missing nodes is orthogonal to conventional learning of attribute weights.

I design different experiments to evaluate the proposed method in different visual tasks. In Experiments 1 and 2, I test my method for mining category models (*i.e.* maximal-size SAPs) from cluttered RGB-D and RGB images, respectively. The category model is constructed using object edge segments as graph nodes, and thus it has good performance of detecting objects with clear edges. Then, in Experiment 3, I extend the model-mining task to more general images, *i.e.* the images searched by search engines from the Internet. Therefore, I propose another ARG model, which takes interesting points of SIFT features as graph nodes, to describe objects in general images.

2.5.1 Experiment 1: mining edge-based models from cluttered RGB-D images

Dataset

As introduced in Chapter 7, I use the category dataset of Kinect RGB-D images [30, 29], published as a standard RGB-D object dataset[6] for graph-matching-based model learning. This dataset have been applied with [29] and the competing method [7]. The seven largest categories—*notebook PC*, *drink box*, *basket*, *bucket*, *sprayer*, *dustpan*, and *bicycle*—in this dataset contain enough RGB-D objects, and are chosen for training. These images depict cluttered scenes containing objects with different textures and rotations, and both experiments were performed on these scenes.

ARG-based category models

I have designed the ARGs to represent objects in RGB-D images in my previous studies [29, 7]. The category model is mined as the SAP among the ARGs. Besides, I also further propose a method for model-based object recognition, *i.e.* identifying whether the object matched by the model is a true detection of the target category.

The ARGs are designed as follows. I first use the edge extraction method [31] to extract object edges from images and then discretize continuous edges into line segments as the graph nodes, as illustrated in Fig. 2.7. Technical details for edge segmentation

---

[6] This is one of the largest RGB-D object datasets, and fits the requirements of learning graph matching.

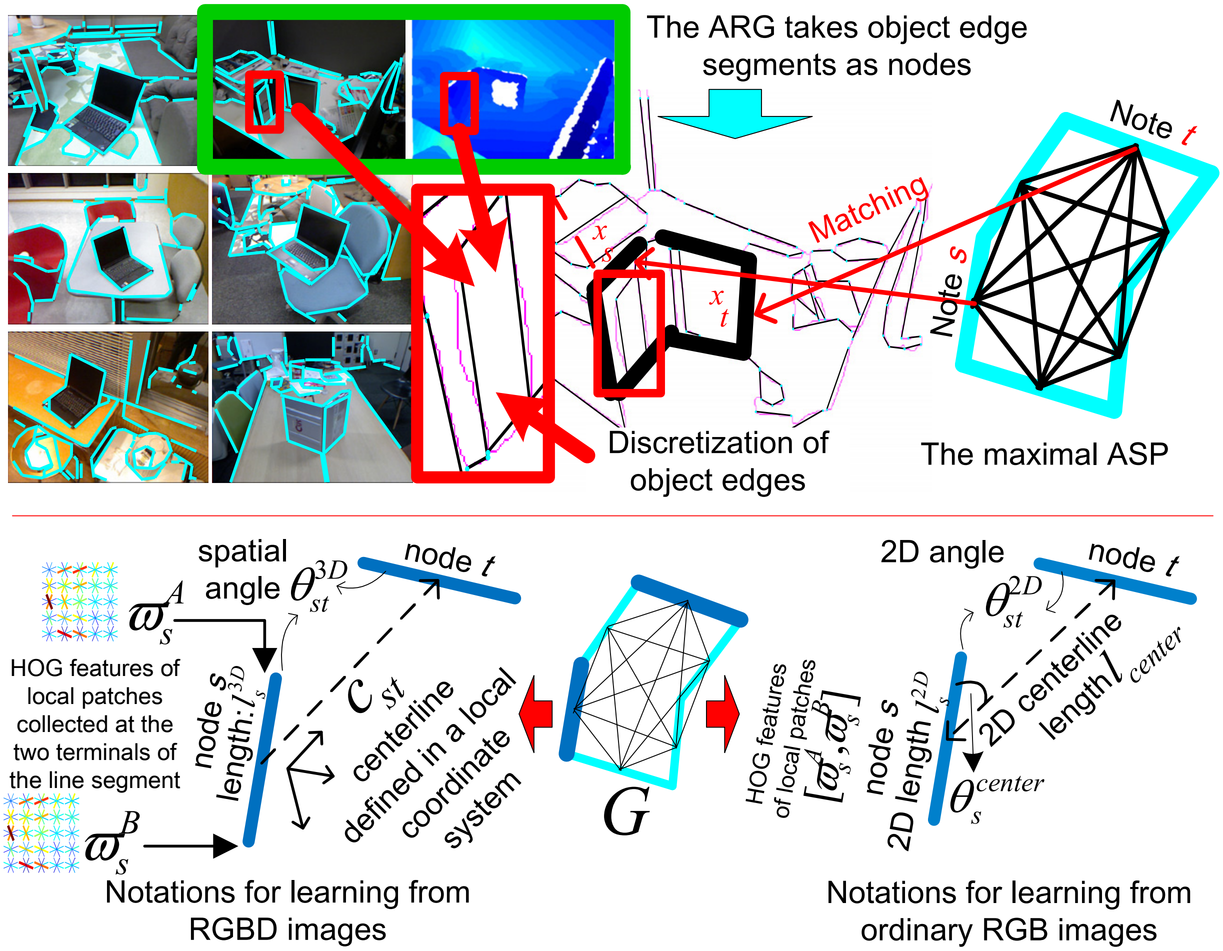

The ARG takes object edge segments as nodes
Note t
Matching
x_s
x_t
Note s
Discretization of object edges
The maximal ASP
spatial angle θ_{st}^{3D}
node t
ϖ_s^A
HOG features of local patches collected at the two terminals of the line segment
node s length: l_s^{3D}
c_{st} centerline defined in a local coordinate system
ϖ_s^B
G
Notations for learning from RGBD images
2D angle
node t
θ_{st}^{2D}
HOG features of local patches $[\varpi_s^A, \varpi_s^B]$
node s 2D length l_s^{2D}
2D centerline length l^{center}
θ_s^{center}
Notations for learning from ordinary RGB images

Figure 2.7: Notation for the ARGs based on line segments of object edges in RGB and RGB-D images [7]. Please see [29, 7] for more details of attribute settings.

are introduced in [29]. I connect each pair of the graph nodes to construct a complete graph. Two local features ($N_P = 2$) and three pairwise attributes ($N_Q = 3$) are designed to describe local features and spatial relationship between local parts in images.

The first unary attribute is the HOG features [32] of two local patches collected at line segment terminals of node s, denoted by $\mathcal{F}_1^s = [\varpi_s^A, \varpi_s^B]$[7]. The HoG features [32] are extracted using 5×5 cells, each of which covers half of its neighboring cells. I use four orientation bins (from 0° to 180°) to compute the gradient histogram in each cell. Because the patch is locally collected without significant illumination changes, I normalize all of the cells within a single block. The second unary attribute is given by $\mathcal{F}_2^s = \log l_s^{3D}$, where l_s^{3D} is the spatial length of the line segment of node s. The first of the three pairwise attributes, $\mathcal{F}_1^{st} = \theta_{st}^{3D}$, denotes the spatial angle between the line segments of nodes s and t in the 3D space. For each edge (s, t), I define the *centerline* as the line connecting the centers of the line segments of s and t. I measure the centerline in a local 3D coordinate system independent of the global object rotation, as the relative spatial translation between two nodes, denoted by $\mathbf{c}_{st}$. Based on this, the second and third pairwise attributes, *i.e.* $\mathcal{F}_2^{st} = \|\mathbf{c}_{st}\|$ and $\mathcal{F}_3^{st} = \mathbf{c}_{st}/\|\mathbf{c}_{st}\|$, represent the length and local orientation of the centerline, respectively. I set the attribute weights as $w_1^P = 0.2$, $w_2^P = 0.1$, $w_{j=1,2,3}^Q = 1$, and set P_{none} and Q_{none} as 0.4 and 0.2, respectively, which are uniformly applied to the model mining for all the categories.

Technical details

I use different thresholds τ from during the mining process to mine the category model (*i.e.* the maximal-size SAP) with different loose constraints. Larger threshold values τ indicate a fuzzier level of the maximal-size SAP, and lead to a larger graph size.

Given each specific setting of τ, I perform the evaluation via cross validation, as in [7, 29]. I pick each RGB-D image in a category to start an individual mining process, thus obtaining a set of maximal-size SAPs. To extract each maximal-size SAP, the target object

---

[7] Actually, this attribute is affected by the order of the two terminals, but there is no good ways to pre-define this order without considering terminal orders of other nodes. Therefore, I slightly modify the distance measurement of $\mathcal{F}_1^s$ in Equation 2.2 to $\|\mathcal{F}_1^s - \mathcal{F}_1^{x_s^k}\|^2$ to $\min\{\|\mathcal{F}_1^s - [\varpi_{x_s^k}^A, \varpi_{x_s^k}^B]\|^2, \|\mathcal{F}_1^s - [\varpi_{x_s^k}^B, \varpi_{x_s^k}^A]\|^2\}$. The final terminal order of each node $\hat{x}_s^k$ in ARG G_k' is determined as the order that best matches to node s in G.

in the picked image is labeled as the initial graph template, and I then randomly select 2/3 and 1/3 of the remaining images for training and testing, respectively. I design different evaluation metrics, which will be introduced in Section 2.5.4. Based on these metrics, the proposed method is evaluated using the average performance among all the mined SAPs.

2.5.2 Experiment 2: mining edge-based models from cluttered RGB images

This experiment is very similar to Experiment 1 in Section 2.5.1. I use the same dataset of Kinect RGB-D images, but only the RGB channels of the RGB-D images are used in this experiment. Therefore, I design a new type of ARGs for object representation in the RGB images. I test the model-mining performance under different settings of the parameter τ. The overall performance is also evaluated via cross validation.

ARG-based category models

The category model for RGB images is designed in the same way as the model for RGB-D images. The model uses edge segments as graph nodes, and I use one unary attribute ($n^U = 1$) and three pairwise attributes ($n^P = 3$) for the model. The notation is also illustrated in Fig. 2.7. The only attribute is the HOG features collected at the terminals of line segments, the same as the first unary attribute for RGB-D images. The first of the three pairwise attributes between nodes s and t is the angle between their line segments, denoted by $\mathcal{F}_1^{st} = \theta_{st}^{2D}$. The second pairwise attribute describes the angles between the centerline and the node line segments, denoted by $\mathcal{F}_2^{st} = [\theta_s^{center}, \theta_t^{center}]$, where θ_s^{center} is the angle between the line segment of s and the centerline. The third pairwise attribute represents relative segment lengths, and is denoted by $\mathcal{F}_3^{st} = \frac{1}{l_{center}}[l_s^{2D}, l_t^{2D}]$, where l_s^{2D} and l_{center} are the lengths of the line segment of s and the centerline, respectively. The attribute weights are simply set to $w_1^P = 0.2$ and $w_{j=1,2,3}^Q = 1$. P_{none} and Q_{none} are set as 0.4 and 0.2, respectively, just like the model for RGB-D images. These parameter settings are uniformly used for model mining of all the categories.

2.5.3 Experiment 3: mining general models from web images

Web images

In this experiment, I focus on a more common case of ubiquitous images, *i.e.* the images collected from the Internet by search engines. I use ten keywords—“bag”, “boot”, “camera”, “coca cola”, “glasses”, “hamster”, “iphone”, “panda”, “sailboat”, “spider”—to collect images of ten categories. I simply select the 200 top-ranked images for each category.

ARG-based model for general images

Generally speaking, the images collected from the Internet are fuzzier than those in my Kinect RGB-D images dataset [30, 29]. Therefore, I design a new type of ARGs for image representation, which take SIFT points, rather than edge segments, as graph nodes. These SIFT-based ARGs are oriented to more general images, especially to those not containing clear edges in them.

The SIFT points are detected all over the images at different scales. I connect each pair of points in an image to construct an ARG. Two unary attributes ($N_P = 2$) and five pairwise attributes ($N_Q = 5$) are designed to describe local features and spatial relationship between local parts in images. The notation is illustrated in Fig. 2.8. The two unary attributes are the 128-dimensional descriptor and the orientation of the SIFT feature, denoted by $\mathcal{F}_1^s = f_s$ and $\mathcal{F}_2^s = o_s$. The first of the five pairwise attributes, $\mathcal{F}_1^{st} = angle(o_s, o_t)$, denotes the spatial angle between the SIFT orientations of nodes s and t. For each edge (s, t), I use d_{st} and o_{st} to represent its length and orientation. I directly set the second pairwise attribute as $\mathcal{F}_2^{st} = o_{st}$. Then, the third pairwise attribute, $\mathcal{F}_3^{st} = [angle(o_{st}, o_s), angle(o_{st}, o_t)]^T$, is defined as the angle between edge (s, t) and each of SIFT orientations of nodes s and t. Let s_s and s_t denote the SIFT scales of nodes s and t. I set the fourth and fifth attributes as $\mathcal{F}_4^{st} = \log(s_s/s_t)$ and $\mathcal{F}_5^{st} = [\log(s_s/d_{st}), \log(s_t/d_{st})]^T$, respectively. I set the attribute

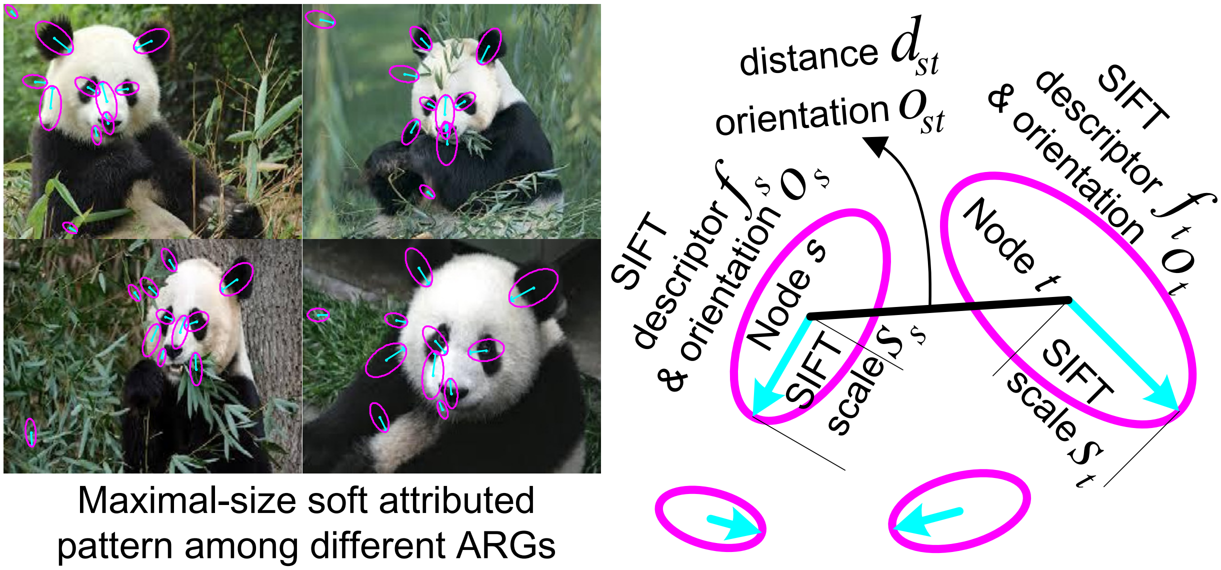

Maximal-size soft attributed pattern among different ARGs
distance d_{st}
orientation o_{st}
SIFT descriptor & orientation f_s o_s
Node s
SIFT scale S_s
SIFT descriptor & orientation f_t o_t
Node t
SIFT scale S_t

Figure 2.8: Notation for the ARGs that take the interesting points of SIFT features as graph nodes

weights as follows. $w_1^P = 3$, $w_2^P = 1$, $w_{j=1,2,4}^Q = 0.5$, and $w_{j=3,5}^Q = 1$. P_{none} and Q_{none} are set as 3 and 5, respectively. These parameter settings are uniformly used for model mining of all the categories.

Technical details

I set the threshold τ to be 2.5 in this experiment. Furthermore, considering the fuzzy cases of web images, I cannot ensure that each image contains an object in the target category. Therefore, in each iteration of model mining, not all the images can be used for model mining, and I set two criterions to control the quality of the images. The first criterion is that when I match the model to the image, the ratio of model nodes are matched to *none* should be greater than 40%. The second criterion is that in this iteration, the number of web images used for model mining should be no more than 20. I simply select 20 top-ranked images among all the collected web images, *i.e.* the 20 images whose ARGs can match the current model (graph template) with the lowest energy. All of these settings are uniformly applied to the mining of models for all the ten categories.

2.5.4 Quantitative analysis and evaluations

Competing methods

I compare the proposed method with totally seven approaches ranging across broad fields in graph matching and graph mining, although my graph-mining method is orthogonal to the learning concepts in the competing methods. All the competing method should follow the scenario of "learning graphical models (or target patterns) from large-size ARGs with a single labeled graph template", to enable a fair comparison. All the competing methods use the same initial graph templates and the same training and testing ARGs to start each learning process in cross validation.

Therefore, in the field of graph matching, I limit my comparison to unsupervised learning of graph matching, which does not require the labeling of matching assignments. From this view, six competing methods are applied. They mainly train matching parameters for the model to improve matching accuracy, and the learning of graphical structures is not involved. First, I take graph matching without training, denoted by *MA*, as the baseline. *MA* uses TRW-S [28] to match the initial graph template to the target objects in images. As the benchmark method in unsupervised learning for graph matching, [5], proposed by Leordeanu *et al.*, is also used for comparison. This iteratively trains the attribute weights for matching, *i.e.* w_i^P and w_j^Q in the matching penalties $P_s(x_s)$ and $Q_{st}(x_s, x_t)$. Different from my energy minimization in (2.1), their graph-matching assignments are computed based on another typical form, *i.e.* compatibility maximization $\text{argmax}_{\mathbf{x}}\, \mathcal{C}(\mathbf{x}) = \sum_{s,t} e^{-P_s(x_s)-P_t(x_t)-Q_{st}(x_s,x_t)}$, where $P_s(\cdot)$ and $Q_{st}(\cdot,\cdot)$ are defined using absolute differences. Thus, based on [5], the two competing approaches of *LS* and *LT* are obtained by applying spectral techniques [33] and TRW-S [28], respectively, to solve the matching optimization $\text{argmax}_{\mathbf{x}}\, \mathcal{C}(\mathbf{x})$. Note that the original version of [5] applies a uniform initialization for w_i^P and w_j^Q, but risks biased learning (which will be discussed later). To enable a fair comparison and ease the bias-learning problem, *LS* and *LT* are further modified to perform with the same weight initialization as my method[8], denoted by *LS-O* and *LT-O*. Finally, I further use the part of my method for iteratively estimating model attributes, according to Definition 2(a,b), as another competing framework, namely *SM*.

In addition, in the field of graph mining, the competing method should be formulated oriented to graph domain of ARGs without typical tricks, such as using the similarities between unary attributes to pre-determine a set of matching assignment candidates. Thus, I compare my approach to structural refinement [7], denoted by *SR*. This method locates on the boundary between graph mining and learning graph matching, as it only trains matching parameters and attributes, and deletes "bad" nodes to refine the model's structure, rather than a method to mine new nodes from ARGs. From this perspective, the mining of maximal-size iSAP proposed in this study is more close to the spirit of graph mining. Note that my method can mine iSAPs with different graph sizes by setting different thresholds τ. Therefore, to enable a fair comparison, *SR* is required to modify the initial graph template to a model containing the same number of nodes as the iSAP trained by my method, given a specific value of τ.

Evaluation metrics

Generally speaking, I can use the graph-size changes to illustrate the performance of my method using different settings of threshold τ.

I use the average matching rate (AMR) to evaluate the matching performance. This is widely used for the evaluation of learning graph matching [7, 5, 4]. The AMR is measured across all matching results produced by the extracted maximal SAP in the cross validation.

---

[8] As a tradeoff, I apply a raw setting for the weights of just one or two attributes to ease the bias learning problem.

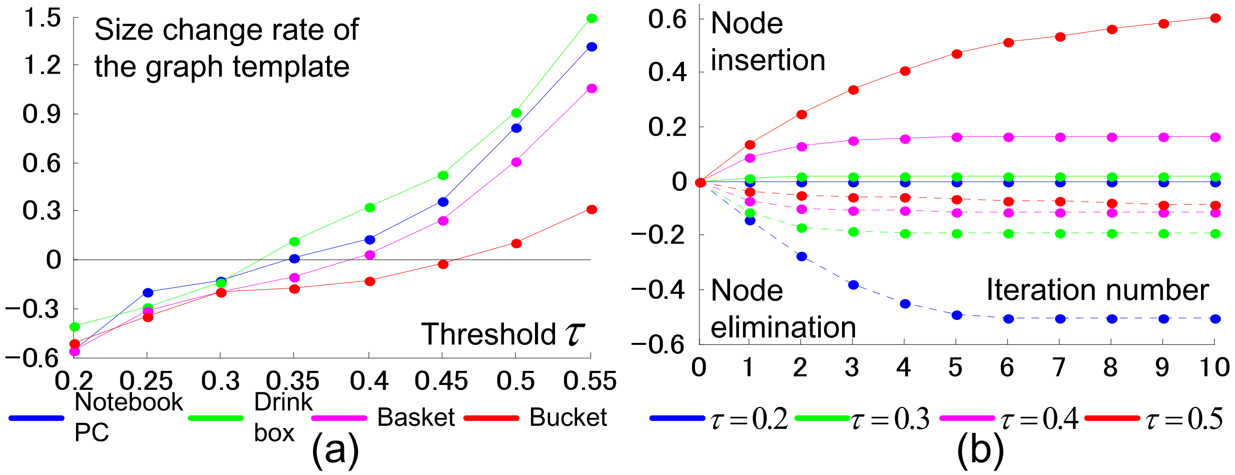


Figure 2.9: Size change rates of the graph template with different thresholds (τ) (a) and the rates of node insertion and elimination in different iterations (b). Vertical axes in both (a) and (b) indicate the percentage of the graph size changes, relative to the initial graph template. Solid and dotted lines in (b) indicate the effects of node insertion and elimination, respectively.

Results

Fig. 2.9 shows the size growth of the SAP with an increase of the threshold in Experiment 1. Fig. 2.10 illustrates the object detection performance of the maximal SAPs extracted using RGB-D images in Experiment 1. Table 2.1 lists the quantitative results for comparison, where the threshold τ is set to 0.25 for the learning of all the categories using RGB-D and RGB images in Experiments 1 and 2. Except *SR*, the competing methods do not have the ability to refine the topological structure of the graph template. Thus, they are sensitive to the bias in the initial graph template, including biased attributes, occluded nodes, and redundant nodes. The biased graph template may produce a biased matching, and this would, in turn, increase learning bias, thus propagating into a significant bias. In contrast, my method modifies the biased structure in early iterations to reduce the prevalence of biased matching in further iterations. Besides the elimination of "bad" parts as in *SR*, my approach also discover missing parts, thereby exhibiting better performance.

Category	From RGB-D images						
Method	NP	DB	BA	BU	SP	DU	BI
MA	69.00	75.25	57.97	75.33	72.65	83.96	77.78
LS	57.40	55.66	56.99	59.15	none	none	none
LS-O	64.63	74.18	60.75	77.99	76.48	84.53	87.61
LT	61.94	57.91	59.59	60.51	none	none	none
LT-O	69.04	75.09	65.37	80.44	77.31	85.47	89.33
SR	72.23	85.84	88.65	86.91	84.69	95.47	91.05
SM	89.05	85.97	75.61	84.95	90.85	95.31	94.82
Mine	**99.06**	**98.74**	**98.57**	**96.76**	**93.62**	**96.65**	**97.69**
Category	From RGB images						
Method	NP	DB	BA	BU	SP	DU	BI
MA	44.30	69.93	45.44	68.71	66.88	58.43	58.94
LS	46.40	49.46	50.25	54.74	none	none	none
LS-O	53.62	69.52	55.21	70.74	71.31	73.24	81.44
LT	48.83	51.24	50.97	56.39	none	none	none
LT-O	56.57	73.01	57.35	73.66	72.84	80.15	82.60
SR	60.31	79.38	79.59	85.92	91.76	93.43	84.15
SM	72.02	85.90	72.16	83.56	79.87	83.91	71.75
Mine	**72.91**	**96.18**	**91.49**	**91.18**	**94.45**	**99.01**	**88.99**

Table 2.1: Comparison of average matching rates in Experiments 1 and 2. *NP*, *DB*, *BA*, *BU*, *SP*, *DU*, and *BI* indicate the *notebook PC*, *drink box*, *basket*, *bucket*, *sprayer*, *dustpan*, and *bicycle* categories.

In addition, Fig. 2.11 shows the model-mining performance on web images collected from the Internet in Experiment 3.

2.6 Discussion and conclusions

In this chapter, I redefined the unsupervised learning of graph matching to model the discovery of missing parts, and thus idealize the spirit of structural learning.

The proposed method corrects errors in the topological structure of the initial graph template. As the threshold τ controls the fuzziness of the maximal SAP, it should be set up corresponding to the maximal graph (object) deformability in the ARGs. Given a suitable setting of τ, my method exhibits very good performance.

In terms of graph mining, this study can also be understood as the mining of maximal-size subgraph patterns. I proposed the SAP as the subgraph pattern of fuzzy ARGs, and demonstrated a plausible method of achieving the idea of mining the maximal-size subgraph pattern in the challenging graph domain of ARGs. I provided an approximate solution for maximal SAP extraction that does not require node enumeration. Another difference between conventional graph mining methods and my approach lies in the need for the graph template. This is because the matching between ARGs is formulated as a QAP, meaning that this graph matching can only be reliably achieved when an approximate area of interest for the subgraph pattern has been provided.

2.7 APPENDIX: Demonstration of Equation (2.8)

It is easy to prove

$$\begin{aligned}&\sum_{k=1}^{N}\|\mathcal{F}_i^{x_y^k}-\frac{1}{N}\sum_{l=1}^{N}\mathcal{F}_i^{x_y^l}\|^2\\=&\frac{1}{2N}\sum_{1\leq k,l\leq N}\|\mathcal{F}_i^{x_y^k}-\mathcal{F}_i^{x_y^l}\|^2\end{aligned}$$

and

$$\sum_{k:\delta(\hat{x}_t^k)=1}\|\mathcal{F}_i^{x_y^k\hat{x}_t^k}-\mathcal{F}_i^{yt}\|^2$$

$$=\frac{1}{2\sum_k \delta(\hat{x}_t^k)} \sum_{\substack{k,l:\\ \delta(\hat{x}_t^k)\delta(\hat{x}_t^l)=1}} \|\mathcal{F}_i^{x_y^k \hat{x}_t^k} - \mathcal{F}_i^{x_y^l \hat{x}_t^l}\|^2$$

Thus, I can obtain

$$\begin{aligned}
\mathbf{P}_y =& \frac{1}{N}\sum_{k=1}^{N} P_y(x_y^k | G^{new}, G_k')\\
=& \frac{1}{N}\sum_{k=1}^{N}\sum_{i=1}^{N_P} w_i^P \|\mathcal{F}_i^{x_y^k} - \mathcal{F}_i^{y}\|^2\\
=& \frac{1}{N}\sum_{i=1}^{N_P} w_i^P \sum_{k=1}^{N} \|\mathcal{F}_i^{x_y^k} - \frac{1}{N}\sum_{l=1}^{N} \mathcal{F}_i^{x_y^l}\|^2\\
=& \frac{1}{N}\sum_{i=1}^{N_P} w_i^P \frac{1}{2N} \sum_{1\le k,l\le N} \|\mathcal{F}_i^{x_y^k} - \mathcal{F}_i^{x_y^l}\|^2\\
=& \frac{1}{2N^2}\sum_{i=1}^{N_P} w_i^P \sum_{1\le k,l\le N} \|\mathcal{F}_i^{x_y^k} - \mathcal{F}_i^{x_y^l}\|^2
\end{aligned}$$

$$\begin{aligned}
\mathbf{Q}_{yt} =& \frac{1}{N}\sum_{k=1}^{N} Q_{yt}(x_y^k, \hat{x}_t^k | G^{new}, G_k')\\
=& \frac{1}{N}\sum_{i=1}^{N_Q} \Big[\frac{w_i^Q}{\|V\|} \sum_{k:\delta(\hat{x}_t^k)=1} \|\mathcal{F}_i^{x_y^k \hat{x}_t^k} - \mathcal{F}_i^{yt}\|^2 + \sum_{k:\delta(\hat{x}_t^k)=0} \frac{Q_{none}}{\|V\|} \Big]\\
=& \frac{1}{\|V\|N}\sum_{i=1}^{N_Q} w_i^Q \sum_{k:\delta(\hat{x}_t^k)=1} \|\mathcal{F}_i^{x_y^k \hat{x}_t^k} - \mathcal{F}_i^{yt}\|^2 + C_t\\
=& \frac{\sum_{i=1}^{N_Q} w_i^Q \sum_{k:\delta(\hat{x}_t^k)=1} \|\mathcal{F}_i^{x_y^k \hat{x}_t^k} - \frac{\sum_{l:\delta(\hat{x}_t^l)=1} \mathcal{F}_i^{x_y^l \hat{x}_t^l}}{\sum_{l'=1}^{N} \delta(\hat{x}_t^{l'})}\|^2}{\|V\|N} + C_t\\
=& \frac{\sum_{i=1}^{N_Q} w_i^Q \sum_{k,l:\delta(\hat{x}_t^k)\delta(\hat{x}_t^l)=1} \|\mathcal{F}_i^{x_y^k \hat{x}_t^k} - \mathcal{F}_i^{x_y^l \hat{x}_t^l}\|^2}{2\|V\|N\sum_k \delta(\hat{x}_t^k)} + C_t
\end{aligned}$$

where $C_t = \sum_{k:\delta(\hat{x}_t^k)=0} \frac{Q_{none}}{\|V\|N} = \frac{Q_{none}\sum_k (1-\delta(\hat{x}_t^k))}{\|V\|N}$ is a constant with respect to x_y^k, given $\{\hat{\mathbf{x}}^k\}$.

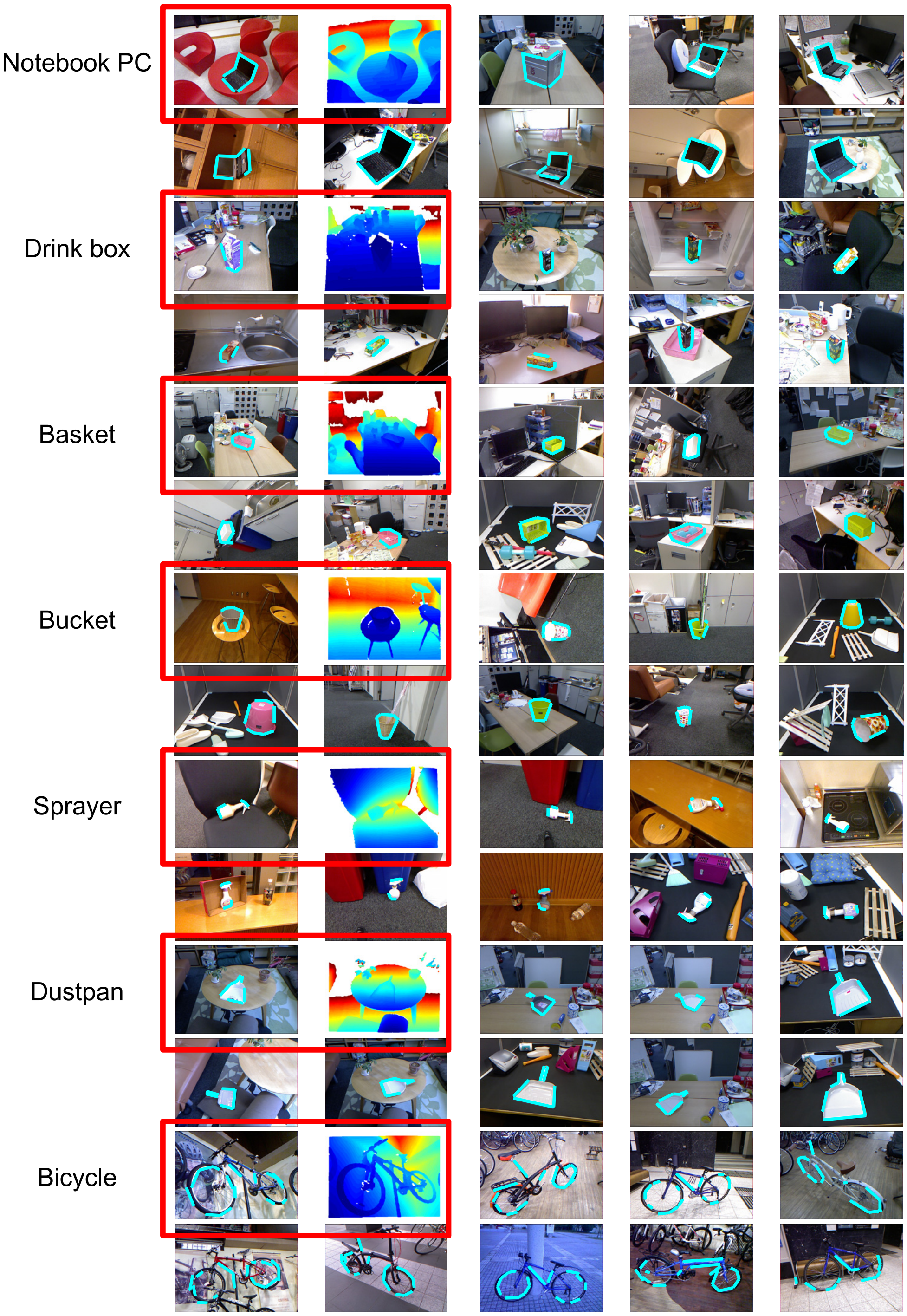


Figure 2.10: Object detection performance on the maximal SAPs trained from RGB-D images in Experiment 1. The second image in each category shows the depth images corresponding to the first image.

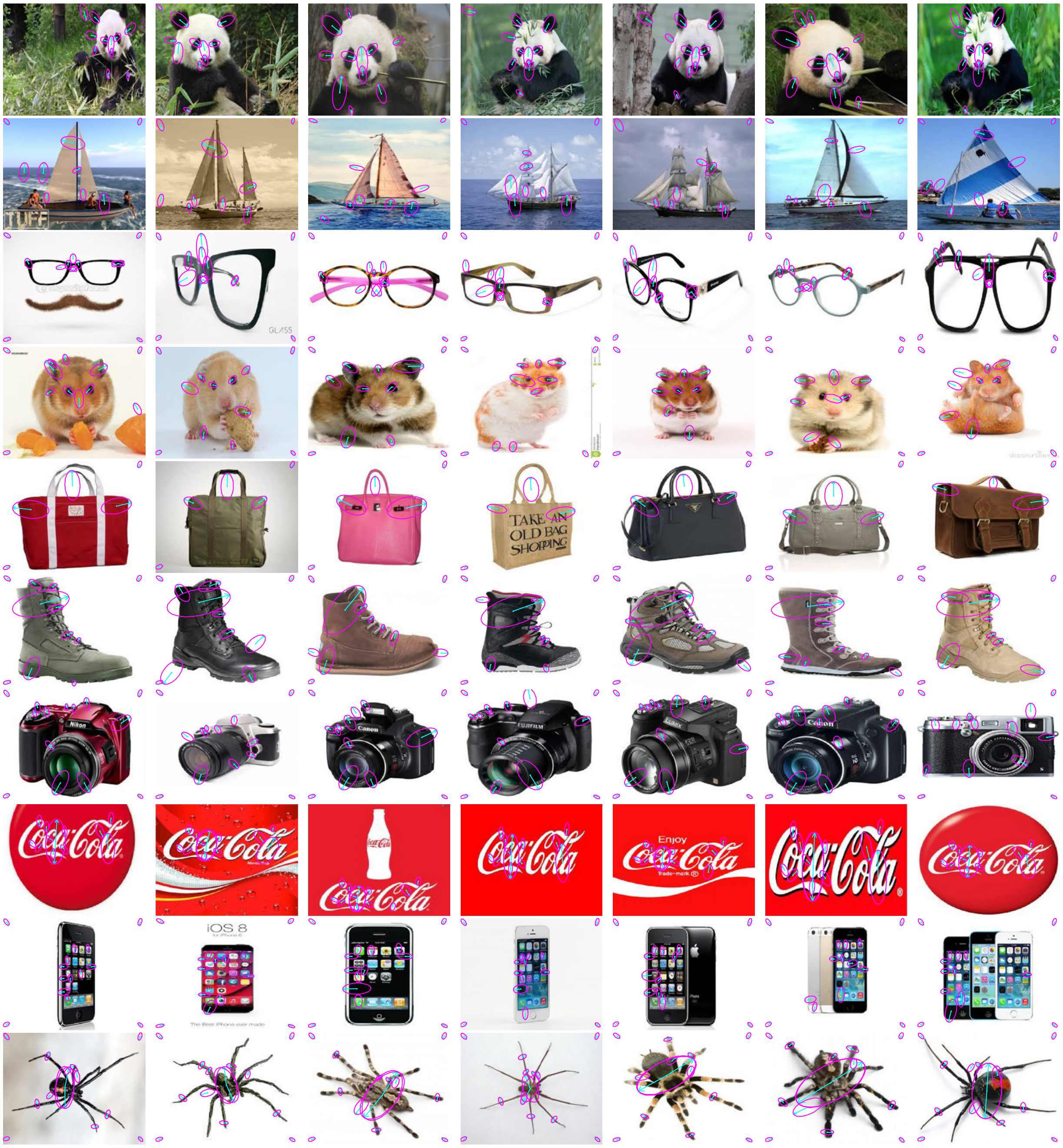


Figure 2.11: Category mining performance on web images in Experiment 3.

Chapter 3

Secondary Platform: Unsupervised Learning of Graph Matching

In the previous chapter, I have introduced the general platform for visual mining, which is based on attributed graph mining. In this chapter, I will present the secondary choice for visual mining. I extend the concept of unsupervised learning of graph matching, so as to make it applicable to the task of visual mining.

Graph matching is a fundamental problem in pattern recognition, and has drawn broad interest from many fields. However, there is still no other techniques bridging the idea of graph matching and the concept of visual mining. In addition, the literature of learning graph matching has not received much attention, although its superior performance has been demonstrated in pioneering studies.

Hence, in order to apply the concept of learning graph matching to the task of visual mining, I make the following two extensions. First, the learning process should be achieved in an unsupervised[9] fashion without labeling "what is where" in the ubiquitous images. Second, the prototype model, which represents the common pattern of a category, should be trained during the learning process. Third, the proposed method combines the refinement of graph structure as a part of learning graph matching, so as to estimate the best structure of the category model. In this way, the learning method is oriented toward both matching and recognition performance.

---

[9] In the context of learning graph matching, the term "unsupervised" [5] refers to the ability to learn the model without manually specifying each individual matching assignment within the graphs, rather than the labeling of positive and negative graphs.

By the way, as mentioned in the previous chapter, attributed graph mining can be understood as a type of unsupervised learning of graph matching. However, attributed graph mining can only be applied, when people use squared attribute difference to construct matching penalties. In contrast, the method proposed in this chapter formulates the problem of graph matching in a more general form, *i.e.* the maximization of matching compatibilities in (3.1). Therefore, theoretically, this method can be applied to more graph-matching cases.

The rest of this chapter is organized as follows. The introduction and discussion of related work are presented in Sections 3.1 and 3.2. Section 3.3 defines the general case of graph matching, and Section 3.4 presents the detailed algorithm. In Section 3.5, I introduce the design of experiments and evaluate the algorithm performance. Finally, the overall chapter is summarized in Section 3.6.

3.1 Introduction

Attributed graph matching is a fundamental problem ranging across broad fields in computer vision and data mining, and numerous approaches have been proposed for the problem of graph matching optimization [34]. Even so, the literature on learning graph matching remains limited, despite the demonstrated power of learning techniques in this area. The few pioneering studies of learning graph matching mainly aimed to train matching parameters, so as to obtain correct matching assignments for mapping from a graph template to a number of relatively large target graphs.

In this research, I approach the learning of graph matching from the perspective of category modeling. My aim is to incrementally modify the graph template to produce a graphical model representing the general structural knowledge of the targets objects in target graphs, and not merely train matching parameters. Therefore, this research is of great significance for object knowledge mining from cluttered scenes.

In so doing, I also aim to transform the conventional concept of *learning for graph*

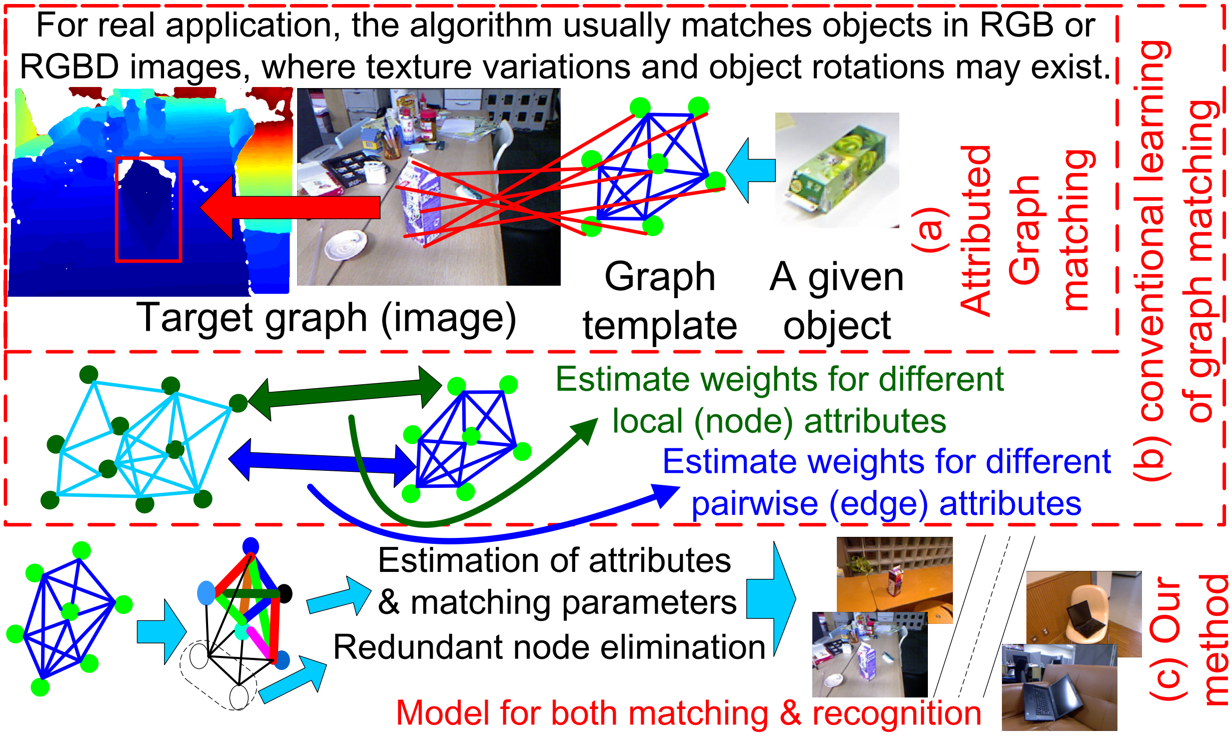

For real application, the algorithm usually matches objects in RGB or RGBD images, where texture variations and object rotations may exist.
Target graph (image)
Graph template
A given object
(a) Attributed Graph matching
(b) conventional learning of graph matching
Estimate weights for different local (node) attributes
Estimate weights for different pairwise (edge) attributes
Estimation of attributes & matching parameters
Redundant node elimination
Model for both matching & recognition
(c) Our method

Figure 3.1: Concept extension from pure attributed graph matching (a) to the proposed learning of graph matching (c). Different from conventional learning of graph matching (b), my method aims to modify the initial graph template into a graphical model, so as to achieve good performance in both matching and recognition. Thus, this research establishes a bridge between the learning of graph matching and the category modeling from cluttered scenes.

matching to *learning based on graph matching for both object matching and recognition*. Here, object recognition refers to determining whether a graph contains the target object, based on the trained model. I call the graphs that contain target objects *positive* graphs, and those that do not *negative* graphs. This idea for learning gives consideration to both matching performance and recognition performance.

The goal of this chapter can be described as follows. Given an initial graph template and a number of positive and negative graphs for training, I aim to 1) learn matching parameters in an unsupervised[9] fashion, 2) incrementally refine the local and pairwise attributes of the graph template, and simultaneously 3) modify the structure of the graph template by eliminating incorrect or redundant parts, thus generating a model that achieves good performance in both matching and recognition.

To perform estimates of both matching parameters and model attributes, my work extends the unsupervised^9 learning of graph matching proposed by Leordeanu *et al.* [5]. Meanwhile, my approach to structural modification of the graph template uses a novel technique based on the mechanism of object recognition.

The contributions of this chapter can be summarized as follows. I redefine the learning of graph matching as a category modeling problem that is oriented toward not only conventional matching performance, but also object recognition. It includes the estimation of matching parameters, attributes, and graphical structure. In particular, this is the first attempt to use negative graphs in the learning of graph matching, to the best of my knowledge.

A preliminary version of this chapter appeared in [7].

3.2 Related work

Most conventional algorithms for the learning of graph matching are supervised^9 methods that require detailed labeling of each template node's matching assignment in each positive graph for training. [3], [6], and [4] used large-margin methods [35], non-linear inverse optimization [36], and smoothing-based techniques to train matching parameters in a supervised fashion, respectively. Compared to supervised methods, the unsupervised^9 method [5] does not require a large amount of node-level labeling. Thus, it is closer to my approach and resembles, at least philosophically, my idea of category knowledge mining from cluttered scenes. Indeed, I can derive another type of supervised learning from [5] as a compromise, greatly reducing manual interactions by applying object-level labeling, and thereby supporting fairer comparison.

All studies mentioned above are focused on training matching parameters for good matching performance. By contrast, my approach emphasizes ont only the matching rate, but also the recognition performance of the trained, matching-based model. In addition to training the matching parameters, I modify the model (graph template) structure and estimate model attributes.

Then, I give an in-depth discussion on the structural modification, which is the main part of the learning process. To be exact, if I simplify the problem by only considering the matching between two graphs, the structural modification is related to the progressive graph matching [8] proposed by Cho *et al.*, which made a great contribution to the selection of reliable nodes and edges for a more efficient matching. If I do not limit my discussion within the range of general-form graph matching, this problem is also related to category modeling for recognition [20], the model training for the Hough-style matching [37], and common object extraction from two images based on maximal clique mining [13, 14], such as [18, 19]. Most methods for object extraction from multiple images [23, 24, 25, 26, 27] related to clique mining uses a combination of two-image matching results. [22] extracted object models from images based on page-rank mechanisms. [1, 11, 16, 17] aim to learn the maximal frequent subgraph among several graphs with distinct node or edge labels. Above all, most of these methods mentioned above limit their interests to the geometric consistency and similarity of local patches. As thus, they usually need additional data constraints. *E.g.* [20, 37] require there is no roll rotations for matching, and object extraction methods usually require the local features in images to be distinguishing enough to determine a set of potential image matching assignments during the preprocessing.

In contrast, by emphasizing a general algorithm for learning graph matching, my approach remains free of such constraints. I formulate the learning problem strictly under a common paradigm of graph matching based on various local and pairwise attributes, and so are able to apply my method to cluttered scenes containing target objects with different scales, textures, and rotations simply by designing a set of suitable attributes. Nevertheless, I still compare my recognition-oriented structural modification strategy with strategies of the related studies in experiments.

3.3 Preliminary: graph matching problem

The objective of graph matching is to find correspondences between a graph template (the category model) $G=(V,E,F_V,F_E)$ and an attributed graph $G'=(V',E',F_{V'},F_{E'})$. V and E denote the node set and the edge set of G, respectively. F_V and F_E denote the attribute sets for local and pairwise attributes. Let G have n_v nodes, $V=\{1,2,...,n_v\}$, and G' have $n_{v'}$ nodes, $V'=\{1,2,...,n_{v'}\}$. Each node $i\in V$ of G has n^U unary attributes ($f_i^{(k)}\in F_V, k=1,2,...,n^U$) and each edge $(i,j)\in E$ has n^P pairwise attributes ($f_{ij}^{(l)}\in F_E, l=1,2,...,n^P$). The matching assignments between G and G' are represented by a binary matching matrix $\mathbf{Y}\in\{0,1\}^{n_v\times n_{v'}}$. If node $i\in V$ matches node $i'\in V'$, then $Y_{i,i'}=1$, otherwise $Y_{i,i'}=0$. In fact, I use a column-wise vectorized replica of $\mathbf{Y}$, denoted by $\mathbf{y}\in\{0,1\}^{n_v n_{v'}}$. $y_{ii'}$ in $\mathbf{y}$ corresponds to $Y_{i,i'}$ in $\mathbf{Y}$. Thus, I obtain a typical form [33, 8, 5] of attributed graph matching as follows:

$$\begin{aligned} &\hat{\mathbf{y}}=\underset{\mathbf{y}}{\operatorname{argmax}}\,\mathcal{C}(\mathbf{y}|G,G'),\quad \mathcal{C}(\mathbf{y}|G,G')=\mathbf{y}^T\mathbf{M}\mathbf{y}\\ &\text{s.t.}\quad \forall i\in V,\sum_{i'\in V'}y_{ii'}\le 1,\quad \forall i'\in V',\sum_{i\in V}y_{ii'}\le 1 \end{aligned} \tag{3.1}$$

This is a quadratic assignment problem, where $\mathcal{C}(\mathbf{x}|G,G')$ is the function measuring the matching compatibility between G and G'. $\mathbf{M}$ is a $(n_v n_{v'})$-by-$(n_v n_{v'})$ compatibility matrix containing non-negative elements. In most cases [33, 8], the matching compatibility $M_{ii',jj'}$ can be represented as a function of attribute distances as follows.

$$\begin{aligned} &M_{ii',jj'}=\begin{cases}\Phi^P(\mathbf{d}_{ii',jj'}|\mathbf{w}^P), & (i,j)\in E,(i',j')\in E'\\ 0, & \text{Otherwise}\end{cases}\\ &M_{ii',ii'}=\Phi^U(\mathbf{d}_{ii'}|\mathbf{w}^U),\quad i\in V,i'\in V' \end{aligned} \tag{3.2}$$

where $\Phi^U(\mathbf{d}_{ii'}|\mathbf{w}^U)$ is set on the diagonal of $\mathbf{M}$ and measures the unary compatibility for a node pair of $i\in V$ and $i'\in V'$; the non-diagonal element of $\mathbf{M}$, $\Phi^P(\mathbf{d}_{ii',jj'}|\mathbf{w}^P)$, measures the pairwise compatibility for an edge pair of $(i,j)\in E$ and $(i',j')\in E'$.

I define $\mathbf{d}_{ii'}=[d_{ii'}^{(1)},d_{ii'}^{(2)},...,d_{ii'}^{(n^U)}]^T$ as the Euclidean distances of the unary attributes, $d_{ii'}^{(k)}=\|f_i^{(k)}-f_{i'}^{(k)}\|$. Whereas $\mathbf{d}_{ii',jj'}=[d_{ii',jj'}^{(1)},d_{ii',jj'}^{(2)},...,d_{ii',jj'}^{(n^P)}]^T$ denote the Euclidean distances of the pairwise attributes, $d^{(l)}_{ii',jj'} = \|f^{(l)}_{ij} - f^{(l)}_{i'j'}\|$. $\mathbf{w}^U = [w^U_1, w^U_2, ..., w^U_{n^U}]^T$ and $\mathbf{w}^P = [w^P_1, w^P_2, ..., w^P_{n^P}]^T$ denote the weights for each unary and pairwise attribute, respectively.

As in [5], without loss of generality, I transform (3.2) to absorb the unary compatibilities into the pairwise compatibilities and leave zeros on the diagonal, achieving better performance.

$$M_{ii',jj'} = \begin{cases} \Phi(\mathbf{d}_{ii'}, \mathbf{d}_{jj'}, \mathbf{d}_{ii',jj'} | \mathbf{w}^U, \mathbf{w}^P), & (i,j) \in E, (i',j') \in E' \\ 0, & \text{Otherwise} \end{cases} \tag{3.3}$$

Note that when the structure of the graph template G is not well segmented and needs further modification, it is meaningful to bring in a dummy choice—*none*—for the matching assignments of nodes in G. Without an accurate structure, G may have some redundant nodes that should be matched to *none*. Thus, I re-write (3.1) and (3.3) as:

$$\begin{aligned} \hat{\mathbf{x}} &= \underset{\mathbf{x}}{\operatorname{argmax}}\, \mathcal{C}'(\mathbf{x}|G, G') \\ \mathcal{C}'(\mathbf{x}|G, G') &= \sum_{i,j \in V \cup \{none\}} c_{ij}(x_i, x_j | G, G') \\ c_{ij}(x_i, x_j | G, G') &= \begin{cases} M_{ix_i, jx_j}, & x_i \neq x_j \in V' \\ -\infty, & x_i = x_j \in V' \\ \frac{\lambda(\mathbf{1}^T \mathbf{M} \mathbf{1})}{n_v^2 n_{v'}^2}, & x_i \text{ or } x_j = none \end{cases} \end{aligned} \tag{3.4}$$

where x_i indicates the matching assignment of node $i \in V$, and $x_i = i' \in V'$ if and only if $y_{ii'} = 1$. λ is the parameter weighting for the matching compatibility of *none*. The setting of *none* reduces incorrect matching and eases the bias learning problem that commonly afflicts the unsupervised learning of graph matching (which will be explained later).

The maximization of the compatibility function $\mathcal{C}'(\mathbf{x}|G, G')$ can be achieved using various graph matching optimization techniques, and I choose TRW-S [28] here.

3.4 Learning of graph matching

Given an initial graph template $G = (V, E, F_V, F_E)$, a set of N^+ positive graphs $PG = \{G^+_k | k = 1, 2, ..., N^+\}$, $G^+_k = (V^+_k, E^+_k, F_{V^+_k}, F_{E^+_k})$, and a set of N^- negative graphs, $NG =$

$\{G_l^-|l=1,2,...,N^-\}$, $G_l^-=(V_l^-,E_l^-,F_{V_l^-},F_{E_l^-})$, the objective for learning graph matching is to estimate an induced subgraph of G, $\tilde{G}=(\tilde{V},\tilde{E},F_{\tilde{V}},F_{\tilde{E}})$, $\tilde{V}\subseteq V$, $\tilde{E}\subseteq E$, as the category model, simultaneously training matching parameters $\{\mathbf{w}^U,\mathbf{w}^P\}$ and modifying model attributes $\{F_{\tilde{V}},F_{\tilde{E}}\}$, so as to achieve good matching performance in positive graphs in PG and high recognition accuracy among graphs in PG and NG.

3.4.1 Parameter and attribute estimation

For the matching between the current G and G_k^+ based on (3.1), let the principal eigenvector of $\mathbf{M}$ be $\mathbf{a}^k$. According to [33], its element $a_{ii'}^k$ can be taken as the confidence value of the corresponding assignment between node $i\in V$ and node $i'\in V_k^+$. Leordeanu *et al.* [5] proposed to increase the elements corresponding to the correct assignments. Meanwhile, as $\|\mathbf{a}^k\|$ is normalized to 1, the elements for incorrect assignments will decrease, thus achieving greater reliability in matching.

To reduce the large computation, an approximate principal eigenvector is calculated as $\mathbf{a}^k=\frac{\mathbf{M^n1}}{\sqrt{(\mathbf{M^n1})^T(\mathbf{M^n1})}}$. Thus, the partial derivative of $\mathbf{a}^k$ is computed as follows:

$$(\mathbf{a}^k)'=\frac{(\mathbf{M^n1})'\|\mathbf{M^n1}\|-((\mathbf{M^n1})^T(\mathbf{M^n1})')\mathbf{M^n1}/\|\mathbf{M^n1}\|}{\|\mathbf{M^n1}\|^2}$$
$$(\mathbf{M^n1})'=\mathbf{M}'(\mathbf{M^{n-1}1})+\mathbf{M}(\mathbf{M^{n-1}1})' \tag{3.5}$$

Here, I choose $n=10$, as in [5].

The benchmark method for unsupervised learning of graph matching [5] proposed by Leordeanu *et al.* focuses exclusively on learning matching parameters. I extend this method to include the learning of model attributes $\{\mathbf{w}^U,\mathbf{w}^P,F_V,F_E\}$, which is similar to [29]. The objective is to maximize the following function:

$$\mathcal{G}(\mathbf{w}^U,\mathbf{w}^P,F_V,F_E)=\sum_{k=1}^{N^+}\sum_{\substack{i\in V\\ \hat{x}_i^k\neq none}}a_{i\hat{x}_i^k}^k(\mathbf{w}^U,\mathbf{w}^P,F_V,F_E) \tag{3.6}$$

where $\hat{\mathbf{x}}^k=\{\hat{x}_i^k\in V_k^+|i\in V\}$ is the predicted assignments between the current G and G_k^+ based on (3.4).

Algorithm 2 Learning of graph matching

```
Input: An initial graph template $G^*$; a set of $N^+$ positive graphs, $PG$; a set of $N^-$ negative graphs, $NG$; the iteration number $T$ for the estimation of matching parameters and attributes; a threshold $\tau$.
Output: The category model $\tilde{G}$.
Set initial leave-one-out (LOO) classification accuracy as $\tilde{A} = 1$ and node reliability of $G$ as $\forall i \in V, \tilde{R}_i = -\infty$.
repeat
    1. Initialize the category model $G = G^*$ and the weights for unary and pairwise attributes $\mathbf{w}^U = \mathbf{1}_{n^U \times 1}$, $\mathbf{w}^P = \mathbf{1}_{n^P \times 1}$.
    for $iteration = 1$ to $T$ do
        2.1. Use the current $G$ to predict the matching assignments to $G_k^+$, $\hat{\mathbf{X}}$, based on (3.4).
        2.2. With $\hat{\mathbf{X}}$, update the matching parameters $\mathbf{w}^U$,$\mathbf{w}^P$ and attributes $F_V$,$F_E$ of $G$, based on (3.7).
    end for
    3. Match the current $G$ to graphs in $PG$ and $NG$ based on (3.4)[10] and obtain $\hat{\mathbf{X}}$ and $\mathring{\mathbf{X}}$.
    4. Given $\hat{\mathbf{X}}$ and $\mathring{\mathbf{X}}$, train the classifier with the normal vector $\mathcal{W}$ and a new LOO accuracy $A$, based on (3.10).
    5. If $A < \tilde{A}$ and $\min_i \tilde{R}_i > \tau$ then break.
    6. Given $\mathcal{W}$, update node reliability $\tilde{R}_i$, based on (3.11). Update $\tilde{G} = G$ and $\tilde{A} = A$.
    7. Eliminate node $i^* = \mathrm{argmin}_{i \in V} R_i$ from $G^*$ and set it as the induced subgraph $G^* \leftarrow G^*(V \backslash \{i^*\})$.
until $n_v = 2$
```

As shown in Algorithm 2, I can achieve the maximization of $\mathcal{G}(\mathbf{w}^U,\mathbf{w}^P,F_V,F_E)$ iteratively. In each iteration, I use the current G to predict the matching assignments $\hat{\mathbf{x}}^k$, $k = 1, 2..., N^+$, and then modify the matching parameters $\mathbf{w}^U$,$\mathbf{w}^P$ and attributes F_V,F_E via gradient ascent:

$$\begin{aligned}
w_k^U &\leftarrow w_k^U + \zeta \textstyle\sum_{k'=1}^{N^+} \sum_{\substack{i'\in V\\ \hat{x}_{i'}^{k'}\neq none}} \frac{\partial a^{k'}_{i'\hat{x}_{i'}^{k'}}(\mathbf{w}^U,\mathbf{w}^P,F_V,F_E)}{\partial w_k^U} \\
w_l^P &\leftarrow w_l^P + \zeta \textstyle\sum_{k'=1}^{N^+} \sum_{\substack{i'\in V\\ \hat{x}_{i'}^{k'}\neq none}} \frac{\partial a^{k'}_{i'\hat{x}_{i'}^{k'}}(\mathbf{w}^U,\mathbf{w}^P,F_V,F_E)}{\partial w_l^P} \\
f_i^{(k)} &\leftarrow f_i^{(k)} + \zeta \textstyle\sum_{k'=1}^{N^+} \sum_{\substack{i'\in V\\ \hat{x}_{i'}^{k'}\neq none}} \frac{\partial a^{k'}_{i'\hat{x}_{i'}^{k'}}(\mathbf{w}^U,\mathbf{w}^P,F_V,F_E)}{\partial f_i^{(k)}} \\
f_{ij}^{(l)} &\leftarrow f_{ij}^{(l)} + \zeta \textstyle\sum_{k'=1}^{N^+} \sum_{\substack{i'\in V\\ \hat{x}_{i'}^{k'}\neq none}} \frac{\partial a^{k'}_{i'\hat{x}_{i'}^{k'}}(\mathbf{w}^U,\mathbf{w}^P,F_V,F_E)}{\partial f_{ij}^{(l)}}
\end{aligned} \tag{3.7}$$

3.4.2 Structural modification

In this subsection, I use the matching results of the current G to train a classifier for object recognition. In order to train the model for good recognition performance, I propose a new method that uses the parameters of the classifier to guide the structural modification of G.

There are a variety of approaches to classification based on graph matching [38, 39], and I will obtain different classification performances by applying different classifiers to different feature. In order to achieve a natural connection between the structural modification of G and the classification mechanism, I select the linear-SVM classifier and attribute distances as my target features.

Feature extraction: Let $\hat{\mathbf{X}}=\{\hat{\mathbf{x}}^k|k=1,2...,N^+\}$ and $\mathring{\mathbf{X}}=\{\mathring{\mathbf{x}}^l|l=1,2...,N^-\}$ denote a set of predicted assignments matching to positive graphs $G_k^+\in PG$ and a set of predicted assignments matching to negative graphs $G_l^-\in NG$, based on graph matching[10]. Thus, according to (3.3), $\mathbf{d}_{i\hat{x}_i^k}$ indicates the distance of the unary attributes for matching node $i\in V$ to node $\hat{x}_i^k$ in positive graph G_k^+. $\mathbf{d}_{i\mathring{x}_i^l}$ is for the matching to negative graph G_l^-. Similarly, $\mathbf{d}_{i\hat{x}_i^k,j\hat{x}_j^k}$ and $\mathbf{d}_{i\mathring{x}_i^l,j\mathring{x}_j^l}$ are for pairwise attribute distances.

Features for object recognition are generated from these attribute distances. I define

---

[10] Note that graph matching based on (3.4) is applied by setting $\lambda=-\infty$ in (3.4) to avoid $\hat{x}_i^k$ or $\mathring{x}_i^l=none$.

the feature vector to recognize the matching between G and G_k^+ as follows:

$$\begin{aligned} \hat{\mathcal{F}}^k &= [\hat{\mathbf{u}}_1^k, \hat{\mathbf{p}}_1^k, \hat{\mathbf{u}}_2^k, \hat{\mathbf{p}}_2^k, ..., \hat{\mathbf{u}}_{n_v}^k, \hat{\mathbf{p}}_{n_v}^k]^T \\ \hat{\mathbf{u}}_i^k = \mathbf{d}_{i\hat{x}_i^k}^T, &\quad \hat{\mathbf{p}}_i^k = \sum\nolimits_{j:j\neq i} \mathbf{d}_{i\hat{x}_i^k, j\hat{x}_j^k}^T \Big/ \sum\nolimits_{\substack{j:(i,j)\in E \\ (\hat{x}_i^k, \hat{x}_j^k)\in E_k^+}} 1 \end{aligned} \tag{3.8}$$

For the matching between nodes $i \in V$ and $\hat{x}_j^k \in V_k^+$, $\hat{\mathbf{u}}_i^k$ and $\hat{\mathbf{p}}_i^k$, $(i = 1, 2, ..., n_v \in V)$, are two n^U-dimension and n^P-dimension vectors for node $i \in V$, indicating the distance of the n^U unary attributes and the marginal penalty for the distance of the pairwise attributes, respectively.

Similarly, the feature vector for the matching between G and G_l^- is represented as

$$\mathring{\mathcal{F}}^l = [\mathring{\mathbf{u}}_1^l, \mathring{\mathbf{p}}_1^l, \mathring{\mathbf{u}}_2^l, \mathring{\mathbf{p}}_2^l, ..., \mathring{\mathbf{u}}_{n_v}^l, \mathring{\mathbf{p}}_{n_v}^l]^T \tag{3.9}$$

Both $\hat{\mathcal{F}}^k$ and $\mathring{\mathcal{F}}^l$ are vectors with $n_v(n^U + n^P)$ dimensions.

Classification for object recognition: I train a linear-SVM classifier for object recognition as follows.

$$\begin{gathered} \min_{\mathcal{W},\xi,b} \Big\{ \frac{1}{2}\|\mathcal{W}\|^2 + C\sum_{k=1}^{N^+ + N^-} \xi_k \Big\}, \\ \text{s.t.} \quad \forall k = 1, 2, ..., N^+, \mathcal{W}\cdot\hat{\mathcal{F}}^k - b \geq 1 - \xi_k,\ \xi_k \geq 0; \\ \forall k = 1, 2, ..., N^-, -(\mathcal{W}\cdot\mathring{\mathcal{F}}^k - b) \geq 1 - \xi_{k+N^+},\ \xi_{k+N^+} \geq 0 \end{gathered} \tag{3.10}$$

where $\mathcal{W} = [\boldsymbol{\mu}_1, \boldsymbol{\rho}_1, \boldsymbol{\mu}_1, \boldsymbol{\rho}_1, ..., \boldsymbol{\mu}_{n_v}, \boldsymbol{\rho}_{n_v}]^T$ represent the normal vector to the hyperplane. $\boldsymbol{\mu}_i$ is a n^U-dimension vector and corresponds to the weights for the n^U unary attribute distances in $\hat{\mathbf{u}}_i^k$ and $\mathring{\mathbf{u}}_i^l$. $\boldsymbol{\rho}_i$ is a n^P-dimension vector for pairwise attribute distances in $\hat{\mathbf{p}}_i^k$ and $\mathring{\mathbf{p}}_i^l$.

Classifier-guided structural modification: Here, I combine graph matching and the SVM-based classification to identify reliable and unreliable nodes in G, as follows. Clearly, G should be better matched to positive graphs $G_k^+ \in PG$ than negative ones $G_l^- \in NG$. In other words, attribute distances for matching to positive graphs (*i.e.* $\hat{\mathbf{u}}_i^k$ and $\hat{\mathbf{p}}_i^k$) should be less than those for matching to negative graphs ($\mathring{\mathbf{u}}_i^l$ and $\mathring{\mathbf{p}}_i^l$), when node

i is a reliable node in G. Consequently, the weights of node i (*i.e.* $\boldsymbol{\mu}_i$ and $\boldsymbol{\rho}_i$) should be negative, according to (3.10).

Therefore, I use the following metric to evaluate the reliability of node $i \in V$:

$$R_i = -\frac{\sqrt{n_v}}{\|\mathcal{W}\|}\Big[\sum_{j=1}^{n^U}\mu_i^{(j)} + \sum_{j=1}^{n^P}\rho_i^{(j)}\Big] \tag{3.11}$$

where $\mu_i^{(j)}, \rho_i^{(j)}$ are the j-th elements of $\boldsymbol{\mu}_i, \boldsymbol{\rho}_i$. R_i is normalized by $\frac{\sqrt{n_v}}{\|\mathcal{W}\|}$ to make it invariable to size changes of G.

I perform structural modification iteratively. In each iteration, after the estimation of matching parameters and attributes (see Section 3.4.1), I eliminate the node with the lowest reliability $i^* = \mathrm{argmin}_{i\in V} R_i$ from the template, and use this induced subgraph of the template to retrain a new model, thereby replacing current model. Meanwhile, I calculate the classification accuracy in a leave-one-out (LOO) cross validation for each of the models. The stopping condition is that R_{i^*} is greater than a threshold τ and the elimination of node i^* would decrease the LOO accuracy. Please see Algorithm 2 for details.

3.5 Experiments

The proposed method is especially useful in the field of computer vision, enabling the discovery of general object structures for image matching when the target objects are randomly placed in cluttered scenes. To evaluate my method in this regard, I designed two category modeling experiments, one using ordinary RGB images and the other using RGB-D images captured by a Kinect device [40].

As introduced in Chapter 7, I used the category dataset of Kinect RGB-D images, published in [29] as a standard RGB-D object dataset oriented to graph matching[11]. Four largest categories—*notebook PC*, *drink box*, *basket*, and *bucket*—in this dataset contained enough RGB-D objects and were chosen to construct both the positive and the negative

---

[11] This is one of the largest RGB-D object datasets, and fits the requirements of graph matching well. *http://sites.google.com/site/quanshizhang*

graph sets for training. These images depicted cluttered scenes containing target objects with different textures and rotations, and the both experiments were performed on these scenes.

I compared the proposed method with other approaches to learning graph matching and various common strategies in object extraction[12].

3.5.1 Category modeling from RGB & RGB-D images

In cluttered scenes, objects in the same category usually contain a variety of textures, and may be positioned at various rotations. Considering the need for robustness with respect to texture variations, I applied two graphical models proposed in Sections 2.5.2 and 2.5.1, each of which uses [31] to extract object edges and then discretizes continuous edges into line segments to produce the graph nodes. The two models use different attributes to represent objects in RGB and RGB-D images, respectively. The model for RGB images contains one local attribute and three pairwise attributes, while the model for RGB-D images contains two local attributes and three pairwise attributes. Please see Sections 2.5.2 and 2.5.1 for detailed settings of the attributes.

3.5.2 Experiments and quantitative evaluations

For both two experiments, I used the following compatibility function, corresponding to (3.3).

$$\begin{aligned}&\Phi(\mathbf{d}_{ii'},\mathbf{d}_{jj'},\mathbf{d}_{ii',jj'}|\mathbf{w}^U,\mathbf{w}^P)\\ =&\exp\left(-(\mathbf{w}^U)^{\mathbf{T}}\mathbf{d}_{ii'}^2-(\mathbf{w}^U)^{\mathbf{T}}\mathbf{d}_{jj'}^2-(\mathbf{w}^P)^{\mathbf{T}}\mathbf{d}_{ii',jj'}^2\right)\end{aligned}\tag{3.12}$$

There is, in fact, a fair variety of compatibility functions (*e.g.* $\exp(-\mathbf{w}^{\mathbf{T}}\mathbf{d})$ and $\alpha/(\beta+\mathbf{w}^{\mathbf{T}}\mathbf{d})$). Note that my proposed method is not limited to the particular compatibility function used in these experiments. Any compatibility function that can be cast into the form of (3.2) or (3.3) will do.

---

[12] Please see Section "Related work" for more discussion.

I set the iteration number as $T = 5$ and the parameter for matching to *none* as $\lambda = 5$. The iteration number T is a general setting for [5] and is applied to all competing techniques of [5], so as to ensure fair comparison.

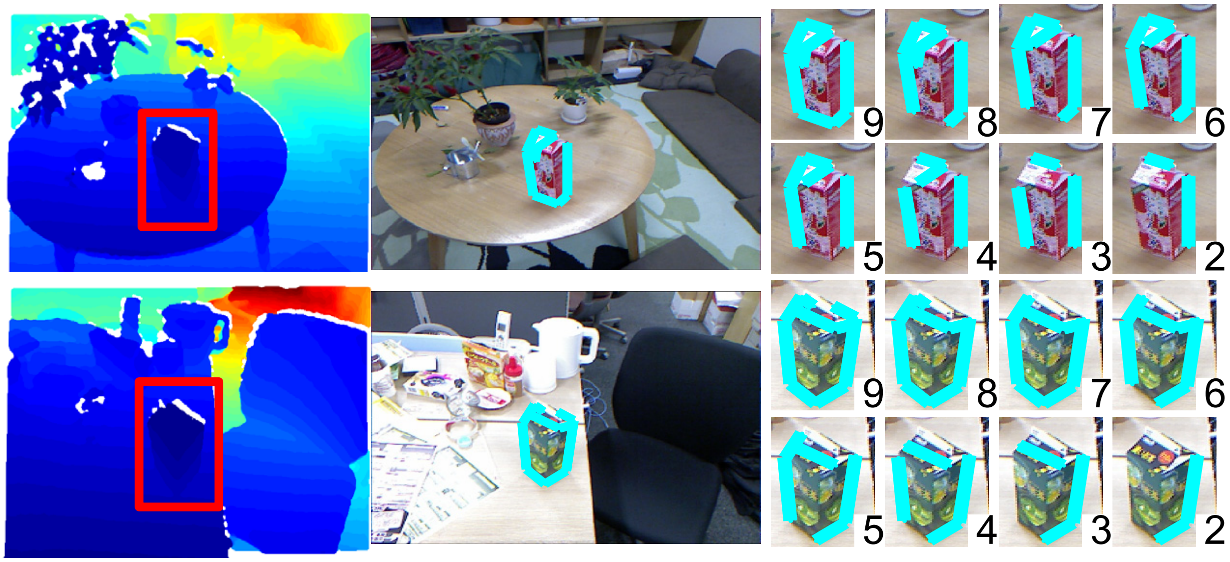


Figure 3.2: Process of node elimination. The bottom-right number indicates the model node number.

Cross validation and evaluation metrics: Each labeling of the target object in a given RGB or RGB-D image can produce an initial graph template and begin an individual model learning process. I labeled the images for a given category in sequence to begin multiple learning processes. In each of these processes, the remaining (*i.e.* unlabeled) images of this category were used to generate positive graphs. I then randomly selected the same number of images from other categories to generate negative graphs. I used 2/3 and 1/3 of these graphs for training and testing in this learning process, respectively. The end result is a set of models for the evaluation of the category.

I used the average matching rate (AMR) to evaluate the matching performance in positive graphs. AMR is widely used to evaluate the learning of graph matching [5, 4]. The matching rate of each individual matching result indicates the proportion of model nodes that are correctly matched to the target object. AMR represents the average of individual matching rates across all matching results produced by the trained models. Similarly, the average recognition accuracy (ARA) (*i.e.* the average value for recognition accuracy in the cross validation) was used to evaluate model-based recognition performance among both positive and negative graphs.

	Category	MA	LS	LT	WM	SU	Mine^{avg}
Matching	Notebook PC	**56.05**	46.40	48.83	49.93	52.90	50.74
	Drink box	56.15	49.46	51.24	58.55	52.70	**89.11**
	Basket	55.28	50.25	50.97	59.60	51.68	**81.33**
	Bucket	58.02	54.74	56.39	61.76	56.80	**86.85**
Recognition	Notebook PC	67.63	67.36	66.60	**77.07**	67.98	75.21
	Drink box	72.74	69.73	68.75	82.47	72.05	**89.53**
	Basket	63.95	67.13	67.01	74.94	66.32	**76.39**
	Bucket	80.90	78.96	77.83	86.12	78.54	**89.07**

Table 3.1: Comparison of average matching rates and average recognition accuracy in the experiment of category modeling from RGB images.

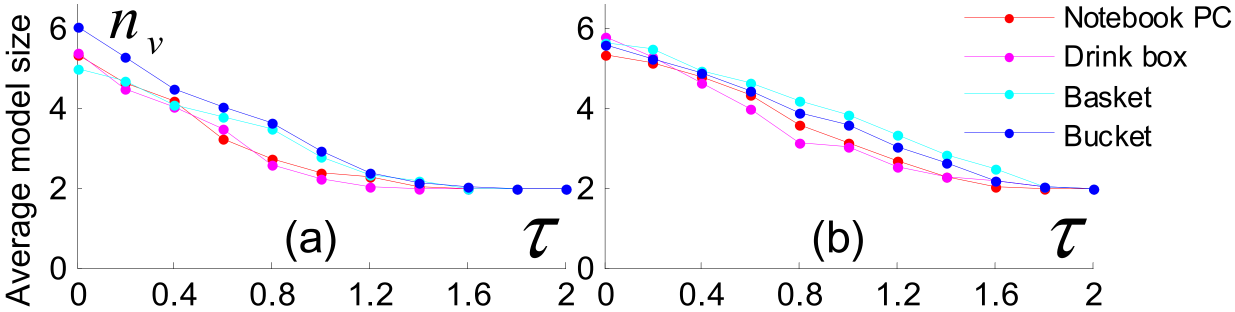


Figure 3.3: Average model size learned by using different thresholds of τ. (a) For 2D models learned from RBD images and (b) for 3D models learned from RGB-D images.

Competing methods: In the first step of the evaluation, I compared my method to

	Category	MA	LS	LT	WM	SU	Mine^{avg}
Matching	Notebook PC	62.02	57.40	61.94	61.91	67.71	**67.79**
	Drink box	59.11	55.66	57.91	62.41	60.68	**88.25**
	Basket	60.88	56.99	59.59	66.10	61.52	**88.61**
	Bucket	61.13	59.15	60.51	64.41	61.90	**89.36**
Recognition	Notebook PC	75.41	70.94	72.11	**82.78**	76.79	81.51
	Drink box	80.09	76.33	78.36	87.73	78.65	**93.42**
	Basket	73.09	70.66	74.59	86.11	71.88	**87.71**
	Bucket	75.52	78.34	77.18	84.63	77.36	**87.97**

Table 3.2: Comparison of average matching rates and average recognition accuracy in the experiment of category modeling from RGB-D images.

several competing approaches to the learning of graph matching across both experiments. First, I performed graph matching without training, denoted by *MA*, to establish a baseline. *MA* uses TRW-S [28] to match the initial graph template to the target objects in images. Second, as the benchmark method for unsupervised learning of graph matching, I used [5] proposed by Leordeanu *et al.*, which does not modify model structure, but rather iteratively train the attribute weights for matching, *i.e.* $\mathbf{w}^U$ and $\mathbf{w}^P$. Two competing approaches were obtained by applying spectral techniques [33] (*LS*) and TRW-S [28] (*LT*), respectively, to solve the matching optimization of (3.4). Third, I designed a framework that iteratively estimates matching parameters $\mathbf{w}^U, \mathbf{w}^P$ and model attributes F_V, F_E according to techniques presented in Section 3.4.1, but did not perform structural modification (*WM*). The final competing technique involved supervised learning of graph matching (*SU*) based on [5]. *SU* required to label target objects in positive graphs and regarded matching assignments mapped onto target objects as correct ones.

In the second step of the evaluation, I compared the performances of the proposed method using different structural modification strategies. The first strategy, (*CB*), is based

		Methods	Average recognition accuracy			
			Notebook	Drink box	Basket	Bucket
The average performance	RGB images	Mine+CB^{avg}	60.96	81.72	72.51	85.58
		Mine+WB^{avg}	**75.36**	87.38	73.87	88.18
		Mine^{avg}	75.21	**89.53**	**76.39**	**89.07**
	RGB-D images	Mine+CB^{avg}	76.46	85.80	85.67	86.48
		Mine+WB^{avg}	80.63	90.64	86.51	86.77
		Mine^{avg}	**81.51**	**93.42**	**87.71**	**87.97**
The best performance	RGB images	Mine+CB^{best}	76.17	85.76	78.94	89.01
		Mine+WB^{best}	80.03	87.91	77.84	89.67
		Mine^{best}	**82.16**	**90.45**	**82.52**	**91.81**
	RGB-D images	Mine+CB^{best}	84.92	91.32	89.77	88.56
		Mine+WB^{best}	84.78	91.78	89.06	88.57
		Mine^{best}	**88.57**	**94.85**	**92.01**	**90.63**

Table 3.3: Recognition performance of different structural modification strategies that can be applied to the proposed learning framework.

on the matching compatibility/penalty, and has been widely used by [11, 24, 18]. *CB* eliminates the node with the lowest average compatibility by replacing (3.11) with $R_i = \sum_k \sum_{j\in V} c_{ij}(\hat{x}_i^k, \hat{x}_j^k|G, G_k^+)/n_v$. The second strategy [41], (*WB*), is oriented toward linear SVM, and uses weights $\mathcal{W}$ for feature selection. *WB* eliminates the node with the smallest weight amplitude by replacing (3.11) with $R_i = \|[\boldsymbol{\mu}_i, \boldsymbol{\rho}_i]^T\|$. Note that for this step of the evaluation, all learning components expect the above structural modification strategies are fixed.

Comparison details: Since my method can obtain different models by setting different values of τ for structural modification, I set τ to be $0, 0.2, 0.4, ..., 2$ during training. This produced different matching and recognition performances, *i.e.* different AMRs and

		Methods	Average matching rate			
			Notebook	Drink box	Basket	Bucket
The average performance	RGB images	Mine+CB^{avg}	37.90	87.75	**82.56**	**88.11**
		Mine+WB^{avg}	49.12	82.89	74.20	84.50
		Mine^{avg}	**50.74**	**89.11**	81.33	86.85
	RGB-D images	Mine+CB^{avg}	63.71	88.06	**93.44**	**90.62**
		Mine+WB^{avg}	65.12	84.75	87.69	86.89
		Mine^{avg}	**67.79**	**88.25**	88.61	89.36
The best performance	RGB images	Mine+CB^{best}	50.39	90.72	**85.45**	89.78
		Mine+WB^{best}	51.71	86.81	76.09	86.29
		Mine^{best}	**56.12**	**94.10**	84.37	**90.22**
	RGB-D images	Mine+CB^{best}	74.40	**93.07**	**96.16**	**91.34**
		Mine+WB^{best}	67.74	89.59	90.40	89.64
		Mine^{best}	**76.92**	93.02	92.40	90.69

Table 3.4: Matching performance of different structural modification strategies that can be applied to the proposed learning framework.

ARAs. Fig. 3.3 shows the changes in model size according to τ, and Fig. 3.2 illustrates the models in the node elimination process. Larger values of τ indicate stricter structural constraints and lead to smaller models.

I used the average/best performance among all settings of τ (*i.e.* the average/largest values for AMR and ARA) to evaluate the proposed method, denoted by $Mine^{avg}/Mine^{best}$. To ensure a fair comparison, for each given τ, *CB* and *WB* were allowed to eliminate nodes until they obtained models with the same size as that produced by *Mine*. Similar to $Mine^{avg}/Mine^{best}$, CB^{avg}/CB^{best} and WB^{avg}/WB^{best} correspond to the average/best performance among all setting of τ. For the comparison of recognition performance, I trained the proposed classifiers using the matching results produced by the competing methods.

Fig. 3.4 illustrates the object detection performances, and Tables 3.1, 3.2, 3.3, and 3.4 list quantitative comparison. Because the competing methods for learning graph matching (*MA*,*LS*,*LT*, *WM*,*SU*) do not have the ability to refine the topological structure of the graph template, they are sensitive to the bias of the initially labeled graph template (including biased attributes and redundant nodes), especially for the unsupervised methods *LS*,*LT*,and *WM*. This bias may produce matching errors, which, in turn, increase the bias in the unsupervised model learning, thus propagating into a significant model bias. In contrast, my method modifies biased structure in early iterations by eliminating badly matched parts, thereby reducing the prevalence of biased matching in later iterations. As a result, my method exhibits better performance.

3.6 Conclusions

In this chapter, I proposed an algorithm for the learning of graph matching. This method trains the structure and attributes of the graph template, as well as matching parameters, to obtain a graphical model. By including negative graphs in the learning process, I orient the model learning toward both object matching and recognition. Experiments show that my approach outperforms competing methods.

My strategy for structural modification is based on the recognition mechanisms between positive and negative graphs, and exhibits better performance than conventional structural modification strategies based only on positive graphs. As the proposed strategy iteratively corrected errors in the topological structure of the initial graph template, it reduced the bias learning problem, which so afflicted pioneering studies in the field.

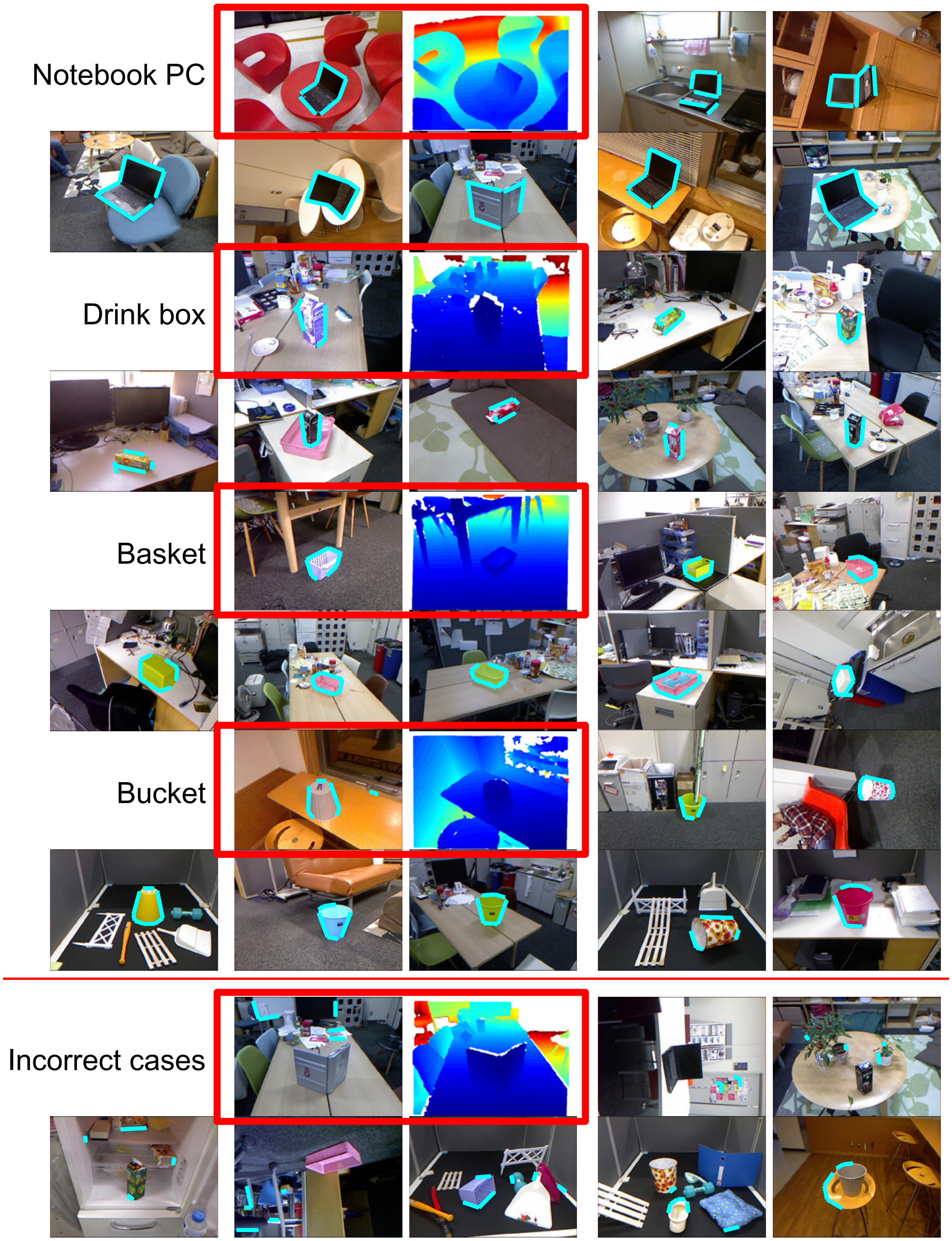


Figure 3.4: Object detection performance in RGB-D images. I only show depth images corresponding to the first RGB image in each category.

Chapter 4

Visual Mining Beyond RGB Images: Use of RGB-D Images

In this chapter, I focus on another direction, *i.e.* using RGB-D images to avoid suffering typical challenges in visual mining. The idea about how to make full use of depth information for visual mining is orthogonal to previous attempts to develop visual-mining platforms that are introduced at the algorithm level in Chapters 2 and 3.

Generally speaking, the additional depth information contains the explicit spatial structures of objects, thereby increasing the robustness to problems of scale and rotation changes in conventional 2D images. Meanwhile, depth information also provides clear object boundaries to guide object segmentation.

Therefore, in this chapter, I propose a strategy that mines the category model from ubiquitous RGB-D images, whereas applies it to object detection and recognition in ordinary RGB images.

The rest of this chapter is organized as follows. The introduction and discussion of related work are presented in Sections 4.1 and 4.2. Section 4.3 introduces the graphical models for RGB-D and RGB images. The basic algorithm for model learning and its technical extensions, as well as model-based recognition, are presented in Sections 4.4. Section 4.5 presents the experiments and the overall study is summarized in Section 4.6.

4.1 Introduction

Knowledge mining of big visual data presents a significant challenge to the field of artificial intelligence. Cognition-level symbolization of visual data is considerably more difficult than that of text data. A number of pioneering studies, such as those involved with deep learning [42] and a variety of visual data mining techniques based on attributed graph mining [10, 9], have addressed this challenge in recent years. However, without sufficient manual labeling and training, the gap between feature-level image processing and object-level visual knowledge remains large. To bridge this gap, I propose use of RGB-D image data, instead of conventional RGB data, as a more productive basis for training.

To be precise, I focus on the problem to construct a category model base that can provide a high-level guidance for many visual tasks, such as image understanding and object recognition. The construction of such a category base poses three main challenges, illustrated in Fig. 4.1.

- **Single labeling:** Model base construction requires some minimum amount of manual labeling, but given a more idealized implementation of semi-supervised learning, *can I learn a category model from just one labeled object and a large set of cluttered images?* Such images are typical of what can be retrieved using most search engines. The target objects they contain are usually small, and require significant hand-cropping and manual alignment for detailed learning, since automatic image segmentation usually cannot ensure object-level results.

- **Structural knowledge:** In many cases, it is structures, rather than textures, that determine functions and categories, especially for many daily-use commercial objects that have regular shapes but a variety of textures. For this reason, I hope to model structural knowledge of different object categories, unlike the conventional mining of “bag-of-words” knowledge.

- **Bias problem:** With the first two challenges, model learning in RGB images is

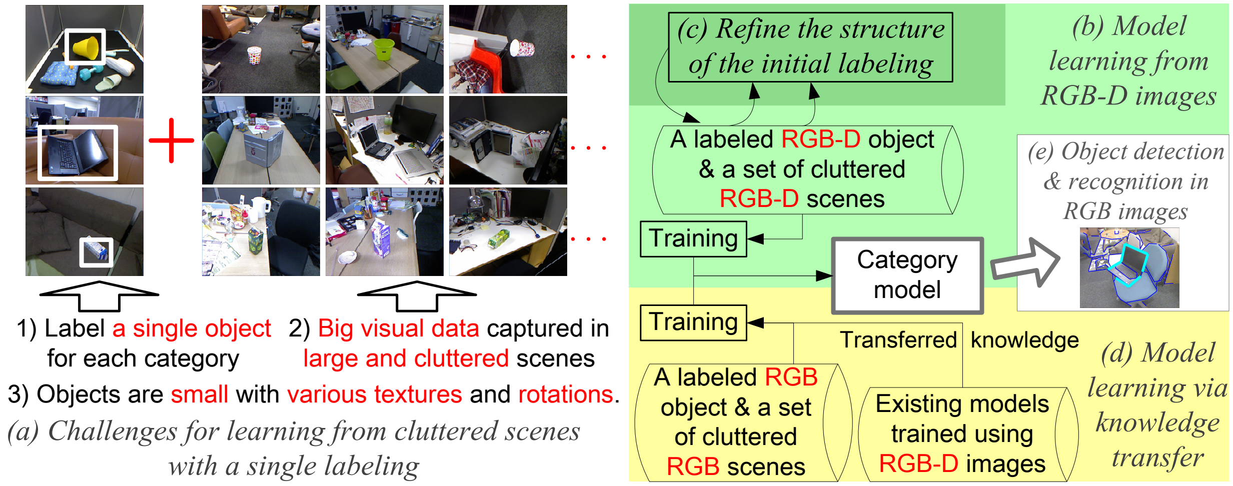

1) Label a single object for each category
2) Big visual data captured in large and cluttered scenes
3) Objects are small with various textures and rotations.
(a) Challenges for learning from cluttered scenes with a single labeling
(c) Refine the structure of the initial labeling
(b) Model learning from RGB-D images
A labeled RGB-D object & a set of cluttered RGB-D scenes
(e) Object detection & recognition in RGB images
Training
Category model
Training
Transferred knowledge
(d) Model learning via knowledge transfer
A labeled RGB object & a set of cluttered RGB scenes
Existing models trained using RGB-D images

Figure 4.1: How can a system learn a structure-based category model from a single labeled object and a number of cluttered scenes? (a) I need to correctly detect and incrementally collect more object samples for training, but since my target objects in large scenes are usually captured informally, they will show large variations in texture and rotation. (b, green part) Thus, I can use 3D shapes in RGB-D images to guide model learning. (c) In particular, I propose a method to refine the initially labeled object using the cluttered RGB-D images, in order to remove inaccuracy or subjective bias in manual labeling. (d, yellow part) When a large number of RGB-D images cannot be collected for a category, I can use knowledge transfer to learn the category model directly from RGB images. The knowledge is transferred from the models that are pre-trained from RGB-D images to guide the training sample collection for a new category. (e, white part) The trained category model can be applied to ordinary RGB images.

caught in a dilemma, which makes the bias problem extremely serious. On one hand, training the structure-based model requires a large collection of small target objects from the image pool, as well as the extraction of part correspondences between these objects, so as to overcome intra-category variations. On the other hand, without sufficient labeling and training, object detection and matching based on a single labeling is hampered by large variations in texture and rotation, both great challenges even for state-of-the-art algorithms. Thus, in this case, bias in object collection in early learning steps will affect subsequent steps, and accumulate into significant model bias.

Thus, *the goal of this chapter is to present an approach to learning a structure-based category model from a single labeled object and a number of large, cluttered images with significant variations in texture and rotation.*

To deal with these variations, I make use of the Kinect [40] device to detect the 3D shape of the object. Kinect RGB-D images provide explicit spatial structures that are robust across variations in texture, 2D scale, and viewpoint. In many cases, the use of 3D structure can greatly improve the reliability of category detection. Meanwhile, ordinary RGB images are more widely used than RGB-D images.

Therefore, I attempt to build a bridge between the two formats, and employ a model learning strategy where the model is trained with RGB-D images and is then applied to ordinary RGB images (Fig. 4.1). I explore this problem at the following three levels.

First, the general idea of this strategy is simple. *I use the more reliable 3D matching results to guide the learning of the less discriminative image-based models*, as illustrated in Fig. 4.2. I begin by using structure-based 3D matching to collect objects from RGB-D images, simultaneously obtaining part correspondences, even in the presence of significant texture variations. This yields a local codebook of visual words learned for each part of the object. The part correspondences in 3D space are then used to train the 2D structural knowledge in the category model. The category model can be applied to object matching in RGB images, and we can also encode the knowledge for object recognition in the model by combining a set of negative[13] RGB images for training.

The second important issue is transfer learning. RGB-D images are used less widely than RGB images, and sometimes I can only collect RGB images for model training. In this case, I need to design a method for transferring the knowledge extracted from the existing category models to guide the training of new category models.

---

[13] In this case, negative RGB images are RGB images that do not contain objects in the target category, which can be collected directly by search engines.

Finally, I focus on the problem of the inaccuracy of initial labeling. The training object collection is based on and sensitive to the labeling of the single initial object. Thus, inaccurate initial labeling may lead to a major bias in model learning. Therefore, I develop a method that uses the cluttered RGB-D images to refine the labeled object into a good category detector for the further object collection. This method deletes the redundant parts of the object, while simultaneously modifying the local textures and structures of the other parts.

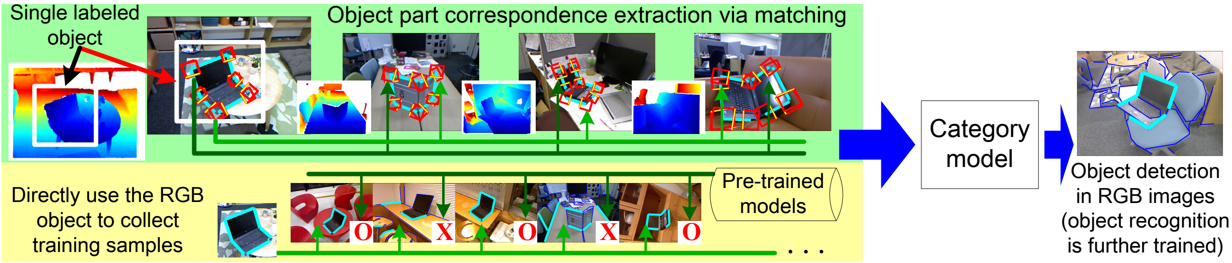

Single labeled object
Object part correspondence extraction via matching
Directly use the RGB object to collect training samples
O
X
O
X
O
Pre-trained models
. . .
Category model
Object detection in RGB images (object recognition is further trained)

Figure 4.2: Flowchart of the proposed method. First, for model learning from RGB-D images (green part), I use the 3D structure of the labeled object to match other objects in the RGB-D images. I then use the part correspondences to train a category model. Second, for model learning via knowledge transfer (yellow part), I label the RGB object to collect training objects directly from RGB images. The models that are trained for other categories provide knowledge to identify the correct and incorrect object matches during sample collection.

To implement these learning strategies, I apply a graphical model that uses object edges as its basic and concise structural elements. Compared to texture features, object edges have a closer relationship to the overall object structure, especially where large texture variations exist. We also develop different sets of attributes to guide the 3D object collection from RGB-D images and the training of 2D category models.

In this study, I use graph matching techniques to achieve both 3D object collection and model-based object matching. Given a template graph (the category model) and multiple target graphs, conventional methods for learning graph matching [5, 43, 3, 6] primarily train matching parameters (*e.g.* the weights of different graphical attributes). In contrast, my method estimates a prototype of the category model and eliminate the specificity of the labeled object in an unsupervised manner. Therefore, I extend the method proposed by [5] to learn model attributes. We have also combined my previous study [7] with this research to refine the structure of the initially labeled object.

The contributions of this chapter can be summarized as follows. I propose three model learning strategies to avoid the bias problem caused by texture variations and various rigid transformations. If RGB-D images in the target category can be collected, I use depth information in the RGB-D images to assist the training of the category model for RGB images. Otherwise, I transfer the knowledge extracted from models of other categories to guide model learning. In particular, when the initial object labeling is inaccurate, I can refine the structure of the labeled object using the cluttered RGB-D images.

A preliminary version of this chapter appeared in [29] and [44].

4.2 Related Work

Visual mining: Learning category models is a familiar problem in the field of computer vision, and many approaches have been proposed over the last few decades. In this section, I limit my discussion to those techniques related to the concept of data mining of big visual data. Generally speaking, these techniques should 1) have loose requirements for training data, and 2) limit the amount of human labeling, perhaps even to the point where unsupervised learning is possible.

The effort to minimize manual labeling makes visual mining related to one-shot learning [45]. Then, for totally unsupervised approaches, object discovery (reviewed by [46]) is a familiar goal in object-level knowledge mining. Most methods that have achieved this goal use bag-of-words models [47] for category representation. Others [48, 49, 50, 51] have managed to detect repetitive objects based on similarities in appearance and visual context. [52, 48, 53] used unsupervised segmentation to generate object candidates, relying on foreground-background discrimination.

In lieu of conventional learning from a large sample pool, Li *et al.* [47] and Grauman *et al.* [54] proposed to collect training images using image search engines via semi-supervised or active learning. These were more efficient ways of constructing a category model base. Approaches to co-segmentation [55, 56, 57, 58, 59] have provided a plausible way to detect and segment common objects from a big image set collected using a search engine.

Most methods for object discovery, one-shot learning, and co-segmentation have proven to be relatively poor for learning structural knowledge of objects. It is much easier to encode the structural knowledge using link analysis (or graph mining) techniques in visual mining. When images are modeled as graphs, it is possible to extract the frequent sub-graphs within these graphs as common objects [1, 11, 22, 23, 24, 25, 18, 19, 26, 20].

However, most of the above methods rely heavily on the similarity of object textures. Many object discovery and one-shot learning methods have directly used bag-of-words models, and co-segmentation approaches have leveraged texture distribution as the primary feature. Even most approaches based on graph link analysis have had to use the inter-image consistency of local patches to generate potential patch correspondences between images, in order to prune the search space of frequent subgraphs. Thus, these approaches are not suitable for mining daily-use objects with large texture variations.

Moreover, I focus on the extraction of precise structural models from large cluttered scenes, rather than the observing probability of patch textures. I use the depth information in RGB-D images to guide the learning process, thus avoiding errors caused by variations in texture, scale, and rotation.

Actually, the RGB-D images have been widely used for object detection and classification. However, conventional methods [60, 61] mainly learned category models from well labeled training samples. In other words, they required a great amount of human labeling to prepare training samples for each category. Whereas, my method is close to the spirit of "model learning", which aims to directly mine the category models from cluttered scenes (a kind of big visual data) without much labeling of "what is where". Model mining involves almost all of the typical issues in computer vision, such as lack of manual alignments and intra-category variations in texture, rotation, scale, and illumination. Thus, it proposes continuous challenges for the state-of-the-art algorithms. In recent years, people began to apply RGB-D images to some "mining" tasks. For example, Fouhey *et al.* [62] and Zhang *et al.* [9] proposed learning reconstruction knowledge from RGB-D images for single-view 3D reconstruction on RGB images.

Category modeling: In this part, I compare different types of category models, and analyze their applicability to informally captured scenes.

Bag-of-words models [47] have been widely used to model categories without encoding structural information. The global structures of objects can be simply represented by the histograms of oriented gradients (HoG) templates [32] or silhouette templates[20, 63, 64]. Hough-style methods [65, 66, 67, 37, 68, 69] have more recently been developed as a sophisticated, supervised means of modeling the spatial relationship between object parts. [70, 71] directly used a 3D model to detect objects in 2D images, whereas [72, 73] used object multi-view appearances to estimate the 3D structure. Recently, the appearance of RGB-D imaging technology has made object detection considerably easier [74, 75, 76], such that some low-level segmentation techniques [77, 61, 78] can roughly achieve object-level results.

However, when I just use a single labeled object to start the category modeling, it can only provide specific 2D structure and appearance from one viewpoint. Thus, the category model should 1) encode the structural information, 2) be able to detect objects with various scales and rotations (even roll rotations), and 3) be learned without further manual labeling.

Graph matching satisfies the first two requirements and has been widely used [38, 79, 80, 81, 82]. The third requirement has been solved by [5], who first proposed an unsupervised[14]

---

[14] In the context of learning graph matching, the term "unsupervised" [5] refers to the ability to learn the model without manually specifying each individual matching assignment from the model to target graphs (cluttered scenes). In other words, under unsupervised learning, I do not have to manually label target objects in the cluttered scenes.

method for learning graph-matching-based models. Though I can use graphical models to represent object categories in both RGB and RGB-D images, 2D object structures in RGB images are not robust with respect to viewpoint changes. Thus, I use a 3D model based on RGB-D images to collect training samples for the learning of 2D models.

4.3 Graphical model of object edge segments and graph matching

I use a graphical model to encode the local and pairwise attributes of objects, *i.e.* the part features and the spatial relationship between them. The model-based object detection is thus achieved via graph matching. This design ensures the robustness of viewpoint variation and roll rotations. In contrast to previous studies based on POI in images, voxels, or surfaces [83, 84] in point clouds, my model uses object edges to represent object structures. I use [31] to extract object edges in RGB images[15] and then discretize them into line segments. The line segments are used as graph nodes, as shown in Fig. 4.3.

I connect each pair of edge segments in the labeled object to generate a complete undirected graph G as the initial category model, in which parameters will be trained later. Given a target scene, let its corresponding target graph be G'. $\mathbf{f}_i$ and $\mathbf{f}_{ij}$ denote the local attributes of node i and pairwise attributes of edge ij in G, respectively. The matching assignments between G and G' can be defined using a matching matrix $\mathbf{y}$, where $y_{ii'} \in \{0,1\}$. Further, $y_{ii'} = 1$ if node i in G maps to node i' in G', otherwise $y_{ii'} = 0$. I set $\sum_{i'} y_{ii'} = 1$ for all i. Therefore, the graph matching is formulated as

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\text{argmax}}\mathcal{C}, \qquad \mathcal{C} = \sum_{ii'} \rho_{ii'} y_{ii'} + \sum_{ii'jj'} \rho_{ii'jj'} y_{ii'} y_{jj'} \tag{4.1}$$

---

[15] Actually, object edges can be directly extracted in 4D space (*i.e. RGB-D space*) [61], but I simply extract edges just from RGB images. This is because I have to ensure edge extraction procedure for training is the same to that for testing in RGB images, in order to avoid potential differences between training and testing.

where $\rho_{ii'}$ and $\rho_{ii'jj'}$ denote attribute compatibility for the unary assignment $i \rightarrow i'$ and the pairwise assignment $ij \rightarrow i'j'$, respectively. In general, they are functions of graph attributes.

$$\rho_{ii'} = \Phi_1(\mathbf{f}_i, \mathbf{f}_{i'}; \mathbf{w}^U), \qquad \rho_{ii'jj'} = \Phi_2(\mathbf{f}_{ij}, \mathbf{f}_{i'j'}; \mathbf{w}^P) \tag{4.2}$$

where $\mathbf{w}^U$ and $\mathbf{w}^P$ are parameter weightings for attributes.

In my study, I bring in an additional dummy matching choice—*none*—that is organized as a node in G'. This is necessary because some parts of the target objects in large cluttered scenes may be occluded; therefore some model nodes should not be matched to target scenes.

$$\rho_{i,none} = \kappa E(\rho_{ii'}), \qquad \rho_{i,none,jj'} = \rho_{ii'j,none} = \kappa E(\rho_{ii'jj'}) \tag{4.3}$$

where κ ($= 1^{16}$, here) controls the matching priority of *none*.

Note that many-to-one matches may introduce errors to the learning of pairwise attributes. Considering that the compatibility in (4.2) is positive in my study, I simply modify unary compatibility as $\rho_{ii'jj'} = -1$ if and only if $i' = j'$ to avoid many-to-one matches.

I design two sets of local and pairwise attributes for the graphical model, so that the model can be applied to object collection (from RGB-D images) and object matching (from RGB images), respectively.

Edge segmentation: I use the local growth strategy to achieve edge segmentation. First, I initialize each pair of neighboring edge points as a tiny line segment, and then gradually merge neighboring segments into longer and straighter lines. I finally map the edge segments in RGB images to the depth space to represent the 3D object structure.

Note that there is local non-smoothness on the edges due to low image quality and texture variations. I therefore design a penalty metric to overcome such non-smoothness in the segment-merging process. As illustrated in Fig. 4.3, I merge neighboring segments u and v into a longer segment. Because angles between shorter segments are more sensitive to

---

[16] In this chapter, most parameters are simply set to 1, and I only manually set a few parameters, such as τ, η, β, and k. This parameter setting is equally applied to all the categories.

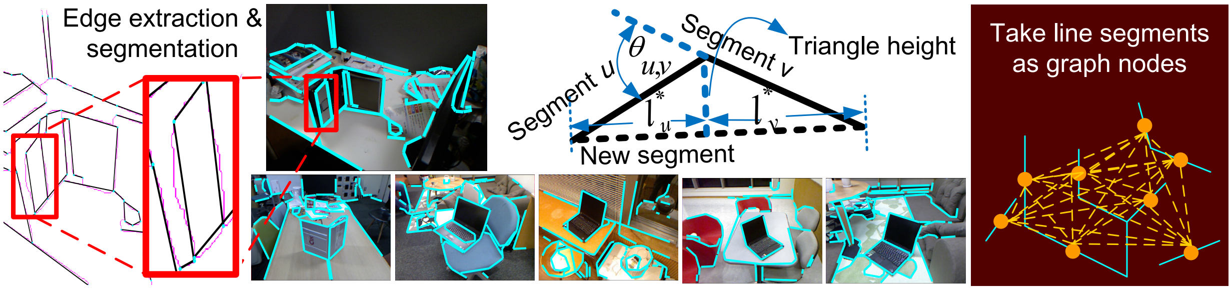


Figure 4.3: Edge segmentation and illustration of variables. (left) Edge extraction and segmentation. (middle) Notation for edge segmentation. (right) Line segments are taken as the graph nodes.

local perturbations, the penalty of their supplementary angle $\theta_{u,v}$ is calculated as $Pen^{\text{angle}}_{u,v} = \theta_{u,v}(1 - U_{u,v})$, where $U_{u,v} = e^{-\tau \min\{l^*_u, l^*_v\}}$ measures the noise level for segment pair of u and v, τ ($= 0.2$, here) controls the decrease speed, and l^*_u and l^*_v are the projected lengths of segments u and v on the new segment.

In addition, I propose another penalty metric that prevents from connecting a very short segment to a long one, as $Pen^{\text{length}}_{u,v} = \frac{l^*_u}{l^*_u + l^*_v} \log \frac{l^*_u}{l^*_u + l^*_v} + \frac{l^*_v}{l^*_u + l^*_v} \log \frac{l^*_v}{l^*_u + l^*_v}$. This metric is created, because the orientation measurement of long segments suffers less from local non-smoothness than that of short segments. I want to avoid transfering the orientation unreliability from the short segment to the long segment during the merge process.

Therefore, the total penalty is calculated as follows.

$$Pen_{u,v} = Pen^{\text{angle}}_{u,v} + \eta Pen^{\text{length}}_{u,v} \tag{4.4}$$

where η ($= 0.5$, here) is a weighting for the two penalty metrics. Segment pairs with lower penalty scores are merged earlier. I set the stopping criterion as follows. For each merge, the height of the triangle consisting of old and new segments should not be greater than six pixel units (Fig. 4.3). Finally, I consider the line segments longer than 15 pixel units to be reliable ones, and select them as graph nodes.

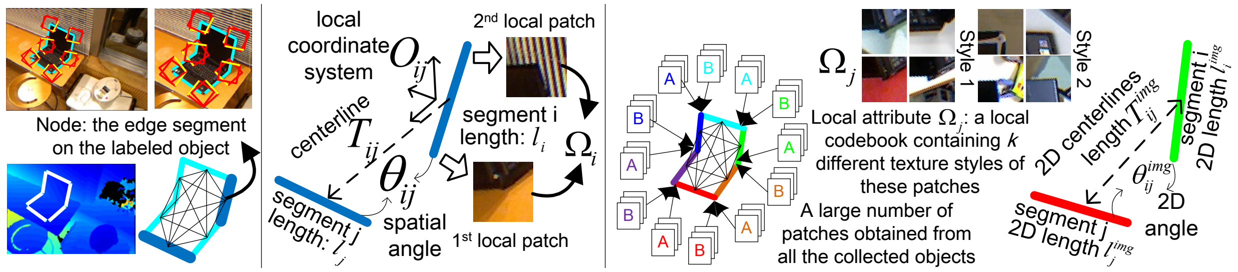


Figure 4.4: Graphical models. (left) Line segments in RGB-D/RGB images are taken as the nodes in a complete graph. The red squares indicate the image patches collected at terminals of the line segments. (middle) Model for object collection from RGB-D images. (right) Category model trained for ordinary RGB images.

4.3.1 Model for object collection from RGB-D images

The graphical model proposed above is a paradigm, and I design a set of attributes to adapt it for collecting objects in RGB-D scenes, while simultaneously extracting the correspondences of local patches between objects for further learning. Please see Fig. 4.4 for the notation of this model.

Spatial length: I take the spatial length, denoted by l_i, as a local attribute. The length penalty for assignment $i \rightarrow i'$ is calculated as $|\log \frac{l_{i'}}{l_i}|$. I can thus obtain length attributes' matching compatibility as

$$P_{ii'}^{length} = e^{-|\log l_i - \log l_{i'}|/\beta} \tag{4.5}$$

where β (= 2, here) responds to the deformability level.

Patch features: I collect two local patches at the terminal points of each edge segment and then normalize them to their *right* orientations. I use their HoG features [32] as another type of local attributes (details follow in Section 4.3.3). Let $\Omega_i = \{\varpi_i^A, \varpi_i^B\}$ denote the HoG features of the two patches of node i in G. I formulate the patch features' compatibility using a Gaussian distribution as

$$P_{ii'}^{patch} = \mathcal{G}([dist(\varpi_i^A, \Omega_{i'}), dist(\varpi_i^B, \Omega_{i'})]^T | \mu = 0, (\sigma^{patch})^2 \mathbf{I}) \tag{4.6a}$$

$$dist(\varpi_i, \Omega_{i'}) = \min_{\varpi_{i'} \in \Omega_{i'}} \|\varpi_i - \varpi_{i'}\|_2 \tag{4.6b}$$

where $\mathcal{G}(\cdot)$ denotes a Gaussian function, and $(\sigma^{patch})^2$ $(= 1$, here) is the covariance. $dist(\cdot, \cdot)$ is a metric for distance measurement between patch features.

Spatial angle: The spatial angle between nodes i and j in G, θ_{ij}, is a widely used pairwise attribute. I assume its compatibility to follow a Gaussian distribution.

$$P^{angle}_{ii'jj'} = \mathcal{G}(\theta_{i'j'} | \mu = \theta_{ij}, (\sigma^{angle})^2) \tag{4.7}$$

where $(\sigma^{angle})^2$ $(= 1$, here) is the variation.

Centerline: Another pairwise attribute describes the relative spatial translation between two nodes. I use the *centerline*—connecting the centers of two node segments—to measure the translation. I define a local 3D coordinate system based on the segments, in order to make the centerline's measurement independent of the global rotation of the object. Let $\mathbf{o_i}$ and $\mathbf{o_j}$ denote the unit 3D orientation of node segments i and j. The three orthogonal unit vectors of this coordinate system are calculated as $\mathbf{O}_{ij} = [\frac{\mathbf{o_i}+\mathbf{o_j}}{\|\mathbf{o_i}+\mathbf{o_j}\|_2}, \frac{\mathbf{o_i}-\mathbf{o_j}}{\|\mathbf{o_i}-\mathbf{o_j}\|_2}, \mathbf{o_i} \times \mathbf{o_j}]$. Thus, the 3D translation $\mathbf{T}_{ij}$ can be measured as $\mathbf{d}^{ij} = \mathbf{O}^T_{ij}\mathbf{T}_{ij}$. Note that I have two choices to define the orientation of node segment i, $\mathbf{o_i}$ and $-\mathbf{o_i}$; therefore, we instead use $\mathbf{c}_{ij} = [\min\{|d^{ij}_1|, |d^{ij}_2|\}, \max\{|d^{ij}_1|, |d^{ij}_2|\}, |d^{ij}_3|]^T$ as the centerline coordinates. The compatibility of centerline coordinates is also assumed to follow a Gaussian distribution.

$$P^{center}_{ii'jj'} = \mathcal{G}(\mathbf{c}_{i'j'} | \mu = \mathbf{c}_{ij}, (\sigma^{\text{cen}}_{ij})^2\mathbf{I}), \qquad (\sigma^{\text{cen}}_{ij})^2 = (\alpha\|\mathbf{c}_{ij}\|_2)^2 + (\sigma^{noise})^2 \tag{4.8}$$

In fact, the variation is caused by both the structural deformability and noise, and I use parameters $\alpha = 1$ and $\sigma^{noise} = 5$ to control the two factors, respectively.

Now, I summarize the model for 3D objects as follows. Its local and pairwise attributes are defined as $\mathbf{f}_i = [l_i, \Omega_i]$, $\mathbf{f}_{ij} = [\theta_{ij}, \mathbf{c}_{ij}]$, and the parameters are denoted by $\mathbf{w}^U = [\beta, \sigma^{patch}]$, $\mathbf{w}^P = [\sigma^{angle}, \sigma^{noise}, \alpha]$. I thus calculate the overall compatibility for unary and pairwise assignments as

$$\rho_{ii'} = \Phi_1(\mathbf{f}_i, \mathbf{f}_{i'}; \mathbf{w}^U) = P^{length}_{ii'} P^{patch}_{ii'}, \qquad \rho_{ii'jj'} = \Phi_2(\mathbf{f}_{ij}, \mathbf{f}_{i'j'}; \mathbf{w}^P) = P^{angle}_{ii'jj'} P^{center}_{ii'jj'} \tag{4.9}$$

Because the local and pairwise attributes are designed based on the explicit 3D object structures, the problem of scale changes in RGB images can be overcome. $\Phi_1(\cdot)$ and $\Phi_2(\cdot)$ are positive bounded functions. I can therefore transform the compatibility maximization in (4.1) to an energy minimization problem and solve it by TRW-S [28]. Finally, I use the matching rate Υ to ensure the overall matching quality: $\Upsilon = N^{detect}/(N^{detect} + N^{none})$, where N^{detect} and N^{none} denote the number of nodes that are matched to real segments in images and *none*, respectively. An incorrect match will produce a large N^{none} and thus a small Υ. Therefore, I only select those matches with $\Upsilon \geq 0.7$ as reliable results for further model learning. Please see Fig.4.6 for object collection performances.

4.3.2 Category model for ordinary RGB images

In ordinary RGB images, depth information can no longer be used. I thereby design new local and pairwise attributes to match objects in RGB images. Please see Fig. 4.4 for the notations.

I learn a local codebook for each node i in G as the only local attribute, which consists of a set of patch features $\Omega_i = \{\varpi_i^k\}, (k = 1, 2, ...)$. The codebook contains different local texture styles to overcome texture variations. Details follow in Section 4.4.1.

I then define three types of pairwise attributes as follows. 1) θ_{ij}^{img} represents the angle between nodes i and j in G on the image plane. 2) The other pairwise attribute is $[\lambda_{ij}^A, \lambda_{ij}^B] = \frac{1}{T_{ij}^{img}}[l_i^{img}, l_j^{img}]$, where l_i^{img} denotes the segment length of node i, and T_{ij}^{img} denotes the length of the centerline between nodes i and j in G. Considering scale changes in RGB images, the length measurement is normalized by T_{ij}^{img}. 3) The third pairwise attribute describes the relative angles between the centerline and line segments of nodes i and j, denoted by $[\theta_{ij}^A, \theta_{ij}^B]$.

I absorb local compatibilities into the pairwise compatibilities, $\mathbf{f}_{ij} = \{\theta_{ij}^{img}$, λ_{ij}^A, λ_{ij}^B, θ_{ij}^A,

θ_{ij}^{B}, Ω_i, $\Omega_j\}$.

$$\rho_{ii'} = 0, \quad \rho_{ii'jj'} = \Phi_2(\mathbf{f}_{ij}, \mathbf{f}_{i'j'}; \mathbf{w}) = e^{-w_1|\theta_{ij}^{img} - \theta_{i'j'}^{img}|^2 - \sum_{k\in\{A,B\}} \left\{ w_2|\lambda_{ij}^{k} - \lambda_{i'j'}^{k}|^2 + w_3|\theta_{ij}^{k} - \theta_{i'j'}^{k}|^2 + w_4[dist^2(\varpi_{i'}^{k}, \Omega_i) + dist^2(\varpi_{j'}^{k}, \Omega_j)] \right\}} \tag{4.10}$$

The distance between the local codebook $dist(\cdot,\cdot)$ is defined in (4.6b). Similar to the model for RGB-D images, I use the TRW-S [28] to solve the maximization problem.

4.3.3 HoG feature extraction

This subsection introduces the details of HoG feature extraction that is present in both the graphical model for RGB-D images and the one for RGB images. For each model node, I extract a set of patches from its matched node segments in the target scenes, as shown in Fig. 4.6. I first extract these patches from the two terminals of the edge segment, and then normalize them to their *right* orientations to eliminate rotation effects. As shown in Fig. 4.5, the patches are collected using a square, which is rotated to the orientation of the edge segment.

HoG features [32] are extracted using 5×5 cells, each of which covers half of its neighboring cells. I use four orientation bins (from 0° to 180°) to compute the gradient histogram in each cell. Because the patch is locally collected without significant illumination changes, I normalize all of the cells within a single block.

4.4 Model learning algorithms

In this section, I focus on the model-learning algorithm and its several technical extensions. The framework of the basic algorithm, *i.e.* model learning from RGB-D images, is proposed in Section 4.4.1. Then, in Section 4.4.2, I present the recognition method based on the trained models. In addition, I further extend the basic model-learning method to overcome two main challenges of visual mining by applying the strategy of knowledge transfer and initial labeling refinement, which are presented in Sections 4.4.3 and 4.4.4, respectively.

4.4.1 Basic framework: model learning from RGB-D images

In this subsection, I use relatively reliable 3D matches to guide the training of the category model for ordinary RGB images, in order to avoid the bias problem. Based on the part correspondences estimated in 3D matching, I construct a local codebook for each model node that covers all possible local texture styles. [5] is then extended for the training of both matching parameters and model attributes.

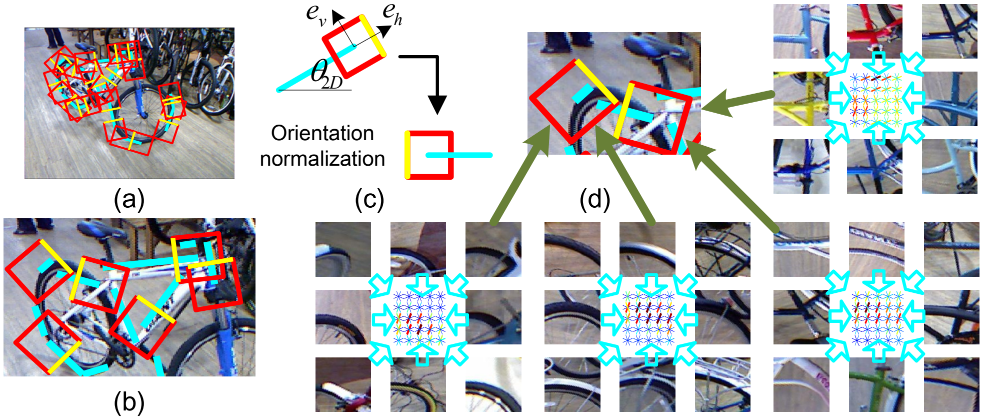


Figure 4.5: Local codebook extraction. (a) The bicycle is detected by 3D matching. Patches (red) are extracted at terminals of the detected segments (blue). Yellow sides indicate patch orientations. (b) A detailed view. (c) Patch orientation normalization. (d) Given images patches corresponding to each of the two bicycle parts, I use k-means clustering ($k = 2$ for clarity) to obtain two texture styles as a sparse local codebook. I use the HoG template to represent the texture style of its surrounding image patches.

Local codebook extraction

During RGB-D object collection, I collect a set of image patches that correspond to each node in the category model. I then cluster the HoG features of each node i's patches via k-means clustering ($k = 5$). Consequently, the cluster centers represent a sparse set of visual words for this node, which compose the local codebook Ω_i. Fig. 4.5(d) shows that I use a set of HoG patterns to describe an object part's different texture styles exhibited in different images. The local codebook therefore contains sufficient texture patterns to overcome texture variations during object detection.

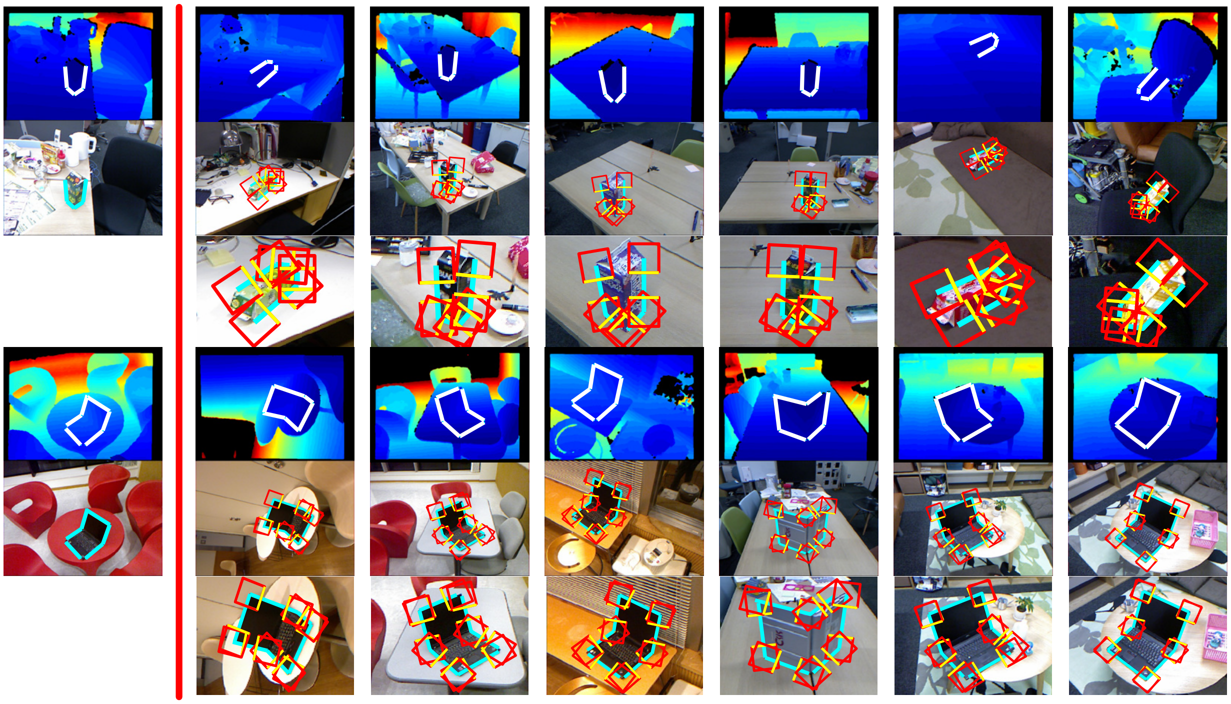


Figure 4.6: Object collection. Given the single labeled object (to the left of the red line), target objects are detected using the 3D structure (1^{st} and 4^{th} rows), and local patches are collected for each object part (2^{nd} and 5^{th} rows). The 3^{rd} and 6^{th} rows show detailed views of patch extraction. Patches (red) are extracted at terminals of the detected segments (blue). Yellow sides indicate patch orientations.

Model-learning algorithm

I can rewrite the model-based graph matching defined by (4.1) and (4.10) as

$$\arg\max_{\mathbf{y}} \mathcal{C} = \arg\max_{\mathbf{y}} \mathbf{y}^T \mathbf{M} \mathbf{y} \tag{4.11}$$

where $M_{(ii'),(jj')} = \rho_{ii'jj'}$. In this equation, $\mathbf{y}$ is transformed from a matching matrix to a vector.

According to [33], elements of the principal eigenvector $\mathbf{x}$ of $\mathbf{M}$, *e.g.* $x_{ii'}$, can be taken as the confidence value of the assignment $i \rightarrow i'$[17]. To reduce the large computation, I apply the approximate principal eigenvector used in [5].

$$\mathbf{x} = \mathbf{M^n 1} / \sqrt{(\mathbf{M^n 1})^T (\mathbf{M^n 1})} \tag{4.12}$$

The partial derivatives of $\mathbf{x}$ are thus computed as

$$\mathbf{x}' = \left[(\mathbf{M^n 1})' \|\mathbf{M^n 1}\| - ((\mathbf{M^n 1})^T (\mathbf{M^n 1})') \mathbf{M^n 1} / \|\mathbf{M^n 1}\| \right] / \|\mathbf{M^n 1}\|^2 \tag{4.13}$$

where $(\mathbf{M^n 1})' = \mathbf{M}'(\mathbf{M^{n-1} 1}) + \mathbf{M}(\mathbf{M^{n-1} 1})'$ and $n = 10$, as in [5].

Leordeanu *et al.* [5] proposed training matching parameters $\mathbf{w}$ to increase confidence values $x_{ii'}$ of the correct assignments. At the same time, confidence values of incorrect assignments will decrease, as $\mathbf{x}$ is normalized. I extend this idea to learn both the parameters and the model attributes $\{\mathbf{w}, \mathbf{f}\}$ by maximizing the following function.

$$\mathcal{F}(\mathbf{w}, \mathbf{f}) = \sum_{i=1}^{N} (\mathbf{x}^{(i)}(\mathbf{w}, \mathbf{f}))^T \mathbf{t}^{(i)} \tag{4.14}$$

where $i = 1, 2, ..., N$ indicates each target scene for training, and $\mathbf{t}^{(i)}$ denotes the predicted matching assignment in scene i.

I implement the whole model training framework as follows. I first initialize $\{\mathbf{w}, \mathbf{f}\}$ using the labeled object, and then iteratively modify $\{\mathbf{w}, \mathbf{f}\}$ to maximize $\mathcal{F}(\mathbf{w}, \mathbf{f})$. Intuitively, I can directly predict the correct matching assignments as the 3D matching results in the RGB-D image, $\mathbf{t}^{(i)} = \hat{\mathbf{y}}^{3D,(i)}$. However, some categories may have several potential assignment states, owing to their symmetric 3D structures, *e.g.* notebook PCs. These matching states are equivalent in terms of the 3D structure; however, they may show different matching compatibilities for ordinary 2D image matching. Thus, the matching assignments predicted by the category model (denoted by $\hat{\mathbf{y}}^{img,(i)}$) are not always the same as $\hat{\mathbf{y}}^{3D,(i)}$.

---

[17] Note that $\rho_{i,none,jj'}$ and $\rho_{ii'j,none}$ are not involved in M.

I therefore use $\hat{\mathbf{y}}^{img,(i)}$ to compute $\mathbf{t}^{(i)}$. Errors in $\hat{\mathbf{y}}^{img,(i)}$ are detected and eliminated by $\hat{\mathbf{y}}^{3D,(i)}$ to avoid the bias problem. I use the following criterion to identify correct matches from $\hat{\mathbf{y}}^{img,(i)}$. Correct 2D matches in $\hat{\mathbf{y}}^{img,(i)}$ should match the nodes in image i that are also matched by 3D matching $\hat{\mathbf{y}}^{3D,(i)}$. Thus, I get

$$\mathbf{t}^{(i)} = diag\{a_{jj'}^{(i)}\}\hat{\mathbf{y}}^{img,(i)}, \qquad a_{jj'}^{(i)} = \sum_j \hat{y}_{jj'}^{3D,(i)} \tag{4.15}$$

In the k-th iteration, I use (4.15) to estimate matching assignment $\mathbf{t}^{(i),k}$ and modify each $w \in \mathbf{w}$ and $f \in \mathbf{f}$ via gradient ascent.

$$w^{k+1} \leftarrow w^k + \zeta \sum_{i=1}^{N} (\mathbf{t}^{(i),k})^T \frac{\partial \mathbf{x}^{(i),k}(\mathbf{w}, \mathbf{f})}{\partial w}, \quad f^{k+1} \leftarrow f^k + \zeta \sum_{i=1}^{N} (\mathbf{t}^{(i),k})^T \frac{\partial \mathbf{x}^{(i),k}(\mathbf{w}, \mathbf{f})}{\partial f} \tag{4.16}$$

4.4.2 Object recognition based on category models

Originally, the category model is trained for object detection in RGB images. This section introduces three methods to further train object recognition, given the trained category models. Successful object recognition is defined as the correct determination of whether an image contains the target object of a model. Given a category model, I use a set of positive RGB images and the same number of negative RGB images to train object recognition. The positive images are the RGB channels of the training RGB-D images used in Section 4.4.1. The negative images are the images that do not contain the target objects, which are selected randomly from the RGB image sets of other categories.

The most popular and simplest method for object recognition is based on matching compatibility[18] [11, 24, 18]. Given a testing RGB image, I compute the matching compatibility $\mathcal{C}$ in (4.1) between this image and the model. If $\frac{\mathcal{C}-\mu_{\mathcal{C}}}{\sigma_{\mathcal{C}}}$ is greater than a threshold υ, the matching results are considered to represent the true detection of the target object, otherwise a false detection. $\mu_{\mathcal{C}}$ and $\sigma_{\mathcal{C}}$ are the mean and standard deviation of $\mathcal{C}$ when I

---

[18] This recognition approach can be used, if the target function of graph matching is designed to maximize the matching compatibilities or to minimize the matching penalties.

match the model to all of the positive and negative training images[19].

The second method trains a kernel-based support vector machine (SVM) for object recognition as

$$f(\mathbf{y}) = sgn\Big(\sum_{i^+} \alpha_{i^+} K(\mathbf{y}_{i^+}, \mathbf{y}) - \sum_{i^-} \alpha_{i^-} K(\mathbf{y}_{i^-}, \mathbf{y}) + b\Big) \tag{4.17}$$

where $\mathbf{y}$ denotes the matching results in the testing image computed in (4.1); $\mathbf{y}_{i^+}$ and $\mathbf{y}_{i^-}$ are the matching results in each positive and negative image, respectively.

I modify the kernel for graph matching proposed in [85, 38] to classify the matching results in positive and negative images.

$$\begin{aligned} K(\mathbf{y}, \mathbf{y}') = \sum_{ij} \exp\Big\{ & - w_1 |\theta^{img}_{y(i)y(j)} - \theta^{img}_{y'(i)y'(j)}|^2 - w_4 |\frac{1}{2} \sum_{k\in\{A,B\}} (\varpi^k_{y(i)} - \varpi^k_{y'(i)})|^2 \\ & - \sum_{k\in\{A,B\}} \big[w_2 |\lambda^k_{y(i)y(j)} - \lambda^k_{y'(i)y'(j)}|^2 + w_3 |\theta^k_{y(i)y(j)} - \theta^k_{y'(i)y'(j)}|^2 \big] \Big\} \end{aligned} \tag{4.18}$$

This is a Mercer kernel (please see [85] for the proof), which measures the similarity between the corresponding parts of $\mathbf{y}$ and $\mathbf{y}'$. The functions $y(i)$ and $y'(i)$ denote the matching assignments of model node i *w.r.t.* $\mathbf{y}$ and $\mathbf{y}'$. Let k be a node in an image. $y(i) = k$, if and only if $\mathbf{y}_{ik} = 1$. The other notation is presented in (4.10). Note that I simply define $f_{y(i)y(j)} = -\mathbf{1}$, if $y(i)$ or $y(j)$ is equal to *none* for any $f_{y(i)y(j)} \in \{\theta^{img}_{y(i)y(j)}, \lambda^A_{y(i)y(j)}, \lambda^B_{y(i)y(j)}, \theta^A_{y(i)y(j)}, \theta^B_{y(i)y(j)}, \Omega_{y(i)}, \Omega_{y(j)}\}$.

The final approach for the recognition of graph matching results is the technique used in [7]. This method designs a set of features to measure the matching quality of each graph matching result, and then trains a linear SVM for object recognition. The features of matching result $\mathbf{y}$ are defined as $\mathbf{f}' =< f'_i >$, where f'_i denotes the feature for node i of the category model, and N^{model} is the node number of the model. f'_i is computed as

$$\begin{aligned} f'_i = \mathrm{sqrt}\Big\{ & w_4 \sum_{k\in\{A,B\}} dist^2(\varpi^k_{y(i)}, \Omega_i) + \frac{1}{N^{model}} \sum_j \big[w_1 |\theta^{img}_{ij} - \theta^{img}_{y(i)y(j)}|^2 \\ & + \sum_{k\in\{A,B\}} \big(w_2 |\lambda^k_{ij} - \lambda^k_{y(i)y(j)}|^2 + w_3 |\theta^k_{ij} - \theta^k_{y(i)y(j)}|^2 \big) \big] \Big\} \end{aligned} \tag{4.19}$$

---

[19] The value range of $\mathcal{C}$ varies among models, because of variations in the value of $\mathbf{w}$, but the normalization of $\mathcal{C}$ aims to remove such effects.

4.4.3 Technical extension 1: model learning via knowledge transfer

In this subsection, I focus on model learning from ordinary RGB images, because it is not realistic to obtain a large number of RGB-D training images for every category. The RGB images can be easily collected using search engines[20]. However, without the guidance of 3D structural information, the bias problem in model learning becomes severe due to incorrect object collection (graph matching) in large and cluttered scenes (which is demonstrated by the experimental results later). Fortunately, I can transfer some common knowledge from the existing models of some categories (trained using RGB-D images) to guide the model learning for a new category. I use this knowledge to identify incorrect matches, which reduces the bias problem to some extent.

Model attributes and parameters of the existing models only contain information related to their own categories, not the new category. However, they have similar rules for identifying correct and incorrect graph matching. In general, if graph matching is correct, the attribute difference between two matched graphs is small, whereas it is large otherwise. Thus, for each type of attribute, I can learn the value ranges of the attribute differences for correct and incorrect graph matching. I define the following feature for the matching result $\mathbf{y}$ to measure the attribute differences.

$$F_{\mathbf{y}} = [\underset{ij}{mean}(F_{ij}^1), \underset{ij}{var}(F_{ij}^1), \underset{ij}{mean}(F_{ij}^2), \underset{ij}{var}(F_{ij}^2), \underset{ij}{mean}(F_{ij}^3), \underset{ij}{var}(F_{ij}^3), \underset{i}{mean}(F_i^4), \underset{i}{var}(F_i^4)]^T \tag{4.20}$$

where the functions $mean(\cdot)$ and $var(\cdot)$ compute the mean and variation, and $y(i)$ denotes the matching assignments of model node i *w.r.t.* $\mathbf{y}$, which is similar to that in (4.18). $F_{ij}^1 = |\theta_{y(i)y(j)}^{img} - \theta_{ij}^{img}|$, $F_{ij}^2 = \sum_{k\in\{A,B\}} |\lambda_{y(i)y(j)}^k - \lambda_{ij}^k|$, $F_{ij}^3 = \sum_{k\in\{A,B\}} |\theta_{y(i)y(j)}^k - \theta_{ij}^k|$, $F_i^4 = |\sum_{k\in\{A,B\}}(\varpi_{y(i)}^k - \varpi_i^k)|$.

Therefore, I can use feature $F_{\hat{\mathbf{y}}^{img,(i)}}$ to identify the correctness of the model's matching result in image i. I prepare a set of correct (positive) and incorrect (negative) matching

---

[20] I use the RGB channels of the RGB-D images in the dataset [30] as the training images in experiments.

results[21] for each existing model. Thus, I obtain a high number of matching results based on the existing models to train a nonlinear SVM for classifying the matching results.

$$\begin{gathered}\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}^T\mathbf{w} + C_1 \sum_{i^+} \xi_{i^+} + C_2 \sum_{i^-} \xi_{i^-} \\ \text{subject to } \mathbf{w}^T\phi(F_{i^+}) + b \geq 1 - \xi_{i^+};\ -(\mathbf{w}^T\phi(F_{i^-}) + b) \geq 1 - \xi_{i^-};\ \xi_{i^+}, \xi_{i^-} \geq 0\end{gathered} \tag{4.21}$$

where $K_{RBF}(\cdot,\cdot) = \phi(\cdot)^T\phi(\cdot)$ is a radial basis function (RBF) kernel. I set $C_1 = 1$ and $C_2 = 10$, which consider the unavoidable incorrect matching results produced by the existing models. F_{i^+} and F_{i^-} denote the positive and negative training samples[22], respectively.

I modify the algorithm for "learning from RGB-D images" proposed in Section 4.4.1 to implement this knowledge-transfer-based learning. Two modifications are applied, as follows. First, I modify the learning of model parameters and attributes in the k-th iteration from Equation (4.16) to

$$w^{k+1} \leftarrow w^k + \zeta \sum_{i=1}^{N} \pi_i^k (\hat{\mathbf{y}}^{img,(i),k})^T \frac{\partial \mathbf{x}^{(i),k}(\mathbf{w},\mathbf{f})}{\partial w},\ \ f^{k+1} \leftarrow f^k + \zeta \sum_{i=1}^{N} \pi_i^k (\hat{\mathbf{y}}^{img,(i),k})^T \frac{\partial \mathbf{x}^{(i),k}(\mathbf{w},\mathbf{f})}{\partial f} \tag{4.22}$$

where $\hat{\mathbf{y}}^{img,(i),k}$ denotes the matching assignments in image i determined by the category model in the k-th iteration, which is the same as the notation in (4.15). The parameter π_i^k corresponds to the transferred knowledge. π_i^k is the reliability of the matching results $\hat{\mathbf{y}}^{img,(i),k}$, which is used to assign a low weight to a possibly biased matching result in the learning process. π_i^k is computed using the decision value of the SVM, as described in [86].

$$\pi_i^k = 1 \Big/ \Big\{ 1 + \exp\big\{ -A\big[\sum_{i^+} \alpha_{i^+} K_{RBF}(F_{i^+}, F_{\hat{\mathbf{y}}^{img,(i),k}}) - \sum_{i^-} \alpha_{i^-} K_{RBF}(F_{i^-}, F_{\hat{\mathbf{y}}^{img,(i),k}}) + b\big]\big\} \Big\} \tag{4.23}$$

I set $A = 3$ in this case.

---

[21] The correct and incorrect matching results are the matching results in the positive and negative RGB images *w.r.t* the category of the model, which are the same as those described in Section 4.4.2.

[22] The existing models are all well trained, which means that they usually produce smaller attribute differences than a target model that is still in training. Thus, during the preparation of each training sample (F_{i^+} or F_{i^-}), I randomly select another positive matching result $\mathbf{y}^+$, and use the matched attributes of $\mathbf{y}^+$ to replace the model attributes. In other words, (4.20) is modified, *e.g.* $F_{ij}^1 = |\theta_{y(i)y(j)}^{img} - \theta_{ij}^{img}| \rightarrow F_{ij}^1 = |\theta_{y(i)y(j)}^{img} - \theta_{y'(i)y'(j)}^{img}|$.

Second, I cannot pre-train a local texture codebook for each node in the category model for preprocessing, because the part correspondences between training images cannot be extracted reliably without depth information. Instead, I simply set the local codebook as the patch features $\Omega_i = \{\varpi_i^A, \varpi_i^B\}$ for each node i, and train Ω_i based on (4.22), just as other parameters w_j.

4.4.4 Technical extension 2: initial labeling refinement

The basic model learning method introduced in Section 4.4.1 demands appropriate labeling of the initial RGB-D object to avoid bias during training object collection. People are capable of accurate labeling in most cases, but subjective labeling cannot be guaranteed to reflect the underlying structural features of a category in all cases. Therefore, I also utilize my previous method [7] to refine the initial labeling before model learning.

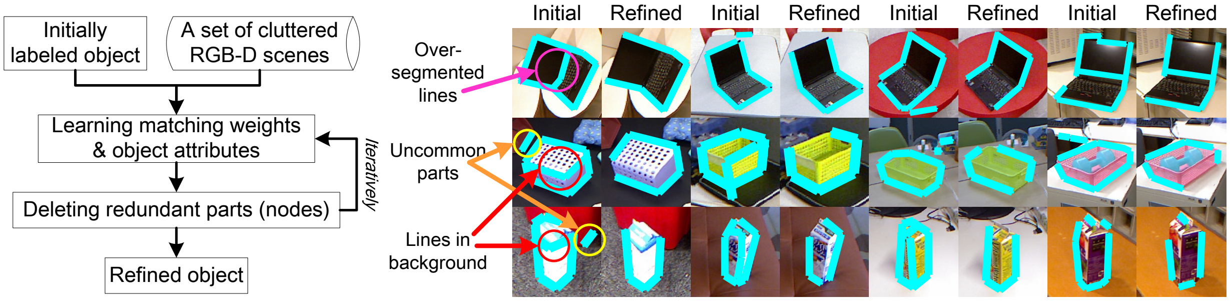


Figure 4.7: Structure refinement of the initially labeled object. The flowchart is shown on the left, and the results are shown on the right.

In addition to the set of positive RGB-D images (containing target objects), I collect a set of negative (background) RGB-D images for learning. I train the local and pairwise attributes of the labeled RGB-D object, which is similar to the method introduced in Section 4.4.1. Further, I modify the object structure at the same time. The labeled object should comprise the object parts (nodes) that perform most reliably during graph matching, thereby facilitating appropriate guidance when training RGB object models.

Graph matching in RGB-D images: Similar to (4.1) and (4.11), I re-write the graph matching between the labeled object $G(V,E)$ and an RGB-D image $G'(V',E')$, as follows.

$$\begin{aligned} &\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmax}}\, \mathcal{C}(\mathbf{y}|G,G'), \quad \mathcal{C}(\mathbf{y}|G,G') = \mathbf{y}^T\mathbf{M}\mathbf{y} \\ &\text{s.t.} \quad \forall i \in V, \sum_{i' \in V'} y_{ii'} \le 1, \quad \forall i' \in V', \sum_{i \in V} y_{ii'} \le 1 \end{aligned} \tag{4.24}$$

In graph G, I use the local and pairwise attributes introduced in Section 4.3.1, including two local attributes ($n^U = 2$) and three pairwise attributes ($n^P = 3$). I clarify the notation as follows. The local attributes of node i include the HoG patch features $f_i^{(1)} = \Omega_i$ and segment length $f_i^{(2)} = \log l_i$. The three pairwise attributes comprise the segment angle $f_{ij}^{(1)} = \theta_{ij}$ and the centerline's attributes $f_{ij}^{(2)} = \|\mathbf{c}_{ij}\|$, $f_{ij}^{(3)} = \mathbf{c}_{ij}/\|\mathbf{c}_{ij}\|$. $F_V = \{f_i^{(k)}|k=1,2; i \in V\}$, $F_E = \{f_{ij}^{(k)}|k=1,2,3;(i,j) \in E\}$. Thus, I apply the following matching compatibility matrix.

$$M_{ii',jj'} = \begin{cases} \exp\Big(-(\mathbf{w}^U)^{\mathbf{T}}\mathbf{d}_{ii'}^2 - (\mathbf{w}^U)^{\mathbf{T}}\mathbf{d}_{jj'}^2 - (\mathbf{w}^P)^{\mathbf{T}}\mathbf{d}_{ii',jj'}^2\Big), & (i,j) \in E, (i',j') \in E' \\ 0, & \text{Otherwise} \end{cases} \tag{4.25}$$

where I define $\mathbf{d}_{ii'} = [d_{ii'}^{(1)}, d_{ii'}^{(2)}, ..., d_{ii'}^{(n^U)}]^T$ as the distances of the corresponding unary attributes during matching, $d_{ii'}^{(k)} = \|f_i^{(k)} - f_{i'}^{(k)}\|$[23]. In addition, $\mathbf{d}_{ii',jj'} = [d_{ii',jj'}^{(1)}, d_{ii',jj'}^{(2)}, ..., d_{ii',jj'}^{(n^P)}]^T$ denote the distances of the pairwise attributes, $d_{ii',jj'}^{(l)} = \|f_{ij}^{(l)} - f_{i'j'}^{(l)}\|$. $\mathbf{w}^U = [w_1^U, w_2^U, ..., w_{n^U}^U]^T$ and $\mathbf{w}^P = [w_1^P, w_2^P, ..., w_{n^P}^P]^T$ denote the weights for each unary and pairwise attribute, respectively.

During the actual application of graph matching, I add a dummy node *none* and a penalty for many-to-one matches. Equation (4.24) is re-formulated as

$$\hat{\mathbf{a}} = \underset{\mathbf{a}}{\operatorname{argmax}} \sum_{i,j \in V \cup \{none\}} \mathcal{C}_{ij}, \quad \mathcal{C}_{ij} = \begin{cases} M_{ia_i,ja_j}, & a_i \ne a_j \in V' \\ -\infty, & a_i = a_j \in V' \\ \frac{\lambda(\mathbf{1}^T\mathbf{M}\mathbf{1})}{n_v^2 n_{v'}^2}, & a_i \text{ or } a_j = none \end{cases} \tag{4.26}$$

where a_i indicates the matching assignment of node $i \in V$ and $a_i = i' \in V'$, if and only if $y_{ii'} = 1$. λ $(=5)$ is the parameter weighting for the penalty of *none*. n_v and $n_{v'}$ denote the node number of G and G', respectively.

---

[23] Note that, the distance of patch features $f_i^{(1)} = \Omega_i$ is defined in (4.6b).

Learning matching weights and attributes: The method used for the learning the matching weights and attributes is similar to that described in Section 4.4.1. Let $\mathbf{x}$ denote the principal eigenvector of $\mathbf{M}$, which is approximated by $\mathbf{x} = \mathbf{M^n 1}/\sqrt{(\mathbf{M^n 1})^T(\mathbf{M^n 1})}$. The learning process is performed iteratively. During each iteration, I use the current G to predict the matching assignments, and then, I modify the matching weights ($w \in \mathbf{w}^U \cup \mathbf{w}^P$) and attributes ($f \in F_V \cup F_E$) by gradient ascent:

$$w^{k+1} \leftarrow w^k + \zeta \sum_{i^+} (\mathbf{t}^{(i^+),k})^T \frac{\partial \mathbf{x}^{(i^+),k}}{\partial w}, \quad f^{k+1} \leftarrow f^k + \zeta \sum_{i^+} (\mathbf{t}^{(i^+),k})^T \frac{\partial \mathbf{x}^{(i^+),k}}{\partial f} \tag{4.27}$$

where similar to (4.16), $\mathbf{t}^{(i^+),k}$ denotes the predicted matching assignments for positive image i^+ computed in iteration k. $t_{jj'}^{(i^+),k}$ is set to 1, if $a_j = j'$, 0 otherwise. Please see [7] for further details.

Structure modification: Structure modification is combined with the iterative framework for training weights and attributes. During each iteration, after parameter regression in (4.27), I delete a redundant part (node) of G, until G has the pre-determined number of nodes. I identify the redundant node based on the assumption that a redundant node usually contributes less than other nodes to the classification of positive and negative images.

First, I need to define the features of each node for classification. Let $\hat{\mathbf{A}} = \{\hat{a}_i^k | k = 1, 2..., N^+, i \in V\}$ and $\mathring{\mathbf{A}} = \{\mathring{a}_i^l | l = 1, 2..., N^-, i \in V\}$ denote the predicted matching assignments of G, where $\hat{a}_i^k$ and $\mathring{a}_i^l$ indicate the assignment mapping node i to positive graph k and that to negative graph l based on (4.26)[24]. Thus, according to (4.25), $\mathbf{d}_{i\hat{a}_i^k}$ indicates the distance of the unary attributes for matching node $i \in V$ to node $\hat{a}_i^k$ in positive graph k. $\mathbf{d}_{i\mathring{a}_i^l}$ indicates such distance for matching to the negative graph l. Similarly, $\mathbf{d}_{i\hat{a}_i^k, j\hat{a}_j^k}$ and $\mathbf{d}_{i\mathring{a}_i^l, j\mathring{a}_j^l}$ denote the pairwise attribute distances in positive and negative graphs, respectively.

Therefore, I use $\hat{\mathbf{u}}_i^k$ and $\hat{\mathbf{p}}_i^k$ (n^U-dimensional and n^P-dimensional vectors) as the matching incompatibilities of the unary and pairwise attributes, respectively, for the match between

---

[24] In this case, I set $\lambda = -\infty$ to avoid matching with *none*.

node $i \in V$ and node $\hat{a}_j^k$.

$$\hat{\mathbf{u}}_i^k = \mathbf{d}_{i\hat{a}_i^k}^T, \quad \hat{\mathbf{p}}_i^k = \sum\nolimits_{j:j\neq i} \mathbf{d}_{i\hat{a}_i^k, j\hat{a}_j^k}^T \Big/ \sum\nolimits_{j:j\neq i} 1 \tag{4.28}$$

The features for recognizing matches with positive graph k and negative graph l are defined as

$$\hat{\mathcal{F}}^k = [\hat{\mathbf{u}}_1^k, \hat{\mathbf{p}}_1^k, \hat{\mathbf{u}}_2^k, \hat{\mathbf{p}}_2^k, ..., \hat{\mathbf{u}}_{n_v}^k, \hat{\mathbf{p}}_{n_v}^k]^T, \quad \mathring{\mathcal{F}}^l = [\mathring{\mathbf{u}}_1^l, \mathring{\mathbf{p}}_1^l, \mathring{\mathbf{u}}_2^l, \mathring{\mathbf{p}}_2^l, ..., \mathring{\mathbf{u}}_{n_v}^l, \mathring{\mathbf{p}}_{n_v}^l]^T \tag{4.29}$$

Consequently, I use these features to train a linear-SVM classifier for match classification and then, I use this classifier to find the redundant node in G.

$$\min_{\mathcal{W},\xi,b} \Big\{ \frac{1}{2}\|\mathcal{W}\|^2 + C\sum_{k=1}^{N^+ + N^-} \xi_k \Big\}, \qquad \begin{array}{c} \text{s.t.} \quad \forall k=1,2,...,N^+, \mathcal{W}\cdot\hat{\mathcal{F}}^k - b \geq 1-\xi_k,\ \xi_k \geq 0; \\ \forall k=1,2,...,N^-, -(\mathcal{W}\cdot\mathring{\mathcal{F}}^k - b) \geq 1-\xi_{k+N^+},\ \xi_{k+N^+} \geq 0 \end{array} \tag{4.30}$$

I represent the normal vector *w.r.t.* the hyperplane $\mathcal{W}$ as $[\boldsymbol{\mu}_1^T, \boldsymbol{\rho}_1^T, \boldsymbol{\mu}_1^T, \boldsymbol{\rho}_1^T, ..., \boldsymbol{\mu}_{n_v}^T, \boldsymbol{\rho}_{n_v}^T]^T$. $\boldsymbol{\mu}_i$ is a n^U-dimensional vector, which weights for the matching incompatibility of node i's unary attributes ($\hat{\mathbf{u}}_i^k$ or $\mathring{\mathbf{u}}_i^l$), whereas the n^P-dimensional $\boldsymbol{\rho}_i$ weights for the matching incompatibility of its pairwise attributes ($\hat{\mathbf{p}}_i^k$ or $\mathring{\mathbf{p}}_i^l$).

Clearly, a good node i in G should match better with positive graphs than negative graphs. In general, therefore, $\hat{\mathbf{u}}_i^k$ and $\hat{\mathbf{p}}_i^k$ should be less than $\mathring{\mathbf{u}}_i^l$ and $\mathring{\mathbf{p}}_i^l$, respectively. Thus, the weights of node i (*i.e.* $\boldsymbol{\mu}_i$ and $\boldsymbol{\rho}_i$) should be negative, according to (4.30). Therefore, I use the following metric to evaluate the reliability of node $i \in V$.

$$R_i = -\sqrt{n_v}(\mathbf{1}^T\boldsymbol{\mu}_i^{(j)} + \mathbf{1}^T\boldsymbol{\rho}_i^{(j)})/\|\mathcal{W}\| \tag{4.31}$$

Iterative structural modification is performed as follows. During each iteration, I eliminate the node with the lowest reliability from G, *i.e.*, $i^* = \text{argmin}_{i\in V} R_i$, thereby updating the structure of G as the induced subgraph. Please see [7] for further details.

4.5 Experiments

I perform two experiments to evaluate the object matching and object recognition performance of the trained category models. I test the models trained using the strategies proposed in Sections 4.4.1, 4.4.3, and 4.4.4. Six competing methods are designed to train the category models, including semi-supervised and supervised approaches.

4.5.1 Data

As introduced in Chapter 7, I use the RGB-D image dataset [30], published as a standard Kinect RGB-D object dataset oriented to graph matching[25]. I use five categories in the dataset that are large enough for training and testing, *i.e. notebook PC*, *drink box*, *basket*, *bucket*, and *bicycle*. These categories contain 33, 36, 36, 67, and 92 scenes, respectively, which are enough for training and testing.

4.5.2 Competing methods

As discussed in Section 4.2, the scenario of learning from large and cluttered RGB-D or RGB scenes proposes several high-level requirements for competing methods. Generally speaking, only four styles of methods (*i.e.* object discovery[26], one-shot learning, multi-image co-segmentation, and learning graph matching) can be used to learn from large and cluttered images where the target objects are not manually aligned. However, except for learning graph matching, these styles are usually hampered by large texture variations (*e.g.* texture variations in the *drink box* category) in the informally captured training images, because they rely heavily on the texture consistency between objects and some related studies have even directly used "bag-of-words" models. I therefore limited my comparison to approaches of learning graph matching that make good use of structural information. Both supervised and semi-supervised methods for learning graph matching are used as competing methods.

---

[25] This is one of the largest RGB-D object datasets, containing about 900 objects in complex environments. `http://sites.google.com/site/quanshizhang`. It is produced mainly for the testing of graph matching.

[26] I regard the semi-supervised learning [47] and active learning [54] from results of image search engines as an extension of the object discovery, here.

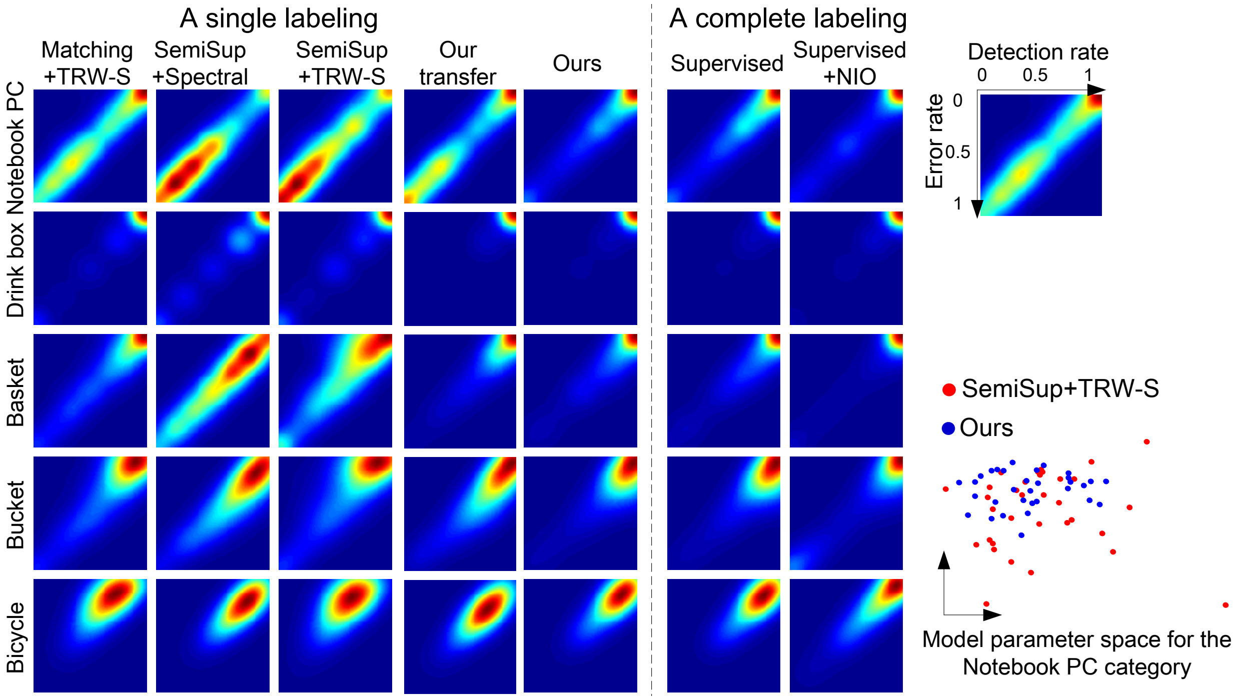


Figure 4.8: Bias problems for different approaches. (Main part) Distribution of the detection (matching) rate and error rate for the learned models. In cross validation, a set of models is learned for each category by setting different initial labeling, and the object matching of each target RGB image based on any of these models produces a pair of detection and error rates (Please see Section 4.5.3 for details). The sub-figure shows the distribution of the detection and error rates. The existence of low detection rates and high error rates is mainly caused by the biased models. Note that, despite being based on a single labeling, *Mine* outperforms other semi-supervised methods, even approaches the performance of supervised methods with complete labeling. Except for the learning of *notebook PC* models, *My transfer* does not show an obvious bias in model learning. (Bottom right) Model parameters ($\mathbf{w}$) of the *notebook PC* category projected onto a 2D space. Different points indicate the values for $\mathbf{w}$ learned from a different initial labeling. Note that the values for $\mathbf{w}$ learned by *Mine* are more convergent, whereas the outliers provided by *SemiSup+TRW-S* indicate biased models.

I use *Mine*, *My transfer*, and *Mine+GraphRefine* to denote the model learning method based on RGB-D images (proposed in Section 4.4.1), the method based on knowledge

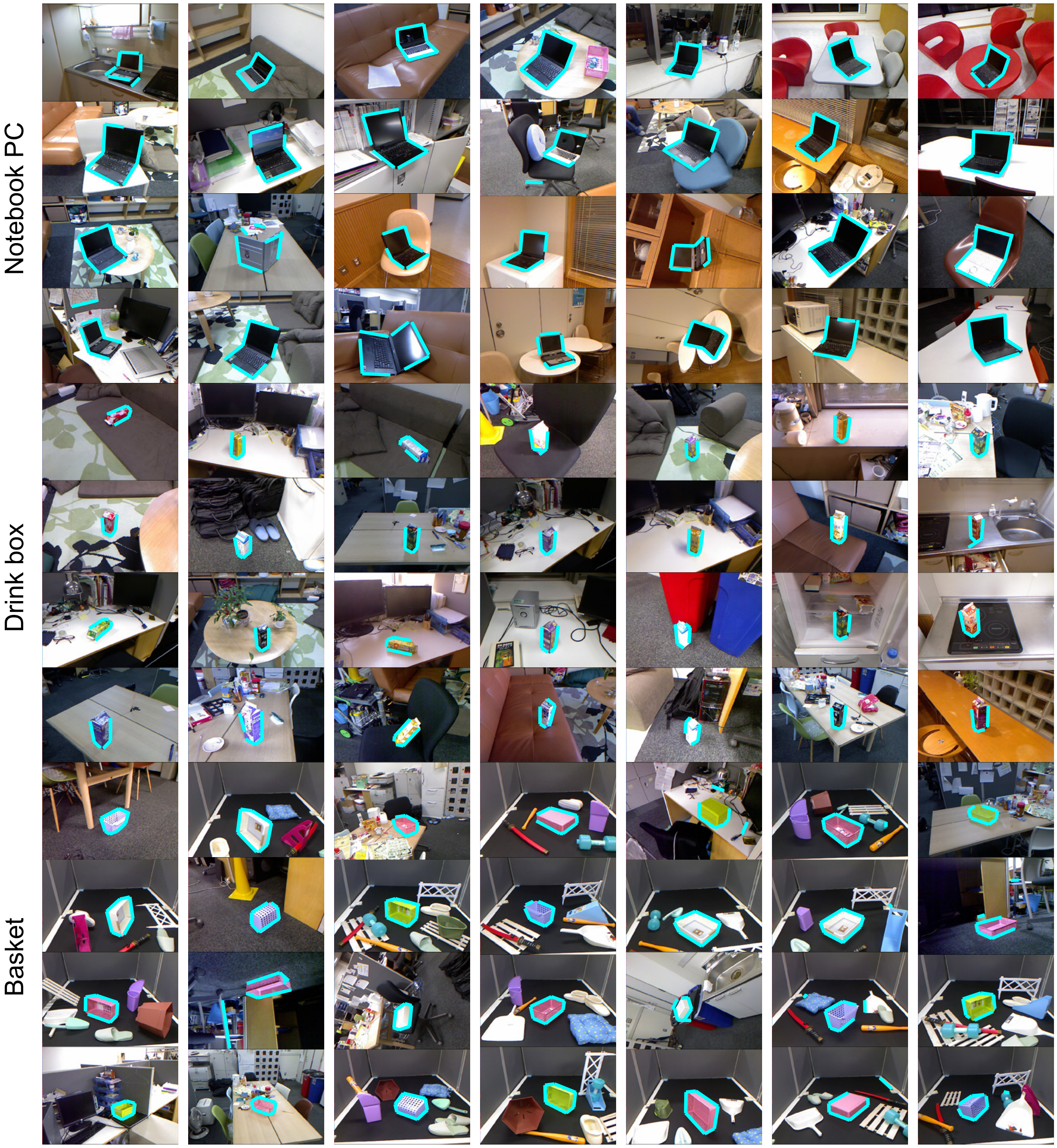


Figure 4.9: Object matching results.

transfer (proposed in Section 4.4.3), and the model learning combined with initial labeling refinement (proposed in Section 4.4.4), respectively. Then, I introduce five compet-

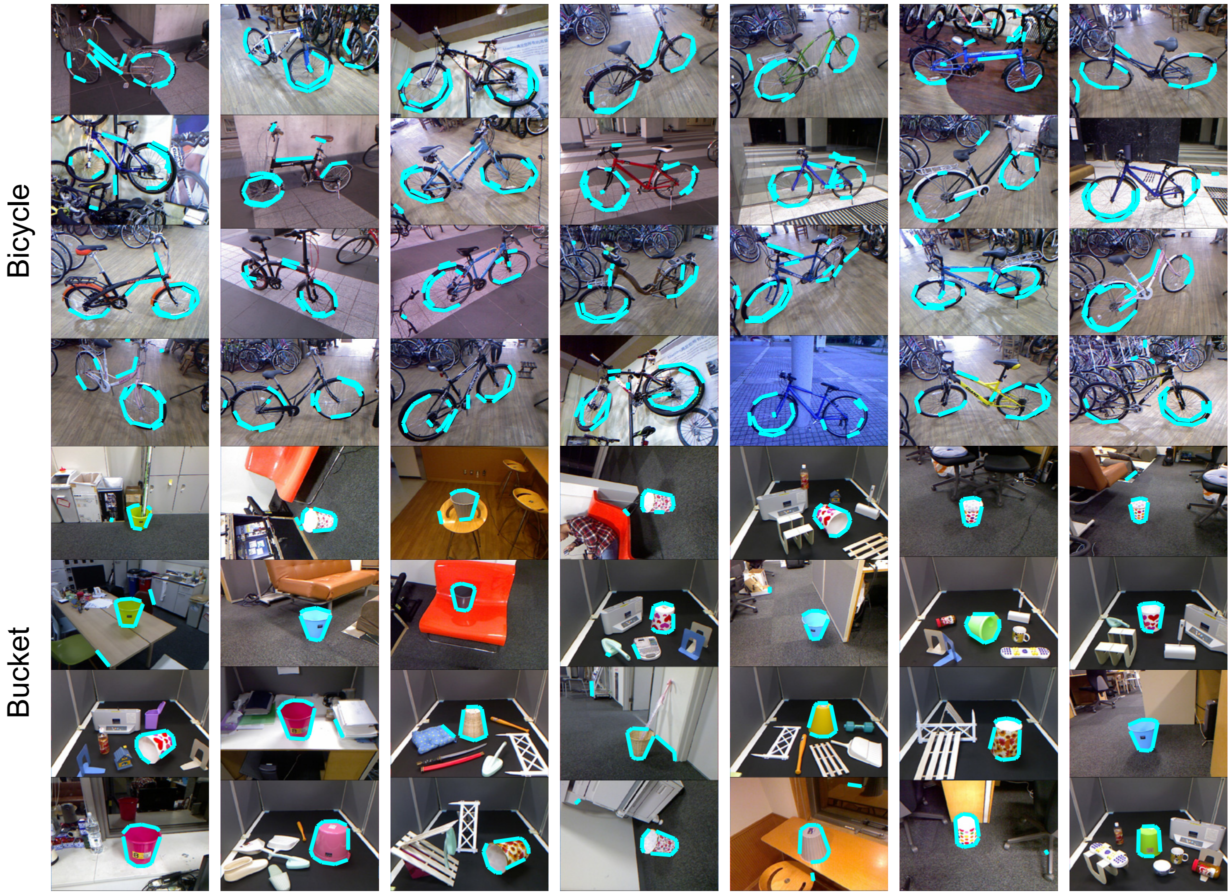


Figure 4.10: Object matching results.

ing methods. Pure graph matching based on TRW-S [28] (without learning) is used as my benchmark method, denoted by *Matching+TRW-S*. I then use four typical methods of semi-supervised and supervised learning of graph matching as the competing methods. The two methods based on [5] learn graph matching in an unsupervised manner, using spectral techniques [33] and TRW-S [28], respectively, to solve graph matching. I call these methods "semi-supervised" approaches, because they require a pre-provided template graph as in my methods. Thus, I refer to them as *SemiSup+Spectral* and *SemiSup+TRW-S*. In particular, I also apply my previous work for semi-supervised learning of graph structures from RGB images [7], denoted by *GraphRefine*, as a competing method for *Mine+GraphRefine*. The remaining two methods perform supervised learning of the proposed category model.

Supervised is an extension of [5] that uses the ground truth, instead of 3D matching assignments, to guide model learning. Whereas, *Supervised+NIO* chooses nonlinear inverse optimization (NIO) to learn models, as introduced in [6, 36].

In particular, I simply set $\mathbf{w} = \mathbf{1}$ for *Matching+TRW-S*, because *Matching+TRW-S* does not learn the matching weights. I transform my semi-supervised learning into supervised learning, *Supervised*. In *Supervised*, $a_{jj'}$ in (4.15) is redefined as 1 or 0 depending on whether the matched node j' in the image is a true detection of the target object part. Finally, in *Supervised+NIO*, model parameters and attributes are trained using the NIO [36]. The NIO minimizes the compatibility gap between the true assignments and predicted assignments, as given by

$$\arg\min_{\mathbf{f}_{ij},\mathbf{w}} \sum_{k=1}^{N} \left\{ \max_{\mathbf{y}} \mathcal{C}(\mathbf{f}_{ij}, \mathbf{w}, \mathbf{y} | G^{'(k)}) - \mathcal{C}(\mathbf{f}_{ij}, \mathbf{w}, \mathbf{y}_{truth}^{(k)} | G^{'(k)}) \right\} \tag{4.32}$$

where $\mathcal{C}(\cdot)$ denotes the matching compatibility in (4.1), $G^{'(k)}$ denotes the k-th target graph for matching, and $\mathbf{y}_{truth}^{(k)}$ represents the matching ground truth of $G^{'(k)}$.

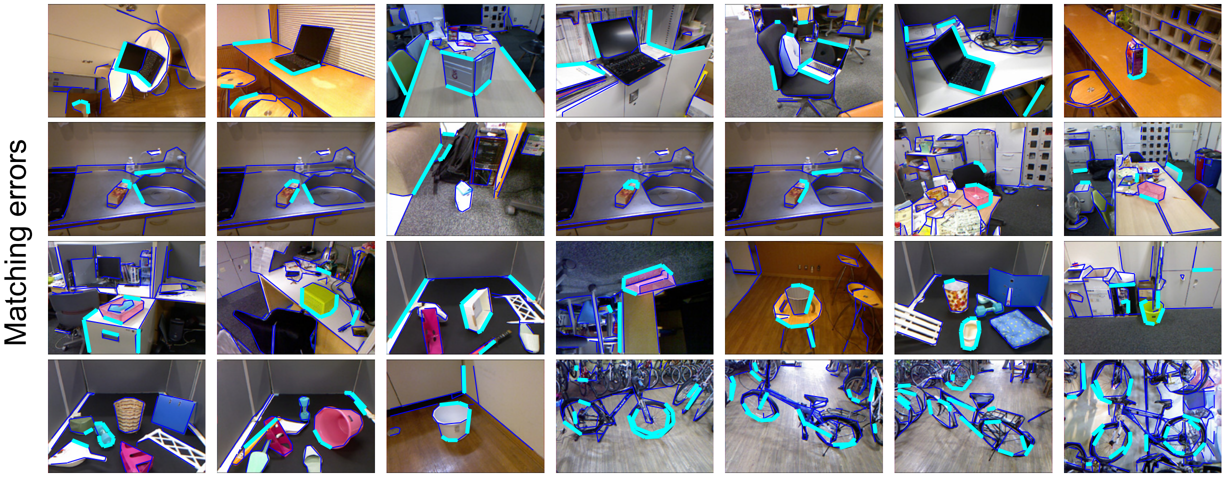


Figure 4.11: Error results of object matching. Detection results are shown in cyan, and other edge segments in images are shown in dark blue.

4.5.3 Experiment 1: learning from a single labeled object

This experiment tests the performance of "model learning from a single labeled object" using different methods, including *Mine*, *My transfer*, and the other five competing methods. Note that method performance is sensitive to the initial labels; therefore, I uniformly apply the object labels used in [29] to all the competing methods to ensure a fair comparison. All the object labeling is published in [30].

Evaluation 1: object matching

This experiment tests object matching between a trained model and RGB images known to contain the target objects. The object matching performance is evaluated via cross validation. For the method *Mine*, I use each RGB-D image to start a single model learning process, and I thus obtain a set of models for each category for cross validation. The training process for each model is performed as follows. Given an RGB-D image, I use the labeled object in this image as the template graph. I then randomly select 2/3 and 1/3 of the remaining RGB-D images in this category for training and testing, respectively. Note that only the RGB channels of the RGB-D images are used for testing.

The cross-validation of the other competing methods is similar to that for *Mine*. The only difference is that I only use RGB channels in the RGB-D images for training. In particular, for the method *My transfer*, I transfer the knowledge extracted from the existing models in any four of the five categories to guide the model learning of the fifth category. The existing models are trained using the method *Mine*, and I use the RGB channels of the RGB-D images in the fifth category as the training images for model learning.

According to convention, I calculate the average detection (matching) rate (ADR) as a measurement of matching performance[27] [5, 4, 7, 29]. Each detection rate is defined as $DR = N^T/\min\{N^{model}, N^{target}\}$, indicating the proportion of model nodes that are correctly matched to the target object. N^T denotes the number of nodes in the model that

---

[27] Note that the matching rate in [7] is defined as $MR = N^T/N^{model}$, which has a slight difference to the detection rate used in [29] and this work.

		NP	DB	BA	BU	BI
Average detection rate ↑	Matching+TRW-S	56.17	84.84	74.12	73.43	67.62
	SemiSup+Spectral	41.89	78.01	61.69	74.60	76.28
	SemiSup+TRW-S	43.57	77.95	62.87	69.47	61.37
	My transfer	51.25	97.97	87.92	80.38	72.45
	Mine	**74.24**	**98.03**	**88.04**	**87.99**	**81.56**
	Supervised	73.13	**98.61**	87.21	**87.69**	80.98
	Supervised+NIO	**78.11**	95.54	**92.05**	79.08	**82.68**
	Object matching in RGB-D images					
	3D matching	93.68	90.57	90.35	96.12	93.87
Average error rate ↓	Matching+TRW-S	42.82	14.93	24.67	22.76	**18.31**
	SemiSup+Spectral	58.16	21.99	39.11	30.17	23.72
	SemiSup+TRW-S	54.43	20.89	30.83	22.41	20.40
	My transfer	48.80	2.03	13.45	24.52	27.55
	Mine	**25.98**	**1.97**	**13.22**	**17.77**	18.44
	Supervised	27.08	**1.39**	14.15	**18.04**	19.02
	Supervised+NIO	**22.13**	4.46	**9.42**	25.55	**17.31**
	Object matching in RGB-D images					
	3D matching	6.49	9.43	11.00	10.57	4.58

Table 4.1: Object matching performance in Experiment 1 "learning from a single labeled object". This table lists the average detection and error rates of different categories, where NP, DB, BA, BU, and BI indicate the *notebook PC*, *drink box*, *basket*, *bucket*, and *bicycle* categories. The models are trained using initial labeling of [29]. *Mine* starts with a minimal labeling, but achieves nearly as well as supervised methods that require to manually label all the training samples. In general, *My transfer* outperforms the other semi-supervised methods.

are matched to the target object; N^{model} and N^{target} indicate the total number of segments in the model and the target object. The ADR represents the average of individual detection rates across all matching results produced by the trained models of a category.

Given the inclusion of *none* as a matching choice, I also care about the average error rate (AER). The error rate of an individual matching result is $ER = \frac{N^B}{N^{model}}$, where N^B is the number of nodes matched to the background. Note that $N^T + N^B \leq N^{model}$, as some model nodes may be matched to *none*. Similarly, AER is the average of ER.

Fig. 4.9 and Fig. 4.10 illustrate object matching using the category models learned by *Mine*. Error matching cases are shown in Fig. 4.11. Table 4.1 shows the quantitative results, and demonstrates that the performance of 3D matching from RGB-D images is good enough to guide the learning of category models. As shown in Fig. 4.8, conventional semi-supervised methods suffer greatly from accumulated bias, whereas the strategy of learning from RGB-D images and that of learning based on knowledge transfer make *Mine* and *My transfer* outperform the other semi-supervised methods. *Mine* approaches the performance of supervised methods with complete labeling, and except for the *notebook PC* category, *My transfer* does not show an obvious bias in model learning. Note that for some categories, *Mine* exhibits a better performance than *Supervised*. This is because for *Supervised*, the labeling of ground truth only determines a subset of line segments in target scenes as target objects, whereas *Mine* uses 3D matching to provide the exact matching assignments that more fit the target model. Moreover, in [5], the regression of the prototype model is not sensitive to outliers in training samples; therefore, *Mine* performs even better than *3D matching* for the *drink box* category.

Evaluation 2: object recognition

I combine each competing method with the three object recognition approaches proposed in Section 4.4.2 to compare their object recognition performance. To test each model trained with each competing method, I use the testing RGB images from Experiment 1 as the positive images. I then randomly select the same number of negative images from the image sets in the other four categories.

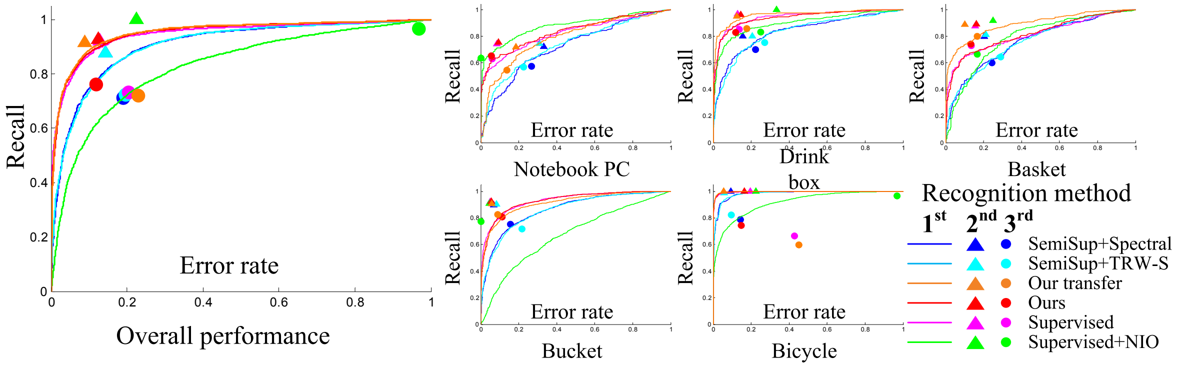


Figure 4.12: Comparison of object recognition performance. I use the curve, triangle, and circle to represent the first, second, and third methods for object recognition. These methods are used to evaluate the trained category models. The left subgraph shows the overall recognition performance based on the three recognition methods proposed in Section 4.4.2, and the other subgraphs show recognition performance for different categories. Generally speaking, *Mine* and *My transfer* approach the performance of *Supervised* that require complete labeling, and exhibit superior performance to other competing methods.

In the same manner as object matching, object recognition is evaluated by cross validation. I generate a curve of the recall and error rate by setting different values of threshold υ to assess the matching-compatibility-based object recognition. All of the models trained for a category[28] are used in the cross validation. Given a specific value of υ, each model produces a set of object recognition results, and I can obtain a high number of recognition results from all the models. I compute the curve based on all of these results to represent the overall recognition performance. Next, to assess the [85, 38]-based and [7]-based object recognition methods, I compute the average recall and error rates for all of the trained models.

Object recognition performances of different competing methods are compared in Fig. 4.12. Fig.4.13 shows the object recognition performance of *Mine* for each category. Except for the *bicycle* category, *Mine* and *My transfer* outperform the other semi-supervised meth-

---

[28] Please see Experiment 1 for the training of these models in the same category.

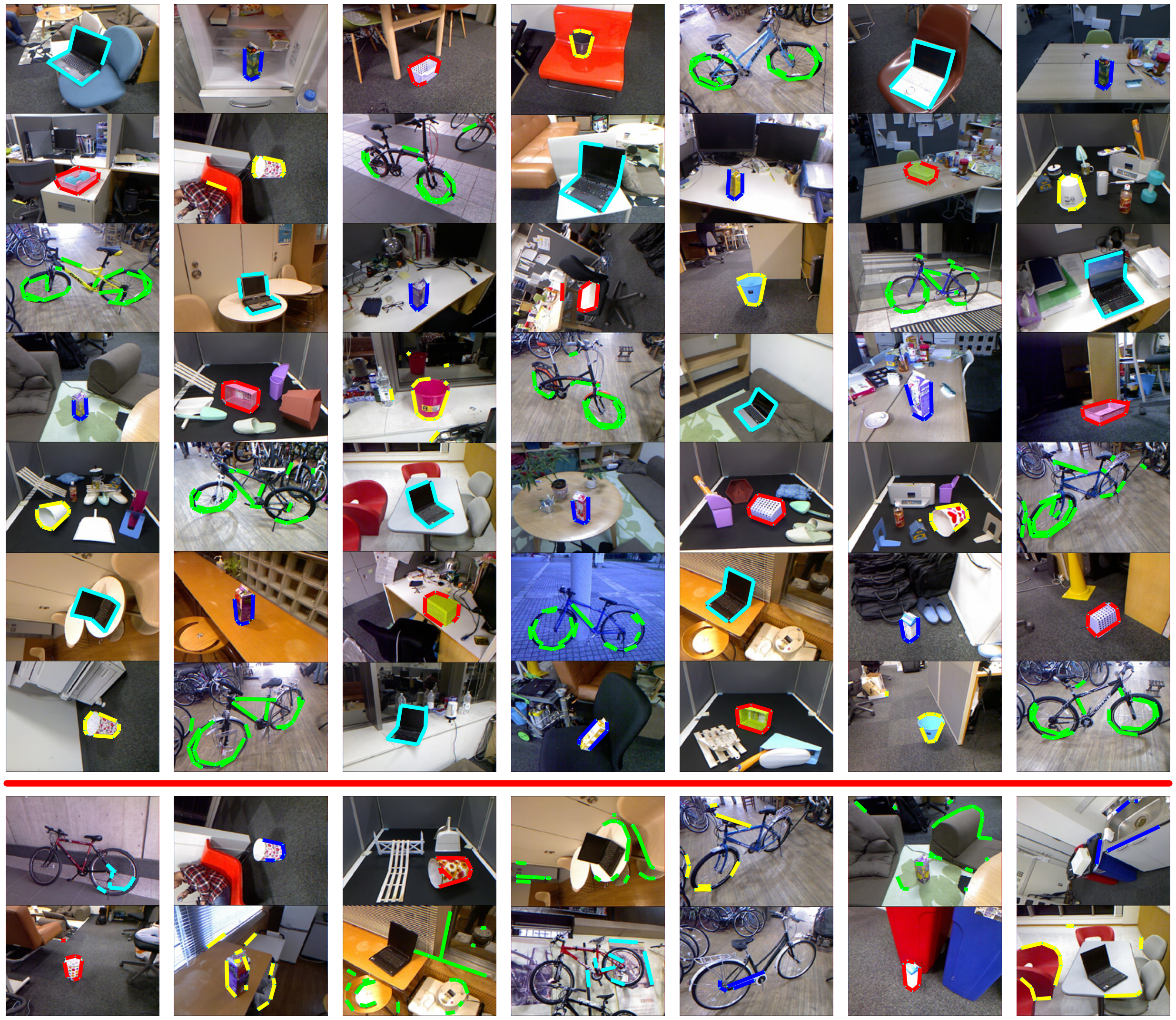


Figure 4.13: Recognition performance for each category. Different colors indicate different categories: *notebook PC* (cyan), *drink box* (blue), *basket* (red), *bucket* (yellow), and *bicycle* (green). The last two rows show some error results.

ods, and even approaches the performance of the *Supervised* method. For the learning of the *bicycle* models, *SemiSup+Spectral* and *SemiSup+TRW-S* are usually biased to some specific parts of the bicycle, but these parts are often more distinguishing than the entire shape of the bicycle. Therefore, *SemiSup+Spectral* and *SemiSup+TRW-S* exhibit better performance for the *bicycle* category. Based on the NIO, *Supervised+NIO* is prone to forcing “bad” parts of objects to be well matched, rather than achieving a high value of total matching compatibility. Thus, *Supervised+NIO* has poor performance for complex objects (with many edge segments) *i.e. bucket* and *bicycle*.

4.5.4 Experiment 2: Learning from refined object labeling

In this experiment, I also test the performance of “model learning from inaccurately labeled objects”. I compare the proposed *Mine+GraphRefine* with the original method *Mine* and other competing methods.

I apply the “inaccurate object labeling” used in [7], which is also published in [30], in the experiment. I evaluate the object matching performance of models trained using different methods. The object-matching performance is evaluated by cross validation. The design of the cross validation is the same as the one described in Section 4.5.3. I select each image to start a single model-learning process, and use the same distribution of the training and testing images as before. The only difference is that the initial template is not accurately labeled in the selected image as it was in Section 4.5.3.

In addition to ADR and AER, I also use the average matching rate (AMR) described in [7] as an evaluation metric. Each matching rate is defined as $MR = N^T/N^{model}$, where N^T and N^{model} denote the model node number that match to the target object and the total model node number, respectively. In the same manner as ADR and AER, the AMR is the average of MR.

I set the proposed method *Mine+GraphRefine* to refine the initial labeled objects in the *notebook PC*, *drink box*, *basket*, and *bucket* categories to yield five, four, five, and nine parts (nodes), respectively. These settings are also applied to *GraphRefine*, which learns the graph structure from RGB images, thereby ensuring a fair comparison.

Table 4.2 provides the quantitative comparisons, demonstrating the superior performance of the proposed *Mine+GraphRefine*. Generally speaking, compared to results in Table 4.1, the results in Table 4.2 show higher error rates and lower detection rates, due to the inaccurate manual labeling. Note that in Table 4.2, the *bucket* category shows

Average (detection↑ / error↓ / matching↑) rate		
	Notebook PC	Drink box
Matching+TRW-S	57.6 / 44.0 / 56.1	69.1 / 43.9 / 56.2
SemiSup+Spectral	47.8 / 53.6 / 46.4	61.1 / 50.5 / 49.5
SemiSup+TRW-S	50.2 / 51.2 / 48.8	63.3 / 48.8 / 51.2
GraphRefine	54.5 / 43.8 / 54.5	83.4 / 16.6 / 83.4
Mine	57.9 / 43.8 / 56.1	71.8 / 42.0 / 57.8
Mine+GraphRefine	**68.6** / **31.2** / **68.6**	**89.0** / **11.0** / **89.0**
Supervised	54.5 / 47.1 / 52.9	65.0 / 47.3 / 52.7
	Basket	Bucket
Matching+TRW-S	67.3 / 44.7 / 55.3	72.7 / 42.0 / 58.0
SemiSup+Spectral	61.2 / 49.8 / 50.3	69.5 / 45.3 / 54.7
SemiSup+TRW-S	62.1 / 49.0 / 51.0	71.5 / 43.6 / 56.4
GraphRefine	81.8 / 18.0 / 81.8	77.4 / 28.1 / 70.4
Mine	75.8 / 38.3 / 61.7	**78.9** / 37.4 / 62.5
Mine+GraphRefine	**86.3** / **13.7** / **86.3**	78.6 / **28.4** / **71.6**
Supervised	62.9 / 48.3 / 51.7	72.0 / 43.2 / 56.8

Table 4.2: Object matching performance in Experiment 2 “learning from refined object labeling”. This table lists detection, error, and matching rates. Category models trained are trained using initial labeling of [7].

lower detection rates for *Matching+TRW-S* and *SemiSup+TRW-S*, they also exhibit significantly lower error rates. Only for the *Notebook PC* category, I see *Matching+TRW-S*, *SemiSup+Spectral*, and *SemiSup+TRW-S* exhibiting better performance in Table 4.2 than in Table 4.1. However, because *Matching+TRW-S* directly matches initially labeled templates without training, its superior performance for the *Notebook PC* category in Table 4.2 demonstrates that compared to subjectively well labeled notebook PCs in [29], subjectively inaccurate labeling of PCs provide by [7] can actually produce a better fit to the target objects in cluttered images, in some cases.

Then, I limit my comparison to the results given in Table 4.2. Compared to *Mine*, *Mine+GraphRefine* clearly benefits from the [7] technique, thereby exhibiting more robustness under inaccurate manual labeling.

4.6 Conclusions

In this study, I developed a method for category model learning that proceeds from a single labeled object to a number of informally captured RGB-D images. I used the RGB-D information to guide the training of the category model, which was applied to ordinary RGB images. The trained category models were also used to provide knowledge to guide the model training for new categories, if users could only obtain RGB images in these categories. All of these techniques were designed to facilitate efficient model learning. The minimization of labeling saves considerable human labor during the construction of the model base. Both the depth information obtained from RGB-D images and the knowledge transferred from other category models can guide the learning framework to overcome the problem of bias. The effectiveness of the proposed method has been demonstrated in various experiments.

In this study, I only used the trained model to match a single object in an image for testing. However, graph matching can easily be extended to matching multiple objects in the same image in real applications. I can use the model directly to match a new object, if I remove all the parts (nodes) of the previously matched object from the target image (graph). Recognizing that most objects in daily use have standard and recognizable shapes, but may have a variety of tones and textures, I designed my category model to focus on structural information, namely, object edge segments. This design makes the model robust across texture variations. However, it is difficult for my category model to describe largely occluded objects or objects with highly deformable or irregular shapes, such as natural scenes and animals. Therefore, I need to develop more kinds of graphical models with new local and pairwise attributes to represent these irregular and deformable objects in the future work.

Chapter 5

Extended Application 1: Model Mining for Single-View 3D Reconstruction

The studies in Chapters 2, 3, and 4 focus on the methods automatically discovering common objects hidden in ubiquitous images and simultaneously training the category model for object detection. I call these methods general platforms for visual mining, because the mined category models and the automatically labeled objects can be directly used to guide other model-learning tasks for a diversity of applications, such as object tracking, segmentation, and 3D reconstruction.

Therefore, in the following chapters, I will introduce some extended applications based on the platform of visual mining. In particular, this chapter introduces a method to mine category models for single-view 3D reconstruction from ubiquitous RGB-D images. Actually, this study is similar to the one in Chapter 4. They both mine models from ubiquitous RGB-D images but apply the mined models to ordinary RGB images.

In this research, I aim to extract the knowledge about how to use a single 2D appearance of an object (*i.e.* the 2D spatial relationship between different object parts, here) to estimate its pose (*i.e.* the object rotation) and 3D structure deformation. In addition, the 2D object appearance may suffer from intra-category texture variations, this knowledge should also be robust to such variations.

Obviously, this knowledge cannot be pre-provided, as objects in different categories have different structures and thus correspond to different category-specific regulations for

3D reconstruction. Therefore, I propose a method that directly mines a category model from ubiquitous RGB-D images, so as to encode such knowledge for each category. This method only requires the labeling of a single object to start the model mining for each category. Thus, the low cost of human labeling makes it plausible to achieve the ultimate goal of this study, *i.e.* building a comprehensive model base that can provide the 3D reconstruction knowledge for each arbitrary daily-use object in each arbitrary RGB image.

The rest of this chapter is organized as follows. The introduction and discussion of related work are presented in Sections 5.1 and 5.2. The whole algorithm consists of the mining of category detectors and the training of category models, which are presented in Section 5.3 and Section 5.4, respectively. Then, Section 5.5 presents the experiments and the overall study is summarized in Section 5.6.

5.1 Introduction

3D reconstruction is a classical area in the field of computer vision, but 3D reconstruction from a single image still has great challenges. In this chapter, I re-consider the problem of single-view 3D reconstruction in terms of two CV areas: category modeling and knowledge mining from big visual data (see Fig. 5.1).

Category modeling & task output: For the bottleneck in single-view 3D reconstruction, *i.e.* the reconstruction of objects with irregular[29] structures, I have to return to the concept of "category modeling".

Therefore, the objective is to train a model to detect objects in the target category in large images, while simultaneously projecting their 2D shapes into the 3D space at the pixel level. The category model encodes the knowledge of how intra-category structure deformation and object rotations affect 2D object shapes.

---

[29] The word "irregular" is used to indicate that my approach focuses on general object categories without setting strong assumptions for object shapes. In contrast, Cheeger-set-based methods focus on ball-like surfaces, and perspective-based methods require vanishing points.

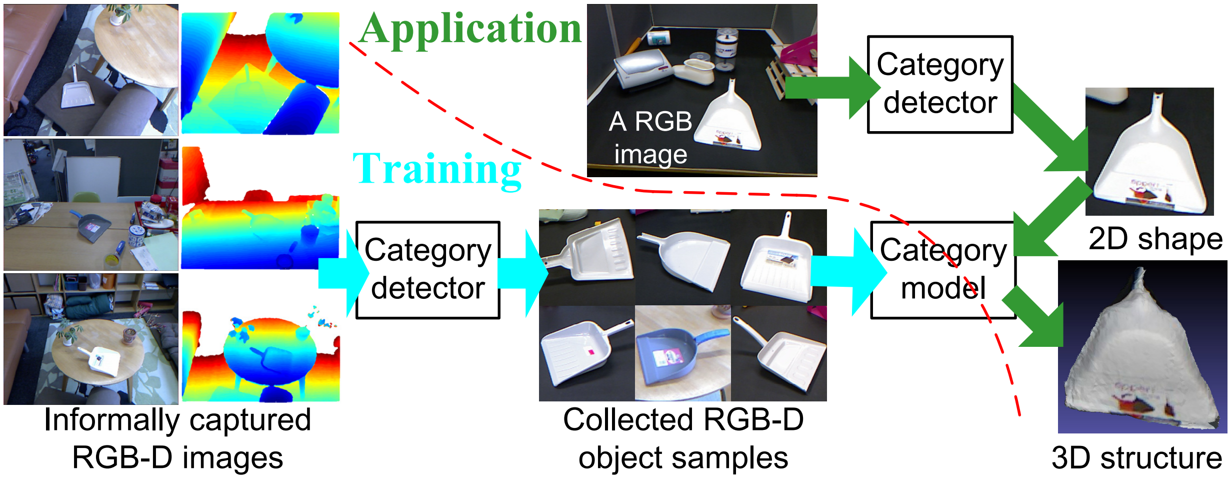


Figure 5.1: Flowchart for training (cyan arrows) and testing (green arrows) processes. Single-view 3D reconstruction is hampered by objects with irregular shapes. Therefore, for objects in each category, I aim to learn a specific category model for 3D reconstruction. I propose to directly train the category model from "informally captured" RGB-D images (where target objects are NOT aligned) to save human labeling, thereby ensuring high efficiency in model learning. I first mine the category detector from informally captured RGB-D images to collect RGB-D objects, and then use these object samples to train the category model. For testing, the category detector and category model are used in sequence to localize the target object and estimate its 3D structure.

Mining from big visual data & task input: *Another bottleneck lies in efficiently learning the category-specific knowledge of 3D reconstruction for a huge number of categories.* Ideally, I would need to train a model for each object category in daily use, so as to construct a knowledge base to provide a 3D reconstruction service for arbitrary RGB images. Therefore, I hope to learn from big visual data[30] to avoid the labor of manually preparing training samples (*e.g.* well built 3D models), and thus ensure a high learning

---

[30] Actually, learning from "big visual data" is still in the early stages of development, and its scope is quite extensive. It usually involves two aspects. This first is learning from a large amount of web data, such as the widely used deep learning [42]. The second is learning under challenging conditions of "informally captured" images that contain small objects and are ubiquitous in everyday life, as is the case in my method. [29] also provides detailed description of the second aspect.

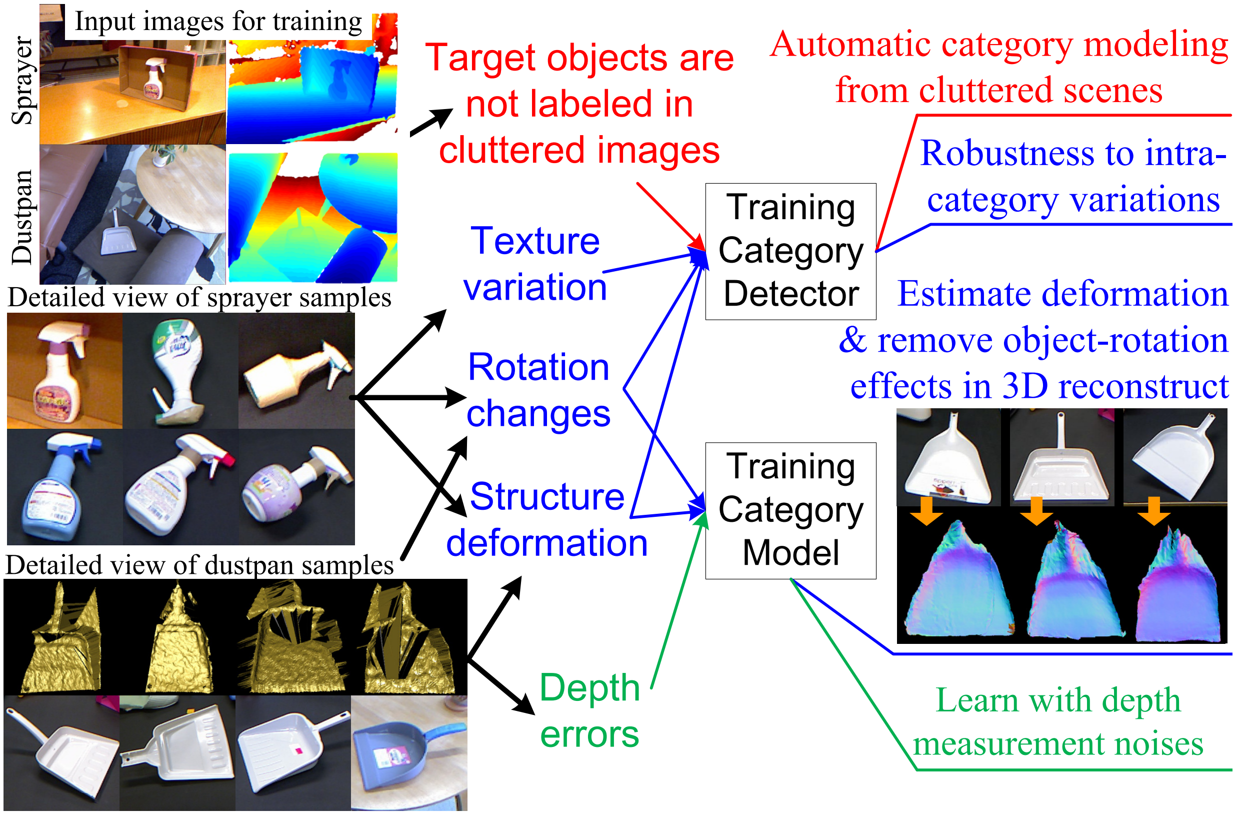

Input images for training
Sprayer
Dustpan
Detailed view of sprayer samples
Detailed view of dustpan samples
Target objects are not labeled in cluttered images
Texture variation
Rotation changes
Structure deformation
Depth errors
Training Category Detector
Training Category Model
Automatic category modeling from cluttered scenes
Robustness to intra-category variations
Estimate deformation & remove object-rotation effects in 3D reconstruct
Learn with depth measurement noises

Figure 5.2: Overview of challenges.

efficiency.

In this chapter, I train category models directly from informally captured and "unaligned" RGB-D images. I use the phrase "informally captured" to describe loose requirements for ubiquitous[30] images. They are typical of what can be directly collected by search engines (Figures 5.1 and 5.2).

The informally captured images are not manually aligned, and they consist of small objects that are randomly positioned. In particular, these daily-use objects are usually designed with texture variations, various rotations, and some structure deformation for commercial purposes. Technically speaking, these images can be loosely regarded as a kind of big visual data[30].

Challenge analysis: Just like the training, the RGB images for testing 3D reconstruction are also such kind of informally captured images in this research. The use of informally captured images raises a number of challenges, as shown in Fig. 5.2.

For application purposes, the category model should have the ability to detect small target objects with various textures, rotations, and scales in large RGB images before 3D reconstruction. Similarly, the learning process should also be able to handle these challenges to collect RGB-D objects in cluttered RGB-D images. Moreover, this detection has to be accurate to the part level, since 3D reconstruction uses the parts' 2D positions to estimate the pose and 3D structure of an object.

Then, I analyze the challenges of learning 3D reconstruction knowledge from the collected RGB-D objects[31]. I need to simultaneously consider the following three factors that affect the correspondence between the 2D and 3D structure of the object. 1) Rotation changes. This is the primary factor. 2) Deformation of 3D object structures. 3) Depth errors near object edges in training samples (measured by the Kinect)

Proposed method: As shown in Fig. 5.1, the training of category detectors is the first step. The category detectors collect[31] RGB-D/RGB object samples from the informally captured RGB-D/RGB images, which serves as the basis for further training/testing of 3D reconstruction. I use unsupervised category modeling based on graph matching [10] to train the category detector (specified in [29]). This algorithm automatically discovers a set of relatively distinguishing parts (namely key parts) of objects for each category. Thus, the category detector achieves a part-level detection. I then train the category model to use the parts' 2D positions to determine the object's pose and 3D structure.

Unlike conventional methods that use well built 3D models to provide multi-view object appearances and prior regularity of 3D structure deformation, my approach needs to learn this knowledge in an unsupervised manner. In other words, I need to train the category model to simultaneously remove the effects of viewpoint changes, and estimate the 3D structure deformation using the 2D shape of an object. The category model consists of

---

[31] In my study, these objects are collected from RGB images (or RGB dimensions of the RGB-D images) using the trained category detectors. The 3D object structure is projected into the 2D shape afterwards.

a number of estimators, each corresponding to a specific position inside the 2D shape. These estimators use the spatial relationship between key object parts to estimate the corresponding 3D coordinates of their positions. I connect neighboring estimators to form a continuous Markov random field (MRF) and thus train these estimators.

The contributions of this chapter can be summarized as follows. I regard 3D reconstruction from a single image as a category modeling problem to overcome the difficulties in the reconstruction of irregular[29] shapes. To ensure a high learning efficiency and a wide application, this is the first attempt to learn category models from informally captured RGB-D images, and then apply the category model back to the informally captured RGB images for 3D reconstruction. I explore a new set of challenges raised by my choice of training and testing data, and provide a solution.

5.2 Related work

3D reconstruction is a large area in the field of computer vision. However, in this chapter, I limit my discussion to 3D reconstruction from a single image. Many methods for single-view 3D reconstruction have strong assumptions for the image environment. For example, "shapes from shading" methods [87, 88, 89, 90] had special requirements for lighting conditions and textures. A large number of studies use the perspective principle for 3D reconstruction. They were typically based on the vanishing points in images, and therefore assumed that these images contained enough cues to extract vanishing points [91, 92]. Such methods were mainly applied to large-scale indoor and urban environments. In addition, some studies used the ground/vertical assumption to assist in vanishing point extraction [93, 94, 95, 96]. Saxena *et al.* [97] proposed to learn an MRF for 3D reconstruction in large-scale environments. Fouhey *et al.* [62] proposed to learn 3D primitives from RGB-D images for single-view reconstruction of indoor environment.

A number of methods relied on the assumption that the object structure in question had smooth surfaces. They usually extracted object-level structures from locally captured images that contained no perspective cues. For example, the Cheeger Set approach [98] assumed that the target object had ball-like surfaces. Given the object contour and the object volume, it computed the 3D structure with the minimum surface area. However, such assumptions for object structure are usually not valid for the reconstruction of irregular[29] shapes. Therefore, many methods [99, 100, 101, 90] combined human interactions with these assumptions to guide the 3D reconstruction.

By and large, the lack of cues for the estimation of irregular[29] object structures is the main challenge for single-view 3D reconstruction. From this perspective, my research is related to the example-based methods [102, 103, 104]. They required a comprehensive dataset of 3D object samples that were captured in all poses from different viewpoints. The 3D structure knowledge of a target object was extracted via a large number of comparisons between the object and the samples in the dataset. Chen *et al.* [105, 106] learned 3D reconstruction knowledge from well built 3D object models. These 3D object models provided knowledge on structure deformation and the multi-view appearance. In this way, they learned to match the 2D shape of a given object to the 3D object models, and thus estimate the object's viewpoint, pose, and 3D structure.

In contrast, I apply much less supervision to idealize the concept of automatic "category modeling" in terms of training. Learning from informally captured RGB-D images without manual alignment saves great labor for preparation for 3D object models or a large 3D-example dataset. The challenges of object detection, rotation changes, and the estimation of 3D structure deformation are all involved in the training process. The trained category model is then applied back to the informally captured RGB images for 3D reconstruction, without example comparison.

5.3 Training of category detectors

Category detectors are not only trained from, but also applied to informally captured and unaligned images. The category models for 3D reconstruction are designed on the basis of category detectors. In my study, I use the graphical model defined in [29] as the category detector, considering the need for robustness to shape deformations and rotations in the informally captured images. I then apply unsupervised learning for graph matching [10] to mine the category detectors. In reality, I simply use the RGB channels of RGB-D images for training. The detector is trained to encode the pattern for the common objects in all the images. A set of key object parts are discovered to form the pattern for a category (see Fig. 5.3(c)).

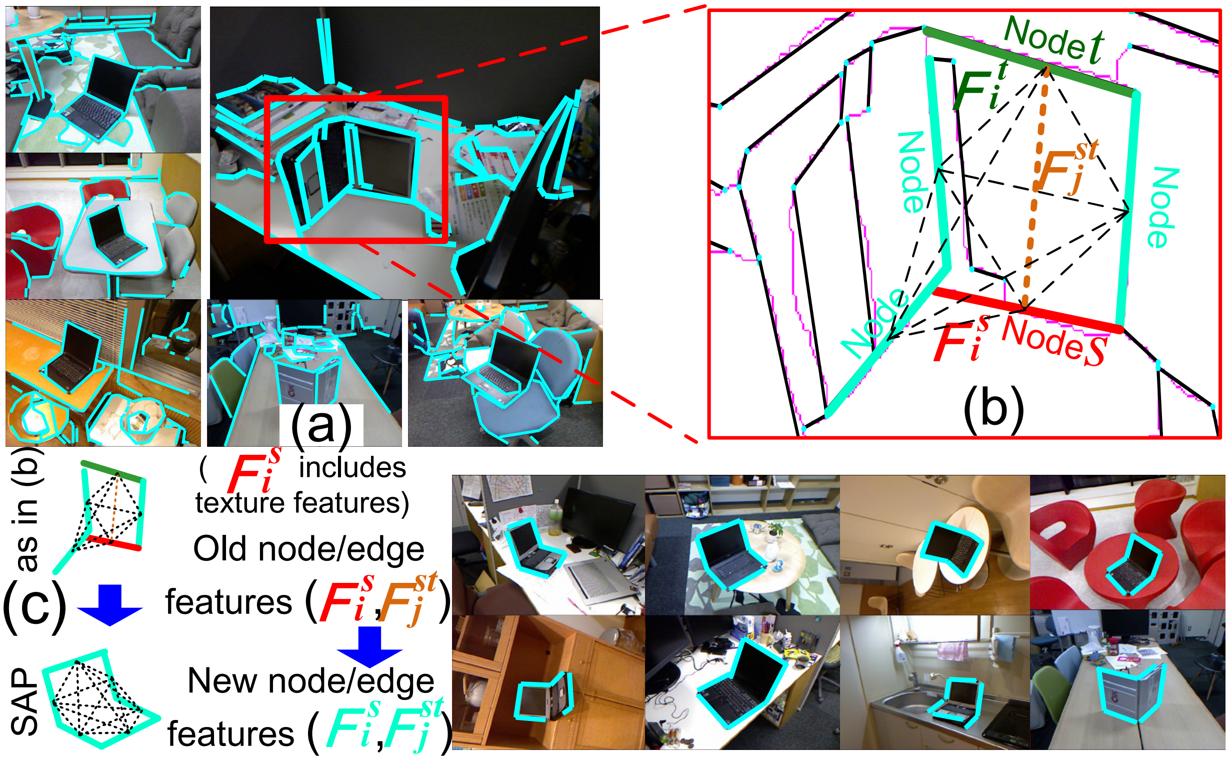

(a)
Node t
F_i^t
F_j^{st}
Node
Node
Node
F_i^s Node s
(b)
as in (b)
(c)
SAP
(F_i^s includes texture features)
Old node/edge features (F_i^s,F_j^{st})
New node/edge features (F_i^s,F_j^{st})

Figure 5.3: Object detectors. (a) Graph-based image representation. Object edge segments (cyan) form the graph nodes. (b) Notation for the graphical model. The initial graphical model consists of five nodes (colored lines) and edges (dotted lines) between them. (c) Learning the graphical model (category detector). The algorithm modifies (i) the initial graphical model into (ii) the SAP (G). The node set and node/edge features are modified so that 1) the node number (size) of G is maximized and 2) all the node/edge features represent common patterns in all the graphs.

Category detectors: As shown in Fig. 5.3(a), the image is represented as a complete attributed relational graph (ARG). In Fig. 5.3(b), continuous object edges (magenta) are discretized into line segments (black), and these line segments form the graph nodes. Thus the objects within it are represented as sub-graphs.

The category detector is a graphical model in [29], which will be automatically trained to be the soft attributed pattern (SAP) among the target sub-graphs (or objects) in different images. Let graph G with node set V denote the model. G is described by a set of node and edge features. Let $\{\mathcal{F}_i^s\}$ and $\{\mathcal{F}_j^{st}\}$ ($i = 1, 2, ..., N_P$, $j = 1, 2, ..., N_Q$) denote the i-th node feature for node s and the j-th edge feature for edge (s, t), respectively. Here, N_P and N_Q indicate the feature numbers for each node and edge, respectively. $\mathbf{F}_V = \{\mathcal{F}_i^s\} \bigcup \{\mathcal{F}_j^{st}\}$, $\forall s, t \in V$. Note that $\mathcal{F}_i^s$ includes the texture features on the terminals of lines segments. The category detector thus also contains the textural knowledge. (Please see [29] for details of the feature settings.)

Object detection: Object detection is achieved via graph matching. Given a target RGB image represented by graph G' with a node set V', graph matching is performed to compute the matching assignments between G and G'. Let $\mathbf{x}$ denote the label set, and let the label $x_s \in \mathbf{x}$ denote the matched node in G' for node s in G. Graph matching is formulated as the minimization of the following energy function w.r.t $\mathbf{x}$.

$$E(\mathbf{x}|\mathbf{F}_V,\mathbf{F}_{V'})=\sum_{s\in V}P_s(x_s|\mathbf{F}_V,\mathbf{F}_{V'})+\sum_{s,t\in V,s\neq t}Q_{st}(x_s,x_t|\mathbf{F}_V,\mathbf{F}_{V'}) \tag{5.1}$$

This is a typical formula for graph matching, where the functions $P_s(\cdot|\cdot,\cdot)$ and $Q_{st}(\cdot,\cdot|\cdot,\cdot)$ measure matching penalties (feature dissimilarity) between the two matched nodes and edges. These are generally defined using squared differences. *E.g.* $P_s(x_s|\mathbf{F}_V,\mathbf{F}_{V'}) = \mathbf{w}^T[\|\mathcal{F}_1^s - \mathcal{F}_1^{x_s^k}\|^2, ..., \|\mathcal{F}_{N_P}^s - \mathcal{F}_{N_P}^{x_s^k}\|^2]$. (Please see [10] for details.)

Learning graph matching with SAPs: As shown in Fig. 5.3(c), [10] proposed an algorithm for mining SAPs. The SAP is a fuzzy attributed pattern, which describes the common sub-graphs in a set of ARGs, in which node and edge features have considerable variations. In other words, the category detector (*i.e.* the SAP) represents a set of key object parts (nodes) that frequently appear in training images of the entire category. The category detector is trained taking into account the robustness to texture variations and structure deformations.

To start the training process, this method requires people only to label the sub-graph of a single object for initializing the graphical model G. [10] is designed to modify G from the specific shape of the labeled object into the SAP among all graphs. Given a set of N training images, each image is denoted by $\{G'_k\}$ $(k = 1, 2, ..., N)$ with node set V'_k. $\mathbf{F}_{V'_k}$ are the attribute sets of G'_k, and $x^k_s \in \mathbf{x}^k \in \mathbf{X}$ indicates the node in V'_k matched by $s \in V$. The "frequency" of node s in G is defined as the average penalty for matching s to all the $\{G'_k\}$, as follows.

$$E_s(\hat{\mathbf{X}},\hat{\mathbf{F}}_V)=\frac{1}{N}\sum_{k=1}^{N}\Big[P_s(\hat{x}^k_s|\hat{\mathbf{F}}_V,\mathbf{F}_{V'_k})+\sum_{t\in V,t\neq s}Q_{st}(\hat{x}^k_s,\hat{x}^k_t|\hat{\mathbf{F}}_V,\mathbf{F}_{V'_k})\Big]$$

If $E_s(\hat{\mathbf{X}}, \hat{\mathbf{F}}_V)$ is less than a given threshold τ, node s is regarded as a frequently appearing part; otherwise not. Hence, the goal of learning graph matching is to train the node set V and attributes $\mathbf{F}_V$ of model G, in such a manner that 1) all nodes of G frequently appear in graphs $\{G'_k\}$ and 2) the size of G is maximized.

$$\begin{aligned} & \max_{V} \|V\| \\ \text{s.t.} \quad & (\hat{\mathbf{F}}_V, \hat{\mathbf{X}}) = \underset{\mathbf{F}_V, \mathbf{X}=\{\mathbf{x}^k\}}{\operatorname{argmin}} \sum\nolimits_{k=1}^{N} E(\mathbf{x}^k|\mathbf{F}_V, \mathbf{F}_{V'_k}); \\ & \forall s \in V, \quad E_s(\hat{\mathbf{X}}, \hat{\mathbf{F}}_V) \leq \tau. \end{aligned} \tag{5.2}$$

[10] proposes an EM framework to solve (5.2).

5.4 Learning 3D reconstruction

I introduce the preparation of training samples, model learning, and the application of 3D reconstruction in the following three subsections.

5.4.1 Object sample collection & pose normalization

The preparation of training samples involves two steps, *i.e.* object sample collection and 3D pose normalization. In the first step, I use the trained category detector to collect RGB-D samples from informally captured RGB-D images for training. Each training sample contains its 2D shape and the 3D coordinates of each 2D pixel within it. The 3D coordinates are measured in the global coordinate system of each RGB-D image, rather than a coordinate system *w.r.t* the object. Therefore, in the second step, I compute the 3D pose of the object, and thereby normalize the 3D coordinates into the object coordinate system.

Object sample collection: I use the category detector to match a set of key object parts (graph nodes) in each image by applying (5.1). However, as mentioned above, the key parts consist of line segments on the object, and do not comprise the entire object body. Therefore, I first recover the entire body area of the object from the detected line segments, before the further learning. Actually, a number of methods for interactive[32] object segmentation are oriented to this application. Nevertheless, in order to make a reliable evaluation of the proposed 3D reconstruction method, I need to simplify the whole system and thereby avoid bringing in uncertainties related to object segmentation. Consequently, given the key line segments of the object, I roughly estimate its body area as the convex hull of the line terminals (see Fig. 5.4(c)).

3D pose normalization: I define centers and 3D poses for the object samples in the RGB-D images, thus constructing a relative coordinate system for each sample, as shown in Fig. 5.4(f,bottom). I then project 3D point clouds of the objects onto these coordinate systems, as the ground truth of their 3D structures.

Fig. 5.4(e) shows the notation. The center of an object is defined as the mean of the 3D coordinates of its key parts $c = \sum_{1 \leq s \leq m} \mathbf{p}_s / m$, where m is the part (node) number and $\mathbf{p}_s$ denotes the center coordinates of part s. I define the orientations of the three orthogonal

---

[32] The key object parts can be labeled as the foreground.

coordinate axes $<\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3>$ as follows.

$$\mathbf{v}_1 = u(\mathbf{v}'_1 + \mathbf{v}'_2),\ \mathbf{v}_2 = u(\mathbf{v}'_1 - \mathbf{v}'_2),\ \mathbf{v}_3 = \mathbf{v}'_1 \times \mathbf{v}'_2 \tag{5.3}$$

where function $u(\cdot)$ normalizes a vector to a unit vector, $\mathbf{v}'_1 = u(\sum_{1\leq s<t\leq m} u(\mathbf{p}_t - \mathbf{p}_s))$, $\mathbf{v}'_2 = u(\sum_{1\leq s<t\leq m}(\mathbf{o}_t \times \mathbf{o}_s))$, and $\mathbf{o}_s$ denotes the 3D orientation of line segment s.

5.4.2 Model learning

I train a category model from RGB-D object samples to estimate the pixel-level 3D reconstruction. Essentially, even if I are given a prior 3D structure for a category, 3D reconstruction from a single image still has great challenges. There are two general hypotheses, *i.e.* object rotations and structure deformations, to interpreting a specific 2D object shape in 3D reconstruction. Each hypothesis can independently estimate the 3D object structure from the 2D shape (*e.g.* computing the rotation or 3D deformation of the prior 3D structure that best fits the contour of the 2D shape).

Obviously, the 3D reconstruction needs to combine the both hypotheses. Unlike [106, 105] using well built 3D object models to provide or train prior regularity of structure deformation, I need to train the category model to simultaneously identify the effects of object rotations and structure deformations from 2D shapes, without prior knowledge. This greatly increases the challenge.

Fig. 5.5 shows the basic design of the category model. The category model handles both the effects and directly projects each 2D pixel into a 3D space. Considering the model's robustness to shape deformation, I use a set of local 2D coordinate systems to simultaneously localize each pixel inside the 2D shape. Each local 2D coordinate system is constructed using each pair of key object parts. For each point in every 2D coordinate system, I train a local regressor to estimate its 3D coordinates from its point features.

The point features are designed to represent the spatial relationship between the point and key objects parts, as well as the 2D shape of these key object parts. Thus, the point features contain sufficient cues for object rotations and structure deformation to guide the

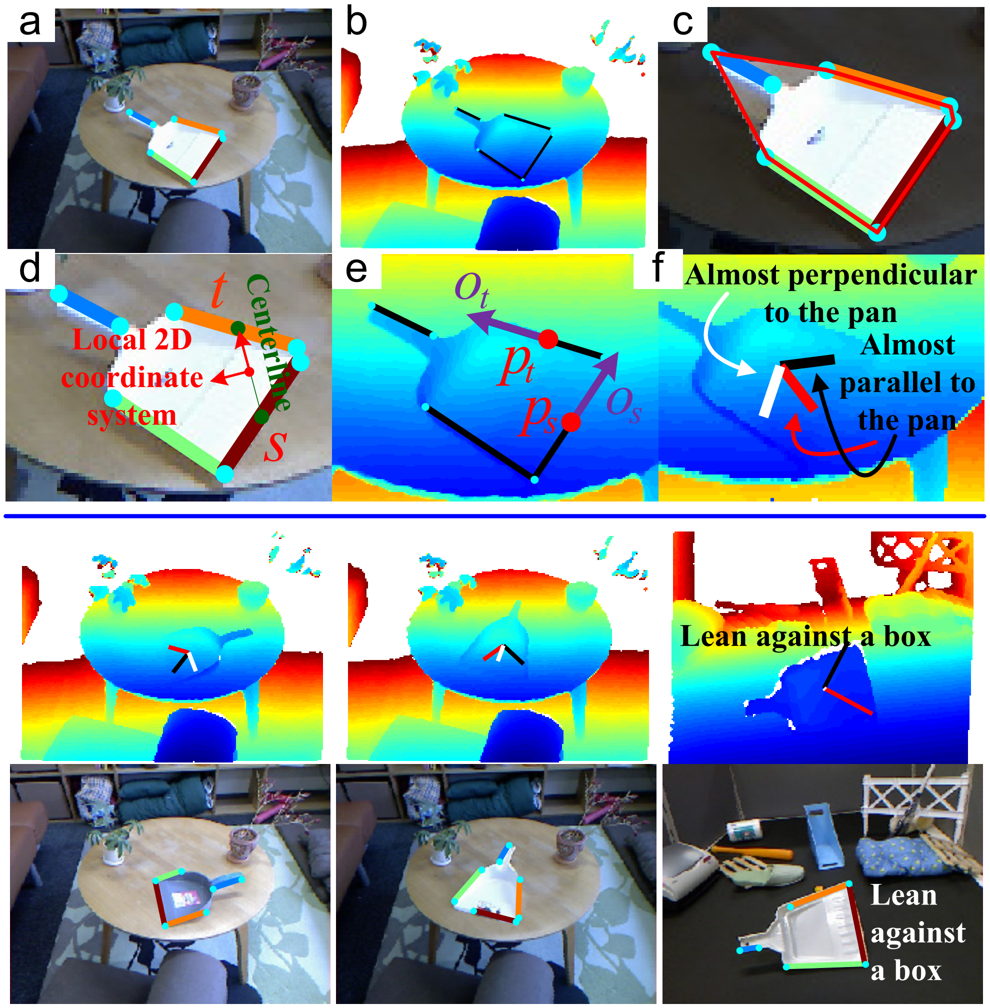

a
b
c
d
t
Centerline
Local 2D coordinate system
s
e
o_t
p_t
p_s
o_s
f
Almost perpendicular to the pan
Almost parallel to the pan
Lean against a box
Lean against a box

Figure 5.4: Some steps in learning 3D reconstruction. (a,b) Visualization of the detected key object parts in the RGB and depth image. (c) Recovery of the object body area. (d) Notation for the local 2D coordinate system. (e) Notation for determining the local 3D coordinate system. (f) Three axes of the local 3D coordinate system. (bottom) Local 3D coordinate systems of three objects.

estimation of the 3D coordinates of each 2D point. For point $\mathbf{p}$ in the local 2D coordinate system *w.r.t* key parts s and t of a given object sample, its point features are defined as $\theta_{st}^{\mathbf{p}} = [\vartheta_1, \vartheta_2, \vartheta_3, \vartheta_4, \vartheta_5, 1]^T$. ϑ_1 and ϑ_2 denote the lengths of s and t, respectively, and ϑ_3 denotes the angle between s and t. ϑ_4 and ϑ_5 measure the distance between $\mathbf{p}$ and the line

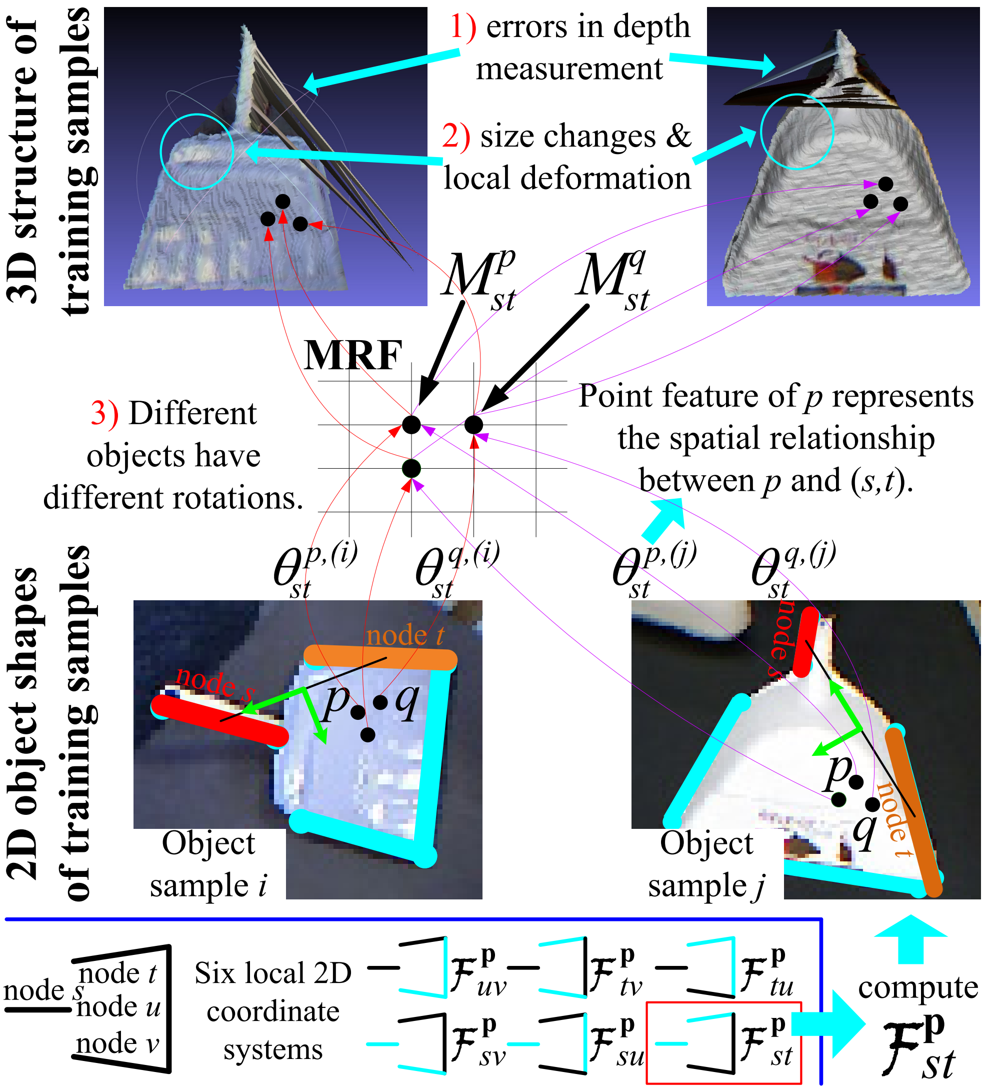

3D structure of training samples
1) errors in depth measurement
2) size changes & local deformation
M^p_{st}
M^q_{st}
MRF
3) Different objects have different rotations.
Point feature of p represents the spatial relationship between p and (s,t).
$\theta^{p,(i)}_{st}$
$\theta^{q,(i)}_{st}$
$\theta^{p,(j)}_{st}$
$\theta^{q,(j)}_{st}$
2D object shapes of training samples
node t
node s
p
q
Object sample i
Object sample j
node s
node t
node u
node v
Six local 2D coordinate systems
$\mathcal{F}^{\mathbf{p}}_{uv}$
$\mathcal{F}^{\mathbf{p}}_{tv}$
$\mathcal{F}^{\mathbf{p}}_{tu}$
$\mathcal{F}^{\mathbf{p}}_{sv}$
$\mathcal{F}^{\mathbf{p}}_{su}$
$\mathcal{F}^{\mathbf{p}}_{st}$
compute $\mathcal{F}^{\mathbf{p}}_{st}$

Figure 5.5: Learning 3D reconstruction. The category model uses different local 2D coordinate systems (bottom) to localize the 2D points, and estimates the 3D structure in these coordinate systems. Given a specific 2D coordinate system *w.r.t* each pair of key parts s and t, I generate a MRF to train the parameter $\mathbf{M}^{\mathbf{p}}_{st}$ for the local regressor $\mathcal{F}^{\mathbf{p}}_{st}(\cdot)$ in each different point $\mathbf{p}$. $\mathcal{F}^{\mathbf{p}}_{st}(\cdot)$ uses the point feature $\theta^{\mathbf{p},(i)}_{st}$ in object sample i to estimate 3D coordinates of $\mathbf{p}$, thus overcoming i's specific object rotations, structure deformation, and depth errors.

segments s and t, respectively. ϑ_1, ϑ_2, ϑ_4, and ϑ_5 are normalized by the centerline length between s and t (Fig. 5.4(d)).

Then, the local regressor $\mathcal{F}_{st}^{\mathbf{p}}(\cdot)$ is modeled as a linear regression of the 3D coordinates of $\mathbf{p}$ using $\theta_{st}^{\mathbf{p}}$:

$$\mathcal{F}_{st}^{\mathbf{p}}(\theta_{st}^{\mathbf{p}}) = (\theta_{st}^{\mathbf{p}})^T \mathbf{M}_{st}^{\mathbf{p}} \tag{5.4}$$

where matrix $\mathbf{M}_{st}^{\mathbf{p}}$ is the parameter for the function $\mathcal{F}_{st}^{\mathbf{p}}(\cdot)$.

The local regressor $\mathcal{F}_{st}^{\mathbf{p}}(\cdot)$ is the basic element of the category model. The category model is formulated as a linear combination of these local regressors, as follows.

$$\mathcal{Y}(x) = \frac{1}{Z} \sum_{1 \leq s < t \leq m} w_{st}^{\mathbf{p}} \mathcal{F}_{st}^{\mathbf{p}}(\theta_{st}^{\mathbf{p}})|_{\mathbf{p}=\mathbf{p}_{st}(x)} \tag{5.5}$$

where $Z = \sum_{1 \leq s < t \leq m} w_{st}^{\mathbf{p}=\mathbf{p}_{st}(x)}$. x is a pixel inside the 2D object shape, and $\mathcal{Y}(x)$ estimates its 3D coordinates. s and t indicate two key parts of the object; m denotes the key part number. Given the local coordinate system constructed using s and t, function $\mathbf{p}_{st}(x)$ determines the 2D position of x in this coordinate system. Thus, function $\mathcal{F}_{st}^{\mathbf{p}}(\cdot)$ is the local regressor of 3D coordinates for point $\mathbf{p}$ in the coordinate system of s and t, as mentioned above. $w_{st}^{\mathbf{p}}$ measures the reliability of the local regressor $\mathcal{F}_{st}^{\mathbf{p}}(\cdot)$ and is regarded as its weight.

Local 2D coordinate system: As shown in Fig. 5.4(d), given the line segments of each pair of key parts s and t ($1 \leq s < t \leq m$), I generate a local 2D coordinate system. The line segment connecting the center of s to that of t is called the centerline between s and t. The origin of the local coordinate system *w.r.t* s and t is defined as the center of the centerline. I take the centerline's orientation as the orientation of the first coordinate axis, and determine the orientation of the second axis using the right-hand rule. The unit vector of the first axis is normalized using the centerline length, and that of the second axis is normalized using the average segment length of s and t.

Model learning based on a continuous MRF: In the local 2D coordinate system determined by the key parts s and t, the local regressors of the neighboring positions are connected to form an MRF. I generate the following objective function to learn the parameters of the regressors.

$$\underset{\mathbb{M}_{st}}{\operatorname{argmax}} \sum_{\mathbf{p}\in\mathcal{V}} \phi_1(\mathbf{M}_{st}^{\mathbf{p}}) + \sum_{(\mathbf{p},\mathbf{q})\in\mathcal{E}} \phi_2(\mathbf{M}_{st}^{\mathbf{p}}, \mathbf{M}_{st}^{\mathbf{q}}) \tag{5.6}$$

where $\mathbb{M}_{st} = \{\mathbf{M}_{st}^{\mathbf{p}} | \mathbf{p} \in \mathcal{V}\}$; $\mathcal{V}$ and $\mathcal{E}$ denote node and edge sets of the MRF, respectively.

In the MRF, the unary term $\phi_1(\cdot)$ measures the similarities between the ground truth and the 3D coordinates estimated by the regressor for each point $\mathbf{p}$.

$$\phi_1(\mathbf{M}_{st}^{\mathbf{p}}) = \frac{1}{\|S\|} \sum_{i\in S} \exp(-\eta \|\mathcal{F}_{st}^{\mathbf{p}}(\theta_{st}^{\mathbf{p},(i)}) - y_i^{\mathbf{p}}\|) \tag{5.7}$$

where S is the set of training samples. For each object sample i, $\theta_{st}^{\mathbf{p},(i)}$ and $y_i^{\mathbf{p}}$ denote the point feature and true 3D coordinates of $\mathbf{p}$, respectively. η is a scaling parameter.

The pairwise term $\phi_2(\cdot,\cdot)$ is a smoothing term between neighboring points.

$$\begin{aligned} \phi_2(\mathbf{M}_{st}^{\mathbf{p}}, \mathbf{M}_{st}^{\mathbf{q}}) &= -\lambda(\mathbf{M}_{st}^{\mathbf{p}} - \mathbf{M}_{st}^{\mathbf{q}})^2 \exp[-\alpha(U^{\mathbf{p}} + U^{\mathbf{q}})^2] \\ U^{\mathbf{p}} &= \frac{1}{\|S\|} \sum_{i\in S} \max_{(\mathbf{p},\mathbf{p}')\in\mathcal{E}} \|y_i^{\mathbf{p}} - y_i^{\mathbf{p}'}\| \end{aligned} \tag{5.8}$$

where $U^{\mathbf{p}}$ use true 3D structures of the training samples to measure the discontinuity around point $\mathbf{p}$. Object areas with high discontinuity, such as object edges, have low weights for smoothing. λ and α are scaling parameters. I use the MCMC method to solve the continuous MRF.

Finally, when the regressor $\mathcal{F}_{st}^{\mathbf{p}}(\cdot)$ has been trained, the weight for $\mathcal{F}_{st}^{\mathbf{p}}(\cdot)$ in (5.5) is computed as

$$w_{st}^{\mathbf{p}} = \frac{1}{\|S\|} \sum_{i\in S} \exp[-\eta(\mathcal{F}_{st}^{\mathbf{p}}(\theta_{st}^{\mathbf{p},(i)}) - y_i^{\mathbf{p}})^2] \tag{5.9}$$

5.4.3 3D reconstruction based on category models

As shown in Fig. 5.1, I use a trained category model to directly perform 3D reconstruction on testing RGB images. I first use the trained category detector to collect target objects in the RGB images, and simultaneously determine the body area of the objects (see Section 5.4.1). I then use the category model in (5.5) to estimate the 3D coordinates for each pixel inside the object.

5.5 Experiments

5.5.1 Data

As introduced in Chapter 7, I use the category dataset of Kinect RGB-D images [30, 29], which is published as a standard RGB-D object dataset[33] for the learning of graph-matching-based models, such as [29, 7]. These RGB-D images depict cluttered scenes containing objects with different textures and rotations. Four categories—*notebook PC*, *drink box*, *sprayer*, and *dustpan*—in this dataset contain a sufficient number of RGB-D objects for training and are thus chosen in the experiments.

5.5.2 Implementation details

Training of category detectors: I train the SAP (category detector) from a set of large graphs (informally captured images). Parameter τ controls the graph size of the SAP, or in other word, the number of key object parts contained by the category detector. When I set a higher value for τ, I can extract more key parts for a category, but the key parts are less reliable in part detection. To simplify the learning process, I require the SAPs (detectors) to have four nodes (key parts) for all the four categories. I try different values of τ in the training process, until the category detector satisfies this requirement. Hence, I can detect four key parts for each object, and the rotation and deformation of the object are determined by the complex spatial relationship between the four key parts.

Learning 3D reconstruction: As shown in Fig. 5.5, in each local 2D coordinate system, I divide the entire object area comprising of 50×50 grids to identify different 2D points. A local regressor with parameter $\mathbf{M}_{st}^{\mathbf{p}}$ is generated for each grid $\mathbf{p}$, and I thus use the local regressors construct the MRF. For the application of 3D reconstruction, many pixels in the 2D shape are not accurately localized in the grid centers. To achieve accurate pixel-level 3D reconstruction, I interpolate the regressor parameter value for each 2D pixel

---

[33] Compared to other RGB-D datasets *i.e.* [60, 107], this is one of the largest RGB-D object datasets, and reflects challenges of graph matching.

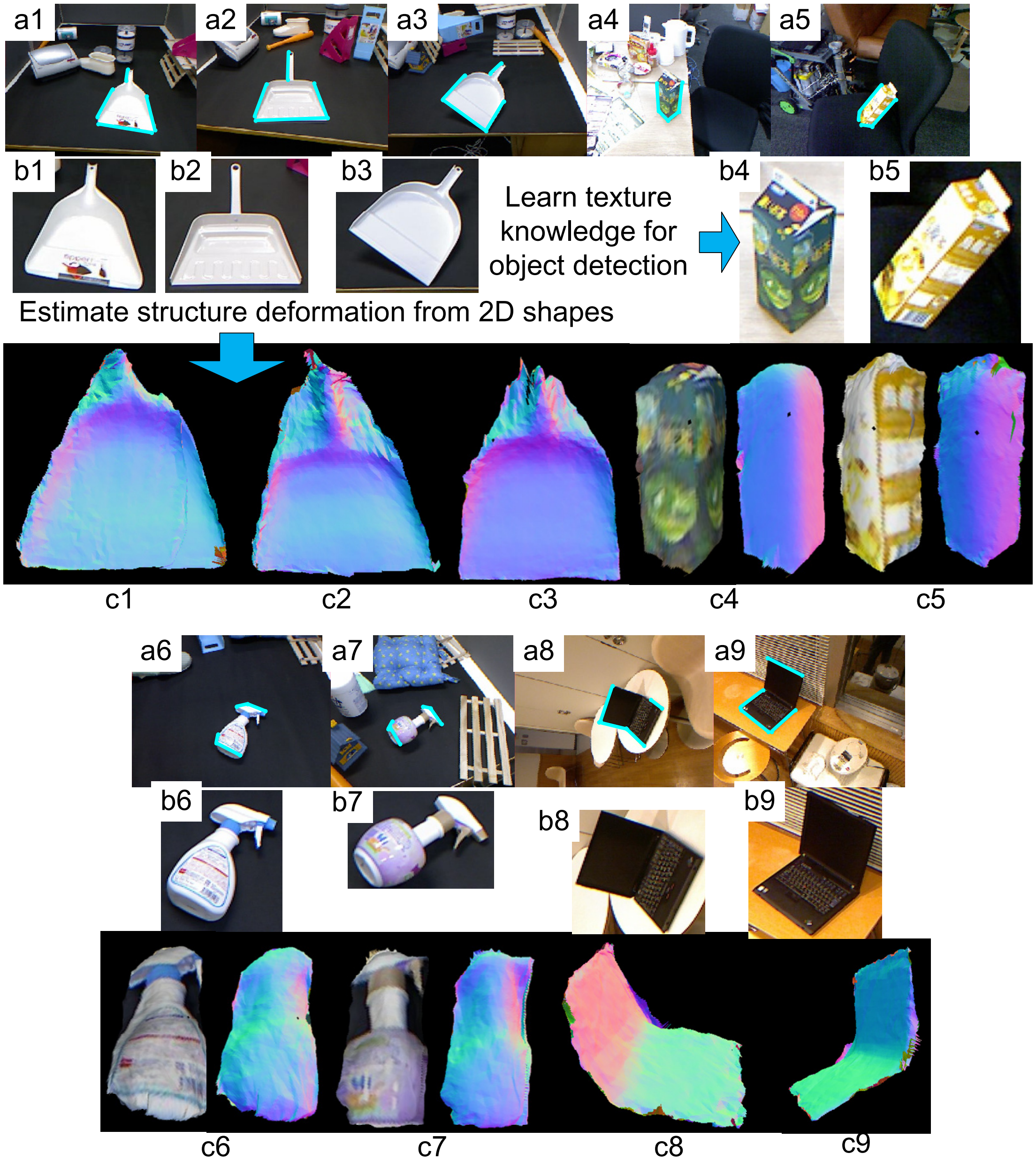

a1
a2
a3
a4
a5
b1
b2
b3
Learn texture knowledge for object detection
b4
b5
Estimate structure deformation from 2D shapes
c1
c2
c3
c4
c5
a6
a7
a8
a9
b6
b7
b8
b9
c6
c7
c8
c9

Figure 5.6: 3D reconstruction performance. (a1–9) Object detection (cyan lines) in RGB images. (b1–9) Detailed view of the target objects. (c1–9) Reconstruction results of the proposed method.

from the trained parameters $\mathbf{M}_{st}^{\mathbf{p}}$ of the nearby grids. I set parameter α as 0.1 for all the categories in model learning, and use different values of λ and η to test the system.

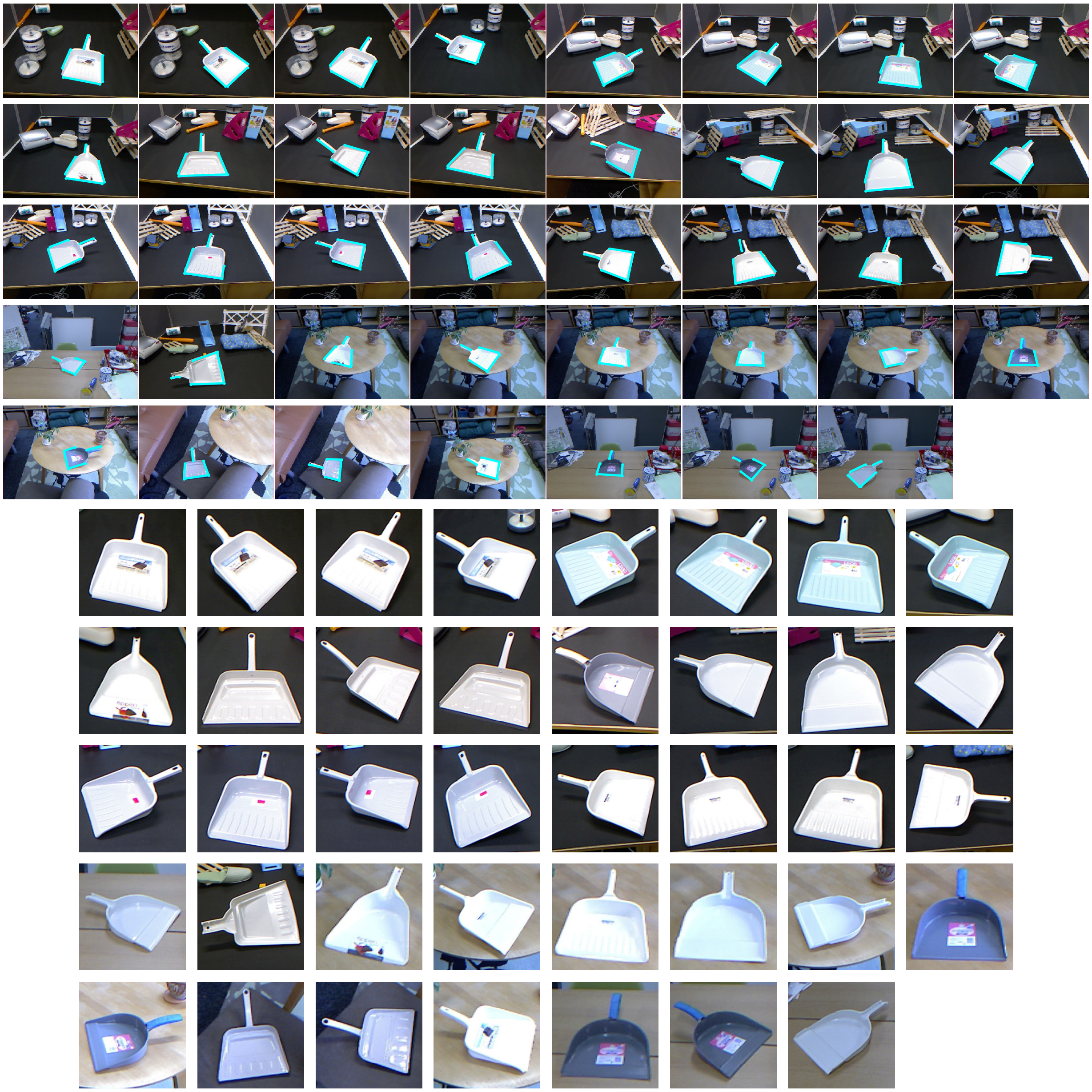


Figure 5.7: Object detection performance based on the mined *dustpan* detector (top) and the detailed view of the detected objects (bottom).

5.5.3 Quantitative comparison

Competing method: I compare my approach with conventional methods for 3D reconstruction from a single image. As discussed in Section 5.2, most related techniques have

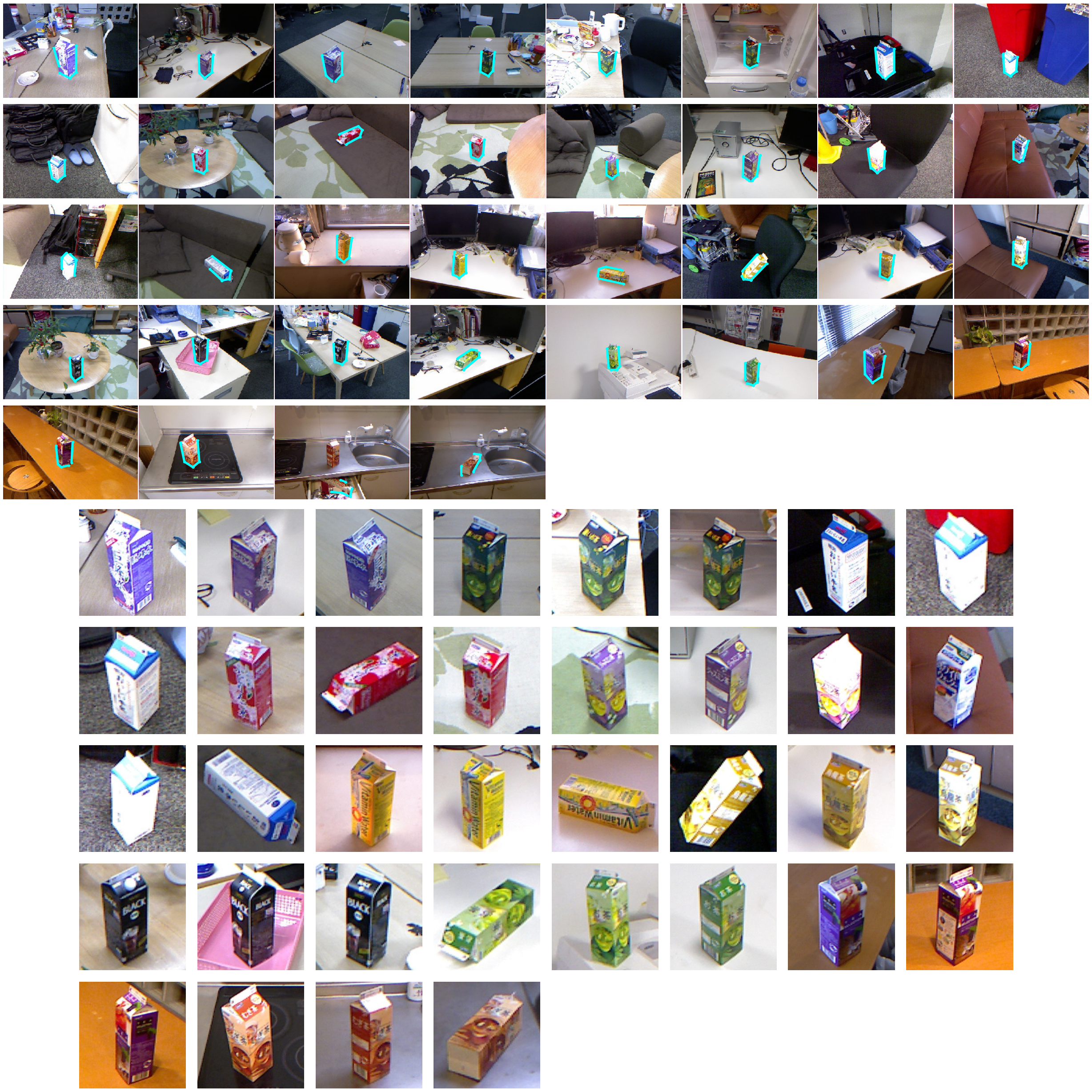


Figure 5.8: Object detection performance based on the mined *drink box* detector and the detailed view of the detected objects (bottom).

their own specific assumptions for the target image and are thus not suitable for 3D reconstruction of irregular[29] shapes. From this viewpoint, example-based 3D reconstruction is close to my method, but conventional techniques are hampered by the use of informal-

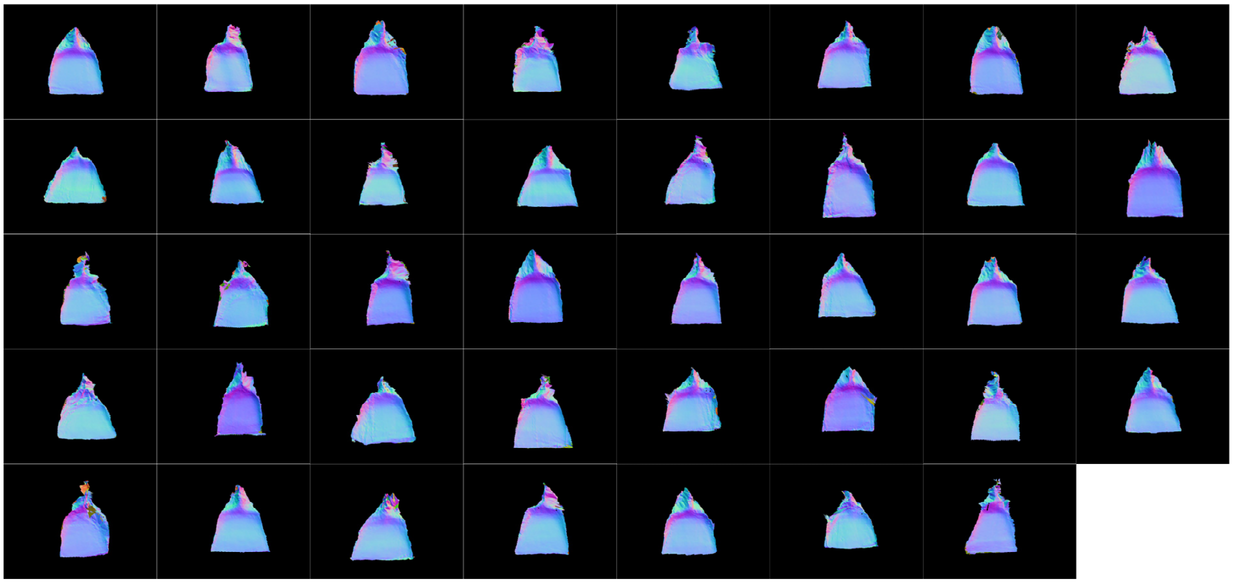


Figure 5.9: 3D reconstruction performance based on the mined *dustpan* model. The results correspond to the detected dustpans in Fig. 5.7. Some objects are not correctly reconstructed because of errors in the detection of object parts.

ly captured RGB-D images for training. Nevertheless, considering the challenges in the training dataset, I design a competing method that achieves a rough idea of the example-based 3D reconstruction, as follows. First, given a RGB image, I use the category detector trained in Section 5.3 to detect the target object in it, as well as its key object parts. I then randomly select an RGB-D object from the training sample set as the example for 3D reconstruction. The 3D coordinate estimation is based on (5.5), just like the proposed method. However, without training, the weight for local regressors ($w_{st}^{\mathbf{p}}$) is set to 1. The local regressors $\mathcal{F}_{st}^{\mathbf{p}}$ is redefined as a constant, *i.e.* the 3D coordinate for point $\mathbf{p}$ on the example, rather than a function in (5.4) *w.r.t.* point features $\theta_{st}^{\mathbf{p}}$.

Pixel-level evaluation: I evaluate the reconstruction errors and surface roughness (non-smoothness) of the proposed approach at the pixel level. Reconstruction errors are widely used to evaluate 3D reconstruction. I manually prepare the ground the truth of the 3D structure for each object using the Kinect measurement. I measure the distance

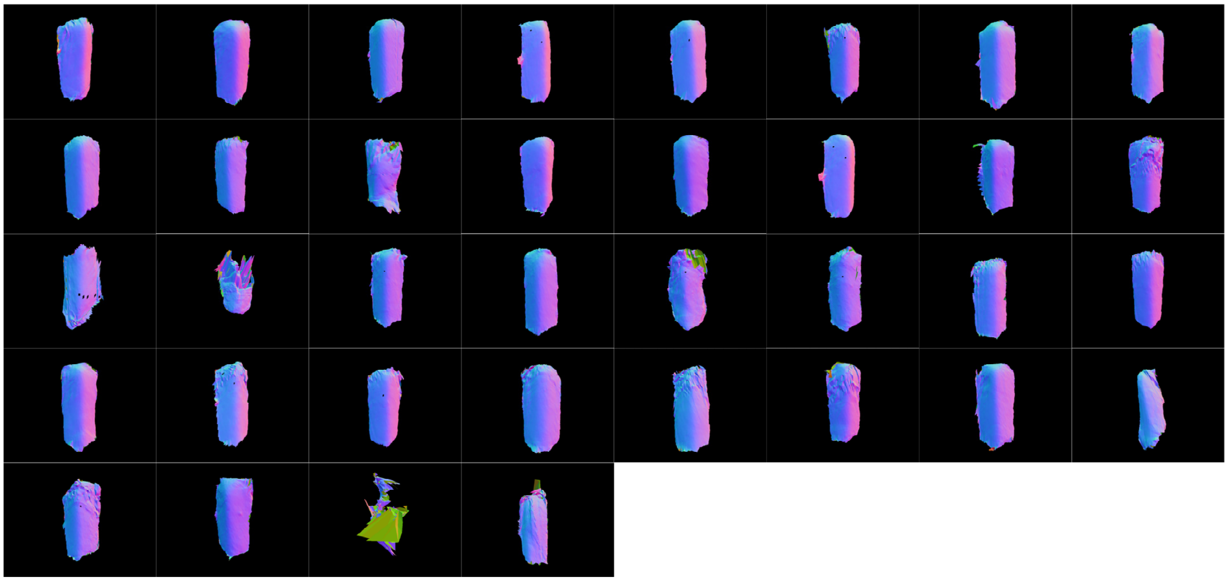


Figure 5.10: 3D reconstruction performance based on the mined *box* model. The results correspond to the detected drink boxes in Fig. 5.8. Some objects are not correctly reconstructed because of errors in the detection of object parts.

between the estimated 3D coordinates of each pixel and its true coordinates, as the pixel reconstruction error. I define the reconstruction error of an object as the average reconstruction error of its constituent pixels. The reconstruction error of a category is defined as the average reconstruction error of all its objects. The other evaluation metric is the surface roughness. The roughness of pixel $\mathbf{p}$ is measured as $r_{\mathbf{p}} = \|\mathcal{Y}^{\mathbf{p}} - \frac{1}{N(\mathbf{p})}\sum_{\mathbf{q}\in N(\mathbf{p})}\mathcal{Y}^{\mathbf{q}}\|$, where pixel $N(\mathbf{p})$ is the set of 4-connectivity neighboring pixels of $\mathbf{p}$, $\mathcal{Y}^{\mathbf{p}}$ denotes the estimated 3D coordinates of pixel $\mathbf{p}$ on the object. Just like the reconstruction error, the object roughness is the average of the pixel roughness, and the surface roughness of a category is defined as the average of object-level roughness.

The evaluation is achieved via cross validation. Just as in [29, 7], I randomly select 2/3 and 1/3 of the RGB-D images in the entire dataset as a pair of sample pools for training and testing, respectively. Each pair of sample pools can train a category model, and provide values of the reconstruction error and surface roughness for the category. I prepare different

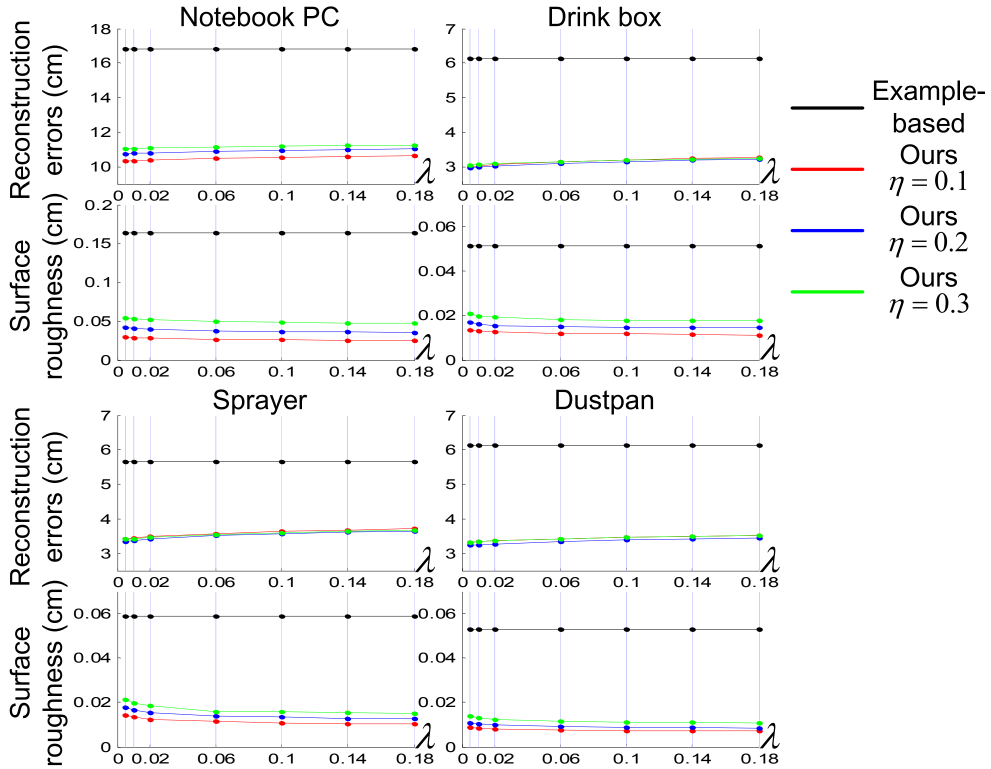


Figure 5.11: 3D reconstruction comparison. The proposed method exhibits smaller reconstruction errors and surface roughness than the example-based method. The learning of category models is not sensitive to different settings of parameters λ and η. λ and η are not involved in the example-based method.

pairs of training and testing pools and compute the average performance by cross validation. Fig. 5.11 and Fig. 5.6 show the comparison of 3D reconstruction performance. Without sufficient learning, the competing method cannot correctly estimate structure deformation for objects, and suffers from the unavoidable errors in Kinect's depth measurement.

Figures 5.7, 5.8, 5.9, and 5.10 show the object detection and 3D reconstruction performance on different RGB images of the *dustpan* and *drink box* categories.

5.6 Conclusions

In this chapter, I proposed to learn a category model for single-view 3D reconstruction, which encoded knowledge for structure deformation, texture variations, and rotation changes. The category model was both trained from and applied to the informally captured images. Experiments demonstrated the effectiveness of the proposed method.

For the training of category detectors, I used various node and edge features. These included texture features on local image patches. Thus, the texture information contributes to the extraction of key object parts. However, textures contribute much less in further 3D reconstruction, as many daily-use objects (*e.g.* the *drink box*) are designed with different textures that are unrelated to object structures for commercial purposes. The spatial relationship between key object parts was, therefore, taken as more reliable cues for 3D reconstruction and used in model learning.

I did not apply object segmentation and thus avoided the uncertainties related to it, so as to simplify the system and enable a reliable evaluation. The category model did not performed so well on object edges as in other object area due to the time-of-the-flight errors and calibration errors between 3D points and RGB images.

Chapter 6

Extended Application 2: Mining Deformable Models of Animals

The algorithm of attributed graph matching proposed in Chapter 2 is a general platform for visual mining from big data, and in Chapter 5, I have designed a method that applies this platform to learning category models for single-view 3D reconstruction. In this chapter, I present another extended application of attributed graph matching, *i.e.* mining deformable models of animals from unlabeled videos.

I design a new type of ARGs to represent frames of a video, where the target objects are not manually aligned. Consequently, the maximal-size soft attributed pattern corresponds to the deformable animal model. I label the target object in the first frame to construct the initial graph template, and then use the attributed graph mining to mine the category model.

6.1 ARG-based deformable model for animals

I design a new type of ARGs to represent the deformable animal in unlabeled video frames, which take SIFT points as graph nodes. The SIFT points are detected all over the frames at different scales. I connect each pair of SIFT points in a frame to construct an ARG. One unary attribute ($N_P = 2$) and four pairwise attributes ($N_Q = 4$) are designed to describe local features and spatial relationship between local parts in video frames.

The notation is illustrated in Fig. 6.1. The only unary attribute is the 128-dimensional descriptor of the SIFT feature of node s, denoted by $\mathcal{F}_1^s = f_s$. Let s_s and o_s denote the scale

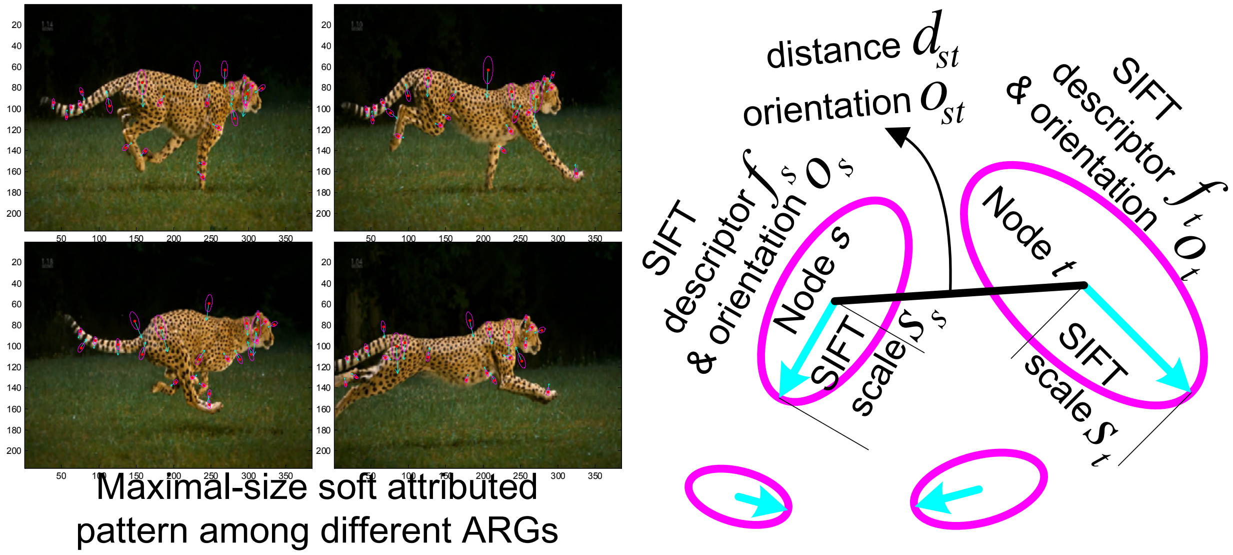

Maximal-size soft attributed pattern among different ARGs
distance d_{st}
orientation o_{st}
SIFT descriptor f_s & orientation o_s
SIFT descriptor f_t & orientation o_t
Node s
Node t
SIFT scale s_s
SIFT scale s_t
20
40
60
80
100
120
140
160
180
200
50
100
150
200
250
300
350

Figure 6.1: Notation for the ARGs that take the interesting points of SIFT features as graph nodes. The ARGs are designed for deformable animals.

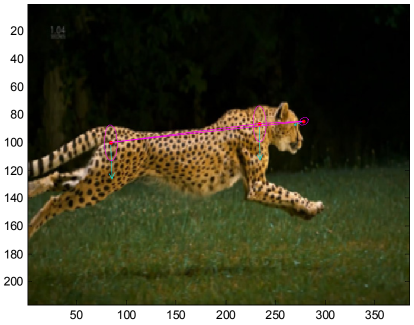


Figure 6.2: The single labeled object that is used to construct the initial graph template. Only three SIFT points are labeled.

and orientation for node s, respectively. The first of the four pairwise attributes is set as the spatial angle between the SIFT orientations of nodes s and t, *i.e.* $\mathcal{F}_1^{st} = angle(o_s, o_t)$. For each edge (s,t), I use d_{st} and o_{st} to represent its length and orientation. The second pairwise attribute, $\mathcal{F}_2^{st} = [angle(o_{st}, o_s), angle(o_{st}, o_t)]^T$, is defined as the angle between edge (s,t) and each of SIFT orientations of nodes s and t. Then, the third and fourth pairwise attributes are set as $\mathcal{F}_3^{st} = \log(s_s/s_t)$ and $\mathcal{F}_4^{st} = [\log(s_s/d_{st}), \log(s_t/d_{st})]^T$, respectively. I

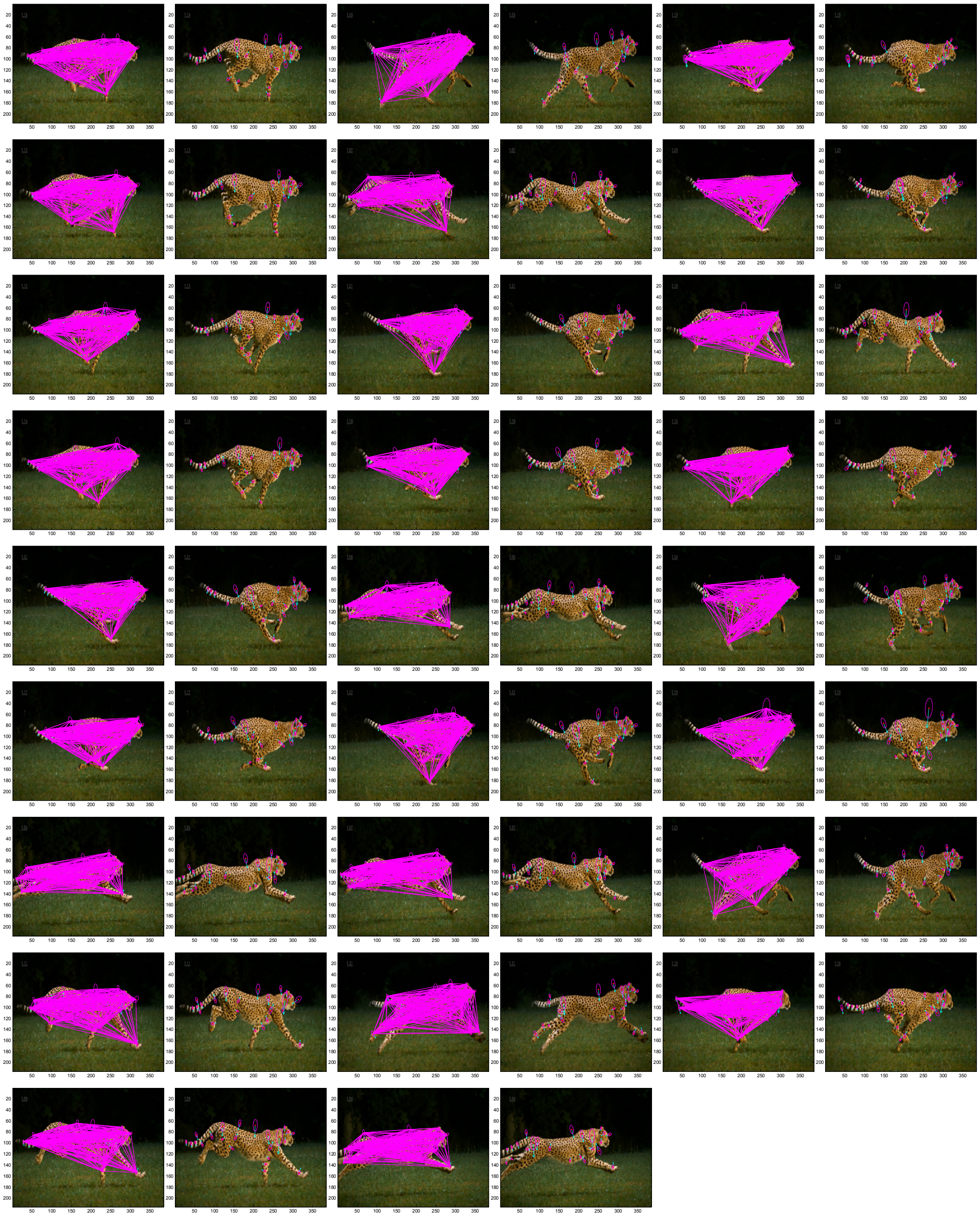


Figure 6.3: Detection of the cheetah in video frames based on the trained deformable model.

set the attribute weights as $w_1^P = 2$ and $\forall j, w_j^Q = 1$. Penalties for matching to *none* are set as $P_{none} = 2$ and $Q_{none} = 4$.

6.2 Experiments

I set the threshold τ to be 4.0 in this experiment. Fig. 6.2 illustrates the initial graph template, which only consists of three SIFT points. The algorithm of attributed graph mining automatically discovers new parts of objects from video frames. Fig. 6.3 shows the detection of the cheetah in video frames based on the trained deformable model, where the whole body of the animal has been recovered.

Chapter 7

Kinect RGB-D image dataset

I have published a dataset of Kinect RGB-D images. This is one of the largest RGB-D object datasets, and fits the requirements of learning graph matching. This dataset consists of RGB-D images containing about 1200 objects. These RGB-D images are collected in different environments, indoors and outdoors. There are ten large categories, such as basket, bucket, drink box, bicycle, scanner, fridge, notebook PC, sprayer, dustpan, and platform lorry. Each category has a large number of objects. The RGB-D image is in the size of 640 × 480. Objects inside a category usually have different textures, and they are placed in complex environments with different translations and rotations. Moreover, objects within some categories have large intra-category variations in size and local structure. For example, bicycles for men have beams, while those for women do not. Small bicycles are usually with simpler structures, and compared to other parts, the wheel radius changes most in size among different bicycles.

Actually, a number of RGB-D datasets have been built in recent years. Lai *et al.* built a large RGB-D object dataset containing 300 objects in 50 categories [60], as well as an RGB-D scene dataset, primarily for use in object recognition. Koppula *et al.*, Silberman *et al.*, and Browatzki *et al.* constructed RGB-D image sets of indoor environments [108, 109, 110]. A dataset for the perception challenge [111] consists of 35 objects for training; the UBC robot vision survey dataset [112] contains four categories of objects; and the 3D table-top dataset [113] covers three categories without significant variation in viewpoint. Janoch *et al.* [114] and Wohlkinger *et al.* [115] also built large datasets.

However, in my scenario of model mining from large and cluttered RGB-D (or RGB) scenes, I have two requirements for the dataset. First, the target objects in the dataset

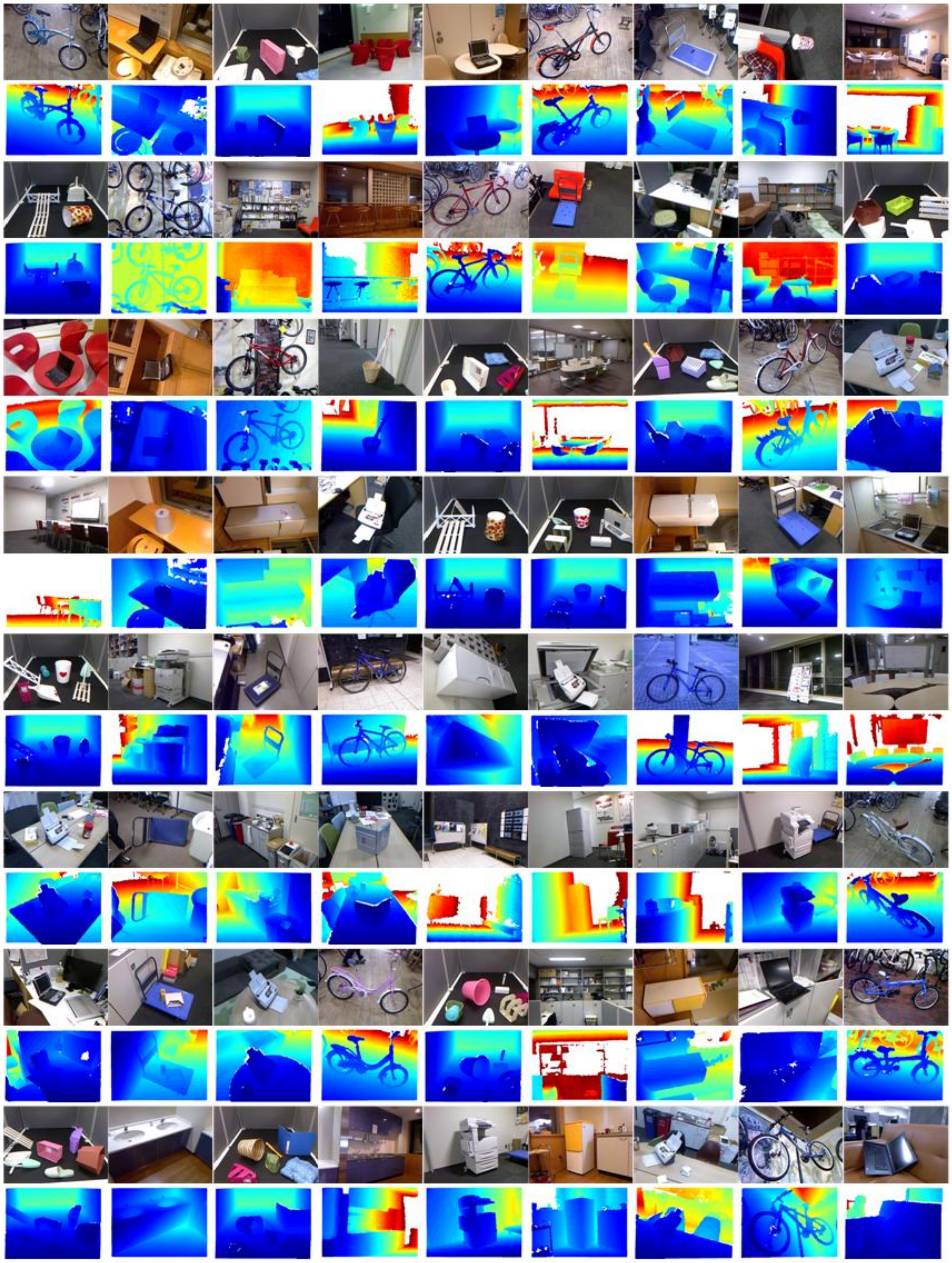


Figure 7.1: Some object samples in the Kinect RGB-D image dataset

should not be hand-cropped or aligned, and they should have different scales, textures, and rotations. Second, each category should include a large number of samples for training (whereas in most existing datasets, each category contains only a few objects). A relatively large number of training images can ensure stability of model-mining processes, although theoretically, my methods of model mining can be successfully applied, just given two cluttered scenes for each category as training images. Therefore, I need to construct this RGB-D image dataset.

Chapter 8

Conclusion and future work

In this thesis, I have defined the concept of "precise object-level visual mining". I have proposed two general platforms and a mining strategy for visual mining. All the proposed algorithms have a large number of extended applications of visual mining, and thus of board interests in the whole field of computer vision. The typical application of mining category models for object detection from big data, as well as two extended applications of mining models for 3D reconstruction and pose estimation, has been tested in experiments, which demonstrates the superior performance and universal applicability of the proposed platforms.

Technically speaking, my methods, for the first time, propose a strategy to simultaneously model all the typical challenges in image processing, which breaks though conventional "model training" to the level of "model mining".

I use the graph theory to model these typical challenges. The problem with the unaligned small-size objects in large and cluttered images can be modeled as a graph matching problem. Then, as I use ARGs to represent ubiquitous images, the intra-category variations in texture, rotation, structure, scale, and illumination can be modeled as attribute variations in ARGs.

However, this is just a new strategy to simultaneously model these challenges, and it is still far from a perfect solution to all of these challenges. It is because that theoretically, my methods can only deal with the challenges when they are able to formulated using ARGs. This is the biggest limitation of this study.

In the future work, I will continue this study to improve the theory of attributed graph mining and apply the general visual-mining platform to more applications, such as object segmentation, and object recognition, and object-level image retrieval.

References

[1] H. Jiang, C.-W. Ngo, Image mining using inexact maximal common subgraph of multiple args, In *International conference on visual information system* (2003) 63–76.

[2] M. Cho, K. Alahari, J. Ponce, Learning graphs to match, In *ICCV*.

[3] T. S. Caetano, J. J. McAuley, L. Cheng, Q. V. Le, A. J. Smola, Learning graph matching, In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 31 (6) (2009) 1048–1058.

[4] M. Leordeanu, M. Hebert, Smoothing-based optimization, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2008) 1–8.

[5] M. Leordeanu, M. Hebert, Unsupervised learning for graph matching, In *International Journal of Computer Vision* 96 (1) (January 2012) 28–45.

[6] L. Torresani, V. Kolmogorov, C. Rother, Feature correspondence via graph matching: Models and global optimization, In *Proceedings of the 10th European Conference on Computer Vision (ECCV)* (2008) 596–609.

[7] Q. Zhang, X. Song, X. Shao, H. Zhao, R. Shibasaki, Learning graph matching for category modeling from cluttered scenes, In *Proc. of IEEE International Conference on Computer Vision (ICCV)*.

[8] M. Cho, K. M. Lee, Progressive graph matching: Making a move of graphs via probabilistic voting, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[9] Q. Zhang, X. Song, X. Shao, H. Zhao, R. Shibasaki, When 3d reconstruction meets ubiquitous rgb-d images, In *International Conference on Computer Vision and Pattern Recognition (CVPR).*

[10] Q. Zhang, X. Song, X. Shao, H. Zhao, R. Shibasaki, Attributed graph mining and matching: An attempt to define and extract soft attributed patterns, In *International Conference on Computer Vision and Pattern Recognition (CVPR).*

[11] P. Hong, T. S. Huang, Spatial pattern discovery by learning a probabilistic parametric model from multiple attributed relational graphs, In *Discrete Applied Mathematics* 139 (2004) 113–135.

[12] L. Thomas, Maximal frequent subgraph mining, International Institute of Information Technology, Hyderabad, India.

[13] J. Wang, Z. Zeng, L. Zhou, Clan: An algorithm for mining closed cliques from large dense graph databases, In *ACM SIGKDD.*

[14] Z. Zeng, J. Wang, L. Zhou, G. Karypis, Coherent closed quasi-clique discovery from large dense graph databases, In *ACM SIGKDD.*

[15] C. Jiang, F. Coenen, M. Zito, A survey of frequent subgraph mining algorithms, In *The Knowledge Engineering Review* (2012) 1–31.

[16] J. Huan, W. Wang, J. Prins, J. Yang, Spin: Mining maximal frequent subgraphs from graph databases, *In* ACM SIGKDD.

[17] L. Thomas, S. Valluri, K. Karlapalem, Margin: Maximal frequent subgraph mining, *In* Transactions on KDD 4 (3).

[18] H. Xie, K. Gao, Y. Zhang, J. Li, H. Ren, Common visual pattern discovery via graph matching, In *ACM Multimedia* (2012) 1385–1388.

[19] H. Liu, S. Yan, Common visual pattern discovery via spatially coherent correspondences, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010) 1609–1616.

[20] M. Leordeanu, M. Hebert, R. Sukthankar, Beyond local appearance: category recognition from pairwise interactions of simple features, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2007) 1–8.

[21] W. Brendel, S. Todorovic, Learning spatiotemporal graphs of human activities, In *ICCV*.

[22] G. Kim, C. Faloutsos, M. Hebert, Unsupervised modeling of object categories using link analysis techniques, *In* Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2008) 1–8.

[23] H.-K. Tan, C.-W. Ngo, Localized matching using earth movers distance towards discovery of common patterns from small image samples, In *Image and Vision Computing* 27 (2009) 1470–1483.

[24] J. Yuan, G. Zhao, Y. Fu, Z. Li, A. Katsaggelos, Y. Wu, Discovering thematic objects in image collections and videos, In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 21 (4) (2012) 2207–2219.

[25] G. Zhao, J. Yuan, Mining and cropping common objects from images, In *ACM Multimedia* (2010) 975–978.

[26] M. Cho, Y. M. Shin, K. M. Lee, Unsupervised detection and segmentation of identical objects, In *International Conference on Computer Vision and Pattern Recognition (CVPR)* (2010) 1617–1624.

[27] D. Parikh, C. Zitnick, T. Chen, Unsupervised learning of hierarchical spatial structures in images, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009) 2743–2750.

[28] V. Kolmogorov, Convergent tree-reweighted message passing for energy minimization, In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 28 (10) (2006) 1568–1583.

[29] Q. Zhang, X. Song, X. Shao, R. Shibasaki, H. Zhao, Category modeling from just a single labeling: Use depth information to guide the learning of 2d models, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013) 193–200.

[30] Q. Zhang, Category dataset of kinect rgbd images, http://sites.google.com/site/quanshizhang.

[31] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour detection and hierarchical image segmentation, In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 33 (5) (2011) 898–916.

[32] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2005) 886–893.

[33] M. Leordeanu, M. Hebert, A spectral technique for correspondence problems using pairwise constraints, In *10th International Conference on Computer Vision (ICCV)* (2005) 1482–1489.

[34] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, C. Rother, A comparative study of energy minimization methods for markov random fields, *In* ECCV.

[35] I. Tsochantaridis, T. Joachims, T. Hofmann, Y. Altun, Large margin methods for structured and interdependent output variables, In *J. Machine Learning Research* 6 (2005) 1453–1484.

[36] Z. P. C. K. Liu, A. Hertzmann, Learning physics-based motion style with nonlinear inverse optimization, In *ACM Transactions on Graphics (TOG)—Proceedings of ACM SIGGRAPH* 24 (3) (2005) 1071–1081.

[37] C. S. Vittorio Ferrari, Frederic Jurie, From images to shape models for object detection, In *International Journal on Computer Vision (IJCV)* 87 (3) (2010) 284–303.

[38] O. Duchenne, A. Joulin, J. Ponce, A graph-matching kernel for object categorization, In *Proc. of IEEE International Conference on Computer Vision (ICCV)* (2011) 1792–1799.

[39] B. G. Park, K. M. Lee, S. U. Lee, J. H. Lee, Recognition of partially occluded objects using probabilistic arg-based matching, In *Computer Vision and Image Understanding (CVIU)* 90 (3) (2003) 217–241.

[40] Microsoft, Introducing kinect for xbox 360, http://www.xbox.com/en-US/Kinect/.

[41] J. Brank, M. Grobelnik, N. Milic-Frayling, D. Mladenic, Feature selection using support vector machines, In *international conf. on data mining methods and databases for engineering.*

[42] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, A. Ng, Building high-level features using large scale unsupervised learning, In *ICML*.

[43] M. Cho, K. Alahari, J. Ponce, Learning graphs to match, In *Proc. of International Conference on Computer Vision (ICCV).*

[44] Q. Zhang, X. Song, X. Shao, H. Zhao, R. Shibasaki, From rgb-d images to rgb images: Single labeling for mining visual models, to appear in *ACM Transactions on Intelligent Systems and Technology (TIST).*

[45] F.-F. Li, R. Fergus, P. Perona, One-shot learning of object categories, In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 28 (4) (2006) 594–611.

[46] T. Tuytelaars, C. H. Lampert, M. B. Blaschko, W. Buntine, Unsupervised object discovery: A comparison, *In* International Journal on Computer Vision 88 (2) (2010) 284–302.

[47] L.-J. Li, G. Wang, F.-F. Li, Optimol: automatic online picture collection via incremental model learning, *In* International Journal on Computer Vision (IJCV) 88 (2) (2010) 147–154.

[48] H. Kang, M. Hebert, T. Kanade, Discovering object instances from scenes of daily living, *In* 13th International Conference on Computer Vision (ICCV) (2011) 762–769.

[49] C. Li, D. Parikh, T. Chen, Automatic discovery of groups of objects for scene understanding, *In* International Conference on Computer Vision and Pattern Recognition (CVPR) (2012) 2735–2742.

[50] A. Faktor, M. Irani, "clustering by composition"–unsupervised discovery of image categories, In *12th European Conference on Computer Vision (ECCV)* (2012) 474–487.

[51] J.-Y. Zhu, J. Wu, Y. Wei, E. Chang, Z. Tu, Unsupervised object class discovery via saliency-guided multiple class learning, *In* International Conference on Computer Vision and Pattern Recognition (CVPR) (2012) 3218–3225.

[52] Y. J. Lee, K. Grauman, Learning the easy things first: Self-paced visual category discovery, *In* International Conference on Computer Vision and Pattern Recognition (CVPR) (2011) 1721–1728.

[53] Z.Liao, A.Farhadi, Y.Wang, I.Endres, D.Forsyth, Building a dictionary of image fragments, *In* International Conference on Computer Vision and Pattern Recognition (CVPR) (2012) 3442–3449.

[54] S. Vijayanarasimhan, K. Grauman, Large-scale live active learning: Training object detectors with crawled data and crowds, *In* International Conference on Computer Vision and Pattern Recognition (CVPR) (2011) 1449–1456.

[55] G. Kim, E. P. Xing, L. Fei-Fei, T. Kanade, Distributed cosegmentation via submodular optimization on anisotropic diffusion, In *Proc. of IEEE International Conference on Computer Vision (ICCV)* (2011) 169–176.

[56] W.-C. Chiu, M. Fritz, Multi-class video co-segmentation with a generative multi-video model, In *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013) 321–328.

[57] A. Joulin, F. Bach, J. Ponce, Multi-class cosegmentation, In *International Conference on Computer Vision and Pattern Recognition (CVPR)* (2012) 542–549.

[58] G. Kim, E. Xing, On multiple foreground cosegmentation, In *International Conference on Computer Vision and Pattern Recognition (CVPR)* (2012) 837–844.

[59] L. Mukherjee, V. Singh, J. Xu, M. Collins, Analyzing the subspace structure of related images: Concurrent segmentation of image sets, In *12th European Conference on Computer Vision (ECCV)* (2012) 128–142.

[60] K. Lai, L. Bo, X. Ren, D. Fox, A large-scale hierarchical multi-view rgb-d object dataset, In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)* (2011) 1817–1824.

[61] X. Ren, L. Bo, D. Fox, Rgb-(d) scene labeling: Features and algorithms, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012) 2759–2766.

[62] D. Fouhey, A. Gupta, M. Hebert, Data-driven 3d primitives for single image understanding, In *Proc. of International Conference on Computer Vision (ICCV).*

[63] X. Wang, X. Bai, T. Ma, W. Liu, L. J. Latecki, Fan shape model for object detection, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012) 151–158.

[64] Y. Xu, Y. Quan, Z. Zhang, H. Ji, Contour-based recognition, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012) 3402–3409.

[65] S. Maji, J. Malik, Object detection using a max-margin hough transform, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2009) 1038–1045.

[66] N. Razavi, J. Gall, P. Kohli, L. v. Gool, Latent hough transform for object detection, In *12th European Conference on Computer Vision (ECCV)* (2012) 312–325.

[67] K. Liu, Q. Wang, W. Driever, O. Ronneberger, 2d/3d rotation-invariant detection using equivariant filters and kernelweighted mapping, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012) 917–924.

[68] T. Wang, X. He, N. Barnes, Learning structured hough voting for joint object detection and occlusion reasoning, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013) 1790–1797.

[69] H.-Y. Chen, Y.-Y. Lin, B.-Y. Chen, Robust feature matching with alternate hough and inverted hough transforms, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013) 2762–2769.

[70] K. Lai, D. Fox, Object recognition in 3d point clouds using web data and domain adaptation, In *Proceedings of International Journal of Robotic Research* 29 (8) (2010) 1019–1037.

[71] E. Hsiao, A. Collet, M. Hebert, Making specific features less discriminative to improve point-based 3d object recognition, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2010) 2653–2660.

[72] W. Hu, Learning 3d object templates by hierarchical quantization of geometry and appearance spaces, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012) 2336–2343.

[73] B. Pepik, P. Gehler, M. Stark, B. Schiele, 3d2pm—3d deformable part models, In *12th European Conference on Computer Vision (ECCV)* (2012) 356–370.

[74] A. Aldoma, F. Tombari, L. D. Stefano, M. Vincze, A global hypotheses verification method for 3d object recognition, In *12th European Conference on Computer Vision (ECCV)* (2012) 511–524.

[75] K. Lai, L. Bo, X. Ren, D. Fox, Sparse distance learning for object recognition combining rgb and depth information, In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)* (2011) 4007–4013.

[76] W. Susanto, M. Rohrbach, B. Schiele, 3d object detection with multiple kinects, In *12th European Conference on Computer Vision (ECCV)* (2012) 93–102.

[77] A. Collet, S. S. Srinivasay, M. Hebert, Structure discovery in multi-modal data: a region-based approach, *In* ICRA.

[78] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from rgbd images, In *12th European Conference on Computer Vision (ECCV)* (2012) 746–760.

[79] K. I. Kim, J. Tompkin, M. Theobald, J. Kautz, C. Theobalt, Match graph construction for large image databases, In *12th European Conference on Computer Vision (ECCV)* (2012) 272–285.

[80] M. Cho, K. M. Lee, Progressive graph matching: Making a move of graphs via probabilistic voting, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012) 398–405.

[81] N. Hu, R. M. Rustamov, L. Guibas, Graph matching with anchor nodes: A learning approach, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013) 2906–2913.

[82] F. Zhou, F. D. la Torre, Deformable graph matching, In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013) 2922–2929.

[83] C. Olsson, Y. Boykov, Curvature-based regularization for surface approximation, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012) 1576–1583.

[84] H. Liu, S. Yan, Efficient structure detection via random consensus graph, In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012) 574–581.

[85] C. Wallraven, B. Caputo, Recognition with local features: the kernel recipe, In *9th International Conference on Computer Vision (ICCV)* (2003) 257–264.

[86] C.-J. Lin, R. C. Weng, Simple probabilistic predictions for support vector regression, In *Technical report* Department of Computer Science, National Taiwan University.

[87] A. Varol, A. Shaji, M. Salzmann, P. Fua, Monocular 3d reconstruction of locally textured surfaces, In *PAMI* 34 (6) (2012) 1118–1130.

[88] J. T. Barron, J. Malik, Shape, albedo, and illumination from a single image of an unknown object, In *CVPR*.

[89] J. T. Barron, J. Malik, Color constancy, intrinsic images, and shape estimation, In *ECCV*.

[90] K. Karsch, Z. Liao, J. Rock, J. T. Barron, D. Hoiem, Boundary cues for 3d object shape recovery, In *CVPR*.

[91] T. Xue, J. Liu, X. Tang, Symmetric piecewise planar object reconstruction from a single image, In *CVPR*.

[92] K. Köser, C. Zach, M. Pollefeys, Dense 3d reconstruction of symmetric scenes from a single image, In *DAGM*.

[93] D. Hoiem, A. Efros, M. Hebert, Geometric context from a single image, In *ICCV*.

[94] O. Barinova, V. Konushin, A. Yakubenko, K. Lee, H. Lim, A. Konushin, Fast automatic single-view 3-d reconstruction of urban scenes, In *ECCV*.

[95] E. Delage, H. Lee, A. Ng, Automatic single-image 3d reconstructions of indoor manhattan world scenes, In *Robotics Research*.

[96] E. Delage, H. Lee, A. Ng, A dynamic bayesian network model for autonomous 3d reconstruction from a single indoor image, In *CVPR*.

[97] A. Saxena, M. Sun, A. Y. Ng, Make3d: Learning 3d scene structure from a single still image, In *PAMI* 31 (5) (2009) 824–840.

[98] M. R. Oswald, E. Töppe, D. Cremers, Fast and globally optimal single view reconstruction of curved objects, In *CVPR*.

[99] L. Zhang, G. Dugas-Phocion, J.-S. Samson, S. M. Seitz, Single view modeling of free-form scenes, In *CVPR*.

[100] E. Töppe, M. R. Oswald, D. Cremers, C. Rother, Image-based 3d modeling via cheeger sets, In *ACCV*.

[101] E. Toppe, C. Nieuwenhuis, D. Cremers, Relative volume constraints for single view 3d reconstruction, In *CVPR*.

[102] T. Hassner, R. Basri, Single view depth estimation from examples, In *CoRR, abs/1304.3915*.

[103] T. Hassner, R. Basri, Example based 3d reconstruction from single 2d images, In *CVPR workshop*.

[104] A. Thomas, V. Ferrari, B. Leibe, T. Tuytelaars, L. Gool, Shape-from recognition: Recognition enables meta-data transfer, In *CVIU*.

[105] Y. Chen, T.-K. Kim, R. Cipolla, Inferring 3d shapes and deformations from single views, In *ECCV*.

[106] Y. Chen, R. Cipolla, Single and sparse view 3d reconstruction by learning shape priors, In *CVIU* 115 (5) (2011) 586–602.

[107] E. Herbst, X. Ren, D. Fox, Rgb-d object discovery via multi-scene analysis, In *IROS*.

[108] H. S. Koppula, A. Anand, T. Joachims, A. Saxena, Semantic labeling of 3d point clouds for indoor scenes, In *Neural Information Processing Systems (NIPS)* (2011) 244–252.

[109] N. Silberman, R. Fergus, Indoor scenen segmentation using a structrued light sensor, In *IEEE International Conference on Computer Vision Workshop (ICCV Workshops)* (2011) 601–608.

[110] B. Browatzki, J. Fischer, G. Birgit, H. Bulthoff, C.Wallraven, Going into depth: Evaluating 2d and 3d cues for object classification on a new, large-scale object dataset, In *IEEE International Conference on Computer Vision Workshop (ICCV Workshops)* (2011) 1189–1195.

[111] G. Bradski, T. Hong, Nist and willow garage: Solution in perception challenge, http://www.willowgarage.com/blog/2011/02/28/nist-and-willow-garage-solutions-perception-challenge.

[112] S. Helmer, D. Meger, M. Muja, J. J. Little, D. G. Lowe, Ubc robot vision survey, http://www.cs.ubc.ca/labs/lci/vrs/index.html.

[113] M. Sun, G. Bradski, B.-X. Xu, S. Savarese, Depth-encoded hough voting for joint object detection and shape recovery, In *The 11th European Conference on Computer Vision (ECCV)* (2010) 658–671.

[114] A. Janoch, The berkeley 3d object dataset. `http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-85.html`, Master's thesis, EECS Department, University of California, Berkeley (May 2012).

[115] W. Wohlkinger, A. Aldoma, R. Rusu, M. Vincze, Large-scale object class recognition from cad models, In *Proc. of IEEE International Conference on Robotics and Automation (ICRA)* (2012) 5384–5391.

Acknowledgements

Within these five years, I have forgotten how many times I doubted about my research. "Can my current topics really make some changes to this field?" "Based on current methodologies, has this filed fallen into a bottleneck or does it still have any hidden hope for new technical breakthroughs?" I have changed my topics from the field of robotics to the fields of computer vision and machine learning. I have tried to search the hope in directions of intelligent vehicles, 3D point cloud processing, RGB-D image processing, pedestrian detection, pose estimation, motion capture, and object recognition, and then escaped from them.

Thanks to the God, who I may know or may not know, after the first three years of continuous failure and depression, I can finally find the topic of visual mining and obtain the gift to invent the powerful tool of attributed graph mining to deal with all the challenges related to visual mining, which makes the impossible task possible.

The person who I would like to thank most is my supervisor, Prof. Shibasaki Ryosuke. He has made a loose and free research environment for me, which made me be able to survive with great pressure and self-doubt deep in my mind in these years.

I have received much academic guidance from Dr. Xuan Song. He made me keep confident when I had experienced much failure and could not determine my research direction. Many ideas in my studies are inspired by discussion between us.

I also want to thank Dr. Xiaowei Shao, Dr. Yulin Duan, and Prof. Huijing Zhao for their supports.

Curriculum Vitae

Brief Background Description

My name is Quanshi Zhang. I was born on 22, October 1986, and received the Bachelor of Science from the School of Electronics Engineering and Computer Science, Peking University, China, in 2009, and the Master of Engineering, from the Civil Engineering Department, University of Tokyo, in 2011. From 2011 to 2014, I am a doctoral student in Center for Spatial Information Science, Civil Engineering Department, University of Tokyo, supervised by Prof. Ryosuke Shibasaki. I will get the doctoral degree in September 2014. My main research interests are computer vision, machine learning, and robotics, especially on the graph theory and visual knowledge mining from big visual data.

Education

9/2005—8/2009 Peking University
Undergraduate student in Department of Computer Intelligence Science (CIS), School of Electronics Engineering & Computer Science (EECS)
10/2009—9/2011 University of Tokyo
Master student in Department of Civil Engineering
10/2011—9/2014 University of Tokyo
Doctoral student in Department of Civil Engineering

Publication List

Journal paper:

(1) **Quanshi Zhang**, Xuan Song, Xiaowei Shao, Huijing Zhao, and Ryosuke Shibasaki,

"From RGB-D Images to RGB Images: Single Labeling for Mining Visual Models", to appear in ACM Transactions on Intelligent Systems and Technology (TIST), 2014.

(2) **Quanshi Zhang**, Xuan Song, Xiaowei Shao, Huijing Zhao, and Ryosuke Shibasaki, "Unsupervised skeleton extraction and motion capture from 3D deformable matching", Neurocomputing, Elsevier, pp.170–182 2013.

(3) Xuan Song, **Quanshi Zhang**, Y. Sekimoto, T. Horanont, S. Ueyama, Ryosuke Shibasaki, "An Intelligent System for Large-scale Disaster Behavior Analysis and Reasoning", accepted by IEEE Intelligent Systems.

(4) Xuan Song, Xiaowei Shao, **Quanshi Zhang**, Ryosuke Shibasaki, Huijing Zhao, Jinshi Cui, Hongbin Zha, "A Fully Online and Unsupervised System for Large and High Density Area Surveillance: Tracking, Semantic Scene Learning and Abnormality Detection", ACM Transactions on Intelligent Systems and Technology (ACM-TIST), 4(2): 20, 2013.

(5) Xuan Song, Xiaowei Shao, **Quanshi Zhang**, Ryosuke Shibasaki, Huijing Zhao, Hongbin Zha, "A Novel Dynamic Model for Multiple Pedestrians Tracking in Extremely Crowded Scenarios", Information Fusion, Elsevier, 2012.

Conference paper:

(6) **Quanshi Zhang**, Xuan Song, Xiaowei Shao, Huijing Zhao, and Ryosuke Shibasaki, "Unsupervised Learning for Graph Matching: An Attempt to Define and Extract Soft Attributed Patterns", in Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2014.

(7) **Quanshi Zhang**, Xuan Song, Xiaowei Shao, Huijing Zhao, and Ryosuke Shibasaki, "When 3D Reconstruction Meets Ubiquitous RGB-D Images", in Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2014.

(8) **Quanshi Zhang**, Xuan Song, Xiaowei Shao, Huijing Zhao, and Ryosuke Shibasaki, "Start from Minimum Labeling: Learning of 3D Object Models and Point Labeling from a Large and Complex Environment", in Proc. of IEEE International Conference on Robotics

and Automation (ICRA) 2014.

(9) X. Song, **Quanshi Zhang**, Y. Sekimoto, R. Shibasaki, “Prediction of Human Emergency Behavior and their Mobility following Large-scale Disaster”, to appear in Proc. of 20th SIGKDD conference on Knowledge Discovery and Data Mining (KDD 2014), 2014.

(10) Xuan Song, **Quanshi Zhang**, Y. Sekimoto and Ryosuke Shibasaki, “Intelligent System for Urban Emergency Management During Large-scale Disaster”, in Proc. of Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI) 2014.

(11) **Quanshi Zhang**, Xuan Song, Xiaowei Shao, Huijing Zhao, and Ryosuke Shibasaki, “Learning Graph Matching for Category Modeling from Cluttered Scenes”, in Proc. of 14th International Conference on Computer Vision (ICCV) 2013.

(12) **Quanshi Zhang**, Xuan Song, Xiaowei Shao, Huijing Zhao, and Ryosuke Shibasaki, “Category Modeling from just a Single Labeling: Use Depth Information to Guide the Learning of 2D Models”, in Proc. of IEEE International Conference on Computer Vision and Pattern Recognition (CVPR) 2013.

(13) **Quanshi Zhang**, Xuan Song, Xiaowei Shao, Huijing Zhao, and Ryosuke Shibasaki, “Unsupervised 3D Category Discovery and Point Labeling from a Large Urban Environment”, in Proc. of IEEE International Conference on Robotics and Automation (ICRA) 2013.

(14) Xuan Song, **Quanshi Zhang**, Y. Sekimoto, T. Horanont, S. Ueyama, Ryosuke Shibasaki, “Modeling and Probabilistic Reasoning of Population Evacuation During Large-scale Disaster”, in Proc. of 19th SIGKDD conference on Knowledge Discovery and Data Mining (KDD 2013). (full paper)

(15) Xuan Song, X. Shao, **Quanshi Zhang**, Ryosuke Shibasaki, Huijing Zhao, Hongbin Zha, “Laser-based Intelligent Surveillance and Abnormality Detection in Extremely Crowded Scenarios”, in Proc. of IEEE International Conference on Robotics and Automation (ICRA), pp. 2170-2176, 2012.

(16) Huijing Zhao, **Quanshi Zhang**, M. Chiba, Ryosuke Shibasaki, Jinshi Cui, Hongbin Zha, “Moving Object Classification using Horizontal Laser Scan Data”, in Proc. of IEEE

International Conference on Robotics and Automation (ICRA), 2009.