PUTTING THE USER IN THE LOOP FOR IMAGE-BASED MODELING

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Adarsh Prakash Murthy Kowdle

August 2013

PUTTING THE USER IN THE LOOP FOR IMAGE-BASED MODELING

Adarsh Prakash Murthy Kowdle, Ph.D.

Cornell University 2013

Image-based modeling, the task of recovering the 3D structure of an object or a scene using 2D images is one of the primary goals of computer vision. However, 3D reconstruction in practice is a hard task due to the numerous scene irregularities. When a scene is captured from multiple viewpoints, irregularities such as textureless surfaces, specularities, thin structures, etc., make the reconstruction inaccurate. In addition, image sequences or videos of a scene captured in the wild often consist of dynamic or moving objects such as people, which makes the task of image-based modeling extremely hard. In this thesis, our goal is to tackle these problems to obtain a more accurate reconstruction of the scene. In particular, using the intuition that humans easily discern the 3D structure behind the photons, this thesis addresses these problems by putting the user in the loop with the image-based-modeling algorithm.

In the first part of the thesis, we focus on image-based modeling of *static scenes*. We explore putting the user in the loop interacting with the algorithm by providing constraints via simple interactions on the image data, to help overcome the ill-effects of the scene irregularities. We introduce two algorithms within this framework. First, an algorithm where the user drives the process of image-based modeling. Second, a novel active-learning algorithm, which initiates the process of image-based modeling via an unsupervised algorithm and then guides the user to provide constraints when and where needed.

In the second part of the thesis, we focus on image-based modeling of *dy-*

namic scenes captured by either a dynamic camera or a static camera. We develop algorithms that leverage the dynamic objects in the scene to aid the image-based modeling algorithm. We propose novel algorithms that use the sparse, yet strong occlusion cues between the moving object and the scene, along with a realistic motion prior for the moving object to recover the depth of the scene. We explore putting the user in the loop within the framework, guided by the algorithm to provide useful depth-order constraints and show that we further improve the solution of the unsupervised algorithm.

Finally, we present an end user application: iModel, which puts the user in the loop for object of interest 3D modeling on a mobile device and 3D printing of an object of interest.

THESIS COMMITTEE

Prof. Tsuhan Chen
School of Electrical and Computer Engineering,
Cornell University

Prof. Anthony P. Reeves
School of Electrical and Computer Engineering,
Cornell University

Prof. Noah Snavely
Department of Computer Science,
Cornell University

Prof. Ashutosh Saxena
Department of Computer Science,
Cornell University

Dr. Richard Szeliski
Microsoft Research, Redmond

I dedicate this thesis to my parents:

Your support is why I am here.

ACKNOWLEDGEMENTS

I thank my advisor, Prof. Tsuhan Chen, for his support and guidance through every step of my PhD. Deciding to move with you from Carnegie Mellon to Cornell was the best decision I have ever made. The freedom you gave me with my research, the opportunities you gave me, the guidance and push you gave me to think differently is what has shaped me. I have always admired your demeanor and how you approach every problem, breaking it down to the most simplest of problems. I hope your attitude rubs off on me. I will always cherish the time I have spent working with you.

I thank my thesis committee: Prof. Anthony Reeves (thank you for your input on this thesis. I have always loved your sense of humor and I am happy I was part of your lectures), Prof. Noah Snavely (you are probably the most polite person I have ever met. I had a wonderful time as your teaching assistant, helping you organize the seminars and collaborating with you), Prof. Ashutosh Saxena (it was a delight collaborating with you. You have a passion for research that I will always admire. Thank you for your guidance in shaping this thesis), and Dr. Richard Szeliski (thank you for your guidance. Working with you as an intern was a wonderful experience and an honor).

I thank Dhruv Batra and Devi Parikh, my collaborators, my mentors and very close friends all through my PhD. Your advice and help, every step of the way has helped me immensely. I will always remember those late nights we were working, the coffee pyramids, the post paper submission celebrations at Nines, the sky-dive and all those crazy things we have done together.

I thank my collaborators, Andrew Gallagher, Yao-Jen Chang, Congcong Li and Sudipta Sinha. It was a pleasure working with you, I have learnt a lot from each of you.

I thank all the Advanced Multimedia Processing lab members: Amandy Nwana, Amir Sadovnik, Henry Shu, Kuan-chuan Peng, Ruogu Fang, Yimeng Zhang and Zhaoyin Jia. I also thank the visitors, the undergraduate students and MEng students I had the pleasure of collaborating and mentoring over the years. The stimulating discussions at the group meetings, the post deadline celebrations and all the fun times we enjoyed together are memories I will always cherish. I am grateful to the stimulating environment at Cornell, the seminars and reading groups, which have helped me explore new ideas and continue learning.

I thank my room-mate Tanay Gosavi, and all my wonderful friends at Cornell. You guys have made my life at Cornell truly memorable.

I thank my parents, my brother Ashwin and my sister-in-law Sarayu for helping me make all my decisions and being there every step of the way. Lastly, but in no way the least, a special thank you to my wife Gauri. The last year of my graduate life has been special and memorable because of you.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

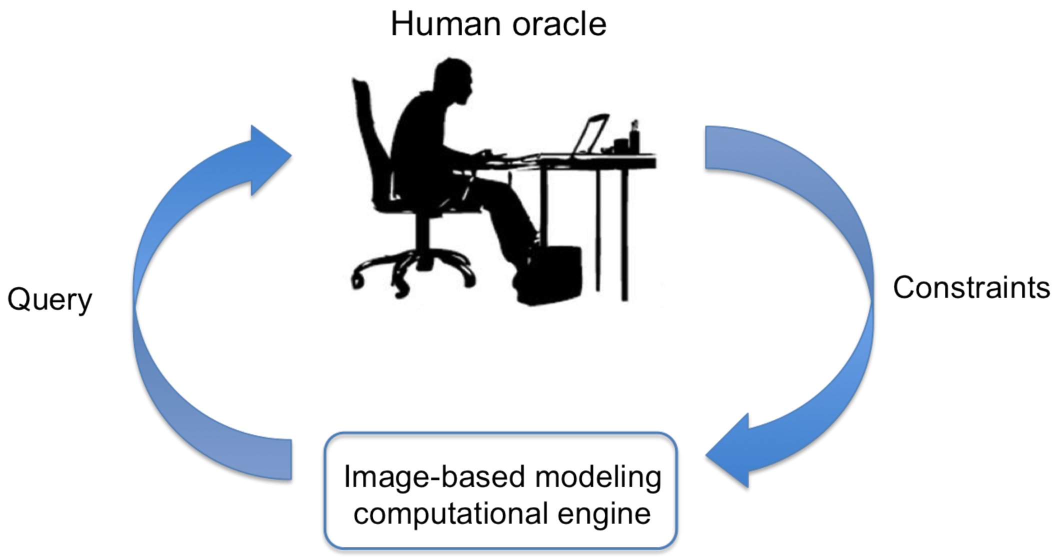


Figure 1.1: Putting the user in the loop for image-based modeling.

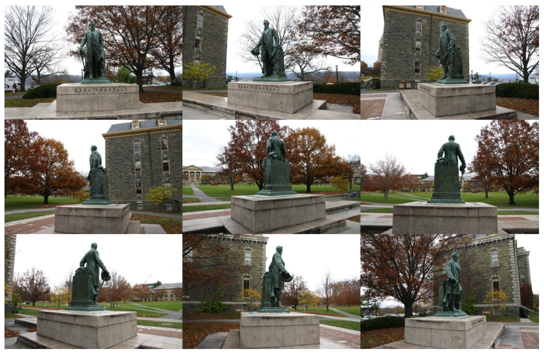
Image-based modeling, the task of recovering the 3D structure of an object or a scene using 2D images is one of the primary goals of computer vision. Estimating the depth of the scene has been explored for a variety of applications ranging from image-based rendering, image editing, to scene understanding. There has been significant progress in image-based modeling using input images that vary from the very constrained setup of a single image [40, 80], a single stereo-pair [82, 84], to multiple images of the scene captured from different viewpoints or a video of the scene [26–28, 30, 70, 75, 76, 85, 88, 100]. However, 3D reconstruction of natural scenes is a hard task, due to the numerous scene irregularities such as textureless surfaces, specularities, thin structures, background clutter, etc., resulting in inaccurate reconstructions. In addition, unconstrained image sequences often consist of dynamic or moving objects such as people walking about in the scene, which makes the task of image-based modeling extremely hard. In this thesis, we address these issues by putting the user in the loop for image-based modeling.

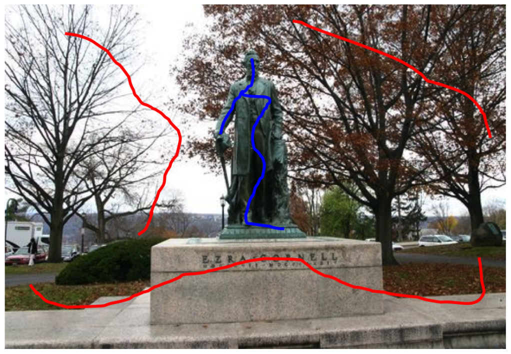
The core of this thesis is the intuition that when humans look at a scene they much better discern the geometric structure behind the photons. However, the key to put the user into the loop to aid the image-based modeling algorithm is to design a smart computational engine and keep the user interaction simple and intuitive, yet powerful. We illustrate the basic idea of putting the user in the loop using Figure 1.1. The computational engine is the workhorse that uses the constraints provided by the user (i.e., oracle) and recovers the 3D structure of the scene. The computational engine uses the current solution to the problem queries the user for useful constraints, placing the user in a closed loop with the computational engine. The questions to address within this framework is what is the computational engine? What are the constraints the user provides and how is this incorporated by the computational engine?

In contrast to prior interactive works that require very involved user inputs such as feature correspondence across images and line drawings of objects [2, 19, 20, 38, 86, 87, 90], we use simple scribbles in all our algorithms as user constraints, as illustrated in Figure 1.2(c), Figure 1.3(d) and Figure 1.5. In addition, we show in this thesis that the user in the loop does not directly solve the problem of image-based modeling. The user is guided by the computational engine to provide simple and intuitive constraints, which are incorporated into the computational engine to indirectly aid the task of image-based modeling.

In the first part of this thesis, we consider image-based modeling of *static* *scenes* captured from multiple viewpoints. We put the user in the loop to providing constraints to the algorithm via simple scribbles to indicate the object of interest or provide 3D support constraints. We also introduce a novel active-learning algorithm for image-based modeling that can intelligently guide the user constraints to when and where needed. In the second part of the thesis, we consider scenes with dynamic objects in them, we propose novel unsupervised algorithms that serve as the computational engine that leverages the depth or occlusion constraints revealed by the interactions between the moving object and the real 3D scene to implicitly aid modeling the scene. We put the user in the loop guided by the computational engine to provide simple relative depth cues to further improve the solution to the problem. We give an overview of the two parts of this thesis below, followed by the structure of this thesis.

First, consider a scenario where we wish to obtain a volumetric 3D model of the Ezra Cornell Statue shown in Figure 1.2(a). One approach to achieve this, is to haul an expensive laser scanner to get precise depth estimates in a controlled setup, and reconstruct the statue [67]. However, this might be not be a feasible solution for average users. Another typical approach is to capture images of the object in a controlled environment like a multi-camera studio with mono-color screen and structured lighting, and use a shape-from-silhouette algorithm [15, 21, 23, 91] to render the 3D model. While these techniques produce promising results in these constrained settings, this is a tedious process, and in some cases such as the statue (or other immovable objects such as historically or culturally-significant artifacts) this is not an option. In a world where we are surrounded by portable cameras, a more accessible approach is to capture images of the object in its natural environment and directly estimate the 3D structure from these natural images. The images captured in the wild typically have


(a)


(b)

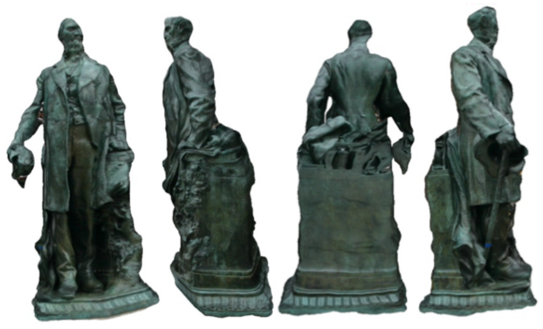
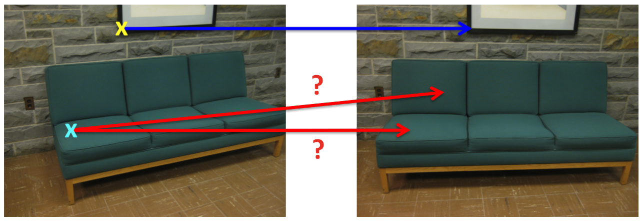

(c)

Figure 1.2: (a) Images of a static scene captured from multiple viewpoints. What is the user's object of interest? The statue, or perhaps the pedestal the statue is on? (b) The user in the loop guides the algorithm by indicating the object of interest via simple scribbles. (b) Output obtained using the proposed interactive object of interest 3D modeling algorithm.

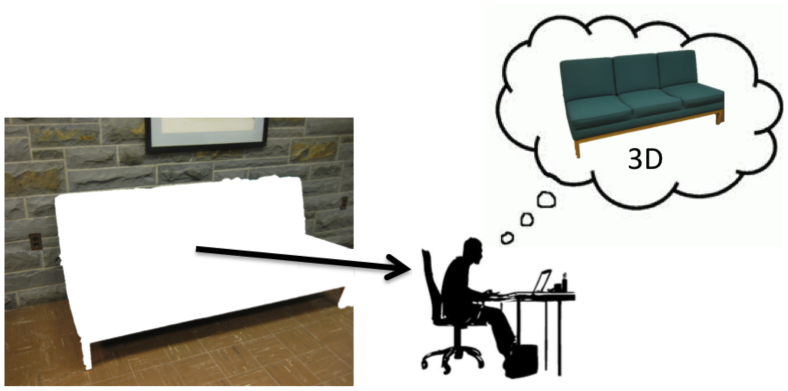
cluttered backgrounds making it hard for unsupervised algorithms to obtain a good reconstruction. In addition the notion of *object of interest* is also ill-posed. In Figure 1.2(a) how would the algorithm know if the object of interest is the statue, or perhaps the pedestal the statue is on? This is the first scenario we explore in Chapter 2. We put the user in the loop with the algorithm to indicate the object of interest using simple and intuitive scribbles shown in Figure 1.2(b), resulting in the model in Figure 1.2(c).

The scenario discussed above requires the user to initiate the process of

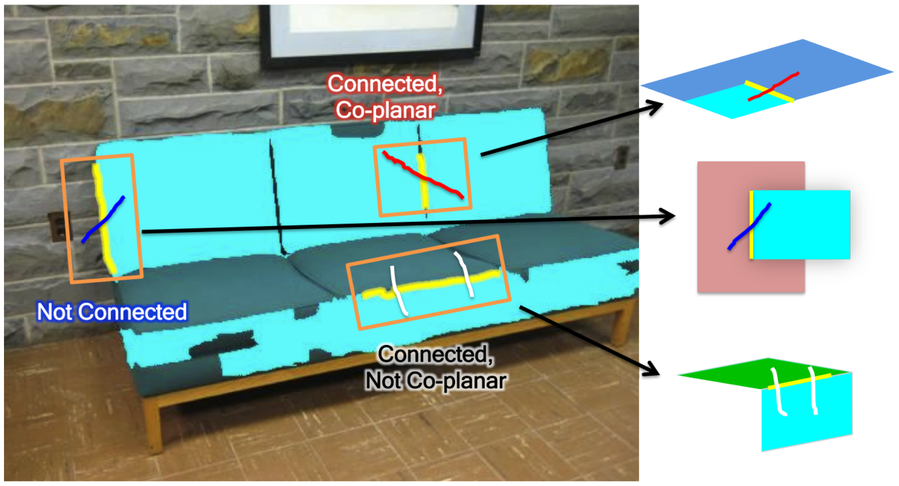
(a)



(b)



(c)



(d)

Figure 1.3: (a) Images of a static scene captured from multiple viewpoints. (b) Finding a unique match for some features such as the yellow '×' on the wall is well structured but, the matching point for the cyan '×' on the textureless surface of the couch is ambiguous. (c) When humans look at a scene they much better discern the geometric structure behind the photons, which can help reconstruct ambiguous portions of the scene. (d) The algorithm guides the user input by indicating that the cyan region is the region it needs some support constraints. The user provides constraints across the yellow edges via the red, blue and white scribbles.

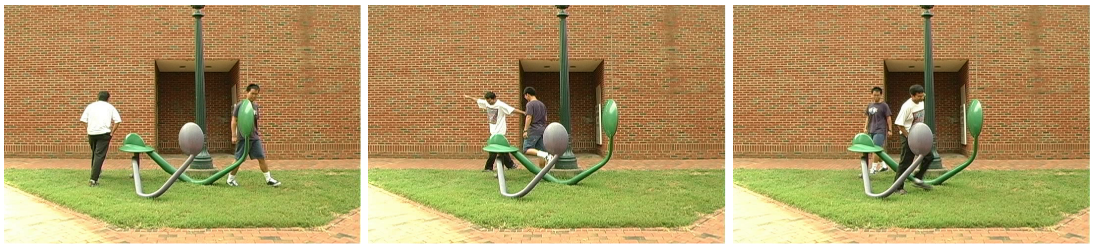


Figure 1.4: Dynamic scene captured by static camera. Note that the moving objects reveal sparse yet power pairwise depth ordering cues. (Data from [32])

image-based modeling. Is there a way where we can reduce the effort of the user? Is there a way where we can exploit the significant success with automatic image-based modeling algorithms that use the rich cues present in multi-view images [26–28, 30, 70, 75, 76, 85, 88, 100]? Our first observation is that in case of images of natural scenes automatic algorithms fail to produce an accurate reconstruction in case of scene irregularities such as textureless and specular surfaces. Consider that we wish to model the scene shown in Figure 1.3(a). When we observe two images of the scene from different viewpoints, finding a match for a unique match for features such as the yellow '×' on the wall is well structured however, the matching point for the feature on the textureless surface of the couch shown in the cyan '×' is ambiguous. Thus, while the unsupervised algorithm can reconstruct the wall and perhaps even the ground surface, it makes errors in modeling the couch. On the other hand, when humans look at a scene they much better discern the geometric structure behind the photons (Figure 1.3(c)). This intuition forms the basis of the second algorithm we describe in in Chapter 2, where we allow the unsupervised image-based modeling algorithm to reconstruct the scene, automatically identify uncertain regions (such as the couch) and guide the user to provide support constraints via simple scribbles as shown in Figure 1.3(d) to improve the 3D reconstruction.

While the first part of the thesis focuses on image-based modeling of static scenes captured from multiple viewpoints, there has been an increasing trend in the use of image sequences or videos captured in natural scenes using handheld devices, surveillance cameras, or even professionally captured movie footage, for a number of applications including recovering depth of the scene, video summarization, activity recognition, etc. These unconstrained image sequences could be captured by static or dynamic cameras and the scene could also contain dynamic objects, which makes the problem of recovering depth a non-trivial task. We note that most of the prior works including algorithms we discuss in the first part of the thesis focus on estimating the depth of a static scene captured by a dynamic camera. How do we handle scenarios where a dynamic scene is captured by a dynamic camera or a static camera? Our intuition for the novel computational engine is that the dynamic content moving about the scene implicitly provide useful depth cues to the algorithm. For example, consider Figure 1.4 where a static camera captures two people walking around in the scene. Even with the sampling of images shown here, we can easily observe useful occlusion cues that provide depth ordering information about the different regions of the scene. In Chapter 3, we consider image-based modeling of dynamic scenes. In case of a dynamic scene captured by a dynamic camera, we use the occlusions between the moving object and the scene, and a realistic motion prior for the moving object to recover the depth of the dynamic scene. We then address the scenario of a dynamic scene captured by a static camera, and show that even in the absence of camera motion, we can use the sparse, yet strong occlusion cues revealed by moving object interacting with the static scene to aid the image-based modeling, and decompose the scene into fronto-parallel depth layers. We then put the user into the loop within this framework

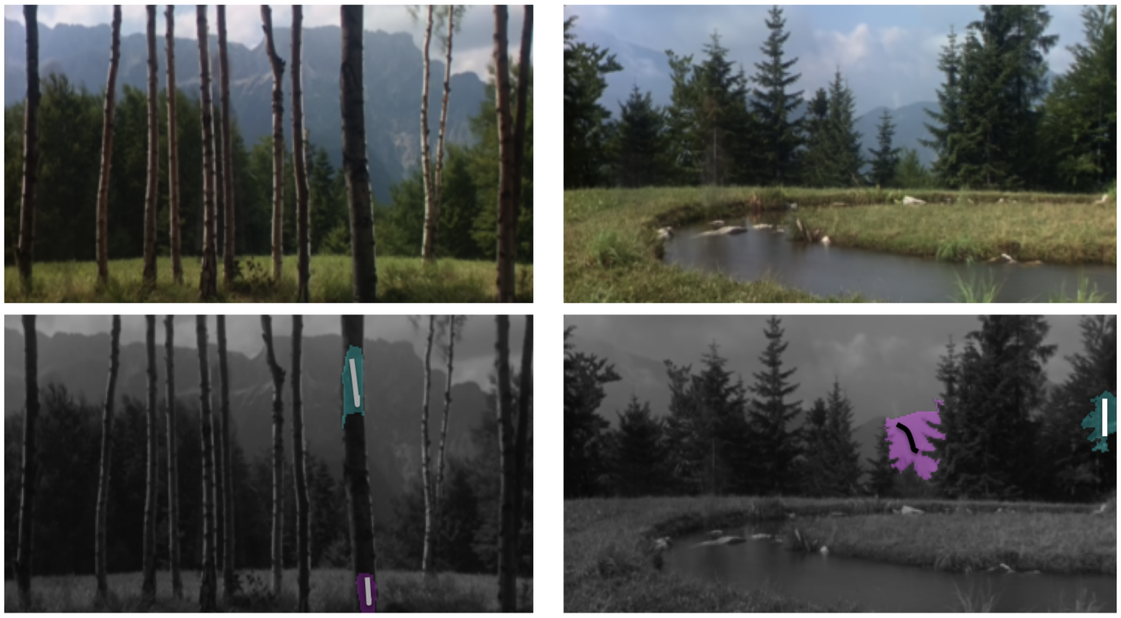


Figure 1.5: Simple pairwise constraints provided by the user to aid the process of image-based modeling of dynamic scenes.

by leveraging the computational engine to guide the user to provide useful pairwise depth ordering constraints, as shown in Figure 1.5 to further improve the solution of the algorithm.

The rest of this thesis is organized as follows. Chapter 2 presents algorithms to put the user in the loop for image-based modeling of static scenes. Chapter 3 presents algorithms that address the scenario of image-based modeling of dynamic scenes, captured by a dynamic camera and a static camera. Chapter 4 then briefly describes an end-user application developed to aid object of interest 3D modeling on a mobile device. The thesis is concluded in Chapter 5 with a discussion of potential future work.

1.1 First Published Appearances of Described Contributions

Most contributions or their initial versions, described in this thesis have first appeared as various publications:

- Chapter 2: Kowdle, Batra, Chen, Chen [53]; Kowdle, Chang, Batra, Chen [55]; Kowdle, Chang, Gallagher, Chen [56]
- Chapter 3: Kowdle, Snavely, Chen [63]; Kowdle, Gallagher, Chen [59]
- Chapter 4: Kowdle, Batra, Chen, Chen [53]; Kowdle, Liu, Hsu, Lew, Puri, Batra, Chen [61]

The following contributions have appeared as various publications: Kowdle, Chen [57], Kowdle, Sinha, Szeliski [62]; Kowdle, Gallagher, Chen [58]; Li, Kowdle, Saxena, Chen [60, 68, 69]; Kowdle, Chang, Chen [54]; Batra, Kowdle, Parikh, Luo, Chen [3–7]. However, they are beyond the scope of this dissertation, and therefore are not discussed here.

CHAPTER 2

IMAGE-BASED MODELING OF STATIC SCENES

2.1 Introduction

Image-based modeling, the recovery of 3D structure of a scene using 2D images, is an active research topic in the computer vision community. There has been significant success with automatic algorithms [26–28, 30, 70, 75, 76, 85, 88]. However, when only a few images are available, these automatic algorithms fail to produce a dense reconstruction, leaving holes in case of scene irregularities such as textureless and specular surfaces. While planar approximations to the scene [25, 28, 70, 85] help obtain more visually pleasing reconstructions, the cues in a number of scenes are not sufficient to hypothesize a good model. In particular, textureless and specular surfaces and a lack of geometric cues such as lines hinders their performance.

A number of works formulate the task of image-based modeling as a discrete labeling problem. For example, shape from silhouette algorithms formulate a binary labeling problem to separate the object from the background [8, 12, 66], a number of stereo matching algorithms treat the pixel disparities as discrete labels and formulate a labeling problem to recover the depth of the scene [82], piecewise planar stereo works discretize the scene into a finite number of 3D planes and treat these planes as label set [25, 28, 70, 85]. Motivated by these works, we consider the task of putting the user into the loop. A common framework of all these works is the construction of a Markov Random Field (MRF)

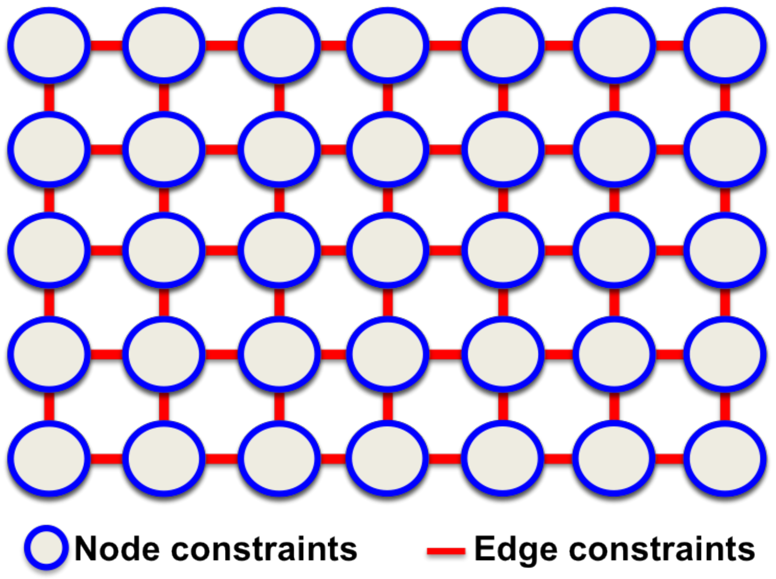


Figure 2.1: Discrete labeling problem: The user constraints are provided over the nodes and edges of the graph to help modulate the unary term and the pairwise term, respectively.

over the image illustrated in Figure 2.1.

We propose algorithms that use superpixels extracted from the image as the labeling sites[1] i.e., nodes in the MRF, with edges to all adjacent superpixels. User-provided input is incorporated into the node and edge terms of the model as constraints. First, we propose an algorithm where the user provides constraints on the nodes of the graph (Section 2.4). In this algorithm, the user initializes the loop by providing annotations on the nodes. These node constraints allow the user to define the label space for the problem and modulate the unary term in the energy function. In the second algorithm, an automatic piecewise planar reconstruction algorithm first tries to reconstruct the scene initiating the loop. Given the reconstruction, we propose an active-learning algorithm that uses intuitive cues to quantify the uncertainty of the algorithm. The algorithm

[1]Superpixels are used to help reduce computational complexity

then queries the user to provide support for the uncertain regions via edge constraints on the pairwise term that lead to better reconstructions (Section 2.5). We will discuss the details of the algorithm and applications in the following sections.

Organization. The rest of this chapter is organized as follows: Section 3.2 discusses related work; Section 2.3 describes the preprocessing performed on the images in the proposed algorithms; Section 2.4 presents our approach where the user provides node constraints to initiate the loop and reconstruct the object of interest; Section 2.5 describes our approach of active learning where the automatic computational engine initiates the loop and guides the user towards where it needs help; Section 3.3.2 describes the experiments and results; Finally, Section 3.6 summarizes the chapter.

2.2 Related Work

Automatic algorithms. 3D reconstruction from multiple images is an active research topic in the computer vision community. While some 3D reconstruction works [75, 76] are geared towards video, some [30, 88] are geared towards unordered photo collections on the internet. Most require a large photo collection. When the number of input images is small, the automatic algorithms fail to produce a dense reconstruction. A survey of multiview stereo methods has been provided by [84]. With a small set of images the reconstruction is incomplete, leaving holes on textureless and specular surfaces. Planar approximations to the scene [25, 28, 70, 85] help obtain more visually pleasing reconstructions. However, these algorithms use image features such as strong edges and lines, which may be absent in textureless surfaces, motivating interactive algorithms.

Interactive algorithms: user driven. A typical approach to obtain the 3D model of a non-planar object is to capture images of the object in a controlled environment like a multi-camera studio with mono-color screen where background subtraction is a well structured problem, and use a shape-from-silhouette algorithm [15, 21, 23, 91] to render the 3D model. Although these techniques have produced promising results in these constrained settings, this is a tedious process, and in some cases not an option (for example, immovable objects like a statue, historically or culturally significant artifacts). However, a more realistic approach is to capture images of the object in its natural environment and directly estimation the 3D structure from these natural images. The images captured in this case would typically have cluttered backgrounds, which is known to be problematic for background subtraction algorithms. There have been many interactive 3D reconstruction algorithms that uses a piecewise planar representation of the scene [2, 19, 20, 38, 86, 87, 90]. The user interactions required range from providing feature correspondence, to providing plane boundaries and line models of the scene. [20] proposed an algorithm to reconstruct man-made architectures by marking the edges in the structure and by exploiting symmetry in man-made structures. [38] and [86] require the user to provide a detailed line model of the object or mark all the 2D plane polygons in the scene, respectively; and reconstruct the scene by incorporating geometric information from structure-from-motion. [89] used scribbles as input to help improve the 3D reconstruction obtained from a single image. In the first algorithm proposed in this paper, we leverage the user input to provide node constraints in the MRF formulation. We propose interactive algorithms driven by the user via simple scribbles that are used to reconstruct non-planar objects, planar scenes, and even render non-planar objects as part of a planar scene.

Interactive algorithms: active-learning. Active learning is a well established subfield of machine learning, which has been shown to benefit a number of computer vision applications such as object categorization [48], image retrieval [31, 101], video classification [97], dataset annotation [16], and interactive co-segmentation [7]; maximizing the knowledge gain while valuing the user effort [93]. However, such an algorithm has not been proposed for image based modeling. [7] proposed an approach for interactive co-segmentation where, starting from the user interactions (scribbles) to identify the object of interest, the algorithm exploits a number of cues using the scribbles, and identifies informative regions to request the user for more interactions. Interactive 3D reconstruction, however, is not a trivial extension of this binary problem to multi-class segmentation. Rich information is already embedded in multiple images of a scene, which an automatic algorithm can fully utilize. However, the automatic algorithms fall short where texture or geometry cues cannot be easily identified from the images. Therefore, we formulate interactive 3D reconstruction as an error-correction and learning problem, where active-learning identifies uncertain regions, requests the user to provide geometric cues, and adapts the algorithm for the specific scene based on the user inputs.

2.3 Pre-processing

In this paper we work with multiple images of a scene captured from different viewpoints. We perform the following pre-processing steps. We first run structure from motion (SfM) using the algorithm by [88] on the multiview images to recover the camera projection matrices for all the views, a sparse 3D point cloud and the set of the points visible by each camera. We construct a graph, $G = (V, E)$, over the superpixels[2], with edges between adjacent superpixels to formulate our discrete labeling problem. The graph is a planar graph, but not typically a grid graph we use for illustration in Figure 2.1. We have developed a Java-based user interface, which we have made publicly available [92]. This interface is used in all our interactive algorithms.

In the following sections, we will first describe the algorithm that is initiated by the user via node constraints to obtain the final 3D reconstruction via scene co-segmentation. We will then describe the algorithm that is driven by the computational engine and guides the user constraints via an active-learning algorithm.

2.4 Node constraints

In our first approach, the user provides annotation on the nodes of the graph as shown in Figure 2.2 and guides the process by initiating the algorithm. The algorithm first displays the image collection to the user. The user selects an image and provides scribbles on the image with different colors indicating the label space for the segmentation algorithm. Given these scribbles on the nodes of the graph we define an energy function over the graph G as described below.

[2]We use mean-shift segmentation [18] to break an image to about thousand superpixels.

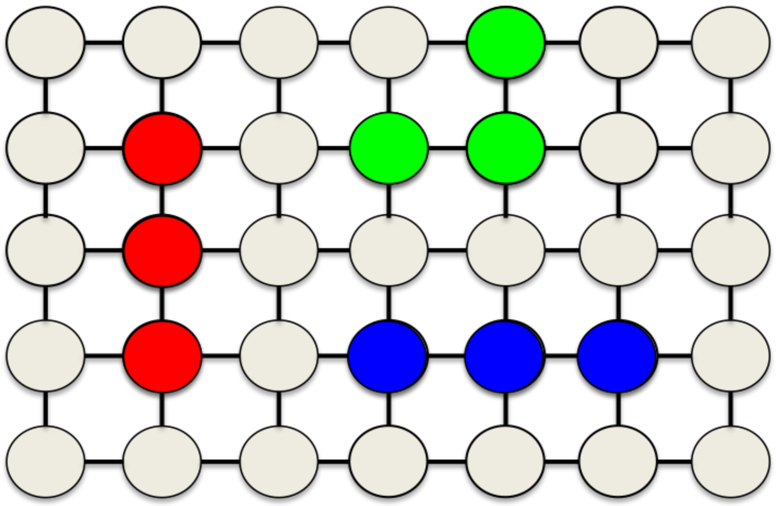


Figure 2.2: Node constraints. The colored nodes in the graph represent the node constraints provided by the user via multiple colored scribbles to indicate the different labels. More description about the node constraints and incorporating them into the algorithm is in Section 2.4.

2.4.1 Energy minimization

Let the set of images be $\mathcal{X}$. Consider an image-scribble pair $D = \{X, S\}$, where the image X chosen by the user is represented as a collection of n nodes (superpixels) to be labeled, $X = \{X_1, X_2, \ldots, X_n\}$. The user provides a set of scribbles S on the image with multiple labels (suppose that the user defines L labels in the scene), which is represented as the partial set of labels for these nodes $S = \{S_1, S_2, \ldots, S_n\}$ where, $S_i = \{0, 1, \ldots, L-1\}$. Using these labeled nodes (node constraints), we learn an appearance model A described below. We define an energy function over the image as:

$$E(X : A) = \sum_{i \in V} E_i(X_i : A) + \lambda \sum_{(i,j) \in E} E_{ij}\left(X_i, X_j\right), \tag{2.1}$$

where the first term (unary term) indicates the cost of assigning a node to one of the labels, while the second term (pairwise term) is used for penalizing label disagreement between neighbors. The colon (:) in the equation indicates that the term is dependent on the learnt appearance model.

Unary Term. The unary term is modeled via the node constraints provided by the user. Given the node constraints we learn the appearance model, which consists of a Gaussian Mixture Model for each of the L labels defined by the user i.e, $A = \{\text{GMM}_0, \ldots, \text{GMM}_{L-1}\}$. Specifically, we use color features (Lab space) extracted from superpixels on the labeled nodes and fit GMMs for the corresponding classes. We use MDL to estimate the right number of components to use to describe the data (allowing a maximum of 10 Gaussian components). The unary term for all nodes are then defined as the negative log-likelihood of the features given the class model. We set the unary term of the superpixels labeled by the user to $-\infty$ (a large negative value) as hard constraints in the energy minimization.

Pairwise Term. We use the commonly used Potts model to model the pairwise term,

$$E_{ij}(X_i, X_j) = \mathrm{I}\,(X_i \neq X_j)\;\exp(-\beta d_{ij}), \tag{2.2}$$

where $\mathrm{I}\,(\cdot)$ is an indicator function that is 1(0) if the input argument is true(false), d_{ij} is the distance between features at superpixels i and j and β is a scale parameter. Intuitively, this smoothness term tries to penalize label discontinuities among neighboring sites but modulates the penalty via a contrast-sensitive term. Thus, if two adjacent superpixels are far apart in the feature space, there would be a smaller cost for assigning them different labels than if they were close. We use features such as color, texture, and shape features (for more details about features refer [42]).

Finally, we use graph-cuts (with α-expansion) to compute the MAP labels for all superpixels [1, 9, 10, 51]. The parameters λ and β were empirically chosen and fixed for all scenes. This was found to work well in practice. Given, the above formulation of the energy minimization problem we discuss below how these constraints allow the user to reconstruct non-planar objects, planar scenes and even render non-planar objects as part of the planar scene.

2.4.2 Reconstructing non-planar objects

Consider reconstructing a non-planar object of interest such as a statue as shown in Figure 2.3(a). A popular approach to obtain the 3D model is shape from silhouette i.e., obtain the silhouette of the object from multiple viewpoints and infer the volume of intersection (visual hull). We wish to avoid the tedious (sometime impossible) process of taking the object of interest into a controlled setup such as an studio with chroma-keying setup. We instead capture images of the object of interest from multiple views and get the user into the loop to obtain the silhouettes.

The user provides node constraints by providing scribbles of two colors (two labels) on one image to indicate the object of interest and the background as shown in Figure 2.3(b). The task is setup setup as a binary labeling problem as described above in Section 2.4.1 with $L = 2$. While the user may chose only one image to provide the node constraints, we use the idea that the images are tied together through the shared appearance models (A) learnt using the user inputs. The shared appearance models help formulate the energy function described in Eqn 2.1 for each image. Using graph-cuts we obtain the cosegmentation of the object in each view as shown in Figure 2.3(c). More details about the two class

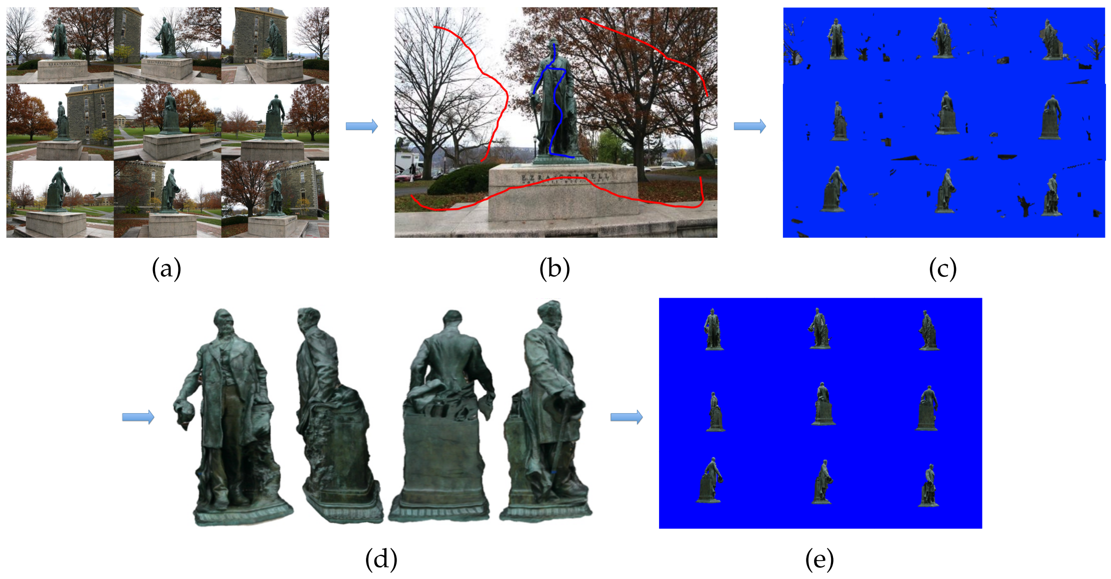


Figure 2.3: Object of interest 3D modeling. (a) Input multiview images of the object of interest (statue); (b) User node constraints that are used as hard constraints to learn the appearance models a perform the co-segmentation; (c) Resulting co-segmentation that has some inaccurate labeling in the background; (d) Shape from silhouette reconstructs the 3D model by the finding the volume of intersection given the camera parameters; (e) Projecting the 3D model back into each image allows fixing the segmentation errors that existed earlier.

co-segmentation formulation and an extension to intelligently guide the user input is available in [7].

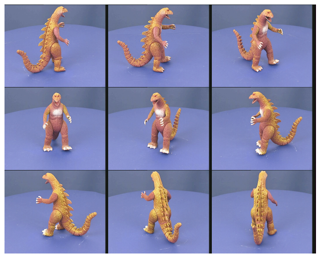
Note that the segmentation is noisy with small regions in the background that share similar appearance to the object of interest incorrectly labeled. While one approach to fix this is to modulate the smoothness parameter (β), this is a sensitive parameter to tune since it can result in regions of the object of interest being incorrectly labeled as background. We instead use the 3D geometry to help fix errors. Using the camera parameters recovered in the pre-processing stage (Section 2.3) and the co-segmentation as the silhouettes of the object we use shape from silhouette [15] to recover the 3D model of the object of interest shown in Figure 2.3(d). The volume of intersection recovered eliminates the sparse errors in the background. We now project the 3D model back into each view to fix the errors and obtain a clean co-segmentation of the object of interest Figure 2.3(e).

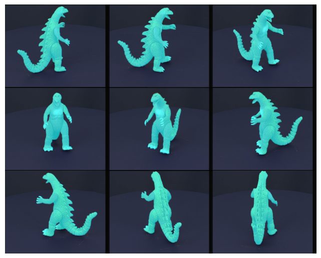
Non-planar object of interest modeling

We demonstrate the effectiveness of our algorithm on a number of datasets ranging from a simple collection taken in a controlled setup to a community photo collection and a video captured in cluttered scenes. In this section, we show the rendered 3D model for each dataset, captured from novel view-points. For all datasets except the dino dataset, we texture map the model by back-projecting the faces of the mesh onto a single image. We observed that in outdoor scenes texture mapping from multiple views can lead to some artifacts at the seams due to changes in illumination.

Dino Dataset. The first dataset we use is a standard dataset from the Oxford Visual Geometry Group[3], shown in Figure 2.4(a). One of the images in the dataset was chosen at random and the interactions were provided to indicate the dino as foreground and the blue screen as the background. The resulting silhouettes are shown in Figure 2.4(b). The 3D model obtained from the shape-from-silhouette algorithm. This dataset was captured in a controlled setup which allowed us to texture map the model using multiple views i.e. by projecting the faces onto the corresponding image where they are visible. Occlusion poses a significant problem for multiview texturing, we use the approach of Chen et al. [14] to overcome this by using the depth buffer (z-buffer) data from the graphics card.

[3]Oxford Visual Geometry Group multiview dataset: http://www.robots.ox.ac.uk/~vgg/data/data-mview.html


(a)


(b)

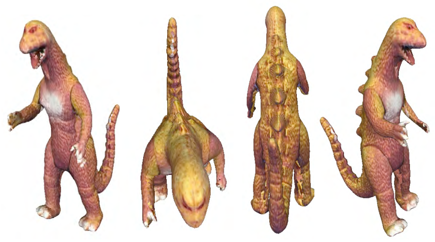
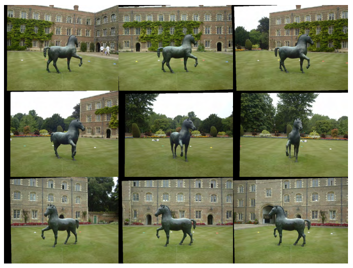
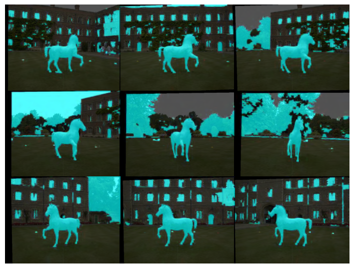

(c)

Figure 2.4: Dino dataset (36 images): (a) Subset of the collection of images given to the system where the dino was marked the object of interest; (b) Resulting silhouettes after initial co-segmentation (in cyan color); (c) Some sample novel views of the 3D model.


(a)


(b)

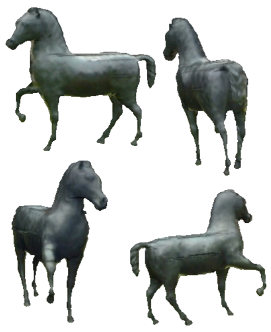
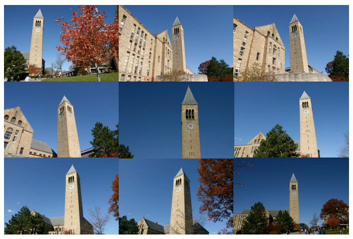

(c)

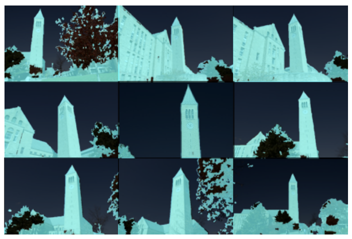
Figure 2.5: Cambridge unicorn dataset (14 images): (a) Subset of the collection of images given to the system where the unicorn statue was marked as the object of interest; (b) Resulting silhouettes after initial co-segmentation (in cyan color); (c) Some sample novel views of the 3D model.

This result simply serves as a proof of concept under a controlled setup, and it is encouraging to see that our approach is able to render a good reconstruction *without* any prior knowledge about this setup.

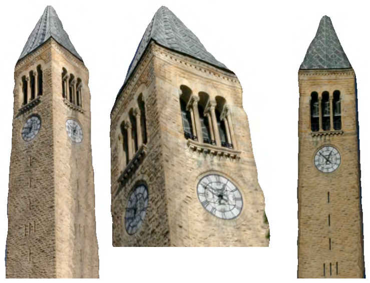
Cambridge Unicorn Dataset. We use the Cambridge unicorn dataset [96], shown in Figure 2.5(a). Using interactions to indicate the unicorn as the object of interest, we obtain the silhouettes as shown in Figure 2.5(b) which results in the texture-mapped 3D model as shown in Figure 2.5(c).



(a)



(b)

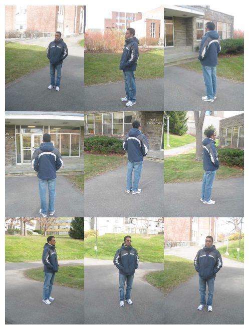


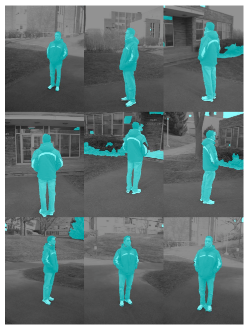
(c)

Figure 2.6: Clock tower dataset (32 images): (a) Subset of the collection of images given to the system where the clock tower was marked as the object of interest; (b) Resulting silhouettes after initial co-segmentation (in cyan color); (c) Some sample novel views of the 3D model.

Clock Tower Dataset. We now try to reconstruct immovable objects that cannot be taken to a standard studio setup as shown in Figure 2.6(a). With interactions we obtain the silhouettes of our object of interest (clock tower), as shown in Figure 2.6(b) which can be used to obtain texture mapped 3D model using the shape-from-silhouette algorithm as shown in Figure 2.6(c). We note that algorithms like structure-from-motion and patch based multi-view stereo would try to reconstruct the whole scene and result in an incomplete reconstruction in this case.

Statue Dataset. We now demonstrate the effectiveness of our algorithm on images where the background changes drastically as shown in Figure 2.3(a). The silhouettes of the object of interest (statue) obtained using our algorithm are shown in Figure 2.3(c). The texture mapped 3D model of the statue obtained using these silhouettes are shown in Figure 2.3(d). Note here that a part of the head of the statue gets clipped off in the generated model. The reason for this is a leak in the superpixels where a portion of the head became part of the sky superpixel. We can overcome this problem by working on pixels instead of su-


(a)


(b)

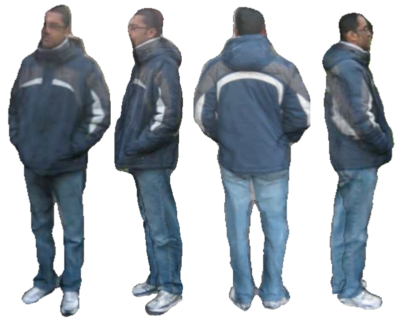
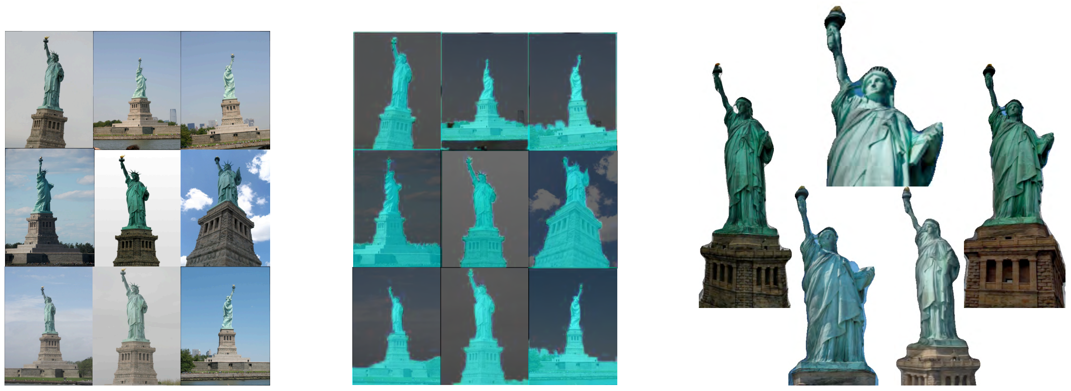

(c)

Figure 2.7: Video dataset (17 images obtained by sampling the video): (a) Subset of the collection of images given to the system where the person was considered the object of interest; (b) Resulting silhouettes after initial co-segmentation; (c) Some sample novel views of the 3D model.

perpixels i.e. set up the energy minimization over a graph of pixels instead of superpixels. This would increase the computational complexity but result in better silhouettes.

Video Dataset. Our work opens up the possibility of allowing users to render themselves as avatars in virtual worlds. We consider this scenario of reconstructing a person in 3D. We captured a video of the person to be modeled by walking around them. Selected frames this video are shown in Figure 2.7(a). With interactions, we obtain the silhouettes as shown in Figure 2.7(b) which results in the texture mapped 3D model as shown in Figure 2.7(c). We can see that the reconstruction is fairly complete, however we observe a leak in the superpixel map here as well.

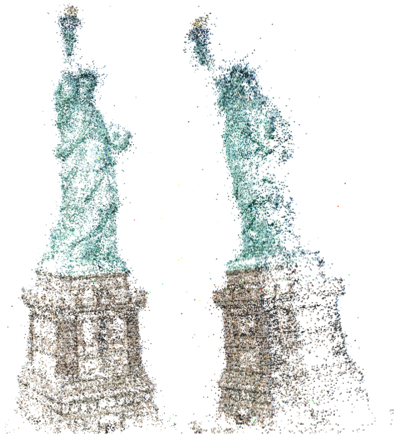
Community Photo collection - Statue of Liberty dataset. With millions of images available on the internet, we consider an application geared towards internet-scale reconstruction of objects where the user searches for an object of interest, in this case the *Statue of Liberty*. We start with a set of 1600 images of



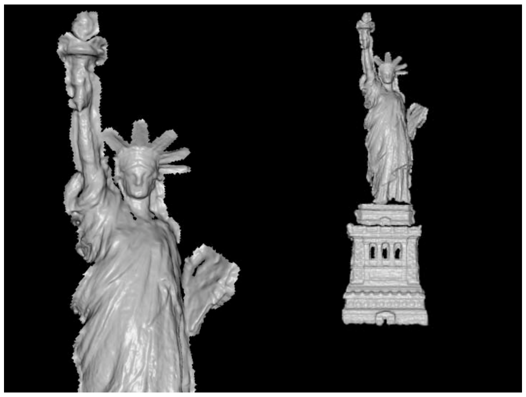
(a) (b) (c)

Figure 2.8: Community photo collection - Statue of Liberty dataset: (a) Subset of the collection of images given to the system - for our co-segmentation algorithm we use a subset of 15 images spanning a large field of view from a collection of 1600 images; (b) Resulting silhouettes after initial co-segmentation (in cyan color); (c) Some sample novel views of the 3D model.

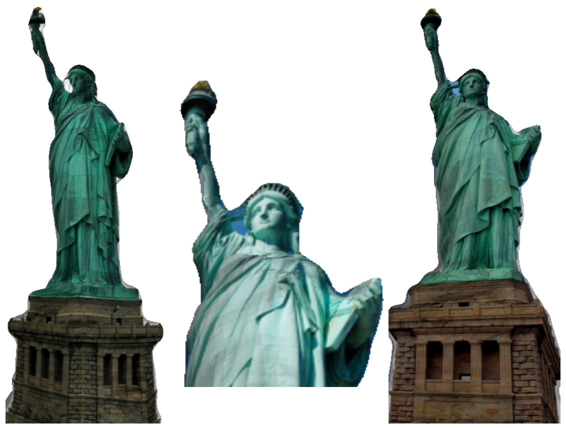
the Statue of Liberty collected by Snavely et al. from Flickr®. We use all the images to estimate the camera matrices using structure-from-motion [88]. For our algorithm, we sampled a subset of 15 images spanning a large field of view, as shown in Figure 2.8(a). The silhouettes are shown in Figure 2.8(b) and the texture-mapped 3D model are shown in Figure 2.8(c). We note here that there are a few artifacts like the blue sky above the shoulder as well as the thinned arm. Some of these problems (like the superpixel leaks) may be corrected by working with pixels. However, some (like the lack of detail on the face of the Statue of Liberty) are a direct result of our reliance on a segmentation framework and may not be possible to fix. The results on this dataset have also been reported by the multi-view stereo work of Goesele et al. [30], where they obtain a dense depth model for the statue. A comparison between the model we generate, the point cloud model from photo-tourism [88] and multi-view stereo model [30] is shown in Figure 2.9.



(a)



(b)



(c)

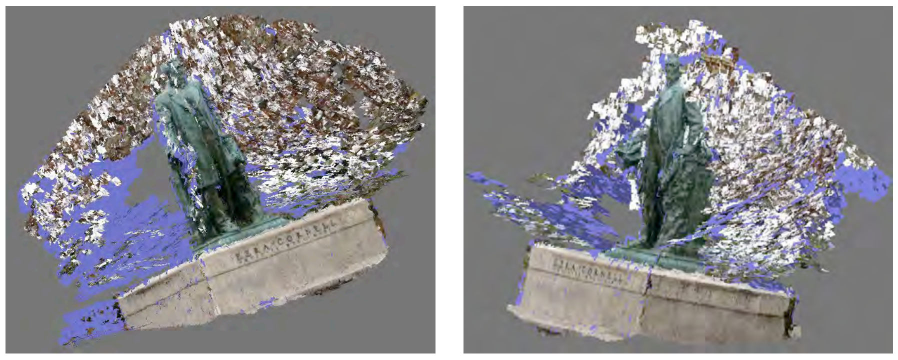
Figure 2.9: Statue of Liberty comparison: (a) Point cloud reconstruction using 1600 images ([88]); (b) Dense reconstruction using multi-view, using 72 images (figure from [30], used with permission). With a lot of images, multi-view stereo can give a good depth model; (c) Pleasing texture mapped reconstruction rendered using 15 images.

It is worth mentioning that once we obtain the silhouettes of the object of interest, we can use any known shape-from-silhouette algorithm at this stage to obtain the 3D model (not necessarily the octree-based reconstruction approach we used here), for example, the approach by Wong et al [95]. In addition, we can use this co-segmentation algorithm with the popular multi-view stereo reconstruction approach, to focus the output of the multi-view stereo algorithm on the object of interest. As an illustration, we show the model generated using patch-based multi-view stereo (PMVS) [4] in Figure 2.10 when constrained by the silhouettes extracted using our interactive co-segmentation algorithm. We used the statue dataset in Figure 2.3(a) for this experiment. In Figure 2.10(a), we show the result of PMVS without any prior knowledge of the object of interest. In Figure 2.10(b) we show the 3D model obtained from PMVS using our silhouettes. As we explained earlier, multi-view stereo algorithms would try to reconstruct the whole scene without giving importance to the object of interest. We can see that use of silhouettes helps obtain a more accurate 3D model of the object of interest. Another crucial advantage of using the silhouettes is to speed up the multi-view stereo algorithm with geometrically consistent reconstructions. In our experiment with PMVS, it took 3 hours to obtain the model in Figure 2.10(a), as opposed to 8 minutes using the silhouettes to render Figure 2.10(b). However, faster implementations may be available for PMVS.

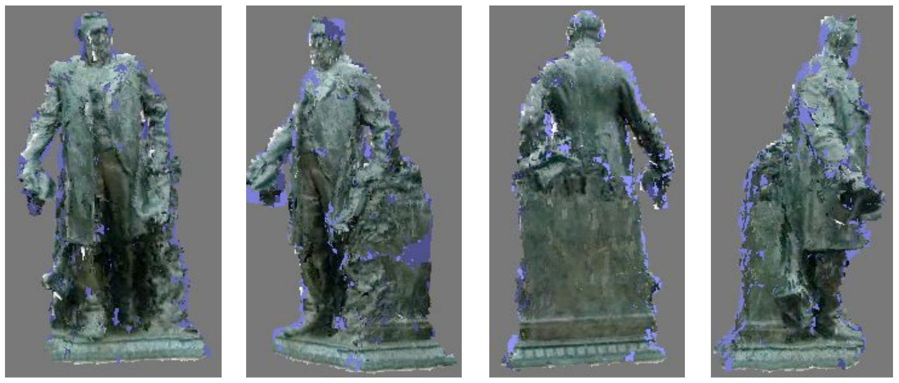
2.4.3 Reconstructing planar scenes

We have considered reconstructing non-planar objects in the previous section, we now consider obtaining piecewise planar reconstructions of the scene. We

[4]We use the PMVS implementation described in [26] available at http://grail.cs.washington.edu/software/pmvs/pmvs-1/index.html



(a)



(b)

Figure 2.10: Patch-based multi-view stereo experiment using images in Figure 2.3(a) where the statue is the object of interest: (a) When the silhouettes are not available PMVS tries to reconstruct the whole scene as shown; (b) Using the silhouettes produced by the co-segmentation algorithm, we can use PMVS to obtain the 3D model of the statue which was the object of interest.

use the interface to display the multiview image collection to the user. The user selects an image and provides scribbles on the image with different colors indicating different planar surfaces as shown in Figure 2.11(b). In the context of the formulation described in Section 2.4.1, the user provides node constraints for L planar surfaces, the algorithm learns the appearance model to describe each surface and sets up the energy function solved via graph-cuts (with α-expansion). The result segments the image into the different surfaces labeled by

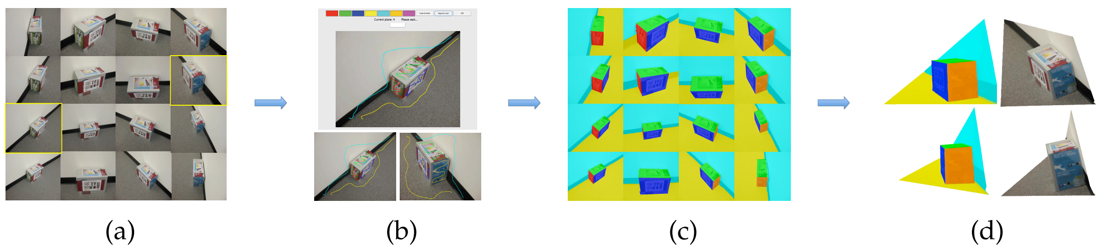


Figure 2.11: Interactive piecewise planar 3D reconstruction: (a) Input images (image selected by user shown in yellow box); (b) User interactions to indicate the surfaces in the scene; (c) Scene co-segmentation of all images by using the idea of 3D scribbles to propagate scene geometry; (d) Some sample novel views of the reconstruction of the scene, with and without texture.

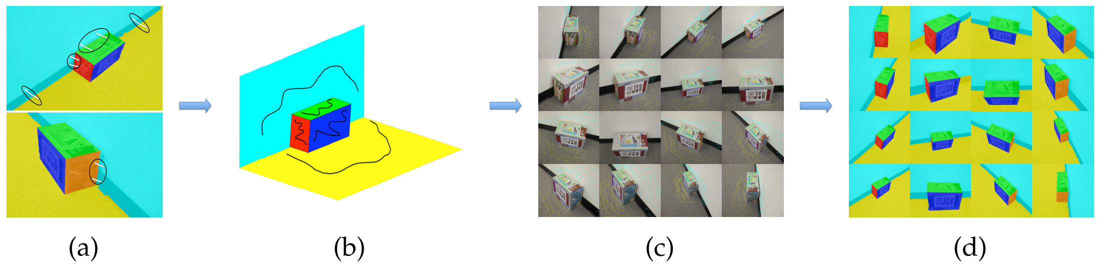


Figure 2.12: Scene co-segmentation: (a) Scene segmentation with user interaction indicating connected planes (white scribbles in black ellipses); (b) 3D scribbles inferred from the segmentation; (c) 3D scribbles warped onto the other images to propagate scene geometry (Note: scribbles have been increased to improve visibility; the scribbles used for the results are in Figure 2.11(b)); (d) Scene co-segmentation.

the user as shown in Figure 2.12(a); we call this *scene segmentation*.

Scene segmentation to 3D geometry

Using SfM we have a sparse 3D point cloud and the 2D feature correspondence across the images for this point cloud. We therefore know the subset of 3D feature points seen from the current view (scribbled image). This information helps transfer the labels from the 2D scene segmentation to the 3D points, based on which scene segment the 3D points project onto. We now use RANSAC-based plane-fitting on the labeled 3D points to estimate the plane parameters enforcing that the plane normal points outwards i.e., towards the camera looking at the scene.

We note that there may be featureless surfaces like the wall in the scene, which lacks enough 3D point support to be reconstructed. The algorithm then prompts the user for some simple additional interactions to indicate the edges shared by this surface with the other surfaces in the scene by easily scribbling two lines across the edge shared as shown in black ellipses in Figure 2.12(a). We obtain an estimate of the plane parameter by enforcing that the boundary points correspond to the 2D projection of the line of intersection of the connecting 3D planes, thus, resulting in globally optimal plane parameters. However, if the featureless surface shares just one edge with another plane, we make perpendicularity assumptions for that surface to choose the most probable plane amongst the infinite planes which shares that edge. This assumption has been shown to work well [39] and would be the best possible estimate, given the support.

3D scribbles and scene co-segmentation

Our goal is to obtain a co-segmentation of the planar surfaces in each of the images. Co-segmentation of the multiple surfaces in the scene is not as trivial as the binary image co-segmentation since, it is hard to define features discriminative between the geometric surfaces. However, when a user provides scribbles on an image, they are doing so based on their perception of the geometry of the scene, i.e., they are not just indicating surfaces and objects in *that* image but, are giving us cues about the 3D scene geometry common across all the images. This is the common thread between the images we exploit to perform the co-segmentation.

3D scribbles. Using the estimated plane parameters and the camera projection matrix of the scribbled image, we develop the idea of *3D scribbles*. Let the projection matrix of camera i be defined as $M_i = K_i R_i (I - C_i)$ where, K_i is the intrinsic matrix, R_i is the rotation matrix and C_i is the camera center in the world coordinate system. Consider, a 2D scribble point $s_{1,j}$ seen from the first camera, on a segment which corresponds to the plane l parameterized by $[\hat{n}_l \ d_l]$ where, $\hat{n}_l$ is the plane normal and d_l is the plane constant. The projection of this scribble point on another image seen from the second camera ($s_{2,j}$) is given by,

$$s_{2,j} = K_2 R_2 \left(\Big(\frac{(-d_l - \hat{n}_l.C_1)}{\hat{n}_l.([K_1 R_1]^{-1} s_{1,j})} [K_1 R_1]^{-1} s_{1,j} + C_1 \Big) - C_2 \right) \tag{2.3}$$

We take care to avoid warping the scribbles onto occluded planes by using the scene geometry and camera pose. For example, we consider the warped scribbles only on the planes visible from a particular view.

Scene co-segmentation. The resulting scribbles on the images are as shown in Figure 2.12(c). Using these scribbles as node constraints on all the images, we extend the energy minimization based multi-class labeling described in Section 2.4.1 to all the images thereby achieving co-segmentation of all the images into the multiple planar surfaces Figure 2.12(d).

We use the back-projection algorithm to evaluate the point of intersection of a ray from the camera center through every pixel on the image plane, and the estimated 3D surface. Using these 3D points, we generate a mesh for the scene with the corresponding image texture and render a texture mapped planar reconstruction of the scene as shown in Figure 2.11(d), enabling pleasing fly-throughs.

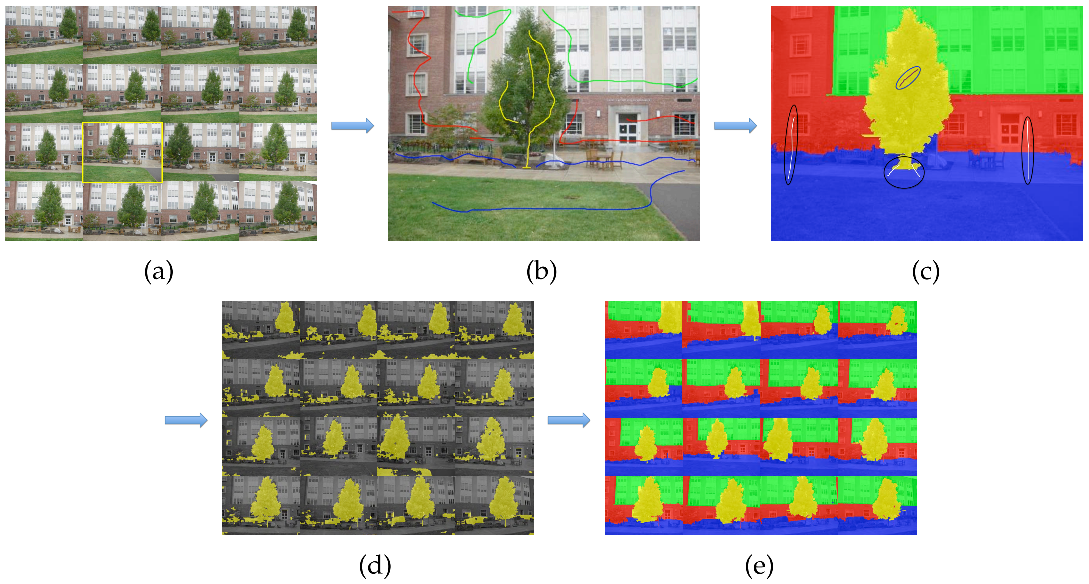


Figure 2.13: Outdoor scene with occluding non-planar object: (a) Input images (image selected by user shown in yellow box); (b) User interactions; (c) Resulting scene segmentation with the additional interactions to indicate surface connectedness (white scribbles shown in black circles) and non-planar objects (magenta scribble shown in blue scribble); (d) Object co-segmentation (foreground non-planar object in yellow); (e) Scene co-segmentation by using 3D scribbles to propagate scene geometry.

Rendering non-planar objects in planar scenes

The algorithm thus far renders a planar reconstruction of the scene. In case of non-planar objects in the scene, we get an input from the user to indicate these objects, as shown in the blue ellipse in Figure 2.13(c). This tells the algorithm which surface and node constraints correspond to the non-planar object. Note that recent automatic approaches [28, 64] can also be used to identify non-planar regions. We estimate an approximate planar proxy for the object, which helps position the object as part of the rendered scene. We then use the algorithm described in Section 2.4.2 to obtain a visual hull of the non-planar object, which is rendered as part of the scene using an independent mesh.

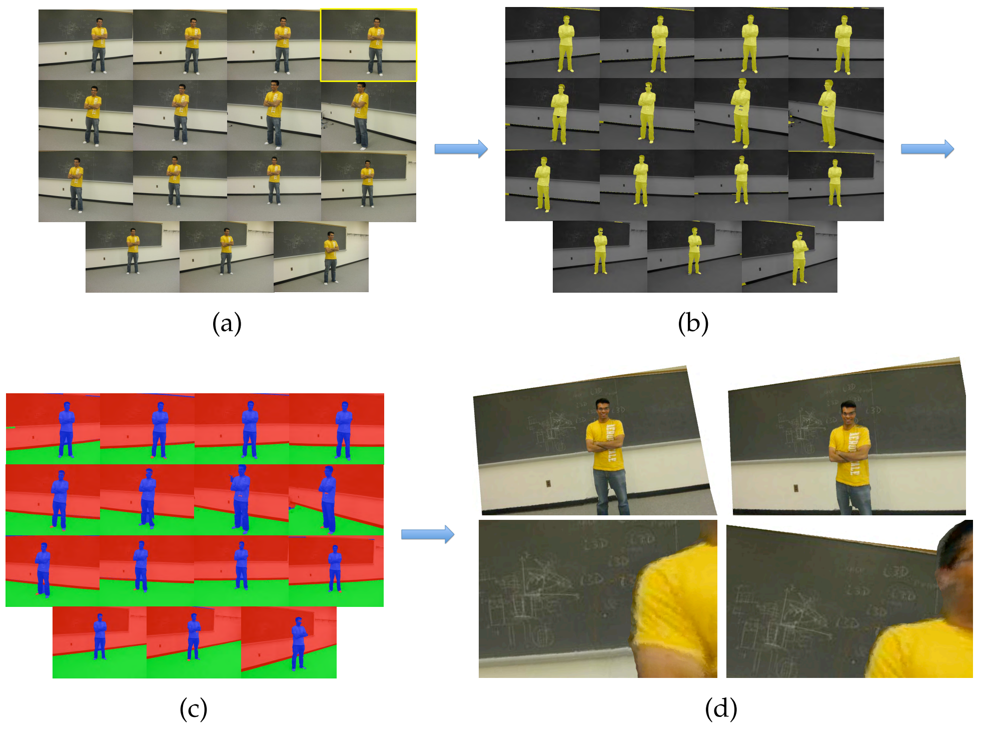
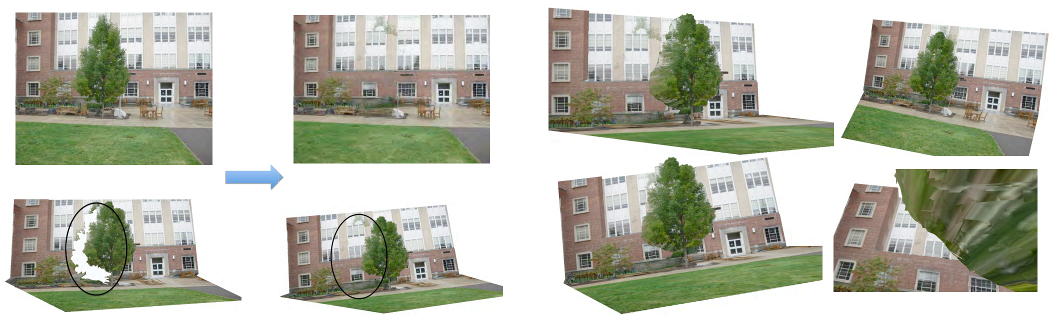


Figure 2.14: Indoor scene with occluding non-planar object: (a) Input images (image selected by user shown in yellow box); (b) Non-planar object co-segmentation; (c) Final scene co-segmentation; (d) Novel views of the reconstruction with volumetric rendering of the person.

The scene co-segmentation allows us to create a composite texture map for the scene covering up holes due to the occluding non-planar object as shown in Figure 2.15(a). The algorithm renders the non-planar objects as part of the planar scene as we show with the tree in the outdoor scene in Figure 2.15(b) and the person in the indoor scene in Figure 2.14. Once the algorithm generates the 3D reconstruction, the user can provide more scribbles to indicate new or previously occluded planes, and improve the result, thus closing the loop on our interactive 3D reconstruction algorithm that is initiated by the user via node constraints. Please see video summary[5] with fly-through of the 3D reconstructions.

[5]http://chenlab.ece.cornell.edu/projects/Interactive_3D



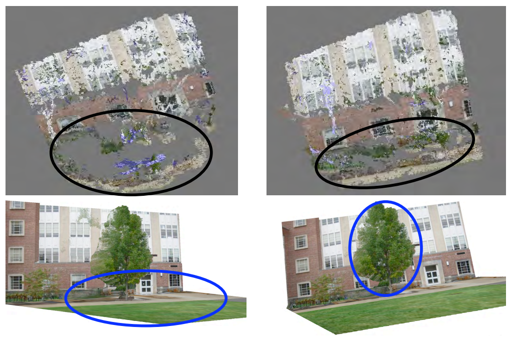
(a) (b)

Figure 2.15: Non-planar objects: (a) Composite texture map for the scene (top) allows covering up holes due to occlusions (ellipse); (b) Novel views of the reconstruction with a volumetric model of the tree.

Comparison

We compare our results with other publicly available algorithms to reconstruct a scene. Using SfM [88], on a huge image collection can render dense point clouds however, in this scenario, the point cloud is very sparse. Multi-view stereo algorithms like patch-based multi-view stereo (PMVS)[6] render a denser reconstruction. However, this fails to render a complete reconstruction, leaving holes and rendering inaccurate geometric reconstructions, in the presence of textureless surfaces and specular surfaces. As we show in Figure 2.16, the results from our interactive reconstruction algorithm is more complete and geometrically accurate.

[6]We use the PMVS implementation by Furukawa et. al. [26] and available at http://www.di.ens.fr/pmvs/


(a)

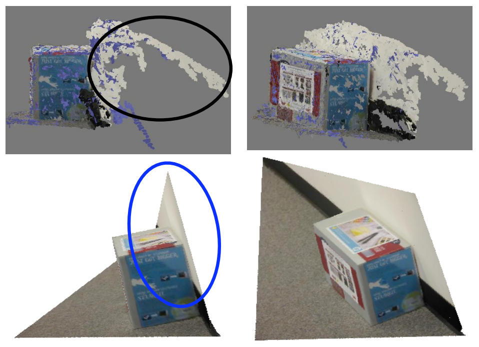

(b)

Figure 2.16: Comparison with patch-based multi-view stereo: The top images show the reconstruction generated by PMVS with the errors shown in black ellipses, while bottom images show our results with corrected reconstructions shown in blue ellipses.

2.5 Edge constraints: An active learning formulation

In our alternate algorithm, we intend to accept edge constraints from the user. Therefore, we need a smart computational engine that can automatically estimate the 3D structure of the scene and accept the user input across edges when, and where needed. We do so using an active-learning algorithm. We refer to Figure 1.1 and consider the ingredients for an active-learning algorithm in the context of image-based modeling. The integral components are: an automatic 3D reconstruction algorithm (computational engine); an approach to quantify the uncertainty of the algorithm and sample the most informative queries for user feedback; the human oracle who provides suitable interactions in response to the query; and lastly, an approach to seamlessly incorporate the feedback from the user into the algorithm. We describe each of the above aspects with respect to our algorithm in detail in the following sections.

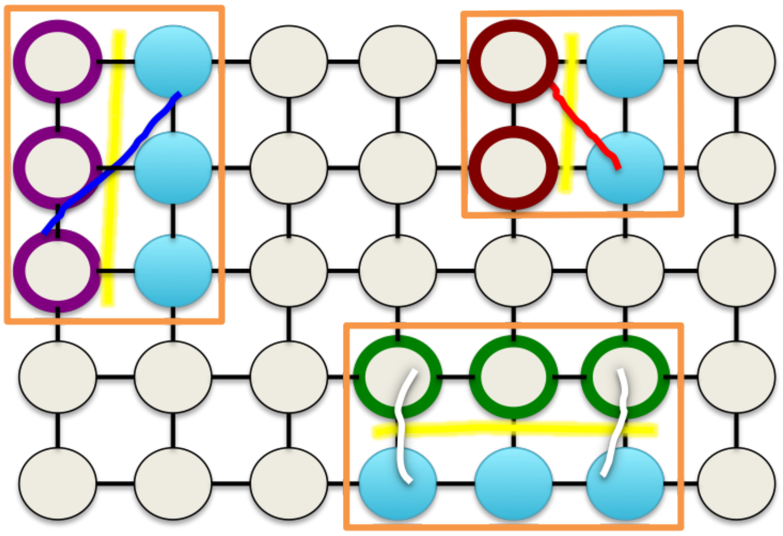


Figure 2.17: Edge constraints. The blue, white and red scribbles across the yellow edges in the graph illustrate the edge constraints provided by the user to provide support for the cyan nodes (guided by the computational engine). More description about the edge constraints and incorporating them into the framework is in Section 2.5.

2.5.1 Automatic 3D reconstruction algorithm

We develop a piecewise planar 3D reconstruction algorithm described below using successful ideas from recent works [25, 28, 70, 85].

Dense plane hypothesis generation

We use patch-based multiview stereo (PMVS) by [26] as a pre-processing step, which although not as accurate as the sparse point cloud from SfM [88], provides a much denser set of points that span the scene. Similar to [85], we hypothesize dominant planes by analyzing the distribution of depths of the 3D points along each hypothesized normal (using the estimated vanishing direc-

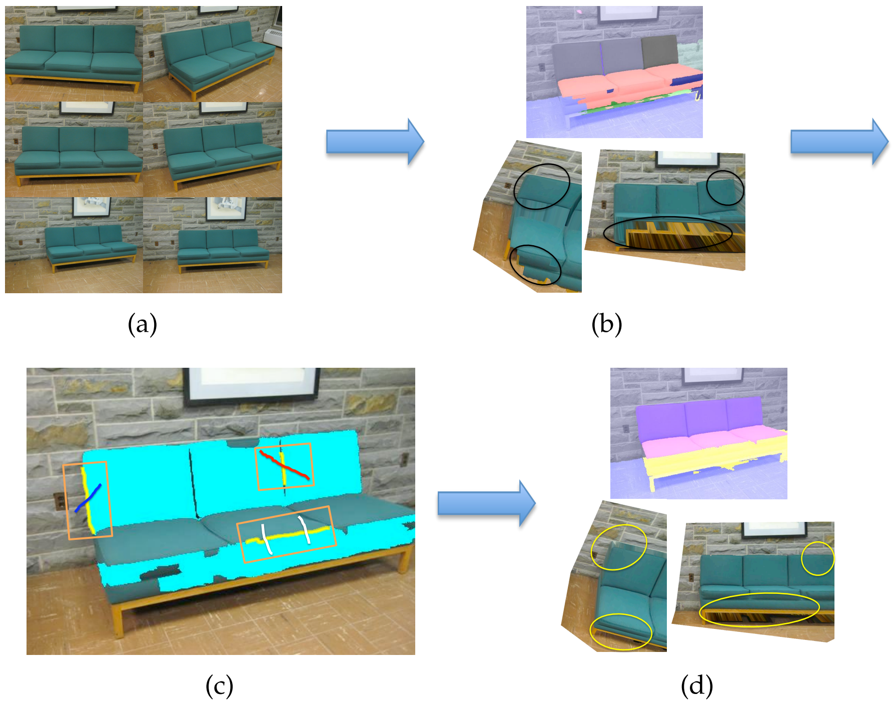


Figure 2.18: (a) shows a set of multiview images of a scene; (b) shows the result of the automatic algorithm, the plane labeling shown on the top indicates the inaccurate labeling, the novel views of the 3D model are shown at the bottom with black circles showing the errors. (c) the proposed active-learning algorithm quantifies the uncertainty of the algorithm and detects the uncertain regions (in cyan), the uncertainty boxes (in orange) with the highlighted edges (in yellow) are used to query the user for support, the user provides any of three types of interactions within each box via simple scribbles across the highlighted edge, coplanar scribbles (red), not-coplanar scribbles (white) or not-connected scribbles (blue) as shown; (d) shows the result of the algorithm after incorporating the information provided by the user, plane labeling on top shows the improved labeling, the improved reconstruction is shown below through novel viewpoints with yellow circles illustrating the corrected geometry.

tions). We break down an image into superpixels[7] and use the assumption that every superpixel would lie on a planar surface [70, 81]. Using these superpixels, we hypothesize additional planes by fitting planes to 3D points that project

[7]We use graph based segmentation [22] to break each image down to about 400 superpixels.

onto the same superpixel. In practice, we observe that this allows us to add new planes not hypothesized before as their normals are different from the dominant normal directions.

Energy minimization

The dense plane hypothesis stage results in P (about sixty) planes. These hypothesized planes serve as the set of discrete labels, which changes the piecewise planar reconstruction problem to a multi-label segmentation problem, formulated as an energy minimization problem over the superpixels. The formulation is similar to that described in Section 2.4.1, with the discrete label space $\{0, 1, \ldots, P-1\}$. The unary and pairwise term for the automatic piecewise planar stereo algorithm is defined below.

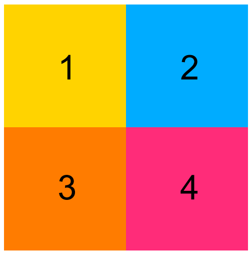
Unary term. For a particular view, we compute homographies for each plane to warp the other images to that view. We use normalized cross-correlation (NCC) to quantify the warp error. We refer the reader to [85] for more details. We compute the NCC using the superpixel as support at each pixel as opposed to a constant window. We also compute a color term that measures the mean color difference of each superpixel between the original and the warped image. We use a weighted combination of the two normalized terms as the unary term with the weights tuned by observing the performance on one of the datasets.

Pairwise term: Co-planar classifier. We introduce an *adaptive* co-planar classifier to model the pairwise term. We learn a classifier that given a pair of adjacent superpixels returns a score representing the co-planarity of the superpixels. We use the geometric context dataset by [42] (with seven ground truth geometric labels). Adjacent superpixels with the same geometric label are used as positive data points while pairs with different labels, are used as negative data points. We note that adjacent superpixels lying on occluding 'parallel' planes would be bad data points, but, in practice this does not hinder the performance. We use *relative* features (difference features) such as color, texture, and shape features (more details about features in [42]) for each pair of superpixels as the feature vector for each data point and learn a logistic regression model. This model is continuously updated by the active-learning algorithm. We note that one can also use laser image data to learn a co-planar classifier by fitting planes to the laser data to obtain the samples needed [81]. We use a Contrast Sensitive Pott's Model to model the pairwise term.

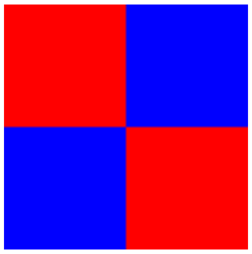
$$E_{ij}\left(X_i, X_j\right) = I(X_i \neq X_j)\ exp\left(-\beta d_{ij}\right) \tag{2.4}$$

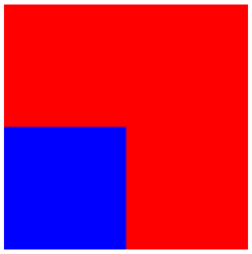
The pairwise term when adjacent superpixels take different labels should be high when the contrast d_{ij} is low or when the superpixels are likely to be co-planar and high otherwise. Thus, given a pair of adjacent superpixels, using the learnt co-planar classifier, we obtain a score that represents the likelihood of this pair being co-planar. This score is used to model the contrast d_{ij} (1 - similarity score) in the Contrast Sensitive Pott's Model.

Finally, we use graph-cuts (with α-expansion) to compute the MAP labels for all superpixels [1, 9, 10, 51]. This allows us to automatically obtain the piecewise planar reconstruction of the scene. At this stage the algorithm has used the observed multiview stereo cues to obtain the piecewise planar reconstruction albeit with errors as shown in Figure 2.18(b). We explain below our active-learning algorithm to fix the errors in the reconstruction by putting the user into the loop to provide edge constraints deriving support from the current 3D reconstruction of the scene.



(a) Four node graph


(b) Ground truth labels


(c) Incorrect labeling using ambiguous unary terms

Smooth term	Label a	Label b
Label a	0	0.1
Label b	0.1	0

(d) Pairwise term

Data term	1	2	3	4
Label a	0.1	0.5	0.5	0.1
Label b	0.5	0.1	0.1	0.5

(e) Good unary terms

Data term	1	2	3	4
Label a	0.1	0.5	0.5	**0.8**
Label b	0.5	0.1	0.1	**0.9**

(f) Low confidence unary terms

Data term	1	2	3	4
Label a	0.1	**0.1**	0.5	0.1
Label b	0.5	**0.1**	0.1	0.5

(g) Ambiguous unary terms – Example 1

Data term	1	2	3	4
Label a	0.1	**0.2**	0.5	0.1
Label b	0.5	**0.05**	0.1	0.5

(h) Ambiguous unary terms – Example 2

Figure 2.19: Synthetic example to illustrate the uncertaintly of the algorithm (Best viewed in color). More details in Section 2.5.2.

2.5.2 What is the uncertainty?

An important aspect of an active-learning algorithm is to identify the uncertainty of the algorithm. Intuitively, since our algorithm follows an energy minimization framework to solve the multilabel problem over the graph of superpixels, we quantify the uncertainty of the algorithm with respect to the uncertainty in labeling the superpixels. At a high level, we evaluate the uncertainty of a superpixel in terms of *confidence* and *ambiguity*.

Synthetic example. We explain our intuition through a small synthetic example. Consider, a four node graph with their 4-connected neighborhood as shown in Figure 2.19a. Let us suppose the ground truth labeling consists of two labels as shown in Figure 2.19b. Now, Table 2.19d shows the pairwise term and Table 2.19e shows an instance of unary terms which gives the ground truth labeling. Note that the unary terms are energies so the lower the value the more affinity to the label. Also note that the unary terms for only the two relevant labels are shown, assume that the energies for the other labels are high and hence not relevant. In Table 2.19f we observe that while the energies of the two labels for node 4 reflect that it has a preference for Label a, both the energies are very large (shown in red) indicating a *low confidence* in the decision. In Table 2.19g we observe that two labels have low energies indicating the *ambiguity* (shown in red) making the correct decision ambiguous. Similarly, in 2.19h the unary terms (shown in red) indicate that the node 2 should take the label b but incorporating the pairwise term causes the final labeling to be erroneous as in Figure 2.19c. We need an approach to label these nodes as uncertain. We note that entropy of the unary terms (say the ratio of the unary terms for these two labels) would help in case of Table 2.19g while the entropy in case of Table 2.19h would be low. We therefore need to incorporate the effect of pairwise to determine ambiguity. We explain how we identify these contributors to the uncertainty in detail below.

Confidence

Confidence quantifies how confident the algorithm is to assign a particular plane hypothesis to the superpixel. Low confidence superpixels represent high uncertainty regions, for example, occlusions. We obtain these regions via the energy minimization framework. Motivated by the multi-view stereo work by [13], we add an additional label to our set of discrete labels and refer to it as the *unknown* label. For every superpixel, X_i where $i \in V(\textit{all superpixels})$, the unary term $E_i(X_i)$ for the *unknown* label is set at a constant penalty. Intuitively, this penalty is large enough so it does not affect the unary terms of the more confident superpixels while low enough so that low confidence superpixels are separated out. We use the median of all the unary terms, which serves as a safe unary term value in practice for the *unknown* label. As opposed to using a simple threshold on the unary terms to determine low confidence regions, this approach gives the pairwise term an opportunity to try to derive support,when possible, for the low confidence superpixels from their neighbors. The superpixels that take the *unknown* label after the minimization are called *uncertain* superpixels.

Ambiguity

Ambiguity quantifies the uncertainty of the algorithm between different plane hypotheses. Superpixels that are ambiguous about multiple plane hypotheses represent high uncertainty regions, for example, textureless surfaces, specular surfaces, inaccurate plane hypotheses, etc. One approach to determine ambiguous data points in a multiclass labeling problem would be to analyze the unary terms, using the idea that the entropy of the unary terms of ambiguous data points would be high [44]. However, the entropy in the unary terms is not sufficient to capture *all* the ambiguity because the effects of the pairwise term are ignored. We thus evaluate the ambiguity by determining the ambiguity of resulting MAP labeling after incorporating the effect of the pairwise. We do so by using the *Graph-cut uncertainty* similar to [7], as explained below.

Let the minimum energy $E(X)$ for the graph $G = (V, E)$ be E_{min}. Given the complete set of plane hypotheses (L labels), suppose that for a superpixel X_i the minimum energy label is l_i. We flip the label of superpixel X_i from l_i to one of the the other labels l_j in L and recompute the energy, $E_{i \to j}$ of the labeling. At each such flip stage, we compute the absolute difference between the minimum energy (E_{min}) and flip energy ($E_{i \to j}$),

$$E(X_i)_{(\Delta[i \to j])} = \left|\left(E_{min} - E_{i \to j}\right)\right| \tag{2.5}$$

The ambiguity for every superpixel is computed by measuring the minimum of all such flip energy differences,

$$E(X_i)_{ambig} = \min_{j \in L \setminus i} E(X_i)_{(\Delta[i \to j])} \tag{2.6}$$

The intuition behind this is simple. If the algorithm does not have high ambiguity about assigning a particular plane hypothesis to a superpixel, the ambiguity energy difference, $E(X_i)_{ambig}$ should be high. However, if this value is low, it amounts to ambiguity between different plane hypotheses and hence uncertainty. We normalize the ambiguity energy differences and threshold that at 95% to obtain the top 5% of ambiguous superpixels. These are again called *uncertain* superpixels. We note that min-marginals by [50] could also be used to capture ambiguity.

Region level uncertainty

In addition to the superpixel level uncertainty, we determine *region level* uncertainty. We determine regions (groups of superpixels) that take a particular independent plane label but have no support from the 3D point cloud, i.e. none of the 3D points project onto the region, and label them as uncertain. The intuition here is that, a set of superpixels with no support from the 3D points, taking their own independent plane label amounts to uncertainty.

Quantifying uncertainty

Grouping the uncertain superpixels to uncertain regions, we first identify and highlight all the boundary or support edges where user interaction might be needed. To ease the interactive process, we draw a box (uncertainty box) centered at this edge, scaled to be the larger of a minimum predefined size or two standard deviations of the edge size. Our active-learning algorithm then queries the user with the regions with the highest uncertainty or information gain. We thus need a metric to quantify the uncertainty of each box.

Consider n normals that span all the planes in the scene (from the initial plane hypothesis step). Superpixels are organized in increasing order of cost, based on the lowest cost the superpixel pays for adopting a particular normal (e.g. $C1_s, C2_s, \ldots, Cn_s$). This gives an indication about how certain it is about taking a particular normal. For a low uncertainty region, the value $C1_s$ would be considerably lower than the next best normal, i.e. $C2_s$.

Let R_i indicate the region under a box i that represents the set of all superpixels part of the uncertain region under the box. Let $R_{i,support}$ indicate the region under box i *not* part of the uncertain region under the box. Let $coplanarity(e)$ represent the score of the co-planar classifier for an edge e between two superpixels, and E_i indicate the set of all edges under a box i. The uncertainty is quantified through four terms: Ambiguity of the region in the box (A), Confidence of the support region in the box (F), Graph-cut uncertainty (GCU), and Co-planar classifier uncertainty (CoP).

$$A_i = \max_{s \in R_i} \frac{C1_s}{C2_s} \tag{2.7}$$

$$F_i = \max_{s \in R_{i,support}} (1 - C1_s) \tag{2.8}$$

$$GCU_i = \max_{s \in R_i} E(X_s)_{ambig} \tag{2.9}$$

$$CoP_i = \max_{e \in E_i} coplanarity(e) \tag{2.10}$$

Our final uncertainly score for each box i, is the sum of each of the component uncertainties defined in Eqn (6)-(9), using an equal weighting for each term as a fair setting. In practice, equal weights work well, as we show in Section 3.3.2. We rank the boxes according to this score and query the user with the top three uncertainty boxes for some support. We note that we can achieve a steady improvement by querying the user with only *one* most uncertain box instead of the top three, however, this would need additional iterations of the algorithm, requiring additional user interactions and incurring processing overhead.

2.5.3 User in the loop

In our active-learning framework, given the uncertainty boxes, we wish to obtain user interactions in the form of support for the uncertain regions or edge constraints and incorporate this feedback into the algorithm to improve the reconstruction. The user (oracle) provides one of three scribble based interactions described below, within each box as shown in Figure 2.20.

Connected and co-planar regions. When the edge highlighted in the uncertainty box is an edge between connected and co-planar regions, i.e. *same plane*, the user provides a scribble as support across the edge to indicate co-planarity, shown as the red scribble (Figure 2.20). We use this additional information to improve the support for the uncertain superpixels. This is done by adding long-range edges (non adjacent nodes) between the nodes (superpixels) scribbled on by the user to allow the algorithm to propagate the confident label to the uncertain superpixels.

Connected but not co-planar regions. In case the highlighted edge is an edge between connected but not co-planar regions, i.e. *different planes*, the algorithm would need cues about the edge shared between these two regions in order to hypothesize a good plane for the uncertain region. We do so by allowing the user to use two white scribbles across the edge to indicate the edge segment shared by the planes (Figure 2.20). The edge constraint from the user is first used to break edges of the graph to avoid inaccurate labeling. In addition, using the confident region we obtain the positions of these edge points in 3D. Given this information and the hypothesized normals (Section 2.5.1), we use a RANSAC based approach to find the best fit plane through the 3D edge marked by the user. We add this new plane hypothesis and estimate the corresponding unary term as described in Section 2.5.1, adding hard constraints to ensure that the uncertain superpixels choose this new plane. This is therefore both an edge and a node constraint.

Not connected regions. If the highlighted uncertain edge corresponds to an edge between not connected regions, i.e. *occluding planes*, the user can indicate not-connected regions by using the blue scribble as shown (Figure 2.20). We in-

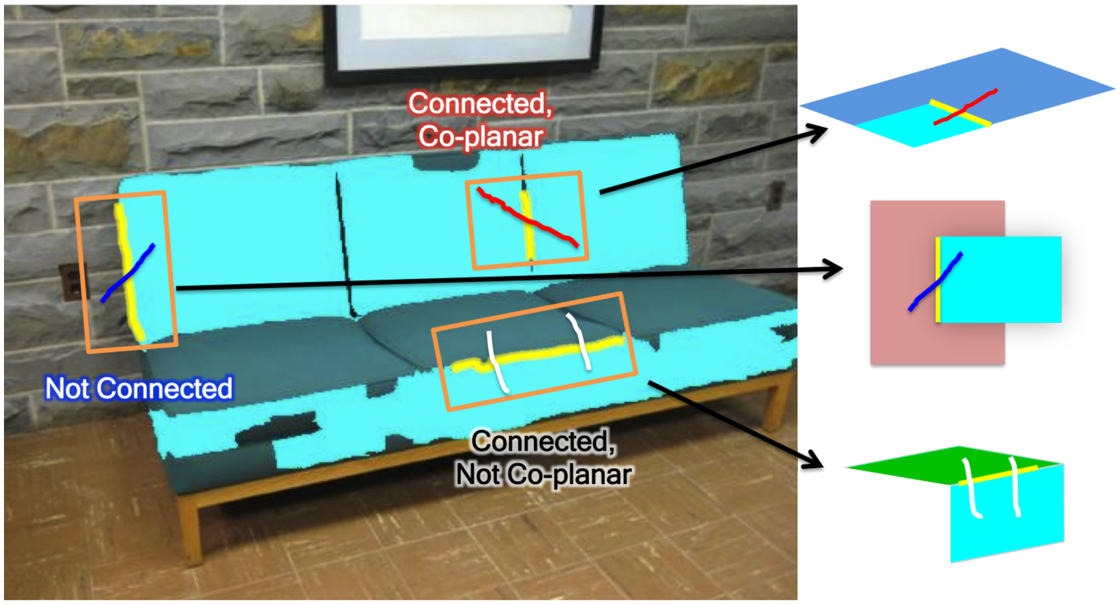


Figure 2.20: The user can provide three types of interactions to indicate coplanar regions (red), not-coplanar regions (white) and not-connected regions (blue) across the highlighted edge (yellow) within each uncertainty box (orange), to provide support for the uncertain regions (cyan).

corporate this information into the algorithm by breaking edges between these superpixels in our graph, thereby hindering these regions from taking the same plane.

We incorporate all the constraints provided by the user and suitably reformulate the graph over superpixels as illustrated in Figure 2.21. The connected and co-planar scribble (red) adds more edges to the graph and strengthens the edges to encourage that the uncertain nodes choose the neighboring confident node label. The not-connected scribble (blue) breaks edges in the graph ensuring that information is not passed between the nodes. Lastly, the connected and not co-planar scribble (white) breaks edges in the graph to ensure that the neighboring nodes do not choose the same label. A new planar surface is then

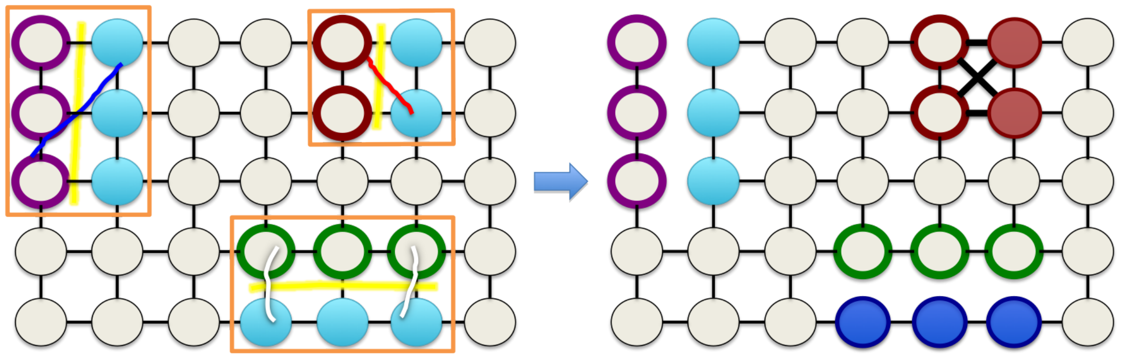


Figure 2.21: Incorporating the user constraints to update the structure of the graph. Note how the red scribble (connected and co-planar) adds more edges and strengthens edges; blue scribble (not-connected) breaks edges in the graph; white scribble (connected and not co-planar) breaks edges and hypothesizes a new planar surface for the uncertain region.

hypothesized for the uncertain region by using the uncertain edge as the 2D projection of the line of intersection of the 3D planes (similar to Section 2.4.3). In addition to modifying the graph, these constraints provide more information about the co-planar regions of the scene, which are used as additional samples to update the co-planar classifier. This updates the pairwise term, which makes the co-planar classifier scene specific. Using the energy minimization framework (Section 2.5.1) on this updated graph, we again obtain the MAP labels for the superpixels, which gracefully propagates the additional information given by the user. The process of obtaining uncertain regions, quantifying uncertainty, querying the user for support, and then updating the algorithm with the additional information is repeated using the new result, closing the loop on the active-learning algorithm.

2.6 Experiments and Results

In this section, we describe the datasets, the evaluation metric we use, and we discuss experiments to *quantitatively* evaluate the performance of the proposed active learning approach via machine experiments and a user study. We also discuss qualitative improvements in the reconstructions.

2.6.1 Datasets

We collect images spanning six scenes (each with about ten images) that lack geometric cues such as lines essential to the automatic algorithm and, include textureless surfaces or specular surfaces that hinder the performance of the automatic algorithm. We also use two standard datasets that have been used in prior automatic works [85]. We make all the datasets used in our works [53, 55, 56] publicly available[8].

2.6.2 Ground truth

To quantitatively evaluate the performance of the proposed active-learning algorithm, we first obtain pixel-wise ground truth segmentation of the planes for all the datasets. To capture some 3D information, we label ground truth normals for each segmented region. The ground truth pixel-wise segmentation along with their ground truth normals serves as a good quantitative indicator of the performance of the algorithm. Given the algorithm's result, we map

[8]*http://chenlab.ece.cornell.edu/projects/ActiveLearningFor3D*

each ground truth region to the largest label in that region in the algorithm's result, which agrees with the ground truth normal. Using these mapped labels we compute the pixel-wise labeling accuracy for each of the ground truth regions and compute the average accuracy across the datasets. We note that this metric can lead to inaccuracies in case of occluding parallel planes, however, it serves as a good metric to determine the relative performance in our experiments.

2.6.3 Machine experiments

In order to perform an exhaustive set of experiments to evaluate the various design choices, we develop a mechanism to generate *synthetic interactions*, which mimic the human user. For every uncertainty box queried by the algorithm, using the ground truth segmentation, normals, and the occlusion boundaries (manually labeled), we provide one of three interactions described in Section 3.5. We note that an iteration in our experiments refers to providing the interactions in any three distinct locations (e.g. within the three uncertainty boxes in the active-learning experiment).

Performance of active learning. We evaluate the performance of the proposed active-learning algorithm against ground truth sampling (an upper bound) and a random sampling experiment as shown in Figure 2.22.

In the ground truth sampling experiment (black curve), at each iteration, we compute a 2D error map using the algorithm's output and the 'ground truth'. The machine interactions are then aimed to provide support to these error regions, beginning from the largest error region, in the order of decreasing size. This is a good upper bound since at each iteration we aim to achieve the best improvement by directly correcting the errors. The active-learning experiment (blue curve) evaluates the performance of the proposed algorithm in which, the machine interactions are guided by the uncertainty boxes indicated by the active-learning algorithm. In the random sampling experiment (red), we do not use the proposed active-learning algorithm to choose the uncertain regions, but instead randomly sample the uncertainty boxes along the segmentation boundaries.

We see from Figure 2.22 that the proposed active-learning algorithm performs much better than random sampling and, in addition, performs respectably when compared to the upper bound, given that it does not have the luxury to access ground truth while querying interactions. We also note that it can achieve the peak performance achieved by the random sampling at the end of more than twenty iterations in as few as four iterations.

Evaluating algorithm design choices. We evaluate the design choices we incorporated into the proposed active-learning algorithm. We first evaluate the effectiveness of incorporating 'ambiguity' to describe uncertainty. The solid green curve in Figure 2.23 shows the performance of the algorithm when we ignore ambiguity and only rely on the confidence measure. In comparison with our active-learning curve (solid blue), we see that when the algorithm quantifies only the low confidence regions as uncertain, it fails to capture several critical uncertain regions, leading to a very slow and minimal improvement in performance.

In our algorithm, we use graph-cut uncertainty to capture ambiguity. We evaluate this choice observing the performance when we use the entropy of the data terms to directly detect the ambiguous regions, forced ambiguity curve

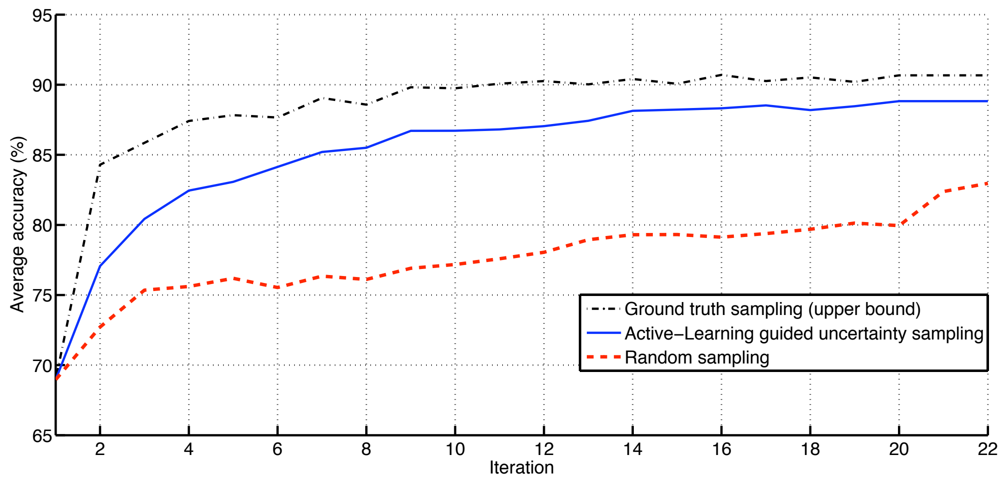


Figure 2.22: Machine experiments: Our proposed active-learning algorithm performs significantly better than random sampling and performs respectably compared to ground truth sampling.

(solid magenta) in Figure 2.23. This firstly strengthens the importance of ambiguity on comparing with the no ambiguity green curve and in comparison with the active-learning curve (solid blue) shows that graph-cut uncertainty captures relevant regions which are missed by the forced ambiguity.

Lastly, we evaluate the adaptive co-planar classifier. In Figure 2.23, comparing the solid curves with the corresponding dashed curves shows that using the adaptive co-planar classifier (CoP) gives steady improvement in performance in all the experiments.

2.6.4 User study

We perform a user study with ten users and three experiments to evaluate the performance of the algorithm. Figure 2.24 shows the performance of the users.

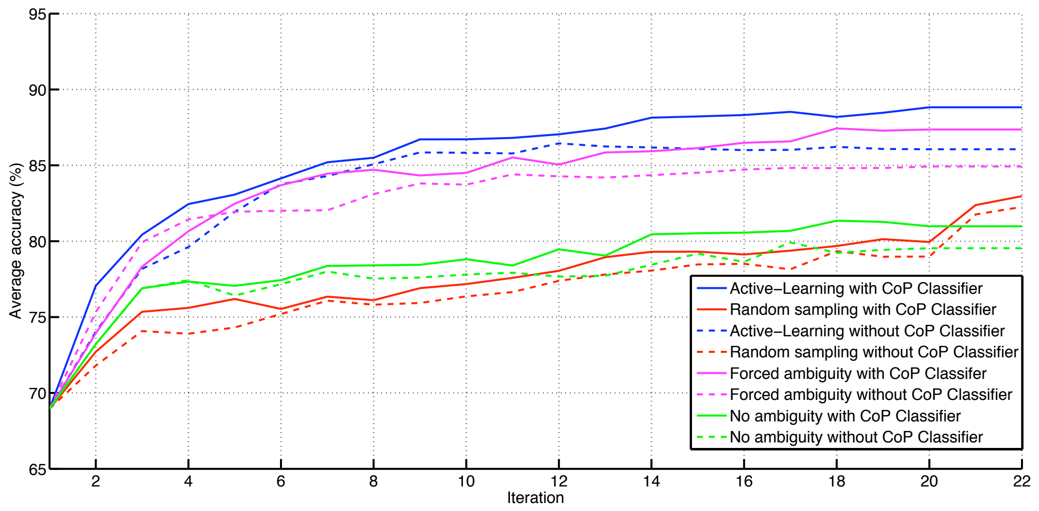


Figure 2.23: Machine experiments: Our proposed active-learning algorithm produces the most accurate reconstructions, validating our design choices. (Section 2.6.3).

We restrict the number of iterations to reduce the effort of the users. The first experiment is the random interactions experiment, in which we show the user the segmentation boundaries from the algorithm, however, with no indication about which regions are erroneous, as shown in Figure 2.25a. The user was instructed to provide three distinct interactions across any edge by observing the segmentations, with the only cue that each segmented region corresponds to a planar surface according to the algorithm. The *red* curve in Figure 2.24 shows the performance of the users. We observe that the human user performs better than the machine with the random interactions experiment because the human user has an implicit notion of the 3D structure of the scene. The annotations from the user are therefore more meaningful.

The second experiment is the exhaustive examination experiment. Here, in addition to the segmentation boundaries, we color code the normals of each seg-

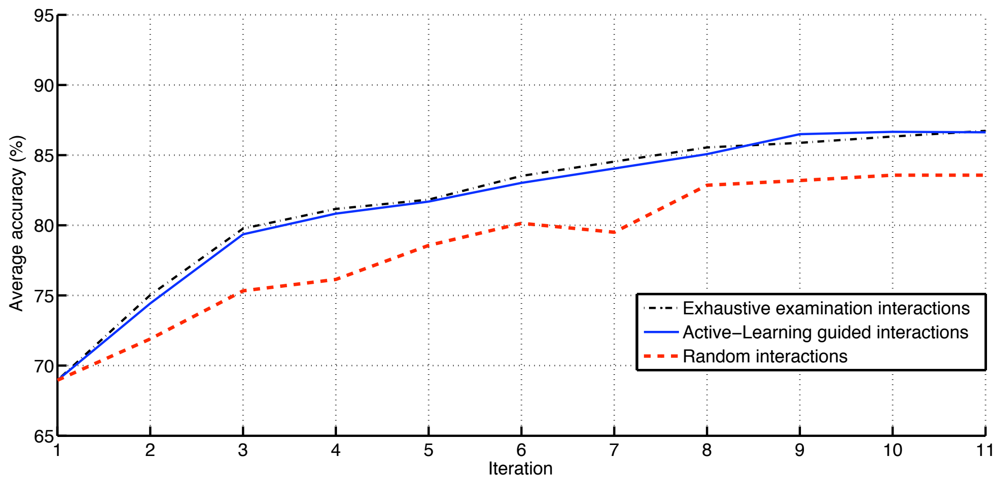


Figure 2.24: User study: The proposed active-learning algorithm not only out performs random interactions, but performs at par with exhaustive examination in significantly lower time (Section 2.6.4).

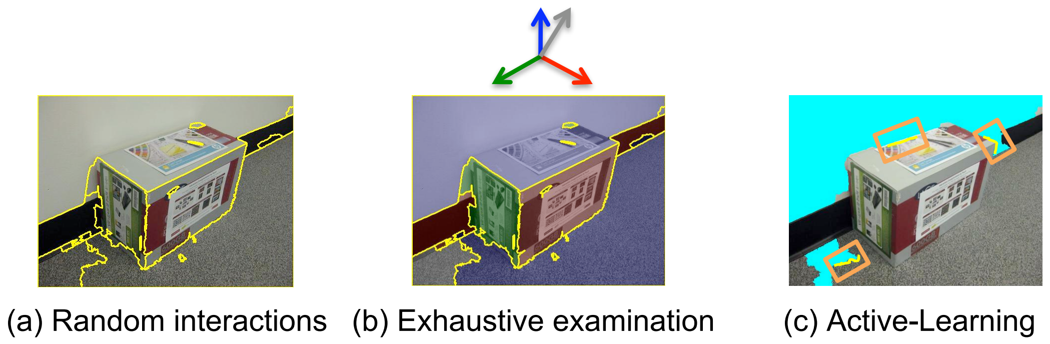


Figure 2.25: The three different user experiments conducted to evaluate the proposed algorithm (Section 2.6.4).

ment as shown in Figure 2.25b. The user was again instructed to provide three distinct interactions across any edge by observing the errors in the segmentations, with the normals guiding them towards erroneous regions. This leads to much better performance as seen by the *black* curve in Figure 2.24.

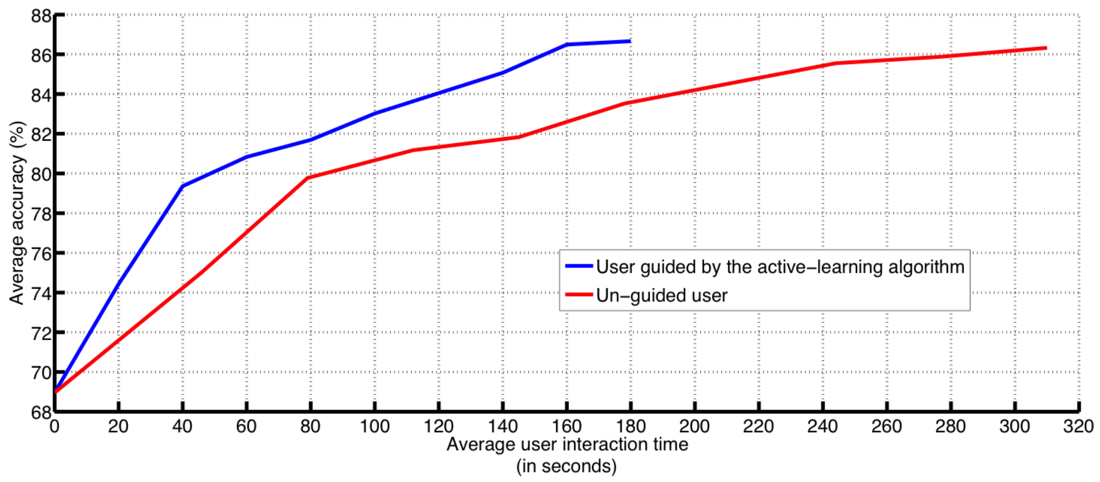


Figure 2.26: User study - time: The proposed active-learning algorithm achieves better performance and significantly faster (Section 2.6.4).

The last experiment evaluates the proposed active-learning algorithm. We show the user the uncertain regions detected by the algorithm in cyan. We highlight the uncertain edge in yellow, and draw three orange boxes to query the user for interactions, as shown in Figure 2.25c. The user was instructed to follow these orange boxes and provide interactions across the edges to provide support for the cyan regions. The *blue* curve in Figure 2.24 shows the performance. We observe that the active-learning algorithm performs much better than random interactions and performs at par with the exhaustive examination, indicating that the algorithm effectively guides the user towards relevant uncertain regions.

We compare the time taken by a user guided by the proposed active learning algorithm vs. an unguided user. We plot the average accuracy across the time taken in Figure 2.26. The proposed active-learning algorithm achieves better performance and significantly faster (almost 2x speed up).

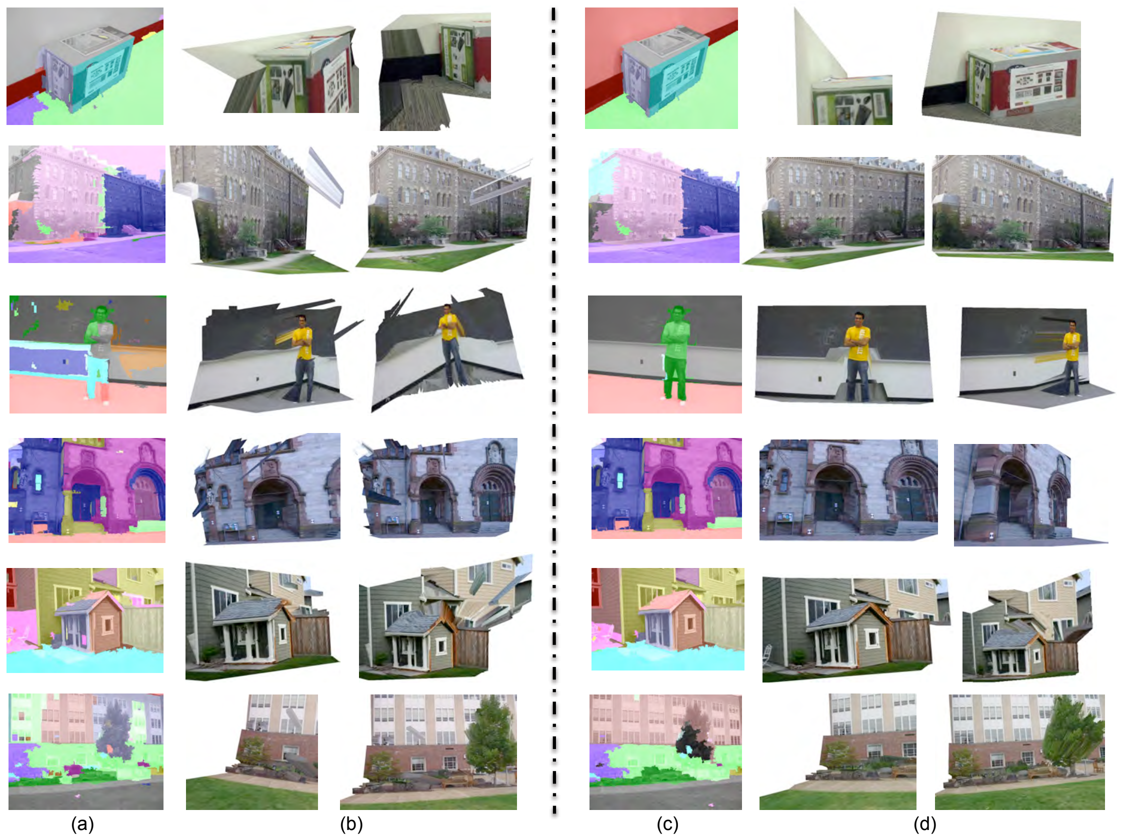


Figure 2.27: Qualitative results: (a) and (b) show the plane labeling and, novel views of 3D reconstruction from the automatic algorithm respectively; (c) and (d) shows the improved results using the active-learning algorithm respectively.

2.6.5 Qualitative analysis

In Figure 2.27, we show improvements in quality of the labeling and the 3D reconstructions as a result of incorporating the user interactions using the proposed algorithm[9].

Row 1 shows the improved reconstructions in presence of homogeneous surfaces like the wall and ground; Row 3 shows the improved result in case of an

[9]http://chenlab.ece.cornell.edu/projects/ActiveLearningFor3D

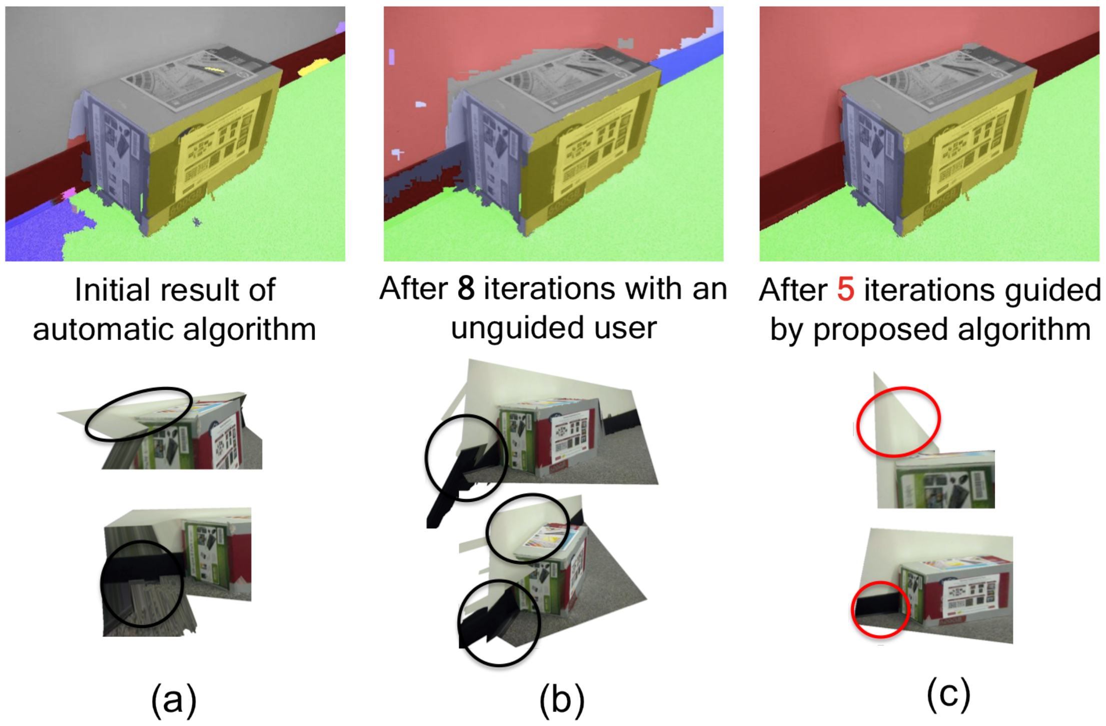

Initial result of automatic algorithm
After 8 iterations with an unguided user
After 5 iterations guided by proposed algorithm
(a)
(b)
(c)

Figure 2.28: Qualitative comparison: (a) The initial reconstruction of the scene, with errors shown in black ellipses; (b) The result after an unguided user provide interactions for 8 iterations, where errors still exist as shown in black ellipses; (c) In comparison, the user guided by the proposed active-learning algorithm achieves accurate reconstruction after only 5 iterations. Errors fixed are shown in red ellipses.

occluding object (planar approximation) and homogeneous background. Rows 4 and 5 show the output of the algorithm on public datasets used in prior work [85]. These are datasets in which the algorithm has enough cues to automatically reconstruct the scene and required minimal user interactions. These show that our automatic algorithm is not sub-optimal.

Relying on superpixels can hinder the performance in some cases. Note, for example, the error near the legs in row 3 due to a narrow superpixel leak.

Row 6 demonstrates a failure case of the algorithm. In this example, there was a superpixel that leaks from the top of the tree onto the building. Since the uncertain edge we show the user always follows the superpixel boundaries, superpixel leaks can affect the performance. In this case, when queried, the user would always mark the regions as co-planar, resulting in a part of the tree labeled as part of the building behind it. However, we note that the proposed algorithm still performs significantly better than the automatic algorithm.

Comparison. We qualitatively compare the performance of a user guided by the proposed active-learning algorithm with the performance of an unguided user in Figure 2.28. The initial reconstruction of the scene has errors that are partially fixed after 8 iterations by constraints provided by the unguided user. In comparison, the user guided by the proposed active-learning algorithm achieves a much more accurate reconstruction twice as fast, after only 5 iterations.

2.7 Summary

In this chapter, we have proposed a framework to put the user in the loop with the algorithm for image-based modeling of static scenes. Motivated by the recent success in discrete labeling formulation for image-based modeling we have leveraged the user input as node and edge constraints for the underlying Markov Random Field. We have considered algorithms where the user initiates the algorithm to indicate the object of interest, allowing for reconstructing non-planar objects and planar scenes. We proposed a novel active-learning algorithm for piecewise planar 3D reconstruction where the computational engine guides the user constraints. The algorithm tries to reconstruct the scene automatically, quantifies uncertainty, and asks the user to provide support for the most uncertain regions via simple and intuitive interactions (coplanar, not-coplanar, and not-connected scribbles). The algorithm incorporates these constraints to obtain better reconstructions, thus closing the loop on the interactive algorithm. We show through a user study and machine experiments that the proposed algorithm not only improves the reconstruction, but does so in significantly lower time than exhaustive examination by the user. In Chapter 4, we discuss some end user applications using these algorithms including object of interest 3D modeling on a mobile device and 3D printing an object of interest.

Acknowledgments: The authors thank Wen-Chao Chen for his help with the shape-from-silhouette algorithm and Anandram Sundar for the data annotation.

CHAPTER 3

IMAGE-BASED MODELING OF DYNAMIC SCENES

3.1 Introduction

While most prior work in image-based modeling, including algorithms in Chapter 2, address the case of a static scene captured by a dynamic camera, unconstrained image sequences captured in natural scenes often consist of dynamic or moving objects (such as people) in the scene that are captured either by a dynamic camera or even a static camera, which makes the task much harder. However, this finds itself a number of applications in video analysis such as video editing, image-based rendering, 3D modeling, scene understanding, etc.

In this chapter, we consider the task of recovering the depth of a scene in the presence of dynamic objects. More specifically we consider two scenarios. First, we consider a *dynamic* camera viewing a dynamic object. The user in the loop helps identify the regions corresponding to the moving object. The interaction between the dynamic object and the reconstructed static background helps recover the depth of the dynamic regions of the scene. Second, we consider a *static* camera capturing a dynamic object where we turn the table around and propose an algorithm where the dynamic object interacting with the scene helps decompose the static background scene into fronto-parallel depth layers. As the dynamic object (person, in our work) moves about the scene, it reveals occlusion cues with respect to the background that are sparse yet strong cues, which aids the task of image-based modeling.This forms the computational engine, that is also capable of incorporating user constraints. We put the user in the loop within this framework where the computational engine guides the user to provide additional pairwise depth ordering constraints to further improve the solution, thus closing the loop. An overview of the two algorithms is given below.

Dynamic camera scenario. In the first part of this chapter we consider the scenario of a dynamic object captured by a dynamic camera, we assume that we see enough static regions in the scene to recover the camera parameters for the frames of the video. Given the camera parameters, we first treat the video as that of a static scene and use a fronto-parallel plane sweep stereo algorithm to obtain a dense depth map of the scene. This is done by casting the problem as a discrete labeling problem formulated over an Markov Random Field (MRF) over pixels similar to dense stereo matching wherein the pixel disparities (inverse depth) are the discrete labels. Clearly, the depth estimates of the moving object would be in error. Given calibrated cameras there is a well defined relationship between the displacement of objects in the real world and the resulting image space displacement, as a function of the depth of the object. With the help of the user in the loop we identify the region of the image corresponding to the moving object in the scene. Using the depth of reconstructed static background occluded by the moving object and constraining the real world speed of the object, we obtain bounds on the depth of the object. We then incorporate these into the original plane sweep stereo framework to estimate a more realistic depth map. We show some results of our algorithm in Figure 3.1.

Static camera scenario. In the second part of the chapter we consider a time-series of images of a scene with moving objects captured from a static camera,

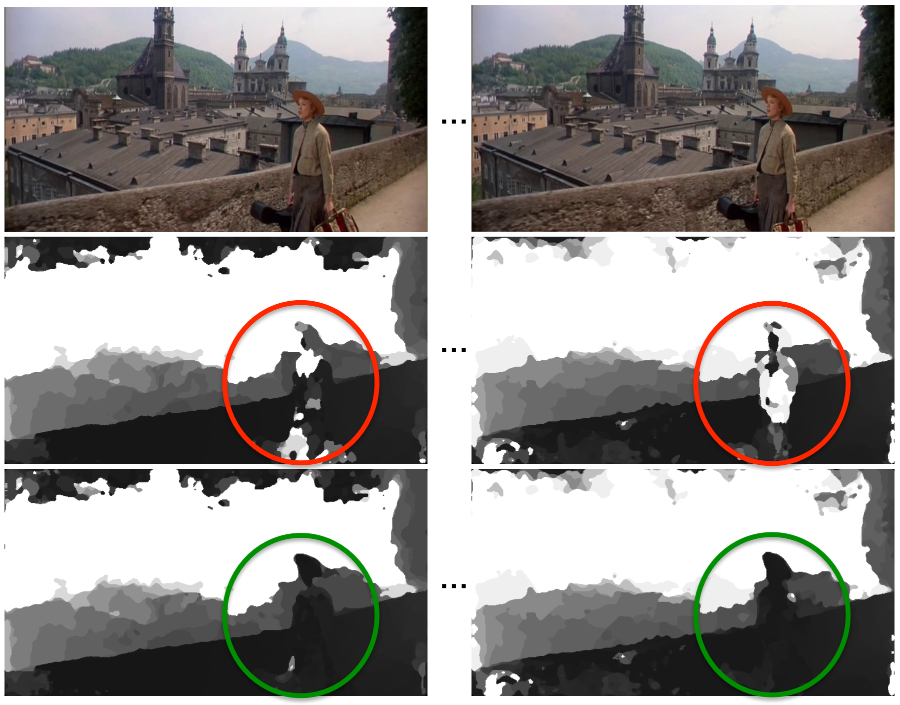
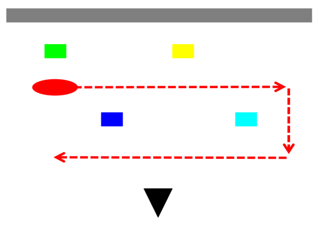


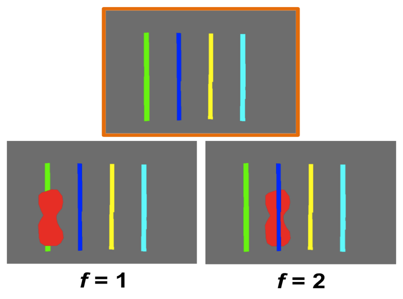
Figure 3.1: ROW 1 shows two frames from a video sequence from the movie *Sound of Music* where, the camera is translating to the left and the person is walking in the same direction. ROW 2 shows the initial depth maps estimated using plane sweep stereo (white is far, black is close). The depth of the moving object is over-estimated as shown in the red circles. ROW 3 shows the final depth maps inferred using the proposed approach after identifying and modeling the motion of the moving object. Note that more accurate depth map for the moving object shown in the green circles.

and our goal is to exploit occlusion cues revealed as the objects move through the scene to segment the scene into front-parallel depth layers. It is worth noting here that in contrast to the first scenario where the fronto-parallel depth map provides a physical estimate of the depth of each pixel, in this scenario the depth layers only provide a relative ordering of the regions of the scene. This task is a much harder problem than the first scenario due to the lack of multiple views of the scene. In the first part of the chapter, the image sequence captured from a dynamic camera allows one to leverage powerful stereo matching cues to recover the depth and occlusion information of the static background, which helped estimate the depth of the dynamic object. These cues are absent in case of a static camera. However, for single images, monocular cues help reveal useful depth information [33, 36, 40, 41, 65, 79, 94, 98]. Therefore, in the second part of the chapter we turn the table around and show how the dynamic object interacting with the static scene helps decompose the scene to fronto-parallel depth layers. We consider a set of images of a scene having moving objects captured from a static camera. As the object moves it is either occluded by or occludes a portion of the scene, consequently revealing sparse pairwise ordering relationships [11, 83] between regions of the scene. These pairwise cues while powerful, are very sparse, which makes our goal of extracting dense pixel-level depth layers a hard problem.

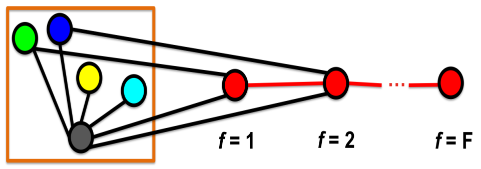
We cast the problem of depth-layer segmentation as a discrete labeling problem. We accumulate the pairwise ordering cues revealed as the object moves through the scene and use the monocular cues to propagate the sparse occlusion cues through the scene. We over-segment the static background scene (scene without any moving objects) and construct a region-level MRF with edges between adjacent regions. In each frame, we identify the pixels corresponding to the moving object and add a node corresponding to each moving object for every frame of the video. We add temporal edges between the corresponding moving object nodes across frames, which allows us to encode a smooth motion assumption for the moving object. As the object moves about the scene, we detect *motion occlusion events* and add edges between the static scene node and the corresponding moving object node, including long range edges between two static scene nodes to encode the pairwise depth-ordering or occlusion cues. The



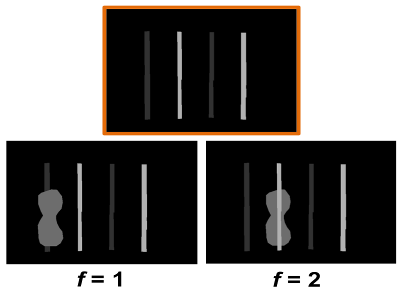
(a)



(b)



(c)



(d)

Figure 3.2: Overview. (a) Ground-truth top view, black triangle shows the camera looking up at a scene with the red moving object region following the path shown in the red arrow; (b) Shows the background scene in the orange box and two frames from the input sequence where the red object interacts with the background regions to reveal pairwise depth-ordering cues such as red occludes green, blue occludes red; (c) A graph constructed over the background regions is shown in the orange box. Each colored node corresponds to the respective colored region in (b). The red nodes correspond to the moving object with a node for every frame f in the input sequence ($\{1, 2, \ldots, F\}$). The black edges enforce the observed pairwise depth-ordering, for instance between the green-red nodes at $f = 1$, and blue-red nodes at $f = 2$. The red edges enforce a smooth motion model for the moving object; (d) Shows the inferred depth layers, white = near and black = far.

task now is to assign a fronto-parallel depth label to each region (node) of this spatio-temporal graph. An overview of our proposed formulation for a single moving object is shown in Figure 3.2, with the extension to handle multiple objects in Section 3.4.1. We show that this proposed algorithm achieves state-of-the-art results in depth-layer recovery. We treat this algorithm as the computational engine that is capable of incorporating additional constraints from the user. In Section 3.5 we show how the algorithm puts the user in the loop by guiding the user to provide additional pairwise depth ordering constraints, which is incorporated back into the computational engine and improves the solution.

Organization. The rest of this chapter is organized as follows: Section 3.2 discusses related work; Section 3.3 is the first part of the chapter that describes the proposed algorithm and results for the scenario of a dynamic scene captured from a dynamic camera; Section 3.4 is the second part of the chapter that describes the algorithm, experiments and results for the scenario of a dynamic scene captured from a static camera; Section 3.5 describes how we put the user in the loop guided by the algorithm to provide useful constraints; Finally, Section 3.6 summarizes the chapter.

3.2 Related work

A number of works have explored the task of image-based modeling from image sequences. In this section, we first discuss prior work that require the constraint of a dynamic or moving camera, followed by works that explore the static camera scenario.

3.2.1 Dynamic camera scenario

The task of estimating the depth of a scene given a sequence of images captured from multiple viewpoints has been very well established. A number of approaches have been proposed to tackle this well defined yet hard task. While enumerating all these is a mammoth task, we refer to some relevant works here that also include exhaustive summaries of the related works.

While one approach is to use the stereo matching algorithms on pairs of rectified frames of the video [82, 84], multiview stereo approaches [17, 46, 73] try to estimate the best depth estimate for each pixel using unstructured images. More recent large scale multiview stereo techniques allow for obtaining a dense point cloud or voxelized representation of the scene [27, 30, 88]. A line of work on depth-from-video by [75, 76] have explored fast, real time depth estimation from monocular videos. A recent line of work by [100] have shown some of the best results on depth from video. We note that some prior work model the scene by breaking it down into piecewise planar regions by hypothesizing global planes in the scene, which also provides a dense depth map of the scene [28, 56, 62, 85]. While these works have considered the task of depth from video, they focus on the regime of static scenes.

A few works have studied dynamic scenes captured simultaneously by multiple cameras to obtain a dynamic point cloud of the scene [32, 74]. Other related works in non-rigid structure-from-motion have not been cited here but the main focus of this line of work is to reconstruct key-points on non-rigid objects such as human face. More recently, [99] explored dynamic scenes with a focus on segmenting the dynamic object from the scene using a semi-supervised algorithm. Our goal in this work however, is to obtain an estimate of the fronto-parallel depth layers for each pixel, including the dynamic object. We exploit the interaction between the dynamic object and the static scene to help obtain a realistic depth estimate for the dynamic object.

3.2.2 Static camera scenario

Research in cognitive science has shown that humans rely on occlusion cues to obtain object boundaries and depth discontinuities even in the absence of strong image cues such as edges and lighting [47, 71] even in a single image. Recovering occlusion boundaries in a scene is a classic problem that has been a topic of wide interests. We broadly classify these works into learning-based approaches and approaches that purely rely on motion occlusion cues revealed by the moving object.

Learning-based approaches. Prior works have explored learning based approaches for estimating the depth of the scene [33, 36, 40, 43, 65, 79, 81, 94, 98] and estimating depth ordering [41, 45] from a single image for 3D scene understanding. Recent work has shown that we can also use the objects (clutter) in the scene to aid better depth estimation of the scene [34, 37] using affordances.

Moving beyond single images to image sequences, [24] showed that the pose of people interacting with a cluttered room can be used to obtain functional regions and recover a coarse 3D geometry of the room. Our work is complementary to this work and in particular is agnostic to priors about the type of moving object and the type of scene (indoor or outdoor). In other words, we do not require that the moving object is a human. We relate back to prior research in cognitive science that show that occlusion cues we observe are agnostic to any prior about the object. We use these sparse, yet strong occlusion cues revealed by the moving object to aid the dense depth layer segmentation of the scene.

Depth layers from motion occlusion. We work with a single static camera image sequence that precludes us from using algorithms for multiview occlusion reasoning using the moving object [32]. We focus on segmenting a scene captured by a single static camera into depth layers using occlusion cues revealed by the moving objects. Our work is inspired by the work of [11] and [83] who use the pairwise occlusion cues to "push" and "pop" the regions of the scene affected by the moving object to obtain depth layers at each frame. A limitation of these works is that they reason only about the portion of the scene the object interacts with, leaving behind huge portions of the scene at an unknown depth layer. In addition, since the interaction with each region is treated independently it leads to excessive fragmentation of the scene as we will see in Section 3.4.2. This fragmentation can be partially avoided [83] by making the (possibly over-restrictive) strong assumption that the moving object stays at a constant depth. Our model includes a more reasonable model of object motion. We revisit depth layers from occlusions and address limitations of prior work via a unified framework that leverages sparse depth-ordering cues revealed by the moving object and gracefully propagates them to the whole scene guided by cues from the image priors. In addition, we reason about the depth layer of the moving object within the same framework with a realistic object motion model.

We note that it is important to first identify the which scenario an input video sequence belongs to in order to leverage the right image-based modeling algorithm. For example, identifying that the video is that of a dynamic (moving) camera indicates that we can leverage multiview stereo cues. In this thesis, we achieve this as a pre-processing step that temporally segments an input video to clips based on scene and camera motion. More details about this is available in [57]. In the following sections, we will describe algorithms that implicitly leverage the moving object to aid the task of image-baed modeling. We first describe the proposed algorithm and results for the scenario of a dynamic scene captured by a *dynamic* camera followed by the proposed algorithm and results in case of a dynamic scene captured by a *static* camera.

3.3 Dynamic camera capturing a dynamic scene

We first consider a dynamic camera capturing a dynamic object (moving person in our work) where the static background reconstructed using a standard multi-view stereo algorithm helps recover the depth of the dynamic regions of the scene.

3.3.1 Algorithm

We describe our algorithm in detail in this section. Given a video of a dynamic scene captured using a dynamic camera, we extract the frames of the video sampled at 30fps. We assume that enough static regions of the scene are observed and recover the camera parameters for the frames of the video using structure-from-motion (SfM) [88].

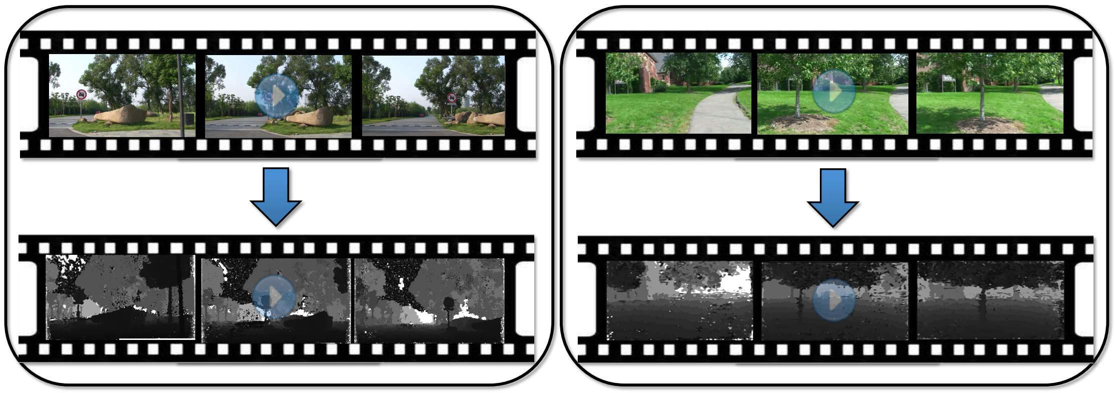


Figure 3.3: Results from the plane sweep stereo implementation. Our simplified implementation works well on a well structured smooth video (LEFT) as well as a user captured amateur video (RIGHT). Details in Section 3.3.1. (black = near, white = far)

Plane sweep stereo

Motivated by the success of the plane sweep stereo algorithm [17, 75, 100], we base our depth from video algorithm on the same framework. We use fronto-parallel planes discretizing the 3D space in inverse depth. In particular, we obtain the minimum ($\frac{1}{D_{max}}$) and maximum inverse depth ($\frac{1}{D_{min}}$) by projecting the 3D points recovered from SfM onto the optical axis. We divide this range into equally spaced bins thus obtaining the 3D planes to perform the plane sweep stereo. The goal now is to build the pixel-level cost cube by evaluating an error metric between the original image and warped image for each of the discrete plane and obtain the minimum cost depth at each pixel. A number of error metric have been explored in the past such as the variance of the color at each pixel, sum of absolute differences, sum of squared differences, etc.

We use a sliding window of 10 frames and normalized cross correlation (NCC) between the reference image and the image obtained by warping the neighboring view onto the 3D plane, as a metric to find the best depth. The result on videos of static scenes are shown in Figure 3.3. However, scene irregularities such as homogenous surfaces, thin structures and specular surfaces result in a very noisy cost cube. We qualitatively compare algorithmic choices to filter this noisy data in Figure 3.4. As we observe, the best depth map was obtained by filtering the cost cube by using guided filtering [35]. Using the original image to guide the filtering of the cost cube, we observe that the dominant edges are preserved resulting in a clean output by using a simple *argmax* [1] operation on the filtered cost cube.

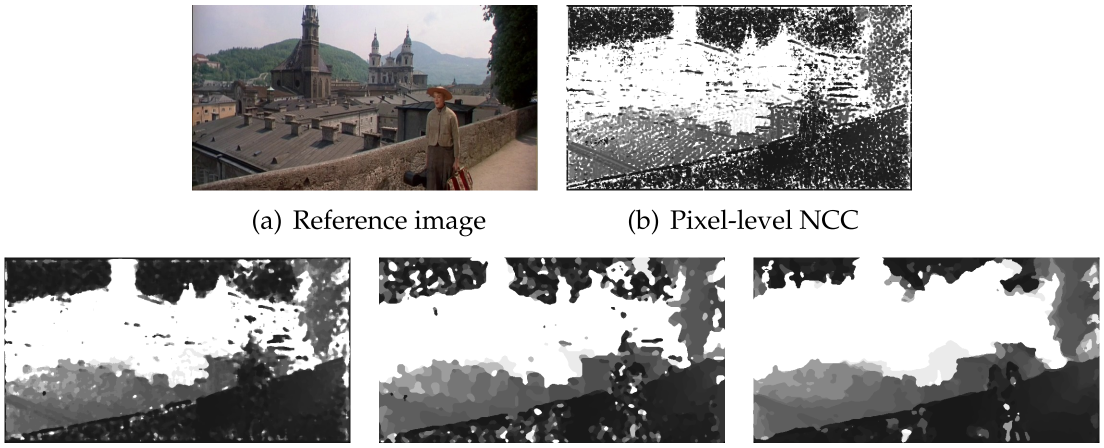
Note that, the inferred depth map can be improved further by using ideas from plane fitting [99] and piecewise planar stereo [28, 56, 62, 85] at the cost of computational complexity, which have not been explored here.

Handling a dynamic scene

We now consider a video of a dynamic scene. We start off by running the plane sweep stereo algorithm assuming the video is that of a static scene. Note that the depth of the region corresponding to the dynamic object would be incorrectly estimated as observed in ROW 2 of Figure 3.1. Intuitively, one can obtain a better depth estimate by identifying the spatial region corresponding to the dynamic object and factoring the real world motion of the dynamic object into the plane sweep stereo framework. While some works attempt to segment out the dynamic object (with supervision), estimating the real world motion of the object is non-trivial.

In this work, we put the user into the loop to segment out the dynamic object

[1] Note that the larger the value, the better since we are using NCC.


(a) Reference image (b) Pixel-level NCC

(c) Median filtering (d) Pixel graphcut (e) Guided filtering

Figure 3.4: Comparison of algorithmic choices (White is far, black is close for the depth maps). (a) Sample image from a video; (b) Depth map using pixel-level NCC score; (c) Cleaner depth map by median filtering result (a); (d) A pixel-level labeling using graph cuts produces a better result but, noisy; (e) The best result was obtained by guided filtering the cost cube, guided by edges in the original image.

via a simple user input in the form of a bounding box around the moving object. To minimize the user effort we use the algorithm described in [57] to segment video to clips. The frame level labels help identify the frames where a moving object passes through the scene. The user draws a bounding box around the moving object in this selected frame. We then track the bounding box over the successive frames using the mean optical flow of the spatial region within the bounding box. This is illustrated in Figure 3.5. We note that we can also use other interactive approaches to perform co-segmentation across the frames [6, 99]. We however focus on the second non-trivial task of modeling real world object motion in the next section.

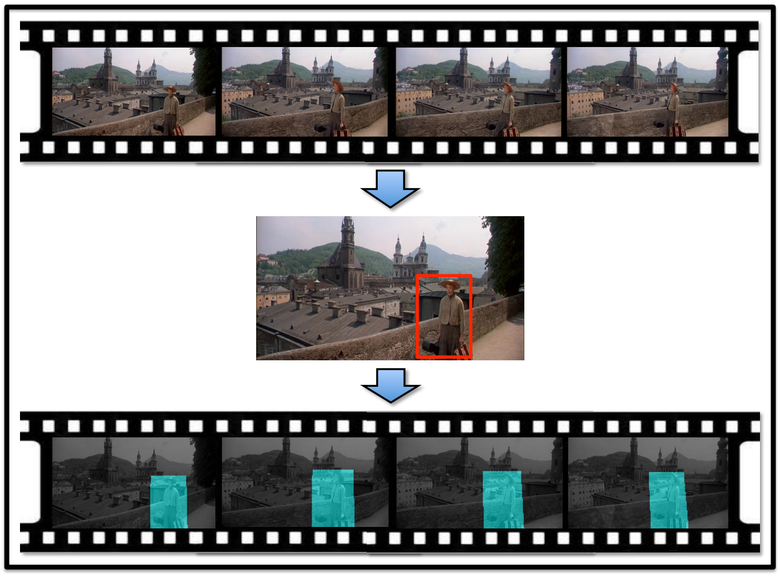


Figure 3.5: Subset of frames that show the semi-supervised co-segmentation of the moving object in the scene.

Modeling the motion of the dynamic object

Estimating the depth of a moving object is a hard and ambiguous task. We estimate the depth using plane sweep stereo by making some assumptions about the object motion in the real world.

Let O_{ref} and O_{nei} be the camera centers of the reference and neighboring frames respectively from a video sampled at r frames per second (Figure 3.6). Let P_A be the position of the dynamic object as seen from O_{nei} but moves to position P' when seen in O_{ref}. Thus, the depth of the object, D is incorrectly estimated as D'. Let ΔY be the real world distance travelled by the object between the frames i.e. the distance traveled by the object traveling at a speed v in time $\Delta t = \frac{1}{r}$. Given the focal length (f), the similarity between the red and green triangles gives the relationship between the disparity Δd in the image space and

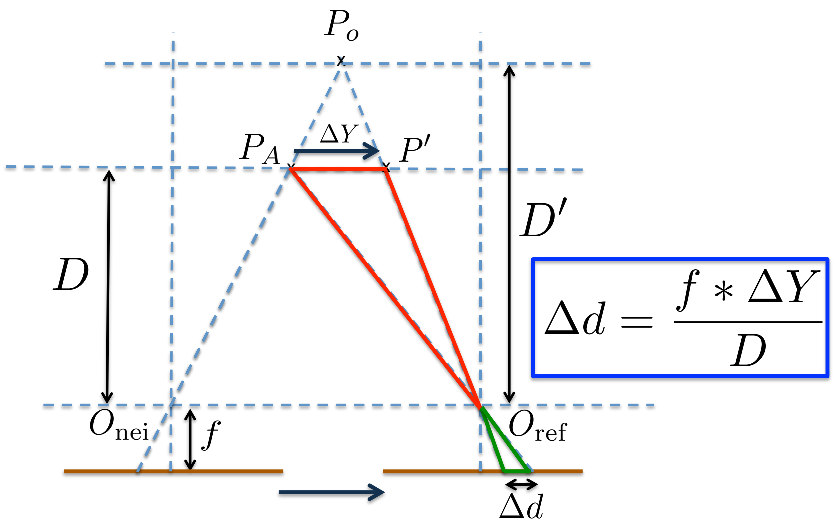


Figure 3.6: Modeling the object motion: The relationship in the blue box results from the similarity between the red and green triangles. More details in Section 3.3.1.

the real world motion as,

$$\Delta d = \frac{f * \Delta Y}{D} = \frac{f * v * \Delta t}{D} = \frac{f * v}{r * D} \tag{3.1}$$

While on one hand the depth of the moving object is unknown, the speed at which it is traveling is also unknown. A large object moving fast and far away from the camera, can appear very similar to a smaller object moving slower and located close to the camera. We see from Eqn 3.1 that this is an under-constrained problem since any pair of the depth (D) and speed (v) can result in the same image projection. In order to relax this ambiguity, we make some assumptions about the object motion in real world. We consider videos with people moving and bound the speed of dynamic object (v) to be within v'. We set $v' = 2$ meters per second in our work (human walking speed). Since the video is sampled at r frames per second, this results in a bound on the image space displacement (Δd) as a function of the depth (D) as follows,

$$\Delta d \leq \frac{f * v'}{r} \frac{1}{D} \tag{3.2}$$

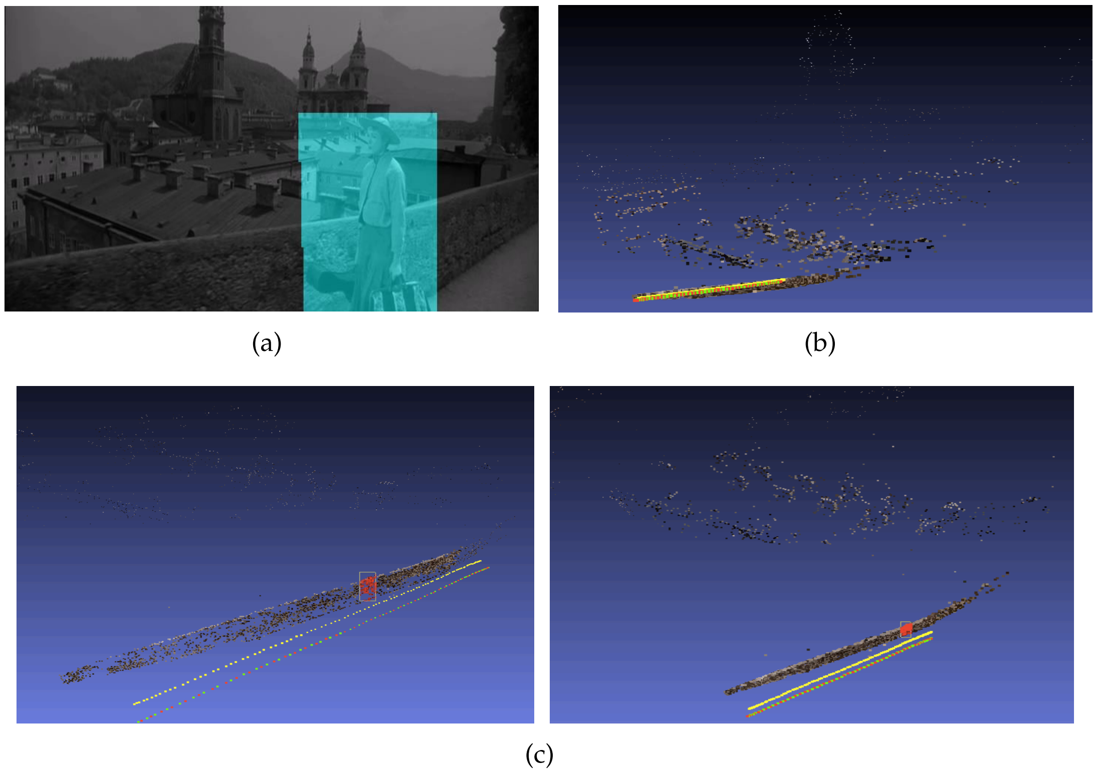


Figure 3.7: (a) A frame from the input sequence with the moving object bounding box shown in cyan. (b) The SfM point cloud reconstructed using [88] with the cameras shown in yellow. (c) The 3D points in the SfM point cloud that project onto the cyan region are illustrated in red color, which helps define the depth bound for the moving object.

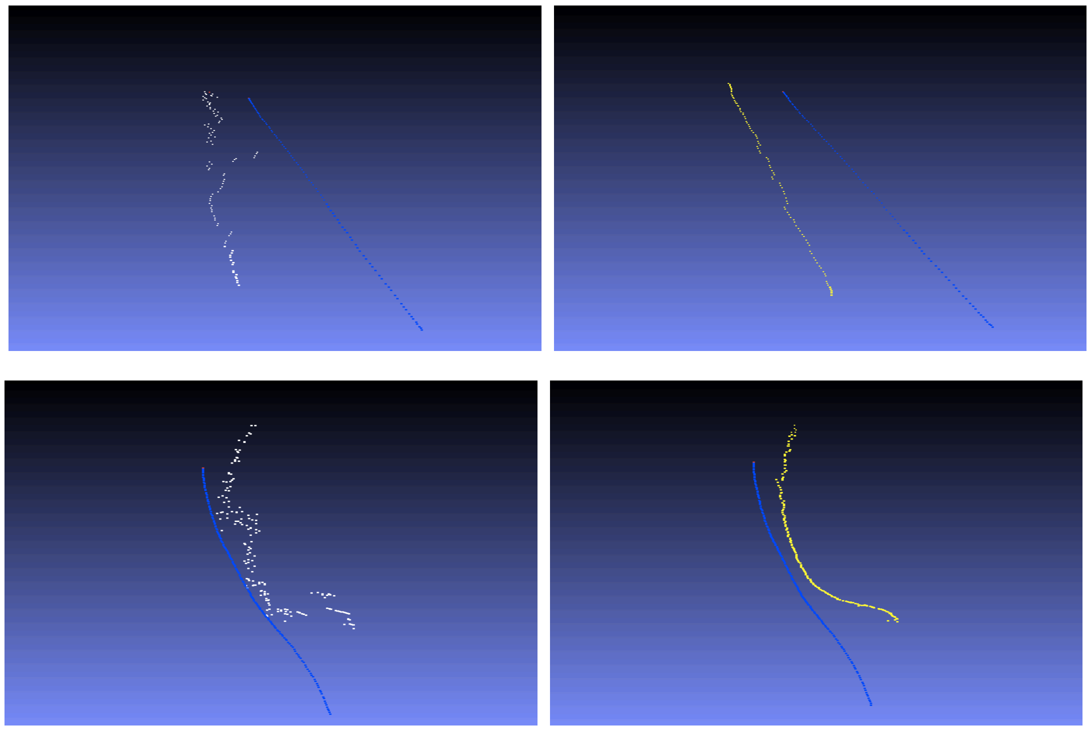
While constraining the speed of the object provides a low bound on the depth of the object, a more useful bound is revealed as the moving object occludes the static scene. We add an additional bound on the actual depth of the object using the region of the scene occluded by the object. We consider the *minimum* depth of the 3D points from SfM that lies in the occluded region or projects onto the segmented out dynamic object region and use it to upper-bound the depth (D) since the object has to be in-front-of the occluded region. This subset of points that the moving object occludes is illustrated in Figure 3.7.

We then incorporate this into the plane sweep algorithm. While evaluating the cost for a plane hypothesis for the reference frame (i.e. a plane that falls within the depth upper-bound), we use the depth of the hypothesized plane to obtain the bound on displacement (Δd) using Eqn 3.2. We allow for the segmented dynamic object region in the reference frame to undergo an in-plane shift of a maximum of Δd in either direction and evaluate the best score (NCC) for this region. Intuitively, at a depth within the bounds, the in-plane shift will allow the segmented region to obtain a better score than before, resulting in a better depth estimate. The result of incorporating the object motion into the depth estimation is evident in Figure 3.10 and Figure 3.11. We note that while the estimated depth is more accurate than before, the solution is not unique and is subject to how tight the bounds are. We refine the depth estimate by fitting a smooth trajectory over the object in the next section.

Enforcing a smooth trajectory

At the end of the previous step we obtain an estimate of the depth of the moving object for each frame however, the solution is not unique and depends on how tight the depth bounds are. We use these estimated depths as our initial estimate for the depth of the moving object enforcing that the object travels in a smooth trajectory.

Consider a frame of the video sequence where the dense depth of the dynamic scene has been estimated as described in the previous section. We estimate the centroid of the bounding box corresponding to the moving object and consider a small window of pixels around the centroid (10×10). The mean depth for each pixel within this window is used as the current estimate of the depth



(a) Before fit (b) After fit

Figure 3.8: The blue points correspond to the camera centers in all the figures. Each row shows the result of a video sequence showing the effect of using the trajectory fitting, (a) The white points correspond to the initial noisy estimate of the position of the moving object, (b) The yellow points correspond the position of the moving object after enforcing that it travels along a smooth motion trajectory.

of the moving object. We obtain the depth of the moving object for each frame, which forms our initial set of observations as shown for two video sequences in Figure 3.8(a). We observe from the trajectory of white points that the initial estimates of the depth can be noisy. Using the intuition that the dynamic object moving about the scene travels in a smooth trajectory, we fit the observed positions of the object using a regression with a polynomial kernel.

In detail, our observations for each frame are the camera center, the 2D position that corresponds to the centroid of the the object and the depth of the dynamic object. The first two parameters define a ray in 3D space along which the moving object lies, which represent our features (X). The depth of the dynamic object is used to obtain the depth of the object along the ray using projective geometry, which serves as the value we are regressing over (Y). We use a polynomial kernel to learn the regression model for the object trajectory, as defined below.

$$K_d = (1 + X^T X)^d \tag{3.3}$$

$$\begin{aligned} \theta^* &= \underset{\theta}{\operatorname{argmin}}\, L_d(\theta) \\ &= \underset{\theta}{\operatorname{argmin}}\, \|K_d^T \theta^* - Y\|_2 + \lambda \|X\theta\|_1 \end{aligned} \tag{3.4}$$

$$d^* = \underset{d}{\operatorname{argmin}} \left(L_d(\theta^*) + \alpha log_2 \left(kd + \binom{k}{d} + 1 \right) \right) \tag{3.5}$$

Here, K_d in Eqn 3.3 indicates the polynomial kernel of degree d and k is the number of dimensions of X, which in out case is 6 (3D position of the camera and the 3D ray through the object center). θ represents the regression coefficients with θ^* corresponding to the optimal parameters for a particular degree d resulting in an fit error $L_d(\theta)$. Note that we use L-1 sparsity in Eqn 3.4 to avoid over-fitting. We enforce a smooth trajectory by constraining the degree of the polynomial kernel. Eqn 3.5 represents the MDL (Rissanen) criterion [77] that is used to pick the right degree for the polynomial. We evaluate the MDL criterion for $d = \{1, 2, 3\}$ and pick the degree (d^*) with the best fit. The resulting trajectory for the two video sequences is shown in Figure 3.8(b). The trajectory now provides a better estimate of the depth of the moving object based on the smooth motion. This new depth is in-turn used to update the depth values of all values within the bounding box corresponding to the moving object. We note in Figure 3.8(b) that upon using the MDL criterion the algorithm chooses $d^* = 1$ for

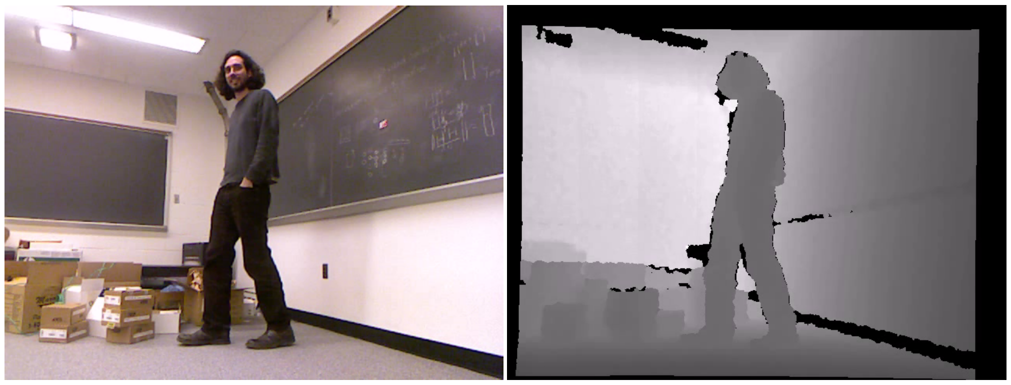


Figure 3.9: An example of the aligned kinect RGBD data used for the quantitative results. The RGB image is on the left and the corresponding aligned depth map is on the right.

the first video resulting in the linear fit, and selects $d^* = 3$ for the second video resulting in a better fit as seen from the trajectory of yellow points.

3.3.2 Experiments and results

We discuss the quantitative and qualitative results using the proposed algorithm in this section.

Quantitative results

We capture five indoor scene videos using a Kinect by moving it on a dolly, and extract the aligned ground truth depth map for each frame. A person walked across the scene in two videos and an RC car was driven across the scene in three videos. An example is shown in Figure 3.9. On an average each video had about 50 frames. We show in Figure 3.10 the average RMS error in the estimated dense depth maps, computed over the spatial region corresponding

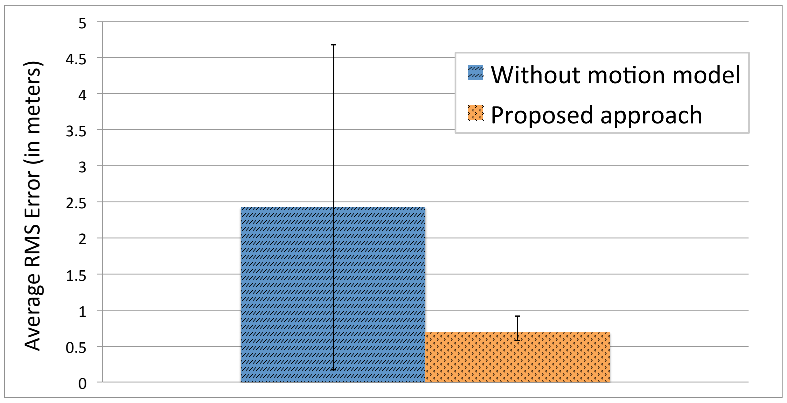


Figure 3.10: Quantitative analysis: We show the average RMS error in estimated depth measure using kinect data. Note that the proposed approach gives significant improvement.

to the dynamic object, averaged over all the frames. We note that the proposed approach significantly reduces the error, in addition it achieves a much lower error bar resulting in a much stable solution.

Qualitative results

We qualitatively evaluate the performance of the algorithm on amateur video sequences captured by a user and video clips extracted from the movie *Sound of Music*. The videos sequences used in the experiments were released as part of the publicly available, Cornell Video2Clips Dataset[2], which provides the video clips corresponding to our setting of dynamic camera capturing a dynamic scene [57]. We show some of the results in Figure 3.11. Note the inaccurate depth estimate of the dynamic object in ROW 2 of each video, and the significantly improved depth map seen in ROW 3. We also use the depth map to synthesize a stereo pair using a baseline of 77cm. The resulting anaglyph images shown on ROW 4 gives the right perception of depth. We note here that the

[2]http://chenlab.ece.cornell.edu/projects/Video2Clips/

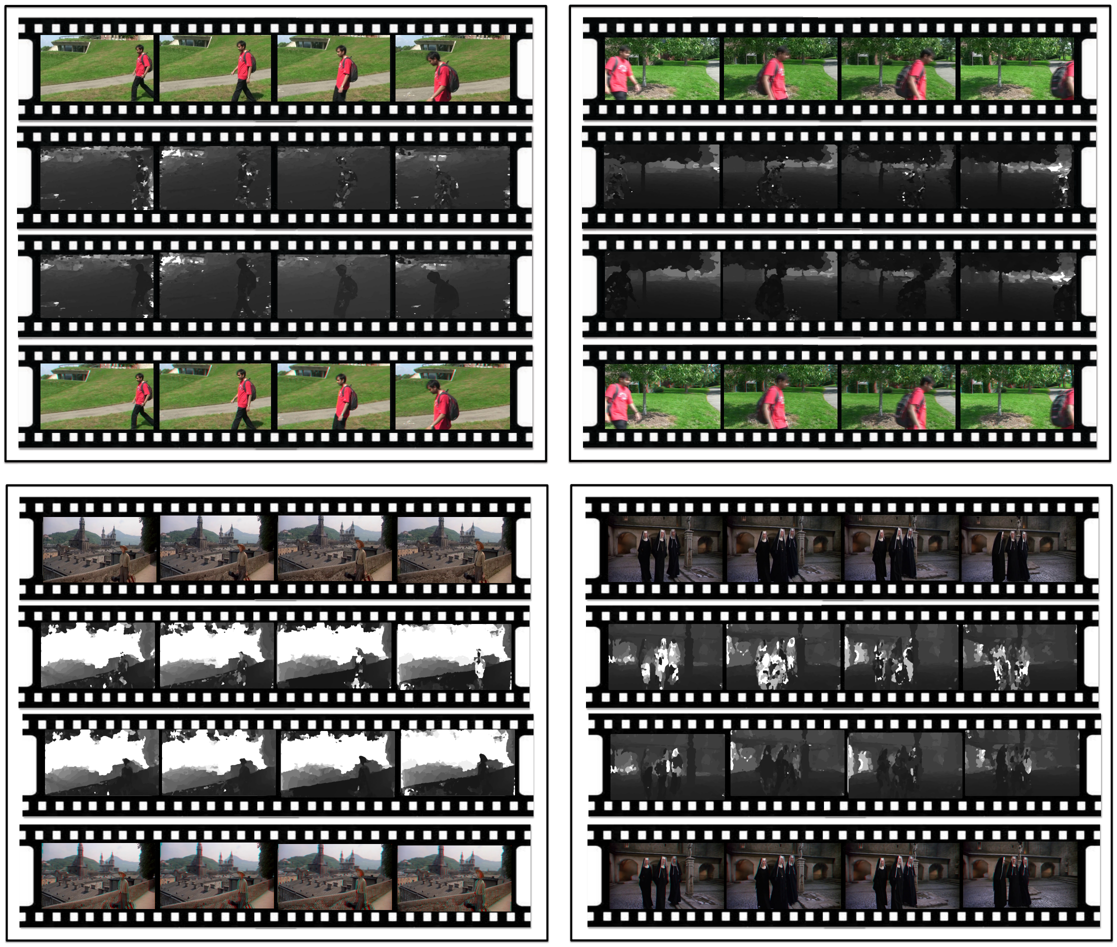


Figure 3.11: Each black box shows results on a video sequence. ROW 1 shows four frames from a video with a moving object. ROW 2 shows the initial depth maps using plane sweep stereo (white is far, black is close). Note that the depth of the moving object is inaccurately estimated. ROW 3 shows the final depth maps inferred using the proposed approach after identifying and modeling the motion of the moving object. Note the more accurate depth map for the moving object in each case. ROW 4 shows anaglyphs obtained by synthesizing the left image of the stereo pair using the original image as the right image and the recovered depth map (Requires red - cyan glasses).

result using the proposed algorithm may not be the accurate depth due to ambiguous continuous space of possibilities mentioned in Section 3.3.1 however, the proposed algorithm that incorporates the object motion gives significant improvement as evident in Figure 3.10.

3.4 Static camera capturing a dynamic scene

In Section 3.3 we showed that the static background reconstructed helps recover the depth of the dynamic object. We now turn the table around and consider a static camera capturing a dynamic object (moving person in our work). We show through the proposed algorithm and results that the dynamic object physically interacting with the scene aids decomposing the scene into fronto-parallel depth layers. We note that in Section 3.3 we obtained a physically representative fronto-parallel depth estimate for each pixel using the multiview stereo cues revealed by the moving camera. In contrast, in this section, we recover ordered depth layers that do not provide a physical depth estimate but focus on a fronto-parallel depth layering based on the occlusion cues.

3.4.1 Algorithm

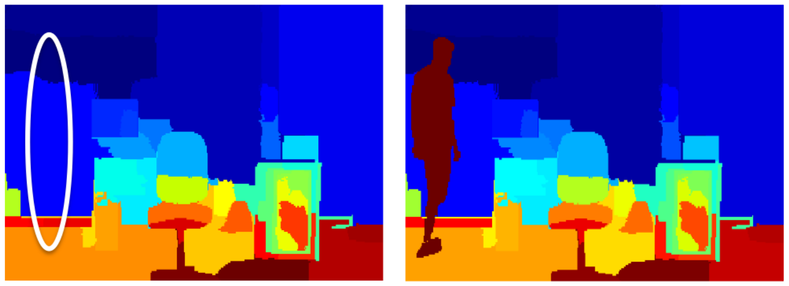
We formulate the task of segmenting the scene into depth layers as a discrete labeling problem. In this section, we first describe our formulation as applied to a scene with a single moving object and then extend the same framework to handle multiple moving objects in the scene.

Spatio-temporal graph

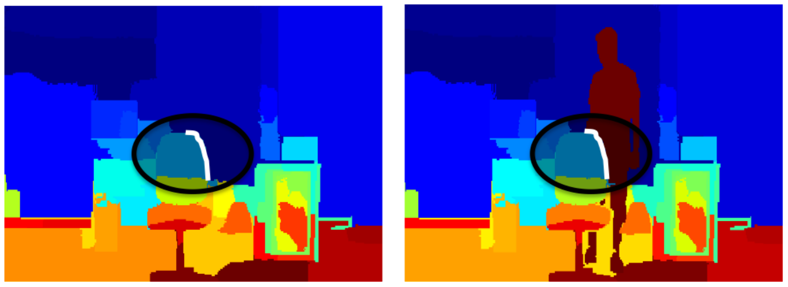
Background scene segmentation. We refer to the scene without any moving objects as the background scene. We use a calibration stage to obtain a clean background image without any moving objects. In the absence of the calibration stage we take advantage of the static camera scenario and obtain an estimate of the background image as the median image over the video. Given the background image we obtain an over-segmentation using mean shift segmentation [18] to give us about 300 superpixels. We treat this segmentation as a stencil of background superpixels that applies to each frame of the video.

Moving object segmentation. Given the superpixel stencil for the background scene, we update this superpixel map for every frame by identifying the pixels corresponding to the moving object via background subtraction. We model the appearance of the background using a per-pixel Gaussian distribution (A_p) centered at the mean color (RGB space) of the pixel across the whole video. Given A_p, for every frame we estimate the likelihood for each pixel belonging to the background. We label pixels with background likelihood above 90% as confident background pixels and below 10% likelihood as confident moving object pixels. Using these as confident initial seeds, we learn an appearance model for the background (BG) and the moving object (FG). The moving object segmentation is obtained using iterative graph-cuts [9, 10, 51] updating the BG/FG color models with each iteration similar to GrabCut [78]. Figure 3.12 shows examples of the moving object segmentation overlaid on the background segmentation.

After this stage, we have the background scene superpixel map and the moving object segmentation for each frame. A region-level MRF is constructed over the background scene superpixels where each superpixel is a node with an edge



(a) Object in-front-of background scene region



(b) Object behind background scene region

Figure 3.12: Pairwise depth-ordering cues. Left image shows the background scene segmentation and the right image shows an intermediate frame segmentation with the moving object segment. (a) A region in the background is covered by the moving object (white ellipse) indicating that the moving object occludes the background region; (b) Observing that the boundary corresponding to the background region (white pixels in black ellipse) does not change when the moving object comes in contact with it reveals that the moving object is occluded by the background region. It also reveals new relationships via transitivity; the chair occludes the object and at the same instant the object occludes regions on the wall; therefore the chair occludes the regions on the wall.

to adjacent superpixels. We add a node corresponding to the moving object for every frame of the video and add temporal edges connecting the moving object nodes on adjacent frames. This graph is illustrated in Figure 3.2(c).

Pairwise depth-ordering cues

The object moving through the scene is either occluded by or occludes portions of the scene. We refer to these as *motion occlusion events*. In our superpixel representation of the scene, we accumulate the pairwise cues using a matrix we call Occlusion Matrix (O) where, $O_{i,j} \in \{-1, 0, +1\}$ indicates the relationship between superpixel i and superpixel j i.e., $\{i$ occluded by j, no cue, and i occludes $j\}$, respectively. O is a skew-symmetric matrix i.e., $O_{i,j} = -O_{j,i}$. The matrix is updated at every frame of the video using detected motion occlusion events or using learnt monocular cues in absence of occlusion cues.

Motion occlusion cues. Low-level cues revealed by the moving object in the scene serve as sparse, yet strong pairwise depth-ordering cues. We work with the abstract superpixel representation of each frame and use cues similar to prior work [11] to obtain pairwise relationship between the moving object segment and the superpixel it interacts with. The cues are intuitive, given a background region the moving object is interacting with, we use the moving object pixels and the boundary pixels of the background region to infer whether the object moved in-front-of this region or behind this region, respectively, as illustrated in Figure 3.12.

We update the corresponding entry of the occlusion matrix with $O_{i,j}$ as +1 to indicate that superpixel i occludes superpixel j and set $O_{j,i}$ to -1. In addition to the pairwise depth-ordering cues between the moving object and the superpixel it is interacting with, we also enforce transitivity while updating the matrix. If the object is occluded by a region of the background scene and is simultaneously occluding several regions of the background scene, via transitivity it establishes a pairwise relationship between the occluding background region and each of the other background regions as shown in Figure 3.12(b). More formally, if m refers to the moving object segment simultaneously involved in motion occlusion events with superpixels k and l then, $O_{k,m} = +1$ and $O_{l,m} = -1$, implies $O_{k,l} = +1$. This provides a strong depth-ordering cue between k and l. In addition, since k and l are not constrained to be adjacent superpixels, long-range edges between non-adjacent superpixels are also a result.

Monocular cues. We use monocular cues to provide evidence about occlusions for the other regions of the scene. Given the superpixel map for each frame, we use the work of [41] that uses learnt priors to determine which of two adjacent superpixels occludes the other. For each frame, we first update the occlusion matrix using the motion occlusion cues where available and update the matrix for all the other spatially adjacent superpixels using the monocular cues. We do not enforce transitivity here since the monocular cues are not as reliable as motion occlusion cues. The occlusion matrix serves as the observations for modulating the terms of the energy function described below.

Energy minimization problem

The goal given the sparse pairwise depth-ordering constraints is to obtain dense depth-layers. One approach is a greedy algorithm where the whole scene starts at layer-0 and with every pairwise depth-ordering constraint regions of the scene are "pushed" and "popped" [11] to obtain the final labeling. [41] use a graph with boundaries between superpixels are nodes connected to adjacent boundaries to encourage continuity and closure. [45] use image junctions as nodes to obtain a globally consistent depth ordering using a minimum spanning tree. In this work, we use superpixels as nodes in the graph. This allows us to directly obtain the depth-layer labeling, and also incorporate long range edges between nodes.

We formulate depth layer segmentation as a discrete labeling problem where every superpixel is assigned a depth label $\{1, 2, \ldots, L\}$ where L is some pre-defined yet large set of discrete labels[3]. The labels are depth-ordered from closer to the camera moving away i.e. $\{1 < 2 < \cdots < L\}$. We formulate this multi-label segmentation problem as an energy minimization problem over the spatio-temporal graph obtained in the previous stage. The graph is a collection of $n + F$ nodes, where n nodes correspond to the background scene and F nodes correspond to the moving object with one node for the moving object for each of the F frames of the video. Our goal is to obtain a labeling $\mathcal{X} = \{X_1, X_2, \ldots, X_{n+F}\}$. We define an energy function over the graph as follows:

$$E(\mathcal{X}) = \sum_{i \in 1,\ldots,n+F} E_i(X_i) + \sum_{(i,j)\in\mathcal{N}_\mathcal{S}} E^S_{ij}(X_i, X_j) + \sum_{(i,j)\in\mathcal{N}_\mathcal{T}} E^T_{ij}(X_i, X_j) \tag{3.6}$$

where $E_i(X_i)$ is the unary term indicating the cost of assigning a depth layer to a node, $E^S_{ij}(X_i, X_j)$ is the spatial occlusion pairwise term updated by the motion occlusion cues and the monocular cues between interacting regions ($\mathcal{N}_\mathcal{S}$), and $E^T_{ij}(X_i, X_j)$ is the temporal pairwise term updated by the object motion model between the temporal edges ($\mathcal{N}_\mathcal{T}$) .

Unary term (E_i). The unary term measures the cost of assigning a particular depth label to a node. We use a uniform likelihood across all labels since a node does not prefer one label over another. However, we note that the moving object can move between two background regions that are in adjacent depth layers. To address this, we ensure that the background regions only take odd or modulo-2

[3]In all our experiments we set $L = 40$. An over-estimate of L allows for enough layers for the background scene. Increasing L beyond 40 did not affect performance but added to the computational complexity.

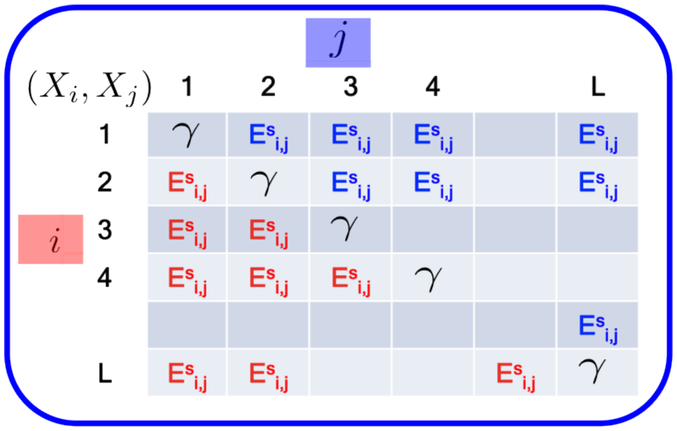


Figure 3.13: Spatial occlusion pairwise term E^S_{ij}. If i occludes j, the pairwise term encourages that i takes a depth label closer (lower) than j via a large penalty for the red terms and zero penalty for blue terms. See Section 3.4.1 and Equation 3.7 for details.

labels, which makes an intermediate layer between two depth layers available for the moving object. We do so using hard constraints where the background region pays infinite penalty for choosing an even numbered depth label.

Spatial occlusion pairwise term (E^S_{ij}). The spatial occlusion pairwise term encodes the pairwise depth-ordering observations we accumulate within the occlusion matrix. Consider two regions (nodes) i and j, using the cues we discussed in Section 3.4.1 let us suppose we know that region i is occludes region j i.e. $O_{i,j} = +1$. Intuitively, the pairwise term for the edge between i and j must encourage i to take a depth label that is smaller than (closer) j. To accomplish this, our pairwise term has the form of an lower triangular matrix where a large cost is incurred for region i taking a depth label larger than region j. We make this term contrast sensitive using the score from a coplanar classifier $(1.0 - \delta^f_{i,j})$ that indicates how likely i and j are coplanar using the relative region-level features similar to [56] for each frame f. More formally,

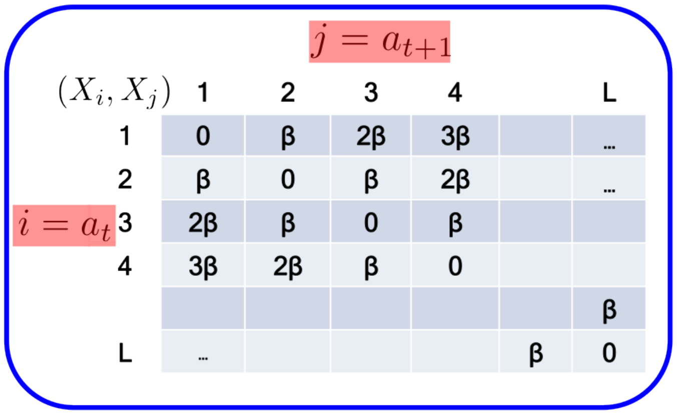
$$E_{ij}^{S,f}(X_i, X_j) = \begin{cases} -log\left(c_{ij}^f \times \frac{1+O_{i,j}^f+\epsilon}{2}\right) & \forall X_i < X_j \\ \gamma & X_i = X_j \\ -log\left(c_{ij}^f \times \frac{1+O_{j,i}^f+\epsilon}{2}\right) & \forall X_i > X_j \end{cases}$$

$$E_{ij}^{S}(X_i, X_j) = \sum_{f \in F} \left(E_{ij}^{S,f}(X_i, X_j) \times \exp\left(-\delta_{i,j}^f\right)\right) \tag{3.7}$$

where, $E_{ij}^{S,f}$ is the pairwise term for frame f, $O_{i,j}^f$ is the occlusion relationship between region i and j in frame f of the image sequence. c_{ij}^f is the confidence of the pairwise occlusion relationship for frame f. We set this value to 1.0 for edges that include the moving object and use the occlusion strength [41] as the confidence score for all other edges. The summation over pairwise terms over all frames helps capture the evidence between two nodes over the whole sequence. The factor γ is a bias that keeps the solution away from the trivial solution of a single depth layer for the whole scene[4]. ϵ is a small value to maintain numerical precision. The form of the spatial occlusion pairwise term is illustrated in Figure 3.13.

Temporal motion pairwise term (E_{ij}^T). The temporal motion pairwise term penalizes label disagreement between the moving object node across frames and encourages a smooth motion for the moving object, illustrated in Figure 3.14. The pairwise penalty is similar to the standard Pott's model, except with an increasing penalty (β) as we go away from the diagonal[5]. Given the depth label of the moving object in one frame, smooth motion is encouraged by making the node pay a lower cost to switch to nearby depth labels but larger penalty for more drastic changes in the depth label. Intuitively, this pairwise term encour-

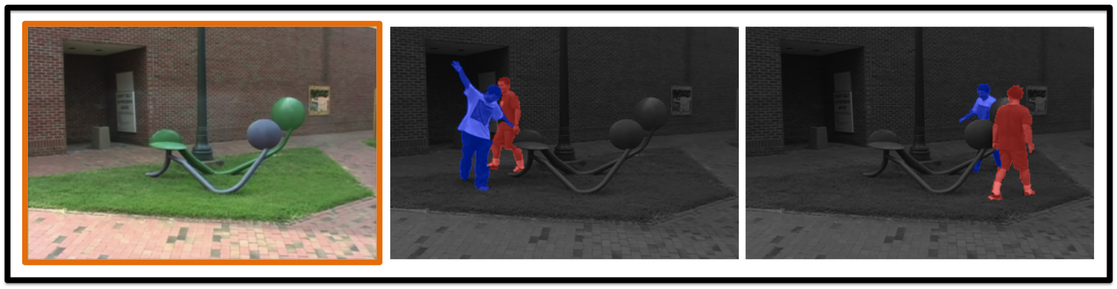
[4]We set the bias $\gamma = -log(0.5)$ for our experiments after parameter sweeping (Section 3.4.2).
[5]$\beta = -log(0.5)$ for our experiments.


$j = a_{t+1}$

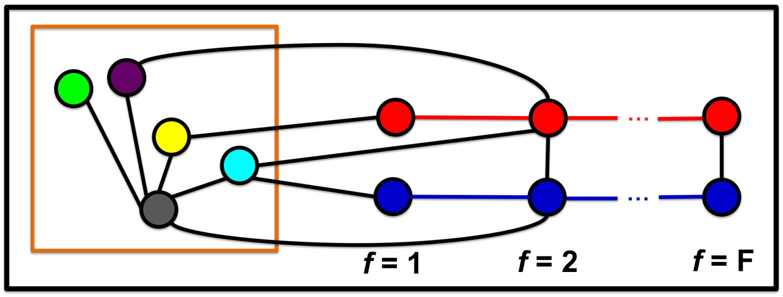
(X_i, X_j)	1	2	3	4		L
1	0	β	2β	3β		...
2	β	0	β	2β		...
$i = a_t$ 3	2β	β	0	β		
4	3β	2β	β	0		
						β
L	...				β	0

Figure 3.14: Temporal motion pairwise term E^T_{ij}. The penalty (β) increases as we go away from the diagonal encouraging a smooth motion of the object across depth layers. See Section 3.4.1.

ages a fronto-parallel motion of the moving object i.e. encourage it to take the same depth label however, consider in particular scenarios where the moving object passes in-front-of a static region and then moves behind it. The algorithm implicitly has a cue that the moving object has to switch depth label to satisfy the two pairwise cues since the static region is at a fixed depth label. Such observations encourage the moving object to smoothly change the depth label via the temporal motion pairwise term. Physically, this motion model assumes that the object does not abruptly change in depth as it moves through the scene. We quantitatively evaluate the impact of using the temporal motion pairwise term in Section 3.4.2. Note that an additional cue that serves as a useful depth ordering cue to help model the pairwise term is the size of the moving object. We can make an assumption that the larger the object the closer it is to the camera, however, this cue was very unreliable due to inaccurate moving object segmentation and is therefore not used in this work.



(a)



(b)

Figure 3.15: Multiple moving objects. (a) The background scene is shown in the orange bounding box. The two moving object segments for intermediate frames are overlaid in red and blue; (b) The spatio-temporal graph constructed. The spatial graph corresponding to the background scene is shown within the orange bounding box and the two nodes for each frame corresponding to the moving objects are shown using the red and blue nodes. See Section 3.4.1.

Handling multiple moving objects

Here, we extend the formulation (single moving object) to handle multiple moving objects. Consider the example in Figure 3.15(a) with the region corresponding to the two moving objects in the scene shown in blue and red overlay. In case of k moving objects, we add k nodes (a node for each moving object) for each frame of the video. The resulting spatio-temporal graph for the example is shown in Figure 3.15(b). We have an edge between the moving objects as shown in the frame, $f = 2$ when the objects cross path. We obtain their pairwise depth-ordering using the cue that when the two objects are in contact, the taller object, i.e., the object with a larger bounding box height, occludes the smaller one. This assumes that the moving objects are the same size in real world; however, more sophisticated classifiers could be used. We modify the unary term to reflect that there are multiple moving objects. In the single object case we used a modulo-2 representation of the depth labels that put hard constraints on the background regions to take only alternate depth labels allowing for the moving object to lie between two background region layers. In case of k moving objects in the scene we extend this to a modulo-(k+1) representation that allows the k objects to lie between two adjacent background region layers. Given this graph, the definition of the energy function is the same as Section 3.4.1.

Inference

In our energy function, each energy term by itself is weak. For instance, the unary term does not provide an affinity of a node towards a particular label but restricts the labels the background regions can take; the spatial occlusion pairwise term bounds the possible labels the adjacent node can take based on the label of the current node. However, the combination of these terms is powerful. The intuition behind the goal of inference is to find a depth labeling that satisfies as many pairwise interaction terms and motion model terms as possible. We perform inference using sequential tree-reweighted max-product message passing (TRW-S) [52]. The algorithm scales linearly with the number of frames and quadratically in the worst case with the number of superpixels (i.e., fully connected graph).

3.4.2 Experiments and results

In this section, we discuss the dataset, the evaluation metric, followed by our quantitative and qualitative results.

Dataset

Our first dataset (SET-A) contains 24 videos with a single moving object. 18 videos are from the publicly available multiview video dataset by [32] that include a person moving through the scene captured from multiple viewpoints. Each of these multiview videos serves as a test video for our scenario. The dataset has 6 additional videos with two clips from the movie 'Sound of Music'.

Our second dataset (SET-B) contains 9 videos from the publicly available video dataset by [32] with two people walking in the scene. In the single object scenario, the moving object segmentation and correspondence across frames was achieved using background subtraction, however, this is not trivial for multiple objects. While we believe that there is scope to leverage prior work on multiple object tracking to achieve this task automatically, in this work we provide correspondence and manually segment the moving objects on 30 frames for each video using GrabCut [78]. An example is shown in Figure 3.15(a).

We manually obtain a pixel-level ground-truth depth layer segmentation for each of the static scenes using the depth-layer annotation tool by [41] and then map it to the static scene superpixel map by labeling all the pixels within a superpixel with the dominant label. An example is shown in Figure 3.16. We make these manually labeled depth layers and the manual multiple object segmentation across frames publicly available, which are also useful in evaluating

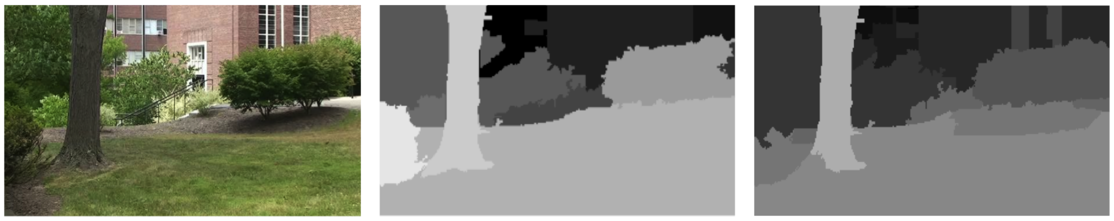

(a) Static scene (b) Ground-truth (c) Estimated

Figure 3.16: (a) Static scene, (b) manually labeled ground-truth depth layers for quantitative analysis and (c) estimated depth layers using our algorithm. white = near, black = far.

tasks such as multiple object co-segmentation[6]. It is worth pointing out that the ground surface has no clear 'ground-truth'. In particular, our instruction to the ground-truth annotator was that any object that stands on the ground surface occludes the ground surface as a basis for evaluations. Preprocessing to perform ground segmentation could be an alternate approach to add more semantics to the framework. However, this does not change the problem formulation or the improvement we obtain over the state-of-art.

Evaluation

We evaluate the performance of the algorithm as the accuracy of pairwise ordering between the regions of the background scene. Using the background superpixel map we translate the ground-truth depth layers into the ground-truth occlusion matrix (O^{gt}), which gives the pairwise depth-ordering between any pair of superpixels. Let the final occlusion matrix from the algorithm be O'. Given the two matrices, we evaluate the performance of the pairwise ordering between the superpixels by accumulating concordant pairs, discordant pairs,

[6]http://chenlab.ece.cornell.edu/projects/DepthLayersMRF/

and compute the accuracy as[7],

$$\begin{aligned}
\text{Concordant pair } (i, j) &: O^{gt}_{i,j} = O'_{i,j} \\
\text{Discordant pair } (i, j) &: O^{gt}_{i,j} \neq O'_{i,j} \\
\text{Accuracy} &= \frac{\#\text{Concordant pairs}}{\#\text{Concordant pairs} + \#\text{Discordant pairs}}
\end{aligned} \tag{3.8}$$

The accuracy measure evaluates the performance of the algorithm over all pairs of regions in the scene. This gives an average score of 25.2% across our dataset even when the whole scene is given a single depth layer. We obtain a metric focused only on the occlusion boundaries by computing the precision and recall of the algorithm evaluating the fraction of recovered occlusion boundaries that are the true occlusion boundaries and the fraction of the true occlusion boundaries recovered by the algorithm, respectively.

Our problem is similar to that of inferring a rank ordered list of entries. We use two standard metrics to evaluate the performance of pairwise ordering, Kendall tau correlation coefficient (τ) and Kendall tau distance (τ_d) [49, 72]. In particular, we use the variant of Kendall's tau (Tau-b) that accounts for ties within the list, because pairs of superpixels can take the same depth label. τ measures the similarity between orderings and has range $[-1, +1]$, the higher the coefficient the better. τ_d is a measure of the distance between the orderings and has range $[0, 1]$, the lower the better.

Parameter sweep (γ). We first sweep the parameter γ in Equation 3.7 to evaluate the impact on the performance of the proposed algorithm. γ acts as a bias that helps rely on the spatial occlusion cues only when the algorithm is confident of the pairwise occlusion cue. We observe that $\gamma = -log(0.5) \approx 0.7$ achieves the best performance as seen in the accuracy metric in Figure 3.17,

[7] $\#x$ = number of x

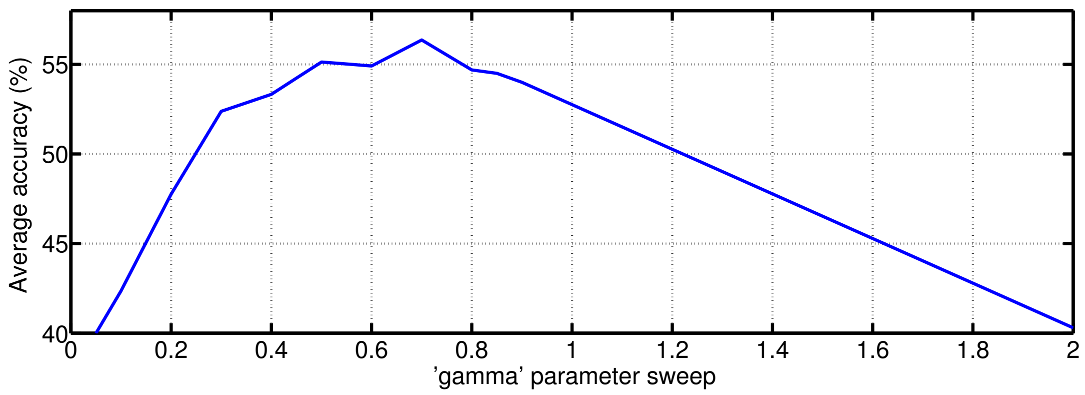


Figure 3.17: Average accuracy across the scenes in SET-A, which shows the effect of the parameter γ in Equation 3.7. This parameter acts as a bias that helps depend on the spatial occlusion cues only when the algorithm is confident of the pairwise occlusion cue. γ is set to $-log(0.5) \approx 0.7$ for all the experiments.

which is fixed for all our experiments.

Quantitative results

We quantitatively evaluate the performance of our algorithm, comparing with several baselines. First, we compare with prior works that use only motion occlusion cues [11] or only monocular cues [41]. We then evaluate the performance of a naïve combination of the motion occlusion and monocular cues using a greedy algorithm similar to [11]. We first use all the motion occlusion cues to obtain the pairwise depth-ordering and then use the monocular cues to update the pairwise orderings only for adjacent superpixels that do not yet have a pairwise ordering constraint to obtain the final depth labeling. This baseline does not enforce a global consistency in combining the cues. In our full algorithm, we use a spatio-temporal graph to combine the two cues and enforce global consistency. In addition to evaluating the performance of the proposed algorithm (full), we evaluate variants of the proposed algorithm where we use all the pairwise occlusion cues as hard constraints i.e. by setting c_{ij}^{f} and δ_{ij}^{f} in spatial occlusion pairwise term (Equation 3.7), to 1.0 and a variant where we drop the temporal links that enforce a smooth object motion i.e. setting the parameter β in the temporal motion pairwise term to a very large value.

Tables 3.1 and 3.2 summarize the results. We see that using motion occlusion cues alone (ROW-1) performs the worst, for two main reasons - fragmentation of the scene due to the greedy algorithm [11] and sparsity of the cues i.e., it only reasons about regions the object interacts with. Monocular cues (ROW-2) do better because it reasons about the whole scene and encourages global consistency with a graph model [41]. While the naïve combination of the cues (ROW-3) performs better than only motion occlusion cues, it performs poorly in comparison to using only monocular cues, due to fragmentation and lack of global consistency.

ROW-4 shows the performance of the proposed framework when we use all the occlusion cues as hard constraints. We see from the results that the soft constraints we use are useful to avoid making hard decisions. Even without temporal links (ROW-5), we outperform baselines in each metric. This clearly indicates that our improvements are not based on tracking per se, and shows that our algorithm is applicable to scenarios like time-lapse sequences. Finally, in both test sets, our full proposed approach (ROW-6) gives an additional boost in performance and significantly outperforms the other algorithms in each metric. Across the datasets, the proposed algorithm achieved the best performance in 19 out of 24 videos in SET-A and 8 out of 9 videos in SET-B, matching the performance of using only monocular cues for the other videos.

Single moving object (SET-A)	Accuracy (%)	Precision (%)	Recall (%)	F-measure [0.0, 1.0]
Only motion cues [11]	38.2±6.5	40.0±11.1	38.3±8.4	0.39±0.09
Only monocular cues [41]	49.0±10.6	55.1±12.5	50.0±12.8	0.52±0.12
Naïve combination	42.1±7.3	46.3±10.4	38.8±9.7	0.42±0.10
Proposed (Hard occlusions)	**43.5±6.5**	**52.4±10.9**	**42.5±10.3**	**0.47±0.10**
Proposed (No temporal)	**54.9±10.6**	**60.8±12.2**	**55.4±11.8**	**0.58±0.13**
Proposed (Full)	**56.5±9.3**	**62.6±12.3**	**57.5±10.9**	**0.61±0.11**

Single moving object (SET-A)	Kendall tau coefficient [-1.0, 1.0]	Kendall tau distance [0.0, 1.0]
Only motion cues [11]	+0.01±0.13	0.40±0.09
Only monocular cues [41]	+0.15±0.20	0.33±0.10
Naïve combination	+0.03±0.15	0.36±0.10
Proposed (Hard occlusions)	**+0.10±0.11**	**0.36±0.06**
Proposed (No temporal)	**+0.33±0.19**	**0.26±0.09**
Proposed (Full)	**+0.36±0.18**	**0.24±0.08**

Table 3.1: Quantitative results and comparisons for the single moving object scenario (SET-A). Each measure is averaged across the videos in the dataset. The table on top reports the performance using the metrics of Accuracy, Precision, Recall and F-score. While the one at the bottom reports the Kendal-Tau measures. In both the tables, ROW-1 shows the performance when we use only the motion occlusion cues [11]; ROW-2 shows the performance when we use only the learnt monocular cues [41]; ROW-3 shows the performance of a naïve combination of the motion occlusion and monocular cues; ROW-4 shows the performance of the proposed framework when we use all the occlusion cues as hard constraints i.e. by setting c_{ij}^{f} and δ_{ij}^{f} in Equation 3.7, to 1.0; ROW-5 shows the performance of the proposed approach but without the temporal links enforcing the object motion model, which is equivalent to a very large value for β in the temporal motion pairwise term; Finally ROW-6 shows the performance of the full proposed approach that combines the motion occlusion and monocular cues into one framework. Note that the error bars show the standard deviation of the performance *across the dataset*. In summary, the proposed algorithm (in green) outperforms the other algorithms in each metric in 19 out of 24 scenes.

Multiple moving objects (SET-B)	Accuracy (%)	Precision (%)	Recall (%)	F-measure [0.0, 1.0]
Only motion cues [11]	40.5±6.8	43.4±11.9	40.7±6.9	0.42±0.09
Only monocular cues [41]	50.9±10.1	55.6±12.1	50.7±10.2	0.53±0.11
Naïve combination	44.6±7.4	54.2±7.1	43.0±7.0	0.48±0.09
Proposed (Hard occlusions)	**45.3±7.5**	**55.7±6.9**	**44.5±7.5**	**0.49±0.06**
Proposed (No temporal)	**56.3±9.8**	**60.3±9.5**	**55.5±10.5**	**0.58±0.10**
Proposed (Full)	**58.2±10.0**	**62.4±10.2**	**59.1±10.4**	**0.60±0.10**

Multiple moving objects (SET-B)	Kendall tau coefficient [-1.0, 1.0]	Kendall tau distance [0.0, 1.0]
Only motion cues [11]	+0.02±0.07	0.37±0.05
Only monocular cues [41]	+0.20±0.10	0.35±0.12
Naïve combination	+0.06±0.09	0.36±0.05
Proposed (Hard occlusions)	**+0.16±0.12**	**0.36±0.05**
Proposed (No temporal)	**+0.30±0.14**	**0.26±0.07**
Proposed (Full)	**+0.33±0.10**	**0.24±0.10**

Table 3.2: Quantitative results and comparisons for the multiple moving objects scenario (SET-B). The table on top reports the performance using the metrics of Accuracy, Precision, Recall and F-score. While the one at the bottom reports the Kendal-Tau measures. The rows are the same algorithms as Table 3.1. Note that the error bars show the standard deviation of the performance *across the dataset*. The proposed approach (in green) outperforms the other algorithms in each metric in 8 out of 9 scenes.

Qualitative results

We show qualitative results obtained using only motion occlusion cues, only monocular cues and the proposed algorithm in Figure 3.18. Figure 3.18(b) shows the ground-truth depth layers for each scene. We first observe the drawback of using only motion occlusion cues in Figure 3.18(c), such as the fragmentation in the labeling due to the greedy algorithm and the unknown layer for pixels untouched by the moving object (in blue). Using the monocular cues results in a better dense labeling but errors due to the image-based features exist,

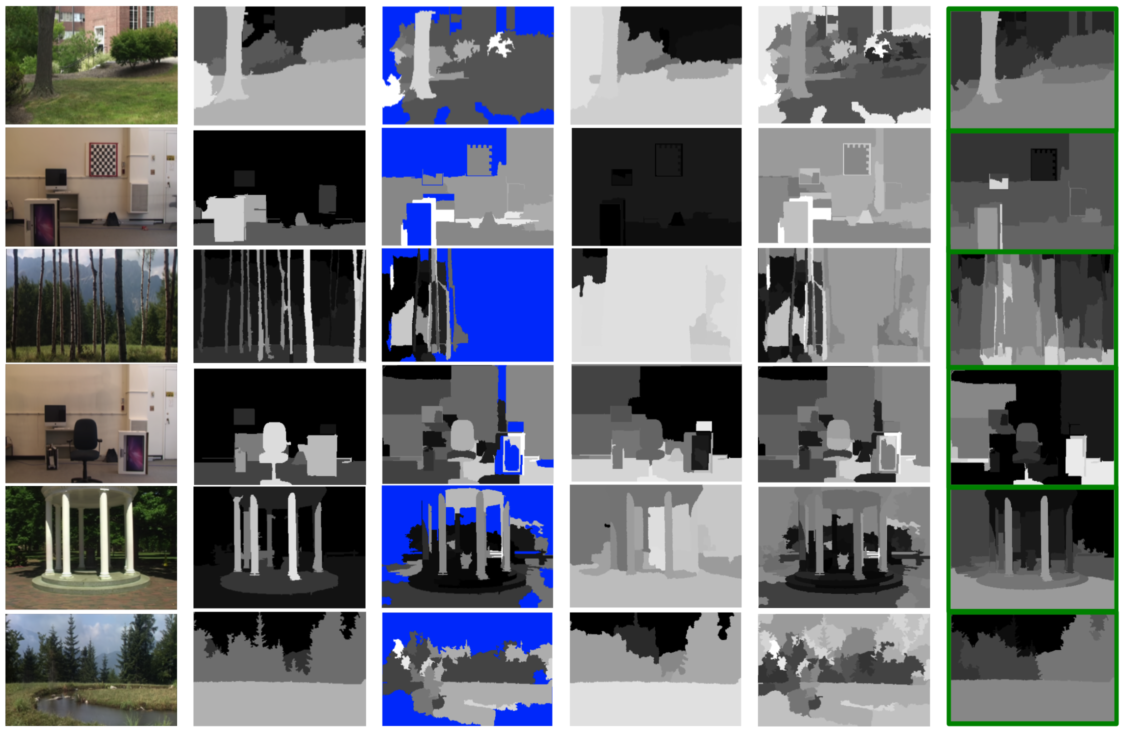

(a) Static scene (b) Ground truth layers (c) Only motion cues [11], pixels with no cues are colored blue (d) Only monocular cues [41] (e) Naïve combination (f) Proposed algorithm

Figure 3.18: More qualitative results and comparisons. For all the depth-layers, white = near, black = far. Discussion in Section 3.4.2.

Figure 3.18(d). In contrast, the proposed algorithm achieves a better labeling of the scene as seen in Figure 3.18(e). In particular, we see that occlusion cues captured in the motion occlusion cues but missing in the monocular cues such as the tree occluding the static scene in ROW-1, the chair and box occluding the static scene in ROW-2, 4, the pillars in ROW-5 are all carried forward to improve the result using the proposed algorithm. Errors due to pairwise cues unseen by the moving object but present in the monocular cues are carried forward to the final result (ROW-3, 6). In ROW-6 the proposed algorithm favors smoothness instead of the excessive fragmentation found from the motion occlusion cues.

The sensitive stage of the algorithm is foreground segmentation (background subtraction) especially in case of scene irregularities such as specular surfaces and thin structures (computer monitor in ROW-2 Figure 3.18), which can lead to errors in the sparse occlusion cues. In our work, we handle this using the MRF over all the regions and incorporate temporal dependency via smooth motion of the moving object. We make a joint solution given all the (soft) occlusion cues, reducing the errors in comparison with prior work that make hard decisions using occlusion cues.

3.5 Putting the user in the loop

The object moving about the scene helps reveal pairwise depth ordering constraints. We have presented an algorithm in Section 3.4 to combine these sparse, yet strong motion occlusion cues revealed by moving objects in a static scene along with monocular cues in a unified framework. The results discussed in the previous section shows that the proposed approach improves the performance of prior approaches, and also handles multiple objects moving in the scene.

The scenario discussed thus far in this chapter explores the user implicitly guiding the image-based modeling algorithm. However, the amount of information obtained is constrained by the observed motion of the object (including the space the object could potentially move) and the monocular cues observed in the scene. In order to boost the performance of the algorithm we need to observe additional pairwise depth-ordering constraints. We do so by going back to what we learnt from Chapter 2 and letting the user guides the algorithm by providing additional pairwise constraints. This gives rise to two questions to be

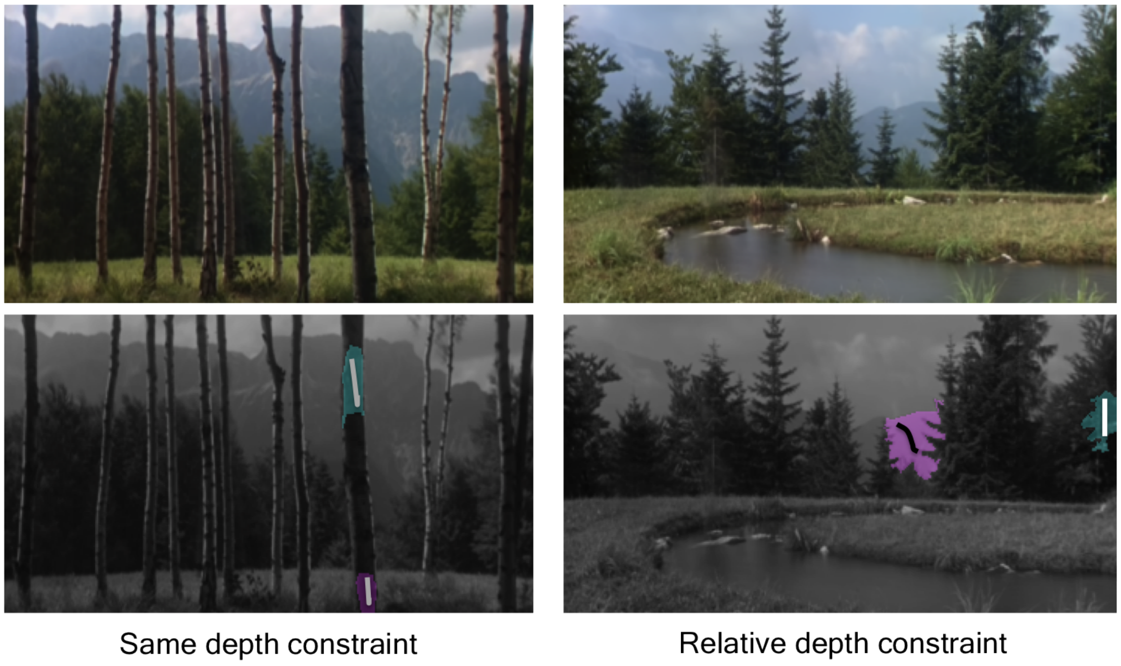


Figure 3.19: Pairwise constraints provided by the user. The grey color is used to indicate two regions at the same depth layer, while the white and black colors are used to indicated relative depth ordering where white = near, black=far.

answered. First, since the observed pairwise constraints are very sparse, there is a very huge space of possible queries to prompt to the user. What is a smart sampling strategy that could be used in this scenario? Second, given that we know which query to prompt to the user, what is the the feedback the user is required to provide to the algorithm and how is this feedback incorporated into the original framework? In the following sections we will first address the more easier question of how we can incorporate the user feedback in the algorithm and then discuss the sampling strategy.

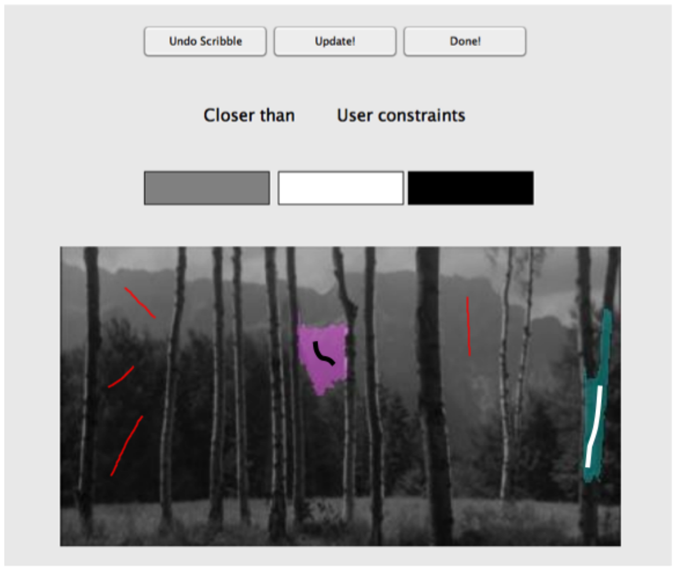
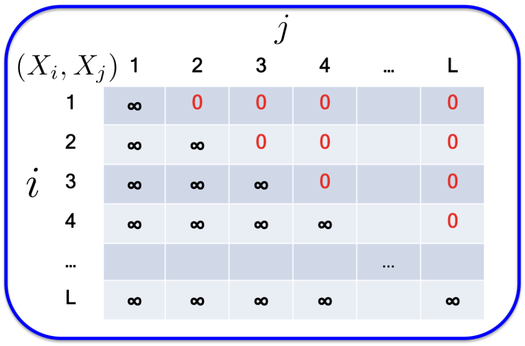


Figure 3.20: User interface to accept pairwise constraints.

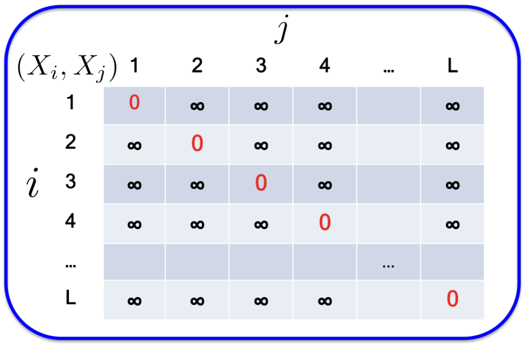
3.5.1 Incorporating the user constraints

We consider that the sampling strategy has picked the superpixels i and j i.e. two nodes in the graph over the background scene for the user to provide feedback. We allow the user to provide feedback (U_i, U_j) in the form of one of three responses, i and j belong to the same depth layer $(U_i = U_j)$, i occludes j $(U_i < U_j)$ or i is behind j $(U_i > U_j)$, shown in Figure 3.19. Such queries could also be crowd sourced similar to the work by Gingold et. al. [29]. We modify our earlier user interface as shown in Figure 3.20. The occlusion matrix entry $O_{i,j}$ is accordingly updated as before. The new information from the user corresponds to an additional edge in the graph over the background scene. This is now incorporated



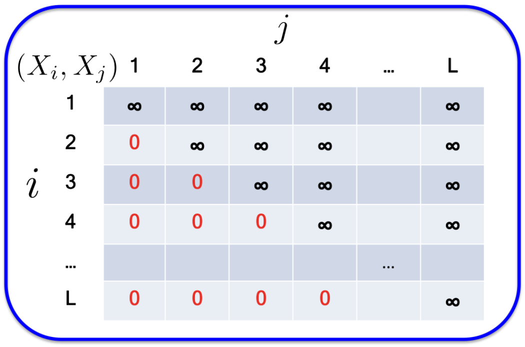
				j			
	(X_i, X_j)	1	2	3	4	...	L
	1	∞	0	0	0		0
	2	∞	∞	0	0		0
i	3	∞	∞	∞	0		0
	4	∞	∞	∞	∞		0
	...					...	
	L	∞	∞	∞	∞		∞

(a) TRI$_L$



				j			
	(X_i, X_j)	1	2	3	4	...	L
	1	0	∞	∞	∞		∞
	2	∞	0	∞	∞		∞
i	3	∞	∞	0	∞		∞
	4	∞	∞	∞	0		∞
	...					...	
	L	∞	∞	∞	∞		0

(b) DIAG$_O$



				j			
	(X_i, X_j)	1	2	3	4	...	L
	1	∞	∞	∞	∞		∞
	2	0	∞	∞	∞		∞
i	3	0	0	∞	∞		∞
	4	0	0	0	∞		∞
	...					...	
	L	0	0	0	0		∞

(c) TRI$_U$

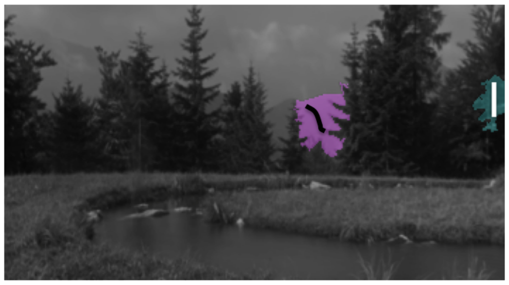
Figure 3.21: Spatial pairwise term - incorporating user constraints (U_i, U_j) that gives the pairwise relationship between superpixels i and j. (a) i occludes j $(U_i < U_j)$. (b) i and j belong to the same depth layer $(U_i = U_j)$. (c) i is behind j $(U_i > U_j)$.

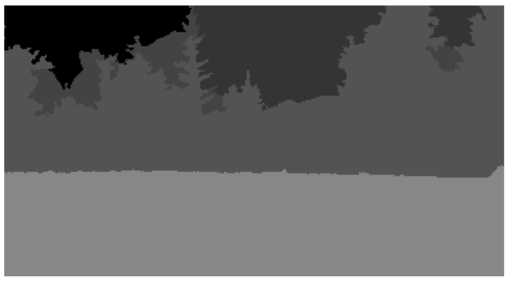
into the energy function as a new pairwise term as follows,

$$E^S_{ij}(X_i, X_j) = \begin{cases} \text{TRI}_L & U_i < U_j \\ \text{DIAG}_O & U_i = U_j \\ \text{TRI}_U & U_i > U_j \end{cases} \tag{3.9}$$

where, TRI$_U$, DIAG$_O$ and TRI$_L$ are pairwise matrices that enforce hard constraints based on the user input. The pairwise matrices are square matrices illustrated in Figure 3.21. This is the foundation for an iterative algorithm where, with every iteration the user provides a new pairwise constraint that is incorporated into the energy function (Eqn 3.6) as discussed above. We then perform

(a) Static scene


(b) User constraint


(c) Initial depth layers

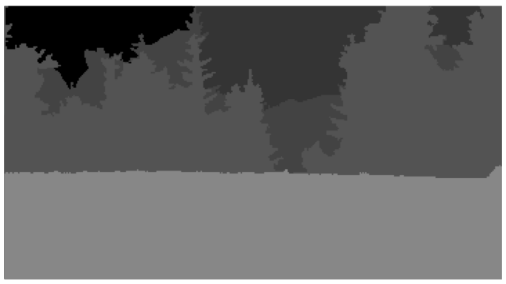

(d) Depth layers with user constraint

Figure 3.22: Note the change in the depth layers upon incorporating the user constraint. The pairwise terms ensure that the pink region be pushed back as apposed to the cyan region being pulled in front since that would have resulted in more pairwise penalties given the evidence.

inference using TRW-S [52] to obtain an updated depth ordering for the scene. Figure 3.22 shows the change in the depth layers upon incorporating a user constraint.

3.5.2 Guiding the user feedback

There is a huge space of possible queries where we can require the user input. We illustrate this using the skew-symmetric occlusion matrix for one of the videos in SET-A in Figure 3.23. Note the large green space that indicates the unknown pairwise depth ordering cues. Our goal is to guide the user and choose queries amongst the large number of options at each iteration in an

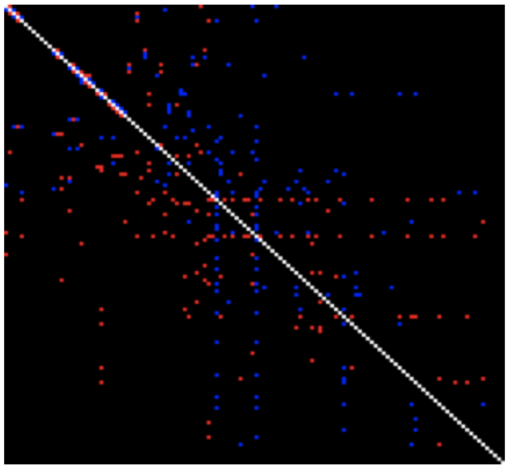


Figure 3.23: Observed occlusion matrix (static scene) for a video in SET-A. This is a skew-symmetric square matrix of dimension as the number of superpixels in the static scene. Red and blue points indicate pairwise occlusion relationships, +1 and −1 respectively and white points are the dominant diagonal that does not have pairwise relationships. The black region corresponds to the space of unknown pairwise relationships.

active-learning framework.

At every iteration, given the current solution to the depth layer segmentation problem that includes all the observed pairwise depth ordering, and the current depth ordering, we use a learnt regression function that provides a score for each possible query. This scoring function is used to rank the huge space of queries and select the best query for the user input.

3.5.3 Query scoring function

Given the current depth ordering inferred using the algorithm and the structure of the graph over the background scene we learn a regression over extracted features. We list the features extracted for each query superpixel pair in Table

Pairwise query features	**dim**
Relative query appearance features [42]	59
Coplanar classifier score (Refer Section 3.4.1 for details)	1
Sum of coplanar classier scores using all superpixels connected to the query	1
Sum of normalized size of the query	1
Normalized spatial distance between the centroids of query	1
Binary feature that indicates whether the query are spatially adjacent	1
Sum of number of adjacent superpixels for the query	1
Maximum number of adjacent superpixels for query	1
Depth label difference between the query in the current solution	1
Binary feature that indicates if the query has the same depth label in the current solution	1

Table 3.3: Features (68-dim) for the regression function to rank the queries. Here, each 'query' is a pair of superpixels to be queried for pairwise constraint.

3.3.

Training. In order to learn our regression function we use the ground-truth depth layers to obtain the training data. For each video in SET-A we start with the solution from the automatic algorithm and enumerate all the possible query pairs of superpixels i.e. the unknown pairs as shown in Figure 3.23. We start with the result of the automatic algorithm, consider the query pairs one at a time and use the ground-truth as the oracle to provide the pairwise relationship for the query superpixels. This additional pairwise constraint is incorporated into the energy function as described in Section 3.5.1. We measure the improvement in accuracy over the original result from the automatic algorithm by incorporating the additional constraint and repeat this process for each possible query. This vector of improvement in accuracy serves as the output vector for our regression and the features for the regression function are defined in the Table 3.3. We use the training data and learn a linear regression function using ridge regression.

Inference. Given a test video sequence, the original automatic algorithm infers the initial depth-layers of the scene and the observed occlusion matrix indicates the space of possible queries. We compute the regression features and use the learnt regressor to obtain the importance score for each query pair. The regressed scores are then ranked to obtain the best query for user input. Once the user is queried with this best query pair, the feedback is incorporated as described in Section 3.5.1 to obtain an updated solution. The process is repeated at each iteration, where we use the updated list of possible queries, obtain the new feature vectors given the current solution and again regress the importance score for each query.

3.5.4 Experiments and results

We evaluate the performance of the proposed algorithm using a machine user i.e. we use the ground-truth depth layers as the human oracle that is queried with the query pair of superpixels to obtain the pairwise constraint between them allowing for more exhaustive experiments. The videos in SET-A (Section 3.4.2) are used for all the experiments.

The person walking about the scene reveals pairwise depth ordering cues between adjacent superpixels. We first compare the improvement in performance obtained by the user in the loop guided by random sampling of queries. We perform three experiments within this setup. We first restrict the space of queries to only the adjacent superpixels. This is shown using the blue curve in Figure 3.24. Note that this gives us an indication of the gain we stand to obtain as the person walks about the various regions of the scene since the moving person reveals cues only between adjacent superpixels. In contrast, when we restrict the space of queries to only the *non*-adjacent superpixels, there is a sharp increase in performance as shown by red curve in Figure 3.24. This is an important result since it demonstrates that there is we stand to gain significantly by querying non-adjacent superpixels, i.e., pairwise relationships we could have never obtained relying on the moving person. In addition, most works in occlusion or depth reasoning use a graph similar to our spatial graph wherein a node (superpixel) is connected only to the immediately adjacent nodes (superpixels) however, this result shows the importance of long distance edges. The performance of random sampling of all the queries is shown in the green curve in Figure 3.24.

We report the quantitative performance in Figure 3.25. In Figure 3.25(a), the blue curve shows the performance when the algorithm uses random sampling as described above. The red curve shows the performance of the algorithm using the learnt scoring function. We observe that the scoring function guides the user towards better queries and performs better than the random sampling. Note that the error bars show the standard deviation of performance *across the dataset*. The learnt scoring function outperforms the random sampling in each of the videos. In Figure 3.25(b), we show the performance of the upper-bound using the green curve, i.e., the performance when the ground-truth is used to

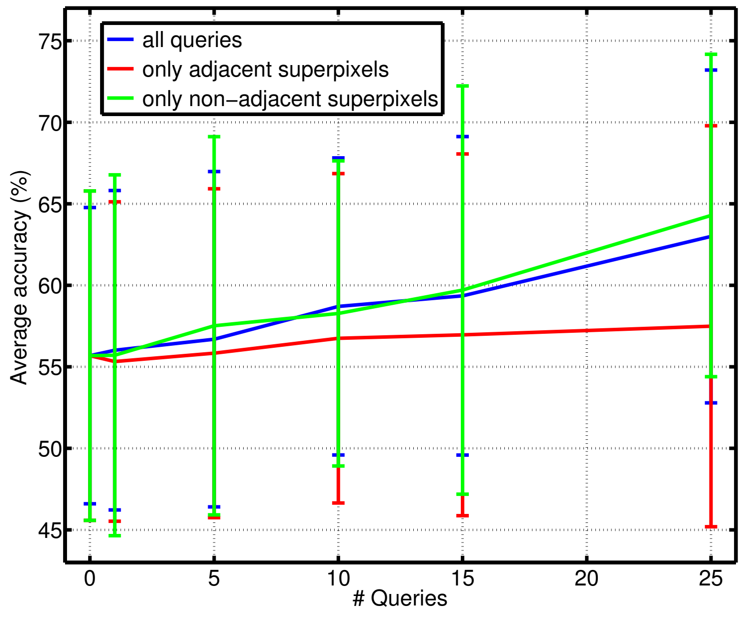
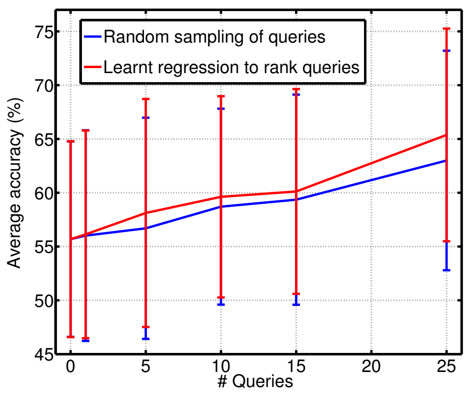
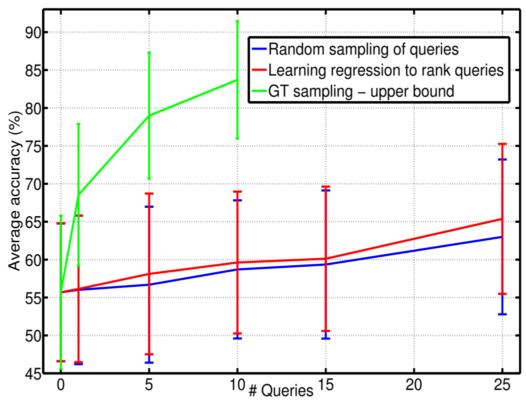


Figure 3.24: Random query sampling - performance of adjacent vs. non-adjacent superpixel queries. Every query is randomly selected from all the possible queries. The performance of the algorithm is significantly better when we use non-adjacent superpixel queries as apposed to adjacent superpixels. The error bars show the standard deviation across the videos in the dataset.

sample the query that provides the maximum improvement in performance. Note that with as few as 10 queries the upper-bound shows the huge increase in accuracy by selecting the optimal queries across the videos. However, we observe that the learnt function still has a significant gap from the upper bound. Several experiments using other variants of the ranking function (such as SVM-rank, kernel regression) were performed however, they failed to perform better than the linear regression. This is an interesting sub-problem that has scope for further improvement to study whether it is realistic to close the gap between the upper-bound and the performance of the query ranking function in practice.



(a)



(b)

Figure 3.25: Performance putting the user in the loop. The figure shows the average performance across the 24 videos in SET-A. (a) The performance of the random sampling where each query is randomly chosen from all the possible queries is shown in blue. The learnt scoring function that ranks the queries and picks the best query is shown in red, which outperforms the random sampling. The error bars show the standard deviation across the dataset. We note that the learnt scoring function outperforms the random sampling in each of the videos. (b) The green curve shows the performance of the upper-bound obtained by exhaustive ground-truth sampling.

3.6 Summary

The dynamic object moving about the scene are often considered clutter in many computer vision tasks. Our focus in the chapter was to leverage the dynamic object in the scene to aid image-based modeling and recovering depth information of the scene containing dynamic content. In particular we developed novel computational engines for two scenarios. First, we considered the dynamic scene captured by a dynamic camera and showed that we can use the background scene reconstructed via standard multi-view stereo along with the interaction between the moving object and the background scene to recover a dense depth map of the dynamic scene. Second, we considered the dynamic scene captured by a static camera and showed that even in the absence of camera motion we can turn the table around and use the sparse, yet strong motion occlusion cues revealed by moving objects interacting with the static scene along with monocular cues for occlusion reasoning in a unified framework to decompose the scene into depth layers. The results in both the scenarios show the efficacy of the proposed algorithms. We finally explored putting the user in the loop to provide pairwise depth constraints to improve the solution of the algorithm. We showed that the user guided by the the algorithm to provide useful pairwise constraints performs better than an unguided user.

Acknowledgements. The work in this chapter was partly supported by NSF DMS-0808864.

CHAPTER 4

APPLICATIONS: IMAGE-BASED MODELING ON A MOBILE DEVICE

4.1 iModel: Interactive 3D modeling on a mobile device

If there is one thing the growing popularity of immersive virtual environments (like Second-Life® with 6.1 Million members) and gaming environments (like Project Natal®) has taught us – it is that people crave personalization. For example, gamers want to be able to 'scan' and use their own gear (such as skateboards) in a skateboarding game; people want to be able to take something from the real world (such as a statue, or your house) into the virtual environment. We use the proposed idea of putting the user in the loop to develop an easy approach to obtain a 3D model of their object of interest. In particular, driven by the ubiquitous spread of mobile devices with touch-screen interfaces, we develop a mobile application to perform this task.

We give an overview of our mobile application [61] in Fig. 4.1. The application uses a client-server setup and is developed for iOS devices. The client (the user) captures a video of the object of interest by walking around the object. This video is then sent to the server that samples frames from the video, starts running structure-from-motion to extract the camera parameters and sends the sampled frames back to the client. The user is now allowed to flip through the images, select any image and provide user interactions via the touch-screen to indicate the object of interest and the background via scribbles. These scribbles are then sent to the server, which performs interactive co-segmentation,

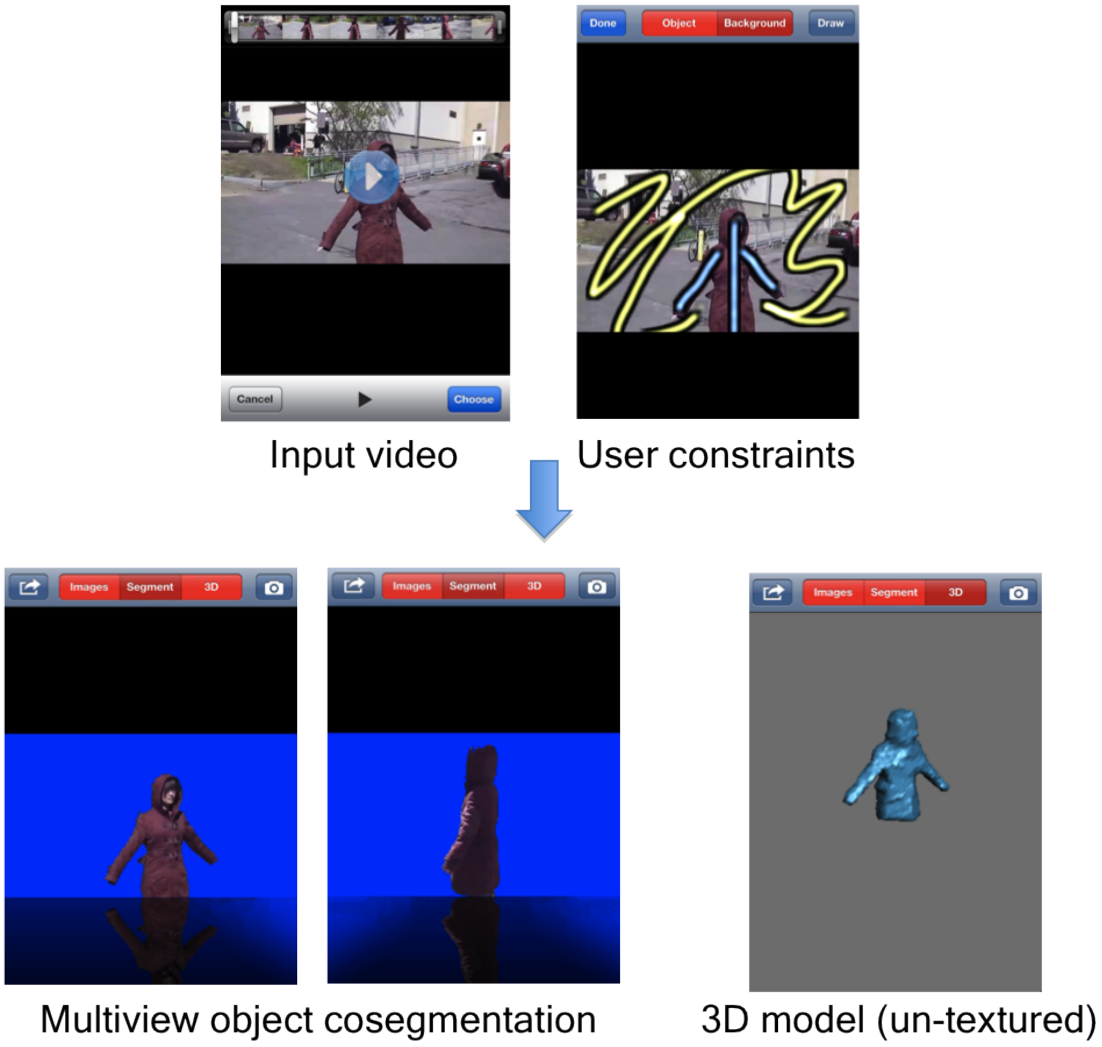


Figure 4.1: Overview of object of interest 3D modeling on a mobile device. Please refer website[1] for a demo video of the application.

followed by shape-from-silhouette using the algorithm described in Chapter 2. The co-segmentation of the object of interest from each view and the 3D model of the object are now sent back to the client; which the user can visualize. Please refer to our website for a demo video of the application[1].

[1]http://chenlab.ece.cornell.edu/projects/iModel

4.2 From images to physical 3D printouts

While augmented reality is a well established application that allows for virtually placing novel objects in a scene, an interesting application of the reconstructed 3D model follows the recent trend in 3D printing. Suppose that we wish to obtain a 3D model of the Ezra Cornell statue. Current 3D printing tools allow users to print existing models on the web or create new 3D models using CAD tools, neither of which are feasible in the scenario we consider. We use the reconstructed model obtained using the algorithm described in Chapter 2 to obtain a physical 3D printout of the object[2] as shown in Fig. 4.2, allowing for an interesting new application to obtain physical 3D models from images of the object captured in it's natural environment.

In addition, the trend in 3D printing is driven by the end user being able to print customized models, which is one possible future direction with this application. Using the 3D model obtained using the image data as an initial estimate of the 3D structure, we can envision an algorithm that translates it to a CAD model by representing the model using a combination of 3D geometric primitives. The CAD model thus obtained would form an initial point for the user to start and customize.

[2]The 3D printouts were obtained using the online service *http://www.shapeways.com*

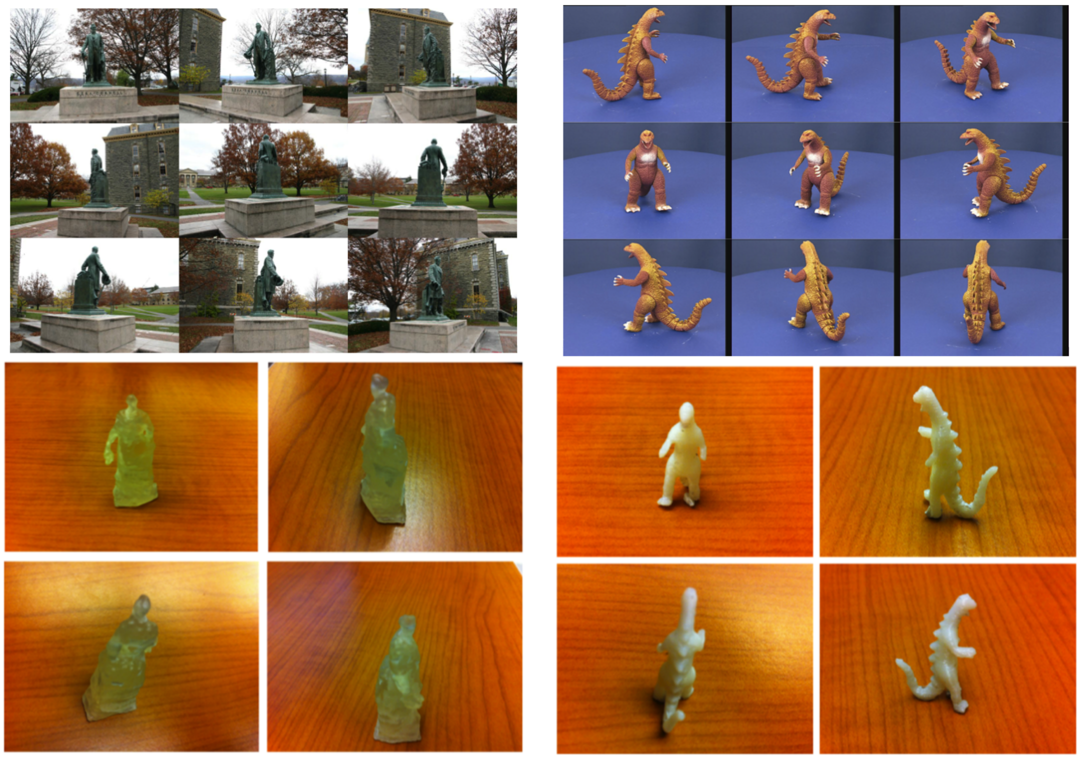


Figure 4.2: Physical 3D printout of the object of interest obtained using the proposed algorithm. The top row shows the set of multiview images of the object of interest used to obtain the 3D printout below.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

Image-based modeling, the task of recovering the 3D structure of an object or a scene using 2D images has seen significant progress in the recent years. However, 3D reconstruction of scenes captured in natural scenes is a hard task, due to the numerous scene irregularities such as textureless surfaces, specularities, thin structures, background clutter, etc., resulting in inaccurate reconstructions. In addition, unconstrained image sequences often consist of dynamic or moving objects such as people walking about in the scene, which makes the task of image-based modeling extremely hard. In this thesis, we have showed that we can address these issues by using the intuition that humans are good at interpreting the 3D structure behind the photons they view, and putting the user in the loop to aid image-based modeling.

We have showed in this thesis that while putting the user in the loop is intuitive, it requires designing a good computational engine that is capable of incorporating user constraints, while keeping the user constraints simple and intuitive, yet powerful. We leveraged the use of simple and intuitive scribbles as user constraints in all our works. In particular, we note that the user in the loop did not directly solve the same problem as image-based modeling. The role of the user was to either provide sparse labels to the computational engine to indicate to the algorithm the object of interest, or provide scene-specific support constraints and relative depth constraints to indirectly improve the solution provided by the computational engine. In addition, we showed that we can exploit the power of the computational engine to perform all the computationally intensive tasks and suitably guide the user input when and where needed via simple tasks.

In case of a static scenes captured from multiple viewpoints, we developed algorithms where the user guided the task of image-based modeling by providing constraints using simple and intuitive scribbles. Within a discrete labeling framework, we first demonstrated how the user can initiate the process by providing node constraints. We showed that the node constraints helped identify the object of interest to be modeled, which was an ill-posed problem without the user in the loop. We then explored how we can leverage the power of an unsupervised image-based modeling algorithm and get the user into the loop to provide scene specific cues to the algorithm. We demonstrated through our experiments and results that we can minimize the user effort via a novel active-learning algorithm that intelligently guides the user to providing edge constraints or 3D support cues to obtain a better reconstruction, and much faster.

In case of a dynamic scenes, we first focused on the computational engine and developed algorithms that exploit the dynamic objects such as people walking about the scene to implicitly aid the task of image-based modeling. Observing the object motion reveals useful depth or occlusion cues about the scene, implicitly revealing an idea about the 3D structure of the scene, where in the scene the objects can move, where is the scene occluding their path, etc. We first considered a dynamic object captured by a dynamic camera and developed an algorithm where we use the occlusions between the moving object and the scene along with the recovered depth for the static background to infer the depth of the dynamic scene with the user in the loop to co-segment the moving object in the scene. We then considered the scenario where the dynamic object was captured by a static camera that is void of any multiview stereo cues, and developed a novel graph based algorithm that used the sparse, yet powerful pairwise depth ordering cues revealed by the moving object to decompose the scene into depth layers. We then put the user into the loop within the same framework where the algorithm guides the user to provide additional useful depth ordering constraints to improve the solution. Our experiments and results showed the efficacy of the proposed algorithms.

5.1 Future work

The following are different directions to pursue in the future:

5.1.1 Leveraging the active-learning framework

Interactive image-based modeling is an active topic that has applications in consumer image-based rendering applications such as image and video editing, and in commercial applications such as converting monoscopic videos to stereo videos. The current approaches require a very involved user providing an enormous amount of annotation. In this thesis, we proposed ideas to leverage the power of computer vision algorithms and introduced the idea of active-learning for image-based modeling. Our formulation was within the framework of piecewise planar modeling of the scene, however the proposed idea of active-learning for putting the user in the loop has potential beyond piecewise planar reconstructions. The framework of guiding the user to provide feedback to obtain better reconstructions can be extended to multi-view stereo approaches (which can aid dense surface reconstruction). For example, in case of dense surface reconstructions, the user input could be in the form of drawing curves that act as guides to recover the surface of the object. It can also be used to reconstruct the scene using a single image, which has an inherent learning-based framework. The active-learning framework incorporates the positive aspects of both the automatic as well as the interactive algorithms, using the *scene-specific* user constraints when and where needed, to render improved reconstructions.

5.1.2 Leveraging semantics for image-based modeling

The focus of this thesis was about putting a human user in the loop since humans have a much better interpret the 3D representation of a scene. A question to ask here is can the actual human user be replaced by human priors that leverage semantics within the image-based modeling algorithm? For example, in our example of the green couch captured from multiple viewpoints, the scene irregularities resulted in the algorithm making errors on the homogeneous surfaces of the couch. However, understanding that the object we are looking at is a couch, provides a strong prior to the structure of the object aiding the task of image-based modeling. In case of dynamic objects in the scene, estimating the geometric layout such as the ground surface of the scene can help constrain the depth of the moving object, for example, an estimate of the point of contact of the object with the ground (when visible) significantly constraints the depth of the object. In [69], we developed an algorithm called Feedback Enabled Cascaded Classifier Models (FE-CCM) with the idea that the various vision tasks such as object detection, saliency detection, geometric labeling and even depth estimation share a lot of information between each other. For example, given a test image, inferring the depth of the scene can aid better performance at object detection. While this algorithm explored the single image scenario, extending this to multiview images presents itself as a unified framework to leverage semantics for image-based modeling and multiview scene understanding.

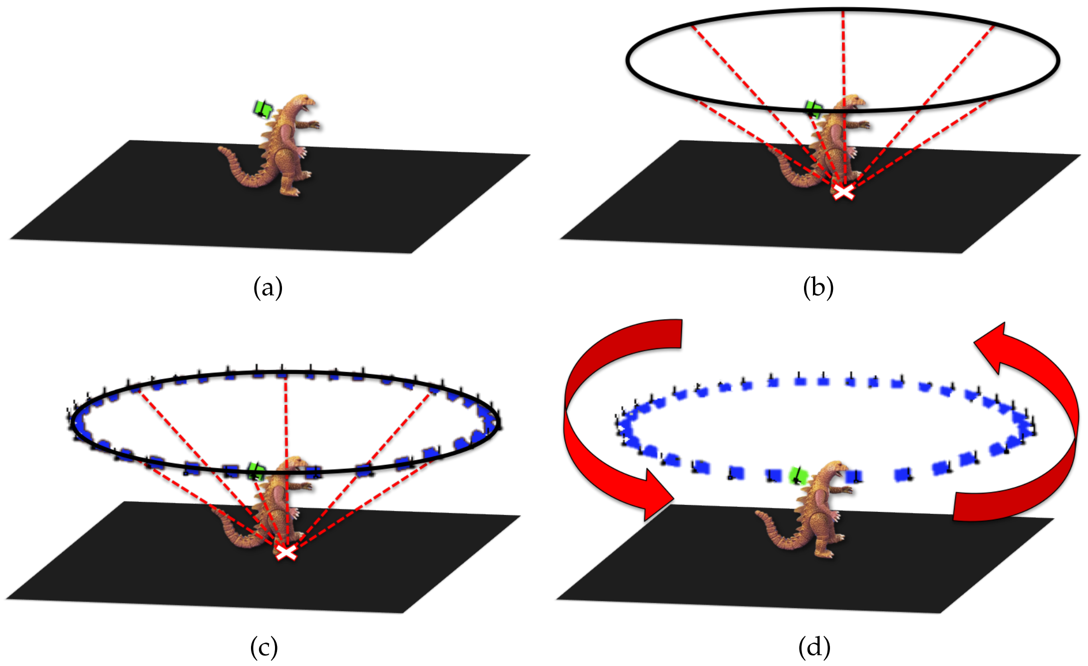


Figure 5.1: Capturing the right image by actively guiding the user in 3D. (a) The object of interest lies on a planar surface. The green square illustrates the initial position of the camera after the first capture. The red dashed line is illustrated between the camera center and the origin of the reference co-ordinate system (shown in red $\times$). (b) A ring of possible camera positions is obtained by fixing the height of the camera, illustrated in the black ring. (c) The required number of camera positions are uniformly sampled along the ring. (d) The final camera positions to guide the user are thus obtained.

5.1.3 Putting the user in the loop during image capture

With an increasing number of sensors and metadata available on portable cameras, the algorithms can leverage more than just the image data. A benefit of this is that algorithms can put the user in the loop *during image capture* by leveraging the accelerometer and the gyroscope data along with the image data to guide the user towards capturing the next image based on the task at hand. As an example, consider the application of object of interest 3D modeling, where we wish to obtain images of the object from different viewpoints. We constrain the problem the following, guide the user around the object of interest such that the user is made to capture images uniformly spaced on a ring of possible camera locations as illustrated in Figure 5.1(d). It is intuitive that this framework aids the iModel application we described in Section 4.1. We skip the time consuming step of structure from motion since we have a predefined camera pose for each image. The computational engine can therefore leverage the user to aid much faster 3D modeling of the object of interest. This is just one application but, the idea of putting the user in the loop during image capture is an interesting direction to pursue in the future.

APPENDIX A

RELATED PUBLICATIONS

Books and Journal papers:

- Adarsh Kowdle, Andrew Gallagher and Tsuhan Chen. "Leveraging the Dynamic Object in the Scene for Image-Based Modeling", IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2013. (Submitted)
- Adarsh Kowdle and Tsuhan Chen. "Putting the User in the Loop for Image-Based Modeling", International Journal of Computer Vision (IJCV), 2013. (Submitted)
- Congcong Li, Adarsh Kowdle, Ashutosh Saxena, Tsuhan Chen. "Towards Holistic Scene Understanding: Feedback Enabled Cascaded Classification Models." IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), July, 2012.
- Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo and Tsuhan Chen. "Interactive Co-segmentation of Objects in Image Collections", SpringerBriefs in Computer Science, 2011.
- Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo and Tsuhan Chen. "Interactively Co-segmenting Topically Related Images with Intelligent Scribble Guidance", International Journal of Computer Vision (IJCV), 2011.

Conference and Workshop papers:

- Adarsh Kowdle, Andrew Gallagher and Tsuhan Chen. “Revisiting Depth Layers from Occlusions”, IEEE Computer Vision and Pattern Recognition (CVPR), 2013.
- Adarsh Kowdle and Tsuhan Chen. “Learning to Segment a Video to Clips Based on Scene and Camera Motion”, European Conference on Computer Vision (ECCV), 2012.
- Adarsh Kowdle, Sudipta Sinha and Richard Szeliski. “Multiple View Object Cosegmentation using Appearance and Stereo Cues”, European Conference on Computer Vision (ECCV), 2012.
- Adarsh Kowdle, Andrew Gallagher and Tsuhan Chen. “Combining monocular geometric cues with traditional stereo cues for consumer camera stereo”, Workshop on Unsolved Problems in Optical Flow and Stereo Estimation, European Conference on Computer Vision (ECCV), 2012.
- Adarsh Kowdle, Noah Snavely and Tsuhan Chen.“Recovering Depth of a Dynamic Scene using Real World Motion Prior”, IEEE International Conference on Image Processing (ICIP), 2012.
- Adarsh Kowdle, Yao-Jen Chang, Andrew Gallagher and Tsuhan Chen. “Active Learning for Piecewise Planar 3D Reconstruction”, IEEE Computer Vision and Pattern Recognition (CVPR), 2011.
- Adarsh Kowdle, Yao-Jen Chang, Dhruv Batra and Tsuhan Chen. “Scribble Based Interactive 3D Reconstruction via Scene Co-segmentation”, IEEE International Conference on Image Processing (ICIP), 2011.

- Congcong Li, Adarsh Kowdle, Ashutosh Saxena, Tsuhan Chen. "Towards Holistic Scene Understanding: Feedback Enabled Cascaded Classification Models", Neural Information Processing Systems Conference (NIPS), 2010.
- Adarsh Kowdle, Dhruv Batra, Wen-Chao Chen and Tsuhan Chen. "iModel: Interactive Co-segmentation for Object of Interest 3D Modeling", Workshop on Reconstruction and Modeling of Large-Scale 3D Virtual Environments, European Conference on Computer Vision (ECCV), 2010.
- Adarsh Kowdle*, Congcong Li*, Ashutosh Saxena, Tsuhan Chen. "A Generic Model to Compose Vision Modules for Holistic Scene Understanding", Workshop on Parts and Attributes, European Conference on Computer Vision (ECCV), 2010.
- Adarsh Kowdle, Kuo-Wei Chang, and Tsuhan Chen. "Video Categorization using Object of Interest Detection", IEEE International Conference on Image Processing (ICIP), 2010.
- Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jeibo Luo and Tsuhan Chen. "iCoseg: Interactive Co-segmentation with Intelligent Scribble Guidance", IEEE Computer Vision and Pattern Recognition (CVPR), 2010.
- Adarsh Kowdle, Yao-Jen Chang and Tsuhan Chen. "i3D: Interactive Planar Reconstruction of Objects and Scenes", IEEE Western New York Image Processing Workshop (WNYIPW), 2009.
- Dhruv Batra, Adarsh Kowdle, Devi Parikh and Tsuhan Chen. "Cutout Search: Putting a Name to the Picture", Workshop on Internet Vision, IEEE Computer Vision and Pattern Recognition (CVPR), 2009.
- Dhruv Batra, Devi Parikh, Adarsh Kowdle, Tsuhan Chen, and Jiebo Luo. "Seed Image Selection in Interactive Cosegmentation", IEEE International Conference on Image Processing (ICIP), 2009.

Demos:

- Adarsh Kowdle, Haochen Liu, ShaoYou Hsu, Jason Lew, Charvi Puri, Dhruv Batra and Tsuhan Chen. “iModel: Object of Interest 3D Modeling via Interactive Co-segmentation on a Mobile Device”, Demo session at IEEE Computer Vision and Pattern Recognition (CVPR), 2012.
- Dhruv Batra, Adarsh Kowdle, Kevin Tang, Devi Parikh, Jiebo Luo and Tsuhan Chen. “Interactive Cosegmentation by Touch”, Demo session at IEEE Computer Vision and Pattern Recognition (CVPR), 2009.

BIBLIOGRAPHY

[1] Shai Bagon. Matlab wrapper for graph cut. `http://www.wisdom.weizmann.ac.il/~bagon`, December 2006.

[2] Adrien Bartoli. A random sampling strategy for piecewise planar scene segmentation. *CVIU*, 105(1):42–59, 2007.

[3] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. Cutout search: Putting a name to the picture. *Workshop on Internet Vision, CVPR*, 2009.

[4] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. Seed image selection in interactive cosegmentation. *ICIP*, 2009.

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

[6] D. Batra, A. Kowdle, D. Parikh, J. Luo, and T. Chen. *Interactive Co-segmentation of Objects in Image Collections*. SpringerBriefs in Computer Science, 2011.

[7] Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, and Tsuhan Chen. Interactively co-segmenting topically related images with intelligent scribble guidance. *IJCV*, 93(3):273–292, 2011.

[8] Bruce Guenther Baumgart. *Geometric modeling for computer vision.* PhD thesis, Stanford University, 1974.

[9] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004.

[10] Y. Boykov, O. Veksler, and R. Zabih. Efficient approximate energy minimization via graph cuts. *PAMI*, 20(12):1222–1239, 2001.

[11] G. Brostow and I. Essa. Motion based decompositing of video. In *ICCV*, 1999.

[12] Neill Campbell, George Vogiatzis, Carlos Hernndez, and Roberto Cipolla. Automatic 3D object segmentation in multiple views using volumetric graph-cuts. In *BMVC*, 2007.

[13] Neill D. Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *ECCV*, 2008.

[14] Wen-Chao Chen, Hong-Long Chou, and Zen Chen. A quality controllable multi-view object reconstruction method for 3D imaging systems. *JVCIR*, 21(5-6):427 – 441, 2010.

[15] Zen Chen, Hong-Long Chou, and Wen-Chao Chen. A performance controllable octree construction method. In *ICPR*, 2008.

[16] B.n Collins, J. Deng, K. Li, and L. Fei-Fei. Towards scalable dataset construction: An active learning approach. In *ECCV*, 2008.

[17] Robert T. Collins. A Space-Sweep Approach to True Multi-Image Matching. In *CVPR*, 1996.

[18] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *PAMI*, 24(5):603–619, 2002.

[19] A. Criminisi, I. D. Reid, and A Zisserman. Single view metrology. In *ICCV*, 1999.

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

[21] Yen-Hsiang Fang, Hong-Long Chou, and Zen Chen. 3D shape recovery of complex objects from multiple silhouette images. *Pattern Recogn. Lett.*, 24(9-10):1279–1293, 2003.

[22] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2):167–181, 2004.

[23] Keith Forbes, Fred Nicolls, Gerhard de Jager, and Anthon Voigt. Shape-from-silhouette with two mirrors and an uncalibrated camera. In *ECCV*, pages 165–178, 2006.

[24] D. Fouhey, V. Delaitre, I. Laptev, J. Sivic, A. Efros, and A. Gupta. People watching: Human actions as a cue for single view geometry. In *ECCV*, 2012.

[25] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *ICCV*, 2009.

[26] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *PAMI*, 2009.

[27] Yasutaka Furukawa, Brian Curless, Steven M. Seitz, and Richard Szeliski. Towards internet-scale multi-view stereo. In *CVPR*, 2010.

[28] David Gallup, Jan-Michael Frahm, and Marc Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *CVPR*, 2010.

[29] Y. Gingold, A. Shamir, and D. Cohen-Or. Micro perceptual human computation. *ACM Transactions on Graphics (TOG)*, 31(5):119:1–119:12, August 2012.

[30] Michael Goesele, Noah Snavely, Brian Curless, Hugues Hoppe, and Steven M. Seitz. Multi-view stereo for community photo collections. In *ICCV*, 2007.

[31] P. H. Gosselin and M. Cord. Active learning methods for interactive image retrieval. *IEEE Trans. on Image Processing*, 17(7):1200–1211, 2008.

[32] L. Guan, J.S. Franco, and M. Pollefeys. Probabilistic multi-view dynamic scene reconstruction and occlusion reasoning from silhouette cues. In *IJCV*, 2010.

[33] A. Gupta, A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010.

[34] A. Gupta, S. Satkin, A. Efros, and M. Hebert. From 3D scene geometry to human workspace. In *CVPR*, 2011.

[35] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. In *ECCV*, 2010.

[36] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[37] V. Hedau, D. Hoiem, and D. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV*, 2010.

[38] A. Hengel, A. R. Dick, T. ThormŁhlen, B. Ward, and P. H. S. Torr. Video-

trace: rapid interactive scene modelling from video. *ACM Trans. Graph.*, 26(3):86, 2007.

[39] D. Hoiem, A. Efros, and M. Hebert. Automatic photo pop-up. In *SIGGRAPH*, 2005.

[40] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. In *IJCV*, 2008.

[41] D. Hoiem, A. Efros, and M. Hebert. Recovering occlusion boundaries from an image. In *IJCV*, 2011.

[42] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Recovering surface layout from an image. *IJCV*, 75(1), 2007.

[43] Y. Horry, K. Aniyo, and K. Arai. Tour into the picture: Using a spidery mesh interface to make animation from a single image. In *SIGGRAPH*, 1997.

[44] P. Jain and A. Kapoor. Active learning for large multi-class problems. In *CVPR*, pages 762–769, 2009.

[45] Z. Jia, A. Gallagher, Y. Chang, and T. Chen. A learning based framework for depth ordering. In *CVPR*, 2012.

[46] Sing Bing Kang and Richard Szeliski. Extracting view-dependent depth maps from a collection of images. *IJCV*, 58:139–163, 2004.

[47] G. Kanizsa. Organization in vision: Essays on gestalt perception. In *Praeger*, 1979.

[48] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *ICCV*, 2007.

[49] M. Kendall. A new measure of rank correlation. *Biometrika*, 30:81–93, 1938.

[50] Pushmeet Kohli and Philip H. S. Torr. Measuring uncertainty in graph cut solutions. *CVIU*, 112(1):30–38, 2008.

[51] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *PAMI*, 26(2):147–159, 2004.

[52] Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *PAMI*, 28(10):1568–1583, October 2006.

[53] A. Kowdle, D. Batra, W. Chen, and T. Chen. iModel: Interactive co-segmentation for object of interest 3D modeling. In *ECCV - RMLE Workshop*, 2010.

[54] A. Kowdle, K. Chang, and T. Chen. Video categorization using object of interest detection. *ICIP*, 2010.

[55] A. Kowdle, Y. Chang, D. Batra, and T. Chen. Scribble based interactive 3D reconstruction via scene cosegmentation. In *ICIP*, 2011.

[56] A. Kowdle, Y. Chang, A. Gallagher, and T. Chen. Active learning for piecewise planar 3D reconstruction. In *CVPR*, 2011.

[57] A. Kowdle and T. Chen. Learning to segment a video to clips based on scene and camera motion. In *ECCV*, 2012.

[58] A. Kowdle, A. Gallagher, and T. Chen. Combining monocular geometric cues with traditional stereo cues for consumer camera stereo. *Workshop on Unsolved Problems in Optical Flow and Stereo Estimation, ECCV*, 2012.

[59] A. Kowdle, A. Gallagher, and T. Chen. Revisiting depth layers from occlusions. In *CVPR*, 2013.

[60] A. Kowdle*, C. Li*, A. Saxena, and T. Chen. A generic model to compose vision modules for holistic scene understanding. *Workshop on Parts and Attributes, ECCV*, 2010.

[61] A. Kowdle, H. Liu, S. Hsu, J. Lew, C. Puri, D. Batra, and T. Chen. iModel: Object of interest 3D modeling via interactive co-segmentation on a mobile device. In *Demo session at CVPR*, 2012.

[62] A. Kowdle, S. Sinha, and R. Szeliski. Multiple view object cosegmentation using appearance and stereo cues. In *ECCV*, 2012.

[63] A. Kowdle, N. Snavely, and T. Chen. Recovering depth of a dynamic scene using real world motion prior. In *ICIP*, 2012.

[64] F. Lafarge, R. Keriven, M. Brédif, and V. Hiep. Hybrid multi-view reconstruction by jump-diffusion. In *CVPR*, 2010.

[65] D. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, 2010.

[66] Wonwoo Lee, Woontack Woo, and Edmond Boyer. Identifying foreground from multiple images. In *ACCV*, 2007.

[67] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. The digital michelangelo project: 3D scanning of large statues. In *Siggraph*, pages 131–144, 2000.

[68] C. Li, A. Kowdle, A. Saxena, and T. Chen. Towards holistic scene understanding: Feedback enabled cascaded classification models. *NIPS*, 2010.

[69] C. Li, A. Kowdle, A. Saxena, and T. Chen. Towards holistic scene understanding: Feedback enabled cascaded classification models. *TPAMI*, 2012.

[70] B. Micusík and J. Kosecká. Multi-view superpixel stereo in urban environments. *IJCV*, 89(1):106–119, 2010.

[71] K. Nakayama. Biological image motion processing. In *Vision Research*, volume 25, pages 625–660, 1985.

[72] G. E. Noether. Why kendall tau? *Teaching Statistics*, 3(2):41–43, 1981.

[73] M. Okutomi and T. Kanade. A multiple-baseline stereo. *PAMI*, 15:353–363, April 1993.

[74] Hyun Soo Park, Takaaki Shiratori, Iain Matthews, and Yaser Sheikh. 3D reconstruction of a moving point from a series of 2d projections. In *ECCV*, 2010.

[75] M. Pollefeys, D. Nistr, J. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. Kim, P. Merrell, C. Salmi, S Sinha, B. Talton, L. Wang, Q. Yang, H. Stewnius, R. Yang, G. Welch, and H. Towles. Detailed real-time urban 3D reconstruction from video. *IJCV*, 78(2-3):143–167, 2008.

[76] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *IJCV*, V59(3):207–232, 2004.

[77] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[78] C. Rother, V. Kolmogorov, and A. Blake. "Grabcut" - Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.

[79] A. Saxena, S. Chung, and A. Y. Ng. Learning depth from single monocular images. In *NIPS*, 2005.

[80] A. Saxena, S. Chung, and A. Y. Ng. 3-D depth reconstruction from a single still image. *IJCV*, 76(1):53–69, 2008.

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

[82] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 47(1-3):7–42, 2002.

[83] A. Schodl and I. Essa. Depth layers from occlusions. In *CVPR*, 2001.

[84] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, 2006.

[85] S. Sinha, D. Steedly, and R. Szeliski. Piecewise planar stereo for image-based rendering. In *ICCV*, 2009.

[86] S. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3D architectural modeling from unordered photo collections. *SIGGRAPH Asia*, 2008.

[87] Sketchup. Google sketchup: `http://www.sketchup.com`. 2000.

[88] N. Snavely, S. Seitz, and R Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH*, 2006.

[89] S. Srivastava, A. Saxena, C. Theobalt, S Thrun, and A. Y. Ng. i23 - rapid interactive 3D reconstruction from a single image. In *Vision, Modeling and Visualization*, 2009.

[90] Peter F. Sturm and Stephen J. Maybank. A method for interactive 3D reconstruction of piecewise planar objects from single images. In *BMVC*, 1999.

[91] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, 1993.

[92] K. Tang, A. Kowdle, D. Batra, and T. Chen. iScribble, `http://chenlab.ece.cornell.edu/projects/iScribble/iScribble.html`. 2009.

[93] S. Vijayanarasimhan, P. Jain, and K. Grauman. Far-sighted active learning on a budget for image and video recognition. In *CVPR*, 2010.

[94] H. Wang, S. Gould, and D. Koller. Discriminative learning with latent variables for cluttered indoor scene understanding. In *ECCV*, 2010.

[95] Kwan-Yee K. Wong and Roberto Cipolla. Structure and motion from silhouettes. In *ICCV*, 2001.

[96] Kwan-Yee Kenneth Wong and Roberto Cipolla. Reconstruction of sculpture from its profiles with unknown camera positions. *IEEE Transactions on Image Processing*, 13(3):381–389, 2004.

[97] Rong Yan, Jie Yang, and Alexander Hauptmann. Automatically labeling video data using multi-class active learning. In *ICCV*, 2003.

[98] S.X. Yu, H. Zhang, and J. Malik. Inferring spatial layout from a single image via depth-ordered grouping. In *POCV Workshop*, 2008.

[99] Guofeng Zhang, Jiaya Jia, Wei Hua, and Hujun Bao. Robust bilayer segmentation and motion/depth estimation with a handheld camera. *PAMI*, 33:603–617, March 2011.

[100] Guofeng Zhang, Jiaya Jia, Tien-Tsin Wong, and Hujun Bao. Consistent depth maps recovery from a video sequence. *PAMI*, 31, June 2009.

[101] X. S. Zhou and T. S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia Systems*, 8(6):536–544, 2003.