2013

VISUAL SEMANTIC SEGMENTATION AND ITS APPLICATIONS

Jizhou Gao
University of Kentucky, zhou0620@gmail.com

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained and attached hereto needed written permission statements(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine).

I hereby grant to The University of Kentucky and its agents the non-exclusive license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless a preapproved embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's dissertation including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Jizhou Gao, Student

Dr. Ruigang Yang, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

2013

VISUAL SEMANTIC SEGMENTATION AND ITS APPLICATIONS

Jizhou Gao
University of Kentucky, zhou0620@gmail.com

STUDENT AGREEMENT:

I represent that my thesis or dissertation and abstract are my original work. Proper attribution has been given to all outside sources. I understand that I am solely responsible for obtaining any needed copyright permissions. I have obtained and attached hereto needed written permission statements(s) from the owner(s) of each third-party copyrighted matter to be included in my work, allowing electronic distribution (if such use is not permitted by the fair use doctrine).

I hereby grant to The University of Kentucky and its agents the non-exclusive license to archive and make accessible my work in whole or in part in all forms of media, now or hereafter known. I agree that the document mentioned above may be made available immediately for worldwide access unless a preapproved embargo applies.

I retain all other ownership rights to the copyright of my work. I also retain the right to use in future works (such as articles or books) all or part of my work. I understand that I am free to register the copyright to my work.

REVIEW, APPROVAL AND ACCEPTANCE

The document mentioned above has been reviewed and accepted by the student's advisor, on behalf of the advisory committee, and by the Director of Graduate Studies (DGS), on behalf of the program; we verify that this is the final, approved version of the student's dissertation including all changes required by the advisory committee. The undersigned agree to abide by the statements above.

Jizhou Gao, Student

Dr. Ruigang Yang, Major Professor

Dr. Miroslaw Truszczynski, Director of Graduate Studies

VISUAL SEMANTIC SEGMENTATION AND ITS APPLICATIONS

DISSERTATION

A dissertation submitted in partial
fulfillment of the requirements for
the degree of Doctor of Philosophy
in the College of Engineering at the
University of Kentucky

By
Jizhou Gao
Lexington, Kentucky

Director: Dr. Ruigang Yang, Associate Professor of Computer Science
Lexington, Kentucky 2013

ABSTRACT OF DISSERTATION

VISUAL SEMANTIC SEGMENTATION AND ITS APPLICATIONS

This dissertation addresses the difficulties of semantic segmentation when dealing with an extensive collection of images and 3D point clouds. Due to the ubiquity of digital cameras that help capture the world around us, as well as the advanced scanning techniques that are able to record 3D replicas of real cities, the sheer amount of visual data available presents many opportunities for both academic research and industrial applications. But the mere quantity of data also poses a tremendous challenge. In particular, the problem of distilling useful information from such a large repository of visual data has attracted ongoing interests in the fields of computer vision and data mining.

Structural Semantics are fundamental to understanding both natural and man-made objects. Buildings, for example, are like languages in that they are made up of repeated structures or patterns that can be captured in images. In order to find these recurring patterns in images, I present an unsupervised frequent visual pattern mining approach that goes beyond co-location to identify spatially coherent visual patterns, regardless of their shape, size, locations and orientation.

First, my approach categorizes visual items from scale-invariant image primitives with similar appearance using a suite of polynomial-time algorithms that have been designed to identify consistent structural associations among visual items, representing frequent visual patterns. After detecting repetitive image patterns, I use unsupervised and automatic segmentation of the identified patterns to generate more semantically meaningful representations. The underlying assumption is that pixels capturing the same portion of image patterns are visually consistent, while pixels that come from different backdrops are usually inconsistent. I further extend this approach to perform automatic segmentation of foreground objects from an Internet photo collection of landmark locations.

New scanning technologies have successfully advanced the digital acquisition of large-scale urban landscapes. In addressing semantic segmentation and reconstruction of this data using LiDAR point clouds and geo-registered images of large-scale residential areas, I develop a complete system that simultaneously uses classification and segmentation methods to first identify different object categories and then apply

category-specific reconstruction techniques to create visually pleasing and complete scene models.

KEYWORDS: Automatic Segmentation, Frequent Visual Pattern Mining, Simultaneous Classification and Segmentation, Internet Photo Collections, LiDAR Data

Author's signature: Jizhou Gao

Date: December 10, 2013

VISUAL SEMANTIC SEGMENTATION AND ITS APPLICATIONS

By
Jizhou Gao

Director of Dissertation: Ruigang Yang

Director of Graduate Studies: Miroslaw Truszczynski

Date: December 10, 2013

To my parents, Guoying Wang and Bo Gao.

ACKNOWLEDGMENTS

First, I would sincerely like to thank my advisor, Dr. Ruigang Yang, for all the guidance and support that have made my Ph.D. journey so enjoyable. Without his help, I could never have earned a doctorate degree. He introduced me to computer vision, and taught me a lot about critical thinking, technical writing and presenting. I am fortunate to have had him as my advisor. I would also like to thank my co-advisor Dr. Jinze Liu especially for her inspiring advice on the project of frequent image pattern mining.

I am grateful to my committee members, Dr. Sen-ching Cheung, Dr. Jerzy W Jaromczyk and Dr. Qiang Ye for reviewing my dissertation. My thanks also go to my coauthors, friends and colleagues in the GRAVITY lab, including Yin Hu, Hui Lin, Dr. Yu Zhou, Dr. Xinyu Huang, Dr. Liang Wang, Dr. Miao Liao, Dr. Xianwang Wang, Dr. Jiejie Zhu, Dr. Yu Huang, Chenxi Zhang, Mao Ye, Qing Zhang, Bo Fu, Changpeng Ti, Yan Huang, Jin Zhou, Fengtao Fan, Nan Hu, Xinan Liu and Yongwook Song.

Most importantly, I would like to deeply thank my parents Guoying Wang and Bo Gao for all their unconditional love and encouragement throughout my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1 Introduction

1.1 Motivation and Contributions

Segmentation is the process of dividing data into a set of groups, each of which contains similar elements. In image segmentation, for example, the pixels of an image are decomposed and grouped into regions that have visual similarities and/or share common properties. Image segmentation is one of the most intensely studied processes because it serves as a starting point for solving many other vision-related problems and has greatly advanced the field of object recognition, 3D reconstruction and image understanding. In the more recent state-of-the-art algorithms, the boundaries of the detected regions [7, 60, 94] often coincide with the true object boundaries; nevertheless, cutting out the objects of interest using semantic labels still requires high-level knowledge for further merging, refinement and understanding. This problem is known as the visual "semantic gap," defined in [98] as "the lack of coincidence between the information that one can extract from the visual data and the interpretation that the same data have for a user in a given situation".

Interactive segmentation, on the other hand, uses a persons knowledge of the location, size, colors and/or boundaries to help segment a target object from an image, for example via a user-provided bounding box [59, 86] and strokes [11, 62]. As the digital cameras that capture our world become more and more ubiquitous, however, the sheer amount of data that is being generated makes manual labeling

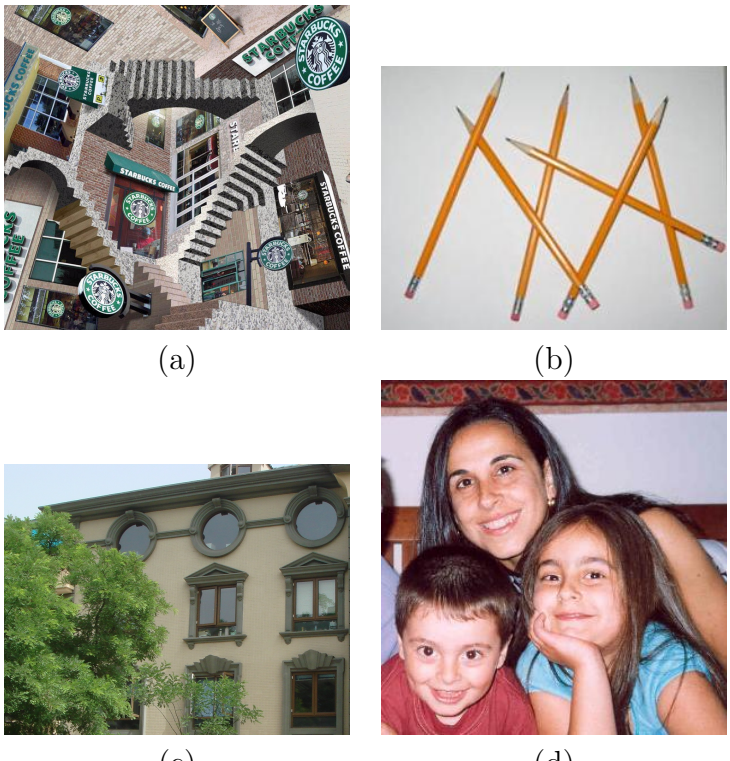
extremely labor intensive, and even simple tasks such as selecting bounding boxes become daunting.

Another related line of research is object recognition, which focuses on detecting and classifying objects in images. The idea of concurrently recognizing and segmenting objects has been employed by [13, 37, 105, 116]. However, most object recognition algorithms need a relatively large training dataset and target object categories that are predefined, which is impractical due to the huge number of object classes in the real world.

Besides images, significant advances in scanning technologies have been useful in the digital acquisition of large scale urban landscapes. For example, Google has added laser scanners to its StreetView sensor platform, making it possible to acquire both range and appearance information from urban environments at the same time. Furthermore, we expect that this 3D point cloud and image data will be available to the public in the near future. Many efforts have already been made to achieve accurate semantic segmentation and classification, but the typical method involves a monocular video sequence filmed at the street level. Traditional approaches employ 2D appearance information, such as color, texture, and shape [47, 67, 71] and have achieved impressive results. However, a drawback of appearance-based features is that appearances may change dramatically under different imaging conditions. For example, a scene may have different appearances depending on the time of day or the season. With recent advances in 3D imaging, 3D structure information has been used for semantic segmentation and recognition [12, 121, 126]. Specifically, when the input is a video sequence rather than a still image, the available motion-based cues

contain a large amount of 3D information that can be used for segmentation and recognition. Unlike the information derived from still images, moreover, these cues are not subject to appearance changes. Due, in part, to the inaccurate or low-density point clouds reconstructed from these images, however, the recognition rate still needs to be improved.

The research presented in this dissertation aims at automatic semantic segmentation and its applications for images and ground-based LiDAR (LIght Detection And Ranging) point clouds. Towards this goal, I have contributed a few innovative algorithms for three different visual data sources including a set of images with repetitive patterns embedded, Internet photo collections taken at landmark locations, and residential scenes captured in ground-based LiDAR data.



(a) (b)

(c) (d)

Figure 1.1: Sample images with repeated visual patterns.

Repetitive Pattern Mining and Segmentation from Images Visual semantics can be regarded as a specific set of relationships that connect visual words carrying specific meanings. For example, the four corners of a window and the relationships between these corners as shown in Figure 1.1(c) suggest the existence of a window. Assume that semantics manifest themselves through *repetition*. For example, the structure of windows is repeatedly exhibited in building structures. The ultimate goal, therefore, is to find frequently occurring patterns that are composed of visual features in images. To achieve this goal, existing methods typically sample the features randomly within a small spatial neighborhood (e.g., [16, 124]), and search for frequently co-occurring features within the sampled neighborhood and/or require a supervised training set of features for a particular classification purpose(e.g., [22–24]). These approaches lack a systematic method for globally capturing visual structural semantics or patterns and are often unable to detect complex patterns that appear at random locations within one or multiple images and that vary in either size or shape, or have missing features. I develop a method for *unsupervised* learning of high-order structural semantics in images represented by frequently co-occurring features with consistent spatial relationships. In addition to the detection problem, I develop a frequent visual pattern segmentation algorithm that provides more semantically rich representation. A simple method is to use the bounding box or convex hull of the key features that make up the visual patterns. However, this is unlikely to achieve satisfactory results for complex shapes such as a human figure. Instead, I present an *unsupervised* and *automatic* segmentation method of the frequently occurring patterns by leveraging the unique advantage provided by large collections

of images: redundancy. With the assumption that subjects of interest are often consistent across multiple images with varying backdrops, the repeatedly appearing objects are segmented based on an integrated set of evidence including low variance in foreground and/or consistent visual cues in color and edges.

Image Segmentation based on Internet Photo Collections Given the growth of Internet Photo collections, we now have a visual index of all major cities and tourist sites in the world. However, it is still a difficult task to capture a perfect shot with one's own camera when visiting these places. With the seminal work by Snavely et al. [99] that used Internet photo collections (IPC) for 3D reconstruction and visualization, many image editing operations have been developed to unlock the rich information contained in IPCs. Among these image enhancement techniques, one important prerequisite is the segmentation of the foreground. As opposed to interactive segmentation methods that rely on user interaction to learn the foreground and background appearance models, I develop an algorithm that can automatically acquire training data based on the IPC and achieves high quality foreground segmentation.

Semantic Segmentation and Reconstruction of Residential Scenes from LiDAR Data Due to their vast applications in many areas, 3D reconstruction and modeling of urban environment have long been an active topic in many research communities, from computer graphics, computer vision, and photogrammetry to remote sensing. Based on the type of input data (2D vs 3D), the reconstruction scale (single

building vs. block/city), and the output models (facade, 2.5D, or full 3D), many algorithms and systems have been developed, enabling the faster production of better 3D models. Even though much progress has been made, as pointed out in a recent survey by Musialski et al. [78], "automatic large-scale reconstruction remains an open problem." Existing methods either use airborne LiDAR data to generate 2.5D models that lack street-level details (e.g., [56, 83, 131]) or use ground-level images or LiDAR for street-side modeling only (e.g., [26, 119]). While fusion of ground-and-airborne data can produce the most complete model (e.g., [29]), the fidelity of the model could be improved. In addition, the output models from existing approaches are usually produce a lot of polygons (sometimes per building) with little semantic labeling, which is why I develop a complete system to semantically segment and reconstruct 3D models from ground-based LiDAR point clouds. As opposed to previous urban modeling approaches, our system is designed for residential scenes, which has long been overlooked. Admittedly, the low-rise houses and landscape in surburban/residential areas are less glamorous to work with from a visualization standpoint, but they are equally important from a simulation or city planning standpoint since they are literally everywhere. For these areas, automation is important because of the large scale. Our system fills the void of automatic segmentation and reconstruction of residential areas with semantic tags from ground-based LiDAR data.

1.1.1 Innovations

The overall contribution of this dissertation are its innovations in semantic segmentation and its applications to images and 3D point clouds. It makes a few solid steps

towards the goal of visual data segmentation and semantics understanding. It also provides some new insights into these challenging research areas. In addition to the overall contributions, some of the technical innovations necessitated by this research have led to several publications in related conferences and journals.

Finding Repetitive Patterns in Images My approach starts by cataloguing the visual features of images into a set of visual items. Next, strong associations between two classes of visual features evidenced by consistent and frequent geometric relationships are identified. Finally, these pairwise associations are used to compose more complex and complete structural patterns present in the images. My approach makes the following contributions: I articulate an idea for mining frequent consistent associations among visual items in images regardless of their size, shape and orientation; I develop a polynomial-time algorithm for mining maximal associations between pairs of visual items; and I develop a polynomial-time method for extracting frequent high-order structural visual patterns among more than two visual items. Experiments on synthetic data and a variety of real-world datasets demonstrate the efficiency and effectiveness of the proposed methods. This is the joint work with Yin Hu, Jinze Liu and Ruigang Yang, first published in [31], ICCV[1] 2009.

Automatic Visual Pattern Segmentation In addition to the detection problem, I also address the problem of frequent visual pattern segmentation. The basic idea is straightforward. Through frequent pattern mining, feature correspondences between different instances of a visual pattern can be established. Using these cor-

[1]IEEE International Conference on Computer Vision

respondences, I can warp all instances of the same visual pattern into a reference view. For a foreground pixel, all of the warped pixels should be similar; and for a background pixel, the warped pixels should differ significantly since they are from different backdrops. By counting the statistical coherence of warped pixels, I can make a per-pixel decision whether it is foreground or not, achieving the goal of automatic segmentation. A similar idea has been explored in [65], IEEE TVCG[2].

Automatic Image Segmentation based on Internet Photo Collections I design a novel foreground and background segmentation algorithm that automatically differentiates foreground based on image appearance statistics obtained from Internet photo collections of landmark locations, surveyed over a variety of different illumination conditions. Compared to many existing segmentation methods, this approach requires no user interaction to obtain a high quality foreground segmentation. Even for complicated scenes with multiple foreground layers, only minimal user interaction is needed to separate foreground layers. This is a part of the joint work with Chenxi Zhang, Oliver Wang, Pierre Georgel, Ruigang Yang, James Davis, Jan-Michael Frahm and Marc Pollefeys, first published in [125], IEEE TVCG[2].

Semantic Segmentation and Reconstruction of Residential Scenes from LiDAR Data Starting with ground-based LiDAR point clouds with registered color images, I first segment the unorganized 3D points into distinctive categories including houses, plants, street lights, etc. Then for each category we develop unique solutions to reconstruct its 3D model, taking advantage of prior information about this partic-

[2]IEEE Transactions on Visualization and Computer Graphics

ular category. For example, common objects, such as street lights, are replaced by similar 3D models found on the Internet. Plants are modeled with billboard techniques, which are known to be visually convincing. The outcome of our system is a set of visually complete 3D models consisting of common static objects in an urban scene, including not only houses, but also plants, street lights, mailboxes, etc. Each object has its own semantic labeling. To the best of my knowledge, it is the first complete system that can automatically segment and reconstruct high-quality 3D models from ground-based LiDAR data. This is the joint work with Hui Lin, Yu Zhou, Guiliang Lu, Mao Ye, Chenxi Zhang, Ligang Liu and Ruigang Yang, first presented in [66], ACM SIGGRAPH[3] 2013.

Online Building Segmentation from Ground-based LiDAR Data in Urban Scenes I present a fast and accurate building segmentation algorithm from ground-based LiDAR points. The basic idea is that buildings can be observed in a street view and are separated by empty spaces such as alleys. By progressively projecting 3D points onto street views along the scanning path, buildings can be detected as large regions with dense points. The main contribution is to automatically segment buildings from large scale unorganized point clouds with online performance and without requiring any training data. This is the joint work with Ruigang Yang, first published in [32], 3DV[4] 2013.

[3]ACM Transaction on Graphics - SIGGRAPH 2013 Conference Proceedings
[4]Third Joint 3DIM/3DPVT Conference

1.2 Dissertation Outline

The rest of this dissertation is organized as followings. In Chapter 2, I review the related work on semantic segmentation of images and LiDAR point clouds in the literature. Chapter 3 presents the automatic and unsupervised algorithms used to extract and segment structural semantics, i.e., frequently occurring patterns that are composed of visual features in images. Chapter 4 describes the automatic segmentation of a personal photo based on Internet photo collections. Chapter 5 details semantic segmentation and its application to the 3D reconstruction of residential scenes from ground-based LiDAR data. Finally, I conclude the dissertation in Chapter 6.

Chapter 2 Related Work

Image segmentation is an intensely studied problem that serves as a starting point for solving many other computer vision problems. Given the large body of work on this topic, it is beyond the scope of this dissertation to provide a comprehensive review. Interested readers are referred to computer vision and image processing textbooks such as [101] and [36] that summarize classical segmentation methods. In this chapter, I mainly discuss the semantic segmentation methods used on images and 3D point clouds that are relevant to the methods presented later in the dissertation.

2.1 Finding Frequent Visual Patterns

Frequent visual pattern mining is a relatively new research topic bridging the areas of computer vision and data mining. In this section, I begin by briefly reviewing extant research, most of which is conducted in the fields of computer vision or data mining.

Research in the field of image processing has enabled the extraction of scale-invariant visual primitives. The extracted visual primitives are typically further quantized into a small set of visual items, where each visual item contains primitives of a similar appearance. An image therefore contains a collection of visual items. With a dictionary of visual items available, the bag-of-words model [57, 61, 91] is proposed to treat images analogous to text documents while ignoring spatial information. This simple but effective model has many important applications. For example, given a target object, image retrieval systems (such as [80, 82, 96]) output

a set of representative images from a large database. Our approach differs from the typical image retrieval methods in that we do not have any target object to begin with, but instead we seek to find similar patterns within the entire collection of images. Thus, the search space of our problem is significantly larger than that of other image retrieval systems.

In the data mining community, traditional spatial data mining mainly focuses on finding spatial association rules within the tagged map database, such as [50]. Some image mining approaches such as [41, 49, 97] can also discover frequent spatial patterns in images regardless of rotation, scaling and translation. However, [49] assumes that the model parts can be identified without any ambiguities and missing detections is thus less effective on a real-world image. Some other work, such as [102, 103, 124], proposed a technique for translating image features into a transactional database from which the co-located features can then be mined using frequent itemset mining. However, associated primitives are not limited to being spatially co-located (for example, see the long pencils in Figure 3.2). Our work is substantially different from their work, since we try to detect strong pairwise associations over the entire image space.

Our pattern composition algorithm bears some similarity to frequent subgraph mining. Approaches to frequent subgraph mining have been well studied both in a collection of graphs [75, 85, 122] and in one sparse single graph [52]. However, these approaches are often quite time-consuming, since they need intensive graph isomorphism tests, require exhaustive searching and generate a large number of subgraphs. Our problem formulation develops a heuristic approach, which enables fast and ac-

curate pattern mining. In addition, traditional approaches are limited to unweighted graphs and cannot be effectively applied to weighted graphs. In our case, the numerical weight of the edges of the weighted graph represent the relationship between the features of an image.

Part-based approaches have also been used to learn, detect and recognize object models in images, such as the sparse flexible model [16], the constellation model [23], the star model [24], the pictorial structure [22] and the semi-local parts [89,90]. They typically assign a frequently occurring pattern to a connected graphical structure built upon a collection of parts, where each part corresponds to a local image patch. In general, these methods are computationally expensive, and need to provide some restrictive priors, such as the known number of parts and spatial proximity.

Another related line of research is object recognition which focuses on detecting, classifying and even segmenting objects in images. The idea of concurrently recognizing and segmenting objects has been explored in [5, 13, 37, 58, 88, 105, 116]. Most object recognition algorithms need a relatively large training dataset and require the object to constitute a dominant portion of the input image, which often implicitly brings the need for pre-segmentation.

By providing just one additional image, co-segmentation aims to simultaneously segment the similar objects embedded in a pair of images with different backgrounds. Co-segmentation typically utilizes a global appearance model for the foreground object, such as appearance (color and/or filter bank) histograms [40, 87, 109]. It is further extended to incorporate human interaction in [8] and handle more than two images in [46].

Methods used in texel discovery also relate to our work. A texel is defined as a texture element which repeatedly occurs in a particular texture. Hays et al. fit a 2D lattice structure to detect texels in [39]. In [3], Ahuja et al. detect and infer partially occluded texels by learning the substructures in a segmentation tree. Relying on the assumption that texels are distributed in a regular grid with many occurrences, these techniques will have a difficult time in finding large but less-structured patterns. In addition, we do not assume spatial regularity of similar patterns.

2.2 Image Editing using Internet Photo Collections

With the prevalence of consumer cameras and large-scale photograph storage websites, 3D modeling from IPCs has become a hot topic in recent years [1, 26, 30, 99]. Snavely et al. [99] presented pioneering work using photographs from IPCs to compute a 3D model reconstruction and recover camera poses. Furukawa and Ponce [30] presented efficient clustering and filtering algorithms for parallel reconstruction that enforced inter-cluster consistency constraints over the entire reconstruction. Subsequently, Agarwal et al. [1] and Frahm et al. [26] advanced the state-of-the-art of city scale reconstruction from IPCs, with improved geometric accuracy and computational performance. Our work leverages new sources of prior information for use in personal photo segmentation.

There have been several techniques for image editing using a large number of images downloaded from the web, such as colorizing gray-scale images [18, 69], enhancing CG images [44], enhancing face images using good example prior photos of the same person [45], and image completion [19, 33, 38, 115].

Interactive image segmentation brings a user's prior knowledge of the location, size, color, and depth boundaries to segment a target object from an image, for example via a user-provided bounding box [59,86] and strokes [11,62]. However, even simple labeling tasks such as dragging a bounding box may still be daunting when dealing with lots of images. We leverage the opportunity that IPCs registered to the same model allow, enabling us to measure color consistency between the personal photo and IPC and filter out foreground and background color seeds, in order to obtain high quality segmentation.

2.3 Semantic Segmentation and Modeling for Urban Environment

The problem of 3D point cloud segmentation and classification has received considerable attention in recent years, due to the increasingly availability of active 3D sensors, such as LiDAR devices. A line of research focuses on applying sophisticated machine learning techniques of a Markov Random Field model to classify point clouds, such as [6, 77, 127]. Golovinskiy et al. [35] propose a system to segment and recognize objects in point clouds of urban scenes. A potential object location is found by clustering nearby points, followed by applying a min-cut approach [34] to segment points for each object, and finally by classifying segmented objects based on shape descriptors and contextual information. These methods require manually labeling a few training data sets and the feature computation and inference is often time-consuming for large scale data. Compared to these methods, our approach is fully unsupervised and very efficient at processing a large amount of data.

Recently, Stamos et al. [100] proposed an online algorithm to classify scanned

points into five distinct classes (vegetation, vertical, horizontal, car and curb regions) during data acquisition. This is a very interesting and promising approach; however, they are somehow working on organized point clouds as data points are arriving at a sensor one by one in a scanline so that many useful local features can be computed more efficiently. Our method, on the other hand, can work on a large unorganized list of points, even when no spatial information between points is readily available, and still achieves online performance. Another method for the real-time detection of repeated structures in point clouds is presented in [27].

Korah et al. [51] used a strip histogram grid representation encoding an urban scene as a grid of vertical strips, each of which corresponds to a 3D population histogram. This representation segments objects by grouping similar adjacent strips together. Although this method is very efficient, it does not work well for terrestrial scans with partial roofs or no roofs, and it would break a building object into many segments if there were a large height discontinuity within a single building, such as Fig. 5.14(a)(d). In addition, as we have discussed, the approaches that only work on airborne LiDAR data (e.g., [74]) are not easily extended to ground-based data because roofs are only partially observed or even completely missing.

Wang et al. [112] propose an automatic heuristic window detection algorithm on facades detected by RANSAC [25] plane fitting. In [127, 132], some image processing and segmentation techniques are applied to range images from laser scanners, which require a relatively long running time. Carlberg et al. [14] propose a segmentation method that uses triangulation to reconstruct surfaces. Alternatively, our online segmentation approach works on raw and unorganized large-scale point clouds.

Another research topic is scene parsing, which refers to the process of simultaneously classifying and segmenting objects in images (e.g., [12, 121, 126, 128]). They typically leverage the reconstructed point clouds from images and some contextual features to classify each pixel into ground, trees, buildings, etc. In particular, [128] further partitions a detected building object into sub-facades.

The problem of building modeling has received considerable attention in recent years, due largely to the increasing ubiquity of digital cameras and availability of active 3D sensors, such as LiDAR devices. Given the large body of work in this broad topic, it is beyond the scope of this dissertation to provide a comprehensive review. Interested readers are referred to two recent survey papers by Vanegas et al. [108] and Musialski et al. [78].

One topic of emphasis in recent research has been the reconstruction of 3D buildings from images or video sequences(e.g., [4, 113, 118]). One of the pioneering papers in this area introduces the Facade system [20], in which an operator interactively selects corresponding lines between images and building primitives (blocks etc.), and the system automatically estimates the camera pose and refines the primitives to fit the images. Today, the typical approach is to use structure from motion (SfM) techniques to estimate the camera motion as well as a set of sparse 3D points, followed by either user interaction(e.g., [95]) or dense stereo matching to generate the final model(e.g., [26]).

Alternatively the input can be 3D point clouds from LiDAR devices; the focus is usually to refine the scanned data and create a more usable mesh or parametric model(e.g., [28, 84]). This is also the main focus of our system. Recent papers in this

category often take advantage of the repetition and symmetric structures in buildings. Müller et al. developed an automatic approach for facade reconstruction with CGA shapes using ortho-rectified photos [76]. This method works well with highly regular and repetitive facades. Later the system was extended to allow regular photos that have perspective distortions [106]. Using 3D scans as input, the *SmartBoxes* technique focuses on high-rise buildings with many repetitive structural elements [79]. From a user-selected structural element, the system automatically finds similar copies in the input 3D point cloud. These repetitive elements are merged and refined, leading to a better final model. Automatic detection of symmetric or repetitive patterns has also been developed, in either 2D [92,117], 3D [10,81], or a combination of both [64]. They typically make a strong assumption about the layout of the symmetric or repetitive patterns, usually on a rectilinear grid, to optimize for facade processing. Similar assumptions have also been applied to interior reconstruction with great success [120]. Given our emphasis on residential houses, however, we do not make such an assumption. Li et al. [63] detects global regularities among primitives to better fit the scanned data, but they cannot handle completely missing surface patches like the roof area in our case. More recently, Vanegas et al. [107] presented a system to reconstruct Manhanttan-World Building masses from 3D range scans. Assuming the building is made of axis-aligned boxes, this system is able to produce water-tight building models in the presence of significant missing data. The main limitation is that it is unable to handle slanted surfaces.

In the area of automatic reconstruction on a large scale, approaches using ground-based data, either images (e.g., [26, 42, 119]) or 3D point clouds [28], usually focus

only on the facade, or whatever can be captured. On the other hand, approaches with airborne data generate 2.5D models, since only the roof is captured (e.g., [21,55, 56,83,129]. In both scenarios, few assumptions are made about the scene; therefore, they are usually more adaptable to different building/scene types. The downside is that the model is only in good quality from the viewpoint where it was originally captured. In our targeted setup – houses of a few stories, we can capture parts of both viewpoints, but neither is complete. In order to complete the model, we make a few common assumptions. These assumptions also allow us to provide semantic labeling for the final model. As we will demonstrate later, our approach is able to *handle significant amounts of missing data*, particularly in the roof area. There are approaches that combine both ground and aerial data sources (e.g. [29,48]) for large scale reconstruction, but our approach also benefits from more data coverage.

As pointed out by other researchers [78], it is difficult to directly compare these different reconstruction methods since they were all developed in different contexts with different emphases. Nevertheless, we have compared our results with two state-of-the-art algorithms. The first uses a piecewise-planar assumption about the scene geometry [17], which is also the foundation for our house reconstruction scheme. The second focuses on building 2.5D models for large scale reconstruction from aerial scans [130]. Both methods are automatic. The comparisons demonstrate the advantage of our approach for the task of reconstructing residential houses from ground-based 3D point data.

Chapter 3 Repetitive Pattern Mining and Segmentation from Images

In this chapter, I present a frequent structural pattern mining and segmentation method to discover recurring patterns as visual semantics in images. As digital cameras become popular to capture our real world, how to distill useful information from a vast image collection has been attracting ongoing interest in the fields of computer vision and data mining. Among many aspects of image understanding, the automatic identification of frequently visual patterns is a fundamental topic that serves as the foundations for high-level tasks such as data compression, pattern recognition, and visual information retrieval. Figure 1.1 shows a set of images, each of which contains repeated visual patterns. For example, the pattern of Starbucks logos in Figure 1.1(a) is a major characteristic of the picture. However, given the complex interplay of lighting and perspective, the same visual pattern (e.g., a Starbucks logo) can have dramatically different pose and appearance changes. The general approach to address this problem is to extract repetition from low-level visual items (words). Though visual items, each of which corresponds to similar SIFT features, are typically invariant or less sensitive to lighting or perspective changes, they are limited to small local image patches. Composing complex patterns from these visual items that represent high-order structural meanings is very difficult, if not impossible. We present a frequent structural pattern mining method to discover recurring patterns in images (e.g., the left image in Figure 3.12), and we further explore to address the segmentation problem to get more semantically meaningful representation of visual

patterns (e.g., the right image in Figure 3.12). Most existing approaches to mining visual patterns rely on a preprocessing step that collects image features into a transactional database, where traditional pattern mining algorithms can then be applied. For example, to find patterns from images such as those shown in Figure 1.1, Yuan et al. [124] extract sample patches from the image. Each patch is then converted into a transaction, and visual features within the patch become items in the transaction, allowing the application of frequent itemset mining.

Though promising, this method is limited by a number of factors. First, the locations, sizes and shapes of visual patterns may vary. Without any prior information, it is difficult to automatically determine the appropriate locations and sizes of sampled patches. One possible solution is dense sampling at all possible locations and scales. However, it would artificially increase the occurrences of features in the resulting transactional database, making it difficult to determine whether the frequency of a discovered pattern is due to over-sampling or its true presence in the image. Second and most importantly, patterns detected by transactional pattern mining (such as [102, 103, 124]) remove structural information which encodes crucial semantics of image patterns. As an analogy, the set of letters $\{m, a, n\}$ occurring frequently in a text does not necessarily imply frequent occurrences of the word "man" — the same set of letters may also occur in the words "name", "Maine", "main" and so on. Analogous to the aforementioned problem, different compositions of co-occurring visual items might have vastly different spatial relationships with image semantics. Extracting visual patterns from unstructured data, in particular, images, a set of visual items in a visual pattern not only co-occur, but also maintain consistent spatial

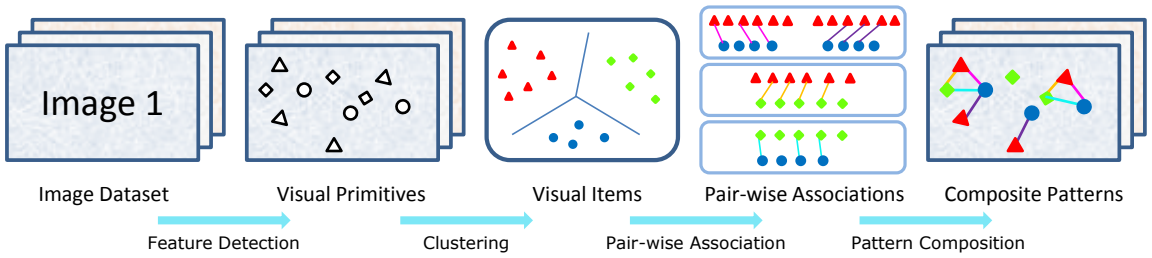


Figure 3.1: A pipeline for the extraction of structural visual patterns from images.

relationships for encoding more structural meanings.

To preserve structural relationships of features in images, one intuitive solution is to treat an image as a graph where each vertex represents an image feature and each edge carries the spatial relationship between two features. Consequently, the frequent visual patterns look like frequent subgraphs. However, the discovery of such visual patterns challenges existing subgraph mining algorithms in the following two aspects. First, traditional subgraph mining is limited to categorically labeled graphs; it is non-trivial to convert the numerical weights of edges representing the relationships between features in images into discrete labels. Second, the obtained graph is usually dense since spatial relationships exist between any pair of features. Realizing subgraph isomorphism is NP-complete, it is computationally intractable to perform subgraph mining on a large image collection (typically with tens of thousands of features).

Our approach starts by cataloguing the visual features of images into a set of visual items. Next, strong associations between two classes of visual features evidenced by consistent and frequent geometric relationships are identified. Finally, these pairwise associations are used to compose more complex and complete structural patterns

present in the images. With the assumption that frequently occurring objects are visually consistent across multiple images with varying backdrops, we are able to further segment visual patterns automatically. The overview of the pipeline is shown in Figure 3.1.

3.1 Preliminary

An image dataset can be represented by a set of visual features, such as SIFT features [73], extracted from salient image patches. We call each feature a visual ***primitive***, denoted as $\mathbf{f}$. For example, in Figure 3.2, every red dot in the middle figure corresponds to an extracted visual feature and hence is recognized as a visual primitive. A primitive defined on an image patch can be represented by a vector $\mathbf{f}_i = [\mathbf{x}_i, s_i, \theta_i, \mathbf{d}_i]$, where 2D vector $\mathbf{x}_i$ is the patch centroid, s_i is the scale of the patch, θ_i is the patch orientation, and the 128-dimensional vector $\mathbf{d}_i$ encodes the appearance of the patch.

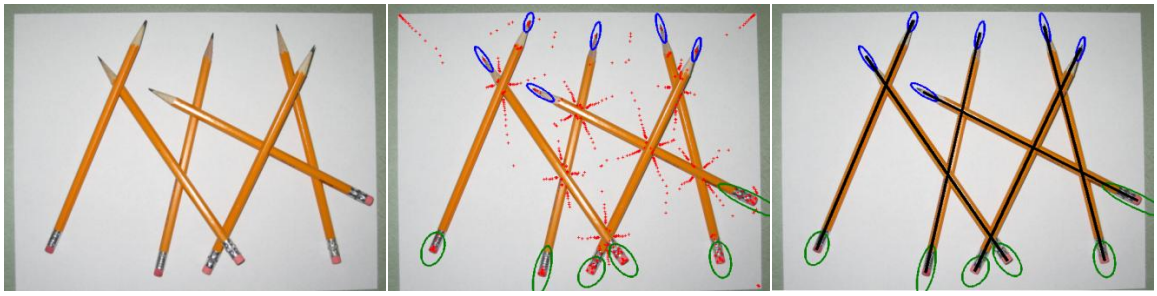


Figure 3.2: An illustration of frequent visual patterns. The left image contains pencils in different poses. The middle one depicts the detected visual primitives centered in red dots, with blue and green ellipses representing the two dominant visual items. The right image reflects the association between the two items.

Primitives in an image have structural relations. More often than not, these

relations encode important structural semantics of image patterns. For example, the primitive locates at the tip of a pencil and the primitive locates at the eraser of the pencil bear a structural relation that implies the presence of the pencil. For two primitives $\mathbf{f}_i$ and $\mathbf{f}_j$, we represent their spatial relationship as the ***relation*** function $\mathbf{l}(\mathbf{f}_i, \mathbf{f}_j)$. As discovered in [15, 43], $\mathbf{l}(\mathbf{f}_i, \mathbf{f}_j)$ can be defined as a 4D vector $[D(\mathbf{f}_i, \mathbf{f}_j)$, $S(\mathbf{f}_i, \mathbf{f}_j)$, $H(\mathbf{f}_i, \mathbf{f}_j)$, $H(\mathbf{f}_j, \mathbf{f}_i)]$, where $D(\mathbf{f}_i, \mathbf{f}_j)$ is the relative spatial distance between $\mathbf{f}_i$ and $\mathbf{f}_j$ (Eq. 3.1a), $S(\mathbf{f}_i, \mathbf{f}_j)$ is their relative scale difference (Eq. 3.1b), $H(\mathbf{f}_i, \mathbf{f}_j)$ is the heading from $\mathbf{f}_i$ to $\mathbf{f}_j$ (Eq. 3.1c), and $H(\mathbf{f}_j, \mathbf{f}_i)$ is the heading from $\mathbf{f}_j$ to $\mathbf{f}_i$ (Eq. 3.1d):

$$D(\mathbf{f}_i, \mathbf{f}_j) = \|\mathbf{x}_i - \mathbf{x}_j\|_2 / \sqrt{s_i^2 + s_j^2} \tag{3.1a}$$

$$S(\mathbf{f}_i, \mathbf{f}_j) = (s_i - s_j) / \sqrt{s_i^2 + s_j^2} \tag{3.1b}$$

$$H(\mathbf{f}_i, \mathbf{f}_j) = \Delta_\theta(\arctan(\mathbf{x}_i - \mathbf{x}_j) - \theta_i) \tag{3.1c}$$

$$H(\mathbf{f}_j, \mathbf{f}_i) = \Delta_\theta(\arctan(\mathbf{x}_j - \mathbf{x}_i) - \theta_j), \tag{3.1d}$$

where function $\Delta_\theta(\cdot) \in [-\pi, +\pi]$ calculates the principal angle. The representation is invariant to translation, scaling and rotation, and robust to small distortion.

Visual primitives can be further cataloged into visual ***items*** based on their similarity of appearance. Using k-means clustering [72], for example, we can identify two dominant visual items from all the visual primitives in Figure 3.2, corresponding to the tips and erasers of the pencils respectively.

Given two visual items α and β, let $\mathcal{I}_\alpha$ and $\mathcal{I}_\beta$ be the primitive sets of item α and β, respectively. Let $\mathcal{I}_x$ be a subset of primitives in $\mathcal{I}_\alpha$, and let $\mathcal{I}_y$ be a subset of primitives in $\mathcal{I}_\beta$, i.e., $\mathcal{I}_x \subseteq \mathcal{I}_\alpha$ and $\mathcal{I}_y \subseteq \mathcal{I}_\beta$. Let $\psi(\mathcal{I}_x, \mathcal{I}_y)$ be a set of relations each of which connects one primitive in $\mathcal{I}_x$ and one primitive in $\mathcal{I}_y$ and no two relations connect

a common primitive. We say $\psi(\mathcal{I}_x, \mathcal{I}_y)$ is an ***association*** between $\mathcal{I}_x$ and $\mathcal{I}_y$, if relations in $\psi(\mathcal{I}_x, \mathcal{I}_y)$ are consistent (defined in Section 3.2). In Figure 3.2, there exist consistent spatial relations, such as distances and relative directions, between every tip primitive and the corresponding eraser primitive on the same pencil. Therefore, there exists an association between the two visual items — the tips and the erasers.

It is worth mentioning that two visual items might have more than one association. In face images, for example, there exist two associations between eyes and noses because the association between the left eye and nose is different from the association between the right eye and nose in terms of the angle. It is important to find all maximal distinct associations between a pair of visual items. We address the problem of mining maximal pairwise associations in Section 3.2.

With a subset of visual primitives $F = \{\mathbf{f}_{(1)}, \mathbf{f}_{(2)}, \cdots, \mathbf{f}_{(p)}\}$, a visual ***pattern*** instance $P = \langle F, R \rangle$ is defined by F and a set of relations R between any pair of primitives in F, $R = \{\mathbf{l}(\mathbf{f}_i, \mathbf{f}_j) | \mathbf{f}_i, \mathbf{f}_j \in F\}$. Given two pattern instances $P_1 = \langle F_1, R_1 \rangle$ and $P_2 = \langle F_2, R_2 \rangle$, we say the two instances are *identical* if there exists a one-to-one mapping $\mathbf{g} : F_1 \mapsto F_2$, such that

1. $\forall \mathbf{f}_i \in F_1$, $\mathbf{g}(\mathbf{f}_i) \in F_2$, and $\mathbf{g}(\mathbf{f}_i)$ belongs to the same visual item as $\mathbf{f}_i$;

2. $\forall \mathbf{f}_i, \mathbf{f}_j \in F_1$, $i \neq j$, $\mathbf{l}(\mathbf{f}_i, \mathbf{f}_j)$ and $\mathbf{l}(\mathbf{g}(\mathbf{f}_i), \mathbf{g}(\mathbf{f}_j))$ belong to the same association.

A pattern is *frequent*, if there exist at least δ identical pattern instances in the images. The method developed for mining all structural patterns is presented in Section 3.3.

In the example of Figure 3.2, the association between the visual item of tips and the visual item of erasers forms a pencil in the figure. Every pencil is a pattern

instance. Please note that this example is a simple pattern consisting of only two items. The method proposed in this chapter can be generally applied to finding patterns composed of an arbitrary number of items if applicable.

3.2 Mining Maximal Pairwise Associations

In this section, we present an algorithm that looks for associations between a pair of visual items.

3.2.1 Maximal Association Subgraph

Given two sets of primitives $\mathcal{I}_\alpha$ of item α and $\mathcal{I}_\beta$ of item β, relations between the pairs of primitives can be represented by a weighted complete bipartite graph $G = \langle U, V, E, \omega \rangle$. In this graph, every vertex $u \in U$ represents a primitive in $\mathcal{I}_\alpha$ and every vertex $v \in V$ represents a primitive in $\mathcal{I}_\beta$. E is the set of edges between nodes in U and V. Every edge $e \in E$ represents a relation between a primitive in $\mathcal{I}_\alpha$ and a primitive in $\mathcal{I}_\beta$. A weight function ω, equivalent to the relation function $\mathbf{l}$, computes the geometric relationship between two primitives. Without loss of generality, we assume all edges can be sorted into descending or ascending orders given weight ω, such as the relative spatial distance D.

Within a bipartite graph G, a matching $M \subseteq E$ is a set of edges, none of which connects the same vertices. Let $U_M \subseteq U$ and $V_M \subseteq V$ be the sets of vertices incident to edges in M. We call $G_M =< U_M, V_M, M, \omega >$ the induced subgraph of G by the matching M. A matching-induced subgraph $G_M \subseteq G$ is a ***maximal association subgraph*** if it is:

1. **weight-consistent:** $MSE(G_M) \leq \epsilon$,

$$MSE(G_M) = \frac{1}{|M|} \sum_{e \in M} (\omega(e) - \omega(\hat{e}))^2,$$

where $\hat{e} = \arg\min_{e \in M} \omega(e)$;

2. **maximal:** there exists no G'_M, s.t. $G_M \subset G'_M$ and G'_M is weight-consistent.

The first condition requires that the relations in an association are consistent so that the weights of all the edges in M should not deviate much from $\omega(\hat{e})$, where $\hat{e}$ is a representative edge of M. The deviation is measured by the mean of squared errors (MSE) between the weight of $\hat{e}$ and the rest edges in M. The weight of the representative edge within a matching, $\omega(\hat{e})$, can be the mean of all the weights, or the minimum or maximum weight. In our case, since the consistency among all the weights within a matching is of a bigger concern than a centered Gaussian distribution of all the weights, we choose the least weighted edge as the representative edge. The second condition indicates that the identified association has to be maximal, i.e., no additional relation can be added to increase the size of the association while preserving the consistency. If G_M is a maximal association subgraph, the set of relations represented by the edges in M can be acknowledged as a candidate association between the visual items α and β. M is called a candidate association matching. The acknowledged association associates the primitives represented by vertices in U_M to the corresponding primitives represented by vertices in V_M. The cardinality of M, equal to the number of relations within the association, is therefore the frequency of the association.

Algorithm 1 Pairwise Association Mining

```
1: W ← ∅, M ← ∅
2: sort E_{i,j} based on ω(e), e ∈ E_{i,j}
3: for e ← arg min_{e∈E_{i,j}} ω(e) do
4:     W ← W ∪ {e}
5:     M' ← maximum matching of G_W
6:     if |M'| ≥ |M| then
7:         if M' does not satisfy consistency then
8:             if M satisfies consistency then
9:                 M is a candidate association matching
10:            end if
11:        end if
12:        M ← M'
13:        while M does not satisfy consistency do
14:            remove ẽ ← arg min_{e∈W} ω(e) from W
15:            if ẽ ∈ M then
16:                M ← maximum matching of G_W
17:            end if
18:        end while
19:    end if
20:    E = E\{e}
21: end for
```

3.2.2 Algorithm

It is important to note that the maximal association problem is different from the conventional maximum cardinality bipartite matching problem in bipartite graph [114], where one maximum matching is found between two sets of nodes. We are looking for a set of maximal weight-consistent matchings and the induced maximal association subgraphs, representing all possible associations between two sets of visual primitives. For example, we are interested in identifying both the association between the left eye and nose and the association between the right eye and nose, instead of a merged association between the eyes and nose with the maximum cardinality though. As a result, the number of putative associations that need to be evaluated

is the number of all possible subgraph matchings in G.

A naive way to find the maximal pairwise association subgraphs is to enumerate all potential subgraphs and identify the ones that satisfy the constraints. However, this approach is intractable, since there exist $\sum_{i=1}^{|U|} \frac{|V|!}{i!}$ possible subgraphs assuming $|U| \leq |V|$. Instead, we develop a polynomial-time algorithm to address this problem.

Our algorithm first sorts the edges in the increasing order of their weights. It then operates by moving a dynamic window across the sorted edges and looking for maximal association subgraphs embedded in the window. Starting empty with no edges, the window W grows by alternating the following two operations iteratively.

- **Adding a new edge:** At each step, the least weighted unvisited edge e is added into the window, and the window becomes $W' = W \cup \{e\}$. We need to determine whether the current maximum matching M_W is still maximum after adding new edge. The problem is equivalent to finding an augmenting path with respect to M_W.

  A vertex in $G_{W'}$ is called unmatched if the vertex is not incident to any edge in M_W. An *augmenting path* is a path that begins and ends on unmatched vertices, while its edges belong alternatively to M_W and not to M_W. If there exists an augmenting path p from v to u with respect to the existing matching M_W, a new maximum matching $M_{W'}$ can be formed by tracking the precedence of u till the other unmatched vertex v, and replacing the edges of p that are in M_W with those not in M_W [9]. The cardinality of the maximum matching $M_{W'}$ grows by one.

The window keeps growing by adding new edges as long as the obtained maximum matching satisfies the weight-consistent property. If the weight-consistent property doesn't hold for the maximum matching $M_{W'}$ in the current window W' after a new edge is added, a local maximum is reached. The subgraph induced by M_W will be acknowledged as a candidate maximal association subgraph.

- **Deleting an edge:** In order to get out of local maximum and to discover new associations, the edge with the smallest weight in the current window will be deleted. If the removed edge is part of the current maximum matching, the current matching will be reduced. To avoid missing other matchings, a new search will be conducted to find potential matching with higher cardinality. The least weighted edges will be deleted continuously until the obtained maximum matching fulfills the weight-consistent property. Then the window will try to add new edges again.

Intuitively, the window growing period corresponds to the process of extending an association into a maximal one, while the shrinking period is the process when the window is shifting into a different association. These two processes alternate, making the window wriggle through the edge list.

After all the edges have been visited, the algorithm terminates and returns all the candidate associations, as presented in Algorithm 1.

Lemma 3.2.1. *The subgraph induced by a candidate association matching is a maximal association subgraph.*

Proof. We prove this lemma by contradiction. First, a matching will not be acknowledged as a candidate association matching until the weight-consistent condition is broken. Assume a candidate association matching M does not hold for the maximal property, then there is an edge $\hat{e}$ such that $M \cup \{\hat{e}\}$ is still a matching and $MSE(M \cup \{\hat{e}\}) \leq \epsilon$. Let $\tilde{e}$ be the edge with the smallest weight in M, $\tilde{e} = \arg\min_{e \in M} \omega(e)$. Find an edge e' such that e' is the edge with the smallest weight among all the edges in the set $\{e | \omega(e) \geq \omega(e_m), e \cap e_m = \emptyset, \forall e_m \in M\}$. Then $\omega(e')$ must be less than or equal to $\omega(\hat{e})$. So we have

$$
\begin{aligned}
& MSE(M \cup \{e'\}) \\
= \; & \frac{1}{|M|+1}\Big(\sum_{e \in M}(\omega(e) - \omega(\tilde{e}))^2 + (\omega(e') - \omega(\tilde{e}))^2\Big) \\
\leq \; & \frac{1}{|M|+1}\Big(\sum_{e \in M}(\omega(e) - \omega(\tilde{e}))^2 + (\omega(\hat{e}) - \omega(\tilde{e}))^2\Big) \\
= \; & MSE(M \cup \{\hat{e}\}) \\
\leq \; & \epsilon.
\end{aligned}
$$

Therefore M shouldn't be acknowledged as a candidate association matching, because e' could be added to make a larger matching without breaking the weight-consistent property, which is a contradiction. So the subgraph induced by M is weight-consistent and maximal, which leads to the conclusion that the induced subgraph is a maximal association subgraph. □

Lemma 3.2.2. *Assume the maximum matching of G_W is unique for any G_W, where G_W is the subgraph of G induced by window W. Then the algorithm will find all candidate association matchings.*

Proof. As indicated in the algorithm, the window W_M grows as long as the maximum matching M within it holds the weight-consistent property. When this condition is broken due to a new incoming edge e', there does not exist an edge $\hat{e}$, where $\omega(\hat{e}) \geq \omega(e')$, such that $M' = M \cup \{\hat{e}\}$ is a weight-consistent maximum matching according to the proof of Lemma 3.2.1. Then M is a candidate association matching.

In the process of removing edges, the window will keep removing the edge having the smallest weight in the window, until the maximum matching within it meets the weight-consistent property. Similarly, when W_M has dropped e', there exists no $\hat{e}$, where $\omega(\hat{e}) \leq \omega(e')$, such that $M' = M \cup \{\hat{e}\}$ is a weight-consistent maximum matching. So the deleting operation will not cause any loss of candidate association matchings as well. □

With Lemma 3.2.1 and Lemma 3.2.2, it is straightforward to conclude that the algorithm can find all maximal association subgraphs, if the subgraph induced by a window has a unique maximum matching for every window.

3.2.3 Complexity

The algorithm visits each edge at most twice, one for adding it into the current window while the other for removing it from the window. When an edge e is added to the current window, a new maximum matching $M_{W\cup\{e\}}$ for $G_{W\cup\{e\}}$ will be searched. This can be done by finding an augmenting path within $G_{W\cup\{e\}}$ with respect to M_W [9]. The time cost for one such step is equivalent to a breadth-first search in $G_{W\cup\{e\}}$, with time complexity $O(|W|)$, where $|W|$ is the number of edges in the

window. When removing an edge e, the time cost is O(1) if e is not in the current maximum matching M_W. Otherwise, augmenting path will also be searched in the remaining window $W\backslash\{e\}$, with respect to the remaining matching $M_W\backslash\{e\}$. So it costs O($|W|-1$) time. Overall, adding or dropping an edge from W triggers the search for the maximum matching which costs O($|W|$) time. For each edge, it will be added and dropped exactly once. Therefore, the total cost for mining association subgraphs is O($|E|^2$) time.

3.3 Pattern Composition

Following the identification of pairwise associations among all visual items, we propose methods to build more complex structural patterns from pairwise associations.

3.3.1 Frequent Subgraph Mining

Intuitively, after clustering visual primitives into visual items and discovering pairwise associations between any two visual items, we can construct a graph $G = (V, E)$, where the set of vertices V corresponds to all detected visual primitives in images; and the set of edges E contains a pair (v_i, v_j), where $v_i \in V$ represents primitive $\mathbf{f}_i$, $v_j \in V$ represents primitive $\mathbf{f}_j$, $\mathbf{f}_i$ and $\mathbf{f}_j$ are from the same image and the link $\mathbf{l}(\mathbf{f}_i, \mathbf{f}_j)$ belongs to an association between the visual item of $\mathbf{f}_i$ and the visual item of $\mathbf{f}_j$. Therefore, we can further label the graph $G = (V, E)$; that is, each vertex is labeled with the visual item it belongs to and each edge is labeled with its corresponding association. Two connected subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic, if there exists a one-to-one and onto mapping π from G_1 to G_2 such that vertex labels of $\pi(V_1)$

and vertex labels of V_2 are equal, and edge labels of $\pi(E_1)$ and edge labels of E_2 are equal; notice that, this definition is exactly the same as the definition of identical pattern instances introduced in Section 3.1. In this way, the problem of frequent visual patterns mining in images becomes a problem of finding frequent subgraphs in the labeled graph G. Frequent subgraph mining algorithms, e.g., [52, 75, 85], can be applied to such graphs for resolving recurring patterns. Here, we briefly summarize the general framework of frequent subgraph mining in Algorithm 2. The algorithm stats by enumerating all frequent single edge subgraphs, i.e., pairwise associations, and then iteratively proceeds as follows: generate all candidate size $(k+1)$ subgraphs by adding one additional edge connecting to each size k subgraph (function *Generate*); compute their frequency by performing the subgraph isomorphism tests (function *Count*); and the candidate subgraphs whose frequency is at least δ are kept for the next level.

Algorithm 2 Frequent Subgraph Mining

```
1: F^1 ← all frequent size-1 subgraphs in G
2: F ← F^1
3: k ← 1
4: while F^k ≠ ∅ do
5:     C^{k+1} ← Generate(F^k, δ)
6:     F^{k+1} ← ∅
7:     for each C in C^{k+1} do
8:         freq ← Count(C, C^{k+1})
9:         if freq ≥ δ then
10:            add C to F^{k+1}
11:        end if
12:    end for
13:    F ← F ∪ F^{k+1}
14:    k ← k + 1
15: end while
16: return F
```

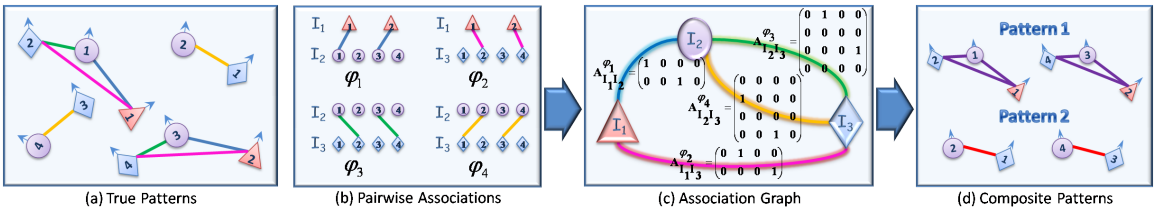
However, this approach might not be feasible for pattern mining from images: first, it is time-consuming due to the inherent complexity of exhaustive subgraph mining algorithms though many speed-up procedures can be applied; second, frequent subgraph mining typically generates a large number of overlapping patterns, making it difficult to interpret the results. A single image can have tens of thousands of extracted visual primitives. The set of possible relations grows exponentially as the number of primitives increasing. Therefore, in order to perform pattern mining on images, it is crucial to address the issue on scalability while preserving the capability of discovering dominant recurring patterns.

3.3.2 A Polynomial-time Heuristic Algorithm

We propose a heuristic method to construct frequent structural patterns based on the pairwise associations among items. Our method aims at discovering dominant frequent patterns, resolving their compositions of primitives and significantly reducing the time complexity of pattern mining in large graphs.

Association Graph

In this section, we define an *association matrix* for every pairwise association in order to capture association information between items. Suppose two primitive sets $\mathcal{I}_\alpha$ of item α and $\mathcal{I}_\beta$ of item β have n_α and n_β primitives respectively, and let φ denote an association between α and β. Let $\mathbf{f}_\alpha^i$ be the ith primitive in $\mathcal{I}_\alpha$ and $\mathbf{f}_\beta^j$ be the jth



(a) True Patterns (b) Pairwise Associations (c) Association Graph (d) Composite Patterns

Figure 3.3: An example of pattern composition. The shape of a vertex represents an item, including triangle($\mathcal{I}_1$), circle($\mathcal{I}_2$) and diamond($\mathcal{I}_3$). The number i inside a vertex refers to the i^{th} primitive of the corresponding item. The pairwise associations are represented by colored edges, blue(φ_1), pink(φ_2), green(φ_3), and yellow(φ_4). In (a), each node represents a primitive labeled by the index of the item that it belongs to, and frequent patterns are exhibited by frequent subgraphs in the figure. Based on the consistent pairwise associations shown in (b), the association graph $\mathcal{G}$ can be constructed as (c), where every vertex represents a visual item and every edge represents a pairwise association. Associated with each edge is the corresponding association matrix. The problem of finding a visual pattern is converted to the search for a maximal path with the support no smaller than the required minimum frequency. One maximal path in this graph is composed of the blue edge, green edge and pink edge. Starting from $\mathcal{I}_1$ and φ_1, the support of this path can be calculated by $\|(1,1)\cdot\mathcal{A}^{\varphi_1}_{\mathcal{I}_1,\mathcal{I}_2}\cdot\mathcal{A}^{\varphi_3}_{\mathcal{I}_2,\mathcal{I}_3}\cdot(\mathcal{A}^{\varphi_2}_{\mathcal{I}_1,\mathcal{I}_3})^T\|_1 = \|\theta_{\mathcal{I}_1}\| = \|(1,1)\|_1 = 2$. Therefore this pattern is supported by 2 instances. The exact composition of primitives can be recovered through following operations: $\theta_{\mathcal{I}_3} = \theta_{\mathcal{I}_1}\cdot\mathcal{A}^{\varphi_2}_{\mathcal{I}_1,\mathcal{I}_3} = (0,1,0,1)$, i.e., $\{1,2\}$ in $\mathcal{I}_1$ connect $\{2,4\}$ in $\mathcal{I}_3$ respectively; $\theta_{\mathcal{I}_2} = \theta_{\mathcal{I}_3}\cdot(\mathcal{A}^{\varphi_3}_{\mathcal{I}_2,\mathcal{I}_3})^T = (1,0,1,0)$, i.e., $\{2,4\}$ in $\mathcal{I}_3$ connect $\{1,3\}$ in $\mathcal{I}_2$ respectively. Another maximal path would be obtained from the yellow edges in (a), which is supported by 2 instances as well.

primitive in $\mathcal{I}_\beta$. An association matrix $\mathcal{A}^{\varphi}_{\alpha,\beta}$ is an $n_\alpha \times n_\beta$ matrix with entries

$$a_{i,j} = \begin{cases} 1 & , \mathbf{l}(\mathbf{f}^i_\alpha, \mathbf{f}^j_\beta) \in \varphi; \\ 0 & , \text{otherwise.} \end{cases}$$

where $\mathbf{l}(\mathbf{f}^i_\alpha, \mathbf{f}^j_\beta)$ is the relation between $\mathbf{f}^i_\alpha$ and $\mathbf{f}^j_\beta$. It is easy to prove that an association matrix $\mathcal{A}^{\varphi}_{\beta,\alpha} = (\mathcal{A}^{\varphi}_{\alpha,\beta})^T$.

To indicate which primitives are involved in the association, we also define an *association vector* $\theta_{\mathcal{I}_\alpha}$ for an item $\mathcal{I}_\alpha$, which is a n_α-dimensional binary row vector with 1's for primitives having the association and 0's for others. Assume φ is an association between $\mathcal{I}_\alpha$ and $\mathcal{I}_\beta$. Given the association vector $\theta_{\mathcal{I}_\alpha}$ of item $\mathcal{I}_\alpha$, the primitives of

$\mathcal{I}_\beta$ that are contained in φ can be determined by calculating its association vector $\theta_{\mathcal{I}_\beta} = \theta_{\mathcal{I}_\alpha} \cdot \mathcal{A}^{\varphi}_{\alpha,\beta}$.

With association matrices, we can abstract the original graph into a much more compact graph which is named as the *association graph*, $\mathcal{G} = \langle V_\mathcal{G}, E_\mathcal{G} \rangle$. Every node in the vertex set $V_\mathcal{G}$ represents an item, and every edge in the edge set $E_\mathcal{G}$ corresponds to an association between two items and is assigned by the corresponding association matrix. Since there might be more than one association for two visual items, the association graph is therefore an undirected multi-graph that allows multiple edges between two vertices.

Pattern Construction

Let $e^{\varphi}_{\mathcal{I}_i,\mathcal{I}_j}$ be an edge in the association graph $\mathcal{G}$ that represents the association φ between $\mathcal{I}_i$ and $\mathcal{I}_j$. For a path $p = e^{\varphi_1}_{\mathcal{I}_1,\mathcal{I}_2} e^{\varphi_2}_{\mathcal{I}_2,\mathcal{I}_3} \cdots e^{\varphi_{m-1}}_{\mathcal{I}_{m-1},\mathcal{I}_m}$ in $\mathcal{G}$, p describes an association pattern among items $\mathcal{I}_1, \mathcal{I}_2, \cdots, \mathcal{I}_m$. The frequency of the pattern is the number of its instances, determined by the *support* of the path p, which is defined as the number of primitives that are involved in this pattern within each item on the path p. Mathematically, we can calculate the association vector of $\mathcal{I}_m$ by

$$\theta_{\mathcal{I}_m} = \theta_{\mathcal{I}_1} \mathcal{A}^{\varphi_1}_{\mathcal{I}_1,\mathcal{I}_2} \mathcal{A}^{\varphi_2}_{\mathcal{I}_2,\mathcal{I}_3} \cdots \mathcal{A}^{\varphi_{m-1}}_{\mathcal{I}_{m-1},\mathcal{I}_m}, \tag{3.2}$$

and then get the support of p by $support(p) = \|\theta_{\mathcal{I}_m}\|_1$.

Starting from an edge $e^{\varphi_i}_{\mathcal{I}_i,\mathcal{I}_j}$ in the association graph $\mathcal{G}$, such as the edge representing the most frequent association, we extend this edge into a maximum path whose support is greater than the required minimum frequency of a pattern. Initiating with

$p^{(1)} = e^{\varphi_1}_{\mathcal{I}_1,\mathcal{I}_2}$, the algorithm gives a path of length i, $p^{(i)} = e^{\varphi_1}_{\mathcal{I}_1,\mathcal{I}_2} e^{\varphi_2}_{\mathcal{I}_2,\mathcal{I}_3} \cdots e^{\varphi_i}_{\mathcal{I}_i,\mathcal{I}_{i+1}}$ at the i-th step. We can compute the association vector of item $\mathcal{I}_{i+1}$ for this pattern, $\theta_{\mathcal{I}_{i+1}}$ using Equation (3.2) by initiating $\theta_{\mathcal{I}_1}$ as $\mathbf{1}$, a row vector with all the entries of 1. For every unvisited edge $e^{\varphi_{i+1}}_{\mathcal{I}_{i+1},\mathcal{I}_j}$ that is incident from the vertex of $\mathcal{I}_{i+1}$, we calculate the 1-norm $\left\|\theta_{\mathcal{I}_{i+1}} \cdot \mathcal{A}^{\varphi_{i+1}}_{\mathcal{I}_{i+1},\mathcal{I}_j}\right\|_1$, which is the resulted support if $e^{\varphi_{i+1}}_{\mathcal{I}_{i+1},\mathcal{I}_j}$ is appended to $p^{(i)}$. The algorithm then constructs $p^{(i+1)}$ by greedily picking an unvisited edge with the largest value of resulted supports and appending it to $p^{(i)}$.

To extract dominant patterns from $\mathcal{G}$, the process is similar to a depth-first traverse of all the edges instead of vertices. A priority queue is maintained to provide the next "best" edge as the starting edge based on their support. Once a pattern is found, all edges contained in the path will be removed from the queue. The algorithm will continue until the queue becomes empty. An example of this composition process is shown in Figure 3.3.

Complexity

Regarding the computational complexity, the algorithm traverses all the edges in $E_{\mathcal{G}}$ to compose a pattern in the worst case, with a time cost of $O(|V_{\mathcal{G}}| + |E_{\mathcal{G}}|)$. Such composing procedures need to be performed at most $|E_{\mathcal{G}}|$ times. Therefore the time complexity of the pattern composition is $(|E_{\mathcal{G}}|^2 + |V_{\mathcal{G}}||E_{\mathcal{G}}|)$.

The proposed pattern composition algorithm significantly improves the complexity by reducing

1. **Scale of graph vertex set:** The vertex set of the association graph, $V_{\mathcal{G}}$, is

the set of all visual items, rather than the set of all primitives;

2. **Scale of graph edge set:** The edge set of the association graph, $E_{\mathcal{G}}$, consequently becomes the set of all pairwise associations, rather than the set of all primitive-primitive relations;

3. **Complexity:** The time complexity of the algorithm is polynomial to the size of the vertex set and the edge set, instead of exponential.

The exact primitive structures are obtained simultaneously through the operations on association vectors/matrices.

3.4 Segmentation of Frequent Occurring Patterns from Images

Given the extracted frequent appearing visual patterns, one trivial segmentation method is to use the bounding box or convex hull of the key features that make up the visual patterns. However, this is unlikely to achieve satisfactory results for complex shapes.

We define "foreground" as a recurring target object, and "background" as the rest unrelated image regions. Foreground-background segmentation is equivalent to computing an optimal binary labeling map Z_i for image X_i, where each element $z_i(\mathbf{p}) \in \{0, 1\}$ indicates whether pixel $\mathbf{p} \in X_i$ belongs to foreground ($z_i(\mathbf{p}) = 1$) or to background ($z_i(\mathbf{p}) = 0$); that is the binary Markov Random Field (MRF) minimization problem. However, segmenting every pattern instance from all images simultaneously needs to optimize a system with a huge number of unknowns, i.e., the total number of pixels in the image dataset. In order to make this problem

tractable, we focus on segmenting one image at a time using information received from the remaining images. Our approach starts from establishing dense correspondences between any pair of images. The next step computes a cost (color variance) map $\mathcal{C}_i$ for image X_i with the help of estimated dense correspondences between X_i and any other image. It is reasonable to expect that pixels capturing the same portion of visual patterns are visually consistent, i.e., with low color variance; in contrast, the color variance of the pixels that come from different backdrops is usually high. Incorporating this cue, we then formulate an MRF energy minimization framework to segment pattern instances. Finally, we adopt the well-known graph-cut approach [11] to solve the problem.

3.4.1 Homography-based Dense Correspondences Establishment

When the target recurring object is planar or its depth variation is small compared to the distance to the camera's center of projection, the target object can be approximately modeled by a planar surface. We can simply use homography transformation to represent the dense correspondences between the instances of the target object documented in different images. Given images X_i and X_j, suppose the pattern instance $P_i \subset X_i$ and $P_j \subset X_j$, we can estimate the homography H_{ij} through the SIFT features only around P_i and P_j without searching the entire image space. The outliers can be removed by RANSAC [25] on a relatively small portion of image features. In this way, we can compute the dense correspondences between any pair of images X_i and X_j. To avoid the exhaustive pair-wise alignments, we can first randomly select a reference image X_r, and then find the correspondences from image X_r to

X_j given $H_{jr} = H_{rj}^{-1}$. Thus, the transformation from X_i to X_j can be computed as $H_{ij} = H_{rj}H_{ir} = H_{rj}H_{ri}^{-1}$. In practice, performing $(n-1)$ homography estimations from one reference image, where n is the number of images, is sufficient to get all dense correspondences between any pair of images.

3.4.2 Variance Map Generation

Given an image X_i and a set of correspondences between X_i and any other image, we then propose to generate a cost map $\mathcal{C}_i$, where each element $c_i(\mathbf{p}_i)$ corresponds to visual dissimilarity measure of the series consisting of pixel $\mathbf{p}_i$ in image X_i and its matched pixels in other images.

To compute the cost $c_i(\mathbf{p}_i)$ of pixel $\mathbf{p}_i$ in image X_i, suppose we find $\mathbf{p}_i$'s correspondences $\mathbf{p}_1$ in image X_1, $\mathbf{p}_2$ in image X_2, and up to $\mathbf{p}_m$ in image X_m; we can simply compute $c_i(\mathbf{p}_i)$ as the color variance of the series:

$$c_i(\mathbf{p}_i) = \mathrm{Var}\left(X_i(\mathbf{p}_i), X_1(\mathbf{p}_1), \cdots X_m(\mathbf{p}_m)\right), \tag{3.3}$$

where $X_j(\mathbf{p}_j)_{1\leq j\leq m}$ is the color of the pixel $\mathbf{p}_j$ in image X_j. However, to be more robust to image noise and matching errors, we use the approximate matching strategy similar to [53] to compute $\mathcal{C}_i$, that is instead of selecting $\mathbf{p}_i$'s exact matched pixel $\mathbf{p}_j$ in image X_j, we search the $\mathbf{p}_j$'s neighboring pixel $\mathbf{q}_j$, where color $X_j(\mathbf{q}_j)$ is closest to the reference color $X_i(\mathbf{p}_i)$, and efficiently compute the color variance in a greedy manner as:

$$\mathbf{q}_j = \underset{\mathbf{q}_j\in N(\mathbf{p}_j)}{\arg\min} \left|X_j(\mathbf{q}_j) - X_i(\mathbf{p}_i)\right|, \quad j \neq i, \tag{3.4}$$

$$c_i(\mathbf{p}_i) = \mathrm{Var}\left(X_i(\mathbf{p}_i), X_1(\mathbf{q}_1), \cdots, X_m(\mathbf{q}_m)\right), \tag{3.5}$$

where $N(\mathbf{p}_j)$ represents a search window centered at pixel $\mathbf{p}_j$ in image X_j with the size of w, and in our experiments, we typically set $w = 5 \times 5$. This procedure is illustrated in Figure 3.4.

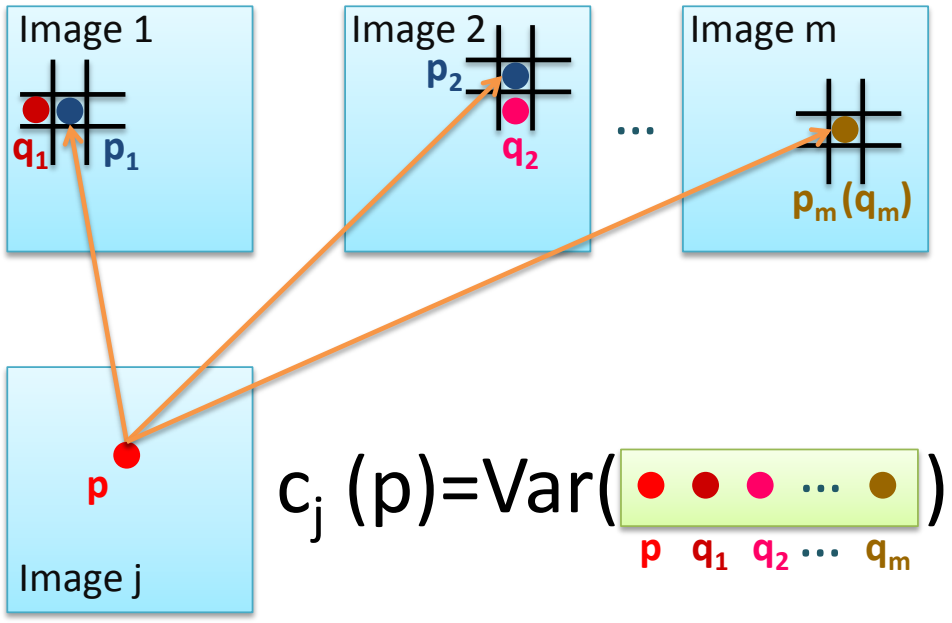


Figure 3.4: An illustration to compute the color variance $c_i(\mathbf{p}_i)$. Suppose pixel $\mathbf{p}_i$ finds its correspondences $\mathbf{p}_1$ in image X_1, and up to $\mathbf{p}_m$ in image X_m, and $\mathbf{p}_1$'s neighboring pixel $\mathbf{q}_1$ has the smallest color difference with $\mathbf{p}_i$ and so on and so forth; then the cost $c_i(\mathbf{p}_i)$ is the variance of the color values of $\{\mathbf{p}_i, \mathbf{q}_1, \mathbf{q}_2 \cdots \mathbf{q}_m\}$.

3.4.3 Incorporating Visual Cues

Given a variance map $\mathcal{C}_i$ of image X_i, we can formulate the variance map $\mathcal{C}_i$ as a cost function V into the data term defined in Eq. 3.6:

$$V(z_i(\mathbf{p})) = \begin{cases} \exp(-c_i(\mathbf{p})/\alpha_b) & \text{if } z_i(\mathbf{p}) = 0, \\ 1 - \exp(-c_i(\mathbf{p})/\alpha_f) & \text{if } z_i(\mathbf{p}) = 1. \end{cases} \tag{3.6}$$

The explanation of Eq. 3.6 is that if the color variance $c_i(\mathbf{p})$ is small, which implies that the pixel $\mathbf{p}$ in image I_i finds visually consistent matches in the rest of images, it should be considered as foreground ($z_i(\mathbf{p}) = 1$). On the contrary, a large color variance

$c_i(\mathbf{p})$ indicates that pixel $\mathbf{p}$ fails to find visually consistent matches. Therefore, it should be labeled as background ($z_i(\mathbf{p}) = 0$).

In addition, we can also take advantage of color information to increase the robustness of segmentation. In the variance map, the color pixels with very low variance can be thought as the seed pixels for foreground while the color pixels with a very high variance are regarded as background seeds. Here, the foreground color seeds F are the color values of the pixels whose variances are less than 0.1 percentile in $\mathcal{C}_i$. Similarly, the background color seeds B are the color values of the pixels whose variances are greater than 99th percentile in $\mathcal{C}_i$. We further cluster the foreground color seeds F into 5 clusters using k-means, and denote the foreground clusters as $\mathcal{F}$. Similarly, we cluster the background color seeds B into 50 clusters and denote the background clusters as $\mathcal{B}$. The function $\min(X_i(\mathbf{p}), \mathcal{F})$ is the normalized distance from the color of pixel $\mathbf{p}$ to the nearest cluster center in $\mathcal{F}$ and so is $\min(X_i(\mathbf{p}), \mathcal{B})$. The color similarity measure can be defined as

$$\Theta(z_i(\mathbf{p})) = \begin{cases} \min(X_i(\mathbf{p}), \mathcal{B}) & \text{if } z_i(\mathbf{p}) = 0, \\ \min(X_i(\mathbf{p}), \mathcal{F}) & \text{if } z_i(\mathbf{p}) = 1. \end{cases} \tag{3.7}$$

The intuition of Eq. 3.7 is that if the color of pixel $\mathbf{p}$ is similar to the foreground pixels, $\mathbf{p}$ is likely to be labeled as foreground as well. This assumption is widely used in the interactive segmentation methods.

In our experiments, suppose v_f is set to the 0.1 percentile in the variance map $\mathcal{C}_i$ and v_b is set to the 99th percentile in $\mathcal{C}_i$, we set $\alpha_f = 7 \times v_f$ and $\alpha_b = 0.5 \times v_b$ in Eq. 3.6, which would make the data cost $V(z_i(\mathbf{p}) = 1) < 0.15$ with the variance $c_i(\mathbf{p}) = v_f$ and $V(z_i(\mathbf{p}) = 0) < 0.15$ with the variance $c_i(\mathbf{p}) = v_b$.

3.4.4 Energy Minimization

Now, we present the complete binary labeling MRF model, and use the graph-cut algorithm [11] to minimize the following energy functional with smoothness constraints to segment the target object:

$$Z_i^* = \arg\min_{Z_i}(E_d + \gamma E_s), \tag{3.8}$$

where E_d is the data term, E_s is the smoothness term and γ balances the relative importance. E_d is the linear combination of the three cost functions defined as

$$E_d = \sum_{\mathbf{p}\in X_i}(1-\lambda)V(z_i(\mathbf{p})) + \lambda\Theta(z_i(\mathbf{p})), \tag{3.9}$$

where V, and Θ are defined in Eq. 3.6 and Eq. 3.7, respectively. The smoothness term is defined as

$$E_s = \sum_{\substack{\mathbf{p},\mathbf{q}\in X_i \\ \|\mathbf{p}-\mathbf{q}\|_\infty=1}} a(\mathbf{p},\mathbf{q})\delta(z_i(\mathbf{p}) \neq z_i(\mathbf{q})), \tag{3.10}$$

which enforces the neighboring pixels should be assigned to the same label except at the object boundaries, where normally $a(\mathbf{p},\mathbf{q})$ is large when pixels $\mathbf{p}$ and $\mathbf{q}$ are visually similar and $a(\mathbf{p},\mathbf{q})$ is close to zero when pixels $\mathbf{p}$ and $\mathbf{q}$ look very different:

$$a(\mathbf{p},\mathbf{q}) = \exp\big(-\beta\|X_i(\mathbf{p}) - X_i(\mathbf{q})\|_\infty\big); \tag{3.11}$$

where in all of our experiments we set $\beta = 1/20$, $\gamma = 0.2$ in Eq. 3.8 and $\lambda = 0.3$ in Eq. 3.9.

3.5 Experiments

We first evaluate the efficiency and effectiveness of the proposed pairwise association mining algorithm and the pattern composition algorithm through synthetic experi-

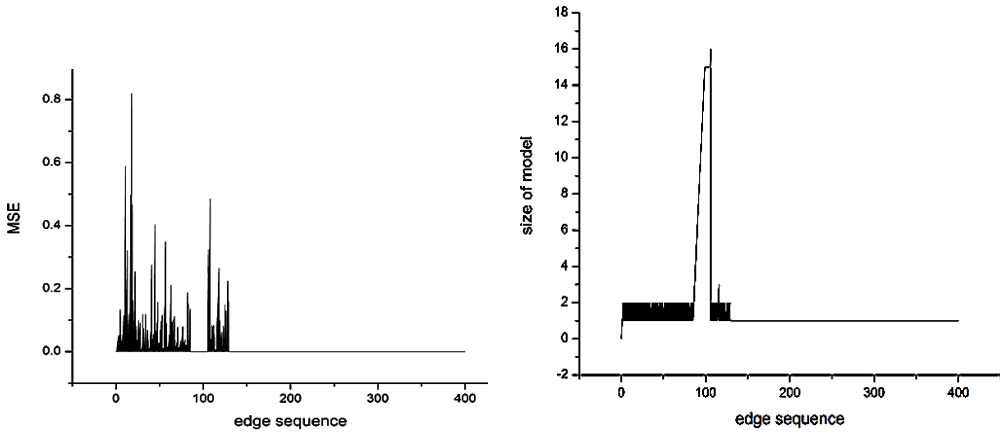
ments. Next, we demonstrate the utility of the proposed algorithms for unsupervised learning of visual semantics from both single images and two image datasets with repeated patterns. Given the extensive amount of labeling required in figures here, we rely on color to differentiate different groups and depict composite patterns. Please refer to the electronic PDF copy for better viewing figures in color with magnification.

3.5.1 Evaluation on Synthetic Datasets

We evaluate the efficiency and effectiveness of the proposed pairwise association mining algorithm and the pattern composition algorithm through synthetic experiments.

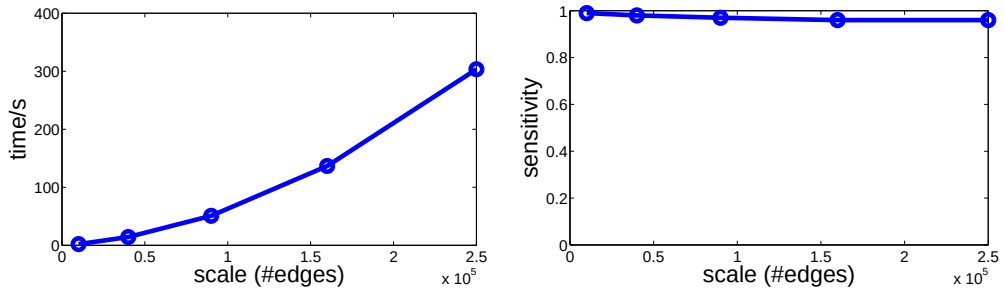
Pairwise Association

In this experiment, we generated two items for every dataset and applied the pairwise association mining algorithm. Each item consisted of a set of primitives, and one or more associations were embedded within a background of random relations between primitives from different items. The first dataset is to show how the algorithm works when discovering a pairwise association. 20 primitives were included in each item, therefore 400 edges (relations) were generated in total. An association that consisted of 15 relations was embedded. A sliding window dynamically grows or shrinks on the ordered edge set and a maximum matching is found in each step. Figure 3.5(a) shows the variations of MSE of the weights of edges in the maximum matchings found at every step, where the horizontal axis represents the position of the window. Figure 3.5(b) plots the number of edges in each maximum matching (size of each matching) found at every step. One candidate (the largest) association is located



(a) (b)

Figure 3.5: An illustration of the pairwise algorithm for finding a single association. (a) shows the change of MSE of the maximum matchings found at every step when the sliding window moves through a sorted edge list. (b) shows the corresponding sizes of the maximum matchings.



(a) (b)

Figure 3.6: Experimental results on synthetic datasets with different scales.

around the 100th edge, where the MSE is small and the size of the maximum matching reaches a local peak. This association consists of 15 edges, which is the same as the ground truth. Generally, a matching would be indicated as an association when the size reaches a local peak and when the MSE remains small. Note that, in Figure 3.5, the MSE of some other regions is also low, but their sizes are too small to be indicated as an association.

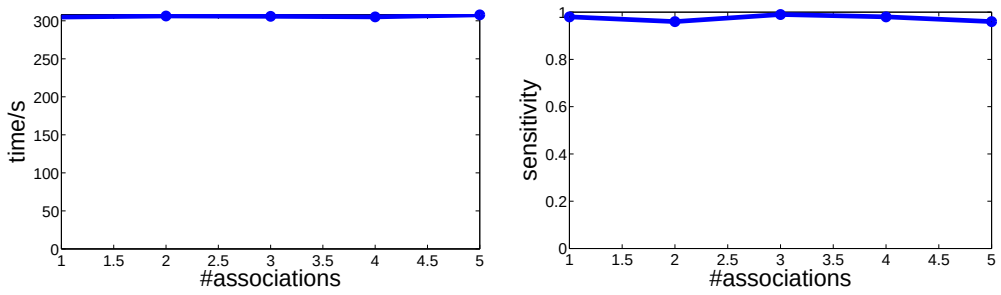
Figure 3.6 shows the scalability of the algorithm. The experiment was carried out on 5 datasets with different sizes of items. The number of primitives in

each item varied from 100 to 500, and the number of edges ranged from $10,000$ to $250,000$ accordingly. For every dataset, two associations were embedded between the pair of items, and the size of each association was one third of the size of the item. Figure 3.6(a) shows the running time of the algorithm, which is consistent with the analytical conclusion that the time complexity for mining pairwise associations is polynomial. Figure 3.6(b) presents the sensitivity of the obtained associations when the size of items is gradually increased. The sensitivity is denoted as $\frac{\#true\ positives}{\#true\ instances\ of\ embedded\ association}$. More than 95% of the embedded associations can be resolved in all datasets.

Next, we evaluated the performance of our algorithm when the number of associations varied. We incrementally embedded up to 5 associations into the dataset in which there were 500 primitives in each item. The size of every embedded association was 100. As shown in Figure 3.7(a), the running time of the algorithm is roughly the same despite the change of the number of associations, since our algorithm is almost invariant to the number of associations. For every association embedded, more than 95% of relations are always resolved, according to Figure 3.7(b), demonstrating its ability to identify multiple associations with two items.

Pattern Composition

To examine the effectiveness of the pattern composition algorithm, a synthetic $2D$ scene was constructed, where the locations and orientations of primitives were randomly generated and several patterns were also embedded with random orientations. In Figure 3.8, each column consists of a group of experimental results on a synthetic



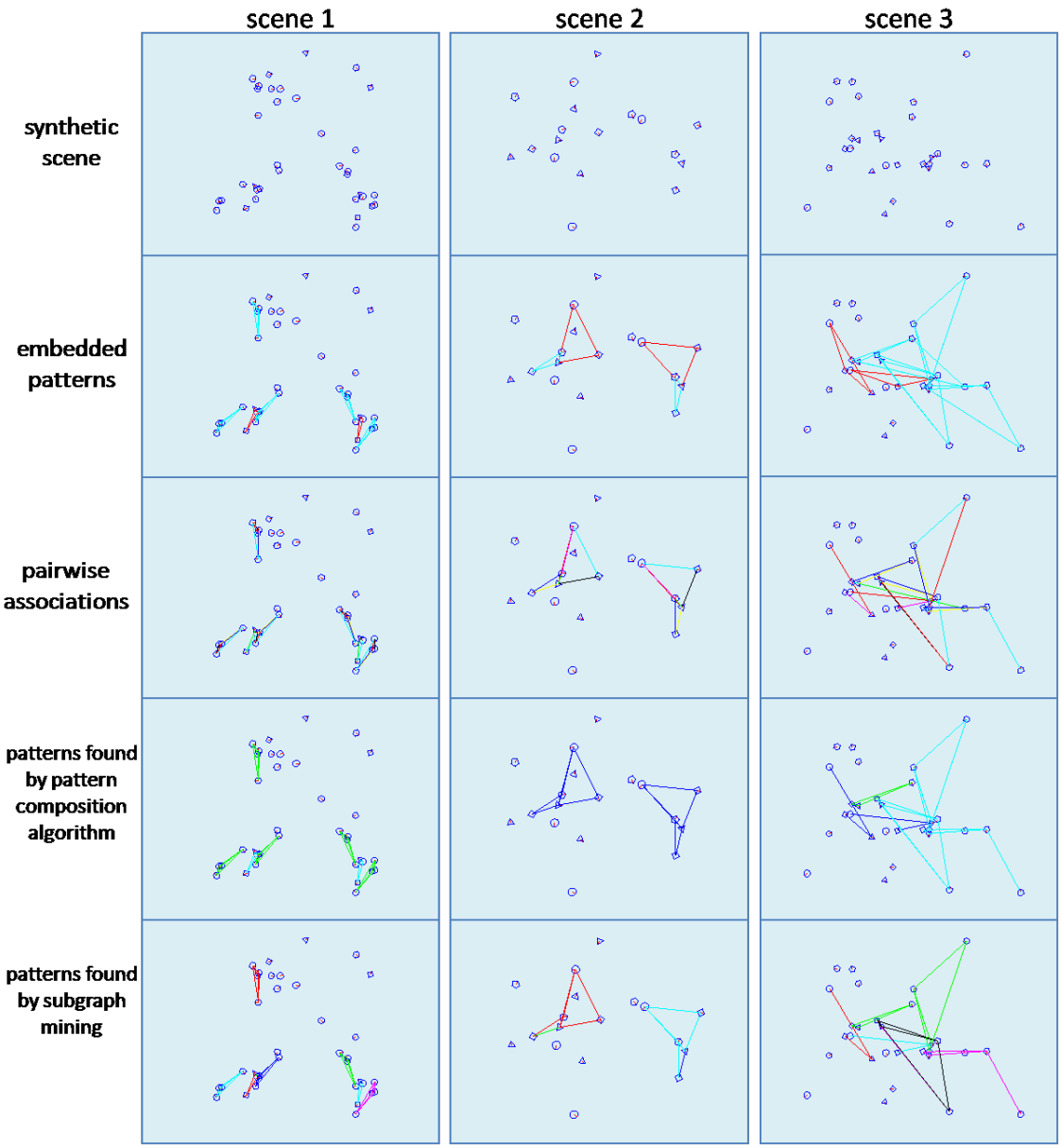
(a) (b)

Figure 3.7: Experimental results on synthetic datasets with a different number of associations.

$2D$ scene. The figures in the first row are the original scenes. The figures in the second row show the embedded ground-truth patterns, where identical pattern instances are colored the same. There were two different patterns in the first scene (red and green), each of which consisted of 2 items. In the second scene, every pattern instance was composed of 3 primitives. The third scene was more complicated with 4 primitives in the green pattern. The third row shows the results after applying the pairwise association mining algorithm on each scene, and the relations belonging to the same pairwise association are colored the same. Primitives with the same shapes are considered to be from the same item. There were respectively 5, 6 and 6 different associations discovered in each scene. Afterwards, both the pattern composition algorithm and the subgraph mining algorithm were applied to extract the set of maximal patterns.

The results of this extraction are respectively shown on the fourth and fifth line. The pattern composition algorithm is able to identify almost exactly the same set of embedded patterns, although there might be some missing edges.

The method of subgraph mining can detect frequently occurring subgraphs as well,



Algorithm	scene 1	scene 2	scene 3
Pattern Composition	0.049788sec	0.042806sec	0.048114sec
Subgraph Mining	0.489734sec	0.527052sec	0.384014sec

Figure 3.8: Experimental results on three sets of synthetic 2D scenes.

which provide possible compositions of patterns. While subgraph mining algorithm is also able to discover all embedded patterns, it performed one order of magnitude slower than our pattern composition algorithm — the run-time comparison is below

the figures representing each scene. We expect the pattern composition algorithm's speed to increase with an experiment of a larger scale. Furthermore, the subgraph mining algorithm may generate far more patterns than what have been embedded, making it difficult to interpret.

With the minimum frequency set as 2, there were 10, 89 and 31 patterns found in each scene. In contrast, the number of patterns resulting from the pattern composition algorithm were respectively, 9, 7 and 9. The multiple colors of the edges in the same pattern instances are due to redundant and overlapping subgraphs discovered.

3.5.2 Evaluation on Real Images

Repeated visual patterns can be found in one single image where multiple copies of patterns are embedded. Besides the simple pencil example as shown in Figure 3.2, our first experiment tested the algorithm on the section of shelves in the grocery store. Due to the space limitation, we only show a small segment of long panorama image [2] in Figure 3.9(a). Repeated patterns representing different snack boxes are present in the image but their positions are random. Our goal is to discover the set of features that can represent or summarize each pattern in the image. There are totally $14,288$ SIFT features extracted in the original image shown Figure 3.9(a). These features are further clustered into $1,000$ clusters using k-means. Associations are then searched between every pair of feature clusters and are used to label the edges between features in the image. The heuristic pattern composition is then applied to discover complex patterns with consistent structural relationships according to the method detailed in Section 3.3 as shown in Figure 3.9 (b)(c). Please note that most of the discovered

patterns in each dataset are strong patterns, or near-strong patterns (almost cliques) meaning that there exists all (major) pair-wise associations among clusters within the patterns.

One thing worth mentioning is that our algorithm is able to automatically discover the global occurrence of patterns independent of its spatial locations and neighborhood. For example, the snack box "CHEEZE IT" appears at several locations in the image, but they are able to be identified due to the fact that they have consistent structural patterns. Approaches based on the spatial random partition technique [123] may find some of the patterns, but there is no guarantee to detect and locate the repeated subimages in each retrieval. However, using our approach, we can successfully and efficiently find most of the common image parts.

Comparison between Pattern Composition Algorithms We test the both pattern composition algorithms, the frequent subgraph mining and the polynomial-time heuristic methods proposed in Section 3.3 for the grocery image (Figure 3.9) on a single core working at 2.53GHz; the running time was 33 minutes for the frequent subgraph mining algorithm, but only 7 seconds for the heuristic approach. Furthermore, frequent subgraph mining outputs $30,653$ visual patterns, most of them are overlapped and this large number of patterns are almost impossible to visualize and interpret; however, the heuristic approach only yields 5 patterns as shown in Figure 3.9 (c).

Feature detection on image may miss pattern components and lead to incompleteness of associations among visual items. For example, a complete pattern of "Wheat Thins" (the green-colored pattern) comprises a clique of 4 features, such as the left-

(a)

(b)

support=3 support=5 support=7 support=15 support=4

(c)

Figure 3.9: Visual patterns discovered in the grocery image. (a) is the input image. (b) shows the frequent visual patterns marked in different colors according to the actual items that they belong for better viewing. (c) shows some patterns with magnification and the corresponding support values.

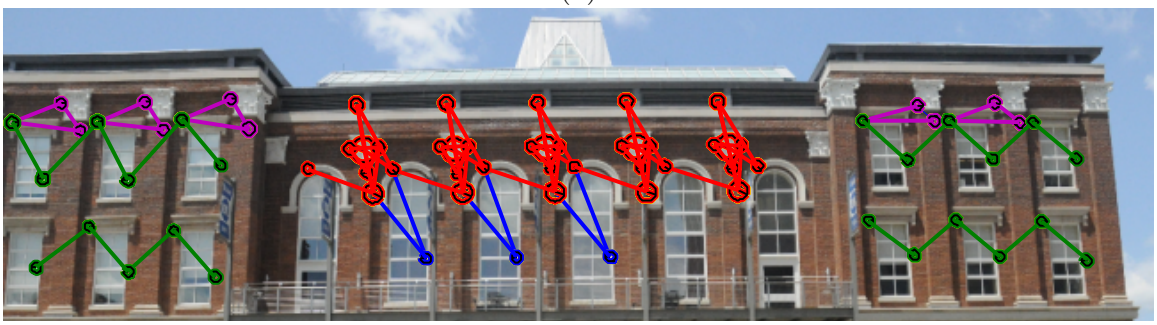
most instance on the top shelf. The rest instances on the top shelf all miss the top feature and its associations. To capture the homogeneity of the pattern structure, our proposed algorithm searches among all instances for the common path (instead of a maximal subgraph) that has the highest support (Figure 3.9 (c)). Features and associations specific to each instance are then used to extend the common path into a subgraph that represents the pattern instance.

Our next experiment concerns repeated patterns in buildings. Recent work on identifying patterns in images typically find only simple and highly repetitive patterns [111]. However, as shown in Figure 3.10 (a), structures such as windows, doors, and sculptures can be versatile and vary greatly in lengths and shapes. The approach [111] will fail in this images. Our approach however is able to detect almost all repeated structural patterns in the building facade as shown in Figure 3.10 (b)(c). Independent of spatial proximity, our approach is able to discover long associations as shown in the repeated long windows in the middle part of the image (the red and blue patterns (the 3rd one in Figure 3.10 (c))).

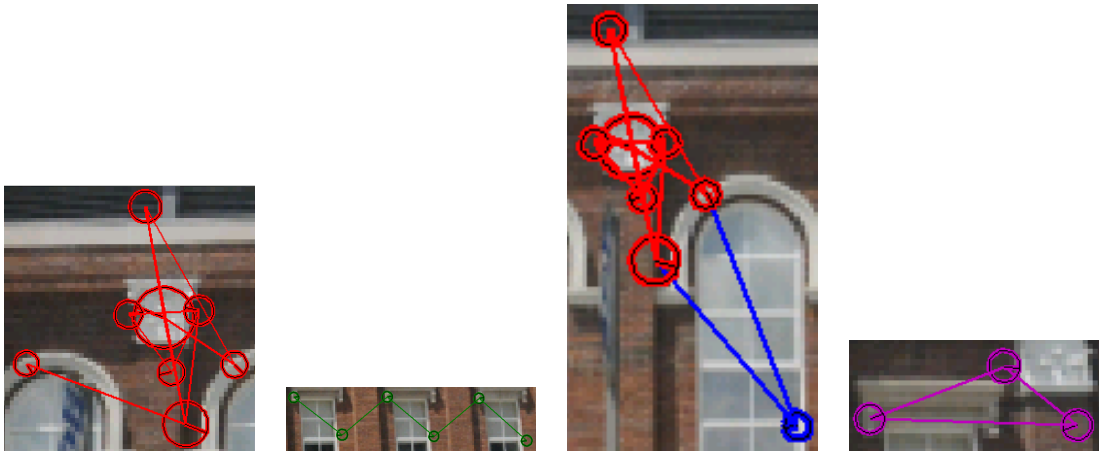
For both of these images, our method is capable of extracting almost all of the embedded structural patterns without any prior knowledge of the locations, the sizes or the number of patterns.

Besides one single image, our method can be easily adapted to mine patterns across multiple images. We show the visual pattern mining and segmentation results on an image dataset consisting of 11 images with 26 Starbucks logos embedded under various pose and illumination changes in Figure 3.11 and Figure 3.12. Our frequent visual pattern mining algorithm can successfully detect 22 visual pattern instances; all

(a)



(b)



support=5 support=3 support=4 support=5

(c)

Figure 3.10: Visual patterns discovered in an image of a building facade. (a) is the input image. (b) shows the frequent visual patterns in different colors. (c) highlights the patterns with magnification and reports the corresponding support values.

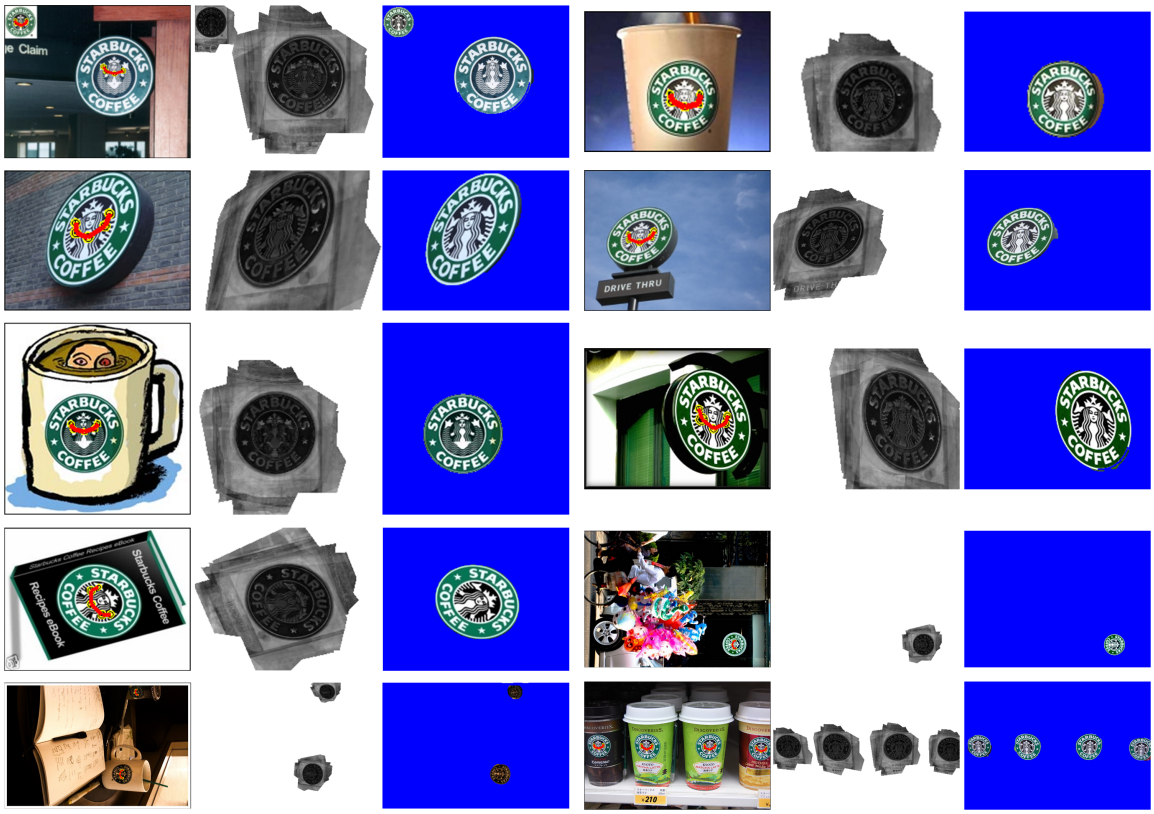


Figure 3.11: Discovered Visual patterns and segmentation of the Starbucks Logo dataset. The 1st and 4th columns consist of original images overlaid by the detected visual patterns marked in red, the 2nd and 5th columns visualize the corresponding variance maps and the 3rd and 6th columns show the corresponding segmentation.

of them correspond to the relationship among some visual primitives detected on the siren embedded in the Starbucks Logos. Our algorithm is able to automatically detect the meaningful repeated structural patterns independent of their spatial locations, scales, orientation variations and lighting changes.

As the Starbucks logo is planar for most cases, we can estimate homography transformation between visual primitives around the detected pattern instances and the resultant variance value is consistently low within the foreground, therefore we can get satisfactory segmentation result. Note that, if an image X_i contains multiple

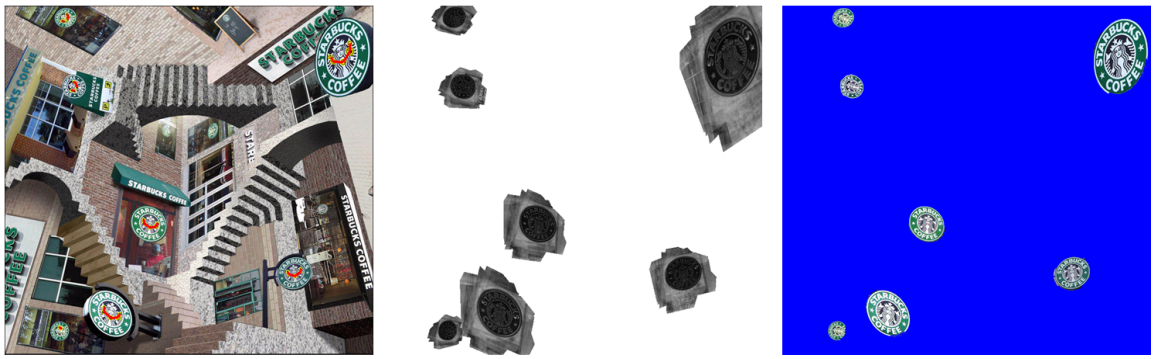


Figure 3.12: Discovered Visual patterns and segmentation of the Starbucks Logo dataset (continue). The left image is the original image overlaid by the same detected pattern as Figure 3.11, the middle image visualizes its variance map and the right one is the segmentation result.

copies of the same pattern, we simply create multiple copies of X_i, each of which corresponds to one and only one pattern instance. We can achieve overall segmentation accuracy of $97.86 \pm 1.72\%$ on the whole dataset. Our algorithm is able to automatically detect and segment the meaningful repeated structural patterns independent of their spatial locations, scales, orientation variations and lighting changes. Moreover, identifying the frequent visual patterns makes a good initial guess on the location of the target object of interests, which improves the homography estimation accuracy and facilitate high quality variance map generation. Last but not least, with the help of multiple pattern instances in images, the illumination bias in a small portion of images is also reduced and we can still segment the image under different illumination conditions without re-estimating its global appearance model.

Last, we demonstrate the results by applying the algorithm on the face dataset from the Caltech-101 database [54] (435 images of 23 persons). Figure 3.13 shows

a few examples of the discovered visual patterns associated with the precision[1] and recall[2] scores. The face dataset is more challenging due to the weak textures and individual difference of human faces. Consequentially, the repeatability of SIFT features is quite limited and the ambiguity of SIFT descriptor is higher than the above shown experiments. Each pattern has high precision means that it is important to capture semantics in face. Still, the recall for each of the pattern is not very high especially for more complicated structural patterns such as the pattern shown in the fourth row. This is because the chance of missing features is too high to support the complex patterns.

Among all discovered patterns, each contains a long-range structural pattern and indeed carries meaningful semantics. For example, the first row contains a pattern composed by forehead and mouth. The second row represents a pattern which consists of forehead and two eyes. The third row depicts a pattern linking forehead, two eyes and mouth. The fourth row shows the pattern with forehead, nose, a left eye and mouth; and notice that there exist a clique linking a left eye, nose and mouth together. Therefore, our method is able to detect more robust long-range patterns comparing to the results demonstrated in [124], where only local patterns can be detected such as features localized at eye corners. We also demonstrate both of the qualitative and quantitative results on segmentation of the face database in Figure 3.14.

[1] precision = ♯positive detects/(♯positive detects + ♯false detects)
[2] recall = ♯positive detects/(♯positive detects + ♯miss detects)

Figure 3.13: Visual patterns discovered in the face dataset. Each row shows the same pattern across different images and the corresponding precision and recall scores are reported in the last column.

Comparison with Co-segmentation

By courtesy of the authors of [40], we used their implementation and the parameters recommended in [40] to generate the co-segmentation results as shown in Figure 3.15. When the foreground color bears great similarity with the background color, e.g., the first and second rows in Figure 3.15, it cannot distinguish the foreground and background well. In addition, when the illumination conditions between the image pair are quite different, e.g., the last row in Figure 3.15, co-segmentation may fail to get meaningful foreground.

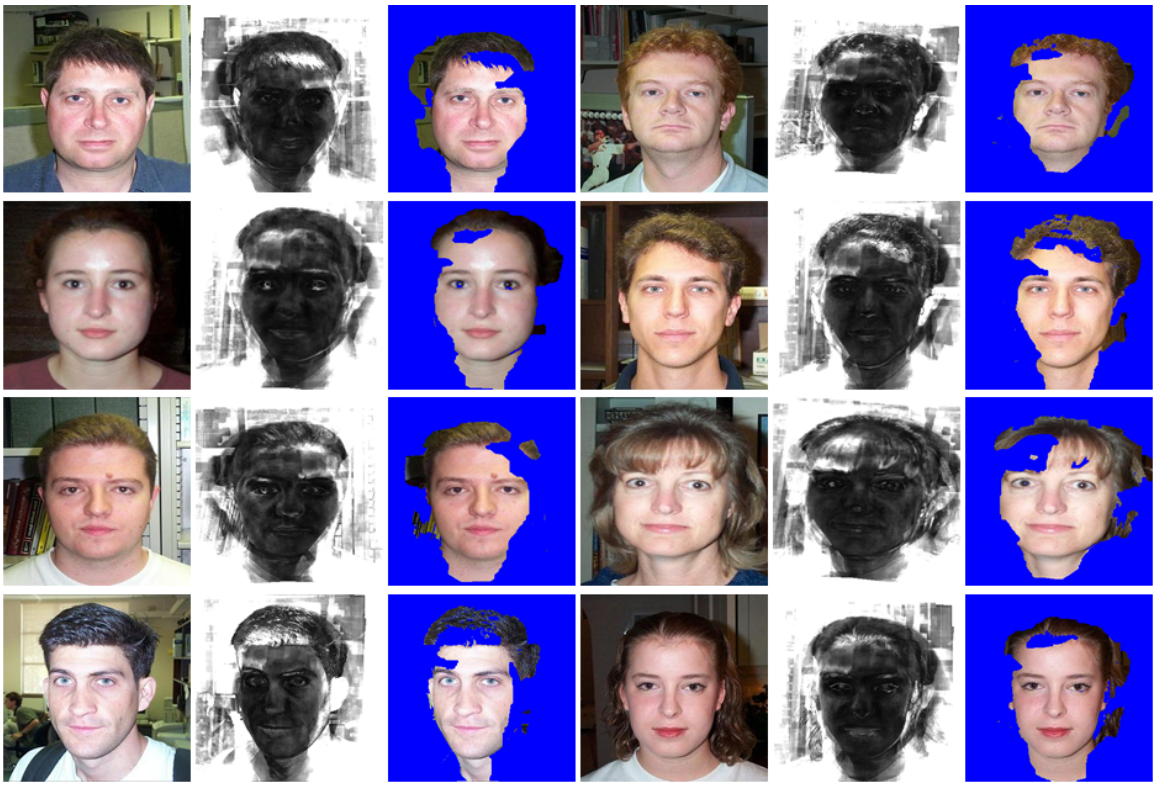


Figure 3.14: Segmentation on the face dataset. The 1st and 4th columns consist of original images, the 2nd and 5th columns visualize the corresponding variance maps and the 3rd and 6th columns show the corresponding segmentation. The overall segmentation accuracy is $81.05 \pm 8.14\%$.

A Failed Segmentation Case

The fundamental assumption we make is that the same visual pattern is embedded on different backdrops. In Figure 3.16, we present a case in which there is no background at all. Even though the visual pattern ("RITZ" snack box in Figure 3.9 (b)) is correctly detected, suppose we expect to only cut out the 3rd "RITZ" snack box in the bottom row of Figure 3.9 (a), due to those "RITZ" snack boxes are put together, the same offset (homography transformation) would also transfer nearby "RITZ" snack boxes accordingly, so that the resultant variance values on other "RITZ" snack boxes

Figure 3.15: Co-segmentation [40]: each row corresponds to an image pair and their segmentation.

are also quite low, and our algorithm is unable to produce a satisfactory segmentation.

3.6 Summary

In this chapter, we present a novel framework to find frequent high-order structural patterns from unstructured images. Compared to previous visual pattern mining approaches that translate an image into a transactional database based on some heuristic rules (such as spatial proximity), we regard spatial coherence as the major criterion to link visual items or primitives to form meaningful patterns. Toward that design goal, we develop a set of novel and efficient algorithms to find optimal

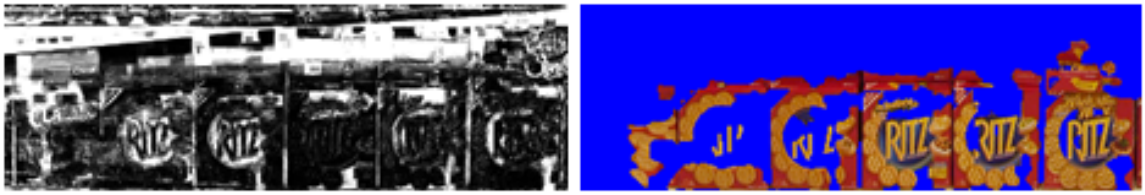


Figure 3.16: A Failed case: the left image visualizes a portion of the variance map targeted at the 3rd "RITZ" snack box in Figure 3.9 and the right one is the segmentation where nearby "RITZ" snack boxes are also labeled as foreground.

pair-wise matches among all detected visual items. This allows us to find visual patterns of all sizes, from small ones close by to large ones that are relatively far apart. As demonstrated by the results, our approach is robust in finding meaningful visual patterns from a variety of images. We also present a novel framework to segment frequent structural patterns from a large collection of images. Compared to the traditional segmentation approaches that involve user interactions, we utilize the redundant information brought by the images themselves to segment the re-occurring visual patterns in a fully automatic and supervised manner. To approach this goal, we first established dense correspondences among pattern instances, computed a per-pixel cost, and finally extracted the target object. As demonstrated by the results, our approach is robust in segmenting meaningful frequent structural visual patterns from a variety of images.

Chapter 4 Image Segmentation based on Internet Photo Collections

With the seminal work by Snavely et al [99] that used Internet photo collections (IPCs) for 3D reconstruction and visualization, many image editing operations have been developed to unlock the rich information contained in IPCs. Among these image editing techniques, one important prerequisite is the segmentation of the foreground. This chapter presents an automatic image segmentation technique for a personal photo taken at a landmark location with the help of an IPC surveying roughly the same place. As demonstrated in Figure 4.1, by using the rich IPC taken at Notre-Dame, the foreground of a casual cellphone image can be automatically segmented out, which can further help to turn the photo into a stereoscopic image with minimal user interaction [125] or even remove the entire foreground and replace with the background Notre-Dame (Internet-based Inpainting [115]).

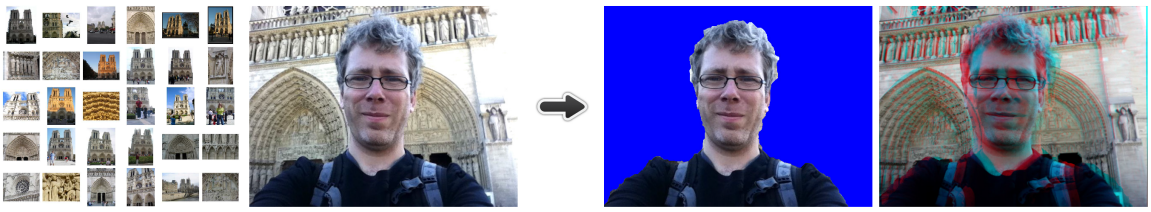


Figure 4.1: Example of Automatic Image Segmentation based on IPC. From left to right: samples of IPC of Notre-Dame, the original image, foreground segmentation and stereoscopic image conversion.

4.1 3D Reconstruction from Internet Photo Collections and Personal Photo Registration

For IPCs that span a unique landmark, we reconstruct the camera locations using Bundler [99], which is an incremental structure-from-motion pipeline. We increase the density of the obtained point cloud by using PMVS [30]. Given a new image $\mathcal{Q}$ of one's personal photo, in order to perform image enhancement we first need to register $\mathcal{Q}$ to the reconstructed 3D landmark model $\mathcal{M}$. This can be considered as performing an incremental update of the reconstructed model.

4.2 Segmentation of Foreground Objects based on Internet Photo Collections

As opposed to interactive segmentation methods that rely on user interaction to learn the foreground and background appearance models, our method can acquire training data automatically based on the Internet photo collection. Similar to Section 3.4, the fundamental assumption we make here is that a pixel belonging to the background landmark is likely to be photometrically consistent across other views, whereas a foreground pixel usually is not.

Our method first projects the 3D model $\mathcal{M}$ onto the image $\mathcal{Q}$ denoted by m. Due to the power of large scale IPCs, we found that often times we have numerous images from nearby camera poses. When selecting from the set $\mathcal{S}$ of these images, we favor images that have not only similar camera location and orientation, but also similar image condition (e.g., color distribution) to image $\mathcal{Q}$. Suppose a visible 2D

point $p \in m$ is projected from the 3D point $P \in \mathcal{M}$ and we denote its neighboring 3D point set as $N(P) = \{P' : \|P - P'\|_2 \leq 3 \cdot l\}$, where l is the average spacing between two closest 3D points in $\mathcal{M}$. We then compute $\text{NCC}(p, I_i)$, the normalized cross correlation of the color values of the projection of $N(P)$ on the image $\mathcal{Q}$ and the projection on an image $I_i \in \mathcal{S}$. We consider p is *consistent* between the image $\mathcal{Q}$ and the image I_i if $\text{NCC}(p, I_i) \geq 0.6$ or p is *inconsistent* if $\text{NCC}(p, I_i) \leq 0.2$. If $\text{NCC}(p, I_i)$ is low because the projection of $N(P)$ lies on occlusion boundary, we still treat p as inconsistent between image $\mathcal{Q}$ and I_i. If p is consistent with the majority, i.e., over 80% of total number of images in $\mathcal{S}$, p is classified into the background seed set $\mathcal{B}$; similarly, if p is inconsistent with majority, p is classified into the foreground seed set $\mathcal{F}$.

Figure 4.2: Segmentation results: (top left) the original image superimposed with the bounding box prior for Grabcut, (top right) red dots for background color seeds and blue dots for foreground color seeds, (bottom left) the result of Grabcut, (bottom right) the result of our automatic approach.

We revise the initial setup of Grabcut [86] framework (similar to the method used in Section 3.4) in two aspects: (1) we use automatically generated training data $\mathcal{F}$ and $\mathcal{B}$ to initially build the Gaussian Mixture Models for foreground and background instead of user-provided bounding box; (2) we add a constant penalty to the unary term of each pixel $p \in \mathcal{F}$ (or $p \in \mathcal{B}$) if p is labeled as background (or foreground) at the first run. We then perform the iterative energy minimization from Grabcut [86] to compute the segmentation. Figure 4.2 and Figure 4.3 compare fully automatic segmentation results from our method with Grabcut. Due to our precise color seeds used to train the appearance models for both foreground and background, our automatic approach achieves more accurate and meaningful segmentation than Grabcut. More results are shown in the experiments.

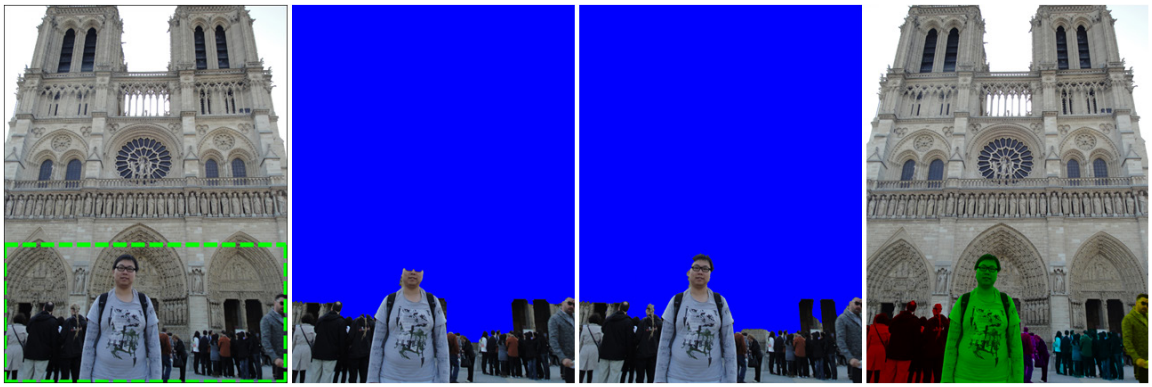


Figure 4.3: Segmentation results for an image taken in a cluttered scene: from left to right, the original image superimposed with the bounding box prior for Grabcut, the result of Grabcut, the result of our automatic approach and interactively separated different (color coded) foreground objects.

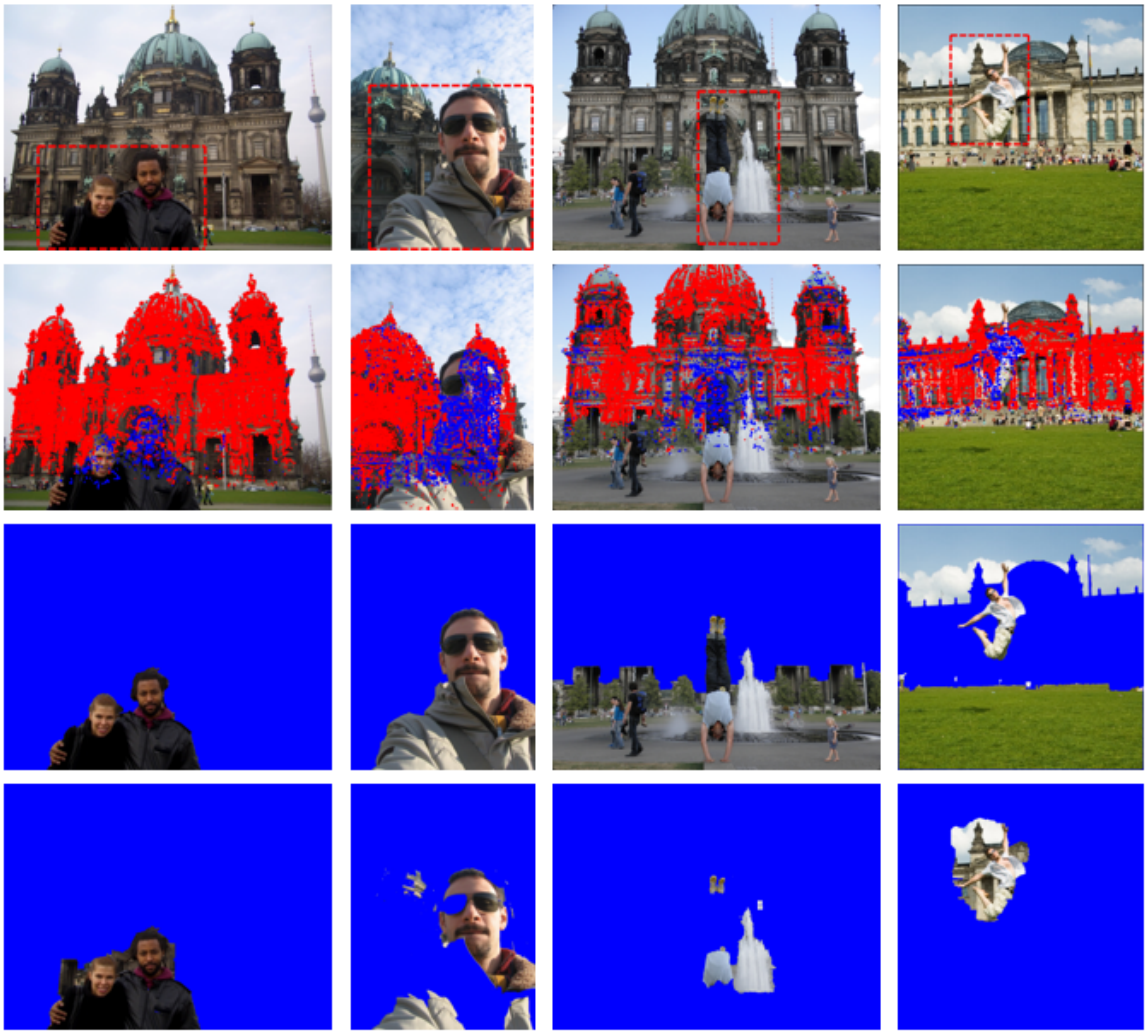


Figure 4.4: Segmentation results of Berlin Dom dataset. Each column shows the results for one personal photo. From top to bottom, the original image superimposed with the bounding box prior for Grabcut, foreground and background color seeds (red for background and blue for foreground), the result of our automatic approach and the result of Grabcut.

4.3 Experiments

We first demonstrate our automatic segmentation results from four scenes and compare them against Grabcut [86] with accurate user-provided bounding box priors as shown in Figure 4.4, Figure 4.5 and Figure 4.6. Thanks to the precise color distribu-

Figure 4.5: Segmentation results of Trevi Fountain and the Great Sphinx of Giza datasets. Each column shows the results for one personal photo. From top to bottom, the original image superimposed with the bounding box prior for Grabcut, foreground and background color seeds (red for background and blue for foreground), the result of our automatic approach and the result of Grabcut.

tion of the seeds automatically generated by visual consistency check based on IPC, our automatic segmentation approach outperforms the state-of-the-art Grabcut [86] approach.

Next, we conducted a user study to evaluate how realistic our automatic segmentation results are compared to the processing results from the state-of-the-art image editing tools, e.g., Photoshop. Four Photoshop experts were asked to process four images for segmentation as shown in Figure 4.7. Based on the design goal of our system, we hypothesize that using our system will be able to complete the segmentation significantly quicker than using Photoshop, the state-of-the-art image editing

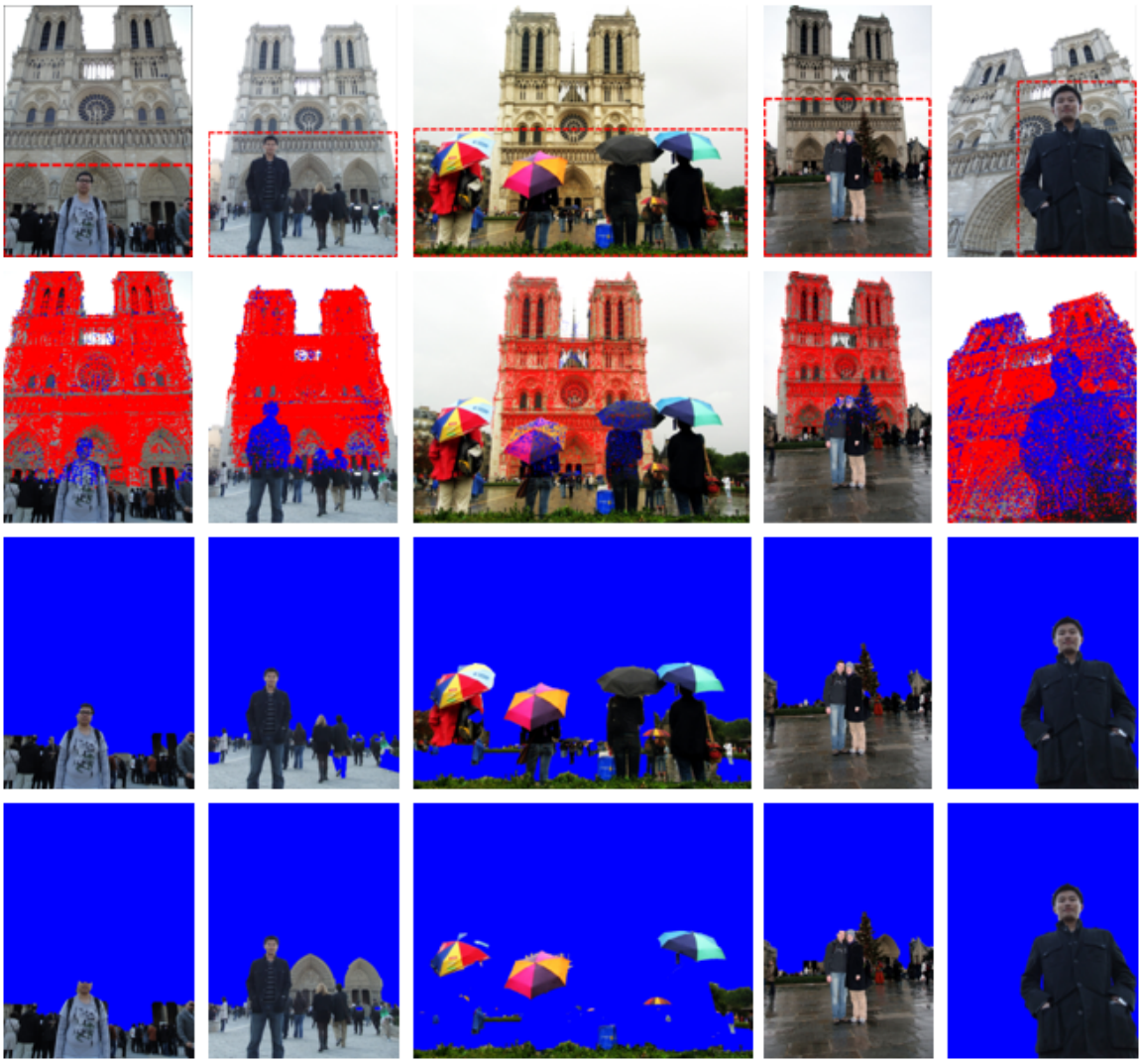


Figure 4.6: Segmentation results of Notre-Dame dataset. Each column shows the results for one personal photo. From top to bottom, the original image superimposed with the bounding box prior for Grabcut, foreground and background color seeds (red for background and blue for foreground), the result of our automatic approach and the result of Grabcut.

tool. The average time required to complete the foreground segmentation task is 34 seconds compared to 140 seconds using Photoshop. ANOVA tests confirm that the time differences are statistical significant ($F = 10.87$, $p - value = 0.008 < 0.01$), which validates our hypothesis.

Figure 4.7: Comparison with Photoshop. From left to right: original images, our results and Photoshop results.

Twenty-five users are asked to compare the Photoshop segmentation results with our results on the same images. To avoid bias, images from two methods are randomly ordered and users do not know which images come from which method. Users were required to select one of five preference choices: (1) Image 1 is much better than Image 2; (2) Image 1 is slightly better than Image 2; (3) Image 1 is equal to Image

2; (4) Image 1 is slightly worse than Image 2; and (5) Image 1 is much worse than Image 2. The user study shows that 6% thought ours are much better than Photoshop's; 22% thought ours are slightly better than Photoshop's; 25% thought ours are equal to Photoshop's; 24% thought ours are slightly worse than Photoshop's and 23% thought ours are much worse than Photoshop's. For this quantitative analysis, we can see that our results achieves comparable quality as Photoshop. It is worth to point out that if taking Photoshop segmentation as the ground truth, our automatic foreground segmentation achieves an average error rate of 3% , with minimum error rate of 0.7% and maximum error rate of 8.7% typically due to multiple foreground layers clutters, for example, Figure 4.3 and the last row in Figure 4.7. This indicates that our approach is capable of automatically producing comparable foreground segmentation to handtuned foreground segmentation maps in Photoshop. The error rate is computed as the percentage of mislabeled pixels. Fully automatic image segmentation is a challenging task. As shown in Figure 4.3, our automatic approach sometimes is not sufficient to further separate foreground layers. Therefore, we still require user interaction to separate the foreground layers in a clutter scene.

4.4 Summary

In this chapter, we approach the automatic photo segmentation from a novel direction – using IPCs. Our work leverages the 3D background models reconstructed from IPCs of the same landmark. With the rich information from large scale IPCs, we believe that by augmenting one's personal photo with depth information, as well as the surrounding appearance information, a number of interesting photo enhancement

can be achieved. The Application that we have explored in this chapter is automatic image segmentation which can further facilitate stereoscopic view synthesis [125] and image impainting [115].

Chapter 5 Semantic Segmentation and Reconstruction of Residential Scenes from LiDAR Data

In this chapter, I present a comprehensive system to semantically segment point clouds and reconstruct detailed 3D models. Starting with LiDAR data with registered color images, we first segment the unorganized 3D points into distinctive categories including houses, plants, street lights, etc. Then for each category we develop unique solutions to reconstruct its 3D model, taking advantage of the prior information about this particular category. For example, common objects, such as street lights, are replaced by similar 3D models found on the Internet. Plants are modeled with billboard techniques, which are known to be visually convincing. Special emphasis is put on the reconstruction of houses. The typical properties of buildings, such as piece-wise planar structures, convexity, and symmetry, are used to develop an efficient reconstruction algorithm that can deal with incomplete data. The outcome of our system is a set of visually complete 3D models consisting of common static objects in an urban scene, including not only houses, but also plants, street lights, mailboxes, etc. Each object has its own semantic labeling. The overview of the pipeline is shown in Figure 5.1.

The primary target of our system is residential areas. Many of existing modeling approaches, in particular those generating models with high details and rich textures, deal almost exclusively with multiple-story or high-rise buildings (e.g. [76,79]). These buildings are typically found in downtown and highly populated urban areas.

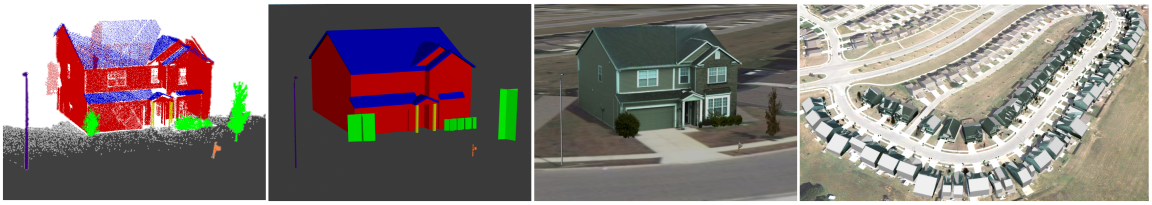


Figure 5.1: The pipeline of semantic segmentation and reconstruction of LiDAR point clouds. From left to right: (a) semantically labeled 3D point cloud; (b) reconstructed objects using category-specific methods, including billboard trees, replaced common objects, and a building. The color-code on the building shows recognized different building parts; (c) textured 3D models on a ground plane, and (d) an overview of an automatically reconstructed large-scale scene.

The structural details are repetitive and regular, from which user-defined grammar rules can be used to regularize the reconstruction to generate clean output. This kind of repetition is not available in residential areas. In addition, thanks to popular modeling tools such as Google Sketchup, many of these metropolitan areas already have 3D models. Admittedly, those low-rise houses in suburban/residiental areas are less glamorous to work with from a visualization standpoint, but they are equally important from a simulation or city planning standpoint since they are literally everywhere. For these areas, automation is important because of the large scale. Prior approaches to reconstruct these areas are developed with aerial data [21, 55], which produce no details below roof. The system presented in this chapter fills the void of automatic decomposition and reconstruction of residential areas from ground-based LiDAR data.

The data is acquired by a mobile LiDAR scanning platform as shown in Figure 5.2 which consists of two LiDAR sensor heads (Optech Lynx Mobile Mapper V200), multi-channel GPS, high-precision inertial measurement unit (IMU), distance

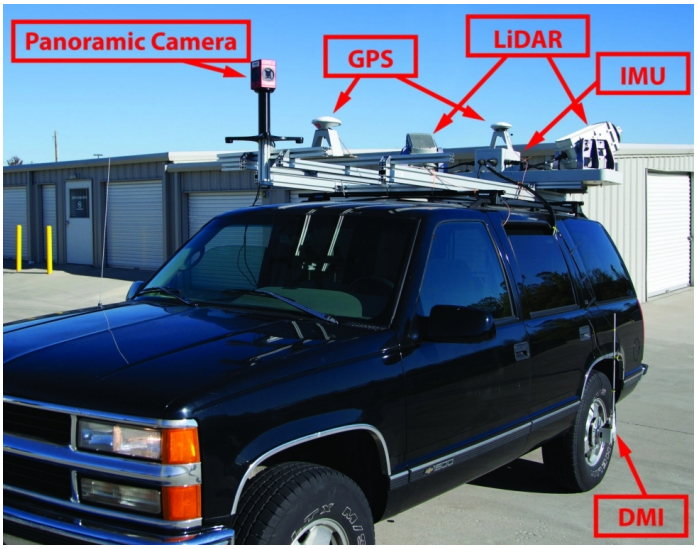


Figure 5.2: A high-quality mobile LiDAR scanning platform.

measurement instrument (DMI), and a high-resolution panoramic video camera (Ladybug 3). The scanning trajectory is also recorded with GPS and IMU. A 3D point is documented in the Universal Transverse Mercator (UTM) coordinate system with easting and northing and elevation reading. The images and point clouds from LiDAR sensors are roughly registered and geo-referenced.

5.1 Semantic Segmentation on Point Clouds

We define the following categories in our current implementation: *houses*, *plants*, *mailboxes*, *street lights*, *waste bins*, *cars* and *ground*, which are common objects seen in a residential area. From our scanned data, we manually label a section as the training data set and adopt the semantic segmentation approach similar to Zhang et al. [126]. It should be mentioned that other alternatives such as [35] and [121] can also be applied to perform this task.

In order to deal with our point cloud data, we have made a few changes to the

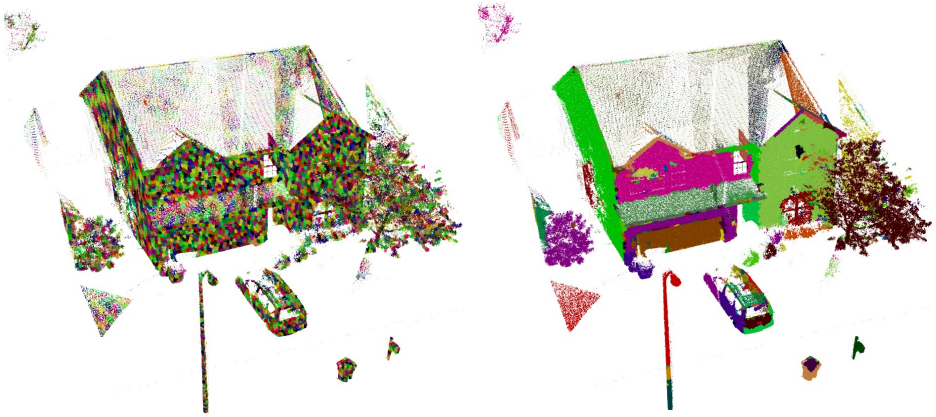


Figure 5.3: Grouping point clouds into superpoints and super-regions. (left) Superpoints of a point cloud; (right) super-regions of the point cloud.

original method. First, we group points into *superpoints*, analogues to the *superpixel* concept in image segmentation. We randomly start with a seed point and group its neighbors into a superpoint based on two thresholds: the maximum number of points and the maximum distance between points. This process repeats until all points are processed. In our experiments, these two numbers are set to 100 and 0.3m, respectively. Second, we use the Adaboost classifiers. Third, we introduce a new concept called *super-region* to differentiate objects with similar superpoint features but at different scales, such as houses and cars, cars and waste bins, etc. We group superpoints by a region growing algorithm with a threshold of the angle difference between superpoint normals; in this way, most coplanar superpoints are grouped into a large super-region and remaining small super-regions can be further combined using a similar grouping method based on distance as above-mentioned. Figure 5.3 shows an example. For each super-region, we compute the following features:

- Height: the maximum height of a super-region.

- Volume: the volume of the bounding box of a super-region.
- Area: the maximum area of the bounding box of a super-region.
- Planarity difference: the sum of difference of planarity among adjacent superpoints in one super-region.
- Length: the length between maximum and minimum superpoint height.

These additional features are added to its superpoints; together with other superpoint features as defined in [126], the augmented feature vectors are used for training. After the classification stage, the result is further grouped into connected components, each with its own semantic label.

For each point set labeled as a house, it will be further segmented into different *classes*, including *columns*, *roofs* and *walls*. The same segmentation method is adopted with one additional superpoint feature: surrounding emptiness, which measures the number of points within a neighboring bounding box around each superpoint center. The bounding box size is set to 0.5m, which is roughly twice the size of a typical column. This feature is mainly used to identify columns.

5.2 Online Building Segmentation from Ground-based LiDAR Data

To further speed-up building segmentation from LiDAR data, we present an automatic and efficient online algorithm to segment buildings from ground-based LiDAR scans in urban scenes. Our main focus is to detect and segment independent buildings that are disconnected from each other; some examples are shown in Figure 5.4.

Figure 5.4: Examples of independent buildings that we aim to segment. Top: a downtown area. Bottom: houses in a suburban area. Notice that, we do not aim to segment the connected sub-facades in the black box.

The basic idea is straightforward. As building facades are often constructed nearly parallel to a street, by projecting a point cloud in a street view, a large dense area can be easily detected as a candidate building. Our main contribution is to automatically segment buildings from large scale unorganized point clouds with online performance and without requiring any training data.

5.2.1 Overview

We focus on segmentation of buildings from ground-level LiDAR data, where both point clouds with rich street-level details and the scanning/driving trajectory can be obtained. We first regularly sample positions every 0.5m along the driving path $\mathcal{P}$ and compute a local orthogonal coordinate system at each position p which consists

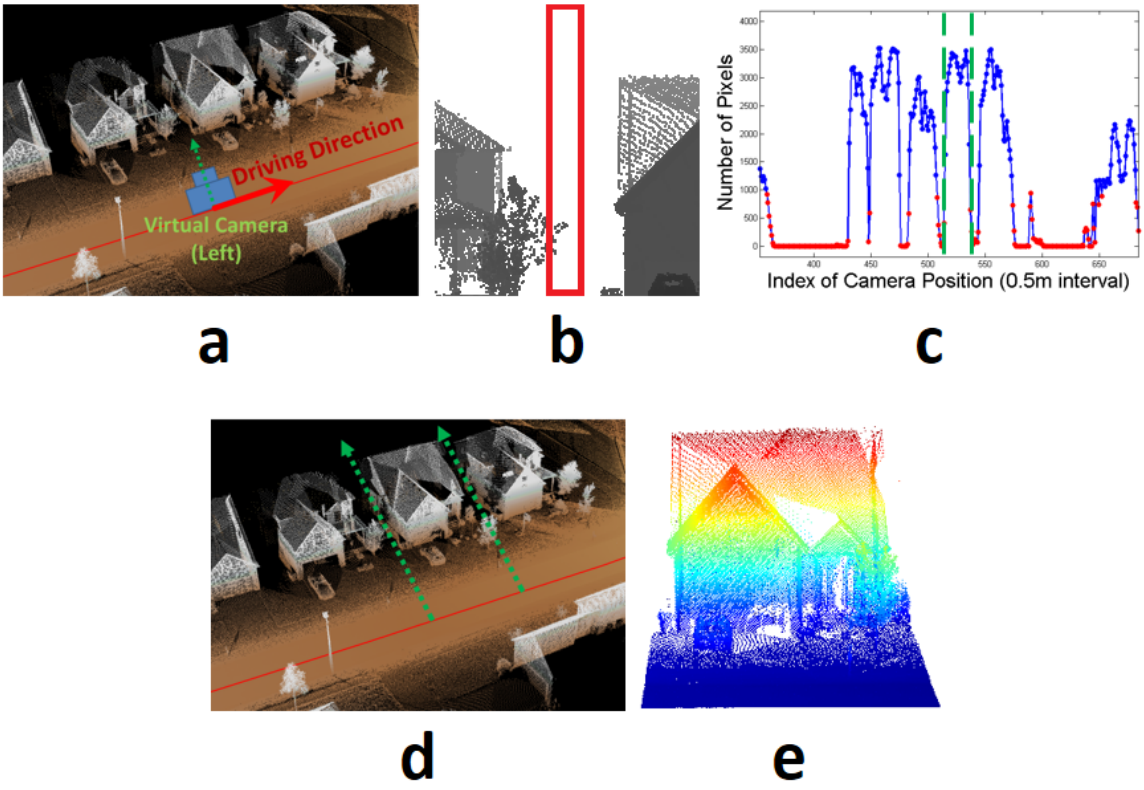


Figure 5.5: Overview of our building segmentation system. The input is a point cloud near a position on the driving path (a), where a virtual camera is placed viewing along the direction (green arrow) perpendicular to the local driving direction (red arrow) and the vector vertically up from the ground to capture the depth map (b) under orthographic projection. We can then create a histogram (c) with the horizontal axis corresponding to the positions sampled at every 0.5m along the driving route and vertical axis corresponding to the number of visible foreground pixels in the box with 1m width in depth maps (e.g., the red box in (b)). A sufficiently long consecutive subsequence where values are all above a certain threshold can be identified as shown in the blue subsequence bounded by two green lines in (c), and the 3D points which lie within the frustum defined by two camera positions corresponding to the endpoints of the subsequence and their viewing directions (d) are regarded as a candidate building (e) with color coding based on height.

of a local driving direction $\mathbf{d}_p$, left-viewing direction $\mathbf{v}_p$ perpendicular to $\mathbf{d}_p$ and the vector vertically up from the ground. At each position p, we can efficiently load the 3D points $\mathbf{X}_p$ at a roughly 50m radius. Next, we virtually place a left-viewing camera C_p^L at position p looking along the direction of $\mathbf{v}_p$. A right-viewing camera

C_p^R can be similarly defined looking along the direction $-\mathbf{v}_p$. As the techniques applied to a left-viewing camera C_p^L also work for a right-viewing camera C_p^R, we only discuss the methods to process C_p^L in the followings. We render a depth map I_p^L by orthographically projecting X_p onto a camera C_p^L. By counting the number of visible foreground pixels n_p^L in I_p^L at each position p, we can generate a histogram H^L with the horizontal axis corresponding to the positions and the vertical axis corresponding to the number of visible foreground pixels. After thresholding H^L, a consecutive subsequence S in H^L where values are all greater than the predefined threshold (1/4 depth map size) and span at least 6m long can be identified and the 3D points which lie within the frustum defined by two camera positions p_1 and p_2 corresponding to the endpoints of S and the viewing directions of p_1 and p_2 are regarded as a building candidate. The whole processing pipeline is summarized in Figure 5.5.

5.2.2 Data Loading

The sheer amount of LiDAR data poses tremendous challenges for point cloud loading, rendering and processing. In order to efficiently manage and process the LiDAR data, we divide the point cloud into several 20m $\times$ 20m tiles along the UTM easting and northing directions. Notice that, a tile only contains an unorganized list of points; we only know the points in a tile are within the 20m $\times$ 20m dimension without any other information on the spatial relationship between points, and thus, we can still say our input point clouds are unorganized. In our cases, the average number of points in a tile is around 700K and the maximum number is around 5.1M. Therefore, it is even possible to load the 5×5 tiled data in a 32-bit machine with single-precision

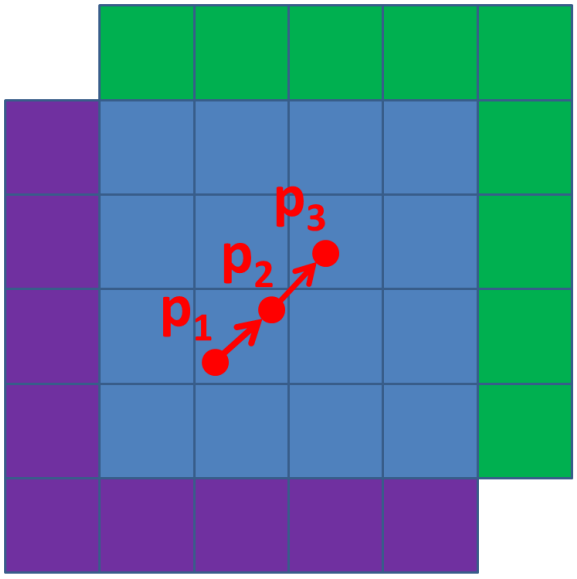


Figure 5.6: Illustration of progressively loading tiled data. Given a position p_1, our system can load the 5×5 tiles around p_1 shown in purple and blue. When it moves to p_2 in the same tile, no update is needed. However, when it moves to p_3 in a different tile, the purple tiles are deallocated and the green tiles are loaded into memory, but the blue ones are unchanged.

(after shifting the large UTM coordinates by a global offset); in practice, loading and processing our data, the memory never exceeds over 1.2GB.

Given a position p, we can easily get the tile T_p containing p and the neighboring 5×5 tiles centered at T_p. Figure 5.6 demonstrates a procedure to progressively load the tiled data as p moves along the driving path. This implies that, at each position p, 3D points $\mathbf{X}_p$ at a roughly 50m radius are loaded. Though our LiDAR sensor can support up to 200m range, in practice, we found the data captured over 40m is already quite sparse for further modeling and rendering (see Figure 5.15). Therefore, 50m is a safe range for efficiently processing the data without sacrificing much performance.

5.2.3 Depth Map Rendering

At a position p, we place two virtual orthographic cameras C_p^L and C_p^R viewing along $\mathbf{v}_p$ and $-\mathbf{v}_p$, respectively. The view volume of a camera is set up with the coordinates for the left and right clipping planes to -0.5m and $+0.5$m, respectively; the coordinates for the bottom and top clipping planes to $+1$m and $+11$m, respectively; and the coordinates for the near and far clipping planes to 0.1m and $+50$m, respectively. An output depth map is rendered with the size of 20×200 pixels (see the red box in Figure 5.5(b)). The parameters reflect physical properties of typical buildings and landscapes. For example, an alley between buildings is usually at least 1m width, a building/house is constructed a little higher than a street but 1m usually works fine to tolerate the height difference, and a U.S. single family house is usually around 10m high, though high-rise buildings are much higher than this.

5.2.4 Building Segmentation

At each position p, we count the number of visible foreground pixels n_p^L in I_p^L, which reflects the number of visible 3D points in C_p^L. As a position moves along the driving path, we generate a histogram $H^L = \{(i(p), n_p^L) | p \in \mathcal{P}\}$, where $i(p)$ represents the integer index of p and the index set $\{i(p)\}$ is consecutive, with the horizontal axis corresponding to the positions and the vertical axis corresponding to the number of visible foreground pixels captured at each position (see Figure 5.5(c)).

At a position p, if $n_p^L \leq 1000$, which is less than 1/4 depth image size, we regard C_p^L as viewing an empty space or an alley between buildings. Accordingly, if a

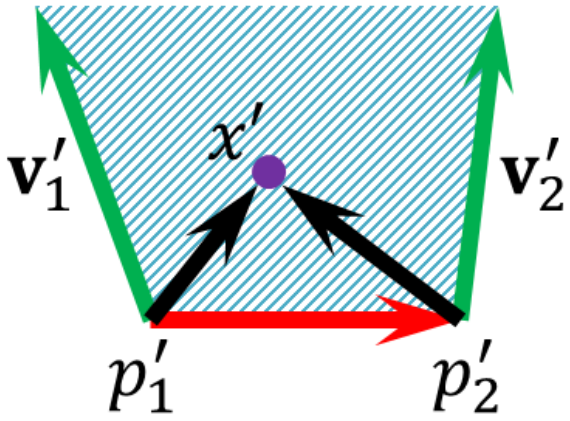


Figure 5.7: Illustration of checking whether the projection x' of a 3D point x lies in the dashed area encapsulated by the projected camera positions p_1' and p_2' and projected viewing directions $\mathbf{v}_1'$ and $\mathbf{v}_2'$.

consecutive subsequence $S = \{(i(p), n_p^L) | n_p^L > 1000\} \subset H^L$ and $|S| >= 12$, which means a camera continuously observes dense areas during a drive at least 6m long as a camera position is sampled every 0.5m, we regard S as corresponding to a candidate building B. (see Figure 5.5(c)(d))

Suppose $i(p_1)$ and $i(p_2)$ are the lower bound and upper bound of the indices of S, respectively, and $n_{p_1}^L, n_{p_1}^L \leq 1000$, the building B represented by S is bounded by two nearby alleys which are viewed by cameras placing at p_1 viewing along the direction $\mathbf{v}_1$, and p_2 viewing along the direction $\mathbf{v}_2$. In order to check whether a 3D point $x \in \mathbf{X}$ belongs to the building B, we first project x, p_1, p_2, $\mathbf{v}_1$ and $\mathbf{v}_2$ onto the UTM easting-northing plane as x', p_1', p_2', $\mathbf{v}_1'$ and $\mathbf{v}_2'$. As shown in Figure 5.7, in order to check whether x' lies within the dashed area encapsulated by the projected camera positions p_1' and p_2' and projected viewing directions $\mathbf{v}_1'$ and $\mathbf{v}_2'$, we compute the following equations: $a = \mathbf{u} \cdot (\overrightarrow{p_1' p_2'} \times \overrightarrow{p_1' x'})$, $b = \mathbf{u} \cdot (\overrightarrow{p_1' x'} \times \mathbf{v}_1')$, $c = \mathbf{u} \cdot (\mathbf{v}_2' \times \overrightarrow{p_2' x'})$, where $\mathbf{u} = (0, 0, 1)^T$. It can be easily verified that x' is in the dashed area if $a \geq 0$,

Figure 5.8: Pictures of various building styles in chronical orders. From left to right: (a) American Colonial styles between the 17 and 19th century; (b) Neoclassical style (early 19th century) that reflects classic ideas of order and symmetry; (c) Victorian house styles (late 19th century) with elaborated decorations; (d) Bungalow Styles in the early 20th century, compact, economical and informal; (e) "Neo" house styles (recently built homes) that borrow details with historic styles and combine them with modern features.

$b \geq 0$ and $c \geq 0$ for a left camera C_p^L, or $a \leq 0$, $b \leq 0$ and $c \leq 0$ for a right camera C_p^R. In this way, all points in B can be segmented.

5.3 House Modeling

Given the segmented buildings, our next focus is automatic modeling of stand-alone buildings, such as single-family houses in suburban areas. As shown in Figure 5.8, there are many different building styles. Compared to high-rises in downtown, the symmetry and repetitiveness are less dominant. This is particularly true for recently built houses with the "neo" style that combine different historic styles with new features. Nevertheless, we can see that even though there are many variations in building styles, the fundamental structures are the same: *a combination of convex*

blocks with tilted roofs. The variations in styles are usually manifested in terms of construction materials, level of decorations, layout of windows, and slopes of roofs, etc. None of them changes the underlying structural semantics of a building.

The key to our modeling pipeline is to decompose and reconstruct the segmented building point cloud into basic *blocks* using a few pre-defined reconstruction constraints, namely, *planarity*, *symmetry* and *convexity*. With these basic blocks in place, details such as columns and eaves are further extracted and processed. The algorithm is proposed by Hui Lin and detailed in [66]. By using the planarity and symmetry assumption, this method can successfully reconstruct complete water-tight 3D building models with sematic tags from significant incomplete and noisy LiDAR scans.

5.4 Texture Mapping

To compute texture maps for a reconstructed building model, we first automatically find the nearby camera views that capture the model, and then back-project the views to the planar surfaces of the model. However, images and point clouds may not be perfectly aligned, and a model is only approximately reconstructed. Thus, texture maps generated from the direct back projection would produce noticeable misalignment. In order to alleviate these errors, we propose an algorithm of content-preserving warps inspired by [68] based on 2D-3D line correspondences. We further fuse the multiple overlapping views using a multi-label MRF energy minimization framework similar to [95]. Details are presented below.

5.4.1 Back-projection

As the reconstructed model M and images are geo-referenced and the scanning trajectory is also recorded, the images capturing M can be easily retrieved. In order to compute the texture map T_p of a plane P in M from an image I, as P might be entirely or partially occluded by other planes in M and/or nearby models (typically two along the driving path), we need to perform a visibility test on P first. We use the standard two-pass z-buffer algorithm: given the projection matrix of I, we render M and nearby models and then render the plane P to get the visibility estimation and the texture map through back-projection.

5.4.2 Content-preserving Warps based on 2D-3D Line Correspondences

The texture maps generated from the direct back projection would produce noticeable misalignment due to the registration error and imperfect model fitting. We propose an algorithm of content-preserving warps to generate a distorted image I' where 2D edges are better aligned with 3D lines in M. Our method is inspired by [68], however, different from point correspondences used in their method, we establish 2D-3D line correspondences between a model M and an image I because line features are much more stable than point features across different modalities.

We extract a set of line segments from an image I using the LSD algorithm [110]. We denote a projected 3D edge segment of M onto an image I as l^M. An image edge segment l^I is matched to the model edge segment l^M if they have short distance (less than 20 pixels), small angle (less than 15°), similar length (the longer one is no

more than twice the length of the shorter one), and sufficient overlapping (at least half of the shorter one can be projected onto the other segment). In order to prune the wrong line matches, one observation is that the larger the model plane is, the more likely line matches can be found. Therefore, we sort the model planes according to plane size in a descending order, and iteratively apply robust planar homography fitting algorithm based on RANSAC [25] to prune wrong line matches for each model plane, so that model edges from the large planes find their matches first, and model edges from the small planes can be adjusted accordingly.

To refine a texture map, we synthesize a new image I' from an image I where matched 2D image edges and projected 3D model edges are co-aligned. We use the content preserving warps similar to [68], which divides an image into a quad mesh and compute a distorted quad mesh minimizing an energy functional consisting of a data term and a smoothness term. Different from their method, we encode 2D-3D line matches into the data term instead of point correspondences. Our new data term ensures a line l^I to align l^M by minimizing the total square of point-to-line distance between sample points on line l^I and line l^M. The warped image I' can then be generated by a standard texture mapping algorithm according to the distorted quad mesh. We then compute a texture map for each model plane through the back-projection algorithm based on the warped image I'. As shown in Figure 5.9, the misalignment artifacts can be reduced by our approach.

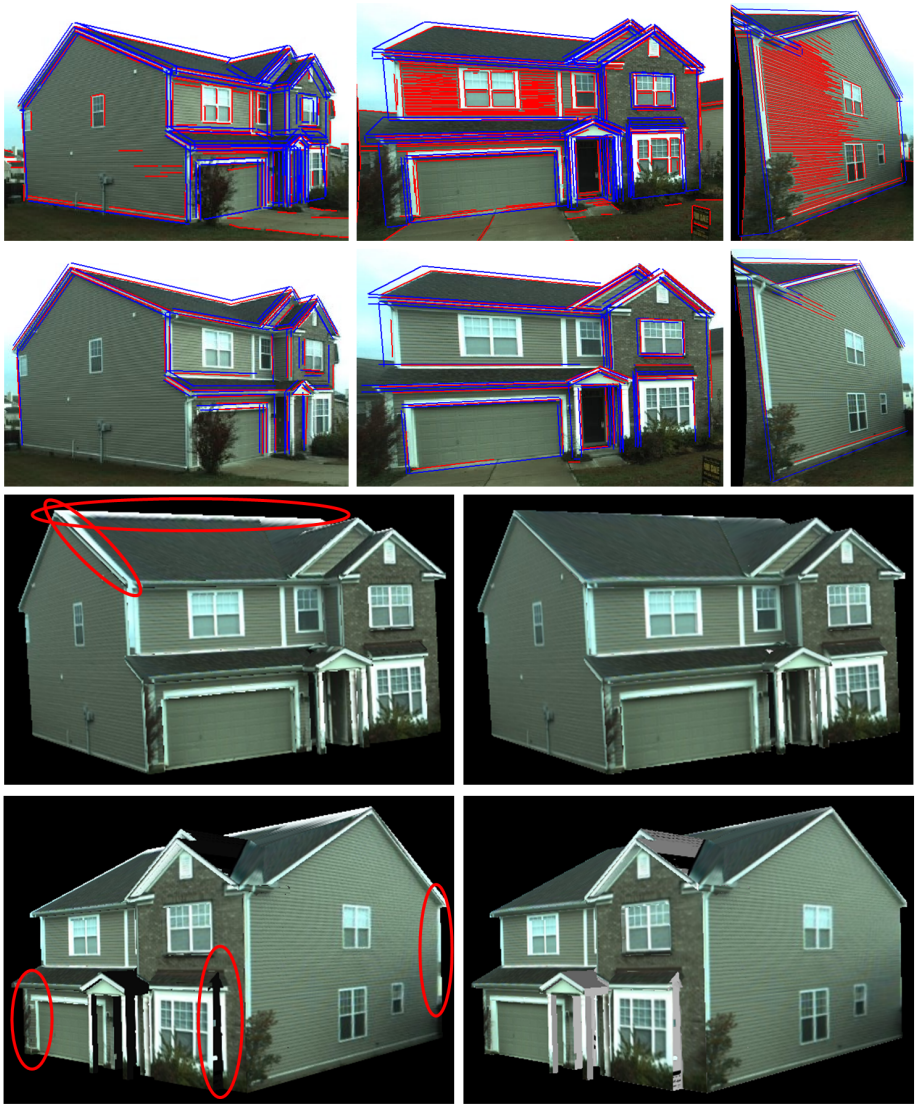


Figure 5.9: Comparison on two texture mapping algorithms. 1st row shows the detected image edges (red) and projected model edges (blue); the misalignment is obvious due to the registration error. 2nd row shows the automatically extracted correspondences between image edges (red) and model edges (blue). In 3rd row and 4th row, the left part shows the results from the original back-projection texture mapping where the imperfection marked by red circles are improved by our method shown in the right part.

5.4.3 Multiple Texture Fusion

To fuse the texture maps for a model plane P from overlapping camera views, we apply a multi-label MRF energy minimization framework similar to [95], where the

data term is a weighted sum of two terms: preference for a frontal view and preference for a camera view that has a large number of visible pixels of P's projection; and the smoothness term encourages neighboring pixels should have same labels.

5.5 Landscape Modeling

Given the semantic labels for the remaining points, we have developed simple and effective methods to reconstruct other objects. These objects enrich the visual realism of the final model.

5.5.1 Plant Reconstruction

Plants, including trees and shrubberies, are integral parts of our living environment. A number of reconstruction methods have been developed to model trees from LiDAR data (e.g., [70], or even images (e.g., [104]). While these methods can generate very high quality geometric tree models, we choose to develop a fully automatic method suitable for large-scale reconstruction. Toward this goal, we choose to adopt the light-weight billboard representation for visually-plausible plant models. Our plant model consists of two orthogonal planes and one billboard image. The key is to extract from the input the billboard image and find the tree trunk (the axis of rotation).

First, we project each plant point set onto the nearest corresponding image. A shape mask is extracted as the billboard image. The point set is also projected onto the ground plane normal direction (i.e., the z axis) and the point density along the z axis is measured. A sharp increase (3 times) in density means a transition from trunk to foliage. If no trunk is detected, the center z axis of 3D bounding box is used

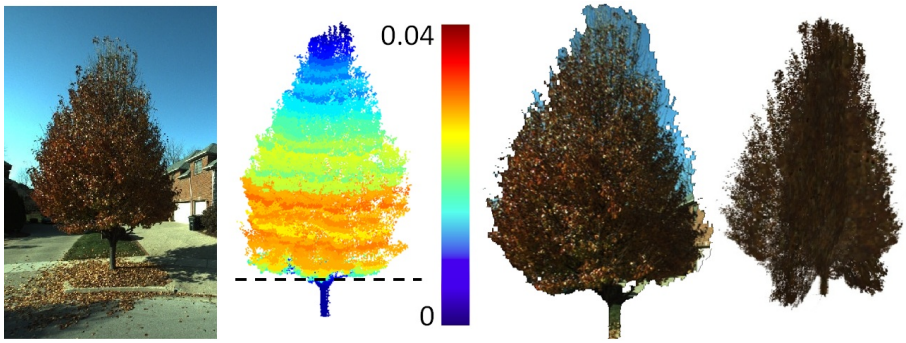


Figure 5.10: Tree Reconstruction. From left to right: (a) corresponding color image of point clouds; (b) point clouds with color coded horizontal density, the dotted line shows the trunk/leaf boundary; (c) the texture and point clouds are off by some pixels; (d) rendered billboard tree with the corrected texture.

as the axis of rotation. If the ratio of x and y axes of the 3D bounding box is too big or too small, it is likely to be a shrubbery, therefore we segment it into parts with equal length of x and y axes. We usually use a 2:1 ratio as the threshold for cutting. Another problem we have to deal with is misalignment of images. We segment the billboard images into superpixels and cluster the superpixels into three groups (trunk, leaf, and error) according to average chroma value of pixels in the image. Then the error superpixels will be automatically replaced by randomly selected leaf superpixels. Figure 5.10 illustrates the entire process.

5.5.2 Model Replacement

Other frequently occurring static objects in residential landscape such as mailboxes and street lights are often hard to directly reconstruct using simplified geometric models from the cluttered, incomplete and noisy data. Inspired by the recent success of model replacement applied to indoor scene modeling (e.g., [93]), we download the similar models from Google 3D Warehouse, and use PCA to estimate global scaling

and an initial pose between the models and the recognized raw points and further align them using iterative closest point (ICP) method; in this way, points from a categorized object (e.g., a mailbox) are replaced by the corresponding model.

5.6 Experiments

5.6.1 Results on Semantic Segmentation

Figure 5.11 shows two datasets from our scanning platform. The top (*Div-A*) is a upscale residential area (containing over 150 million points) while the bottom (*Div-B*) is an average residential area (containing over 183 million points) that was newly built.

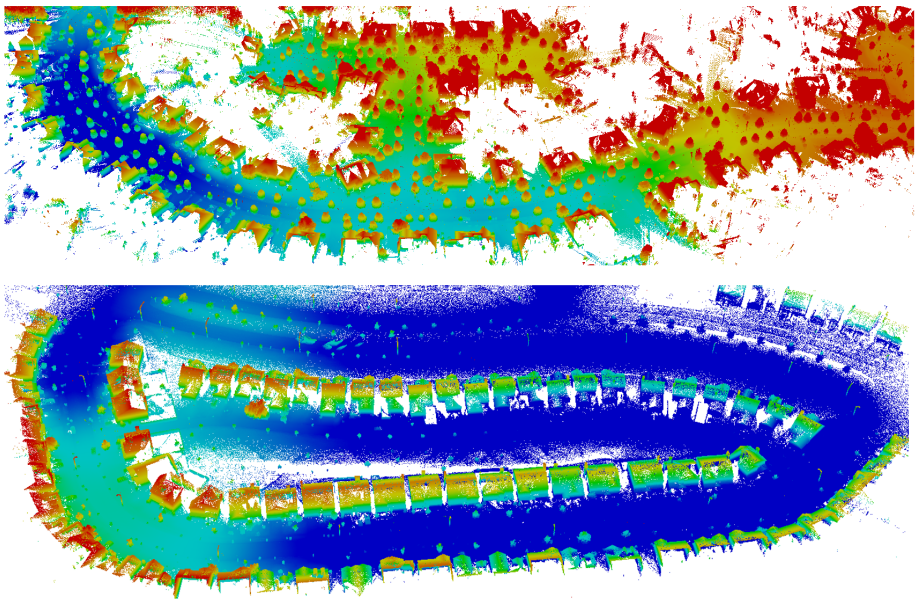


Figure 5.11: Ground-based LiDAR Datasets: (top) a high-end subdivision (Div-A); (bottom) an average subdivision that is newly built (Div-B).

In order for training and evaluation, we have labeled both *Div-A* and *Div-B*. We used *Div-A* for training and *Div-B* for evaluation. One segmented scene is shown

Table 5.1: Precision and recall of semantic segmentation.

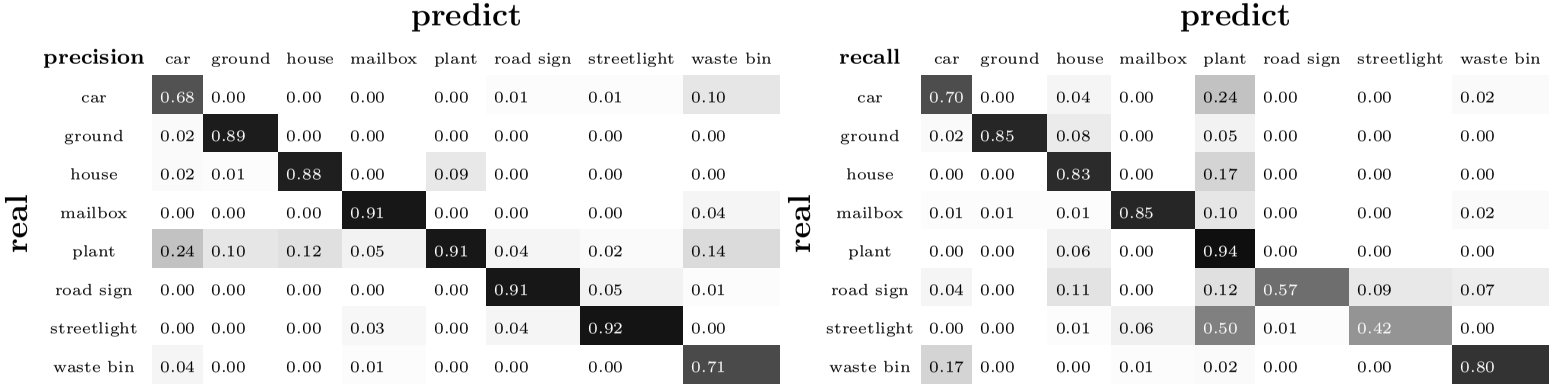


Table 5.2: Data set specification and performance evaluation

Data Set	#Points (Million)	Length (m)	Speed (km/h)	Processing Time (s)	#Segments	#Missing	#Imperfect	#Wrong
Subdivision 1	184.28	2672	5.17	81.01 + 228.76	231	10	7	9
Subdivision 2	61.32	3887	26.06	58.68 + 22.52	133	31	13	16
Downtown	139.20	3614	12.89	75.82 + 137.72	55	3	4	11

in Figure 5.1(a). The precision and recall of automatic segmentation is shown in Table 5.1. Compared to previous semantic segmentation methods (e.g., [126]), our accuracy is noticeably higher, partially due to our high-quality input. We also report the precision (P) and recall (R) of the house classes: *walls* ($P = 94\%$, $R = 98\%$), *roofs* ($P = 94\%$, $R = 81\%$) and *columns* ($P = 72\%$, $R = 87\%$). *It should be emphasized that all of our subsequent modeling results use only semantic labels from the classifier's output, not the ground-truth.* In addition, our house modeling is not sensitive to mislabeling since it is highly regularized by various constraints.

Timing Performance Our semantic segmentation algorithm takes about 40 minutes for labeling *Div-B* including feature extraction, classification and segmentation. This residential subdivision includes 53 houses.

5.6.2 Results on Online Building Segmentation

We have also tested our online building segmentation algorithm on three large-scale data sets: (1) Subdivision 1, a very dense dataset of a typical U.S. subdivision of mass-producted single-family houses contains 184.28M points collected from a $2,672$m drive with average driving speed 5.17km/h; (2) Subdivision 2, a sparse dataset of another subdivision contains 61.32M points collected from a $3,887$m drive with average driving speed 26.06km/h; (3) Downtown, a moderately dense dataset of a typical U.S. downtown area of buildings varying in either sizes or styles contains 139.20M points collected from a $3,614$m drive with average driving speed 12.89km/h. The results are shown in Figure 5.12, Figure 5.13 and Figure 5.14. A scanning path is shown

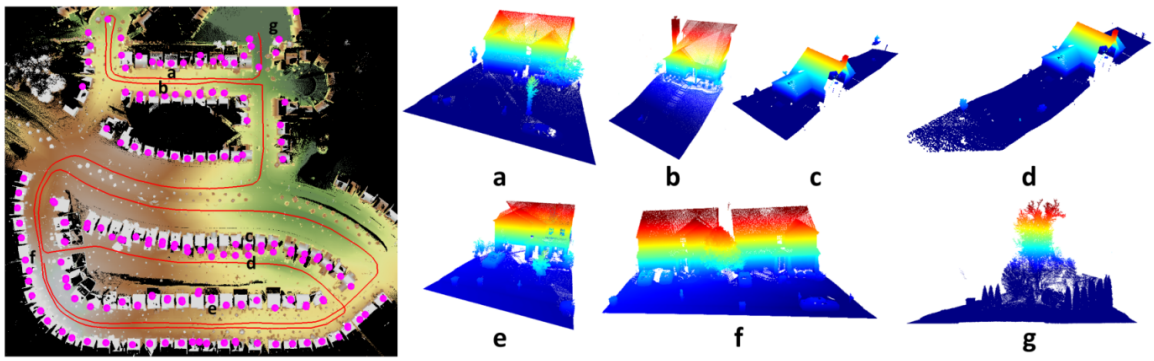


Figure 5.12: Results on the Subdivision 1 dataset. Left: The whole point cloud is rendered based on height in the earth tone color scheme overlaid with the scanning path in red and detected buildings labeled with purple markers. Right: Some examples of detected "buildings" corresponding to the labels shown in the left image with color coding based on height from blue to red.

as a red line strip. For better visualizing a detected building B, we compute the mean coordinate for all points in B whose elevation is 3m higher than the ground height along the corresponding scanning path; each mean coordinate representing a detected building is shown as a purple marker. Please refer to the electronic PDF copy at high magnification for better viewing. We propose the following three measures to quantitatively evaluate our algorithm performance: (1) Detection Rate (DR) = #Correct Detection (without duplicates) / #All Buildings, (2) Perfect Segmentation Rate (PSR) = #Successful Segmented Buildings / #Correct Detection, (3) Correctness Rate (CR) = #Correct Detection / #All Detection.

Figure 5.12 shows the results of building detection and segmentation on Subdivision 1 dataset. There are total 231 buildings detected and segmented, with 10 buildings missing detected along the driving path, 7 imperfect segments of the cases that a building is split into parts and two buildings are wrongly merged in one segment, and 9 false positives; DR = 92.31%, PSR = 94.17%, and CR = 96.10%. Notice

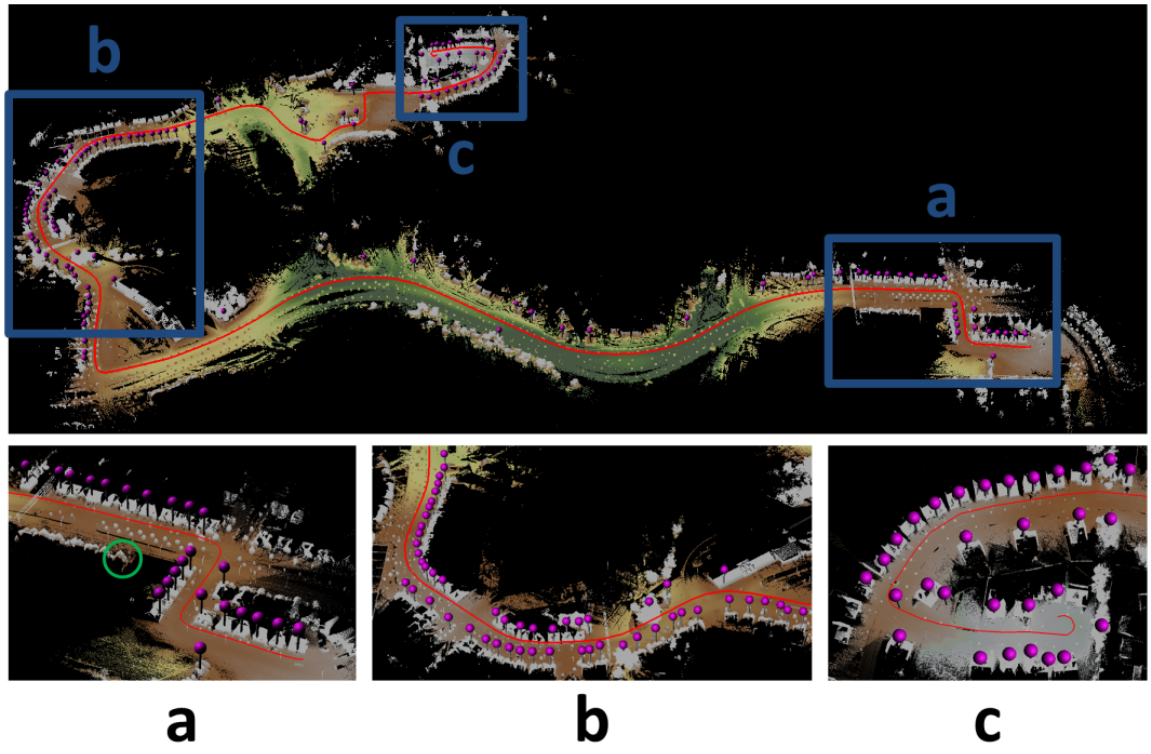


Figure 5.13: Results on the Subdivision 2 dataset. Top: The whole point cloud is rendered based on height in the earth tone color scheme overlaid with the scanning path in red and detected buildings labeled with purple markers. Bottom: Some close-up views highlighting the labeled districts in top.

that, in this scan, some houses are scanned more than once, so that there are duplicates in the detected buildings and the purple markers representing the same building may not be exactly co-located. For example, Figure 5.12(c) and (d) contain the same building, but one is scanned from the front facade and the other is scanned from the backyard. In both cases, the points are enclosed from the corresponding scan path and extending around 50m towards the opposite side, so that the resulting excerpted points have different landscapes and calculated mean locations (purple markers) are shifted. The duplicates can be merged by using a simple heuristic approach, that is checking whether a large amount of data are shared among nearby segments, which

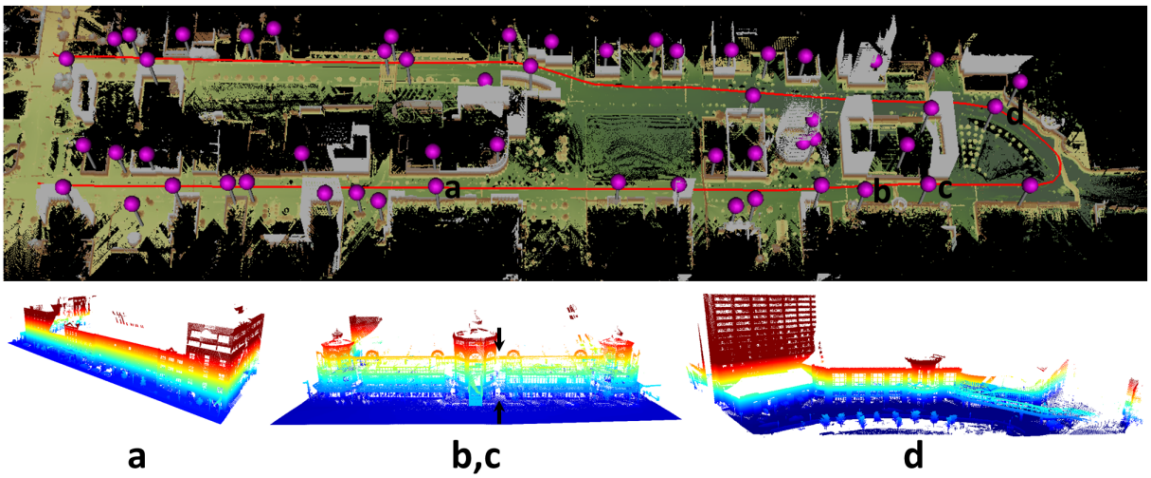


Figure 5.14: Results on the Downtown dataset. Top: The whole point cloud is rendered based on height in the earth tone color scheme overlaid with the scanning path in red and detected buildings labeled with purple markers. Some examples of detected buildings corresponding to the labels shown in the top image with color coding based on height from blue to red. Notice that the middle building is split into two segments due to the insufficient points acquired at the seam indicated by black arrows.

can be efficient to compute the overlapping area ratio between two dashed regions as shown in Figure 5.7.

Figure 5.13 shows the results on Subdivision 2 dataset. There are total 133 segments, with 31 buildings missing detected, 13 imperfect segments and 16 false positives; DR = 79.05%, PSR = 88.89%, and CR = 87.97%.

Figure 5.14 shows the results on Downtown dataset. There are total 55 segments, with 3 buildings missing detected, 4 imperfect segments and 11 false positives; DR = 93.62%, PSR = 90.91%, and CR = 80.00%.

The data set specification and performance evaluation is summarized in Table. 5.2. The overall DR = 86.46%, PSR = 91.46%, and CR = 91.41% across different data sets varying in either point density or building and landscape styles.

Timing Performance We have implemented our algorithm on a desktop PC with an Intel® Core™ i5 processor 3.20GHZ, 4GB of RAM, AMD® Radeon™ HD 7570 graphics card and OCZ® Vertex 4 solid state drive (SSD). The input and output data are all in binary format. Our current unoptimized implementation runs on a single core with the OpenGL rendering pipeline. In the column of "Processing Time" in Table. 5.2, the first number is the run time including data loading, depth map rendering, histogram analysis and building detection for a whole dataset. The second number is the run time for cutting out and outputting the points of all detected buildings. For the Subdivision 1 dataset, loading the 184.28M points, rendering $5,346$ depth maps and detecting and locating the 231 objects takes 81.01 seconds and segmenting and saving the dense points for the detected buildings takes 228.76 seconds; the total run time is 309.77 seconds and frames per second (FPS) is 17.26 in terms of processing the $5,346$ depth maps. As the point density in Subdivision 2 dataset is sparse, the number of points in each segment is much less and the resultant run time for exporting data is only 22.52s and the FPS is 95.76. The FPS for the Downtown dataset is 33.86. To sum up, our algorithm is very efficient and achieves online performance.

Limitations

In addition to the connected sub-facades that our approach is not designed to handle [128], there are some other limitations in our proposed method.

Missing detection As point density decreases with the distance from the sensor, it is expected that the a building far away from the scanning path has much less points.

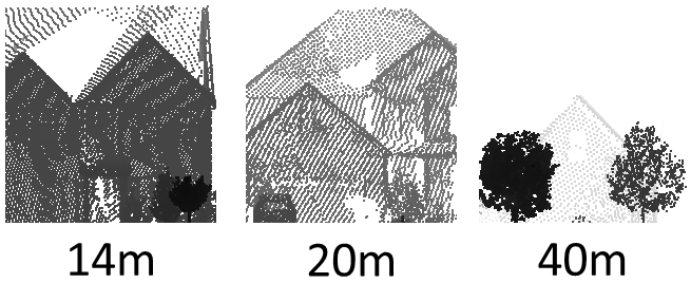


Figure 5.15: Depth map comparison of point density changes at different distances from the Subdivision 2 dataset. Notice that, the driving speeds around the three positions are almost the same, so that distance-to-sensor plays the key role in point density changes.

Figure 5.15 shows a depth map comparison of point density changes at different distances. The rightmost depth map in Figure 5.15 captures the building marked in the green circle in Figure 5.13, and because of the insufficient points acquired at a large distance, our algorithm fails to detect it as a building; however, if the building was detected, it would be very hard to perform further manipulation such as modeling and rendering on this sparse point cloud. This explains why there are a relatively large number of missing detected buildings that are far away from the scanning path in the Subdivision 2 dataset.

Imperfect segments When there are flourishing trees or shrubs planting in front of the space in-between two buildings, our algorithm may falsely combine the two buildings and the trees in a single segment, for example Fig 5.12(f).

Another situation is that a building might be split into two or more segments because of insufficient points (holes) in a building due to transparent objects like glass (see Figure 5.14(b,c)) and structures with self-occlusion like concave areas (see Figure 5.16). And if none of parts is longer than 6m wide, it also causes missing detection.

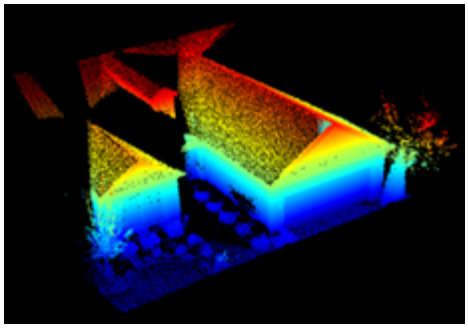


Figure 5.16: An example of a building with self-occlusion. The concave area is not scanned at all, resulting in two separate houses instead of one.

False positives Our current implementation cannot distinguish well between buildings and flourishing trees and shrubs, for example Figure 5.12(g). However, we can still achieve overall CR=91.41% in our data sets of different districts of a city. In a case of scanning on the road with abundant street trees, our algorithm might be brittle; but most likely, buildings are severely occluded by the trees making it hard to perform further reconstruction and rendering on insufficient point samples with many holes on buildings. Scene Parsing algorithms on color images and some point cloud processing techniques like local normal distribution and local plane fitting can help to reduce the false detection but might incur high computation overhead.

5.6.3 Reconstruction Results

We show modeling results on *Div-A* and *Div-B* datasets. Since the symmetry and convexity of the basic blocks are the only assumptions we make, our algorithm is capable of modeling a variety of houses and dealing with significant missing data. As shown in Figure 5.17 (1st, 2nd and 3rd rows), we correctly model the houses composed of several gables and box structures at different scales, location and nested

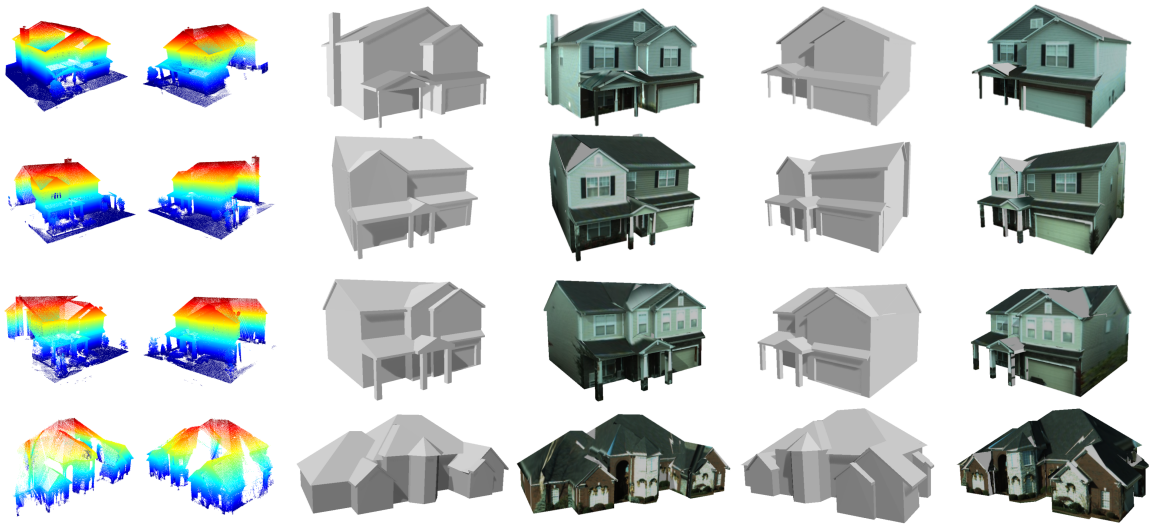


Figure 5.17: Results on ground-based LiDAR datasets. Each row shows the result of one house. 1st and 2nd columns: point clouds color coding based on height in two perspectives, 3rd and 5th columns: reconstructed models, and 4th and 6th columns: reconstructed models with textures.

structures, and with missing point scan at various levels.

All houses in Div-B are automatically reconstructed together with other categorized and modeled objects such as trees and mailboxes in the landscape can be seen in the accompanying video. Figure 5.1(d) shows the overview of the large-scale reconstruction results overlaid on the geo-registered satellite image.

More complex structures and complicated composition of building blocks are often found in *Div-A*. As shown in Figure 5.17 (4th row) and Figure 5.18, the houses consist of multiple gables, hipped structures and even octagonal shapes. Figure 5.18 shows an overall view of the reconstructed *Div-A* dataset without textures.

Figure 5.18: An overall view of reconstructed houses in Div-A.

5.7 Summary

In this chapter, we present a complete system for residential scene modeling from 3D point cloud captured by mobile scanners. By first recognizing individual objects, we develop and apply category-specific reconstruction methods to obtain visually

pleasing polygonal models in the presence of occlusions and incomplete data. Our system requires very little human intervention and therefore is suitable for large-scale modeling. We also present an algorithm to automatically segment buildings from unorganized 3D point clouds. The key idea is to project points onto the street views along the scanning path and identify dense regions as buildings separated by sparse spaces as alleys. Our algorithm is easy to implement and very efficient with online performance. Experiments show that our approach is able to produce satisfactory results for data sets from different districts in a city varying in either point density and building and landscape styles.

Chapter 6 Conclusion and Future Work

In this dissertation, I have explored the semantic segmentation of a collection of images and LiDAR point clouds. My research work consists of three parts:

- Repetitive Pattern Mining and Segmentation from Images
- Image Segmentation based on Internet Photo Collections
- Semantic Segmentation and Reconstruction of Residential Scenes from LiDAR Data

In this chapter, I summarize the technical innovations and propose several possibilities for future work.

6.1 Innovations

This dissertation has included the following five innovations:

- **Finding Repetitive Patterns in Images** I investigate finding frequent and consistent associations among visual items in an image regardless of their size, shape and orientation. I develop a polynomial-time algorithm for mining maximal associations between pairs of visual items, and I develop a polynomial-time method for extracting frequent high-order structural visual patterns among more than two visual items. Experiments on synthetic data and a variety of real-world datasets demonstrate the efficiency and effectiveness of our methods.

- **Automatic Visual Pattern Segmentation** I develop a novel framework to segment frequent structural patterns from a large collection of images. Unlike the traditional segmentation approaches that involve user interactions, my approach utilizes the redundant information brought by the images themselves to segment the re-occurring visual patterns in a fully automatic and supervised manner. To achieve this goal, I first established dense correspondences among pattern instances, then computed a per-pixel cost, and finally extracted the target object. As demonstrated by the results, my approach is effective at segmenting meaningful visual patterns from a variety of images.

- **Automatic Image Segmentation based on Internet Photo Collections** I develop a novel segmentation algorithm that automatically differentiates the foreground from the background using image appearance statistics obtained from Internet photo collections that were taken at landmark locations and that survey the same scenes under a variety of lighting conditions. Compared to many existing segmentation methods, my method requires no user interaction to obtain a high quality foreground segmentation.

- **Semantic Segmentation and Reconstruction of Residential Scenes from LiDAR Data** I design a complete system for modeling residential scenes using 3D point clouds captured by mobile scanners. After first recognizing individual objects, I apply category-specific reconstruction methods to obtain visually pleasing polygonal models despite occlusions and incomplete data. Our system requires very little human intervention and is therefore suitable for large-scale

modeling.

- **Online Building Segmentation from Ground-based LiDAR Data in Urban Scenes** I develop a fast and accurate building segmentation algorithm from ground-based LiDAR points. The basic idea is that buildings observed from a street view can be separated by empty spaces such as alleys. By progressively projecting 3D points onto street views along the scanning path, buildings can be detected as large regions with dense points. The main contribution is to automatically segment buildings from large scale unorganized point clouds with online performance and without requiring any training data.

6.2 Future Work

I believe there are many directions for future research in this area. First, I plan to utilize the semantic information that obtained from frequent visual pattern mining to perform object extraction and modeling. In particular, this information can be used to automatically recover partially occluded regions, as long as they are a part of repeated visual patterns. The discovered patterns can also be used in compression. We already see some progress in this regard. In [111], Wang et al. use a brute force method to find repeated patches within a single image for compression. While some very good results have been obtained, their method will not scale up. Our efficient method will enable compression on a large scale and can potentially increase the compression ratio.

Second, my current automatic segmentation approach is unable to separate fore-

ground layers in a cluttered scene. I think it is worth exploring the information not only from Internet photo collections, but also from one's personal photo collections in order to automatically find frequently appearing persons as a guideline for foreground priors.

Third, regarding the semantic segmentation and reconstruction of residential scenes, we should be able to use the semantic labels inherent in our model to support editing and re-targeting. In terms of modeling quality, some fine details on the house, such as hand-rails and staircases, are not reconstructed in our current approach. These details are only represented with a few sparse points. They are however, visible in the images. I plan to improve our segmentation algorithm by using both depth and color to recognize and replace these details. In addition, I plan to automatically cluster points in the same category to find different objects. Overall, the combination of pattern recognition algorithms with geometry processing should lead to better models that support not only high-fidelity visualization, but also editing and eventually searching.

Last but not least, I think there are some ways to further improve my online building segmentation method. First, I would like to extend our current single-core implementation to multi-core or even to GPU to accelerate the point cloud segmentation. Second, we can work with a fully organized point cloud with rich information on the adjacent points, so that the nearest neighbor operations, normal computation and local plane fitting should be much more efficient. Therefore, we can not only speed up the bottleneck of segmenting and exporting point clouds, but also efficiently filter out many false positives by using simple yet effective techniques

like computing local normal distribution. Third, by incorporating registered color data, we believe it is possible to further improve the performance and to segment sub-facades in a downtown scene such as [128]. Fourth, when a scanning path is not readily available, as the LiDAR data are geo-referenced, we can import road information from a GIS database or simply let a user interactively draw a path to segment buildings along a street.

Bibliography

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *ICCV*, pages 72–79, 2009. 14

[2] A. Agarwala, M. Agrawala, M. Cohen, D. Salesin, and R. Szeliski. Photographing long scenes with multi-viewpoint panoramas. *SIGGRAPH*, 2006. 50

[3] N. Ahuja and S. Todorovic. Extracting texels in 2.1d natural textures. *ICCV*, 2007. 14

[4] A. Akbarzadeh, J.-M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewnius, R. Yang, G. Welch, H. Towles, D. Nistr, and M. Pollefeys. Towards urban 3d reconstruction from video. In *3DPVT*, 2006. 17

[5] B. Alexe, T. Deselaers, and V. Ferrari. Classcut for unsupervised class segmentation. *ECCV*, 2010. 13

[6] D. Anguelov, B. Taskarf, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of markov random fields for segmentation of 3d scan data. In *CVPR*, 2005. 15

[7] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik. From contours to regions: An empirical evaluation. *CVPR*, 2009. 1

[8] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. icoseg: Interactive co-segmentation with intelligent scribble guidance. *CVPR*, 2010. 13

[9] C. Berge. Two theorems in graph theory. In *Proceedings of the National Academy of Sciences*, 1957. 29, 32

[10] M. Bokeloh, A. Berner, M. Wand, H.-P. Seidel, and A. Schilling. Symmetry detection using line features. *Computer Graphics Forum (Proceedings of Eurographics)*, 2009. 18

[11] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in ND images. *IJCV*, 2001. 1, 15, 40, 44

[12] G. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla. Segmentation and recognition using structure from motion point clouds. In *ECCV*, 2008. 2, 17

[13] L. Cao and L. Fei-Fei. Spatially coherent latent topic model for concurrent object segmentation and classification. *ICCV*, 2007. 2, 13

[14] M. Carlberg, J. Andrews, P. Gao, and A. Zakhor. Fast surface reconstruction and segmentation with ground-based and airborne lidar range data. In *3DPVT*, 2008. 16

[15] G. Carneiro and A. D. Jepson. Flexible spatial configuration of local image features. *PAMI*, 2007. 24

[16] G. Carneiro and D. Lowe. Sparse flexible models of local features. In *ECCV*, 2006. 4, 13

[17] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *CVPR*, 2010. 19

[18] Y. S. Chia, S. Zhuo, R. K. Gupta, Y.-W. Tai, S.-Y. Cho, P. Tan, and S. Lin. Semantic colorization with internet images. *ACM Transaction on Graphics(TOG)*, 2011. 14

[19] K. Dale, M. K. Johnson, K. Sunkavalli, W. Matusik, and H. Pfister. Image restoration using online photo collections. In *International Conference on Computer Vision*, pages 2217–2224, 2009. 14

[20] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. In *SIGGRAPH*, pages 11–20, August 1996. 17

[21] P. Dorninger and N. Pfeifer. A comprehensive automated 3d approach for building extraction, reconstruction, and regularization from airborne laser scanning point clouds. *Sensor*, 8:7323–7343, 2008. 19, 73

[22] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 2005. 4, 13

[23] R. Fergus, P. Perona, and A. Zisserman. Object class recognition by unsupervised scale-invariant learning. In *CVPR*, 2003. 4, 13

[24] R. Fergus, P. Perona, and A. Zisserman. A sparse object category model for efficient learning and exhaustive recognition. In *CVPR*, 2005. 4, 13

[25] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communication of the ACM*, 24:381–395, 1981. 16, 40, 86

[26] J.-M. Frahm, P. Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building rome on a cloudless day. In *ECCV*, 2010. 6, 14, 17, 18

[27] S. Friedman and I. Stamos. Real time detection of repeated structures in point clouds of urban scenes. In *3DIMPVT*, 2011. 16

[28] C. Frueh, S. Jain, and A. Zakhor. Data processing algorithms for generating textured 3d building facade meshes from laser scans and camera images. *Int. J. Comput. Vision*, 61:159–184, 2005. 17, 18

[29] C. Frueh and A. Zakhor. Constructing 3d city models by merging aerial and ground views. *IEEE Computer Graphics and Applications*, 23(6):52–61, 2003. 6, 19

[30] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1434–1441. IEEE, 2010. 14, 63

[31] J. Gao, Y. Hu, J. Liu, and R. Yang. Unsupervised learning of high-order structural semantics from images. In *The 12th IEEE International Conference on Computer Vision*, 2009. 7

[32] J. Gao and R. Yang. Online building segmentation from ground-based lidar data in urban scenes. In *Third Joint 3DIM/3DPVT Conference*. IEEE, 2013. 9

[33] R. Garg, H. Du, S. M. Seitz, and N. Snavely. The dimensionality of scene appearance. In *ICCV*, pages 1917–1924, 2009. 14

[34] A. Golovinskiy and T. Funkhouser. Min-cut based segmentation of point clouds. In *IEEE Workshop on Search in 3D and Video (S3DV) at ICCV*, Sept. 2009. 15

[35] A. Golovinskiy, V. Kim, and T. Funkhouser. Shape-based recognition of 3d point clouds in urban environments. In *ICCV*, 2009. 15, 74

[36] R. C. Gonzalez and R. E. Woods. *Digital Image Processing.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2nd edition, 2001. 11

[37] C. Gu, J. Lim, P. Arbeláez, and J. Malik. Recognition using regions. *CVPR*, 2009. 2, 13

[38] J. Hays and A. A. Efros. Scene completion using millions of photographs. *Commun. ACM*, 51(10):87–94, 2008. 14

[39] J. H. Hays, M. Leordeanu, A. A. Efros, and Y. Liu. Discovering texture regularity as a higher-order correspondence problem. *ECCV*, 2006. 14

[40] D. Hochbaum and V. Singh. An efficient algorithm for Co-segmentation. *ICCV*, 2009. viii, 13, 58, 60

[41] W. Hsu, J. Dai, and M. L. Lee. Mining viewpoint patterns in image databases. In *SIGKDD*, 2003. 12

[42] A. Irschara, C. Zach, M. Klopschitz, and H. Bischof. Large-scale, dense city reconstruction from user-contributed photos. *Computer Vision and Image Understanding*, 116(1):2–15, 2012. 18

[43] M. Jamieson, A. Fazly, S. Dickinson, S. Stevenson, and S. Wachsmuth. Learning structured appearance models from captioned images of cluttered scenes. *ICCV*, 2007. 24

[44] M. K. Johnson, K. Dale, S. Avidan, H. Pfister, W. T. Freeman, and W. Matusik. Cg2real: Improving the realism of computer generated images using a large collection of photographs. *IEEE Transactions on Visualization and Computer Graphics*, 2011. 14

[45] N. Joshi, W. Matusik, E. Adelson, and D. Kriegman. Personal photo enhancement using example images. *ACM Trans. Graph*, 29(2):1–15, 2010. 14

[46] A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. *CVPR*, 2010. 13

[47] J.Shotton, M.Johnson, and R.Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008. 2

[48] K. Karantzalos and N. Paragios. Large-scale building reconstruction through information fusion and 3-d priors. *IEEE Transactions on Geoscienec and Remote Sensing*, 48(5):2283–2296, 2010. 19

[49] S. Kim, X. Jin, and J. Han. Sparclus: Spatial relationship pattern-based hierarchial clustering. In *SDM*, 2008. 12

[50] K. Koperski and J. Han. Discovery of spatial association rules in geographic information databases. In *SSD*, 1995. 12

[51] T. Korah, S. Medasani, and Y. Owechko. Strip histogram grid for efficient lidar segmentation from urban environments. In *CVPR Workshops (CVPRW)*,

pages 74–81, 2011. 16

[52] M. Kuramochi and G. Karypis. Finding frequent patterns in a large sparse graph. *Data Min. Knowl. Discov.*, 2005. 12, 34

[53] K. N. Kutulakos. Approximate n-view stereo. *ECCV*, 2000. 41

[54] R. F. L. Fei-Fei and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. *CVPR 2004, Workshop on Generative-Model Based Vision*, 2004. 56

[55] F. Lafarge, X. Descombes, J. Zerubia, and M. P. Deseilligny. Structural approach for building reconstruction from a single dsm. ieee trans. *PAMI*, 32(1):135–147, 2010. 19, 73

[56] F. Lafarge and C. Mallet. Bulding large urban environment from unstructed point data. In *International Conference on Computer Vision*, pages 1068–1075, 2011. 6, 19

[57] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. *CVPR*, 2006. 11

[58] Y. J. Lee and K. Grauman. Collect-cut: Segmentation with top-down cues discovered in multi-object images. *CVPR*, 2010. 13

[59] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. *ICCV*, 2009. 1, 15

[60] A. Levinshtein, A. Stere, K. Kutulakos, D. Fleet, S. Dickinson, and K. Siddiqi. TurboPixels: Fast Superpixels Using Geometric Flows. *PAMI*, 2009. 1

[61] F.-F. Li and P. Perona. A bayesian hierarchical model for learning natural scene categories. *CVPR*, 2005. 11

[62] Y. Li, J. Sun, C. Tang, and H. Shum. Lazy snapping. *SIGGRAPH*, 2004. 1, 15

[63] Y. Li, X. Wu, Y. Chrysanthou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics*, 30(4), 2011. 18

[64] Y. Li, Q. Zheng, A. Sharf, D. Cohen-Or, B. Chen, and N. J. Mitra. 2d-3d fusion for layer decomposition of urban facades. In *ICCV*, Barcelona, Spain, 2011. 18

[65] M. Liao, J. Gao, R. Yang, and M. Gong. Video stereolization: Combining motion analysis with user interaction. *Visualization and Computer Graphics, IEEE Transactions on*, 18(7):1079–1088, 2012. 8

[66] H. Lin, J. Gao, Y. Zhou, G. Lu, M. Ye, C. Zhang, L. Liu, and R. Yang. Semantic decomposition and reconstruction of residential scenes from lidar data. *ACM Trans. Graph.*, 32(4):66:1–66:10, July 2013. 9, 84

[67] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing: Label transfer via dense scene alignment. *CVPR*, 2009. 2

[68] F. Liu, M. Gleicher, H. Jin, and A. Agarwala. Content-preserving warps for 3d video stabilization. *ACM Trans. Graph.*, 28(3):44:1–44:9, July 2009. 84, 85, 86

[69] X. Liu, L. Wan, Y. Qu, T.-T. Wong, S. Lin, C.-S. Leung, and P.-A. Heng. Intrinsic colorization. *ACM Transactions on Graphics (SIGGRAPH Asia 2008 issue)*, 27(5):152:1–152:9, December 2008. 14

[70] Y. Livny, S. Pirk, Z. Cheng, F. Yan, O. Deussen, D. Cohen-Or, and B. Chen. Texture-lobes for tree modelling. In *ACM Transactions on Graphics*, volume 30, page 53. ACM, 2011. 88

[71] L.Li and F.Li. What, where and who? classifying events by scene and object recognition. In *ICCV*, 2007. 2

[72] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 1982. 24

[73] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 23

[74] B. Matei, H. Sawhney, S. Samarasekera, J. Kim, and R. Kumar. Building segmentation for densely built urban regions using aerial lidar data. In *CVPR*, 2008. 16

[75] A. Maunz, C. Helma, and S. Kramer. Large-scale graph mining using backbone refinement classes. In *SIGKDD*, 2009. 12, 34

[76] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool. Procedural modeling of buildings. *ACM Trans. Graph.*, 25:614–623, July 2006. 18, 72

[77] D. Munoz, J. Bagnell, N. Vandapel, and M. Hebert. Contextual classification with functional max-margin markov networks. In *CVPR*, 2009. 15

[78] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. van Gool, and W. Purgathofer. A survey of urban reconstruction. In *EUROGRAPHICS 2012 State of the Art Reports*, pages 6, 17, 19

[79] L. Nan, A. Sharf, H. Zhang, D. Cohen-Or, and B. Chen. Smartboxes for interactive urban reconstruction. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2010)*, 29(4):Article 93, 2010. 18, 72

[80] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. *CVPR*, 2006. 11

[81] M. Pauly, N. J. Mitra, J. Wallner, H. Pottmann, and L. J. Guibas. Discovering structural regularity in 3d geometry. *ACM Trans. Graph.*, 27:43:1–43:11, August 2008. 18

[82] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. *CVPR*, 2007. 11

[83] C. Poullis and S. You. Automatic creation of massive virtual cities. In *IEEE Virtual Reality Conference*, pages 199–202, 2009. 6, 19

[84] C. Poullis and S. You. Photorealistic large-scale urban city model reconstruction. *IEEE Transaction on Visualization and Computer Graphics*, 15(4):654–669, 2009. 17

[85] S. Ranu and A. K. Singh. Graphsig: A scalable approach to mining significant subgraphs in large graph databases. In *ICDE*, 2009. 12, 34

[86] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *SIGGRAPH*, 2004. 1, 15, 65, 66, 67

[87] C. Rother, V. Kolmogorov, T. Minka, and A. Blake. Cosegmentation of image pairs by histogram matching-incorporating a global constraint into MRFs. *CVPR*, 2004. 13

[88] B. C. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. *CVPR*, 2006. 13

[89] C. S. S. Lazebnik and J. Ponce. Semi-local affine parts for object recognition. *BMVC*, 2004. 13

[90] C. S. S. Lazebnik and J. Ponce. A maximum entropy framework for part-based texture and object recognition. *ICCV*, 2005. 13

[91] G. H. Q. H. S. Zhang, Q. Tian and S. Li. Descriptive visual words and visual phrases for image applications. *ACM Multimedia*, 2009. 11

[92] F. Schaffalitzky and A. Zisserman. Geometric grouping of repeated elements within images. In *Shape, Contour and Grouping in Computer Vision*, pages 165–181, 1999. 18

[93] T. Shao, W. Xu, K. Zhou, J. Wang, D. Li, and B. Guo. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Trans. Graph.*, 2012. 89

[94] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 2000. 1

[95] S. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3d architectural modeling from unordered photo collections. In *SIGGRAPH Asia*, pages 159:1–159:10, 2008. 17, 84, 87

[96] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. *ICCV*, 2003. 11

[97] J. Sivic and A. Zisserman. Video data mining using configurations of viewpoint invariant regions. 2004. 12

[98] A. W. M. Smeulders, M. Worring, S. Santini, A. Gupta, and R. Jain. Content-based image retrieval at the end of the early years. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(12):1349–1380, Dec. 2000. 1

[99] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. *ACM Trans. Graph.*, 25(3):835–846, 2006. 5, 14, 62, 63

[100] I. Stamos, O. Hadjiliadis, H. Zhang, and T. Flynn. Online algorithms for classification of urban objects in 3d point clouds. In *3DIMPVT*, 2012. 15

[101] R. Szeliski. *Computer Vision: Algorithms and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010. 11

[102] B. L. T. Quack, V. Ferrari and L. V. Gool. Efficient mining of frequent and distinctive feature configurations. 2007. 12, 21

[103] V. F. T. Quack and L. V. Gool. Video mining with frequent itemset configurations. 2006. 12, 21

[104] P. Tan, G. Zeng, J. Wang, S. Bing, and K. L. Quan. Image-based tree modeling. *ACM Trans. Graph*, 26:43, 2007. 88

[105] S. Todorovic and N. Ahuja. Extracting subimages of an unknown category from a set of images. *CVPR*, 2006. 2, 13

[106] L. Van Gool, G. Zeng, F. van den Borre, and P. Muller. Invited paper: Towards mass-produced building models. In *PIA07*, page 209, 2007. 18

[107] C. A. Vanegas, D. G. Aliaga, and B. Benes. Automatic extraction of manhattan-world building masses from 3d laser range scans. *IEEE Transactions on Visualization and Computer Graphics*, 18(10):1627–1637, 2012. 18

[108] C. A. Vanegas, D. G. Aliaga, P. Wonka, P. Mueller, P. Waddell, and B. Watson. Modeling the appearance and behavior of urban spaces. *Computer Graphics Forum*, 29(1):25–42, 2010. Also in Eurographics 2009 STAR (State-of-the-Art Report). 17

[109] S. Vicente, V. Kolmogorov, and C. Rother. Cosegmentation revisited: models and optimization. *ECCV*, 2010. 13

[110] R. G. von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:722–732, 2010. 85

[111] H. Wang, Y. Wexler, E. Ofek, and H. Hoppe. Factoring repeated content within and among images. *SIGGRAPH*, 2008. 53, 104

[112] R. Wang, J. Bach, and F. Ferrie. Window detection from mobile lidar data. In *WACV*, 2011. 16

[113] T. Werner and A. Zisserman. New techniques for automated architectural reconstruction from photographs. In *ECCV*, 2002. 17

[114] D. B. West. *Introduction to Graph Theory*. Prentice Hall, Chapter 3, ISBN 0-13-014400-2, second edition, 1999. 28

[115] O. Whyte, J. Sivic, and A. Zisserman. Get out of my picture! internet-based inpainting. *British Machine Vision Conference*, 2009. 14, 62, 71

[116] J. Winn and N. Jojic. Locus: Learning object classes with unsupervised segmentation. *ICCV*, 2005. 2, 13

[117] C. Wu, J.-M. Frahm, and M. Pollefeys. Detecting large repetitive structures with salient boundaries. In *ECCV*, 2010. 18

[118] J. Xiao, T. Fang, P. Tan, P. Z. E. Ofek, and L. Quan. Image-based façade modeling. *ACM Trans. Graph.*, 27(5):1–10, 2008. 17

[119] J. Xiao, T. Fang, P. Zhao, M. Lhuillier, and L. Quan. Image-based street-side city modeling. *ACM Transactions on Grpahics*, 28(5):10–19, 2009. 6, 18

[120] J. Xiao and Y. Furukawa. Reconstructing the world's museums. In *ECCV*, 2012. 18

[121] J. Xiao and L. Quan. Multiple view semantic segmentation for street view images. In *ICCV*, 2009. 2, 17, 74

[122] X. Yan and J. Han. gspan: Graph-based substructure pattern mining. In *ICDM*, 2002. 12

[123] J. Yuan and Y. Wu. Spatial random partition for common visual pattern discovery. In *ICCV*, 2007. 51

[124] J. Yuan, Y. Wu, and M. Yang. From frequent itemsets to semantically meaningful visual patterns. In *SIGKDD*, 2007. 4, 12, 21, 57

[125] C. Zhang, J. Gao, O. Wang, P. Georgel, R. Yang, J. Davis, J. Frahm, and M. Pollefeys. Personal photo enhancement using internet photo collections. *Visualization and Computer Graphics, IEEE Transactions on*, PP(99):1–1, 2013. 8, 62, 71

[126] C. Zhang, L. Wang, and R. Yang. Semantic segmentation of urban scenes using dense depth maps. In *ECCV*, 2010. 2, 17, 74, 76, 92

[127] H. Zhao, Y. Liu, X. Zhu, Y. Zhao, and H. Zha. Scene understanding in a large dynamic environment through a laser-based sensing. In *ICRA*, 2010. 15, 16

[128] P. Zhao, T. Fang, J. Xiao, H. Zhang, Q. Zhao, and L. Quan. Rectilinear parsing of architecture in urban environment. In *CVPR*, 2010. 17, 96, 106

[129] Q. Zhou and U. Neumann. 2.5d building modeling with topology control. In *CVPR*, 2011. 19

[130] Q.-Y. Zhou and U. Neumann. 2.5d dual contouring: a robust approach to creating building models from aerial lidar point clouds. In *ECCV*, 2010. 19

[131] Q.-Y. Zhou and U. Neumann. 2.5d building modeling by discovering global regularities. In *CVPR*, 2012. 6

[132] X. Zhu, H. Zhao, Y. Liu, Y. Zhao, and H. Zha. Segmentation and classification of range image from an intelligent vehicle in urban environment. In *IROS*, 2010. 16

Vita

NAME

Jizhou Gao

EDUCATION

- June 2006: B.S. in Computer Science, Zhejiang University, China

PUBLICATIONS

Journal Articles

- Hui Lin*, **Jizhou Gao*** (*joint first authors), Yu Zhou, Guiliang Lu, Mao Ye, Chenxi Zhang, Ligang Liu, and Ruigang Yang, Semantic Decomposition and Reconstruction of Residential Scenes from LiDAR Data, *ACM Transaction on Graphics (TOG) - Proceedings of ACM SIGGRAPH*, Volume 32, Issue 4, 2013.

- Chenxi Zhang, **Jizhou Gao**, Oliver Wang, Pierre Georgel, Ruigang Yang, James Davis, Jan-Michael Frahm and Marc Pollefeys, Personal Photo Enhancement using Internet Photo Collections, *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, to appear, 2013.

- Miao Liao, **Jizhou Gao**, Minglun Gong and Ruigang Yang, Video Stereolization: Combining Motion Analysis with User Interaction, *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, Volume 18, Issue 7, 2012.

- Rebekah Cosden, Christian Lattermann, Spencer Romine, **Jizhou Gao**, S. Randal Voss and James N. Macleod, Intrinsic Repair of Full-thickness Articu-

lar Cartilage Defects in the Axolotl Salamander, *Osteoarthritis and Cartilage*, Volume 19, Issue 2, 2011.

- Jiejie Zhu, Liang Wang, **Jizhou Gao** and Ruigang Yang, Spatial-Temporal Fusion for High Accuracy Depth Maps using Dynamic MRFs, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, Volume 32, Issue 5, 2010.

Referred Conference Proceedings

- **Jizhou Gao** and Ruigang Yang, Online Building Segmentation from Ground-based LiDAR Data in Urban Scenes, In *Proc. of the third Joint 3DIM/3DPVT (3DV)*, 2013.

- **Jizhou Gao**, Yin Hu, Jinze Liu and Ruigang Yang, Unsupervised Learning of High-order Structural Semantics from Images, In *Proc. of the IEEE 12th International Conference on Computer Vision (ICCV)*, 2009.

- Xinyu Huang, **Jizhou Gao**, Sen-ching Cheung and Ruigang Yang, Manifold Estimation in View-based Features Space for Face Synthesis across Poses, In *Proc. of the 9th Asian Conference on Computer Vision (ACCV)*, 2009.

- Xianwang Wang, Xinyu Huang, **Jizhou Gao** and Ruigang Yang, Illumination and Person-Insensitive Head Pose Estimation using Distance Metric Learning, In *Proc. of the 10th European Conference on Computer Vision (ECCV)*, 2008.

- Xinyu Huang, Xianwang Wang, **Jizhou Gao** and Ruigang Yang, Estimating Pose and Illumination Direction for Frontal Face Synthesis, In *Proc. of the Biometric Workshop of CVPR*, 2008.

- Xinyu Huang, **Jizhou Gao** and Ruigang Yang, Calibrating Pan-Tilt Cameras with Telephoto Lenses, In *Proc. of the 8th Asian Conference on Computer Vision (ACCV)*, 2007.

- Xinyu Huang, **Jizhou Gao**, Liang Wang, and Ruigang Yang, Examplar-based Shape from Shading, In *Proc. of the 6th International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 2007.

PATENTS

- **Jizhou Gao**, Yu Huang and Heather Yu, Method and System for Video Summarization, US Patent 13/297,142