Western University
Scholarship@Western

Digitized Theses | Digitized Special Collections

2008

Spatially Coherent Geometric Class Labeling of Images and Its Applications

Xiaoqing Liu
Western University

Follow this and additional works at: https://ir.lib.uwo.ca/digitizedtheses

Recommended Citation

Liu, Xiaoqing, "Spatially Coherent Geometric Class Labeling of Images and Its Applications" (2008). *Digitized Theses*. 4562.
https://ir.lib.uwo.ca/digitizedtheses/4562

This Thesis is brought to you for free and open access by the Digitized Special Collections at Scholarship@Western. It has been accepted for inclusion in Digitized Theses by an authorized administrator of Scholarship@Western. For more information, please contact wlswadmin@uwo.ca.

Spatially Coherent Geometric Class Labeling of Images and Its Applications

(Spine title: Image Geometric Class Labeling)

(Thesis format: Monograph)

by

Xiaoqing Liu

Graduate Program
in
Engineering Science
Electrical and Computer Engineering

A thesis submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy

Faculty of Graduate Studies
The University of Western Ontario
London, Ontario, Canada

© Xiaoqing Liu 2008

Abstract

Automatic scene analysis is an active research area and is useful in many applications such as robotics and automation, industrial manufacturing, architectural design and multimedia. 3D structural information is one of the most important cues for scene analysis.

In this thesis, we present a geometric labeling method to automatically extract rough 3D information from a single 2D image. Our method partitions an image scene into five geometric regions through labeling every image pixel as one of the five geometric classes (namely, "bottom", "left ", "center", "right", and "top"). We formulate the geometric labeling problem as an energy minimization problem and optimize the energy with a graph cut based algorithm. In our energy function, we address the spatial consistency of the geometric labels in the scene while preserving discontinuities along image intensity edges. We also incorporate ordering constraints in our energy function. Ordering constraints specify the possible relative positional labels for neighbor pixels. For example, a pixel labeled as the "left" can not be the right of a pixel labeled as the "right" and a pixel labeled as the "bottom" can not be above a pixel labeled as the "top". Ordering constraints arise naturally in a real scene. We observed that when ordering constraints are used, the commonly used graph-cut based α-expansion is more likely to get stuck in local minima. To overcome this, we developed new graph-cut moves which we call *order-preserving* moves. Unlike α-expansion which works for two labels in each move, order-preserving moves act on all labels. Although the global minimum is still not guaranteed, we will show that optimization with order-preserving moves is shown to perform significantly better than α-expansion.

Experimental results show that it is possible to significantly increase the percentage of reasonably good labeling by promoting spatial consistency and incorporating ordering constraints. It is also shown that the order-preserving moves performs significantly better than the commonly used α-expansion when ordering constraints are used as there is a significantly improvement in computational efficiency and optimality while the improvement in accuracy of pixel labeling is also modest.

We also demonstrate the usefulness of the extracted 3D structure information of a scene in applications such as novel view generation, virtual scene walk-through, semantic segmentation, scene synthesis, and scene text extraction. We also show how we can apply this order-preserving moves for certain simple shape priors in graph-cut segmentation.

Our geometric labeling method has the following main contributions:

(i) We develop a new class of graph-cut moves called order-preserving moves, which performs significantly better than α-expansion when ordering constraints are used.

(ii) We formulate the problem in a global optimization framework where we address the spatial consistency of labels in a scene by formulating an energy function which encourages spatial consistency between neighboring pixels while preserving discontinuities along image intensity edges.

(iii) We incorporate relative ordering information about the labels in our energy function.

(iv) We show that our ordering constraints can also be used in other applications such as object part segmentation.

(v) We also show how the proposed order-preserving moves can be used for certain simple shape priors in graph-cut segmentation.

Keywords:
Automatic, 3D structural information, Geometric labeling, Energy minimization, Graph cut, Ordering constraints, α-expansion, Order-preserving move.

Acknowledgements

I am deeply indebted to many people who have helped me in my graduate research studies in many ways. Foremost among them is my supervisor, Dr. Jagath Samarabandu, for his guidance, advice and substantial support during my study. I am so grateful to have such an excellent supervisor who has been so thoroughly supportive, helpful, and open-minded and provided me the freedom to explore various aspects of this research.

I would also like to express my sincere thanks to Dr. Olga Veksler and Dr. Yuri Boykov for providing such fascinating courses, in depth advice, discussions, help and support, from which I benefited significantly. Without their help, I can not reach this milestone.

I gratefully acknowledge my thesis committee members, Dr. Yuri Boykov, Dr. Ken McIsaac, Dr. Serguei Primak, and Dr. Shahram Shirani for agreeing to spend their precious time spent on reviewing my thesis.

I would like to thank all members in the Computer Vision and Mobile Robotics Laboratory, especially my colleges Zhenhe (Hogan) Chen and Ranga Rodrigo for their friendship and help towards my goal.

Finally my most special thanks go to my parents and my husband Jinhong (Henry) Wang for their love, encouragement, and support throughout these years.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

3D structural information is one of the most important cues for scene understanding in many applications such as robotics and automation, industrial manufacturing, architectural design and multimedia.

We are interested in developing algorithms which can extract 3D structural information from scene images. In general, there are two main approaches which can be used in 3D scene reconstruction: Multiple View approach and Single View approach.

Methods based on multiple view approach methods use feature correspondences [1] between images (e.g. stereo images, multiple-view images, video sequences, and/or 3D range data) and projective geometry constraints to reconstruct the 3D geometry and camera calibration parameters. Traditional 3D scene reconstruction based on multiple view geometry has been an active research area in the past few several decades. Although significant progress has been made in this area, a reliable and automatic 3D reconstruction is still a distant goal.

Traditional multiple view methods require special equipment, such as multiple cameras, range scanners, etc., as they need to use feature correspondence between images to reconstruct the 3D geometry [2]. Moreover, using projective geometry

constraints to recover a metric reconstruction of an architectural scene is usually computationally intensive. Furthermore, when the environment tends to be extremely low textured, using projective geometry is usually unreliable since the absence of reliable features makes the correspondence problem highly ambiguous.

Unlike traditional multiple view 3D reconstruction methods which can provide metric accurate 3D reconstruction, single view methods can only recover approximate 3D information due to the fact that a single view of a scene does not have enough information about the depth of the scene points. Single view reconstruction problem can be solved to some extent by either imposing some constraints on the scene or through user interaction.

In general, single view methods recover approximate 3D information by either manually partitioning the image into several geometrical parts or formulating it as an image segmentation or pixel labeling problem which involves assigning a geometrical label from a finite set of possibilities to each image pixel. The 3D structural information can then be obtained from these geometrical labels.

Although single view methods can only recover approximate 3D information, the recovered approximate 3D information is very useful in many applications such as for scene visualization, object recognition, virtual reality, etc.

More details about 3D reconstruction algorithms from both the multiple view approach and the single view approach are described in chapter 2.

Horry *et al.* provided an interactive user interface to partition the input image into five geometrical parts using a spidery mesh [3]. Hoiem *et al.* extracted only an approximate 3D structure from a single 2D image through learning the preferences of the rough geometric labels (such as "sky", "ground", etc.,) for each image pixel [4]. Inspired by above approaches, we present a geometric labeling method in this thesis, which can automatically extract rough 3D information from a single 2D image by

using a five-parts model (i.e., bottom, top, center, left and right). Unlike Hoiem *et al.*, we formulate the problem in a global optimization framework and optimize the energy with a graph cut based method. In this approach, we address the spatial consistency of the labels in a scene by formulating it as an energy function which encourages spatial consistency between neighboring pixels while preserving discontinuities along the image intensity edges. We also incorporate relative ordering information about the labels in our energy function which allows us to define and enforce spatial consistency rules. These consistency rules are based on such facts as a pixel labeled as the "left" can not be to the right of a pixel labeled as the "right" and a pixel labeled as the "bottom" can not be above a pixel labeled as the "top". During this research, we observed that the commonly used graph-cut based α-expansion is more likely to get stuck in a local minimum when ordering constraints are used. This led us to develop new graph-cut moves which we call *order-preserving* moves. Unlike α-expansion, order-preserving moves act on all labels. Although the global minimum is still not guaranteed, we will show that optimization with order-preserving moves performs significantly better than α-expansion.

Experimental results show that the overall performance is significantly improved by encouraging spatial consistency and incorporating ordering constraints. The percentage of reasonably good labeling[1] is increased from 29.3% to 74.3% for the 300 indoor images and from 16.7% to 61.9% for the 42 outdoor images. There is also a modest improvement in overall pixel labeling accuracy. When ordering constraints are used, comparing the proposed order-preserving moves with the commonly used α-expansion, there is a significant improvement in processing time from $85.5s$ (σ = 219.0s) to $62.3s$ (σ = 25.6s) for 300 indoor images and from $286.4s$ (σ = 480.1s) to

1. Reasonably good labeling refers to labeling with an overall accuracy which is above a certain threshold and can be used successfully for the virtual scene walk-through application.

56.5s (σ = 18.3s) for 42 outdoor images and on average the energy is 27.3% smaller (σ = 9.8%) for the 300 indoor images and is 29.2% smaller (σ = 18.5%) for the 42 outdoor images.

We also demonstrate the usefulness of the extracted 3D structure information of the scene in the applications of novel view generation, virtual scene walk-through, semantic segmentation, scene synthesis, and scene text extraction. We also show the applicability of the proposed order-preserving moves for certain simple shape priors in graph-cut segmentation.

Although Hoiem *et al.* attempted global optimization techniques for geometric labeling, they were not able to improve the performance [5]. We argue that the improvements we were able to achieve are probably due to the following factors. In Hoiem's method, the optimization is performed on superpixel level where a superpixel is simply an image region obtained from a region segmentation algorithm whereas our method is based on individual pixels. Hence their method is dependant on the region segmentation algorithm to ensure that all the pixels in each super-pixel belong to a single label. By optimizing on a pixel level, our algorithm does not suffer from such a limitation as we are able to break apart any superpixel as needed. In particular, we are able to better align the boundaries between the geometric labels with the intensity edges in the image, which provides better results. In addition, our stringent set of ordering constraints and better optimization with order-preserving moves contributes to the improvement in results.

1.2 Contributions

The main contributions of our proposed method are summarized as follows[2]:

(a) We proposed a geometric labeling method which can automatically extract the rough 3D information with high accuracy and efficiency. Here, the accuracy is in terms of correct pixel labeling.

(b) We formulated the problem in a global optimization framework, where we address the spatial consistency of the labels in the scene by formulating an energy function which encourages spatial consistency between neighboring pixels while preserving discontinuities along image intensity edges.

(c) We incorporated relative ordering information about the labels in our energy function as ordering constraints that arise naturally in the scene.

(d) We developed new graph-cut moves called order-preserving moves for the graph cut optimization framework, which performs significantly better than α-expansion when ordering constraints are used.

(e) Apart from applying the proposed ordering constraints in scene segmentation, we have shown that these constraints can also be used in other applications such as object part segmentation.

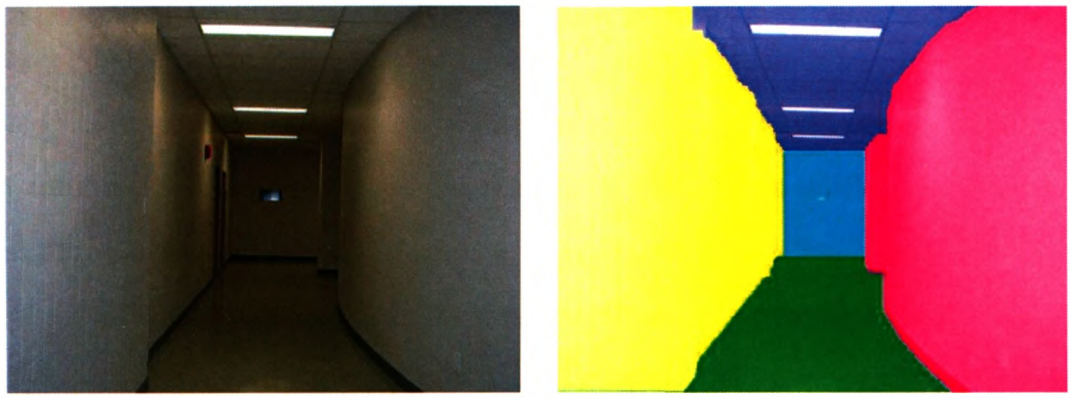
(f) We have shown that the proposed order-preserving moves can also be used for certain simple shape priors in graph-cut segmentation.

---

2. This work has been accepted for presentation at the *IEEE* International Conference on Computer Vision and Pattern Recognition (CVPR2008) which is one of the top conferences in this area.

1.3 Overview of the Proposed Algorithms

In our proposed method, we assume that an image is to be segmented into five geometric parts (each part corresponds to a label), which we call "center", "left", "right", "top", "bottom" as shown in Fig. 1.1.



(a) original image (b) image labels

Figure 1.1: Image and labels. Color scheme: green = "bottom", yellow = "left", cyan = "center", magenta = "right", blue = "top". We use this particular color scheme consistently throughout the thesis.

We formulate the geometric labeling problem in a global energy optimization framework, using our five part model, and optimize the energy with a graph cut based algorithm.

In the graph cut based optimization framework, there are two terms called the data term and the smoothness term, respectively. The data term specifies the penalty for a single pixel p to have a certain label, and thus encourages each pixel to be assigned the label that incurs the smallest penalty. The smoothness term encourages spatial consistency by penalizing neighboring pixels p and q that are not assigned the same label.

In our proposed method, we first train a classifier to find out individual label preferences for each pixel by using a Support Vector Machine (SVM) algorithm. We

address the spatial consistency of the geometric labels in the scene and also incorporate higher level knowledge (i.e. ordering constraints) about the scene labels in our smoothness term as spatial consistency and ordering constraints arise naturally in the scene.

The ordering constraints between the labels are easy to read from their names: a pixel labeled as "left" can not be the right of any pixel labeled as "center", a pixel labeled as "top" can not be below any pixel labeled as "center", etc. In addition, we can enforce a more stringent set of constraints: if a pixel p labeled as "center" has a neighbor q with a different label, then q must have label "left", "right", "top" or "bottom" if it is to the left, right, above, or below p, respectively.

In our method, we develop new graph-cut moves which we call *order-preserving* moves as we observe that the commonly used graph-cut based α-expansion is more likely to get stuck in a local minimum when ordering constraints are used. Unlike α-expansion, order-preserving moves act on all labels. Although the global minimum is still not guaranteed, we will show that optimization with order-preserving moves performs significantly better than α-expansion.

Experimental results show that promoting spatial consistency and incorporating ordering constraints significantly improves the performance. We also demonstrate the usefulness of the extracted 3D structure information of the scene in the applications of novel view generation, virtual scene walk-through, semantic segmentation, scene synthesis and scene text extraction. Meanwhile, we also use order-preserving moves for certain simple shape priors in graph-cut segmentation.

1.4 Organization of the Thesis

This thesis presents a geometric labeling method, which can automatically extract rough 3D information from a single 2D image. In previous sections, we briefly introduced the problems and the motivation behind this thesis.

Chapter 2 reviews the relevant literature on different 3D scene structural information extraction methods from two approaches, namely, the *Multiple View* approach and the *Single View* approach.

Chapter 3 is a brief overview of relevant algorithms including basic image processing, Support Vector Machines and the Graph Cut algorithms, and provides the theoretical base for the proposed algorithm.

Chapter 4 describes the proposed algorithm in detail and Chapter 5 illustrates some of the experimental results.

Finally, in chapter 6, we present the concluding remarks on the proposed algorithm followed by a discussion of future work.

Chapter 2

Related work

Recovering 3D structural information from 2D images is an active research area in computer vision and can be used in many applications such as virtual reality, robot navigation, environment simulation, architectural design, industrial manufacturing and multimedia.

In general, 3D reconstruction methods can be classified into categories: Multiple view approach and the Single view approach. This classification is based on the number of images used in reconstructing a 3D scene. In this section, We will discuss the 3D reconstruction methods from both approaches as 3D reconstruction from Multiple view approach is a classic and still an active research area whereas Single view approach has been attracting more interests recently. For the sake of completeness, an adequate review of multiple view reconstructions algorithms are presented in section 2.1. However, the reader may skip this section without an impact on readability of later sections.

2.1 Multiple View 3D Reconstruction

Traditional multiple view 3D reconstruction methods use feature/pixel correspondences between images (e.g. stereo images, or multiple images, or video sequences, and/or 3D range data) to reconstruct the 3D geometry or camera calibration [6, 7].

All the traditional 3D reconstruction requires special equipment, such as multiple cameras, range scanners [2].

Kolmogorov *et al.* proposed two 3D scene reconstruction methods where they formulated the 3D scene reconstruction problem as an energy minimization problem and optimized the energy with graph-cut algorithm [8, 9]. Both methods treat the input images symmetrically and can handle visibility constraints properly. The method proposed by Kolmogorov and Zabin in 2002 can handle arbitrary number of cameras but imposes the spatial smoothness while preserving discontinuities with respect to a single cameras [8]. The method proposed by Kolmogorov *et al.* in 2003 can impose the spatial smoothness while preserving discontinuities with respect to a large number of cameras [9].

Based on the above methods proposed by Kolmogorov and co-workers [8, 9], Goldlucke and Magnor [10] proposed a simultaneous 3D scene reconstruction and background separation method by adding one more term called background term in the energy function. The added background term introduces the penalty for each pixel being labeled as background where the penalty value is calculated based on the normalized cross-correlation of the image values with the background values as well as the depth information of background. The energy function proposed by Goldlucke *et al.* can still be optimized with the α-expansion based graph cut method proposed by Boykov *et al.* [11].

Yuan and Medioni introduced a novel method to obtain the 3D Euclidean reconstruction for both the moving objects and the background in a video sequence captured from a moving camera [12]. They treated the moving objects as static objects observed by a moving "virtual camera" with a linear constraint. In their method, a homography-based motion segmentation method is first applied to the video sequence to get the static background and motion blobs. Then, the classic

Structure from Motion (SFM) methods are used to estimate camera poses, the 3D shape of the moving objects, as well as the static background.

Cheung *et al.* proposed a method to increase the accuracy of 3D Visual Hull (VH) reconstruction through combining the Shape-From-Silhouette (SFS) technique with the stereo information [13]. In their method, they obtained multiple silhouette images captured across time. They first constructed a representation of VHs called the bounding edge representation. Then, they applied the multi-view stereo to extract points called Colored Surface Points (CSP) on the object surface based on one of the fundamental properties of VH: each bounding edge must touch the object at least one point. Those obtained CSPs were then used in a 3D image alignment algorithm to find the 6 DOF rigid motion between two VHs. Finally they used this rigid transformation to treat all the silhouette images as being captured at the same time, which increase the number of silhouette images used. Therefore, their method can improve the shape approximation than traditional SFS methods.

Liu *et al.* mapped the 2D image texture onto the 3D range data by integrating the multi-view geometry with automated 3D registration techniques [14]. They used the 3D range scans and the 2D photographic images to generate a pair of 3D models of the scene. The first model is a dense 3D point cloud produced by the range images through the a 3D-to-3D registration method. The second model is a sparse 3D point cloud produced by the sequence of 2D photographic images by using the Structure from Motion (SFM) method based on the multi-view geometry. Then, they developed a novel model alignment algorithm to recover the similarity transformation parameters (i.e. rotation, scale, and translation), which finds the best alignment between the dense and sparse models. Finally, these alignment parameters were used to get the optimal texture mapping from the 2D photographic images to the 3D dense model.

Shum *et al.* presented an interactive 3D reconstruction system by using a collection of panoramic image mosaics [15]. A panoramic mosaics is a set of images taken around the same view and associated with a transformation matrix. To generate a 3D scene, their system first recovered the camera pose for each mosaic based on line directions manually specified by a user and points and then constructed the 3D model using all available geometric constraints. In their method, they partitioned the constraints into soft and hard linear constraints, formulated the 3D modeling problem as a linearly-constrained least-square problem, and solved using QR factorization efficiently.

Debevec *et al.* recovered the 3D scenes from a sparse set of still images by using a hybrid geometry- and image-based approach which integrated the geometry-based modeling technique with the image-based rendering technique [16]. In their method, they first provided an easy-to-use interactive modeling interface which allowed the user to construct a geometric model of scene from images. Then, a view-dependent texture mapping method was used to project different images onto the model based on the user's viewpoint. Their method can recover accurate architectural models from a few photographs with a number of user-specified correspondences.

Jiang *et al.* introduced a panoramic 3D scene recovery method by using rotational stereo cameras with simple epipolar constraints [17]. They acquired two sampled spatio-temporal volumes by rotating two parallel stereo camera about a vertical axis with a constant velocity. Those two sampled spatio-temporal volumes consist of two sequential images captured by a uniform angular interval and can be resampled into a set of multi-perspective panoramas. They computed the depth map from four image pairs among 5 images (four panoramas and one original image) using a multi-baseline algorithm with a number of epipolar constraints.

Jiang and Lu also proposed a panoramic 3D scene reconstruction method by fus-

ing the image color intensity information with laser range data [18]. In their method, they acquired two types of panoramic scene data: a dense spatio-temporal volumes from the CCD camera and the distance information from the laser range scanner. They resampled the dense spatio-temporal volumes and estimated the dense-depth panoramic map. They then fused the estimated dense-depth map with the depth map measured using a laser range scanner by formulating the fusing problem as an energy minimization problem and optimized the energy with a belief propagation algorithm. In their energy function, they incorporated the active depth measurements obtained from a 2D laser range scanner with the passive geometry reconstruction obtained from the image sequence captured from the CCD camera. Their experimental results showed that by fusing the image color intensity information with laser range data, they can get a more robust and accurate reconstruction than either using geometry reconstruction or range scanned data alone [19].

There are some other methods which recover the sparse 2D features (such as points and line segments) into the 3D world through using geometric projection among multiple images [20, 21]. Some sparse 3D reconstruction methods apply the classic Structure from Motion (SFM) techniques to recover the sparse 3D world from video sequence [22, 23] while some other methods recover the sparse 3D information by finding the feature correspondences between stereo images [24, 7]. However, all these methods normally can not be directly used to get solid/dense scene reconstruction.

Although traditional 3D scene reconstruction based on multiple view geometry has been heavily studied in the last several decades, a reliable and automatic 3D reconstruction is still a distant goal. When the environment tends to be extremely low textured such as in some indoor environment, using projective geometry is usually unreliable as low texture makes the correspondence problem highly ambiguous.

2.2 Single View 3D Reconstruction

Unlike the traditional 3D reconstruction methods which can provide accurate metric 3D reconstruction, single view methods can only recover approximate 3D information as a single view of a scene does not have enough information about the depth of the scene points. However, the recovered approximate 3D information is very useful in many applications such as in scene visualization, object recognition, virtual reality, etc. In general, the single view reconstruction problem can be solved to some extent by either through user interaction or imposing constraints on the scene.

A general approach for single view methods is to formulate the 3D reconstruction problem as a pixel labeling problem by manually or automatically partitioning the single image into several geometrical parts/regions. Pixel labeling problem involves assigning a label from a finite set of possibilities to each image pixel.

Horry *et al.* developed a Tour Into the Picture (TIP) method to obtain a simple scene model from a single 2D image [3]. In their method, they used a spidery mesh to partition the input image into 5 geometric regions. To generate an animated 3D scene, users have to specify the vanishing point, background, and foreground objects manually using the graphical interface they provided. Thus this approach requires intensive user interaction.

Criminisi *et al.* extracted the geometric information and constructed the 3D models from single uncalibrated perspective images of a scene by using two canonical types of measurements: lengths of segments on planar surfaces and the distances of points from planes [25]. In their method, an image-to-world homography matrix was first estimated by computing the projective transformation between four manually measured world points with four manually selected corresponding points in the image. Therefore, lengths of the segments on planar surfaces can be computed based on the

estimated homography matrix. The distances of points from planes was computed based on the manually selected reference objects, the estimated vanishing point, and the vanishing line. In their method, meaningful objects (such as walls, people, etc.,) are segmented by using an interactive silhouette cut-out method. 3D reconstruction can be completed by repeatedly segmenting, measuring, and inserting the manually selected objects into the reconstructed 3D model with respect to the manually selected reference plane. As the authors stated, their method is especially suitable for man-made environment where there are frequent architectural elements and/or geometric patterns as they exploited orthogonality and parallelism constraints of the scene.

Johansson proposed a simple linear algorithm to reconstruct new views of piecewise planar objects [26]. This method assumes the scene to be piecewise planar patch and calculates the homographies for each patch by manually selecting the intersection lines between scene planes. As the author stated, this method is suitable for man-made environment such as buildings.

Prasad *et al.* proposed a 3D reconstruction method for curved 3D surface [27]. They formulated the 3D reconstruction as a global optimization problem by minimizing a surface smoothness objective function. In their method, they applied the thin-plate energy which was subjected to some constraints from the apparent contour (also called silhouette) as well as some user specified constraints, such as surface normals.

Zhang *et al.* introduced a novel approach to reconstruct the free-form texture-mapped 3D scene models from a single painting or photograph [28]. In their method, after user specified a sparse set of constraints on the local shape of the scene including surface position, normals,silhouettes, and creases, then a smooth 3D surface which satisfies these constraints was generated. In their method, the problem of computing the best surface that satisfy these constraints was cast as a constrained optimization

problem.

Colombo *et al.* presented a metric 3D reconstruction from a single uncalibrated view of Surfaces of Revolution (SOR) [29]. SOR represents a class of scene with repeated structure generated by the rotation of a planar curve around an axis. The presence of these repeated structures can be used as the geometric constraints. With repetitive structures, 3D models are comparably easy to compute. However, their method can only be used in objects with SOR, which are very common on some man-made objects.

Coughlan and Yuille proposed a method for labeling pixels with so called "Manhattan" orientations, which consists of vertical, horizontal, or depth directions [30]. In addition, they have the "clutter" label, for pixels inconsistent with the Manhattan orientations. An individual pixel likelihood for a particular orientation is based on the gradient. A Bayesian framework is used for spatial consistency between pixels. This method is not suitable for images containing textured scene, such as textured walls/floors, since the majority of the edges in such scenes would not be consistent with the Manhattan directions.

Delage *et al.* proposed a dynamic Bayesian network model to automatically recover 3D scene from a single 2D indoor image [31]. However, their method is based on recognizing the "floor-wall" boundary in each column of the image, and is not appropriate for scenes that do not have sufficiently large "floor-wall" boundaries.

Teruel *et al.* also presented a real time 3D reconstruction method for single indoor image by interpreting the indoor scene in terms of a set of vertical planes lying on a common horizontal plane [32].

Zhang and Tsui reconstructed an arbitrary 3D object from a single view of an object and its image in a plane mirror [33]. In their method, they calculated the correspondence between the object and its counterpart in the mirror as they were

auto-epipolar. However, only a partial object can be reconstructed as only a fraction of the object and its corresponding image in a plane mirror can be simultaneously visible in a single view.

All the methods discussed above either require user interaction or make relatively restrictive assumptions about the scene.

Recently, Hoiem *et al.* introduced a novel method that allows the automatic extraction of rough 3D structure from a single 2D image through learning [34, 4, 35]. Unlike most of the scene recognition algorithms [36, 37] which model semantic classes, such as cars, faces, etc., they reconstructed the geometric structure of scene regions through learning geometric labels, such as "vertical", "sky", "ground", based on the appearance of a region. Their method is very accurate and fast. Although it provides only a rough 3D description of a scene, such rough description is quite useful for scene visualization and object recognition. In a later publication, Hoiem *et al.* tried a global optimization for geometric labeling without improvement due to the fact that the optimization in their method is performed on superpixel level and not on pixel level [5]. Therefore in case when a superpixel contains pixels with different true labels, they can not assign the true labels to all the pixels in that superpixel.

Chapter 3

Preliminaries

This chapter reviews two image processing operations: directional filtering and Difference of Oriented Gaussian (DOOG), a machine learning algorithm called Support Vector Machines and the Graph Cut optimization algorithm, which provide the theoretical base for the proposed algorithm.

3.1 Image Processing

In image processing, feature extraction refers to methods that aim at computing abstractions of image information in order to make a local decision for a given task. Choosing discriminating and independent features (i.e. good features) is one of the most critical aspects in machine learning and pattern recognition in order to get a successful classification [38, 39]. Image features are measurable heuristic properties or information extracted from the image. Widely used good features include image edges, corners, ridge and blobs [40, 41].

In this section, I will describe two low level feature extraction methods which are used in our proposed method for training/learning, namely the directional filtering and Difference-of-Oriented-Gaussian (DOOG).

3.1.1 Directional filtering

Directional filtering is a bank of filtering operations which are normally used to extract image edges in different directions. Fig. 3.1 shows a directional filtering operation called compass operator [42]. A convolution operation with the compass operator calculates the second derivative of intensity in four orientations 0^o, 45^o, 90^o and 135^o, where 0^o denotes horizontal direction, 90^o denotes vertical direction and 45^o and 135^o are 45^o and 135^o diagonal directions, respectively. Fig. 3.2 demonstrates an original test image and its corresponding results of four directional filters.

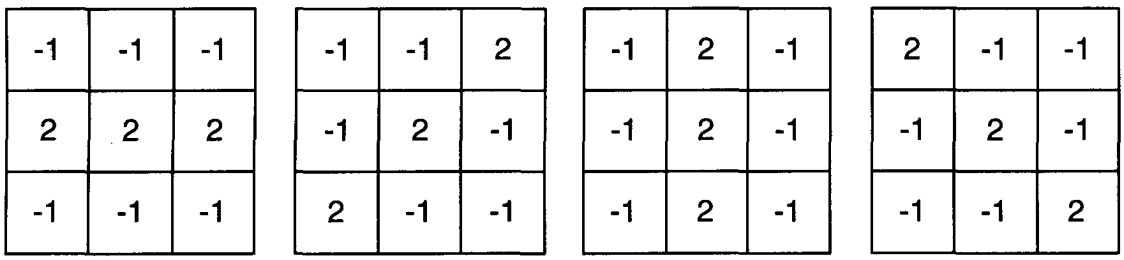


Figure 3.1: Directional filters, (a) 0^o, (b) 45^o, (c) 90^o, (d) 135^o.

3.1.2 Difference-of-Oriented-Gaussian(DOOG)

The Difference of Oriented Gaussian DOOG is an operation which calculates the difference between two oriented Gaussians of different sizes with the width of positive Gaussian being smaller than the width of the negative one [43, 44]. DOOG is normally used for texture analysis in computer vision.

Let $G(x, y, \sigma_x, \sigma_y)$ be the two-dimensional Gaussian function defined in Eq. 3.1,

$$G(x, y, \sigma_x, \sigma_y) = A \exp^{-(\frac{(x-x_0)^2}{2\sigma_y^2} + \frac{(y-y_0)^2}{2\sigma_y^2})} \tag{3.1}$$

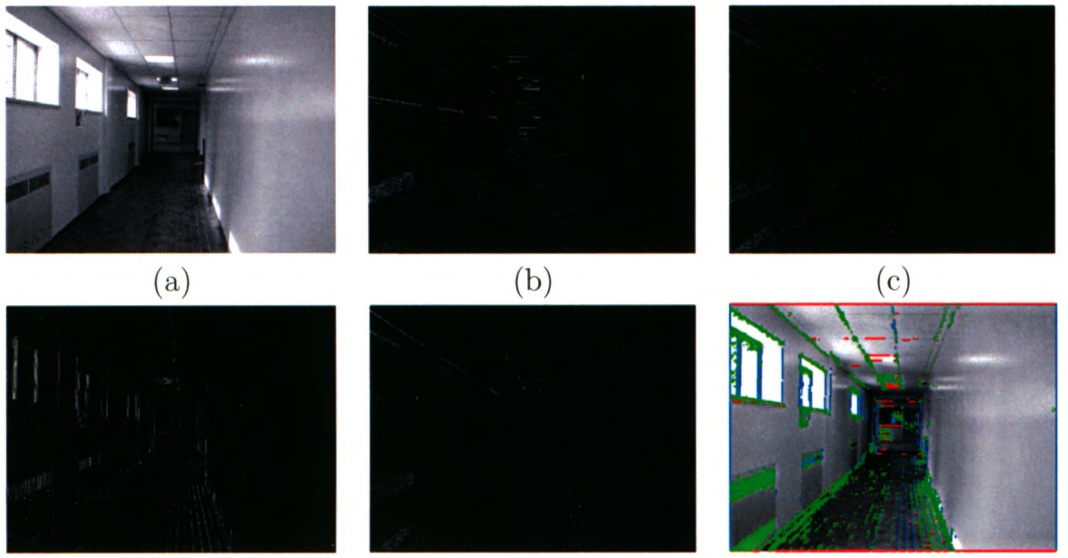


Figure 3.2: Directional filtering (a) original image (b) 0^o (c) 45^o (d) 90^o (e) 135^o (f) edge overlayed image.

where σ_x, σ_y are the variance in x and y direction, respectively, which denote the width of the Gaussian function. The oriented Gaussian $G(x, y, \sigma_x, \sigma_y, \theta)$ is the two-dimensional Gaussian $G(x, y, \sigma_x, \sigma_y)$ rotated by an angle θ defined in Eq. 3.2,

$$G(x, y, \sigma_x, \sigma_y, \theta) = A \exp^{-(a(x-x_0)^2+b(x-x_0)(y-y_0)+c(y-y_0)^2)} \tag{3.2}$$

where,

$$a = (\frac{\cos\theta}{\sigma_x})^2 + (\frac{\sin\theta}{\sigma_y})^2,$$

$$b = -\frac{\sin 2\theta}{\sigma_x^2} + \frac{\sin 2\theta}{\sigma_y^2},$$

$$c = (\frac{\sin\theta}{\sigma_x})^2 + (\frac{\cos\theta}{\sigma_y})^2.$$

The DOOG is the difference between two oriented Gaussians which is defined in

Eq. 3.3,

$$Doog(x, y, \sigma_x, \sigma_y, \theta, r) = K_1 \cdot G(x, y, \sigma_x, \sigma_y, \theta) - K_2 \cdot G(x, y, r \cdot \sigma_x, r \cdot \sigma_y, \theta) \quad (3.3)$$

where $G(x, y, \sigma_x, \sigma_y)$ denotes two-dimensional oriented Gaussian function, and K1 and K2 denote positive constants. Fig. 3.3 and Fig. 3.4 demonstrate the 2D Oriented Gaussian and DOOG with 12 orientations respectively.

Fig. 3.5 shows the corresponding DOOG filtering results for the same testing image as shown in Fig. 3.2.

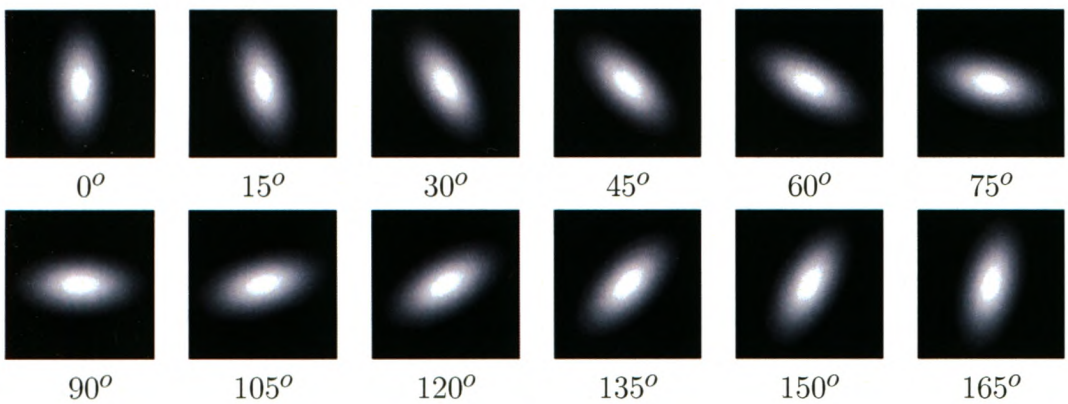


Figure 3.3: Oriented Gaussian.

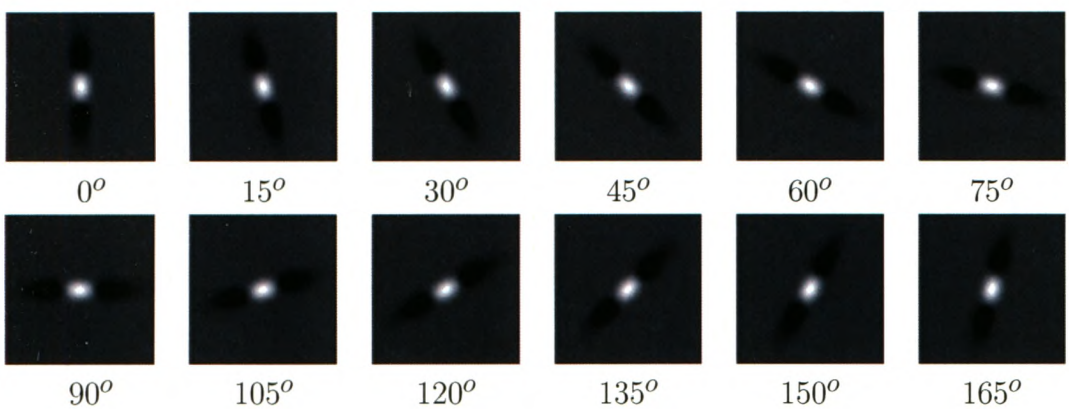


Figure 3.4: Difference of Oriented Gaussian filters.

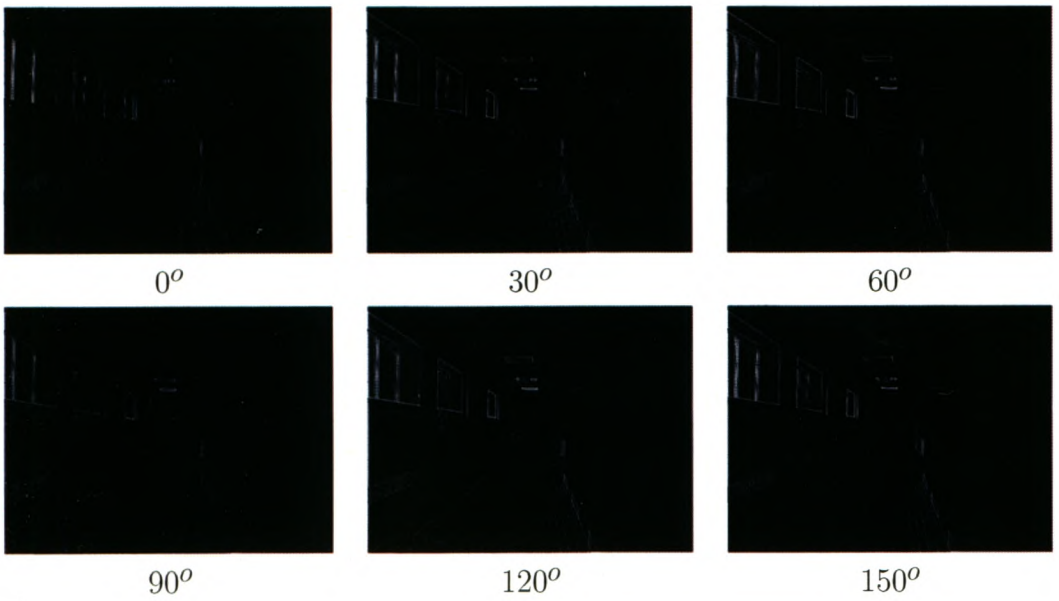


Figure 3.5: DOOG filtering for the same testing image as shown in Fig. 3.2.

3.2 Support Vector Machine (SVM)

Support Vector Machine is a very popular technique for classification. A classification task usually involves training and testing data consisting of data instances, where each data instance in the training set contains a set of features or attributes and a target value, also called class label. The goal of SVM is to find the optimal hyperplanes which can separate classes within the training data set with the largest margin as shown in Fig. 3.6. When the classes are not linearly separable, SVM maps the input data from a low dimensional space into a high-dimensional space where the mapped data are linearly separable. The mapping function is called kernel function. Fig. 3.7 shows an example of data mapping, where the data are not linearly separable in the two-dimensional space. But after mapping, they are linearly separable in the three-dimensional space.

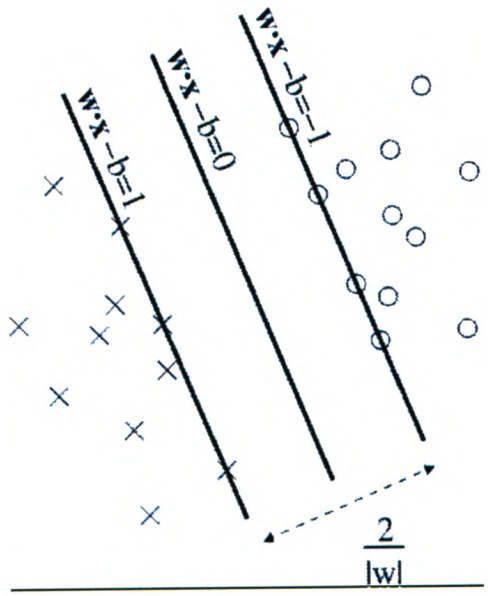


Figure 3.6: Hyperplane of SVM (maximum margin in linearly separable case): the vector w is the normal of the separating hyperplane and b is the offset parameter, which allows the margin increase

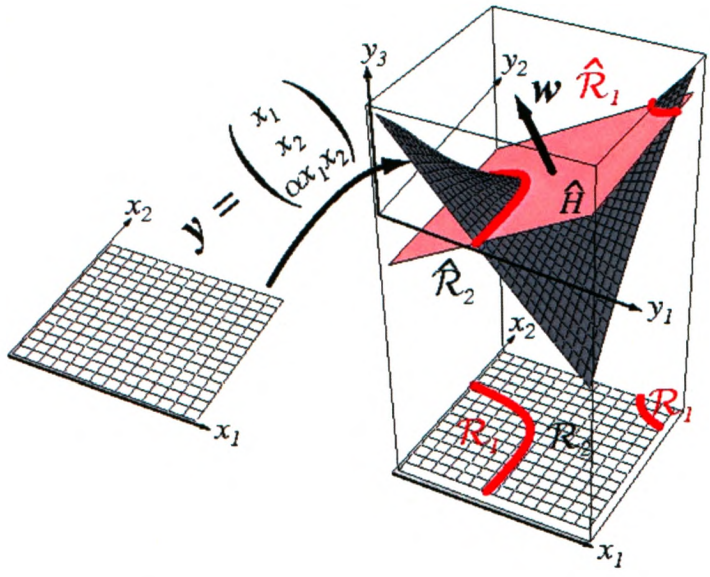


Figure 3.7: SVM kernel function [a]

a. This diagram was reproduced from the textbook titled Pattern Classification [45].

3.2.1 Classification

In this section, we review the most well known SVM method proposed by Vladimir Vapnik in [46, 47].

Let (x_i, y_i) be the instance and label pair, where $x_i \in R^n$, $y_i \in \{1, -1\}$, and $i = 1, \cdots, l$ is the number of instances. SVM finds the optimal hyperplane $w \cdot x - b = 0$, with the largest margin $w \cdot x - b = \pm 1$. The vector w is the normal of the separating hyperplane and b is the offset parameter, which allows the margin increase as shown in Fig. 3.6.

Vapnik introduced the concept of soft margin through applying the slack variables [46]. As shown in Fig. 3.8, by using the slack variables, it allows mislabeled examples, which is suitable in the case when data are not linearly separable even in the high-dimensional space.

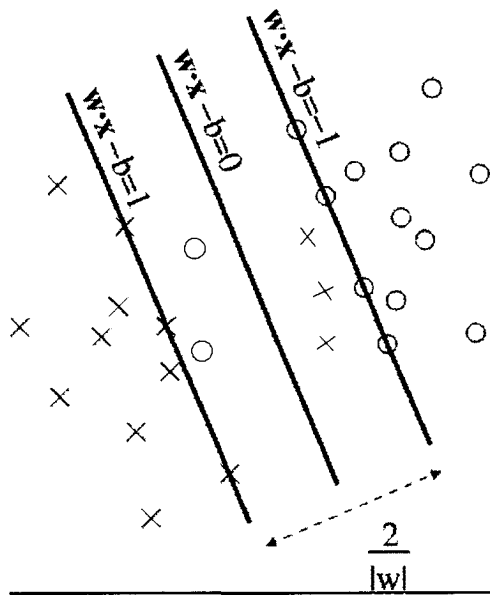


Figure 3.8: Soft margin of SVM: slack variables used in linearly not separable case.

The SVM algorithm introduced by Vapnik is summarized in algorithm 1:

Algorithm 1 (SVM)

Given: a training set of instance–label pairs (x_i, y_i), where $i = 1, \cdots, l$, $x_i \in R^n$, and $y \in \{-1, 1\}^l$

Solving: the optimization problem

$$\min_{\omega, b, \xi} \frac{1}{2}\omega^T\omega + C\sum_{i=1}^{l}\xi_i$$

subject to

$$y_i(\omega^T\phi(x_i) + b) \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

In the above equation, $\phi(\cdot)$ is a function, which maps feature vectors x_i into a higher (even infinite) dimensional space, and ξ_i are slack variables, which allow mislabeled examples through soft margin. $C > 0$ is the penalty parameter of the error term.

The kernel function $K(x_i, x_j)$ is defined as follows:

$$K(x_i, x_j) \equiv \phi(x_i)^T\phi(x_j).$$

Through the use of kernel functions, computation in the higher dimensional case in SVM is performed only implicitly. Therefore, it avoids the curse of dimensionality problems. The following are the most commonly used kernel functions:

(i) Linear: $K(x_i, x_j) = x_i^T x_j$.

(ii) Polynomial: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d, \gamma > 0$.

(iii) Radial Basis Function (RBF): $K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \gamma > 0$.

(iv) Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$.

Here, γ, r, and d are kernel parameters.

3.2.2 Probabilistic outputs

The output of SVM is an uncalibrated value and not a probability distribution of a label given the feature vector [46]. However, in practical recognition situations, the posterior probability $P(class|input)$ is very useful. Although the standard SVM methods do not provide such probabilities, there are ways that can be used to convert uncalibrated SVM values into probabilities [48, 49].

Here, we describe a popular approach proposed by Platt, who proposed to approximate the posterior probability through fitting a parametric sigmoid function on the uncalibrated SVM outputs [48]. The sigmoid function maps the SVM outputs to posterior probabilities. Let the unthresholded output of an SVM be

$$f(x) = h(x) + b \tag{3.4}$$

where

$$h(x) = \sum_i y_i \alpha_i K(x_i, x) \tag{3.5}$$

Platt fit the sigmoid function

$$Pr(y = 1|x) \approx P_{A,B}(x) \equiv \frac{1}{1 + \exp(Af(x) + B)} \tag{3.6}$$

Where the best parameter (A,B) are then estimated by solving the following regularized maximum likelihood problem with a set of labeled examples $\{(x_i, y_i)\}_{i=1}^{l}$

$$\min_{(A,B)} F(A, B) = -\sum_{i=1}^{l} (t_i \log(p_i) + (1 - t_i) \log(1 - p_i)), \tag{3.7}$$

where

$$p_i = \frac{1}{1 + \exp(Af(x_i) + B)}$$

and

$$t_i = \begin{cases} \frac{N_+ + 1}{N_+ + 2} & \text{if } y_i = +1, \\ \frac{1}{N_- + 2} & \text{if } y_i = -1\text{,i=1,...l.} \end{cases}$$

where N_+ and N_- are the numbers of positive and negative examples of y_i, respectively.

So far, we have discussed binary or two-class classification, i.e $y_i \in \{1, -1\}$. For a multi-class classification problem, where $y_i \in \{1, 2, ...k\}$, the classification involves assigning each of the instance into one of the k classes. Since two-class problems are much easier to solve, many authors propose to use two classifier for multi-class classification through pairwise coupling [50, 51]. Pairwise coupling combines all comparisons for each pair of class. Voting is a common way to combine pairwise comparisons [52].

3.3 Graph Cut Optimization

In this section, we briefly give a review on the graph cuts algorithm proposed by Boykov *et al.* [11]. Let $\mathcal{P}$ be the set of all pixels in an image, $\mathcal{L}$ be a finite label set $\mathcal{L} = \{l_1, l_2, \cdots\}$, and f_p be a label assigned to a pixel p (i.e. $p \in \mathcal{P}$, $f_p \in \mathcal{L}$). Let $f = \{f_p | p \in \mathcal{P}\}$ be the collection of all pixel/label assignments, the graph cuts algorithm minimizes the following energy function:

$$E(f) = \lambda \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{(p,q) \in \mathcal{N}} V_{p,q}(f_p, f_q) \tag{3.8}$$

In Eq. 4.1, $D_p(f_p)$ and $V_{p,q}(f_p, f_q)$ are called the data and the smoothness terms, respectively, and $\mathcal{N}$ is a neighborhood system on $\mathcal{P}$.

The data term $D_p(f_p)$ specifies the penalty for pixel p to have label f_p, and thus encourages each pixel to be assigned the label of smallest penalty. The smoothness term $V_{pq}(f_p, f_q)$ encourages spatial consistency by penalizing neighboring pixels p and q that are not assigned the same label.

It has been proved that given a finite set of labels $\mathcal{L}$, when the smoothness term is a discontinuity preserving interaction term, the global minimization of the energy functions above in Eq. 4.1 is NP-hard, even in the simplest discontinuity preserving case [11].

However, discontinuity preserving interaction is one of the most common situations in vision problems, which assumes that features vary smoothly within the objects but vary dramatically at object boundaries.

For example, for Potts model, $V_{p,q}(f_p, f_q) = 0$ if $f_p = f_q$ and $V_{p,q}(f_p, f_q) = w_{pq}$ if $f_p \neq f_q$, where w_{pq} is a positive coefficient that can depend on the particular pixel pair (p, q). Typically w_{pq} is small if there is an intensity edge between pixels p and q, and w_{pq} is large otherwise. This encourages the discontinuities between labels to align with the image edges.

Boykov*et al.* developed the $\alpha - \beta$ swap algorithm and the α-expansion algorithm [11], based on the efficient max-flow algorithm [53]. The α-expansion and the $\alpha - \beta$ swap algorithm are briefly described in algorithm 2 and 3 respectively.

Algorithm 2 (α-expansion algorithm)

1: **Start:** an arbitrary labeling f
2: **Set:** $success = 0$
3: **for** each label $\alpha \in \mathcal{L}$ **do**
4: • Find $\hat{f} = argmin\mathrm{E}(\mathrm{f}^{'})$ among $f^{'}$ within one α-expansion of f
5: • if $\mathrm{E}(\hat{\mathrm{f}}) < \mathrm{E}(\mathrm{f}^{'})$, set $f := \hat{f}$ and $success = 1$
6: **end for**
7: If $success = 1$ goto 2
8: Return f

Algorithm 3 ($\alpha - \beta$ swap algorithm)

1: **Start:** an arbitrary labeling f
2: **Set:** $success = 0$
3: **for** each pair of labels $\{\alpha, \beta\} \subset \mathcal{L}$ **do**
4: • Find $\hat{f} = argmin\mathrm{E}(f')$ among f' within one $\alpha - \beta$ swap of f
5: • if $\mathrm{E}(\hat{f}) < \mathrm{E}(f')$, set $f := \hat{f}$ and $success = 1$
6: **end for**
7: If $success = 1$ goto 2
8: Return f

Both the $\alpha - \beta$ swap algorithm and the α-expansion algorithm can achieve approximate solutions to this NP-hard minimization problem with guaranteed optimality bounds. For Potts model, in case of two labels, the energy in Eq. 4.1 can be minimized exactly, and in the multi-label case a solution that is optimal within a factor of two can be found with the α-expansion.

Given the following conditions for any labels $\alpha, \beta, \gamma \in \mathcal{L}$:

$$V_{p,q}(\alpha, \beta) = 0 \Leftrightarrow \alpha = \beta, \tag{3.9}$$

$$V_{p,q}(\alpha, \beta) = V_{p,q}(\beta, \alpha) \geq 0, \tag{3.10}$$

$$V_{p,q}(\alpha, \beta) \leq V_{p,q}(\alpha, \gamma) + V_{p,q}(\gamma, \beta). \tag{3.11}$$

If $V_{p,q}$ satisfies all the above three conditions in Eq. 3.9 to Eq. 3.11, it is called metric. If $V_{p,q}$ only satisfies the conditions in Eq.(3.9) and Eq.(3.10), it is called semimetric.

The α-expansion algorithm works under the condition that $V_{p,q}$ is metric, while the $\alpha - \beta$ swap algorithm works if $V_{p,q}$ is semimetric. Therefore, the $\alpha - \beta$ swap algorithm has more relaxing conditions, while α-expansion is computational more efficient.

Chapter 4

Geometric Class Scene Labeling

4.1 Graph Cut with Order-Preserving Moves

In this section we describe the graph-cut optimization framework with our order-preserving moves. Suppose we have a pixel labeling problem where the task is to assign to each image pixel p some label from a finite label set $\mathcal{L}$. Let $\mathcal{P}$ be the set of all pixels in an image, and f_p be a label assigned to a pixel p (i.e. $p \in \mathcal{P}$, $f_p \in \mathcal{L}$). Let $f = \{f_p | p \in \mathcal{P}\}$ be the collection of all pixel/label assignments. The energy function is:

$$E(f) = \lambda \sum_{p \in P} D_p(f_p) + \sum_{(p,q) \in \mathcal{N}} V_{pq}(f_p, f_q) \tag{4.1}$$

Eq. (4.1) is the commonly used energy function for graph cut optimization, where $D_p(f_p)$ and $V_{pq}(f_p, f_q)$ are called the data and the smoothness terms, respectively, and $\mathcal{N}$ is a neighborhood system on $\mathcal{P}$. We use the standard 4-connected $\mathcal{N}$ which consists of ordered pixel pairs (p, q). We assume the following image indexing: if p is to the left of q, then $p < q$, and if p is above q, then also $p < q$.

The data term $D_p(f_p)$ specifies the penalty for pixel p to have label f_p, and thus encourages each pixel to be assigned the label of smallest penalty. The smoothness term $V_{pq}(f_p, f_q)$ encourages spatial consistency by penalizing neighboring pixels p and q that are not assigned the same label.

For example for Potts model, $V_{pq}(f_p, f_q) = 0$ if $f_p = f_q$ and $V_{pq}(f_p, f_q) = w_{pq}$ if $f_p \neq f_q$, where w_{pq} is a positive coefficient that can depend on the particular pixel pair (p, q). Typically, $w_{pq} = b \cdot \exp(-\frac{(I_p - I_q)^2}{\sigma^2})$, where I_p and I_q are the image intensity values for pixel p and q, respectively. Therefore, w_{pq} is small if there is an intensity edge between pixels p and q, and w_{pq} is large otherwise. This encourages the discontinuities between labels to align with the image edges.

For Potts model, the graph-cut based α-expansion [11] performs best in terms of speed and accuracy [54] when compared to other popular minimization methods such as Tree-reweighted max-product message passing (TRW) [55] and Belief Propagation (BP) [56].

In case of two labels, the energy in Eq. (4.1) can be minimized exactly, and in the multi-label case a solution that is optimal within a factor of two can be found with the α-expansion [11]. The α-expansion finds a local minimum with respect to expansion moves. Given a labeling f and a label α, a move from f to f^α is called an α-expansion if $f_p \neq f_p^\alpha \Rightarrow f_p^\alpha = \alpha$, i.e the set of pixels labeled as α "expands" in f^α. The optimal α-expansion can be found efficiently using a min-cut/max-flow algorithm [53]. The α-expansion algorithm iterates over all labels α, finding the best α-expansion, until convergence.

In addition to spatial consistency, V_{pq} can also be used to incorporate a smoothness prior on a labeling, such as ordering constraints on labels. For example, if p is immediately to the left of q, to prohibit $f(p) =$ "*center*" and $f(q) =$ "*left*", we set $V_{pq}($"*center*", "*left*"$) = K$, where K is a prohibitively large constant. For example, K can be set equal to $E(f^{center})$, where f^{center} is a labeling which assigns the "center" label to every image pixel. Notice that $E^f center$ only involves the data terms and does not involve any no smoothness terms. Hence our definition of K is not recursive.

Ordering constraints arise naturally in real scenes (for example, "sky" and "ground", "left wall" and "right wall", etc.) and can be used for many applications. An example of ordering constraint is prohibiting a pixel with a "car wheel" label to be above a pixel with a "car roof" label. However, after adding ordering constraints to Potts model, the factor of 2 approximation no longer holds. We observe that the commonly used graph-cut based α-expansion is more likely to get stuck in a local minimum when ordering constraints are used. Researchers who used ordering constraints can not achieve good results with α-expansion alone [57, 58]. This motivates us to develop new graph-cut moves which we call *order-preserving* moves. Unlike α-expansion, order-preserving moves act on all labels. Although the global minimum is still not guaranteed, we will show that optimization with order-preserving moves performs significantly better than α-expansion.

In the geometric scene labeling, we assume that an image is to be segmented into five geometric parts (each part corresponds to a label), which we call "center", "left", "right", "top", "bottom". For compactness, we abbreviate label names with their first letter, i.e. L, R, T, B, C, respectively. The smoothness terms V_{pq} are in Table4.1. The model in Table 4.1 is Potts plus the ordering constraints. Under this model, a labeling has a finite energy only if the "center" part is a rectangle, and the "left", "right", "top", "bottom" parts are to the left, right, above, below the "center" part, respectively. Above model is based on the assumption that the photographs are taken with the typical camera orientation (the camera image plane is perpendicular to the floor for indoor images or ground for outdoor images). In the case when the camera image plane is not perpendicular to the floor/ground, we can use fourier transform to detect the image orientation and rotate it back to the typical camera orientation.

Table 4.1: Smoothness terms V_{pq}

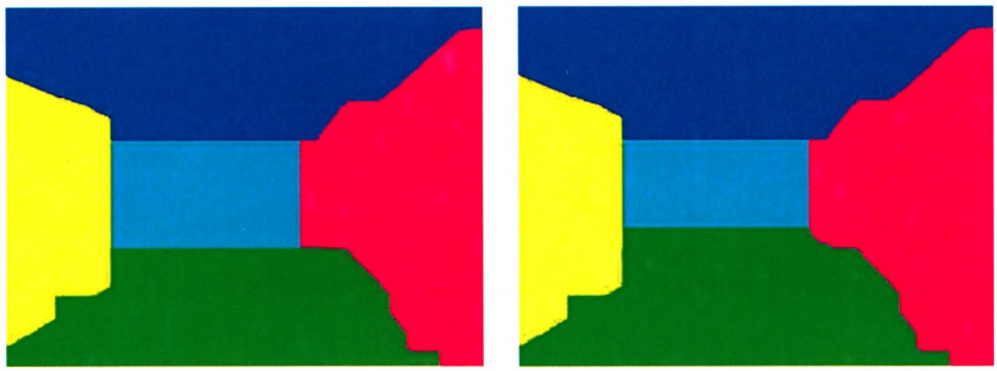
Horizontal Neighbors $p=(x,y), q=(x+1,y)$					
$f_p \backslash f_q$	L	R	C	T	B
L	0	∞	w_{pq}	w_{pq}	w_{pq}
R	∞	0	∞	∞	∞
C	∞	w_{pq}	0	∞	∞
T	∞	w_{pq}	∞	0	∞
B	∞	w_{pq}	∞	∞	0

Vertical Neighbors $p=(x,y), q=(x,y+1)$					
$f_p \backslash f_q$	L	R	C	T	B
L	0	∞	∞	∞	w_{pq}
R	∞	0	∞	∞	w_{pq}
C	∞	∞	0	∞	w_{pq}
T	w_{pq}	w_{pq}	w_{pq}	0	∞
B	∞	∞	∞	∞	0

Fig. 4.1 and Fig. 4.2 show how it is easier to get stuck in a local minimum using alpha-expansion than our order-preserving moves for a same pixel labeling problem when ordering constraints are added to the Potts model.

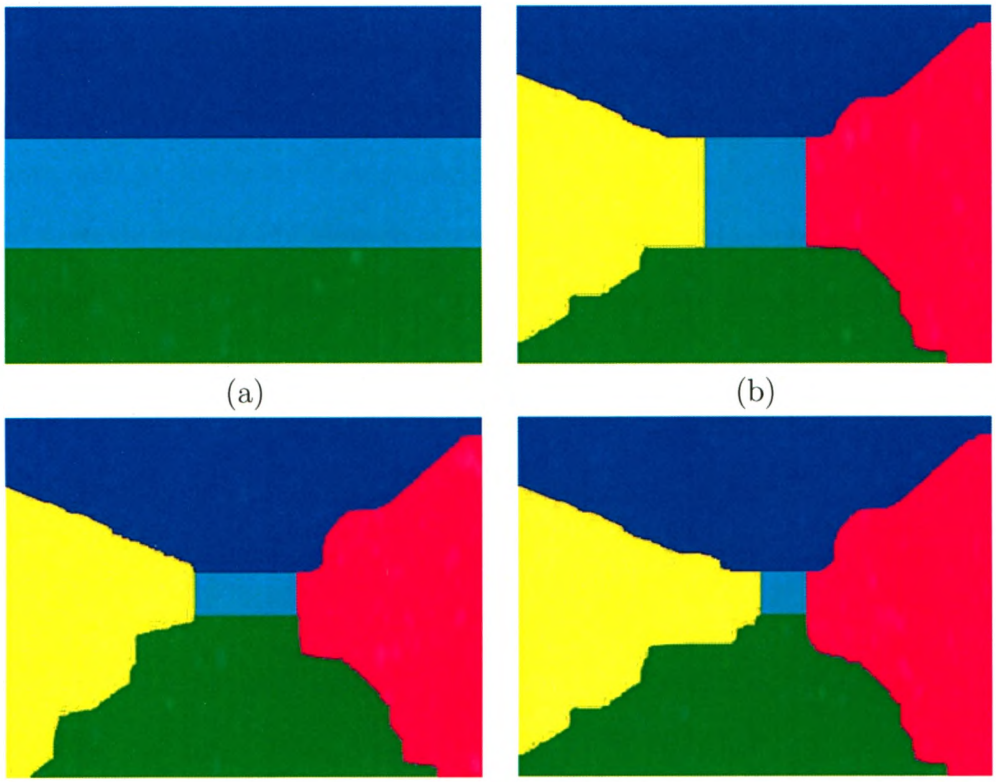
Consider Fig. 4.1, which shows the results of α-expansion for an instance of a geometric class labeling problem. Fig. 4.1(a) shows the labeling after one iteration, where one iteration means one α-expansion for each label $\alpha \in \{L, R, T, B, C\}$. Fig. 4.1(b) shows the labeling after two iterations. Only the B region expands from (a) to (b), and the algorithm, in fact, converges after 2 iterations. However, the labeling in Fig. 4.1(b), which has energy of $1,590,159$, is far from the optimum.

Fig. 4.2(d) shows the labeling (found by our order-preserving moves algorithm) that has a much better energy of $1,443,150$. The problem with the local minimum in Fig. 4.1(b) can be described as follows: To get to a better labeling, a smaller C region is needed. However, expansion on the C label will never shrink it. Labels B, T, L, and R need to expand, each one separately, to obtain a smaller C region. However, each individual expansion on the B, T, L, R does not result in a lower energy, and so the expansion algorithm gets stuck in a local minimum. We also show experimentally in Chapter 5 that the energies obtained by the order-preserving moves are significantly better than those of α-expansion.



(a) after one iteration (b) after two iterations

Figure 4.1: Results with α-expansion. Initial labeling, not shown, is all pixels labeled as "center". Color scheme: green = "bottom", yellow = "left", cyan = "center", magenta = "right", blue = "top". This color scheme is consistent throughout the thesis.



(a) (b)

(c) (d)

Figure 4.2: Results with order-preserving moves on the same problem as in Fig. 4.1. Initial labeling (not shown) was all "center", (a) vertical move from the initial labeling, (b) horizontal move applied to (a), (c) vertical move applied to (b), (d) horizontal move applied to (c).

In order to improve on α-expansion moves in the presence of ordering constraints, we should allow a pixel to have a choice of labels to switch to as opposed to just a single label α. Let L_p be a subset of labels that pixel p is allowed to switch to in one move. Typically, graph-cut algorithms use the same rule for choosing L_p for every pixel. For α-expansion, L_p consists of α and the old label of pixel p. For α-β swap [11], $L_p = \{\alpha, \beta\}$. For global optimization methods, $L_p = \mathcal{L}$, but they can handle only a restricted type of energies, and ours is not of that type [59, 60].

Our insight is that by using different rules when selecting L_p for different pixels, we can have a larger L_p for each pixel, as compared to α-expansion. i.e. In a single move, for each pixel, there are more labels to choose from. Notice that the choice of L_p precisely defines the allowed moves. i.e. A move from f to f' is allowed if $f'_p \in L_p$. Therefore, we must select L_p's in such a way that the allowed move of minimum energy can be computed efficiently. In addition, L_p must have the old label of pixel p, so that the set of allowed moves contains the old labeling and therefore the best allowed move is not worse than the old labeling. We found two such moves and call them *horizontal* order-preserving and *vertical* order-preserving.

Fig. 4.2 shows labelings for order-preserving moves on the same example of geometric labeling as in Fig. 4.1. A horizontal move (from Fig. 4.2 (a) to Fig. 4.2(b)) allows any change in labels except the region labeled as C cannot change its height. Either increase or decrease in width of the C region is allowed. The name "horizontal" reflects the fact that the C region can change in the horizontal, but not in the vertical direction. Similarly, a vertical move (from Fig. 4.2(b) to Fig. 4.2(c)) allows any change in labels except the region labeled as C cannot change its width.

Let f be a labeling, and x_p the horizontal coordinate of pixel p. Let $\underline{x}$ be the smallest x coordinate of any pixel that has label C in f, that is $\underline{x} = \min\{x_p | f_p = C\}$. Similarly, let $\overline{x}$ be the largest x coordinate of any pixel that has label C in f, that is

$\overline{x} = \max\{x_p | f_p = C\}$. Recall that L_p is the set of allowed labels that p can switch to in a single move. It is easy to see that for a vertical move, the following rules apply. If $p_x < \underline{x}$, then $L_p = \{T, L, B\}$. If $\underline{x} \leq p_x \leq \overline{x}$, then $L_p = \{T, C, B\}$. Finally, if $p_x > \overline{x}$, then $L_p = \{T, R, B\}$. In other words, divide f into three rectangles with two vertical lines, one passing through the border of the L and C regions and the other passing through the border of C and R regions. Then pixels in the left rectangle can switch their labels to T, L or B. Pixels in the middle rectangle can switch their labels to T, C or B, and finally pixels in the right rectangle can switch their labels to T, R or B.

To find an optimal vertical move, we use a very important result from Schlesinger [60]. They define a submodular energy in the case of multiple *ordered* labels and give a graph construction that can be used to optimize a submodular energy globally with the minimum cut. An energy is submodular if every V_{pq} term is submodular [60]. In turn, V_{pq} is submodular, if for any $\alpha < \beta$, and $\alpha' < \beta'$, $V_{pq}(\alpha, \alpha') + V_{pq}(\beta, \beta') \leq V_{pq}(\alpha, \beta') + V_{pq}(\beta, \alpha')$. See also [61] for an equivalent result. It is easy to check that the vertical move energy with V_{pq}'s in Table 4.1 and label order $T < L < B$, $T < C < B$, and $T < R < B$ is submodular. Notice that we do not have to order labels L, C, R with respect to each other because a single pixel under vertical move never has to choose between L, C, and R. There is no way to order all labels L, C, R, B, T so that our energy is submodular. Thus the main idea behind our moves is choosing L_p's for each p in such a way that the energy function restricted to the corresponding move is submodular. Due to symmetry, horizontal moves are handled similarly to vertical. In practice, we compute the optimal horizontal move by transposing the image, swapping labels L and T, R and B, and computing the optimal vertical move. Thus, an order-preserving move gives every pixel a choice of 3 labels to switch to, while α-expansion gives a choice of only 2 labels. In addition, α-expansion effectively acts on only one label, since only α label is allowed to increase

its territory during the move. Our moves act on all labels simultaneously, since any label has a chance to increase (as well as shrink) its territory during a single move.

We compute a local minimum with respect to the order-preserving moves. We start with an initial labeling and alternate between the best vertical and horizontal order-preserving moves until no move that would result in a lower energy can be found. The initial labeling has to be order-preserving. In practice, we start with all pixels labeled as C. Fig. 4.2 shows the sequence of labelings that we obtain under the order-preserving moves from the first one in Fig. 4.2(a) to the one at the convergence after 4 steps in Fig. 4.2(d). For the first move in Fig. 4.2(a), no L and R labels are allowed, since the initial labeling has all pixels labeled as C. That is why the labeling in Fig. 4.2(a) appears as horizontal bands where there is no L and R labels at all. For the order-preserving moves, we use the efficient max-flow algorithm [53] for min-cut computation.

4.2 Proposed Method

In this section, we apply the order-preserving moves to the geometric class scene labeling. Here, the goal is to automatically extract a coarse 3D scene structure from a single 2D image by assigning each image pixel its rough geometric label, such as "sky", "ground", etc. Unlike previous approaches, we formulate the problem in a global optimization framework using the five part model discussed in Sec. 4.1 and optimizing the energy in Eq. (4.1) with order-preserving moves.

Our label set is $\mathcal{L} = \{bottom(\mathrm{B}), left(\mathrm{L}), center(\mathrm{C}), right(\mathrm{R}), top(\mathrm{T})\}$. The V_{pq} terms in Eq. (4.1) are as in Sec. 4.1. The set of ordering constraints from Sec. 4.1 ensures that the boundaries between the parts agree with the directions caused by the perspective effects under the standard camera orientation. In other words, the

boundary between the "left" and "bottom" parts is a diagonal, slanted down and to the left. We are also able to align the boundaries between the geometric labels with the intensity edges in the image. For the data terms in Eq. (4.1), we train a classifier in a manner similar to Hoiem *et al.* [4]. The details of this method are described are in Sec. 4.2.1.

4.2.1 Data term

According to the literature on Markov random field (MRF) [62], we would like to model the data term $D_p(f_p)$ as

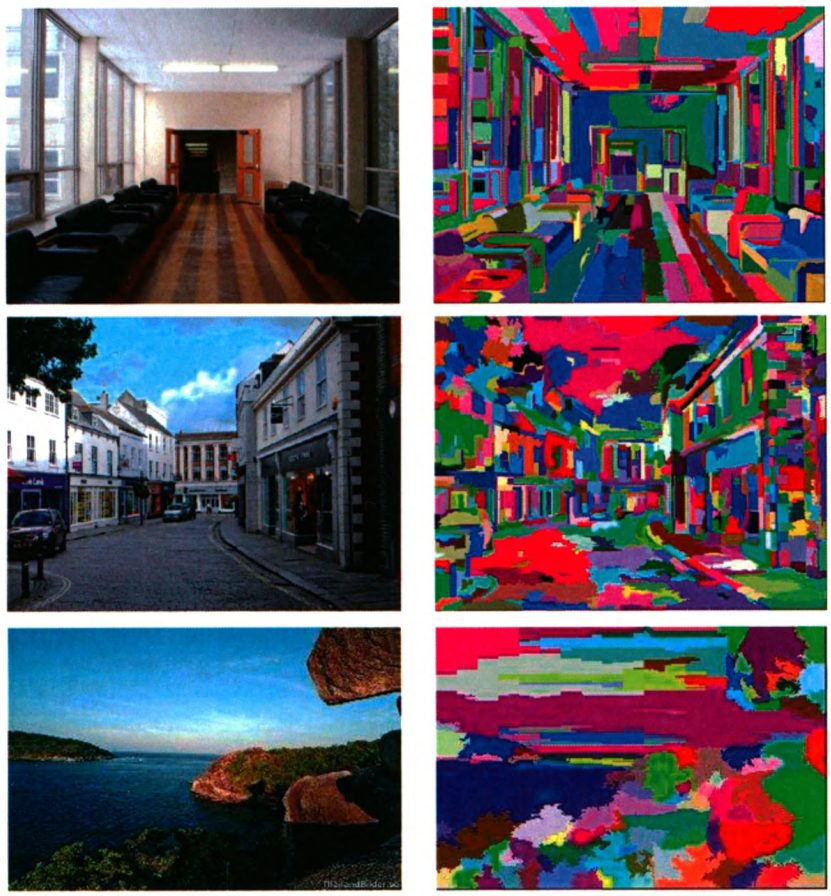
$$D_p(f_p) = -\log Pr(f_p|F_p), \tag{4.2}$$

where F_p is some observed feature vector at pixel p and $Pr(f_p|F_p)$ is the conditional probability of pixel p given feature F_p. However, for the geometric labels used in this thesis, image data at a single pixel does not contain enough information to construct a useful likelihood model in Eq. (4.2).

We take an approach very similar to Hoiem *et al.* [4] who observe that an image region frequently does contain enough data to reliably classify it with a geometric label. We first partition images into "superpixels"[1] by a segmentation algorithm proposed by Felzenszwalb *et al.* [63], where they partitioned an image into a number of segments based on the image color information. Fig. 4.3 shows that the partitioned image regions can interpret more intuitively discriminant features than traditional image grids. Choosing the size of superpixel is critical in our method as if the size is too small, the calculated features are not discriminant whereas if the size is too

1. Superpixel is simply an image region returned by a segmentation algorithm.

large, it is difficult to break up the superpixel. In our implementation, the size of superpixels is chosen by visually looking at the partitioning results on a small set of training images.



(a) (b)

Figure 4.3: Superpixel: (a) original images, (b) Segmented superpixels: different colors correspond to different superpixels.

Then, we compute a large set of features (listed in Table 4.2) for each superpixel. The features that we used are similar to the features used by Hoiem *et al.* [4], which are the statistics on location, color, geometry, texture and edges of the generated superpixels. We did not perform feature analysis as the performance is acceptable. These statistics include the location of the centroid, 10^{th} and 90^{th} percentile of the

Table 4.2: **Features**

Feature Descriptions	**No.**
Location	6
L1. Centroid: x,y	2
L2. x,y: 10^{th} and 90^{th} percentile	4
Geometry	4
G1. Orientation	1
G2. Shape: ratio of MajorAxis/MinorAxis	1
G3. Eccentricity	1
G4. Area	1
Edges	4
E1. Compass Filters: mean	4
Color	7
C1. Intensity value: mean	1
C2. R value: mean	1
C3. G value: mean	1
C4. B value: mean	1
C5. H value: mean	1
C6. S value: mean	1
C7. V value: mean	1
Texture	15
T1. DOOG Filters: mean abs response	12
T2. DOOG Filters: mean of variables in T1	1
T3. DOOG Filters: id of max of variables in T1	1
T4. DOOG Filters: (max-median) of variables in T1	1

superpixel position, orientation, ratio of MajorAxis/MinorAxis, shape, eccentricity, mean RGB and HSV values, mean response of the Compass Filters and mean absolute response of DOOG filters. Finally, we use the generated superpixels as training examples and apply the SVM classifier.

The output of SVM is an uncalibrated value and not a probability distribution of a label given the feature vector. We use a method proposed by Wu *et al.* [50, 64], which is based on the method proposed by Platt [48] to convert the output of SVM

into the distribution $Pr(S = l|F_S)$, where l is a label, F_S is a feature vector computed on superpixel S, and $S = l$ stands for the event that all pixels inside superpixel S have the same label l. Thus, for a given label l, we learn the probability that all pixels inside a superpixel S have label l (under the assumption that all pixels within a superpixel *should* have the same label).

As we already mentioned, we would like to learn the probability that a single pixel p has label l. However, we can not learn these probabilities directly since there is not enough image information at an individual pixel p. Our solution is to simply to apply the distributions learned on the super pixels to the pixel based data term $D_p(f_p)$. That is $D_p(f_p) = -\log Pr(S^p = f_p|F_{S^p})$, where $\log Pr(S^p = f_p|F_{S^p})$ is the learned log probability of label f_p given the superpixel feature vector F_{S^p} s.t. $p \in S$. This approach makes sense since our energy in Eq. (4.1) does not require the true pixel based negative log probabilities. It is sufficient to come up with a reasonable penalty scheme for the $D_p(f_p)$ term, that is the scheme that for a given pixel p imposes higher penalties for the less likely labels. It is a reasonable assumption that if $Pr(S = l_1) < Pr(S = l_2)$, then for most pixels $p \in S$, $Pr(p = l_1) < Pr(p = l_2)$.

An alternative approach would be to formulate the labeling problem on the superpixel level, as has been tried before [5]. However, in case a superpixel does not contain pixels with the same true label, it becomes impossible to assign the true labels to all the pixels in that superpixel. By performing optimization on the pixel level, we are able to break apart any superpixel to better align the boundaries between the geometric labels with the intensity edges in the image, since a change in a geometric label tends to coincide with an intensity edge in the image.

4.2.2 Method summary

In summary, our geometric class labeling method includes four steps. First, we partition the input image into a set of superpixels. Then, we calculate the feature vector for each superpixel. After that, we use the trained classifier to learn the superpixel probabilities. Finally, the learned probabilities are fed into a global optimization framework, using the five part model discussed in Sec. 4.1 and optimized with our order-preserving moves. The four step procedure is illustrated in Fig. 4.4.

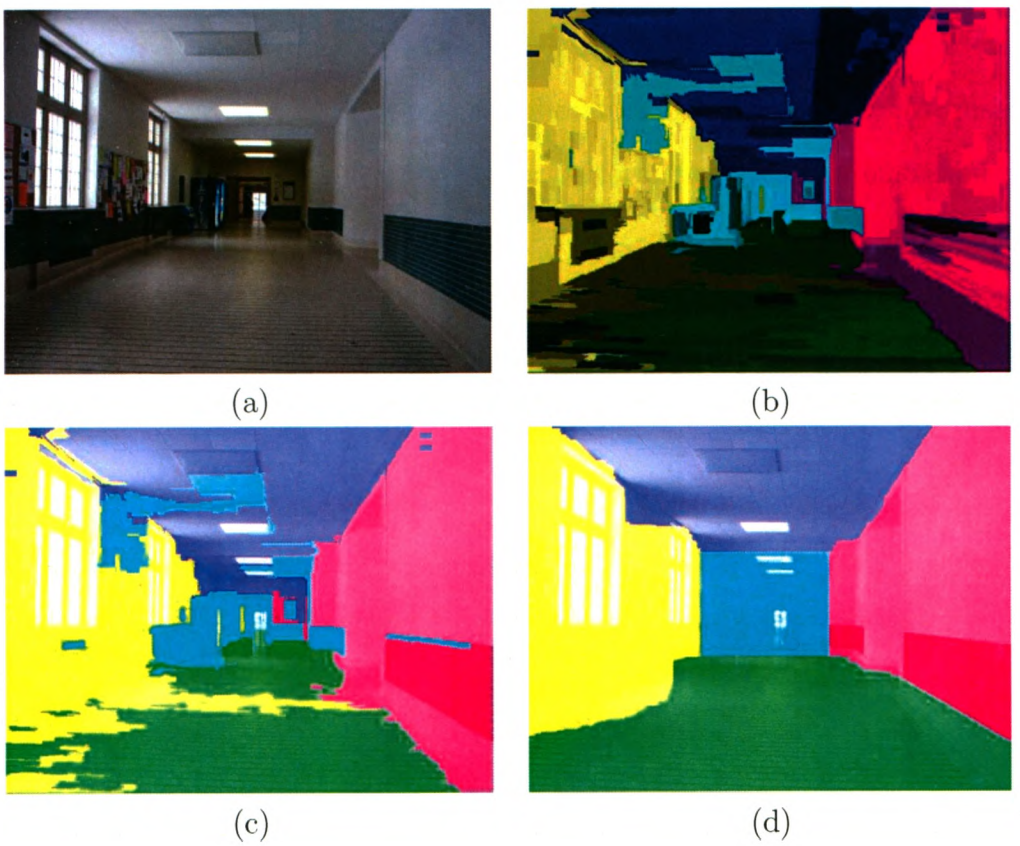


Figure 4.4: Illustration of our approach: (a) original image, (b) most probable label for each superpixel: the lighter colors correspond to higher probabilities, (c) SVM labeling, (d) our graph cut with order-preserving moves labeling, where geometric class labeling colors are overlayed on the original image in (c) and (d).

Chapter 5

Experimental Results

5.1 Indoor Images

We have collected and manually labeled 600 images from different indoor environments, such as corridors, lecture rooms, laboratories, cafeteria, etc. We used 300 images for SVM training, and 300 images for performance testing. Some examples of our images are shown in Fig. 5.1.

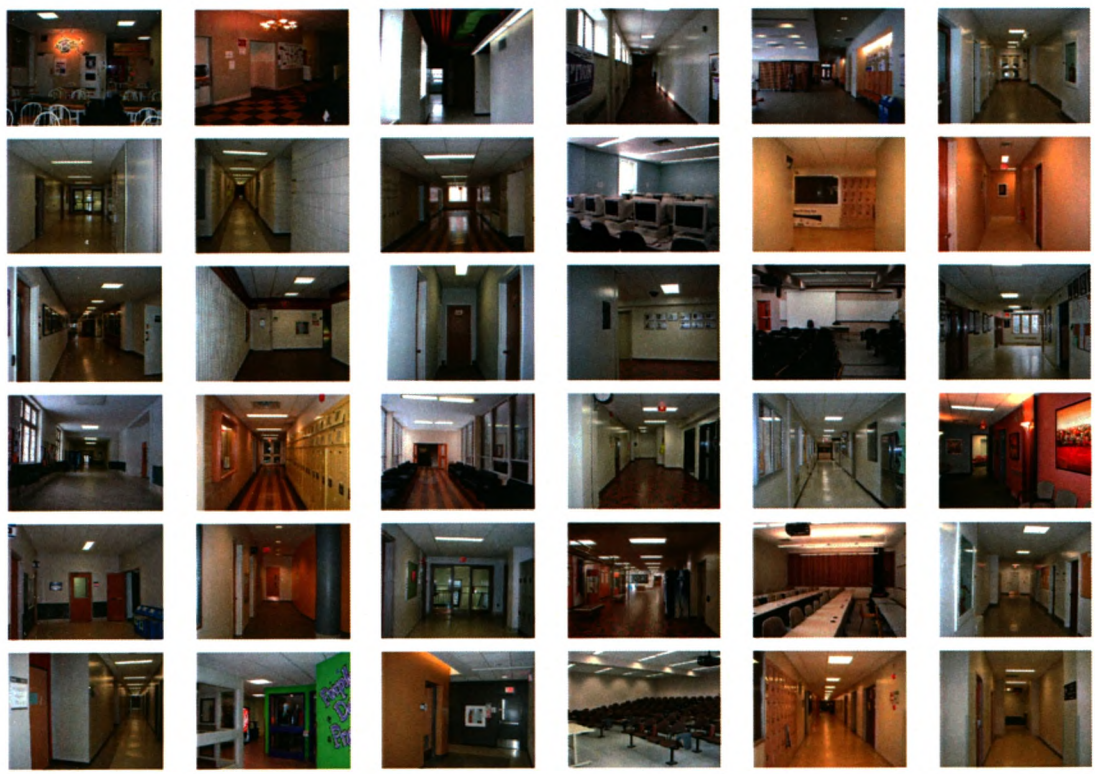
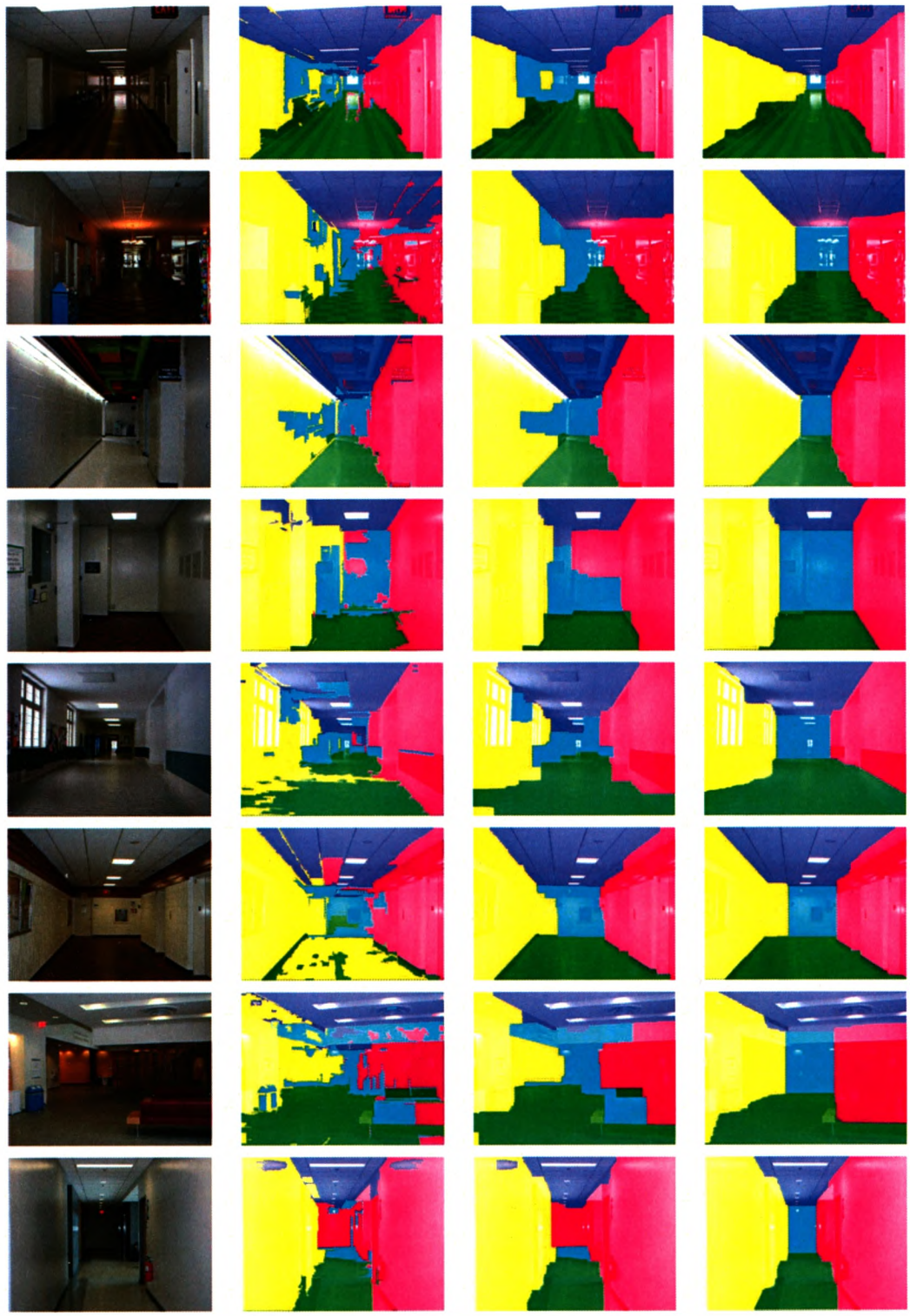


Figure 5.1: Sample indoor images

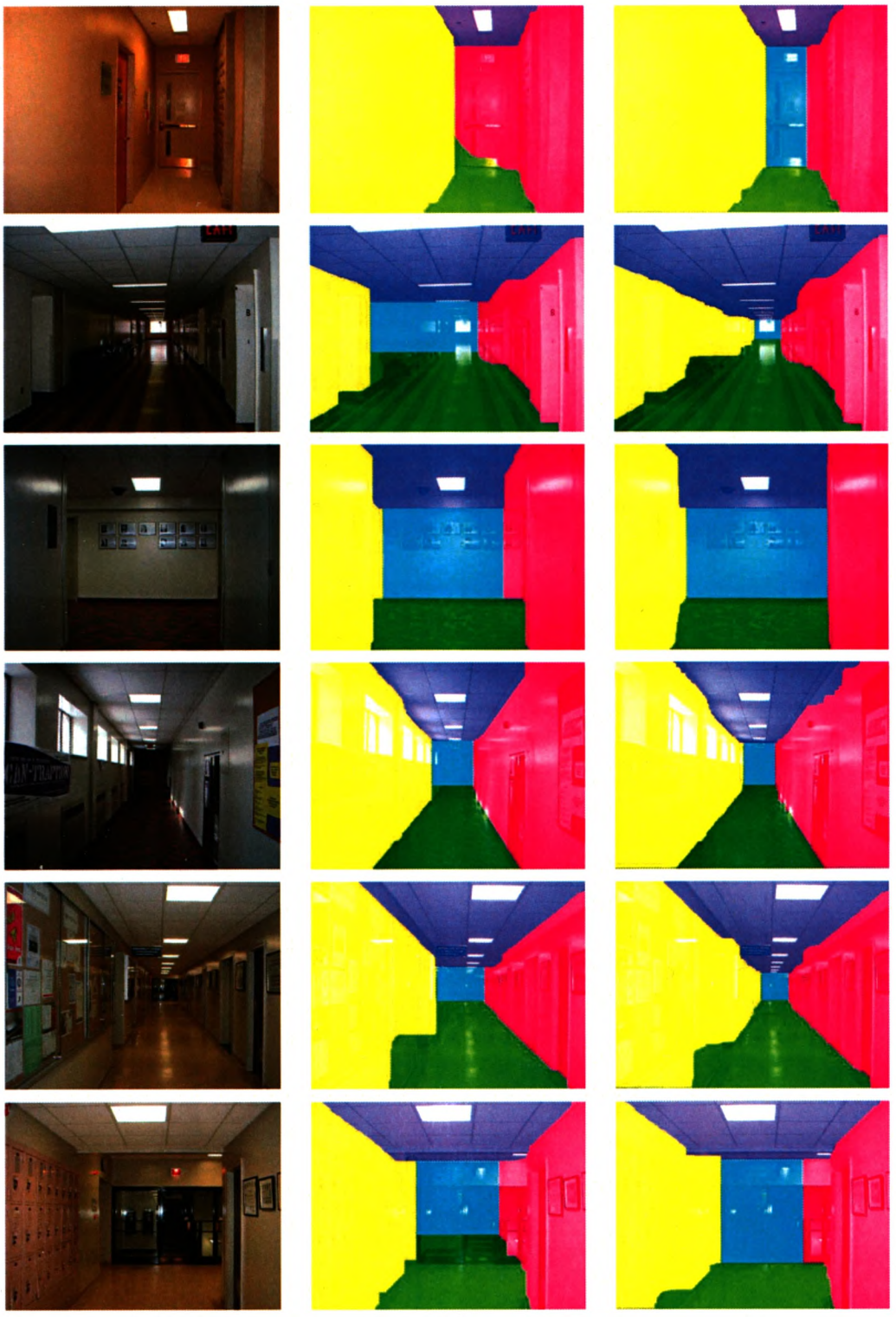
In this experiment, the five geometrical class labels: *left*, *right*, *bottom*, *top*, and *center* exactly correspond to the "left wall", "right wall", "ceiling", "floor", and "back wall" in the indoor scene, respectively.

In Fig. 5.2, we compare the results of SVM classification (5.2b), α-expansion without ordering constraints (5.2c) and our order-preserving moves (5.2d). As expected, the SVM labelings are not nearly as spatially consistent as those obtained with graph cut optimization. In the 6^{th} row in Fig. 5.2 (b), SVM fails to label most of the floor correctly. The spatial smoothness constraints helps label most of the floor correctly for the same image in Fig. 5.2 (c) and (d). Comparing graph cuts with and without ordering constraints, in columns (5.2c) and (5.2d), respectively, implausible regions are frequent in Fig. 5.2 (c). For example, back wall patches appear in the middle of the left wall (in rows $1-4$); right wall patches in the middle of the back wall (in rows $4-8$); ceiling patches appear in the middle of left wall (in row 8) and back wall(in row 5). In the bottommost row of Fig. 5.2 (c), a large area of back wall is smoothed out in Fig. 5.2 (c). In almost all images shown in Fig. 5.2 (c), the back wall is present with significantly distorted shape compared to the results in Fig. 5.2 (d). This clearly shows that incorporating ordering constraints tends to guide the graph cut optimization away from implausible solutions. In Fig. 5.3 we show some results of α-expansion in (5.3b) and order-preserving moves in (5.3c) when ordering constraints are used in the energy function. As expected, α-expansion gets stuck in a local minimum easier. Figs. 5.4 shows more results, illustrating the accuracy the proposed method can achieve without user interaction. It is worth noticing that it is not necessary for an image to contain all the five geometric parts in order to be used in our method. For example, in the first image of the 2^{nd} row, the image does not have a right wall at all. However, our proposed method can produce correct labeling. Therefore, our model can be used in an even more general way.



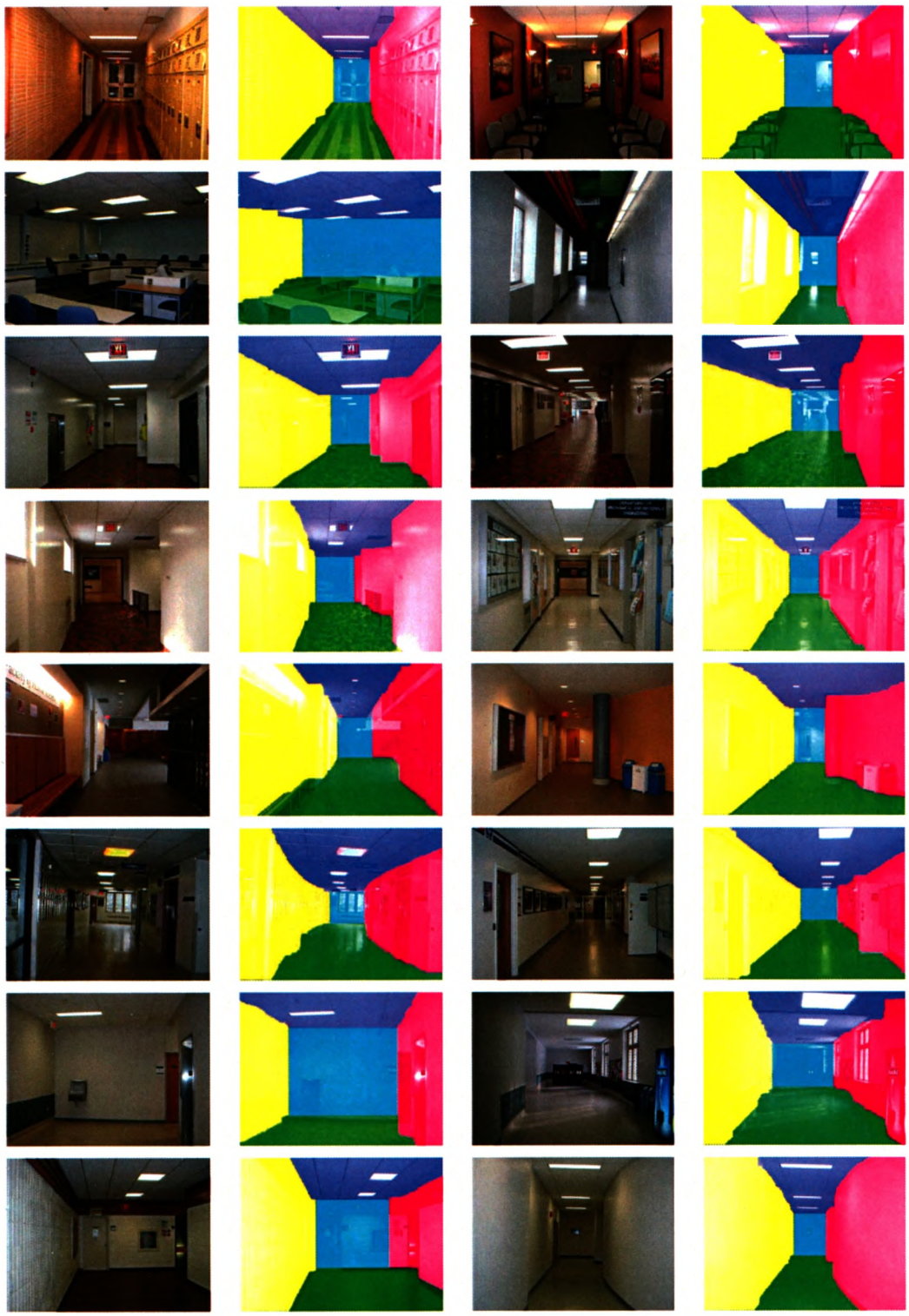
(a) (b) (c) (d)

Figure 5.2: Indoor image result comparison: (a) original images (b) SVM labeling, (c) α-expansion without ordering constraints (d) order-preserving moves.



(a) (b) (c)

Figure 5.3: Indoor image result comparison: (a) original images, (b) α-expansion with ordering constraints, (c) order-preserving moves.



(a) (b) (a) (b)

Figure 5.4: Indoor image results: (a) original images, (b) order-preserving moves.

5.2 Outdoor Images

We downloaded and collected 84 outdoor street images from the Google image and Yahoo image search, and from the PASCAL Object Recognition Database [65] as well. All images were manually labeled. We used half of the images for training and half for testing for testing the performance. Some example images are shown in Fig. 5.5.

Figure 5.5: Sample outdoor images

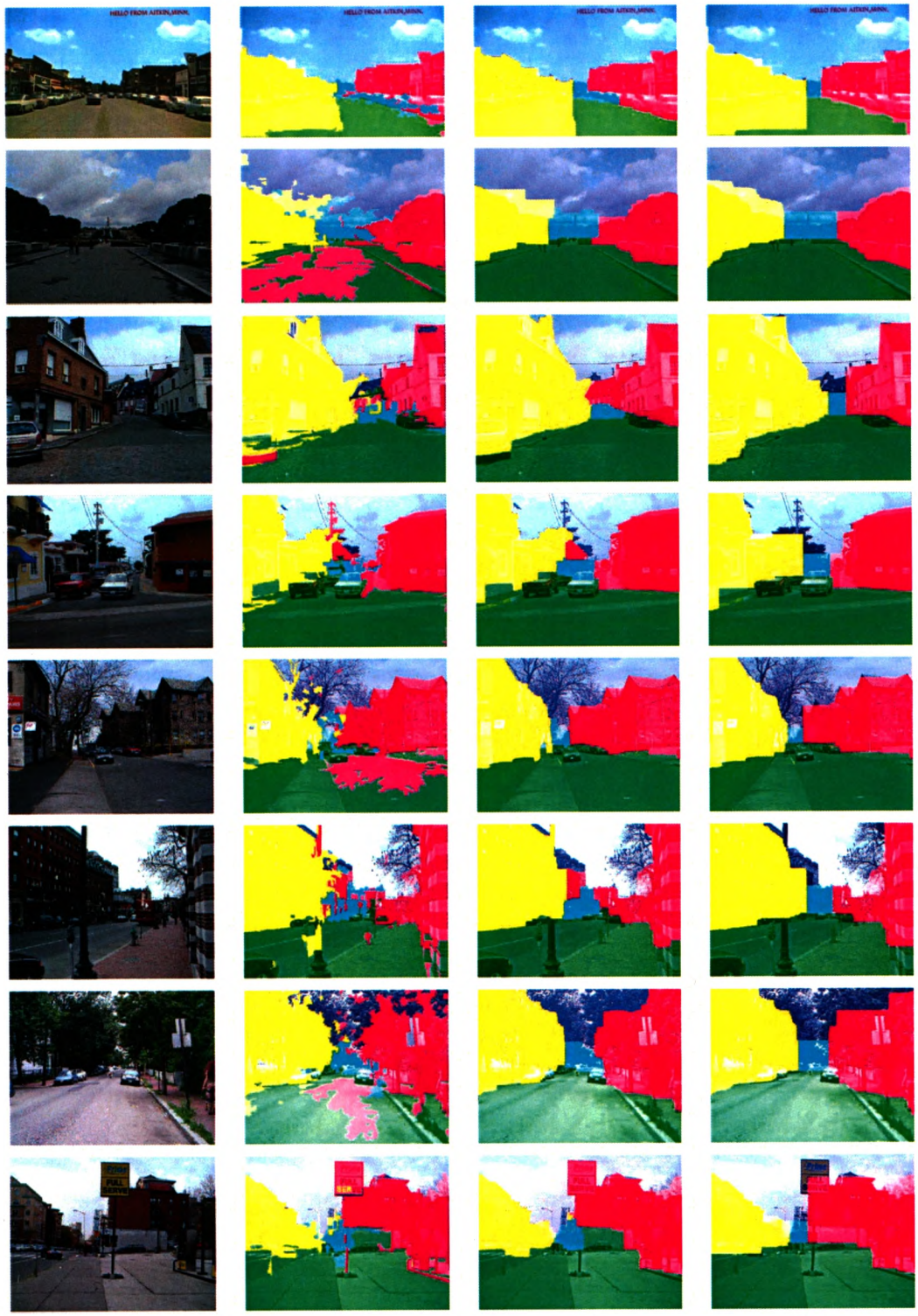
In this experiment, the five geometrical class labels: *bottom*, *top*, *left*, *right*, *center* correspond to the "sky", "ground", "left-side", "right-side", and "back-side" of scene, respectively. Therefore, our 5-part model can also perfectly be used in this application.

In Fig. 5.6, we compare the results of SVM classification (5.6b), α-expansion without ordering constraints (5.6c), and our order-preserving moves (5.6d). As expected, the SVM labelings are not nearly as spatially consistent as those obtained with graph cut optimization. In the 2^{nd}, 5^{th} and 7^{th} rows in Fig. 5.6 (b), SVM fails to label large areas of ground correctly. The spatial smoothness constraints helps label most of the ground correctly for the same images in Fig. 5.6 (c) and Fig. 5.6 (d).

Comparing graph cuts with and without ordering constraints(in columns (c) and (d) in Fig. 5.6), implausible regions are frequent in Fig. 5.6 (c): left-side patches and right-side patches appear connected directly without back-side patches in between (in row 3 and row 4) ; right-side scene patch appears in the middle of the left-side scene (in the 6^{th} row). In the bottommost row of Fig. 5.6 (c), a large area of back-side scene is labeled as ground. For the other images in Fig. 5.6 (c), the back-side scene is present but its shape is distorted compared to the results in Fig. 5.6 (d). It is clear that incorporating ordering constraints helps by guiding graph cut optimization away from implausible solutions.

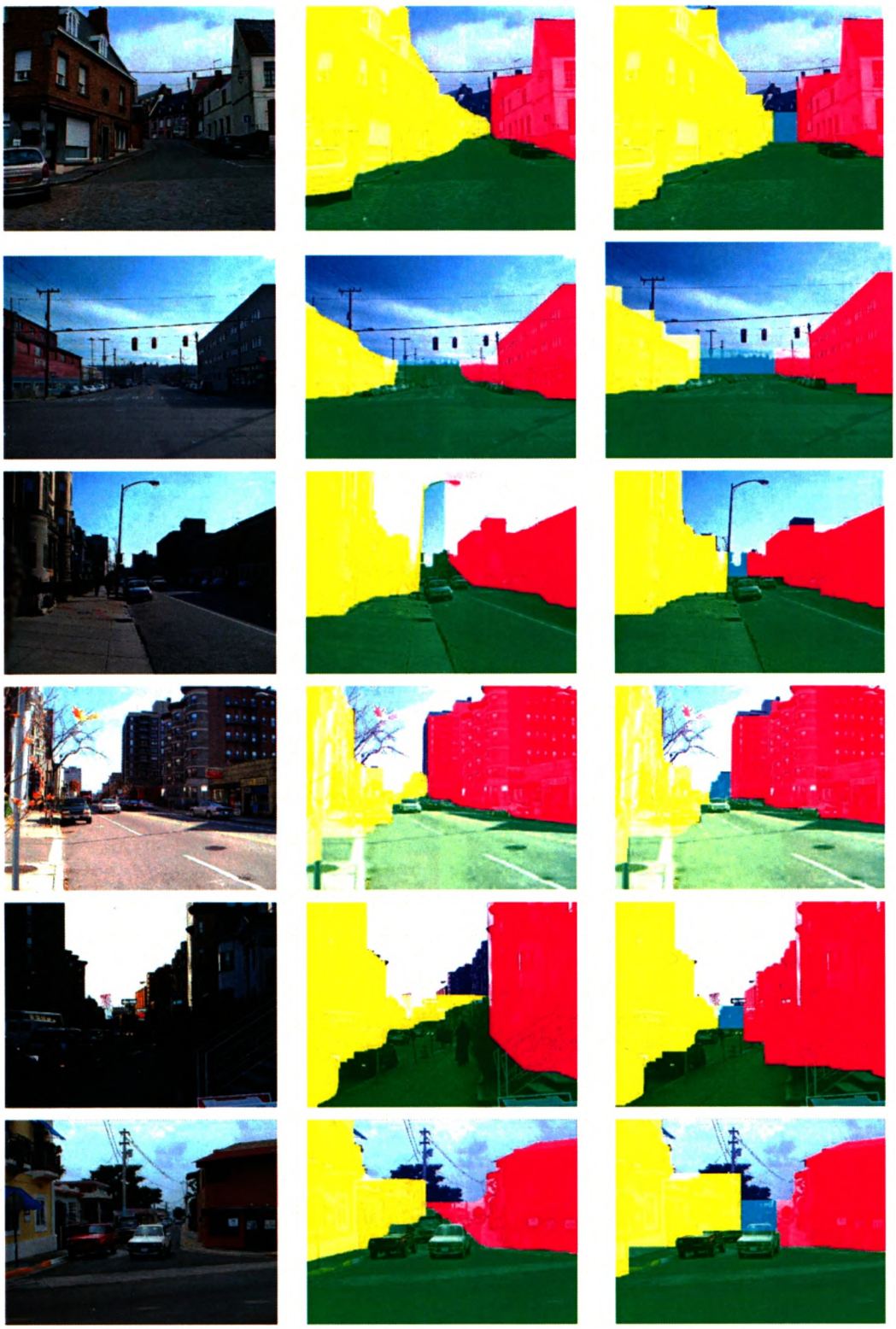
In Fig. 5.7 we show some results of α-expansion (5.7b) and order-preserving moves (5.7c) when ordering constraints are used in the energy function. From Fig. 5.7 we can see that α-expansion gets stuck in a local minimum even easier in the outdoor scene than the indoor scene as almost all the back-side scene in Fig. 5.7 (b) is presented as a thin line which is only a few pixels thick. In the 3^{rd} row in Fig. 5.6 (b), a large back-side patch is in the middle of the sky. Clearly, incorporating ordering constraints help guide the graph cut optimization away from implausible solutions.

Figs. 5.8 also shows more results, illustrating the accuracy the proposed method can achieve with no user interaction.



(a) (b) (c) (d)

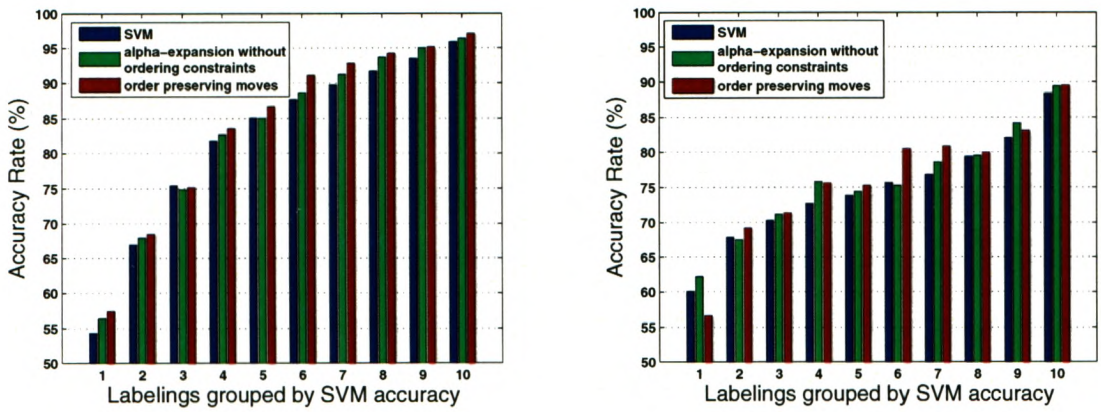
Figure 5.6: Outdoor image result comparison: (a) original images (b) SVM labeling, (c) α-expansion without ordering constraints (d) order-preserving moves.



(a) (b) (c)

Figure 5.7: Outdoor image result comparison: (a) original images, (b) α-expansion with ordering constraints, (c) order-preserving moves.

Figure 5.8: Outdoor image results: (a) original images, (b) order-preserving moves.



(a) indoor images (b) outdoor images

Figure 5.9: Accuracy comparison: in bins by quality vs. accuracy rate, where SVM results are grouped in 10 equal bins, ordered from least accurate to most accurate.

5.3 Discussions

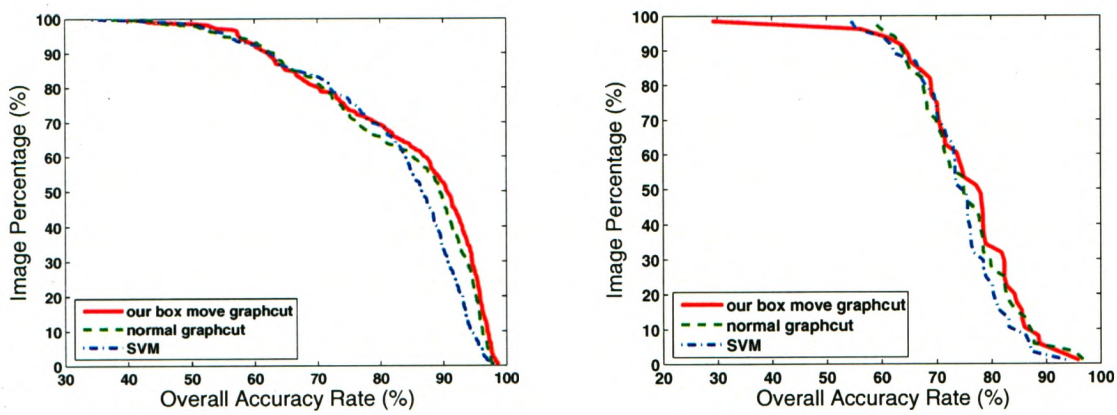
Since the data term in our energy function is calculated based on the probabilities generated by SVM, when SVM gives reasonable label probabilities, our algorithm can significantly improve SVM results. However, when SVM results are far from reasonable, our order-preserving moves can worsen SVM results as the algorithm is trying to satisfy the ordering constraints that can not be reasonably satisfied (see in Fig. 5.11). Therefore, the overall accuracy improvement over SVM computed for all the images is not large. However, when SVM results are not reasonable, they are hardly useful for applications anyway. The overall accuracy rate is computed by Eq 5.1.

$$Accuracy = \frac{N_p}{N_t} * 100\% \tag{5.1}$$

where N_p is the number of pixels with correct labels and N_t is the number of total pixel in an image.

We put SVM results in 10 equal bins, ordered from least accurate to most accurate. The higher the bin number, the more accurate are the SVM labelings in

that bin. Fig. 5.9 shows the accuracy of the algorithms for each bin. For the worst bin (unreliable SVM results), accuracy of order-preserving moves are worse than the accuracy of SVM only labeling for outdoor images. For the best bins (very accurate SVM results), order-preserving moves do not improve SVM results significantly since there is not much room for improvement. However, in the middle range (from about 4th bin to the 8th bin), there is noticeable improvement over SVM and α-expansion, especially for the outdoor images. For example, in the 6th bin, order-preserving moves have about 80% accuracy, followed by approximately 75% accuracy for α-expansion and SVM.



(a) indoor images (b) outdoor images

Figure 5.10: Accuracy rate vs. % of images

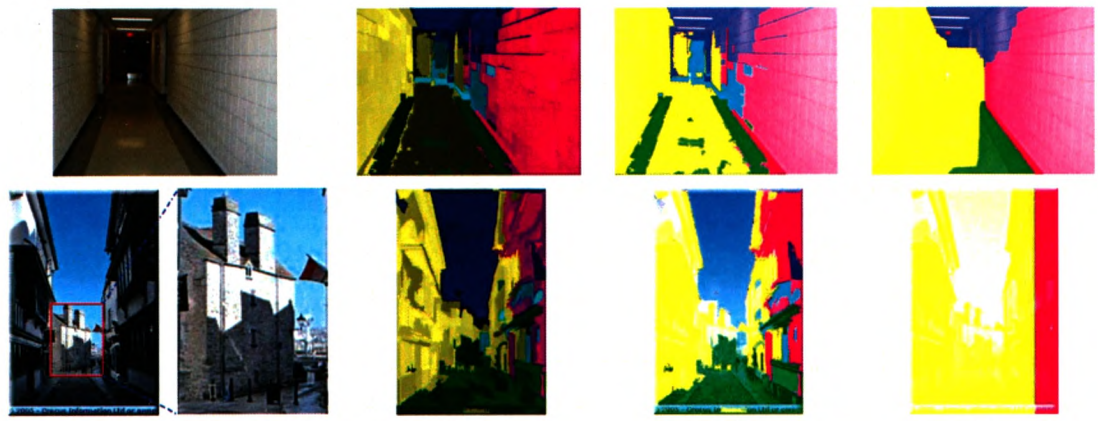
Fig. 5.10 shows the percentage of labelings that have the at least the accuracy rate specified on the horizontal axis. For example, for indoor images, 52% of order-preserving labelings have the accuracy rate of at least 90%, whereas only 33% and 46% of SVM and α-expansion labelings, respectively, have this accuracy rate. Order-preserving moves always have a higher percentage of images at any given accuracy rate in the range between 75% and 100%.

We also directly compare the energy values produced by the two algorithms,

Table 5.1: Energy Reduction Improvement of Order-preserving move over $\alpha-$expansion

	Total energy (%)	Data energy (%)	Smoothness energy (%)
Indoor	27.3 ± 9.8	2.2 ± 3.6	96.0 ± 1.1
Outdoor	29.2 ± 18.5	8.7 ± 24.6	91.5 ± 3.0

the proposed order-preserving moves always give a smaller energy compared to α-expansion as shown in Table 5.1. On the 300 indoor images, on average, the energy is 27.3% smaller ($\sigma = 9.8\%$). On the 42 outdoor images, on average, the energy is 29.2 % smaller ($\sigma = 18.5\%$).



(a) (b) (c) (d)

Figure 5.11: Failure cases (a) original images (b) SVM generated label probabilities (c) SVM labeling, (d) Order-preserving moves.

Some failures are shown in Fig. 5.11. The ordering constraints are not violated in Fig. 5.11(d), but the "center" region between the "left" and the "right" regions it is too thin to be seen at this resolution. Most failures occur when the "center" data terms are far from reasonable, as in Fig. 5.11 (b).

We computed percentage of "successful" labelings, where a labeling is successful if it has at least 90% overall accuracy or at least 80% overall accuracy and the "center" region is at least 60% accurate. We found experimentally that labelings

Table 5.2: Performance Summary

	Percentage of Successful Labelings (%)			
	SVM	α-exp. no OC	α-exp. with OC	Order-pres. moves
Indoor	29.3	61.0	72.3	74.3
Outdoor	16.7	40.5	38.1	61.9
	Overall Accuracy Rate (%)			
	SVM	α-exp. no OC	α-exp. with OC	Order-pres. moves
Indoor	83.0	84.1	84.7	85.0
Outdoor	74.0	75.2	71.0	75.3
	Processing Time (seconds)			
	SVM	α-exp. no OC	α-exp. with OC	Order-pres. moves
Indoor	34.5 ± 2.6	45.5 ± 3.6	85.5 ± 219.0	62.3 ± 25.6
Outdoor	29.3 ± 8.4	43.8 ± 12.4	286.4 ± 480.1	56.5 ± 18.3

satisfying these conditions can be used successfully for virtual scene walk-through. Table 5.2 summarizes the performance of SVM, α-expansion without and with ordering constraints, and the order-preserving moves (in that order). Order-preserving moves algorithm is a clear winner when it comes to the percentage of "successful" labelings and also shows a modest improvement for the overall accuracy rate. From Table 5.2, we can also see that the Order-preserving move method is a lot better than the α-expansion method in the manner of computation time although these two have very similar overall accuracy performance. The average processing time calculated on a personal computer with 2.4GHz CPU and 2048MB memory is also listed in Table 5.2. The computational time includes superpixel segmentation, feature extraction, data terms calculation, and the corresponding energy minimization.

5.4 Other Images

Our geometrical scene class labeling with ordering constraints model can also be used in many applications, such as human figure/gesture detection, computer-screen detection, etc.

In this section, we will show how our model to be used in the applications of the nature scene and object part segmentation, i.e. Mickey mouse segmentation. The labeling of natural scenes involves seascape segmentation whereas the Mickey Mouse segmentation involves labeling the body parts of the Mickey Mouse.

5.4.1 Nature scene

In the application of the nature scene (seascapes), three geometrical class labels: *bottom*, *top*, *center* are used which correspond to the "sea", "sky", and "mountain" in the scene, respectively. In this case, only the vertical ordering constrains are used. We also use one none-geometrical class label called "object", which corresponds to large scene objects. There are no ordering constraints for the "object" label.

We have collected and manually labeled 92 of nature scene images, which are obtained from the website "Thailand Bilder"(Photo and Image Gallery from Thailand) [66]. Out of this 92 images, we used 61 images for training SVM, and 31 images for testing the performance. Fig. 5.12 shows some of the examples.

Fig. 5.13 shows some of the experimental results, where we compare the results of SVM classification (5.13b), α-expansion without ordering constraints (5.13c) and with ordering constraints (5.13d).

In general, all the 3 algorithms work well for most of the images. However, there are still some small spare mislabeled regions for the SVM labeling in Fig. 5.13 (b), which can be smoothed out by smoothness constrains as shown in Fig. 5.13 (c) and Fig. 5.13 (d).

Figure 5.12: Sample nature scene images

Comparing the α-expansion without ordering constraints Fig. 5.13 (c) and with ordering constraints Fig. 5.13 (d), it appears that the improvement with ordering constraints is not huge. However, optimization with ordering constraints does recover some fine details in scene. For examples, in the first row, Fig. 5.13 (c) mislabels part of the small mountain in the right side as sea; in the third row, Fig. 5.13 (c) mislabels part of the small mountain in the right side as sky; in the second last row, Fig. 5.13 (c) mislabels a sea region in the left as mountain.

As we mentioned before, when SVM works almost perfect, there is not too much room for improvement. In our nature scene application, SVM can get almost perfect classification. The reason lies on the features used here are highly discriminant in the nature scene. For example, color information is a very good feature as "sky" is always blue. However, in general, such case does not happen quite often in most of the real applications. In the case when SVM can not generate such perfect results, the order-constraints do have a better performance.

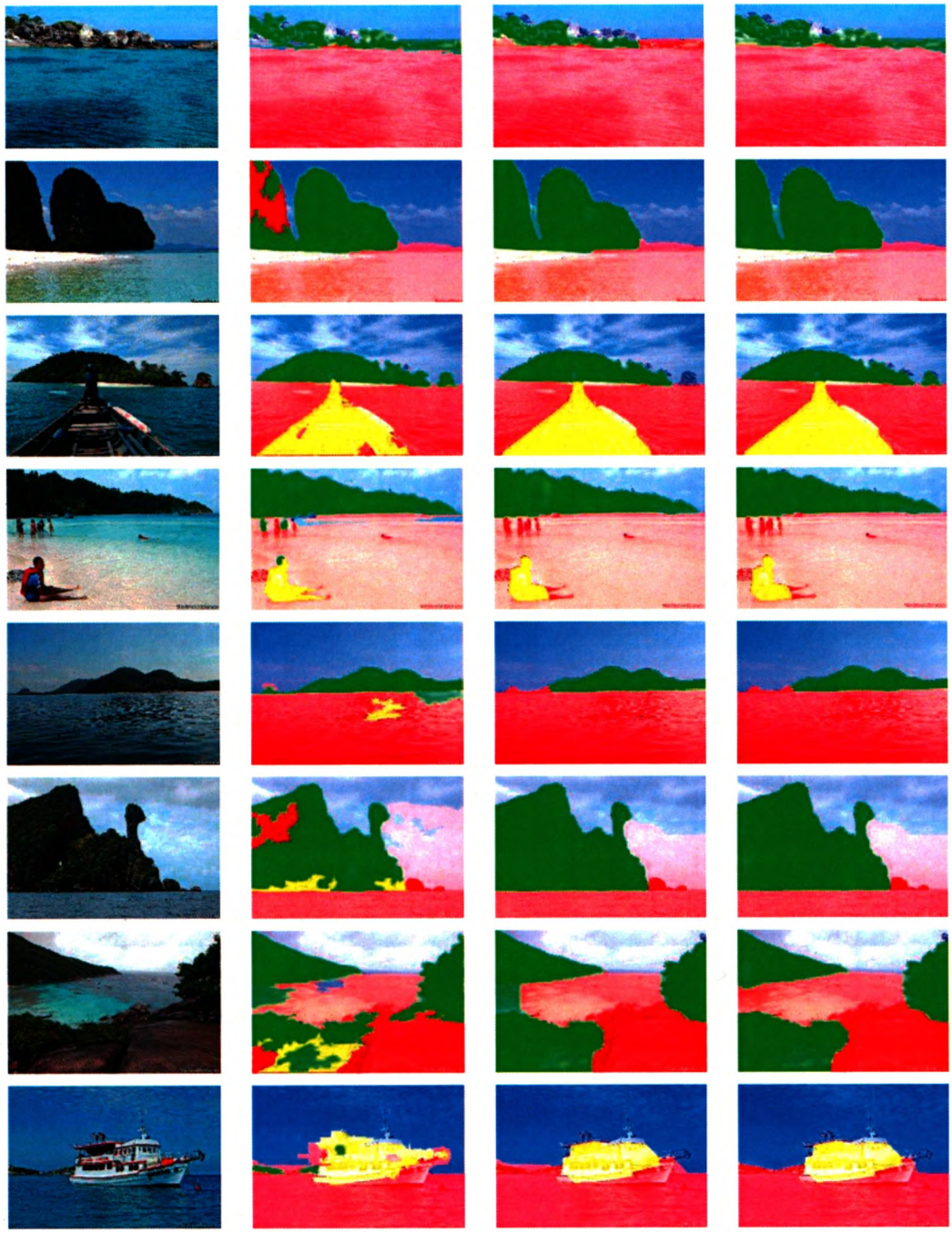


Figure 5.13: Nature scene results: (a) original images, (b) SVM labeling, (c) α-expansion, no ordering constraints (d) with ordering constraints. Color scheme: blue="sky", green = "mountain", red="sea", yellow="object".

5.4.2 Segmenting images with Mickey mouse(tm)

For the Mickey mouse part segmentation, the five geometrical class labels: *bottom*, *top*, *center*, *left*, *right* correspond to the "leg", "head", "body", "left-hand", "right-hand". Therefor, the five-part model is quite appropriate for this application. We also use one non-geometric class label called "background" which corresponds to image background. There are no ordering constraints for the "background" label, either.

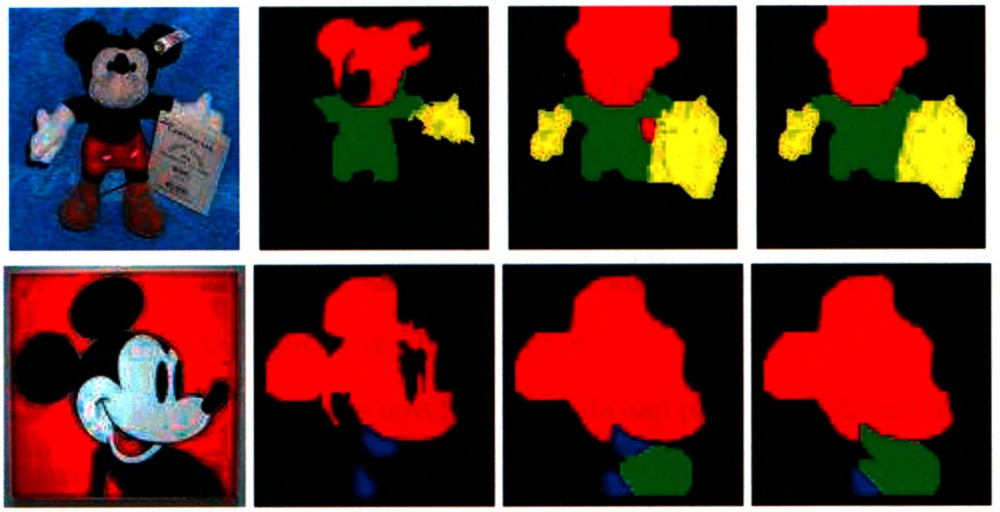
Mickey mouse images are downloaded from Google and Yahoo image search website. We have collected and manually labeled 70 of Mickey mouse images. We use cross validation method for training and testing due to the small data set. Some examples of the images are shown in Fig. 5.14.

Figure 5.14: Sample Mickey mouse images

Figure 5.15: Mickey mouse results: (a) original images, (b) SVM labeling, (c) α-expansion with ordering constraints. Color scheme: blue="leg", green = "body", red="head", yellow="hand".

Fig. 5.15 shows some of the experimental results, which shows that α-expansion with ordering constraints (5.15c) performs significantly better than the SVM results (5.15b).

Fig. 5.16 shows the performance comparison between the α-expansion without and with ordering constraints. It is clear that using ordering constraints algorithm improves the segmentation as shown in Fig. 5.16 (c) and Fig. 5.16 (d).



(a) (b) (c) (d)

Figure 5.16: Mickey mouse result comparison: (a) original images, (b) SVM labeling, (c) α-expansion without ordering constraints, (d) α-expansion with ordering constraints. Color scheme: blue="leg", green = "body", red="head", yellow="hand".

5.5 Applications

In this section, we illustrate the use of scene structure obtained from the proposed segmentation algorithm in applications such as virtual scene walk-through, semantic segmentation, scene synthesis and scene text extraction. We also show how our order-preserving moves can be used for certain simple shape priors in graph-cut segmentation.

5.5.1 Automatic novel view & virtual scene walk-through

In the applications of novel view and virtual scene walk-through, we assume that the scene is a 3D box. We use the idea of spidery mesh to fit perspective projection and mimic 3D camera transformations [3] to navigate a scene. Spidery mesh is composed of four parts (vanishing point, radial lines, inner and outer rectangles), which partitions the 2D image into five regions (left, right, rear, floor, and ceiling). Since we have

already labeled the image into exactly these five regions, generating the spidery mesh is trivial. We fit the radial lines with the RANSAC algorithm based on the boundary points between regions labeled as different geometric classes [67, 68]. Vanishing point is calculated as the weighted average of the intersection of the radial lines, the inner rectangle is the region labeled as the back wall and the rest are outer rectangles. Fig. 5.17 illustrates the spidery mesh generation.

Fig. 5.18 shows some of the novel view results and part of the virtual scene walk-through results for indoor images. In Fig. 5.19, we show that using SVM labeling alone fails to produce satisfactory novel view generation/scene walk-through results as the room appears to have a strangely curved walls and floor. Those results are generated by using the above spidery mesh on a reasonably good SVM labeling. Fig. 5.20 and Fig. 5.21 show part of the virtual scene walk-through results and some of the novel view results for outdoor images.

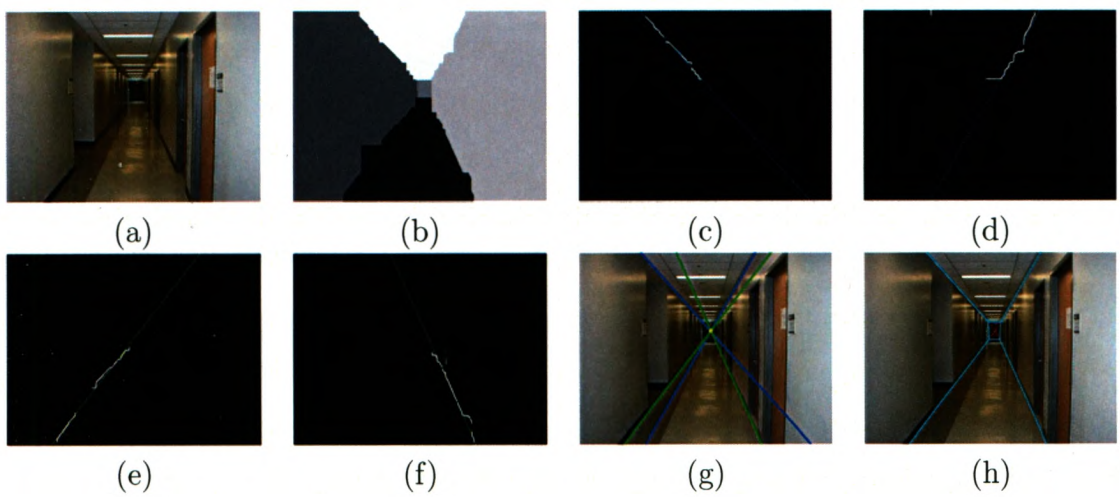


Figure 5.17: Spidery mesh generation: (a) original image, (b) labeled image, (c)~(f) radial lines fitted between the boundaries of left wall and ceiling; right wall and ceiling, left wall and floor, and right wall and floor, (g) vanishing point and fitted radial lines, (h) spidery mesh overlayed on the original image.

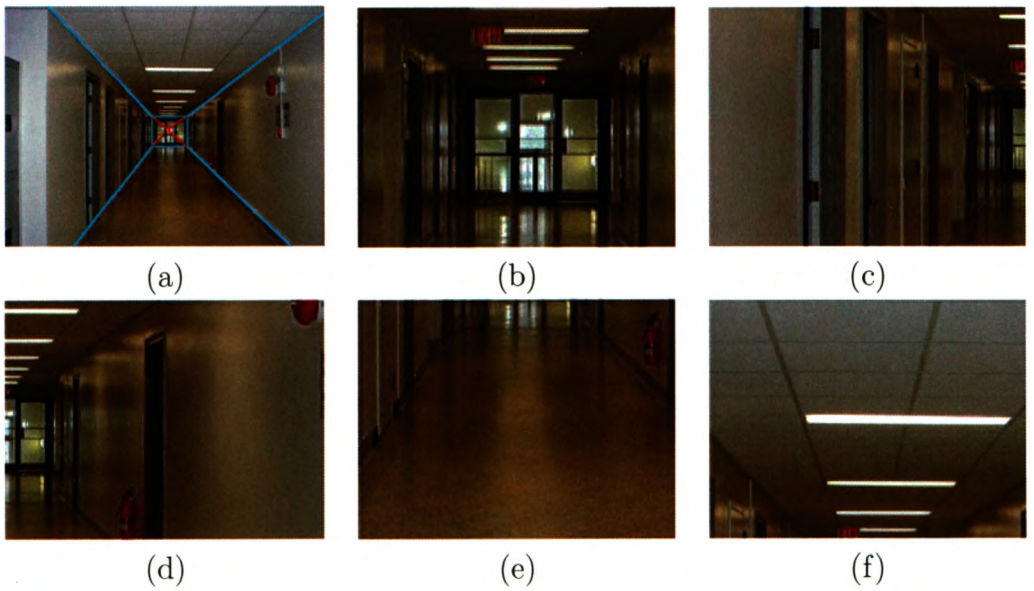


Figure 5.18: Indoor virtual scene walk-through results generated by using our method, (a) spidery mesh overlayed on the original image, (b) walk forward, (c) look left, (d) look right, (e) look down, (f) look up.

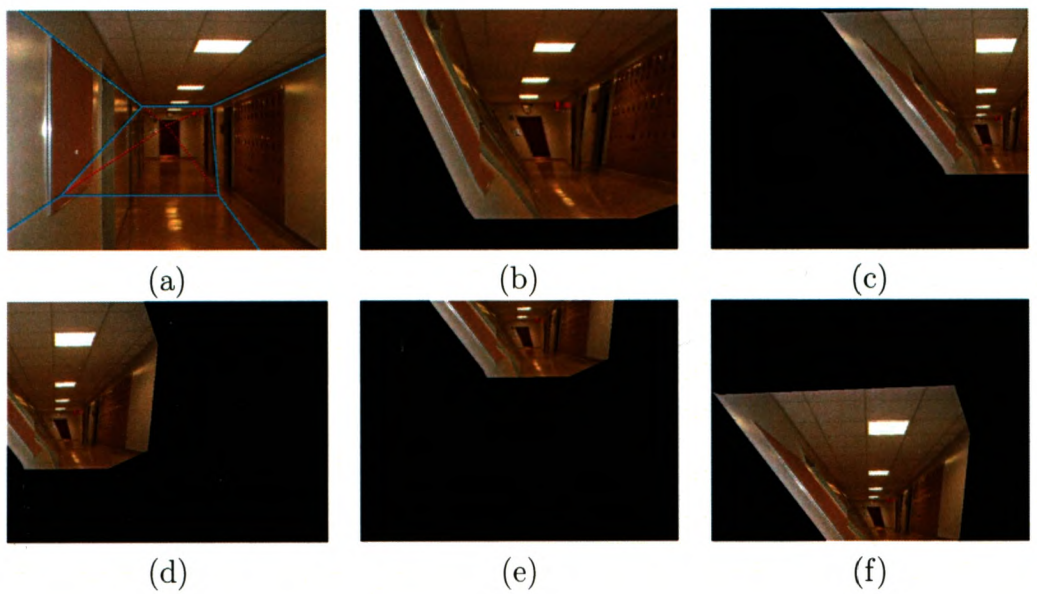


Figure 5.19: Virtual scene walk-through results by using SVM labels directly, (a) spidery mesh overlayed on the original image, (b) walk forward, (c) look left, (d) look right, (e) look down, (f) look up.

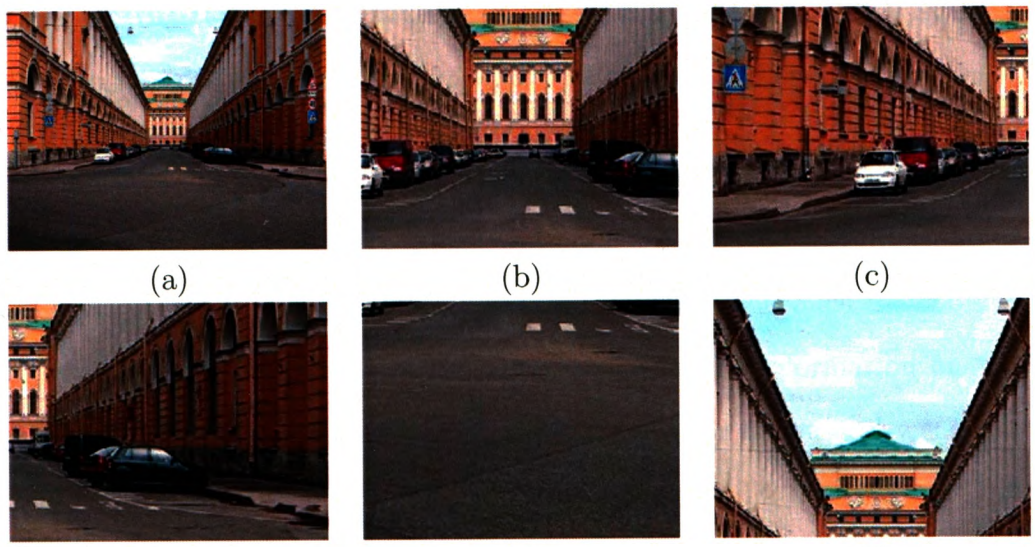


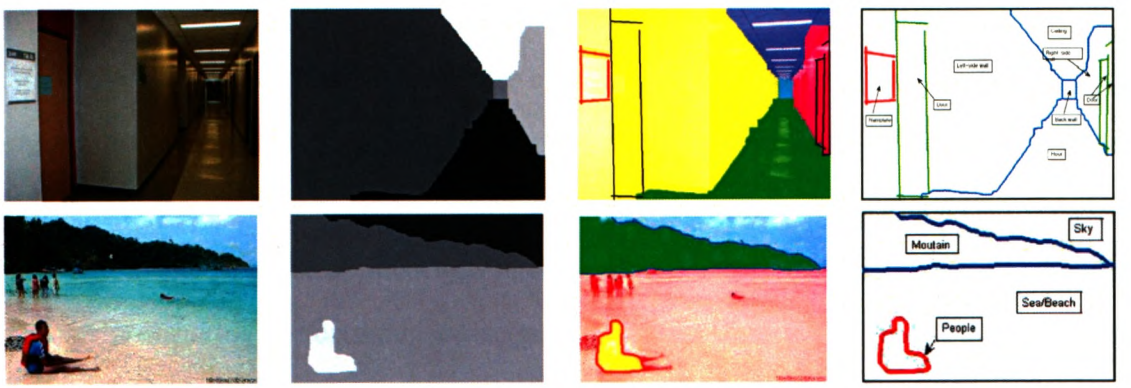
Figure 5.20: Outdoor virtual scene walk-through results generated by using our method, (a) original image, (b) walk forward, (c) look left, (d) look right, (e) look down, (f) look up.

Figure 5.21: Outdoor novel view results generated by using our method, (a) original images, (b1)-(b3) generated views.

5.5.2 Semantic segmentation, scene synthesis and scene text extraction

The obtained scene structure can also be combined with high level knowledge in many applications. For example, doors, windows, nameplates, and information signs are usually on the wall; people, cars, chairs and boxes usually touch the floor/ground. We demonstrate the usefulness of the obtained geometric information on two very popular applications in computer vision: semantic segmentation and scene synthesis [69, 70].

Fig. 5.22 shows the results of semantic segmentation. In the first row of Fig. 5.22, we use the use weighted least squares line fitting method [71] to get line boundaries of the doors and the nameplates.

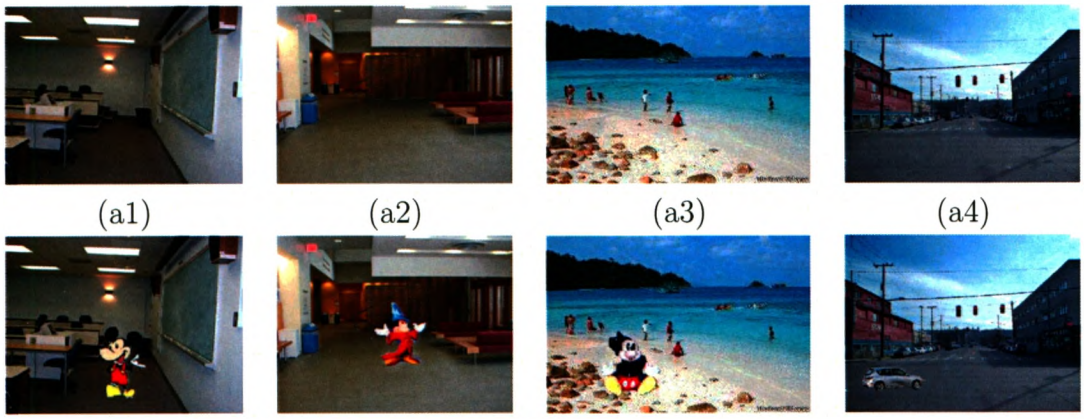


(a) (b) (c) (d)

Figure 5.22: Semantic segmentation, (a) original image, (b) labeled image, (c) semantic segmented image, (d) generated semantic information.

Fig. 5.23 shows some of the scene synthesis results, where segmented objects are naturally inserted into a scene with good reasoning. Since Mickey mouse can only stand on the ground or sit on the beach, by touching the segmented Mickey mouse' foot-part with regions labeled as "floor" or the leg-part with the region labeled as "beach/sea", Mickey mouse can be inserted into indoor (5.23b1 and 5.23b2) and

seascape (5.23b3) environment naturally. By touching the segmented wheels with the region labeled as "ground", segmented cars are naturally inserted on the street ground (5.23b4).



(a1) (a2) (a3) (a4)

(b1) (b2) (b3) (b4)

Figure 5.23: Scene synthesis, (a1-a4) original image, (b1-b4) synthetic image.

Since the obtained scene structure information with some higher level knowledge can help to avoid ambiguous information produced by only local bottom-up information, it also be used to achieve a superior performance on object detection [72, 73], which is another popular applications in computer vision [74, 75, 76]. Fig. 5.24 demonstrates the superior performance on scene text extraction, where, we apply the scene text detection methods proposed by us in previous publications [77, 78].

(a) (b) (c)

Figure 5.24: Scene text detection, (a) original image, (b) scene text detection without high level knowledge, (c) scene text detection with high level knowledge.

5.5.3 Shape prior for segmentation

Shape priors for segmentation in general [79, 80, 81] and segmentation with a graph-cut [82, 83] is a recent area of interest. General segmentation with a shape prior is usually based on local optimization and therefore is prone to getting stuck in a local minimum. The graph-cut methods used in this application have to register the shape model with the image during the segmentation process, which is a difficult task in itself [82, 83].

Instead of shape priors specific to a particular object, we investigate simple generic shapes such as "rectangle", "trapezoid", etc. We observe that by splitting an image into parts with ordering constraints between them, we can enforce the "center" region to be of a certain shape. Usually the object/background segmentation is formulated as a binary labeling: the labels are the object and the background. We use more than two labels to incorporate a shape prior: the object corresponds to the "center" label and the other labels correspond to the background. This is a novel approach for incorporating shape priors. It is the relative order of the parts that enforce a certain shape for the object. We use a rectangular and a trapezoidal shape, although other simple shapes can be implemented as well. In [84], they use a similar idea but only for rectangular shapes.

A recent related work segments rectangles using generalized eigenvectors [85]. In this method, they used the ratio of perimeters from different metrics as the shape measurement and the final segmented shape is obtained by thresholding the optimized eigenvectors. Unlike this method, our method does not need any measurement for shapes and can get direct shape representations from the obtained labels without any other post-processing such as thresholding.

We now explain how to incorporate simple geometric shape priors in graph-cut

segmentation of an object from its background. For a rectangle, we use the same V_{pq} as in Table4.1, except now any V_{pq} not involving label C is set to 0 since a discontinuity between, say labels L and B *does not* correspond to the border between the object and the background. We consider a trapezoid with parallel sides in horizontal orientation, and the shorter side on top (for other trapezoids, an image just needs to be rotated). To get a trapezoid, we relax the following constraints in Table4.1: for vertical neighbors, we set $V_{pq}(L, C) = V_{pq}(R, C) = w_{pq}$, instead of ∞. This change allows the borders between the L and C regions and C and R regions to be diagonals, slanted to the left and to the right, respectively. Strictly speaking, this shape prior is not a true trapezoid since we cannot enforce the borders between the L and C regions and C and R regions to be straight lines. We still use the term "trapezoid" for the lack of a better name.

We can use object-specific data terms based on brightness, user interaction, etc. However, to study the effect of the shape prior in isolation from regional influences, we opted to find regions with strong intensity edges on the boundary that agree with the shape prior. An object-specific D_p can always be added, of course. We do have to set D_p for any p on the image border. We set each border p to strongly prefer its own border, i.e. for p on the left border, $D_p(L) = 0$ and $D_p(C) = D_p(R) = D_p(T) = D_p(B) = \infty$, etc. Thus our cost function (ignoring the border data terms, which are constant for finite energy labelings) is the sum of the w_{pq} on the boundary between the object and the other regions. To avoid a trivial solution (the object of size 1 pixel), we make w_{pq}'s negative whenever there is a strong intensity edge between pixels p and q, biasing towards a larger boundary coinciding with intensity edges. In general, making $w_{pq} < 0$ is not always possible, but it is possible for our vertical/horizontal moves.

Figs. 5.25 and 5.26 show the results with rectangular and trapezoid prior and

illustrate the ability to pick out interesting regions that obey the corresponding shape priors without any knowledge of the object/background regional properties. In both figures, the original images are in the top row, and the results are in the second row. All results were obtained with the same parameter settings.

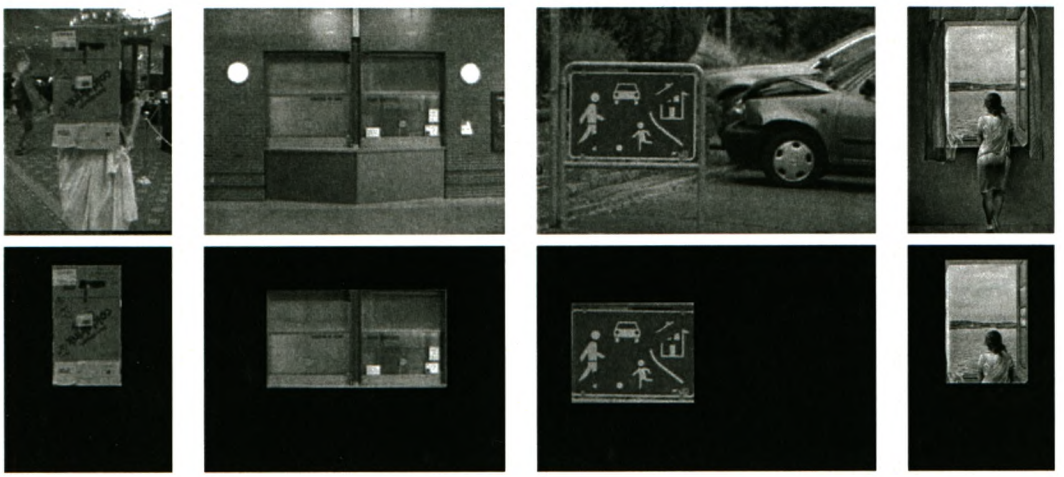


Figure 5.25: Rectangle shape prior.

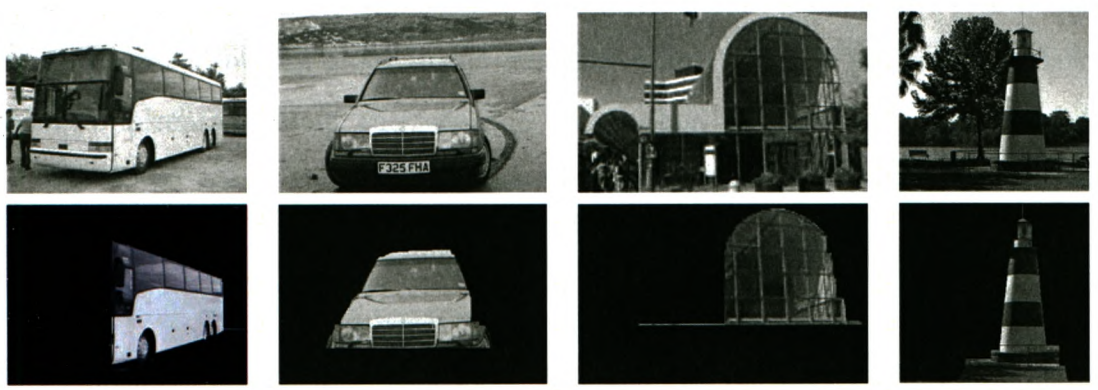


Figure 5.26: Trapezoid shape prior.

Chapter 6

Conclusions

6.1 Conclusions

In this thesis, we present a geometric labeling method to automatically extract rough 3D information from a single 2D image through labeling the scene into five geometric classes (namely, "bottom", "left ", "center", "right", and "top"). We formulate the geometric labeling problem as an energy minimization problem and optimize the energy with a graph cut based algorithm.

Our method encourages spatial consistency between neighboring pixels while preserving discontinuities along image intensity edges. We also incorporate relative ordering information about the labels in our method as ordering constraints arise naturally in the scene. Experimental results show that it is possible to increase the percentage of good labelings significantly by encouraging spatial consistency as well as incorporating the ordering constraints.

In this thesis, we also develop new graph-cut moves which we call order-preserving moves. Experimental results show that order-preserving moves performs significantly better than commonly used α-expansion when ordering constraints are used as there is a significantly improvement in computational efficiency and optimality and a modest improvement in pixel labeling as well.

We have tested our geometric labeling method on a large set of image data including indoor scene, outdoor scene, outdoor nature scene, and on cartoon Mickey

Mouse images. Experimental results show that encouraging spatial consistency and incorporating ordering constraints significantly improves the performance.

We also demonstrate the usefulness of the extracted 3D structural information of the scene in the applications of novel view generation, virtual scene walk-through, semantic segmentation, scene synthesis, and scene text extraction. In addition, we use our order-preserving moves for certain simple shape priors (i.e. rectangle and trapezoid) in graph-cut segmentation.

In summary, our geometric labeling method has the following main contributions:

(i) Our proposed method can automatically extract the rough 3D information with high accuracy and efficiency. Here the accuracy is in terms of correct pixel labeling.

(ii) We formulate the problem in a global optimization framework where we address the spatial consistency of the labels in the scene by formulating an energy function which encourages spatial consistency between neighboring pixels while preserving discontinuities along image intensity edges.

(iii) We incorporate relative ordering information about the labels in our energy function as ordering constraints arise naturally in the scene.

(iv) Most importantly, we develop new graph-cut moves, called order-preserving moves, which perform significantly better than α-expansion when ordering constraints are used.

(v) we have shown that our ordering constraints can also be used in other applications. For example, in the application of Mickey mouse images, we use ordering constraints to do object part segmentation.

(vi) The order-preserving moves can also be used for certain simple shape priors in graph-cut segmentation which is a novel concept for using shape priors on object segmentation.

6.2 Future Work

Experimental results show that when the class probabilities generated by the SVM classifier are reasonable, (i.e. the scene image contain mostly the major surface types: "bottom", "left ", "center", "right", and "top"), then incorporating the label ordering constraints as well as the spatial smoothness helps recover accurate (relative) 3D structure. However, when SVM results are far from reasonable, trying to satisfy the ordering constraints can worsen SVM results.

Major failures of our proposed method arose from the failures of the SVM based classifier. In order to overcome this, we plan to improve our SVM based classifier by learning more discriminant and useful features. Additional ways to improve our algorithm is to incorporate minimum size constraints on the labeled "center" region. In many failure cases, the "center" is unrealistically thin(1 or 2 pixels). It is relatively straightforward to incorporate the constraint that the labeled "center" region be no thinner than a given value into the graph cut optimization.

In addition, we are also planning to achieve the following major goals in the future:

(i) We will extend this algorithm to include temporal image sequences and incorporate temporal consistency and motion cue to the geometric labels. For example, we can incorporate the smoothness along the time axis since geometric labeling is typically continuous in time domain.

(ii) We will also incorporate high level knowledge into our framework. For example, we can incorporate the homographic transform information into our smoothness constraints since pixels within the same geometric labelled region should have the same homographic transformation.

(iii) Incorporating the geometric 3D structural information (monocular vision cues) into other applications such as multi-view 3D reconstruction:

- In the application of multiple view geometry 3D reconstruction, the geometric labeling information obtained from the proposed algorithm can be used to remove outliers and reduce ambiguity in feature correspondence matching. Therefore, it can improve the accuracy and efficiency on computing the homographic transformation from multiple images.
- Indoor scenes tend to be extremely low textured, therefore, the obtained geometric labeling information is also very useful for depth estimation as traditional stereo reconstruction may fail in such textureless scenes.

Bibliography

[1] V. Ferrari, T. Tuytelaars, and L. V. Gool, "Wide-baseline multiple-view correspondences," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, June 2003, pp. I–701–725.

[2] O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint.* MIT Press, 1993.

[3] Y. Horry, K. Anjyo, and K. Arai, "Tour into the picture: using a spidery mesh interface to make animation from a single image," in *SIGGRAPH'97 (Los Angeles, Aug. 3–8)*, vol. 3, 1997, pp. 225–232.

[4] D. Hoiem, A. Efros, and M. Hebert, "Automatic photo pop-up," in *ACM SIGGRAPH*, 2005, pp. –.

[5] ——, "Recovering surface layout from an image," *International Journal of Computer Vision (IJCV)*, accepted.

[6] Z.-G. Yang and L. Ren, "Svd-based camera self-calibration and 3-D reconstruction from single-view," in *the proc. of the Third International conference on Machine Learning and Cybernetics*, 2004, pp. 4090–4095.

[7] B. Micusik and T. Pajdla, "Autocalibration & 3D reconstruction with non-central catadioptric cameras," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR04)*, vol. 1, June 2004, pp. 58–65.

[8] V. Kolmogorov and R. Zabin, "Multi-camera scene reconstruction via graph cuts," in *Proceedings of the European Conference on Computer Vision (ECCV02)*, vol. 3, 2002, pp. 82–96.

[9] V. Kolmogorov, R. Zabin, and S. Gortler, "Generalized multi-camera scene reconstruction using graph cuts," in *In Fourth International Workshop on Energy Minimization Methods in Computer Vision and Pattern Recognition (EMMCVPR03)*, 2003, pp. 501–516.

[10] B. Goldlucke and M. A. Magnor, "Joint 3d-reconstruction and background separation in multiple views using graph cuts," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, vol. 1, June 2003, pp. I–683–I–688.

[11] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 23, no. 11, pp. 1222–1239, November 2001.

[12] C. Yuan and G. Medioni, "3D reconstruction of background and objects moving on ground plane viewed from a moving camera," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR06)*, vol. 2, June 2006, pp. 2261 – 2268.

[13] G. K.M.Cheung, S. Baker, and T. Kanade, "Visual hull alignment and refinement across time: A 3D reconstruction algorithm combining shape-from-silhouette with stereo," in *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR03)*, vol. 2, June 2003, pp. 375–382.

[14] L. Liu, I. Stamos, G. Yu, G. Wolberg, and S. Zokai, "Multiview geometry for texture mapping 2D images onto 3D range data," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR06)*, vol. 2, June 2006, pp. 2293 – 2300.

[15] H. Shum, M. Han, and R. Szeliski, "Interactive construction of 3D models from panoramic mosaics," in *Proc. of the IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 1998)*, 1998, pp. 427–433.

[16] P. E. Debevec, C. J. Taylor, and J. Malik, "Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach," in *SIGGRAPH 96*. New York, NY, USA: ACM Press, 1996, pp. 11–20.

[17] W. Jiang, M. Okutomi, and S. Sugimoto, "Panoramic 3d reconstruction using rotational stereo camera with simple epipolar constriants," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, vol. 1, June 2006, pp. 371–378.

[18] W. Jiang and J. Lu, "Panoramic 3d reconstruction by fusing color intensity and laser range data," in *the proc. of IEEE International conference on Robotics and Biomimetrics*, Dec 2006, pp. 947–952.

[19] Y. Sun, J.K.Paik, A.Koschan, and M.A.Abidi, "3D reconstruction of indoor and outdoor scenes using a mobile range scanner," in *International Conference on Pattern Recognition(ICPR)*, vol. 3, 2002, pp. 653–656.

[20] D. Martinec and T. Pajdla, "3d reconstruction by fitting low-rank matrices with missing data," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, vol. 1, June 2005, pp. 198–205.

[21] P. David, D. DeMenthon, R. Duraiswami, and H. Samet, "Simultaneous pose and correspondence determination using line feature," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, vol. 2, June 2003, pp. II–424–II–431.

[22] E.Mouragnon, M.Lhuillier, M.Dhome, F.Dekeyser, and P.Sayd, "Real time localization and 3D reconstruction," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR06)*, vol. 1, June 2006, pp. 363–370.

[23] T. J. Moyung and P. W. Fieguth, "Incremental shape reconstruction using stereo image sequences," in *the proc. of IEEE International conference on Image Processing*, 2000, pp. 752–755.

[24] T. Mckinley, M.M.McWaters, and V.K.Jain, "3D reconstruction from a stereo pair without the knowledge of intrinsic or extrinsic parameters," in *the proc. of the Second International Workshop on Digital and Computational Video*, 2001, pp. 148–155.

[25] A. Criminisi, I. D. Reid, and A. Zisserman, "Single view metrology," *International Journal of Computer Vision (IJCV)*, vol. 40, no. 2, pp. 123–148, 2000. [Online]. Available: citeseer.ist.psu.edu/article/criminisi99single.html

[26] B. Johansson, "View synthesis and 3d reconstruction of piecewise planar scenes using intersection lines between the planes," in *International Conference on Computer Vision (ICCV)*, vol. 1, 1999, pp. 54–59.

[27] M. Prasad, A. Zisserman, and A. Fitzgibbon, "Single view reconstruction of curved surfaces," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, vol. 2, 2006, pp. 1345–1354.

[28] L. Zhang, G. Dugas-Phocion, J.-S. Samson, and S. M. Seitz, "Single view modeling of free-form scenes," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, vol. 1, 2001, pp. I–990–I–997.

[29] C. Colombo, A. D. Bimbo, and F. Pernici, "Metric 3D reconstruction and texture acquisition of surfaces of revolution from a single uncalibrated view," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 27, no. 1, pp. 99–114, Jan. 2005.

[30] J. Coughlan and A. Yuille., "Manhattan world: Compass direction from a single image by bayesian inference," in *Proc. of the IEEE International Conference on Computer Vision (ICCV 1999)*, vol. 2, Corfu, Greece, 1999, pp. 941–947.

[31] E. Delage, H. Lee, and A. Y. Ng, "A dynamic bayesian network model for autonomous 3D reconstruction from a single indoor image," in *Proceedings of*

the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR06), vol. 2, June 2006, pp. 2418 – 2428.

[32] P. L. de Teruel, A.Ruiz, and L. Fernandez, "Efficient monocular 3d reconstruction from segments for visual navigation in structural environments," in *International Conference on Pattern Recognition(ICPR)*, vol. 1, 2006, pp. 143–146.

[33] Z. Zhang and H.T.Tsui, "3d reconstruction from a single view of an object and its image in a plane mirror," in *International Conference on Pattern Recognition(ICPR)*, vol. 1, 1998, pp. I–990–I–997.

[34] D. Hoiem, A. Efros, and M. Hebert, "Geometric context from a single image," in *IEEE International Conference On Computer Vision (ICCV'05)*, 2005, pp. 654 – 661.

[35] ——, "Putting objects in perspective," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR06)*, vol. 2, June 2006, pp. 2137 – 2144.

[36] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR03)*, vol. 1, June 2001, pp. 511–518.

[37] A. Ferencz, E. Learned-Miller, and J. Malik, "Building a classification cascade for visual identification from one example," in *Proceedings of the 2005 IEEE International Conference of Computer Vision (ICCV05)*, vol. 1, Oct 2005, pp. 286 – 293.

[38] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.

[39] J. Shi and C. Tomasi, "Good features to track," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, June 1994, pp. 593–600.

[40] C. Harris and M. Stephens, "A combined corner and edge detector," in *the proc. of 4th Alvey Vision Conference*, 1988, pp. 147–151.

[41] T. Lindeberg, "Feature detection with automatic scale selection," *International Journal of Computer Vision (IJCV)*, vol. 30, no. 2, pp. 77–116, 1998.

[42] A. K. Jain, *Fundamentals of Digital Image Processing*, T. Kailath, Ed. Englewood Cliff, NJ: Prentice Hall, 1989.

[43] R. Milanese, H. Wechsler, S. Gil, J. Bost, and T. Pun, "Integration of bottom-up and top down cues for visual attention using non-linear relaxation," in *IEEE*

International Conference on Computer Vision and Pattern Recongition (CVPR), June 1994, pp. 781–785.

[44] K. Cheoi and Y. Lee, "A saliency map model for active perception using color information and local competitive mechanism," pp. 315–324, 2002.

[45] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2001.

[46] V. Vapnik, *The Nature of Statistical Learning Theory.* New York: Springer Verlag, 1995.

[47] B. Boster, I. Guyon, and V.Vapnik, "A training algorithm for optimal margin classifier," in *the proc. of the fifth ACM Workshop on Computational Learning Theory*, 1992, pp. 144–152.

[48] J. C. Platt, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," 1999.

[49] H.-T. Lin, C.-J. Lin, and R. C. Weng, "A note on platt's probabilistic outputs for support vector machines," Department of Computer Science, National Taiwan University, Tech. Rep., 2003.

[50] T.-F. Wu, C.-J. Lin, and R. C. Weng, "Probability estimates for multi-class classification by pairwise coupling," *Journal of Machine Learning Research*, vol. 5, pp. 975–1005, 2004.

[51] T. Hastie and R. Tibshirani, "Classification by pairwise coupling," *The Annals of Statistics*, vol. 26, no. 1, pp. 451–471, 1998.

[52] J. Friedman, "Another approach to polychotomous classification," Department of Statistics, Stanford University, Tech. Rep., 1996.

[53] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, September 2004.

[54] R. Szeliski, R. Zabih, D. Scharstein, O. Veksler, V. Kolmogorov, A. Agarwala, M. Tappen, and C. Rother, "A comparative study of energy minimization methods for markov random fields," in *European Conference on Computer Vision (ECCV)*, 2006, pp. II: 16–29.

[55] V. Kolmogorov, "Convergent tree-reweighted message passing for energy minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 28, no. 10, pp. 1568–1583, 2006.

[56] J. Yedidia, W. Freeman, and Y. Weiss, "Bethe free energy, kikuchi approximations, and belief propagation," 2001, pp. xx–yy.

[57] J. Winn and J. Shotton, "The layout consistent random field for recognizing and segmenting partially occluded objects," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, 2006, pp. I: 37–44.

[58] D. Hoiem, C. Rother, and J. Winn, "3d layout crf for multi-view object class recognition and segmentation," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, 2007, pp. 1–8.

[59] H. Ishikawa, "Exact optimization for markov random fields with convex priors," *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 25, pp. 1333–1336, 2003.

[60] D. Schlesinger and B. Flach, "Transforming an arbitrary minsum problem into a binary one," Dresden University of Technology, Technical Report TUD-FI06-01, 2006.

[61] J. Darbon, "Global optimization for first order markov random fields with submodular priors," in *IWCIA*, 2008.

[62] R. Kindermann and J. L. Snell, *Markov Random Fields and Their Applications.* American mathematic Society, 1980.

[63] P.Felzenszwalb and D.Huttenlocher, "Efficient graph-based image segmentation," *International Journal of Computer Vision (IJCV)*, vol. 59, no. 2, pp. –, 2004.

[64] C.-C. Chang and C.-J. Lin, *LIBSVM-A Library for Support Vector Machines*, Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.

[65] *The PASCAL Object Recognition Database Collection*, http://www.pascal-network.org/challenges/VOC/databases.html.

[66] *Thailand Bilder-Photo and Image Gallery from Thailand*, http://www.thailandbilder.se/, 2006.

[67] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. of the ACM*, vol. 24, pp. 381–395, June 1981.

[68] D. A. Forsyth and J. Ponce, *Computer Vision: a modern approach.* Prentice Hall, 2003.

[69] J. Fan, H. Luo, and Y. Gao, "Learning the semantics of images by using unlabeled samples," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05)*, vol. 2, June 2005, pp. 704– 710.

[70] V. Kolmogorov, A. Criminisi, A. Blake, G. Gross, and C. Rother, "Bi-layer segmentation of binocular stereo video," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR05)*, vol. 2, June 2005, pp. 1186–710.

[71] A. A. Goshtasby, *2D and 3D Image Registration for medical remote sensing and industrial application.* Hoboken, New Jersey: John Wiley Sons, Inc., 2005.

[72] P. Viola and M. J. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR01)*, vol. 1, 2001, pp. I–511–I–518.

[73] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman, "Discovering objects and their location in images," in *International Conference on Computer Vision (ICCV)*, vol. 2, June 2005, pp. 1186–710.

[74] T. A., K. Murphy, and W. Freeman, "Sharing features: efficient boosting procedures for multiclass object detection," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR04)*, vol. 2, 2004, pp. II–762– II–769.

[75] M. Vidal-Naquat and S. Ullman, "Object recognition with informative features and linear classification," in *International Conference on Computer Vision (ICCV)*, vol. 1, Oct. 2003, pp. 281–288.

[76] F. Ge, S. Wang, and T. Liu, "Image-segmentation evaluation from the persective of salient object extraction," in *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR06)*, vol. 1, June 2006, pp. 1146–1153.

[77] X. Liu and J. Samarabandu, "Multiscale edge-based text extraction from complex images," in *Proc. of the IEEE International Conference on Multimedia Expo(ICME 2006)*, Toronto, Canada, July 2006, p. In press.

[78] ——, "An edge-based text region extraction algorithm for indoor mobile robot navigation," in *Proc. of the IEEE International Conference on Mechatronics and Automation (ICMA 2005)*, Niagara Falls, Canada, July 2005, pp. 701–706.

[79] M. Leventon, W. Grimson, and O. Faugeras, "Stat. shape influence in geodesic active contours," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, 2000, pp. 316–323.

[80] M. Rousson and N. Paragios, "Shape priors for level set representations," in *European Conference on Computer Vision (ECCV)*, 2002, p. II: 78 ff.

[81] D. Cremers, S. Osher, and S. Soatto, "Kernel density estimation and intrinsic alignment for shape priors in level set segmentation," *International Journal of Computer Vision (IJCV)*, vol. 69, no. 3, pp. 335–351, September 2006.

[82] D. Freedman and T. Zhang, "Interactive graph cut based segmentation with shape priors," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, 2005.

[83] M. Kumar, P. Torr, and A. Zisserman, "Obj cut," in *IEEE International Conference on Computer Vision and Pattern Recongition (CVPR)*, 2005, pp. I: 18–25.

[84] J. Keuchel., "Multiclass image labeling with semidefinite programming," in *ECCV*, vol. II, 2006, p. 454C467.

[85] A. K. Sinop and L. Grady, "Uninitialized, globally optimal, graph-based rectilinear shape segementation," in *International Conference on Computer Vision (ICCV)*, 2007.

Glossary

Binary image – An image whose pixel only has two logical values: 0's (off pixels) and 1's (on pixels).

Compass filtering – A widely used directional filtering operation, which can be used to extract edges in different directions.

Color image – A multispace image in which each space represents a specific color plane.

Data term – An energy cost term, which specifies the penalty for a single pixel to have certain label, and thus encourages each pixel to be assigned the label of smallest penalty.

Digital image – A matrix of numbers that can be displayed.

Digital image processing– Manipulation of images by a digital computer.

Directional filtering– A bank of filtering operations which are normally used to extract image edges in different directions.

Difference of Oriented Gaussian (DOOG)– A bank of filtering operations, which are normally be used to do texture analysis by taking the difference between a set of oriented gaussian operation.

Filtering – A neighborhood operation, which works with the values of image pixels in the neighborhood and the corresponding values of a sub-image which has the same dimensions as the neighborhood.

Graph Cut – An energy minimization method, which can be employed to efficiently solve a wide variety of computer vision problems, where energy minimization problems

can be reduced to instances of the maximum flow problem in a graph (and thus, by the max-flow min-cut theorem, define a minimal cut of the graph).

Gray-level – A value associated with a pixel, corresponding to its brightness.

Histogram – A function which summarizes the gray-level content of an image by showing, for each gray level, the number of pixels in the images that have that gray-level.

Intensity image – pixel value represents the brightness or gray level with a range from 0 to $2^n - 1$, where the intensity 0 usually represents black and the intensity $2^n - 1$ represents full intensity (white), n is the image storage bits.

Pixel – A single point or element in a digital image matrix.

Point operations – also called contrast enhancement or contrast stretching operations, are gray-scale transformations that take a single input image into a single output image in such a way that each output pixels' gray level depends only upon the gray level of the corresponding input pixel.

Segmentation – A partition procedure which subdivides an image into its constituent regions or objects.

Sharpening – A fundamental filtering operation which highlights fine details in an image by spatial differentiation.

Smoothing – A fundamental operation in image filtering which reduces or smoothes out sharp transitions in gray levels.

Smoothness term – An energy cost term, which encourages spatial consistency by penalizing neighboring pixels that are not assigned the same label.

Superpixel – An image region returned by a segmentation algorithm.

Support Vector Machines (SVM) – A set of related supervised learning methods used for classification and regression which belong to a family of generalized linear classifiers.