ROBUST METHODS FOR MULTIPLE MODEL DISCOVERY IN STRUCTURED AND UNSTRUCTURED DATA

BY SAKET ANAND

A dissertation submitted to the

Graduate School—New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Professor Peter Meer

and approved by

_____

_____

_____

_____

_____

New Brunswick, New Jersey

October, 2013

ABSTRACT OF THE DISSERTATION

Robust Methods for Multiple Model Discovery in Structured and Unstructured Data

by Saket Anand

Dissertation Director: Professor Peter Meer

Humans excel at identifying and locating multiple instances of objects or persons in a scene, despite large variations in lighting conditions, pose or scale. In order to achieve this level of robustness, humans naturally use high-level semantic information. The notion of semantics is hierarchical in nature, for example, a house is constituted of walls, floor and ceiling. When certain entities in this hierarchy can be modeled using a functional form known a priori, we refer to the data as structured, while when the functional form is unknown, the data is unstructured. In this thesis, we address the problem of discovering multiple instances of models in structured and unstructured data.

The first part of this thesis deals with structured data. We identify planar regions in an indoor scene by using a single depth image from a Microsoft Kinect sensor. The clutter in the indoor scenes, the depth dependent measurement noise and the unknown number of planar regions pose serious challenges in model discovery. We propose a scalable bottom-up approach that leverages from a heteroscedastic, i.e., point dependent model of the measurement noise.

The second part of the thesis addresses multiple model discovery in unstructured data in a semi-supervised setup. We develop a framework for using mean shift clustering

in kernel spaces with a few user-specified pairwise constraints. A linear transformation of the initial kernel space is learned by the constrained minimization of a Bregman divergence based objective function. We automatically determine the adaptive bandwidth parameter to be used with mean shift clustering. Finally, we compare the performance with state-of-the-art semi-supervised clustering methods and show that kernel mean shift clustering performs particularly well when the number of clusters is large.

We also propose a few directions for future research. Using the planar regions detected from the first frame of Kinect, a sequence of RGB-D images can be rapidly processed to dynamically generate a consistent 3D model of the scene. We also show that for the kernel learning problem, we can use ideas from group theory and semi-definite programming to devise a more efficient algorithm that only uses linearly independent constraints.

Acknowledgements

First of all, I would like to thank my advisor, Prof. Peter Meer for his support and guidance throughout the duration of my doctoral studies. He has contributed immensely, both directly and indirectly, in my learning of the technical and non-technical aspects of research. I want to take this opportunity to express my gratitude to Ioana and Peter Meer for their hospitality and many enjoyable and delicious meals we had together.

This work would not have been possible without the support from Siemens Corporate Technology and I acknowledge, with great appreciation, the research collaboration facilitated by Dr. Dorin Comaniciu. I am grateful to Dr. Maneesh Singh for his guidance throughout the duration of this collaboration. I would also like to thank Dr. Vivek Singh for taking time out for discussions. I have learned much from the discussions with Maneesh and Vivek at Siemens.

I appreciate the involvement of Dr. Oncel Tuzel for his valuable inputs relevant to my work on semi-supervised clustering. I could not have achieved this feat without the innumerable discussions of ideas, both technical and otherwise, with my friend Dr. Sushil Mittal. I am thankful to Swetha Sunkara for her enthusiasm in making plans for vacations, road trips, movie nights and dinners despite many of them being run over by the pressures of academia. Sushil and Swetha certainly made life in graduate school much more fun. I would also like to thank Orsan Aytekin, my colleague at the Robust Image Understanding Lab, for the few but interesting discussions that we have had.

I want to express my deepest appreciation for my parents, Madhu and Gyan Anand, who unconditionally supported me in all my decisions, including that of pursuing my PhD. Their patience and love has made it a lot easier to spend several years in a country far away from home. Finally, I would like to thank my wife Vasvi, who has made the last one year so much more pleasant because of her presence.

Dedication

To Daddy, my grandfather,

for instilling in me, the desire for learning to learn

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Pattern discovery is a very important problem in data analysis. Depending on the type of data, different approaches are employed for discovering patterns of interest. When the data follows a known constraint represented in a functional form, regression analysis can be used, while when the functional form is unknown *a priori*, cluster analysis is often the chosen technique. Traditionally, both these data analysis techniques have been mostly applied in an unsupervised setting. However, with the advances in machine learning, there has been a significant interest in incorporating supervision in clustering and regression [25] methods.

Visual data like images or video usually impose serious challenges in pattern discovery. There are several factors contributing to the huge variability in visual data. The 3D to 2D projective transformation in the image formation process is highly nonlinear and severely distorts the shape of objects in the scene. The scene lighting can cause extreme variations in the image brightness levels like specular reflections and shadows. Moreover, being very high-dimensional, visual data typically contain a lot of spurious information that interferes with the task of pattern discovery. In order to deal with these problems, a feature representation of the image is used that captures the elements relevant to a particular task.

Feature based representations have been extensively explored and are not restricted to visual data alone. Speech and speaker recognition often involves extracting mel cepstral (MFCC) features [92], while text document classification is often done using bag of words. With the variability in visual data, it is imperative to tailor the design of the feature extraction module for a specific task. For example, keypoint detectors like the Scale Invariant Feature Transform (SIFT) [72] or Speeded-up Robust Features

(SURF) [8] are usually employed for point matching, object detection etc. Similarly, Histogram of Oriented Gradients (HOG) features [26] are used for human detection in both images and video. Despite the immense body of work on feature extractors that are designed to compactly represent relevant information in visual data, the variability in such feature descriptors is still large. Pattern discovery methods often work with such feature based representations as opposed to raw data.

When the data satisfies a known parametric constraint, its model is represented by the functional form of the constraint, e.g., a line in a binary edge image. Since real data is noisy and deviates from the model, a pattern discovery method also has to distinguish the *inlier* data points that adequately satisfy the model, from the spurious outliers. For example, pixels in the edge image that lie along the line with residuals, say, smaller than s pixels are inliers while the others are outliers. This problem becomes harder when there are an *unknown* number of instances of the model present in the data, each corrupted with a different and *unknown* scale of noise.

When the parametric form of the model is unknown, clustering methods are often employed to discover multiple patterns and the cluster centers are used to represent the models. Clustering algorithms group together data points that are similar to each other based on some similarity or distance measure. Clustering methods have been applied to different kinds of data [54], in an unsupervised setting as well as with supervision information [7, 113, 74].

In this thesis, we first describe the terminology used and briefly overview different techniques employed for multiple pattern discovery in Chapter 2. We also present some previous work done in indoor scene modeling and motivate the need to exploit the scene geometry that imposes structure in the corresponding 3D visual data. Finally, we state the problem of discovery of multiple models in unstructured data.

In Chapter 3, we present a method to characterize the sensor noise inherent to depth sensors and use it to perform planarity preserving superpixel segmentation of a 3D image. It is known that the noise in depth measurements is *heteroscedastic* [49, 57, 4], i.e. the variance of error for each measurement is different. The noise characterization of depth sensors involves estimating the expression of the noise variance function and

can be analytically derived using the formulation of the depth estimation mechanism. The analytical expression of the standard deviation of error in the depth estimates has a quadratic dependency on the true depth being measured. We apply the heteroscedastic noise model to oversegment the 3D images obtained from the depth data.

In Chapter 4, we describe a scalable, bottom-up approach for parametrizing the depth images from a calibrated Microsoft Kinect sensor. The approach leverages from the fact that depth images of indoor scenes are structured comprising largely of multiple planar surfaces. We robustly identify the planar models in a single frame from Kinect and the pixels associated with them. Using the intrinsic parameters of Kinect, the planar models and the heteroscedastic noise model, we can represent the depth image in a semiparametric form. The depth of pixels associated with a planar model are parametrized by the model parameters and the pixel location, while depth at pixels in nonplanar regions are retained as it is. This representation of the depth image can be useful in multiple applications like 3D model building, image compression for transmission and planar segmentation for applications like object detection or recognition.

We develop a scalable system with most of the modules easily parallelizable. The main contributions in this work are listed below –

- we propose a hierarchical framework with surface normals computed at each pixel at the lowest level, which are agglomerated to form superpixels and then larger planar fragments at the highest level.
- we make use of the sensor noise model to propagate the uncertainty through the levels of this hierarchy.
- we develop a novel region growing algorithm that uses a heteroscedastic planar distance function for superpixels to identify all significant planar regions in the scene.

The design is motivated by developing an automatic, fast and scalable system that can generate reliable estimates at each level in the system hierarchy. We evaluate the system's qualitative and quantitative performance on several images from realistic datasets of different complexities.

In Chapter 5, we develop a semi-supervised kernel mean shift based clustering for pattern discovery in unstructured image data. The method uses pairwise constraints to learn a linear transformation of the initial kernel space and determine the parameters to be used for mean shift clustering. We compare the performance of the proposed method with that of other state-of-the-art semi-supervised clustering methods and show favorable performance of mean shift, despite the lack of knowledge of the number of clusters.

We make the following contributions in this work –

- we develop a semi-supervised framework for mean shift clustering that exploits *both* similarity and dissimilarity constraints.
- we show that a linear transformation of the kernel space is more desirable as opposed to the null space projection proposed by [113], and establish the relationship between learning the linear transformation and the $\log\det$ divergence minimization.
- we reformulate the kernel learning as a $\log\det$ divergence minimization problem and solve it using the approach of [55].
- we design a technique that uses the similarity pairwise constraints to determine a viable adaptive bandwidth parameter for the mean shift clustering.

The evaluation highlights the advantages of using mean shift clustering over other methods, especially when the number of clusters in the data is large.

In Chapter 6, we discuss some possible future directions. We developed a method to parametrize a single depth image from Microsoft Kinect using planar structures in Chapter 4. We propose to use this parametrization of the first frame in a sequence to efficiently process the following RGB-D frames to generate a 3D model of the scene. We also propose to merge information from both the RGB and depth channels for better estimation.

We show that the kernel learning method proposed in [55, 63] used in Chapter 5 does redundant computations by iterating over every user-specified pairwise constraint.

We reformulate the problem to show that the linear transformation needs to be applied only to the subspace spanned by the constraint vectors, which is usually much smaller in dimensionality than the entire kernel space. Using ideas from group theory and semi-definite programming, an algorithm can be devised that only uses linearly independent constraints, thus removing the redundancy in the learning algorithm.

Chapter 2

Multiple Model Discovery

Real world applications often present the problem of detecting multiple patterns in data. The structures of interest can be modeled using a parametric form or can be represented using a nonparametric model. The problem of discovering multiple models in data is ill-posed due to the following reasons. Depending on the dataset being used, the number of models present can vary significantly. The distribution of the noise corrupting the 'inliers', i.e., data points belonging to a structure of interest, is unknown. Additionally, real-data is usually corrupted with spurious 'outliers', i.e., data points without any particular structure, which interfere with the detection of the inliers. Along with design issues like scalability and efficiency, it is important to consider these challenges while designing a practical system for automatic pattern discovery. When the structure of interest can be modeled using a parametric form, we refer to the data as *structured*. When the model representation is nonparametric, or if the parametric form is unknown, we refer to the data as *unstructured*.

2.1 Approaches for Multiple Model Discovery

The goal of model discovery is to determine the complete set of inliers, i.e. points that are consistent with the model, while rejecting the outliers. While working with structured data, robust regression methods are often employed to estimate the model parameters and the associated inliers. In case of unstructured data, usually clustered methods are chosen for grouping similar points. In the following sections, we provide a brief overview of these broad areas that have been studied extensively.

2.1.1 Robust regression

There are several approaches for regression in the presence of outliers. The most popular ones are variants of Random Sample Consensus (RANSAC) [35], which use a hypothesize and test framework. A fairly extensive survey of RANSAC like methods is given in [94]. These methods randomly generate several model hypotheses and optimize an objective function for selecting the best one. Robust least squares estimation methods like M-estimators [79, Sec. 4.4.2, pp 163] can be employed to recover from biased estimates due to the presence of outliers. However, both, the random sampling based methods and the robust least-squares need a user-specified parameter for the scale of inlier noise, which cannot always be supplied.

There exist other methods that do not explicitly detect the scale of the noise to reject outliers, e.g., STARSAC [18] or RECON [93]. Other methods use scale estimation techniques like [119] as a preprocessing step for RANSAC like methods. However these methods are relatively sensitive and break down when the number of outliers grow large.

Generalized Projection Based M-estimator (gpbM)

The gpbM algorithm is a robust regression method that falls under the hypothesize and test category. Manually specifying the scale parameter becomes hard in cases when the noise distribution or scale changes over time [109]. In gpbM, we estimate the scale of the noise in an inlier structure from the data itself. We also account for the heteroscedasticity in the data during the estimation. The entire algorithm is executed in three distinct steps, which we briefly describe below. For a detailed description of the algorithm with comparisons with state-of-the-art robust methods like Ordered Residual Kernel [17] and J-Linkage [111] , we refer the reader to [82].

Step 1 - Heteroscedastic Scale Estimation. Scale estimation is the first step of the algorithm. We reformulate an equivalent problem of estimating the fraction of points belonging to an inlier structure. For this we generate hypotheses using randomly selected *elemental subsets*, i.e., the smallest set of points required to generate a model hypothesis, e.g. three points for a plane. The data points are all projected to the null

space of the model hypothesis. The fraction of points is varied between $(0, 1]$ in 40 uniform steps and for each fraction a measure of volume of the smallest enclosure is computed using the pointwise Mahalanobis distances. This volume measure is used to estimate the density, which is computed for each hypothesis at every fraction. Using the combination of the peak density values for each hypothesis and the consensus among all randomly generated hypotheses, we select the estimated fraction for one structure. The scale is computed as the smallest bounding box enclosing the estimated fraction of points. The corresponding inlier points lying within that scale are retained for the next step.

Step 2 - Heteroscedastic Model Estimation. In this step, we estimate the model by optimizing a heteroscedastic objective function that is similar to kernel density estimate [29]. We again generate random hypotheses by selecting elemental subsets from the candidate inliers discovered in the previous step. After the projection of the data points to the null space of the hypothesis, the optimization is performed using the mean shift algorithm [23] and the mode of the kernel density is found. The model that maximizes the kernel density at the mode is selected as the final model hypothesis.

Step 3 - Inlier/Outlier Dichotomy. In this step, after projecting all the points to the null space of the selected model hypothesis from the previous step, we start mean shift iterations from each projection. The projected points that converge close to the mode are classified as the inlier points for this structure. All the inlier points are removed and the algorithm repeats the entire procedure. The algorithm stops when all the points are assigned to inlier structures or when the kernel density estimate at the mode drops by a factor of 20.

2.1.2 Clustering

When the data is unstructured, clustering methods are usually applied to group together similar data points. The model is identified as the cluster center, and the members of the clusters are the inliers associated with the model. Depending on the clustering technique used, the strategy for grouping together data points is based on a distance

function or a similarity measure. Usually, partitioning based methods like k-means [54] and density based methods like DBSCAN [32] make use of the distance function. Graph based clustering methods like spectral clustering [86, 121] and affinity propagation [37] need a pairwise similarity function between the nodes of the graph. Clustering over graphs is also posed as a graph labeling problem and is solved efficiently using graph cuts [10].

Most RANSAC like algorithms pose the regression problem as a subspace estimation problem. An interesting adaptation of graph labeling using graph cuts was proposed in [52] for geometric model fitting. The method used random sampling for generating label hypotheses and used a label cost term in the energy function being minimized. A new class of robust subspace estimation methods have gathered a lot of interest recently [30, 12]. These methods decompose a data matrix to identify a low-rank matrix and a sparse error matrix. Using this decomposition, the method identifies the outliers and the structures in different subspaces by optimizing an $L1$-norm based objective function.

2.2 Multiple Model Discovery in Structured Data

Indoor man made environments usually comprise entities having simple geometric form like cylindrical columns, planar walls, etc. Data acquired by sensing such environments captures the geometry and is strongly structured. In this section, we present our initial work in building indoor scene model using the Microsoft Kinect sensor, which captures depth augmented RGB (RGB-D) images. We motivate the need to exploit the geometric structure in the scene and state the problem of discovering multiple planar models in an RGB-D image. This is a crucial step towards several applications of scene understanding like building 3D models, planar segmentation for object recognition/detection and range image compression.

Building a model of visual scenes, especially man-made environments, is useful for a variety of applications. Recently, 3D models have been used to assist humans to navigate through large man-made buildings [120]. In virtual and augmented reality

applications, often photo-realistic models are created from real images and GPS information. There exist consumer systems like Photosynth [81] that build panoramic views and 3D models of outdoor scenes from a large number of 2D images. While augmented reality applications serves the consumer with mobile applications [110], urban models are also widely used by government agencies for planning and development, climate studies, public safety and emergency evacuation strategies etc. Large scale urban models are often created by using 2D aerial images [59] or ground and aerial LIDAR scans [91]. These models could be easily incorporated in robotic systems for autonomous navigation and dynamic map building using technologies like Simultaneous Localization and Mapping (SLAM) [106].

Scene modeling methods can be classified into two categories based on the type of input they use. The first category of modeling methods use several 2D images to build 3D models of buildings. The method described in [39, 2] uses a combination of structure from motion, multi-view stereo and a stereo algorithm to automatically generate 3D models of indoor and outdoor architectural scenes. The algorithm makes the *Manhattan world* assumption, i.e. the scene contains predominantly piece-wise planar structures with three dominant directions. Other methods which make use of a single image to create a 3D model also make the Manhattan world assumption. The methods in [67, 47] detect the vanishing plane to identify three mutually orthogonal planes and hence the walls, floor and ceiling. The algorithms were extended to locate objects like furniture in indoor scenes by detecting bounding boxes [48, 66]. More recently, this idea was explored further and more efficient methods were proposed in [100, 89, 101]. Based on these methods, Gupta et. al. used the room layout to identify human action spaces in the scene [44].

The second category of scene modeling systems uses 3D data directly in the form of point clouds. With the availability of 3D-augmented RGB data acquisition systems: LIDAR (light detector and ranging) [90], laser scanners, Microsoft Kinect [80], dense stereo etc., several approaches of indoor scene modeling, specifically for 3D data have been developed. Gallup et. al. [40] present a method that builds upon a stereo based system to compute a disparity map that assigns probabilities to planar and non-planar

regions. There also exist methods to create large urban environments using 3D point clouds obtained from LIDAR (light detector and ranging) like sensors [65]. In order to process dense 3D point clouds obtained from 3D sensors, algorithms have been proposed for building semantically rich 3D models that make use of underlying geometry of the structure. For e.g., Chauve et. al. [14] present a method of representing 3D structures by making use of planar primitives. The authors build a complete 3D model by filling holes and represent complex 3D structures by an adaptive decomposition into planar primitives.

A method that makes use of more complex geometric primitives, i.e. planar surfaces and conic or cylindrical structures is described in [99]. The algorithm adapts RANSAC [35] to efficiently discover geometric structures by modifying the sampling strategy for hypothesis generation and the hypothesis evaluation method. Schindler et. al. [98] propose a method that uses an initial dense mesh over a point cloud to generate a planar segmentation and uses pairwise constraints to assign semantic labels to planar segments. Along with building geometric models of the scene, [60] captures semantic information by using contextual relationships between objects and geometric structures present in the scene. Plane based modeling of 3D point clouds has also been addressed in [116], however in context of planetary robotic exploration. The authors motivate the plane based modeling in order to reduce the complexity of the mapped environment for efficient transmission. Using the Kinect sensor, the authors present a GPU based real-time robust and interactive indoor modeling system named "KinectFusion" in [85, 53]. The method shows that the Kinect sensor can be used to build reliable, high-quality indoor models.

In Section 2.2.1, we present our initial results with different approaches to creating indoor models using 3D data captured from Microsoft Kinect and motivate the need to use geometric primitives to develop a scalable system. In Section 2.2.2, we discuss multiple approaches for discovering multiple planar models motivate a bottom-up approach that retains the flexibility to add top-down semantic constraints.

2.2.1 Indoor Scene Modeling

Since the availability of the Microsoft Kinect sensor, several systems were developed that used the RGB-D images to register 3D point clouds and build indoor scene models. RGB-D SLAM described in [31] is a SLAM algorithm that also generates a dense textured 3D model of indoor scenes. The RGB images are used to find point matches between consecutive frames using features like SIFT [72] and SURF [8]. The point correspondences are used to compute the rigid body transformation between consecutive frames robustly using RANSAC [35]. These transformations are used to track the camera and populate a pose graph which is further used for global optimization of the camera trajectory. A similar approach was used in RGBDemo [76], which relied on SIFT for matching features to compute the incremental transformation. Since both these methods used a color based feature detection, they faced problems when the lighting conditions varied or when there were textureless regions in the scene. KinectFusion avoids these problems by applying a variant of the Iterative Closest Point (ICP) [96] and thus is robust to lighting condition and texture quality in the scene. An open source implementation of KinectFusion was released with the Point Cloud Library [97]

In our experiments, we use RGB and depth frames of indoor scenes grabbed from a calibrated Kinect sensor. Our approach is similar to RGB-D SLAM and RGBDemo for obtaining a registered point cloud representation of the scene. We used the RGB images to detect and match lightweight features from the `libviso` library [41]. The `libviso` is a C++ library with a Matlab interface and supports methods for visual odometry based on 2D streaming video. The feature matching module from the `libviso` library usually returns a large number of reliable matches and a small percentage of outliers when the changes in camera position are small. For a smooth motion of the Kinect sensor, consecutive frames are close, and we obtain a large number of matches for a pair of frames. Fig. 2.1 shows the matches returned by the feature matcher.

Since there still could be wrong matches, we used RANSAC based robust estimation of the incremental transformation over the matched features. For robust estimation, we generated several random hypotheses using *elemental subsets*, i.e. the smallest set

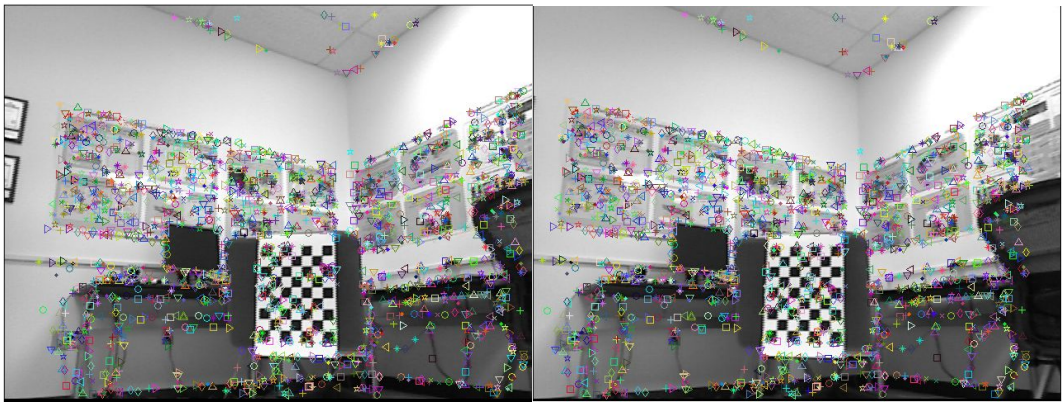


Figure 2.1: Example pair showing point matches found using libviso across two consecutive frames. The camera went through a 5 degrees rotation.

of points used to generate a 3D rigid transformation hypothesis. The RANSAC algorithm is provided with an error threshold of 3mm. The points that lie within this error threshold are called inliers and satisfy the hypothesis. RANSAC selects a hypothesis that maximizes the number of inlier points. Given the set of point matches $\mathbf{P}^i = \left[\mathbf{p}_1^i, \ldots, \mathbf{p}_{n_{1i}}^i\right]$ and $\mathbf{P}^{i-1} = \left[\mathbf{p}_1^{i-1}, \ldots, \mathbf{p}_{n_{1i}}^{i-1}\right]$ between the ith and the $(i-1)^{th}$ frames respectively. Once the algorithm converges, we estimate the incremental transformation $\mathbf{T}_{i|i-1}$ using all the n_i inlier points. The incremental transformation satisfies $\mathbf{P}_{i-1} = \mathbf{T}_{i|i-1}\mathbf{P}_i$, $i = 1, \ldots, n_i$. Using this incremental transformation, the cumulative transformation is computed with respect to the first frame $\mathbf{T}_{i|1} = \mathbf{T}_{2|1}\mathbf{T}_{3|2}\cdots\mathbf{T}_{i|i-1}$. This transformation is applied to the entire point cloud obtained from the current frame. We stored the registered point cloud in an octree [78] based structure. An octree is a tree data structure with eight children at each level. It is commonly used to represent a three dimensional Euclidean space.

An example of reconstruction of an office cubicle space is shown in Fig. 2.2. It can be seen that the planar walls of the cubicles do not form straight lines in the top view in Fig. 2.2c. This is due to errors inherent in the Kinect sensor and the errors in the estimation process. The gap between the walls on the left and right of cubicle 2 are a result of accumulated drift. These artifacts are very difficult to deal with in a point cloud or mesh based model, unless geometric constraints are explicitly enforced. Our point cloud based modeling system hit a memory bottleneck at about 10 million

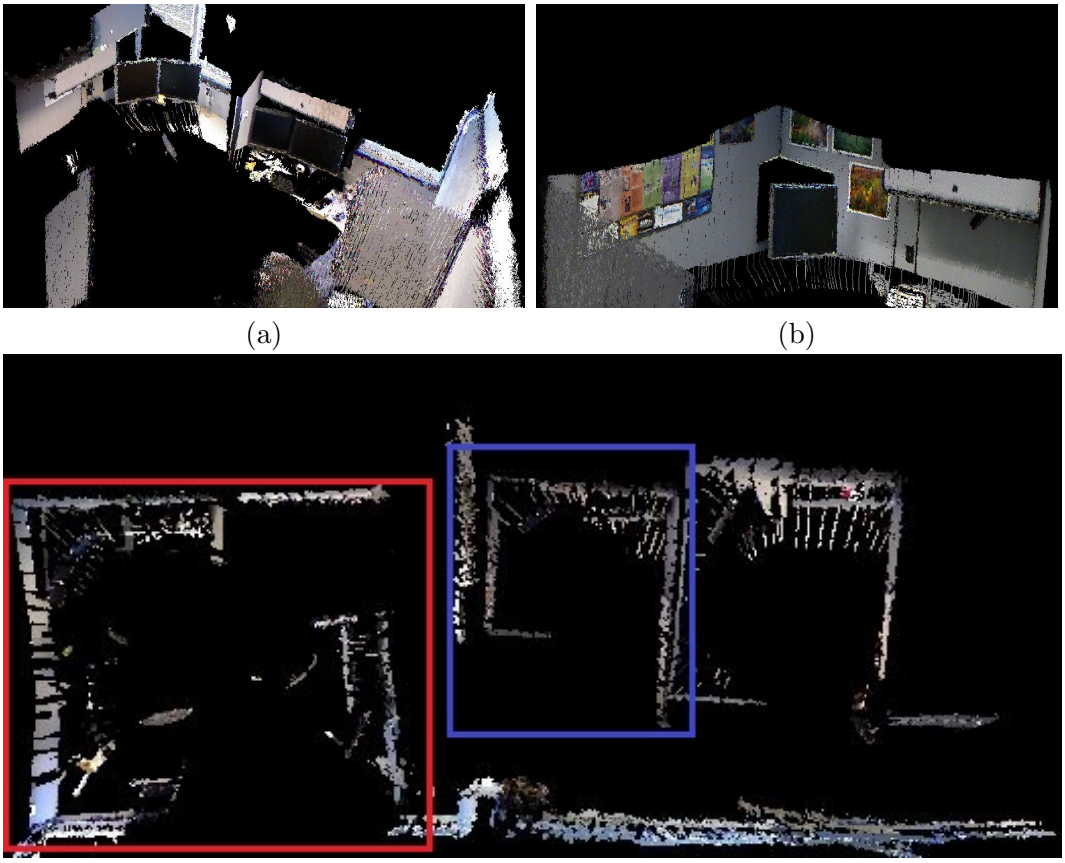
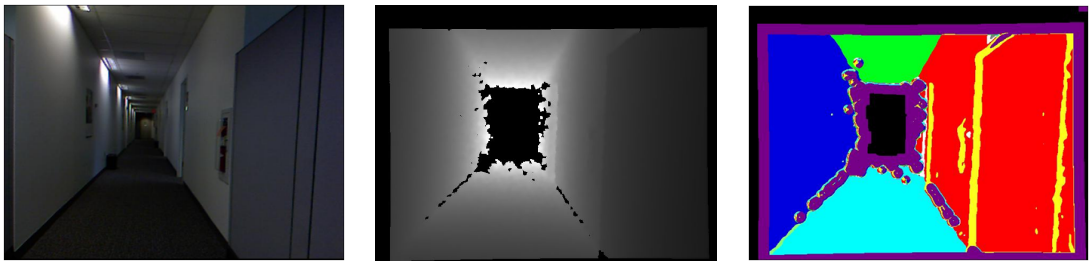


Figure 2.2: Cubicle space point cloud model. (a) Cubicle 1 close up view. (b) Cubicle 2 close up view. (c) Top view of the entire model. Red box: Cubicle 1. Blue box: Cubicle 2. Cubicle 3 in the right is not shown in the close up view.

points, which means that at full VGA resolution (640×480), the model could only handle about 30 nonoverlapping frames. This raises concerns with the scalability of the model with respect to large buildings. Moreover, the feature matching approach was not effective in case of textureless regions, e.g. corridors where the walls have similar colors without unambiguous feature patterns.

Note that neither of the modeling methods discussed in this section scale well. Secondly, it is not straightforward to incorporate semantic information in the representation presented in the methods above. In the following section, we address the scalability issue, by detecting geometric structures, planes in particular, to represent large regions.



(a) (b) (c)

Figure 2.3: Input RGB-D image for plane detection. (a) RGB image (b) Depth image. (c) Segmented image. Red, green, blue, cyan and yellow show the different planes detected in the scene. The purple points are labeled as the nonplanar outliers.

2.2.2 Modeling using Planar Structures

A simple strategy to decrease the memory requirements for storing a model is to replace large planar regions by the bounding polygon and planar equations of point clouds. In case textures are to be retained for photo-realistic models, the planes could be texture mapped by using planar homographies [46, Ch. 13]. Usually indoor environments are cluttered with objects which may have a non-planar shape. It is important to identify these 'outliers' in order to correctly detect the planar models.

2.2.3 Robust Plane Fitting

We apply the gpbM algorithm [82] to automatically detect the planar structures in a 3D point cloud obtained using the depth images from a calibrated Kinect. In order to keep the run time low, we downsample the point cloud by a factor of four. Fig. 2.3 shows the RGB-D image input from Kinect and the corresponding segmented depth image. The individual planes detected are shown in Fig. 2.4. We observe that even though we get a good quality detection of planar segments in this example, from a model building standpoint, the segmentation is not sufficient. For example, the outliers detected at the intersection of the ground plane and the adjacent wall in Fig. 2.3c will not generate clear boundaries. In the following section we discuss some limitations of these methods and list the desirable properties of a model building system.

Figure 2.4: The four dominant planes detected by applying gpbM to the 3D point cloud.

2.2.4 Constrained Planar Modeling

Using robust methods for plane detection results in accurate detection of large planar structures and the corresponding points. However, robust methods that use a hypothesize and test framework are restrictive and have the following limitations. Firstly, applying iterative algorithms for plane detection has high computational cost, since each iteration has at least linear complexity in the number of pixels. The point clouds need to be downsampled in order to achieve feasible run times. Secondly, these algorithms work well when the number of planes to be detected are small. In a cluttered environment, where small planar structures are present, the accuracy of detection deteriorates, while the computational complexity does not reduce. Finally, these algorithms do not guarantee clean boundaries at intersections of planes and could lead to disconnected outlier points.

In order to model larger environments, the modeling system should be flexible and scalable. We list a few desirable properties of a scene modeling system.

- *Bottom-up* – should start grouping at the pixel level to large planar/nonplanar segments. The model should account for measurement errors introduced at the lowest level and propagate this to higher levels.

- *Robust* – to different conditions such as lighting changes, textureless region, clutter, moving entities etc.

- *Flexible* – in its representation to easily incorporate high level semantic constraints to improve final segmentation.

- *Scalable* – to large structures without large memory requirements.

To achieve these properties, we propose a hierarchical framework where we take a bottom-up approach to discover all planar regions in a scene. We begin with the noise characterization of the Kinect depth sensor. We account for this depth dependent noise in our heteroscedastic superpixel segmentation algorithm. We apply this algorithm to generate a segmentation comprising of planar patches. This reduces the complexity of the point cloud from hundreds of thousands of points to a few hundreds of superpixels. Using the oversegmented images and the heteroscedastic noise model we propose to combine planar patches to identify the unique set of planar models that explain the entire scene. The noisy depth image can then be parametrized by this set of planar models and their corresponding regions and the nonplanar regions can be represented as the original measurements. This parametrization provides the desired compression along with a planar segmentation which can be employed for applications like localization, object detection and recognition as well as model building.

2.3 Multiple Model Discovery in Unstructured Data

Unstructured data is often obtained as a result of feature extraction and is grouped together based on some distance (similarity) measure. The change of representation is usually performed to prune away spurious information and extract descriptors that are relevant for a given task. However, this feature extraction is usually a nonlinear process and does not always map the original data to a space where the default distance (similarity) function yields meaningful clusters. Most feature descriptors capture low-level information in the data, while the *default* cluster assignment is largely guided by the distance (similarity) function and may not generate a *desired* clustering. This problem of associating data samples with a model is exacerbated when the desired clustering is based on semantic similarity between data samples that is not captured by the low-level feature representation. This is in contrast with the case of structured data, where the precise mathematical form of the constraint function is known *a priori*.

In order to deal with the nonlinearities in the feature space, kernel methods are often employed to map the input data to a higher dimensional space, where the distance function may be more meaningful. Moreover, semi-supervised variants of clustering

methods have been developed, which either guide the clustering method by imposing constraints [118] on data samples or modify the default distance function [61] to a more meaningful measure. Graph based methods also modify the affinity values based on the user-specified constraints [43]. We propose to use a semi-supervised mean shift based technique applied to kernel spaces for clustering unstructured data. The proposed method uses pairwise constraints and automatically discovers all the modes of the underlying kernel density estimate, which represent the desired models. We plan to evaluate the algorithm on image and scene categorization, face clustering and digit recognition problems.

2.4 Discussion

We showed some initial results with indoor scene modeling using the Kinect sensor. We showed that we can discover the planar structures using the 3D image of an indoor scene. However, for planar model discovery, simply detecting a number of planes is not enough to completely represent the scene. We proposed a hierarchical framework that leverages from the sensor noise model to discover all planar regions in the scene and use it to parametrize the depth image. We also discussed the importance of model discovery in unstructured data and the challenges it imposes.

Chapter 3

Heteroscedastic Superpixel Segmentation

Superpixel segmentation is an important module in several computer vision applications like image segmentation, object detection, recognition and localization. The primary advantage of using superpixels is to reduce complexity for the subsequent modules. This is achieved through oversegmentation of the image by grouping spatially correlated pixels having similar features. This grouping leads to reduction in the redundancy as well as the number of image primitives, often by a few orders of magnitudes - from millions of pixels to a few hundreds of superpixels. There are several existing approaches to superpixel segmentation that use visual features like the RGB color space, or perceptually more uniform color spaces like L*a*b* and L*u*v* [117, 69, 71, 1].

We are interested in oversegmenting a range image such that superpixels comprise of pixels satisfying similar geometric constraints. Range image data is corrupted with *heteroscedastic* or point dependent noise, i.e., the variance of the noise is different at each point. Therefore, we reformulate the superpixel segmentation problem to account for this heteroscedasticity. There exist methods that deal with heteroscedasticity [77, 82] in the context of regression. In this chapter we address the problem of clustering heteroscedastic data and apply it to obtain a superpixel segmentation that respects geometric constraints in an input range image.

The desired properties of a superpixel segmentation algorithm are

- Superpixel boundaries should not cross the true object boundaries.
- A set of superpixels should be able to represent an object of interest.
- The number of superpixels should be as small as possible.

There are several recent methods that address the problem of superpixel segmentation.

Turbopixels [69] is based on a level-set geometric flow algorithm that dilates uniformly placed seeds to form superpixels. The segmentation algorithms mean shift [23] and Quickshift [117] locate modes in the feature space. The superpixels are represented by the modes and comprise of all the pixels in their basins of attraction.

By posing segmentation as a graph optimization problem, graph based methods have also been applied to superpixel segmentation. The N-Cuts algorithm [103] recursively partitions an image based on boundary and texture cues by minimizing an objective function. Felzenswalb and Huttenlocher [34] presented another popular graph based approach, where the image is represented as a graph with the pixels as the nodes. The pixels are then grouped together by agglomerative clustering. A graph cut based method is presented in [83], where the superpixels are forced to conform to a quasi-regular grid. Graph cuts are used to find optimal horizontal and vertical paths that follow visual boundaries. The method proposed in [71] optimizes an objective function on the graph comprising of two terms. The first is an entropy rate term that prefers homogeneity and compactness and the second term biases the algorithm to form similarly sized superpixels.

The Simple Linear Iterative Clustering (SLIC) [1] initializes superpixels with uniformly placed seeds followed by a k-means algorithm applied for pixel association. The algorithm restricts the set of competing seed points within a spatial neighborhood of the pixel, leading to a linear complexity of the algorithm. The distance measure used is a weighted combination of distances in spatial and L*a*b* color space. After the pixel association, the algorithm assigns the remaining unclaimed pixels using connected components.

In this work, we use depth images of indoor scenes. Indoor scenes typically comprise of strongly geometric surfaces. Our goal is to achieve a segmentation of a 3D image into superpixels that satisfy similar geometric constraints. For example, to segment a 3D image into planar superpixels, we locally compute the surface normals at each pixel and use these as the features. Surface normals can be represented by unit vectors in $\mathbb{R}^3$ or as 2D vectors in the (θ, ϕ) space. We use surface normals in $\mathbb{R}^3$ as features for this work.

It is known that accuracy of depth sensors depends on the true depth of the measured point, which means the depth data is heteroscedastic. We address the problem of superpixel segmentation of depth images contaminated with heteroscedastic noise while preserving the underlying geometric properties of the scene, e.g. planarity. Our formulation is generic in nature and can be applied to a variety of stereoscopic depth sensors, however, in this work we limit ourselves to using the Kinect sensor [80]. We use the noise model proposed in [57, 49] and estimate error covariances of surface normals by error propagation. We employ a distance function that uses these covariances to account for the heteroscedasticity in the data.

The remainder of this chapter is organized as follows. We discuss the noise characterization in context of the Kinect sensor in Section 3.1. The planarity preserving heteroscedastic superpixel segmentation is described in Section 3.2 and is applied to real images using the Quickshift algorithm. We conclude in Section 3.3 with a brief discussion and some future directions.

3.1 Sensor Noise Characterization

In this section we discuss the noise characeterization of the depth estimates obtained using the Kinect sensor. Kinect is an active structured light depth sensor that computes the depth estimate of a scene point by using the disparity between corresponding image points. In a generic stereo based system, the depth estimate of a scene point can be computed if both the cameras can observe the scene point. Let C_1 and C_2 be the two cameras in a stereo system and the projection of the ith scene point on to the respective image planes be the image points $\mathbf{p}_i^1$ and $\mathbf{p}_i^2$. Without loss of generality, we can assume that the image points $\tilde{\mathbf{p}}_i^1$ and $\tilde{\mathbf{p}}_i^2$ are the corresponding points in the rectified images. The disparity at point $\mathbf{p}_i$ is then computed as $d_i = \tilde{\mathbf{p}}_i^1 - \tilde{\mathbf{p}}_i^2$. The relationship between disparity and the depth estimate z_i is given as

$$z_i = \frac{f \cdot b}{d_i} \tag{3.1}$$

where both the disparity d_i and the focal length f are in pixels and b is the baseline. The Kinect sensor operates on a similar principle, with one of the cameras replaced by an infra-red (IR) projector. For more details on the depth estimation process, calibration and performance analysis of Kinect, see [57, 19].

3.1.1 The Noise Model

We assume a heteroscedastic noise model, where the depth estimates are corrupted by zero mean Gaussian noise with the pointwise variance as a function of depth.

$$z_i = z_{io} + \mathcal{G}\left(0, \sigma_z(z_{io})\right) \tag{3.2}$$

where $\mathcal{G}(\mu, \sigma)$ is Gaussian noise with mean μ and standard deviation σ. The depth dependent standard deviation is denoted by $\sigma_z(z_{io})$.

In the literature, the inverse relationship between depth and disparity (3.1) is often used to compute the first order approximation of the error in depth. The error in disparity is assumed to have a parametric distribution, e.g. Gaussian or uniform, with a standard deviation σ_d. The expression for the standard deviation σ_z, of the error in depth as given in [57]

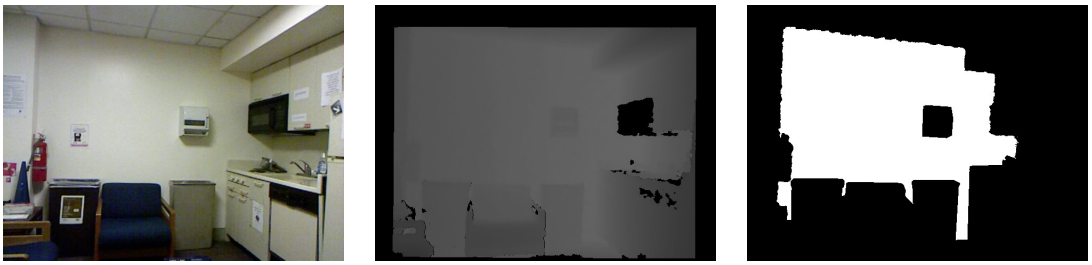
$$\sigma_z(z_{io}) = kz^2 \tag{3.3}$$

where the parameter k is the unknown coefficient of the noise variance function. It can be estimated by analyzing residuals of fitted planar structures in the 3D data from the sensor.

3.1.2 Estimation of Noise Model Parameter

Given a set of 3D images of scenes containing large planar structures at arbitrary depths and orientation, our goal is to estimate the noise model parameter k. We restrict our work to depth images captured using the Kinect sensor. For this purpose, we decided to use the large, publicly available NYU dataset [104].

Fig. 3.1a-b shows a sample pair of RGB and depth images captured using the Kinect



(a) (b) (c)

Figure 3.1: Kinect RGB-D image (NYU Dataset). (a) RGB image. (b) Depth image. (c) A planar segment extracted from the depth image.

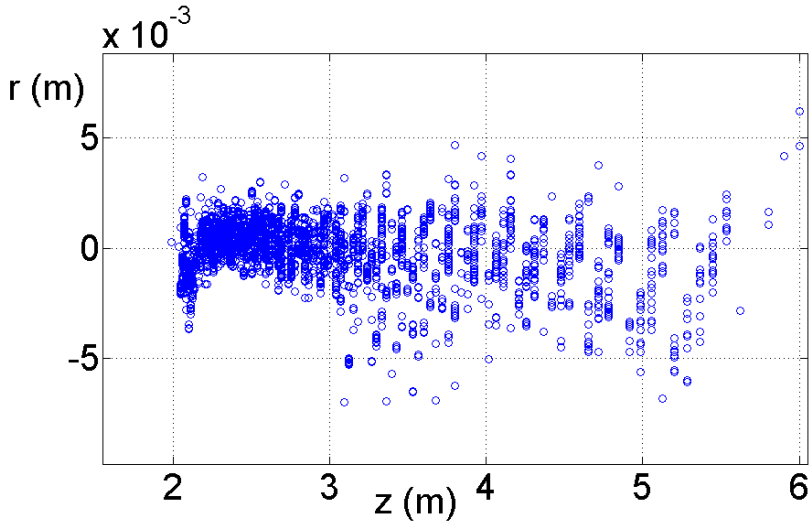


Figure 3.2: RANSAC based plane fitting. Residuals from a RANSAC fitted plane have a small non-zero bias.

sensor. We generate 3D point cloud data from each depth image and robustly detect one large planar region. Fig. 3.1c shows an example of the planar region extracted from the corresponding depth image. From a subset of thirteen different image pairs from the NYU dataset, we extracted one planar segment of arbitrary orientation and depth using off-the-shelf RANSAC implementation [35]. We first decided to use the parameter estimates provided by RANSAC and the corresponding inlier residues to estimate the noise parameter. This approach produced biased results since the RANSAC cost function assumes stationary noise. An example of biased residues using a RANSAC fitted plane can be seen in Fig. 3.2.

Consequently, we take an alternate approach, where we use the depth image pixels lying in the interior of the extracted planar segment. We take a 3×3 region around

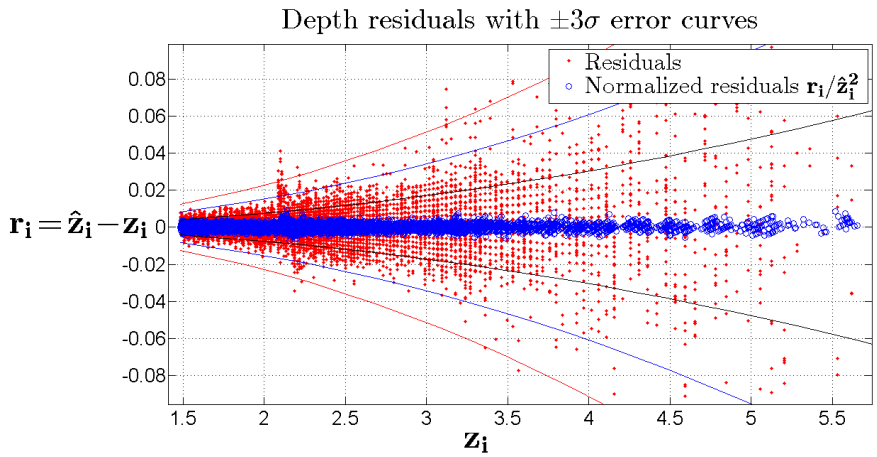


Figure 3.3: Scatter plot of error in depth estimates. The red points are the unnormalized residuals of the depth estimates. Blue points are the residuals normalized by z_i^2. The solid lines are the $\pm 1\sigma, \pm 2\sigma$ and $\pm 3\sigma$ curves.

each such pixel $\mathbf{p}_i$ and compute the mean depth $\hat{z}_i$ over this window. For zero mean noise, the mean depth $\hat{z}_i$ is an unbiased estimator of the true mean z_{io}. However, due to the depth dependent variance, the estimate may deviate from the true value in practice. We first normalize the residuals by $\hat{z}_i^2$ to account for the dependence on depth. The parameter k can then be computed as the standard deviation of the normalized residuals

$$k = \sqrt{\frac{1}{n} \sum_i \left(\frac{r_i}{\hat{z}_i^2} \right)^2}. \tag{3.4}$$

Fig. 3.3 shows the scatter of the residuals $r_i = \hat{z}_i - z_i$ computed from all the planes. The solid lines are the $\pm(1, 2, 3)\sigma$ curves for reference. The blue points are the residuals normalized by $\hat{z}_i^2$.

3.2 Planarity Preserving Heteroscedastic Superpixel Segmentation

We use the noise characterization developed in the previous section and apply it to perform planarity preserving superpixel segmentation of depth images. We operate in the space of surface normals computed locally for each depth pixel. We represent each surface normal as a unit vector in $\mathbb{R}^3$. Since the noise is heteroscedastic, we need to first

compute the covariances of these surface normals to define a statistically meaningful distance function for segmentation.

3.2.1 Normal Computation and Covariance Estimation

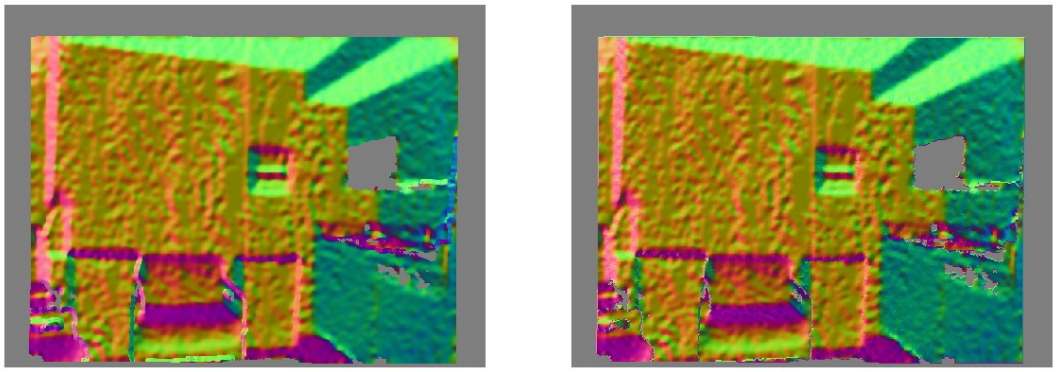
Using the intrinsic camera parameters of the depth sensor, we generate a 3D point cloud where each point corresponds to a pixel in the depth image. For an image pixel $\mathbf{q}_i = [x_i, y_i]^\top$, its corresponding point in 3D is computed as

$$\mathbf{p}_i = \mathbf{p}(x_i, y_i) = \frac{z_i}{f}[x_i - c_x \quad y_i - c_y \quad 1]^\top \tag{3.5}$$

where z_i is the depth of the ith pixel and x_i and y_i are the image coordinates of the ith pixel. The principal point of the camera are denoted by $[c_x \;\; c_y]$.

To compute the surface normals, we take a 9×9 window around each pixel $\mathbf{q}_i$ in the depth image. This amounts to a surface patch around the corresponding 3D point in the scene. The singular value decomposition (SVD) of this surface patch gives the eigenvectors of the covariance matrix of the points. The singular vector corresponding to the smallest singular value gives the surface normal of the patch. A neighborhood size that monotonically increases with depth could also be used effecting more smoothing at larger depths. Let $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_i, \ldots, \mathbf{p}_n]$ be the $3 \times n$ matrix of the 3D world points around $\mathbf{p}_i$. Then the total least squares (TLS) estimate of the surface normal $\widehat{\mathbf{n}}$ is computed as $\widehat{\mathbf{n}} = \mathbf{u}_3$, where the orthogonal matrix $\mathbf{U} = [\mathbf{u}_1 \; \mathbf{u}_2 \; \mathbf{u}_3]$ is obtained by factorizing the centered $\mathbf{P}$ into $\mathbf{USV}^\top$ using SVD.

Another approach of computing the surface normals is to compute the gradient vector of the surface patch. We take the cross product of two linearly independent three dimensional tangent vectors computed on the surface at the 3D world point $\mathbf{p}_i$. We approximate the tangent vectors by applying the difference operator on the image point $\mathbf{p}_i$ along x and y, i.e., the horizontal and vertical axes of the image plane respectively. The 3D normal vector $\widehat{\mathbf{n}}$ is the unit norm cross product of the tangent vectors, or equivalently the direction of the gradient vector at $\mathbf{p}_i$ and can be computed



(a) (b)

Figure 3.4: Normal Image. (a) Normals computed using SVD. (b) Normals computed using the gradient direction.

as

$$\begin{aligned}
\mathbf{T}_x(x_i, y_i) &= \frac{\partial \mathbf{p}_i}{\partial x} \approx \mathbf{p}(x_i{+}1, y_i) - \mathbf{p}(x_i{-}1, y_i) \\
\mathbf{T}_y(x_i, y_i) &= \frac{\partial \mathbf{p}}{\partial y} \approx \mathbf{p}(x_i, y_i{+}1) - \mathbf{p}(x_i, y_i{-}1) \\
\widehat{\mathbf{n}}(x_i, y_i) &= \frac{\mathbf{T}_y(x_i, y_i) \times \mathbf{T}_x(x_i, y_i)}{\|\mathbf{T}_y(x_i, y_i) \times \mathbf{T}_x(x_i, y_i)\|}
\end{aligned} \tag{3.6}$$

where x_i and y_i are the image co-ordinates of the ith scene point. In order to avoid noisy normals and tangents, it is necessary to smooth the image prior to applying the difference operator. This is done by smoothing the depth image using a bilateral filter [112] with a 9×9 window and the spatial bandwidth $\sigma_s = 2$ and the depth bandwidth $\sigma_d = 0.1$. Fig. 3.4a shows the color coded normal image computed using the SVD method, while Fig. 3.4b shows the color coded normal image using the differentiation method.

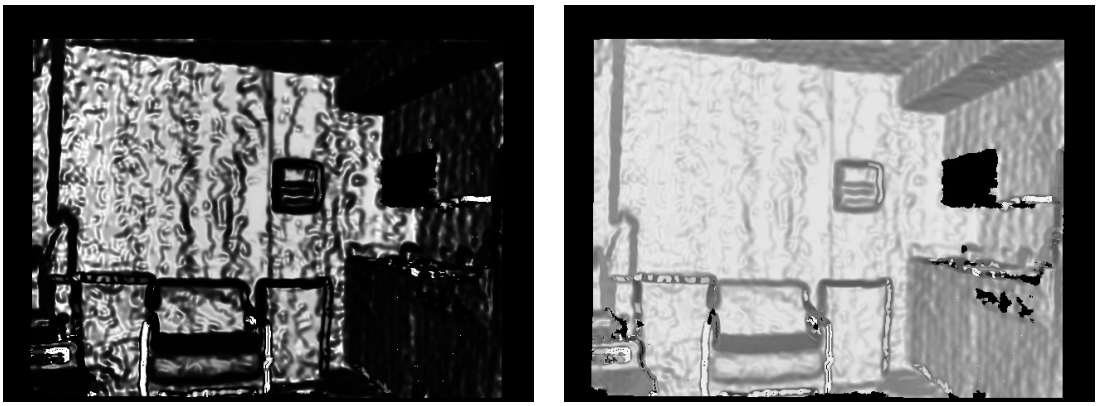
From Fig. 3.1a and Fig. 3.4, we see that in thin planar regions, the SVD method captures the surface orientation better than the cross product method. A detailed analysis of methods for surface normal computation in [58] also indicates that the SVD based approach is favorable when a kNN adjacency graph of the point cloud is available. In our case, the depth image implicitly captures the adjacency in the point cloud, thus we use the SVD based normal images as our feature representation to do the planarity

preserving superpixel segmentation. In order to account for the heteroscedasticity in the data, we compute the covariances of the surface normals. In our analysis, we use empirically computed covariances.

3.2.2 Empirical Covariances

We compute the normal covariances empirically by perturbing the depth estimates z, using the estimated noise standard deviation function (3.3). We parametrize a given planar surface by the orientation of its surface normal: θ, the azimuth angle and ϕ, the elevation. These perturbed depth values are used to generate 3D points at different image locations (u, v) along the plane specified by θ and ϕ. We populate a 5-dimensional look up table with each dimension corresponding to each of the variables in the set $\{\theta, \phi, u, v, z\}$. Given the plane parameters (θ, ϕ), the pixel location (u, v) on the image plane, and the depth value z, we compute the depth values of the pixels in the 9×9 neighborhood of (u, v). We perturb these depth values using the standard deviation (3.3), generate the corresponding noisy 3D points and compute the plane normal $\widehat{\mathbf{n}}$ using SVD. We repeat the normal computation for a thousand perturbations of the depth and compute the empirical covariance to populate the look up table.

For a query depth image, we compute the normal image using the SVD method described in Section 3.2.1. To look up the covariance of the surface normal at a pixel location, we use the 5-dimensional vector $[\theta, \phi, u, v, z]$ to represent the query point. We search for the cell containing the query point in the look up table and retrieve all the 32 nearest neighbors (2^5 for 5-dimensions). We compute the covariance of the query point by multilinear interpolation of the covariances of the neighboring points. In order to maintain the positive semi-definiteness of the interpolated covariance matrix, we find the axis of rotation between the covariance matrices of any two neighboring points and linearly interpolate the angle of rotation. Fig. 3.5 shows a covariance map of the normal image shown in Fig. 3.4a, computed using the look up table. The intensity represents the Frobenius norm and the log of the Frobenius norm of the covariance of the surface normals in Fig. 3.5a and Fig. 3.5b respectively.



(a) (b)

Figure 3.5: Empirical covariance map. (a) Frobenius norm of the covariances of surface normals. (b) The log of Frobenius norm of the covariances of surface normals

3.2.3 Heteroscedastic Superpixel Segmentation

We develop a superpixel segmentation algorithm that accounts for the underlying heteroscedasticity in the data by modifying Quickshift [117] to use the Mahalanobis distances for density computation. The modified Quickshift algorithm is applied to the normal images computed using the SVD method. For computing the pointwise covariances, we use the look up table and interpolation as described in Section 3.2.1.

Heteroscedastic Distance Computation

In Quickshift, the kernel density estimate at each pixel is computed using the neighboring distances in the feature space. Let $\mathbf{x}$ be a point in the feature space corresponding to the image point $\mathbf{p}(u_i, v_i)$. The heteroscedastic kernel density estimate at $\mathbf{x}$ for the kernel function $K(\cdot)$ is computed as

$$f_{\mathbf{\Sigma}}(\mathbf{x}) = \frac{1}{n}\sum_{i}^{n} \frac{1}{\det \mathbf{\Sigma}_i} K\left(d_{\mathbf{\Sigma}}(\mathbf{x}, \mathbf{x}_i)\right) \tag{3.7}$$

$$d_{\mathbf{\Sigma}}(\mathbf{x}, \mathbf{x}_i) = \left((\mathbf{x} - \mathbf{x}_i)^\top \mathbf{\Sigma}_i^+ (\mathbf{x} - \mathbf{x}_i)\right)^{1/2}$$

where the heteroscedastic distance between the point $\mathbf{x}$ and a neighboring point $\mathbf{x}_i$ is given by d_Σ. The Quickshift algorithm locates the local mode by iteratively selecting a neighboring point that yields the maximum increase in the kernel density estimate as

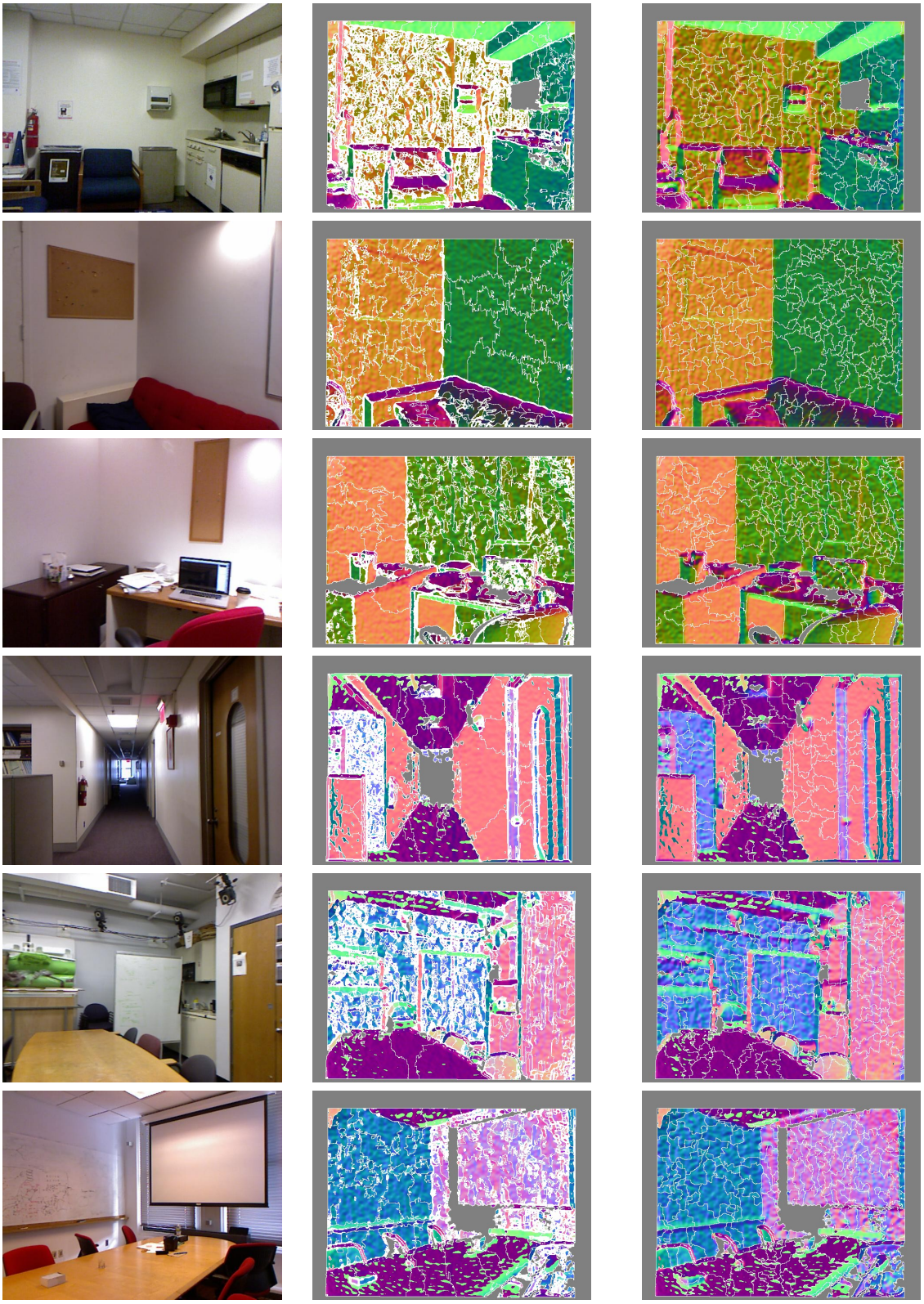
an estimate of the mode. The group of pixels that converge to the same mode form a superpixel.

Heteroscedastic vs. Euclidean Quickshift

We performed experiments to evaluate the performance of the heteroscedastic superpixel segmentation method to achieve a planarity preserving oversegmentation of the images. We compare the performance with the baseline algorithm – the Euclidean Quickshift. For the comparison, we used a subset of the NYU dataset and computed the normal images using the SVD method. Both the superpixel segmentation algorithms were evaluated on the same normal images for a fair comparison.

Qualitative Comparison Fig. 3.6 shows some qualitative results of heteroscedastic Quickshift compared to the baseline algorithm for a subset of the NYU dataset. The white curves in the images are the segment boundaries. The spatial σ_s and the range bandwidth σ_r are specified. It can be seen that in case of Euclidean Quickshift generates segments that are very small in noisier regions, while the heteroscedastic segmentation avoids such artifacts.

Quantitative Comparison The ground truth annotation of the NYU dataset are based on object categories and not on planar segments. Therefore, for quantitative evaluation, we captured a few indoor images using the Kinect sensor and annotated three images by marking the planar segments in each image. These were used to generate Precision-Recall curves [115] for a quantitative comparison of the superpixel segmentation algorithms. Fig. 3.8a shows the images and the annotated planar boundaries in the dataset. The two methods were run for different parameters—the spatial bandwidth varying as $\sigma_s \in \{5, 10, 15\}$ and range bandwidth varying in $\sigma_r \in \{10, 15, 20\}$. The composite precision recall plots for all three images are shown in Fig.3.7, when the two methods were run for all the nine combinations of the parameters. The green curves in the plots correspond to F-score isocontours. Note that the red markers corresponding to the heteroscedastic Quickshift typically have higher F-score. The sample qualitative results and the individual precision - recall curves are shown in Fig. 3.8b.



Original Image | Euclidean($\sigma_r = 15, \sigma_s = 10$) | Mahalanobis($\sigma_r = 15, \sigma_s = 10$)

Figure 3.6: Qualitative comparison of Euclidean vs. Heteroscedastic Quickshift. The poor precision in the Euclidean case is clear.

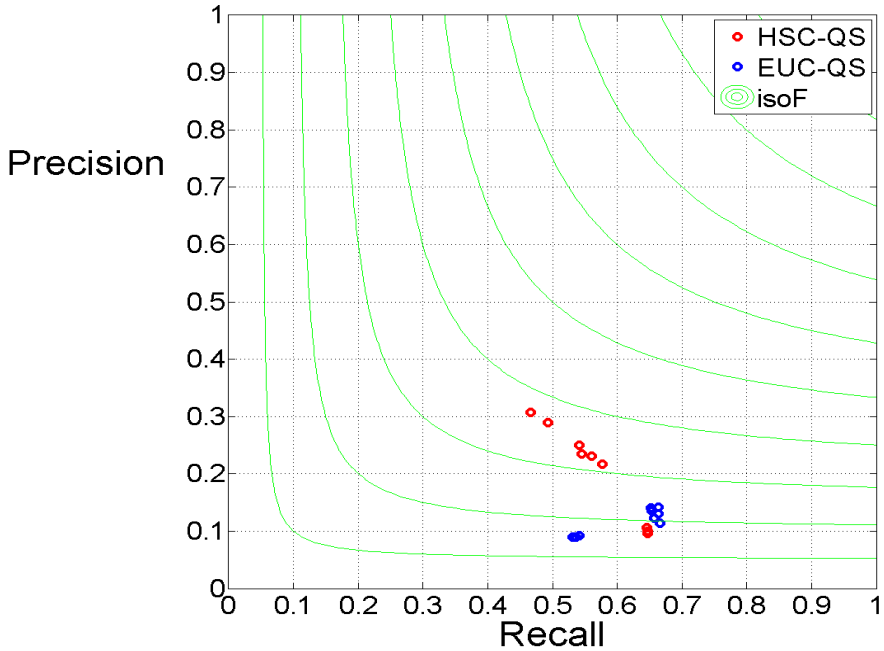


Figure 3.7: Precision - Recall curves for annotated planar dataset.

3.3 Discussion

We presented a method of estimating the noise parameters for a depth dependent noise model for the Kinect sensor. We developed a method for estimating these parameters without explicitly calibrating the sensor, but by using robustly detected planar regions from previously collected data. The noise parameters were used to compute covariances of surface normals. We also developed a heteroscedastic superpixel segmentation algorithm based on Quickshift that uses surface normals and their covariances to perform planarity preserving superpixel segmentation. We compared our proposed heteroscedastic method with baseline Quickshift and show favorable qualitative and quantitative results in terms of the F-score.

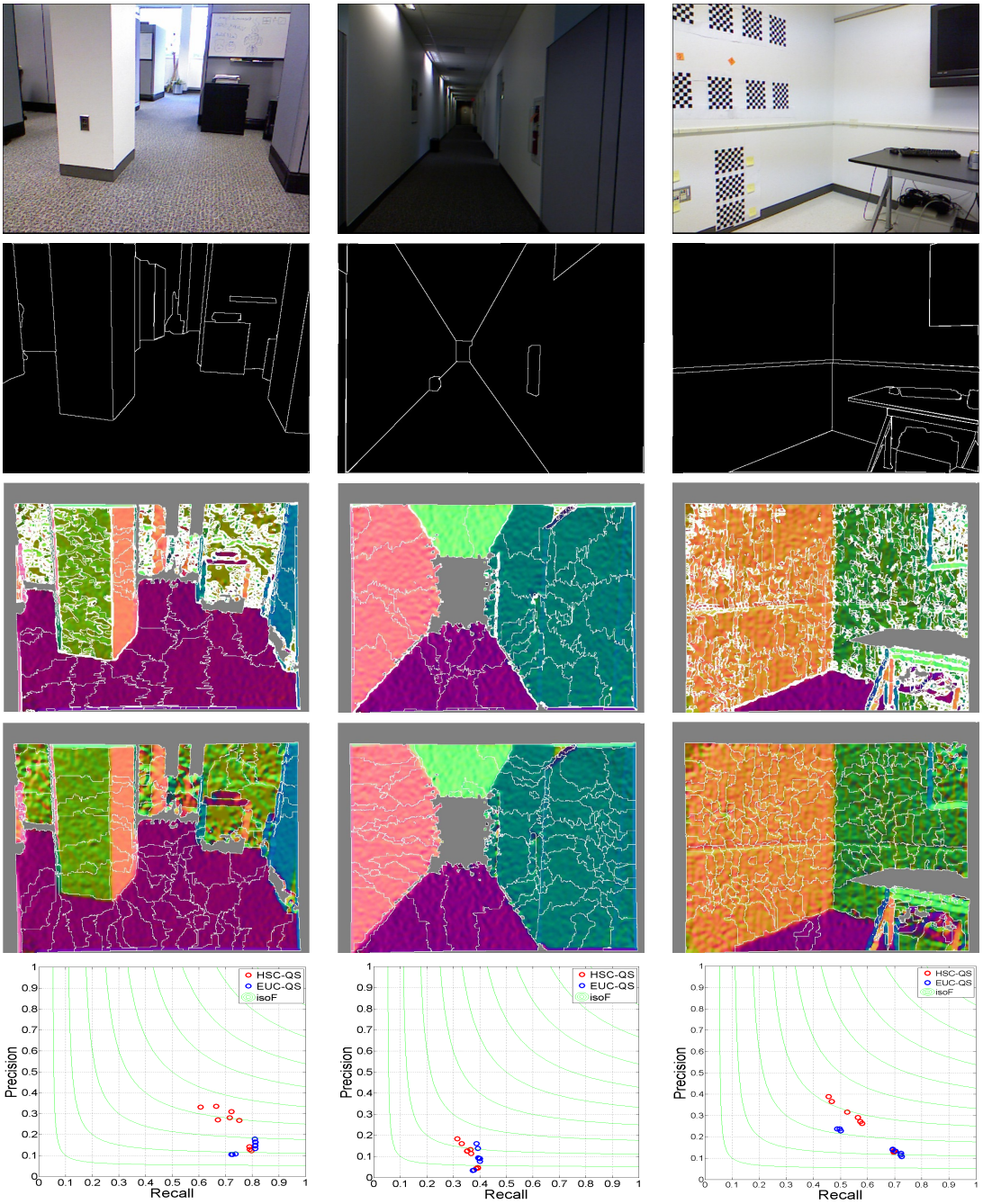


Figure 3.8: Planar dataset. Row 1: RGB images captured from Kinect. Row 2: Annotated ground truth planar boundaries. Row 3: Euclidean Quickshift sample result. Row 4: Heteroscedastic Quickshift sample result. Row 5: Precision-Recall scatter plot for 9 different settings (red) - HSC-QS (blue) - EUC-QS (green)- isocontours of F-score.

Chapter 4

Indoor Scene Modeling Using a Single RGB-D Image

Modeling of indoor spatial environments serves as an important module in several systems, where the model could facilitate operations like assisted navigation for humans, autonomous navigation of robots. Search and rescue operations could leverage from pre-built models by automatically searching for alternate routes in case of emergencies. With recent advances in vision technology for sensing 3D data, the interest and motivation to develop fast, automatic and reliable indoor modeling systems has increased manifold.

In this chapter, we present a framework for building a 3D planar model of local environments using depth augmented RGB data. This data may be captured from a variety of sources: off-the-shelf depth sensors; camera systems that compute depth estimates online (e.g. stereo cameras); structure from motion etc. The system accepts data in the form of rasterable 3D point clouds and processes it to produce a 3D model comprising planar primitives.

Spatial modeling of indoor environments has drawn a lot of interest. Using 2D images, the modeling of indoor scenes is done under the Manhattan assumption, where three orthogonal planes are estimated by searching for the three vanishing points [67, 47, 48, 44, 89, 100]. This is followed by the calibration of the camera and generation of a model by identifying the horizontal and vertical planes corresponding to the walls of the room. Similarly indoor objects were also modeled by identifying bounding boxes containing the objects in the scene [44, 66, 89]. The idea of using planar segments to model outdoor scenes has also been explored [59, 40]. Due to inherent difficulties in 3D structure estimation from single 2D images, these approaches have been somewhat restrictive in the kind of problems they tackle and are unlikely to scale to general indoor

scenes or large models.

Some of these estimation issues can be addressed by directly working on 3D data. Thus, recent advances in availability of sensor systems acquiring and producing 3D data: laser scanners, Microsoft Kinect, dense stereo etc. has predictably generated a lot of interest in solving the scene modeling problem directly using 3D data as input [99, 64, 14, 98]. The work described in [116, 40] uses planar regions to model man-made environments and also to perform semantic analysis of 3D point clouds [60, 108]. However, none of these approaches is fast and scalable.

In this chapter, we address the challenge of scalable 3D modeling of indoor environments using a single frame from a calibrated depth sensor. We make the following contributions:

1. We propose a multi-scale representation that models relevant scene information at several levels: at the lowest level, it models 3D sensory information that is input to the system while at the highest level, it models the geometric (planar) structures in the 3D scene.

2. We propose a scalable system that utilizes characterization of the sensor noise and performs a series of statistical tests to prune information at each level of the hierarchy.

3. We show the plane discovery results on several realistic datasets with differing level of scene complexity.

The rest of the chapter is organized as follows. We introduce the problem under consideration and motivate the proposed approach in Section 4.1. We provide the details of each module in Section 4.2 and evaluate our proposed approach on realistic datasets in Section 4.3. Finally, we conclude with a discussion in Section 4.4.

4.1 System Overview

Understanding of visual environments is equivalent to faithfully describing a scene in terms of its constituent semantic entities as opposed to a low-level representation of

data in terms of point clouds or mesh maps. The notion of semantics is hierarchical in nature, for example, from parts to instances to categories of objects. In a bottom-up sense, following point clouds or mesh models, representing the topology using geometric entities takes the next level in the hierarchy of semantic representation. From a human-centric perspective, such a representation is intuitive, since most man-made environments comprise largely of entities like floor, wall, ceiling, tables, chairs, etc., which can be reliably described using a combination of simple geometric structures. The larger context of this work is the understanding of visual environments from point cloud data.

In this work, we take an important step towards this goal by solving a subset of the problem of scene understanding. Given a range image of an indoor scene comprising largely of planar regions, we create a 3D model of the scene using piecewise structural planar units.

Building a parametrized geometric 3D model from point cloud data is a challenging problem and can be fairly ill-posed. Indoor scenes are typically cluttered with objects of various shapes and sizes. A representation that permits arbitrarily small planar regions would explain the data perfectly, e.g., a mesh model. However, to achieve model parsimony, we need a representation of the scene using a minimal set of parametrized planar segments. The size of this set could vary significantly depending on the complexity of the scene. The clutter in indoor scenes interferes with the estimation of planar segments and should be identified as nonplanar regions or outliers. Moreover, data from depth sensors is corrupted with non-stationary, depth dependent noise and should be accounted for in the estimation process. A combination of these issues make the design of such a geometric model building system challenging.

For a practical system that addresses the aforementioned challenges, we propose an approach based on the following design principles.

Scalability– The system should be able to handle large point clouds.

Modularity– The system should be modular for ease of extension to other geometric entities.

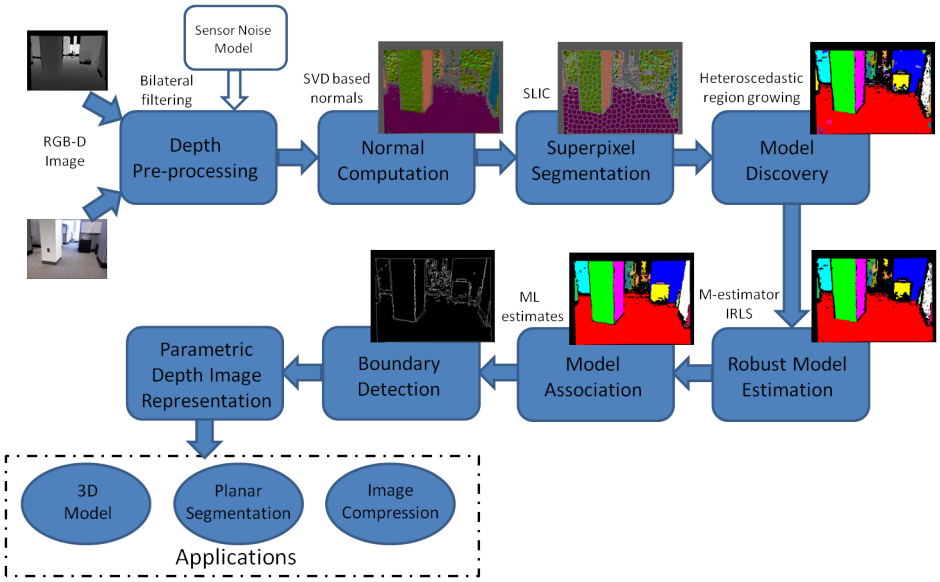


Figure 4.1: The system block diagram.

Analyzable– The behavior of each module should be quantifiable for statistical analysis.

4.2 Planar Model Discovery

The input to the model building system is a synchronized RGB and depth image pair from Microsoft Kinect and the noise model parameter which is estimated beforehand as described in Section 3.1. We assume that the depth sensor is calibrated and the 3D location corresponding to each pixel can be computed using the intrinsic parameters. The goal of model discovery is to identify the minimal set of planar models that can explain the scene. Here, we assume that each planar model represents a single planar segment. If two models have the same planar parameters, they have to be disconnected. In this section we describe our bottom-up approach for discovering multiple planar models.

4.2.1 Normal Computation

We compute the TLS based estimate of the surface normals at each pixel using the SVD method as described in Section 3.2.1. Since we do not know the extent of planar

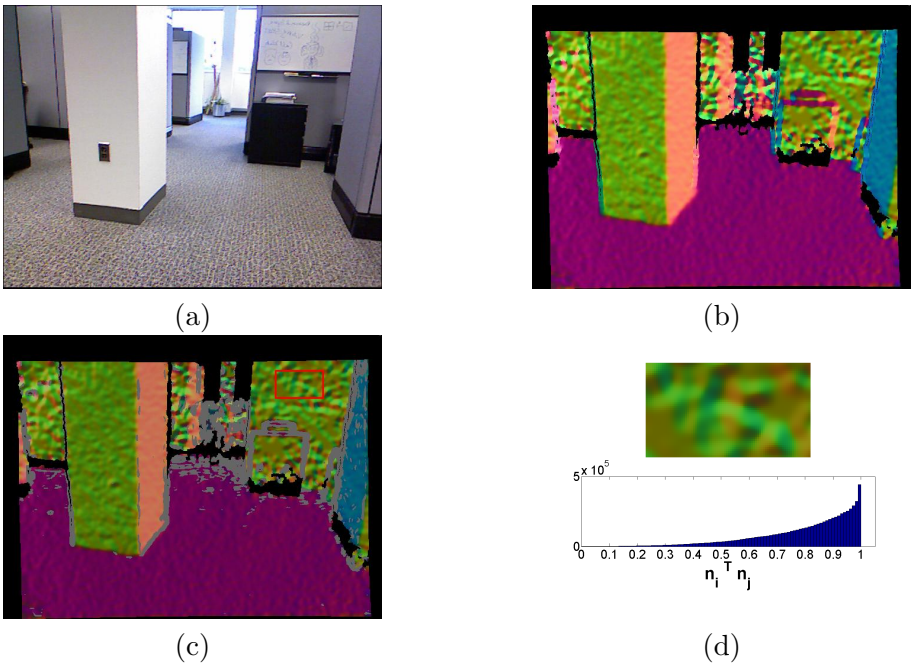


Figure 4.2: Color coded surface normals computed using SVD. (a) RGB image. (b) Surface normals. (c) Normals with high planar probability with a χ^2 significance level of 0.05. Gray regions indicate pixels that fail the test, while black regions indicate missing depth data. (d) Top: inset showing the region shown in the red box. Bottom: Histogram of dot products of the surface normals.

surfaces a priori, the computed normals are not reliable at depth discontinuities and in nonplanar regions. We detect such regions by applying a mask over pixels that have a high probability of lying on a plane and satisfy the χ^2 test at a significance level of 0.05. Fig. 4.2b shows the color coded normal image computed using the SVD method. The black labeled pixels show the missing depth data region. In Fig. 4.2c, the masked normal image is shown, where the grey regions indicate the pixels that failed the χ^2 significance test. It can be observed that many pixels at depth discontinuities, nonplanar regions and pixels at large depths are removed. Note that the process of computing and validating the surface normals is performed locally and therefore is highly parallelizable.

Since we compute the surface normals at each pixel, a large number of hypotheses is obtained for each planar structure present in the scene. However, these surface normals are computed over small spatial neighborhoods and are affected by the noise in the depth data resulting in significant biases in the locally estimated surface normals.

This can be observed in Fig. 4.2d, where the top image shows the close-up of the red rectangular patch in Fig. 4.2c. We computed the dot products between all pairs of surface normals corresponding to the pixels of this patch and their histogram is plotted in Fig. 4.2d. Despite lying on the same planar surface, the histogram shows that the cosine of the angle between the surface normals could be less than 0.5. To reduce the number of potential hypotheses and to alleviate the effect of biased local estimates, we leverage from the image structure and perform superpixel segmentation in the space of surface normals.

4.2.2 Model Discovery

We solve the model discovery problem in a hierarchical framework. At the lowest level, we have a large number of local planar hypotheses as the surface normals at each pixel. Next, we identify superpixels as a smaller set of planar primitives that form an overcomplete basis for the planar structures in the scene. We then robustly merge together the superpixels in a region growing framework to identify a minimal set of planar primitives.

Superpixels: As discussed above, a single frame contains pixels in the order of hundreds of thousands and the locally estimated normals can have a significant bias. We generate superpixel planar primitives by oversegmenting the surface normal image. These superpixels form a set of primitives at the lowest level of our modeling hierarchy. Additionally, they provide adjacency relationships, which is useful in representing the planar primitives in a graphical structure and preserves the topology information of the scene.

We assume that an *inlier* superpixel is homogeneous, i.e., all the pixels comprising the superpixel uniquely belong to a single planar segement and adequately satisfy the corresponding planar constraint. We expect a majority of superpixels to be *inliers* and if there are no *inlier* superpixels on a planar segment, then the planar region cannot be discovered. The size of the superpixels determines the smallest sized planar region that can be detected. We use Simple Linear Iterative Clustering (SLIC) [1] for superpixel segmentation of the surface normal image. This approach performs a restricted k-means

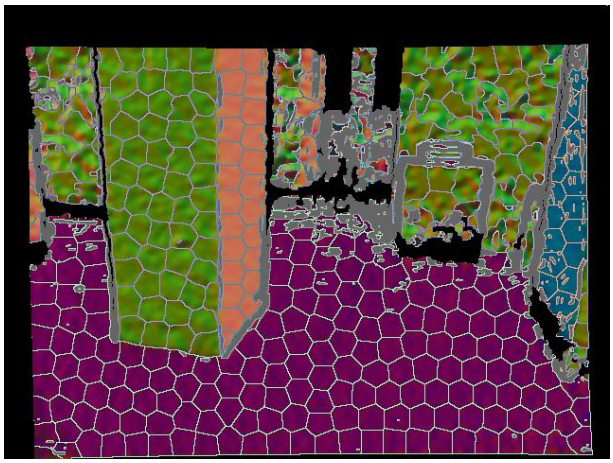


Figure 4.3: Superpixel segmentation using SLIC [1].

clustering using the following weighted distance function

$$d_{SLIC} = d_f + w d_{sp} \tag{4.1}$$

The distance d_f is computed in the space of surface normals $\mathbb{R}^3$ and the spatial distance d_{sp} is computed in the image plane $\mathbb{R}^2$. The scalar weighting parameter w determines the trade-off between compactness of superpixels and adherence to feature boundaries. SLIC also requires the approximate size of the superpixels as input. Since we use Microsoft Kinect images which have a VGA resolution of 640×480, we fix the values of $w = 0.02$ and the superpixel size as 25 in all our experiments. SLIC has a computational complexity that is linear in the number of pixels. Due to these advantages, we chose this method over other approaches like quickshift [117] or Turbopixels [69]. In our experiments, we found that with SLIC, the heteroscedastic superpixel segmentation did not yield significantly different results. Therefore, for computational reasons, we use Euclidean SLIC for superpixel computation. Fig. 4.3 shows the superpixel boundaries overlaid on the masked normal image.

The fraction of inlier superpixels that obey the homogeneity property is significantly increased by considering only the pixels that pass the χ^2 test (significance level 0.05) for planarity during the normal computation step. It is important to note that we are not assuming a perfect superpixel segmentation. We employ robust methods in the

next step to deal with *outliers*, i.e., superpixels which may contain pixels from more than one planar region.

Fragments: The oversegmentation of the surface normal image generates a reduced set of planar primitives of the order of a few hundreds. These superpixels represent small regions of the scene and are still an overcomplete set of planar model hypotheses. Based on the adjacency of these superpixels, we generate an undirected graph $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$ where each node is a superpixel and each of its neighbors (adjacent superpixels) are connected through an edge. Each node $S_i \in \mathcal{V}_S, i = 1, \ldots, |\mathcal{V}_S|$, is characterized by the following four parameters.

- The centroid $\widehat{\mathbf{p}}_i$ of the n_i 3D points corresponding to the constituent pixels.

- The depth dependent covariance of the centroid is computed using error propagation $\mathbf{C}_{\widehat{\mathbf{p}}_i}$

$$\mathbf{C}_{\widehat{\mathbf{p}}_i} = \mathcal{J}_{\widehat{\mathbf{p}}_i|z}^{\top} \sigma_z^2 \mathcal{J}_{\widehat{\mathbf{p}}_i|z} = \frac{1}{\widehat{z}_i^2}\ \widehat{\mathbf{p}}_i \sigma_{\widehat{z}_i}^2\ \widehat{\mathbf{p}}_i^{\top} \tag{4.2}$$

  where $\mathcal{J}_{\widehat{\mathbf{p}}_i|z}$ is the Jacobian of $\widehat{\mathbf{p}}_i$ with respect to the depth z, $\widehat{z}_i$ is the depth at $\widehat{\mathbf{p}}_i$ and $\sigma_{\widehat{z}_i}^2$ is the depth variance computed using (3.3). A more accurate estimate of the covariance would be as described in [15], but we use this estimate as an approximation for computational speed.

- The total least squares (TLS) estimate $\widehat{\mathbf{n}}_i$ of the surface normal computed using the n_i 3D points.

- The covariance of the surface normals $\mathbf{C}_{\widehat{\mathbf{n}}_i}$, which are in general different for each superpixel

$$\mathbf{C}_{\widehat{\mathbf{n}}_i} = \frac{1}{n_i} \sum_{j=1}^{n_i} (\widehat{\mathbf{n}}_{ij} - \widehat{\mathbf{n}}_i)(\widehat{\mathbf{n}}_{ij} - \widehat{\mathbf{n}}_i)^{\top} \tag{4.3}$$

  where $\widehat{\mathbf{n}}_{ij}, \quad j = 1, \ldots, n_i$ are the normals computed at each pixel as described in Section 4.2.1.

Since these parameters are computed by centering the 3D patch corresponding to the superpixel, the normal $\widehat{\mathbf{n}}_i$ and the centroid $\widehat{\mathbf{p}}_i$ are independent.

To further reduce the set of models, we want to group together superpixels that lie in the same planar region. Denoting the ith superpixel by $S_i = \{\widehat{\mathbf{p}}_i, \mathbf{C}_{\widehat{\mathbf{p}}_i}, \widehat{\mathbf{n}}_i, \mathbf{C}_{\widehat{\mathbf{n}}_i}\}$, we define the following squared symmetric planar distance function between two superpixels

$$d_{ij} = d(S_i, S_j) = \left(\widehat{\mathbf{n}}_i^\top \Delta\widehat{\mathbf{p}}_{ij}\right)^2 + \left(\widehat{\mathbf{n}}_j^\top \Delta\widehat{\mathbf{p}}_{ij}\right)^2 \tag{4.4}$$

where we use $\Delta\widehat{\mathbf{p}}_{ij} = \widehat{\mathbf{p}}_i - \widehat{\mathbf{p}}_j$ for convenience. Since the centroids $\widehat{\mathbf{p}}_i$ and $\widehat{\mathbf{p}}_j$ are independent, $\text{cov}(\Delta\widehat{\mathbf{p}}_{ij}) = \mathbf{C}_{\Delta\widehat{\mathbf{p}}_{ij}} = \mathbf{C}_{\widehat{\mathbf{p}}_i} + \mathbf{C}_{\widehat{\mathbf{p}}_j}$. This distance function (4.4) is small if and only if both the superpixels lie on the same planar surface and they have similar normals. Note that this distance function is not a metric as it may violate the triangular inequality. Recall that each superpixel has a different noise covariance, both for the surface normal and the centroid. Therefore, we derive a *heteroscedastic* form of this distance function that accounts for the point dependent covariances.

Differentiating the distance function w.r.t. the independent variables, we compute the Jacobians as

$$\begin{aligned}
\mathcal{J}_{d_{ij}|\widehat{\mathbf{n}}_i} &= 2(\widehat{\mathbf{n}}_i^\top \Delta\widehat{\mathbf{p}}_{ij})\Delta\widehat{\mathbf{p}}_{ij} \\
\mathcal{J}_{d_{ij}|\widehat{\mathbf{n}}_j} &= 2(\widehat{\mathbf{n}}_j^\top \Delta\widehat{\mathbf{p}}_{ij})\Delta\widehat{\mathbf{p}}_{ij} \\
\mathcal{J}_{d_{ij}|\Delta\widehat{\mathbf{p}}_{ij}} &= 2(\widehat{\mathbf{n}}_i^\top \Delta\widehat{\mathbf{p}}_{ij})\widehat{\mathbf{n}}_i + 2(\widehat{\mathbf{n}}_j^\top \Delta\widehat{\mathbf{p}}_{ij})\widehat{\mathbf{n}}_j
\end{aligned} \tag{4.5}$$

We compute the variance of the distance function (4.4) by error propagation

$$\begin{aligned}
\sigma^2_{d_{ij}|\widehat{\mathbf{n}}_i} &= \mathcal{J}^\top_{d_{ij}|\widehat{\mathbf{n}}_i} \mathbf{C}_{\widehat{\mathbf{n}}_i} \mathcal{J}_{d_{ij}|\widehat{\mathbf{n}}_i} \\
\sigma^2_{d_{ij}|\widehat{\mathbf{n}}_j} &= \mathcal{J}^\top_{d_{ij}|\widehat{\mathbf{n}}_j} \mathbf{C}_{\widehat{\mathbf{n}}_j} \mathcal{J}_{d_{ij}|\widehat{\mathbf{n}}_j} \\
\sigma^2_{d_{ij}|\Delta\widehat{\mathbf{p}}_{ij}} &= \mathcal{J}^\top_{d_{ij}|\Delta\widehat{\mathbf{p}}_{ij}} \mathbf{C}_{\Delta\widehat{\mathbf{p}}_{ij}} \mathcal{J}_{d_{ij}|\Delta\widehat{\mathbf{p}}_{ij}} \\
\sigma^2_{d_{ij}} &= \sigma^2_{d_{ij}|\widehat{\mathbf{n}}_i} + \sigma^2_{d_{ij}|\widehat{\mathbf{n}}_j} + \sigma^2_{d_{ij}|\Delta\widehat{\mathbf{p}}_{ij}}.
\end{aligned} \tag{4.6}$$

Substituting from (4.5), we get the expression of the variance of d_{ij} as

$$\begin{aligned}\sigma_{d_{ij}}^2 &= (2\widehat{\mathbf{n}}_i^\top \Delta\widehat{\mathbf{p}}_{ij})^2 \left(\Delta\widehat{\mathbf{p}}_{ij}^\top \mathbf{C}_{\widehat{\mathbf{n}}_i} \Delta\widehat{\mathbf{p}}_{ij} + \widehat{\mathbf{n}}_i \mathbf{C}_{\Delta\widehat{\mathbf{p}}_{ij}} \widehat{\mathbf{n}}_i\right) \\ &+ (2\widehat{\mathbf{n}}_j^\top \Delta\widehat{\mathbf{p}}_{ij})^2 \left(\Delta\widehat{\mathbf{p}}_{ij}^\top \mathbf{C}_{\widehat{\mathbf{n}}_j} \Delta\widehat{\mathbf{p}}_{ij} + \widehat{\mathbf{n}}_j \mathbf{C}_{\Delta\widehat{\mathbf{p}}_{ij}} \widehat{\mathbf{n}}_j\right) \\ &+ 8(\widehat{\mathbf{n}}_i^\top \Delta\widehat{\mathbf{p}}_{ij})(\widehat{\mathbf{n}}_j^\top \Delta\widehat{\mathbf{p}}_{ij})\widehat{\mathbf{n}}_i^\top \mathbf{C}_{\Delta\widehat{\mathbf{p}}_{ij}} \widehat{\mathbf{n}}_j. \end{aligned} \tag{4.7}$$

We define the heteroscedastic planar distance function between superpixels S_i and S_j

$$d_{\mathcal{H}}(S_i, S_j) = \frac{d_{ij}}{\sigma_{d_{ij}}}. \tag{4.8}$$

Note that $\sigma_{d_{ij}}$ is the standard deviation of the *squared* symmetric planar distance (4.4). The heteroscedastic distance function (4.8) accounts for the measurement noise and makes the distance function less sensitive to the depth of the centroids and the orientation of the surface normals of the superpixels.

For grouping the superpixels, performing clustering seems to be a plausible approach. However, most off-the-shelf clustering methods are inadequate to operate with superpixels and the distance function we developed. Variants of algorithms like mean shift clustering [23] and k-means [54] assume the underlying distance function to be a metric. Additionally, these clustering algorithms operate in a feature space and therefore offer little control over imposing spatial contiguity between cluster members. Graph based methods like spectral clustering [86, 103] or graph cuts [10] can impose spatial contiguity, but need the knowledge of the number of clusters *a priori*. Variants of these methods that attempt to estimate the number of clusters automatically are usually sensitive to the selected parameters. For example, spectral clustering is sensitive to the eigenvalue threshold in the presence of noise [75]. When a label cost term is added to the energy function for graph cuts [27, 52], the choice of the relative weights is crucial for acceptable performance.

We develop a region growing algorithm on the superpixel graph to identify the *unknown* number of regions, R. The algorithm takes a distance threshold τ, which we fix at 0.5 for all experiments, and the superpixel graph $\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$ as input.

The output is the set of planar regions comprising the scene and is represented by the set $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_R\}$. Here $\mathcal{I}_r,\ r = 1, \ldots, R$ represents the rth planar region as the set of its constituent superpixels. Using the graph $\mathcal{G}_S$, we define the neighborhood of a superpixel $S_i \in \mathcal{V}_S$ as $\mathcal{N}(S_i) = \{S_j \in \mathcal{V}_S | (S_i, S_j) \in \mathcal{E}_S\}$. In the context of the region growing algorithm, we assume that distance refers to the heteroscedastic planar distance function $d_{\mathcal{H}}(S_i, S_j)$ (4.8) between superpixels.

We maintain a priority queue $\mathcal{Q}_r$ of candidate superpixels that can be added to the region $\mathcal{I}_r$. To impose spatial contiguity, we populate $\mathcal{Q}_r$ by only adding nodes that are connected to $\mathcal{I}_r$ through the edges $\mathcal{E}_S$. The priority of each candidate S_j is set by the number of superpixels in $\mathcal{I}_r$ that are at a distance smaller than τ. Let $\mathcal{D}_r(S_j) = \{d_{\mathcal{H}}(S_i, S_j) \mid S_i \in \mathcal{I}_r\}$ denote the set of distances between S_j and the constituent superpixels of $\mathcal{I}_r$. Using this set $\mathcal{D}_r(S_j)$, we can write the priority function as

$$p_r(S_j) = |\{d_j | d_j \leq \tau, d_j \in \mathcal{D}_r(S_j)\}| \,. \tag{4.9}$$

We assume that after each *enqueue* operation, $\mathcal{Q}_r$ is a queue of *unique* superpixels ordered by the priority function (4.9). We break ties in priority by picking the element that has a lower mean distance computed as

$$\frac{1}{p_r(S_j)} \sum d_j, \qquad d_j \in \{d_j \leq \tau, d_j \in \mathcal{D}_r(S_j)\}. \tag{4.10}$$

In the unlikely case of a second tie, we randomly chose one of the two superpixels. The *dequeue* operation removes one superpixel that has the highest priority in $\mathcal{Q}_r$.

To initialize the region growing, we pick a seed by selecting the superpixel with the smallest covariance of surface normals

$$\widehat{S}_i = \arg \min_{S_i \in \mathcal{V}_S} ||\mathbf{C}_{\widehat{\mathbf{n}}_i}||_F. \tag{4.11}$$

The region $\mathcal{I}_r$ is initialized with this seed node and its neighbors $\mathcal{N}(\widehat{S}_i)$ are enqueued in $\mathcal{Q}_r$. If the next candidate pixel N_j from $\mathcal{Q}_r$ satisfies $\text{med}(\mathcal{D}_r(N_j)) \leq \tau$, it is absorbed in the region $\mathcal{I}_r$ and its neighbors are enqueued in $\mathcal{Q}_r$.

Notation:
$\mathcal{V}_S$ – superpixel vertex set
$S_i = \{\widehat{\mathbf{p}}_i, \mathbf{C}_{\widehat{\mathbf{p}}_i}, \widehat{\mathbf{n}}_i, \mathbf{C}_{\widehat{\mathbf{n}}_i}\}$ – i^{th} superpixel in set $\mathcal{V}_S$
$\mathcal{N}(S_i) = \{S_j | (S_i, S_j) \in \mathcal{E}_S\}$ – set of neighbors of S_i
$\mathcal{I}_r$ – set of superpixels comprising the rth region
$\mathcal{D}_r(S_j) = \{d_{\mathcal{H}}(S_i, S_j) \mid S_i \in \mathcal{I}_r\}$ – set of distances (4.8) between S_j and elements of $\mathcal{I}_r$
$\mathcal{Q}_r$ – priority queue of superpixels with priority $p_r(S_j) = |\{d_j | d_j \leq \tau, d_j \in \mathcal{D}_r(S_j)\}|$
Input:
$\mathcal{G}_S = (\mathcal{V}_S, \mathcal{E}_S)$ – superpixel graph
τ – threshold for region merging
Output:
$\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_R\}$ – The set of R regions given by their constituent superpixels
Algorithm:

- Initialize $r \leftarrow 0$, $\mathcal{Q} \leftarrow \phi$, $\mathcal{I} \leftarrow \phi$
- while $\mathcal{V}_S \neq \phi$, do

  $r \leftarrow r + 1, \quad \mathcal{I}_r \leftarrow \phi$

  $\widehat{S}_i = \arg\min_{S_j \in \mathcal{V}_S} ||\mathbf{C}_{\widehat{\mathbf{n}}_j}||$

  $\mathcal{I}_r \leftarrow \mathcal{I}_r \cup \widehat{S}_i$

  enqueue$(\mathcal{Q}_r, \mathcal{N}(\widehat{S}_i))$

  while $\mathcal{Q}_r \neq \phi$, do

    $N_j \leftarrow$ dequeue$(\mathcal{Q}_r)$

    if med$(\mathcal{D}_r(N_j)) \leq \tau$

      $\mathcal{I}_r \leftarrow \mathcal{I}_r \cup N_j$

      enqueue$(\mathcal{Q}_r, \mathcal{N}(N_j))$

    else

      $\mathcal{Q}_r \leftarrow \phi$, empty the priority queue

  $\mathcal{E}_S \leftarrow \mathcal{E}_S \setminus \{(S_i, S_j), \forall S_i \in \mathcal{I}_r, S_j \in \mathcal{V}_S\}$, update the edge set

  $\mathcal{V}_S \leftarrow \mathcal{V}_S \setminus \mathcal{I}_r$, update the vertex set
- return $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_R\}$

Figure 4.4: Region growing algorithm.

This process of merging candidates and updating $\mathcal{Q}_r$ is repeated until either the queue is empty, or a candidate fails to meet the merging criterion. Since N_j is the candidate with the highest priority, if it does not meet the criterion to merge with $\mathcal{I}_r$, none of the other candidates in $\mathcal{Q}_r$ can be merged. Therefore we empty the queue and update $\mathcal{V}_S$ and $\mathcal{E}_S$ by removing all nodes and edges corresponding to the elements of $\mathcal{I}_r$. The seed node for a new region is selected using (4.11) and the entire process is repeated until all superpixels in $\mathcal{V}_S$ are assigned to regions. Fig. 4.4 summarizes the

region growing algorithm.

The goal of region growing is to identify all significant planar regions in the scene. Even though we pick a conservative threshold τ that prefers overfragmentation to underfragmentation, some regions may still contain a small number of outlier superpixels. We need to eliminate any outliers prior to estimating the planar model parameters for accurate estimates. We apply an M-estimator [79, Sec. 4.4.2, pp 163] with a Tukey biweight loss function on the superpixels for each region. The weights are determined by using (4.8) as the distance function. The scale parameter for the M-estimator is set to one since the distance function is heteroscedastic. At convergence, the superpixels corresponding to zero weights are eliminated as outliers. The planar model parameters are then estimated by computing the TLS estimate over all the pixels belonging to the inlier superpixels.

4.2.3 Model Association

The set of fragments $\mathcal{I}$ obtained in the previous stage, comprise of groups of *inlier* superpixels. Similar to the superpixels, we generate a fragment graph $\mathcal{G}_F = (\mathcal{V}_F, \mathcal{E}_F)$, where each node is given by $F_i \in \mathcal{V}_F = \{\widehat{\mathbf{n}}_i, \mathbf{C}_{\widehat{\mathbf{n}}_i}, \mathcal{P}_i\}$, where $\widehat{\mathbf{n}}_i$ and $\mathbf{C}_{\widehat{\mathbf{n}}_i}$ are the TLS estimate of the surface normal and the empirical covariance of the normals computed over the entire fragment (4.3) respectively. The set $\mathcal{P}_i$ is the *index set of pixels* that belong to the fragment F_i.

Since these fragments were obtained by merging superpixels, they have coarse boundaries. Recall that at the normal computation stage, pixels that failed the χ^2 test were not used for superpixel segmentation and therefore are not assigned to any model yet. At this stage, we perform a pixel-level model association, and also use it to merge any potentially overfragmented regions.

We perform a maximum likelihood pixel-level assignment using two distance functions. The directional planar residual [36] is the deviation of a point from the plane in the direction of the incident ray originating from the camera center. We compute its heteroscedastic form for the 3D point $\mathbf{p}_k$ corresponding to the kth pixel with respect to

the ith plane

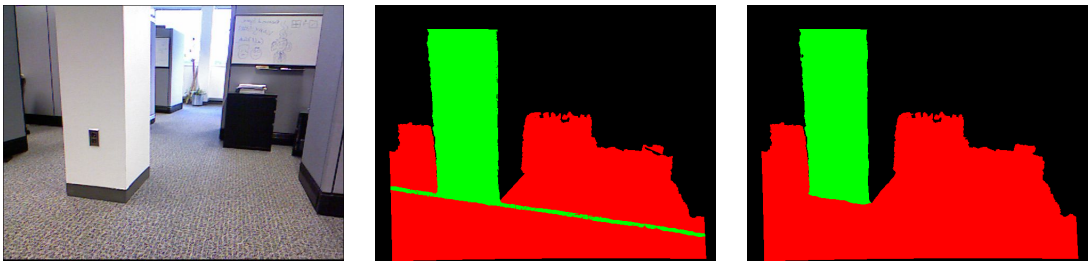
$$
\begin{aligned}
r_{ik} &= \frac{\widehat{\mathbf{n}}_i^\top (\widehat{\mathbf{p}}_i - \mathbf{p}_k)}{\cos\theta_{ik}} \\
\sigma_{r_{ik}} &= \sigma_{z_k} \frac{\|\mathbf{p}_k\|}{z_k} \\
\rho_{ik} &= \frac{r_{ik}}{\sigma_{r_{ik}}}
\end{aligned} \tag{4.12}
$$

where z_k is the depth of the point $\mathbf{p}_k$ and $\widehat{\mathbf{n}}_i$ and $\widehat{\mathbf{p}}_i$ are the normal and the centroid of the plane respectively. The scalar angle θ_{ik} is the angle between the normal $\widehat{\mathbf{n}}_i$ and the incident ray $\mathbf{p}_k$. The choice of this residual function is motivated by [36], where it was shown to be more insensitive to the orientation of the plane compared to orthogonal residuals. For identifying potential outliers, we need a threshold to distinguish between inliers and outliers with respect to a model. Most fragments comprise of at least a few superpixels, and therefore contain thousands of pixels. This being a sufficiently large sample, we simply chose the inlier/outlier threshold as

$$
\varrho_i = \max_{k \in \mathcal{P}_i} \rho_{ik}. \tag{4.13}
$$

Any pixel that satisfies $\rho_{ik} \leq \varrho_i$ is considered as a potential inlier. However, this residual function is inadequate to perform the model association for our purpose. The residual function (4.12) assumes an unbounded plane and generates a distance measure for each individual pixel. This can cause bleeding of regions into neighbors as shown in Fig. 4.5b. The thin green strip on the ground plane shows the set of pixels that actually belong to the red fragment but are assigned to the green fragment. This can be avoided by taking into account the orientation of the surface normal in a local spatial neighborhood of a candidate pixel. Since we have already computed the surface normal at each pixel, we define a heteroscedastic cosine distance function between the plane normal $\widehat{\mathbf{n}}_i$ and the surface normal $\mathbf{n}_k$ computed at the kth pixel

$$
\phi_{ik} = \frac{1 - \widehat{\mathbf{n}}_i^\top \mathbf{n}_k}{\sqrt{\mathbf{n}_k^\top \mathbf{C}_{\widehat{\mathbf{n}}_i} \mathbf{n}_k}} \tag{4.14}
$$



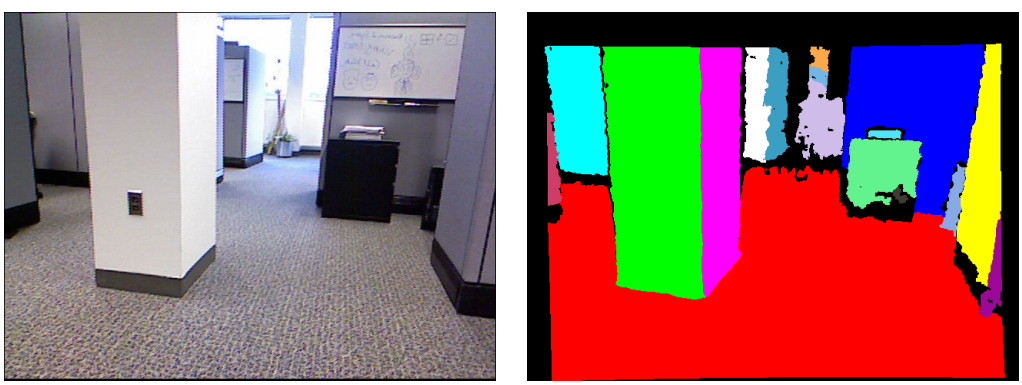
(a) (b) (c)

Figure 4.5: Model association. (a) RGB image for reference. (b) Labels assigned competitively using only the directional planar residuals (4.12). (c) Labels assigned competitively using both (4.12) and (4.14).

where $\mathbf{C}_{\widehat{\mathbf{n}}_i}$ is the covariance of the plane normal $\widehat{\mathbf{n}}_i$. Similar to (4.13), we compute the threshold φ_i for this function (4.14). Pixels that simultaneously satisfy $\rho_{ik} \leq \varrho_i$ and $\phi_{ik} \leq \varphi_i$ are considered as potential inliers. The effect of using both these distance functions is shown in Fig. 4.5c. While maintaing a connectivity constraint, we use these two distances as negative log-likelihoods and perform the final model association by

$$L_k = \underset{\substack{i=1,\ldots,|\mathcal{V}_F| \\ \rho_{ik} \leq \varrho_i,\ \phi_{ik} \leq \varphi_i}}{\operatorname{argmin}} \quad \rho_{ik} + \phi_{ik} \qquad \forall\ k. \tag{4.15}$$

Finally, we check for any unlikely overfragmentations, and merge them together. A fragment F_j is permitted to merge with F_i if and only if all the following criteria are met



(a) (b)

Figure 4.6: Pixel level model association. (a) RGB image for reference. (b) Pixel-level labeled image.

- The fragments should be adjacent to each other, i.e., $(F_i, F_j) \in \mathcal{E}_F$.
- Only smaller fragments can be merged with larger ones, i.e., $|\mathcal{P}_j| <= |\mathcal{P}_i|$.
- At least η percent of pixels, $k \in \mathcal{P}_j$ satisfy $\rho_{ik} \leq \varrho_i$ and $\phi_{ik} \leq \varphi_i$.

The fraction of overlap is set to a large value, $\eta = 90\%$. The labeled image after the post-processing is shown in Fig. 4.6. The boundary contours can easily be extracted from these segments.

4.3 Experimental Evaluation

We present the experimental validation for the proposed approach in this section. First we describe the datasets used for performance evaluation. Then we show qualitative results from a subset of each dataset. We then show a quantitative evaluation of planar segmentation using the Kinect Segmentation dataset [87], the only one which has ground truth available.

4.3.1 Datasets

NYU Depth Planar: NYU depth dataset [104] is a large image dataset with RGB-D images taken from the Kinect sensor of indoor scenes. The dataset comes with manual segmentation and manually assigned semantic labelings for each segment. We choose a small subset of scenes which contain a good number of planar surfaces. Two example image sets are shown in Fig. 4.7 where column (a) shows RGB images (nyu-0031, nyu-0070), column (b) the corresponding depth images and column (c) the corresponding manual segmentation. Note that different wall segments have the same label.

SCT Depth Planar: We found two problems with the NYU dataset: (i) The RGB and depth images are not exactly aligned - thus it is not possible to accuracetly transfer the segmentation and labelings to the depth data. (ii) The manual segmentation is done based on object categories and therefore do not necessarily follow planar boundaries. For example, all walls in an image are often annotated by a single segment labeled 'walls', while for our evaluation we want each planar wall to be segmented distinctly.

(a) (b) (c)

Figure 4.7: Real indoor images (NYU Depth Planar dataset). (a) RGB images (nyu-0031, nyu-0070). (b) Corresponding depth images. (c) Corresponding ground truth segmentation. Note that segments do not respect plane boundaries.

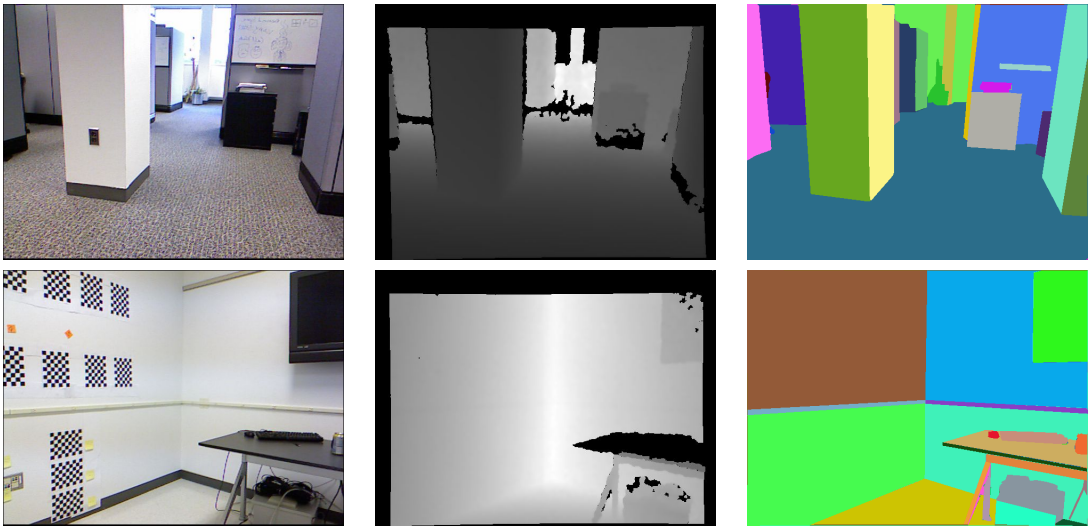
Consequently, we collected our own dataset using the Kinect sensor and manually annotated the images such that distinct planar segments carry a unique label.

Two example image sets (sct-1, sct-3) are shown in Fig. 4.8 where column (a) shows the RGB images, column (b) shows the corresponding depth images and column (c) shows the corresponding manual segmentation. Note that each planar segment is assigned a different label represented by a unique color.

Kinect Segmentation Dataset: The Kinect Segmentation dataset [87] contains 30 RGB-D images taken from the Kinect sensor. The indoor scenes contain mostly planar or cylindrical surfaces with a manual segmentation made available. Two example image sets are shown in Fig. 4.9 where column (a) shows RGB images (ks-003, ks-013), column (b) the corresponding depth images and column (c) the corresponding manual segmentation.

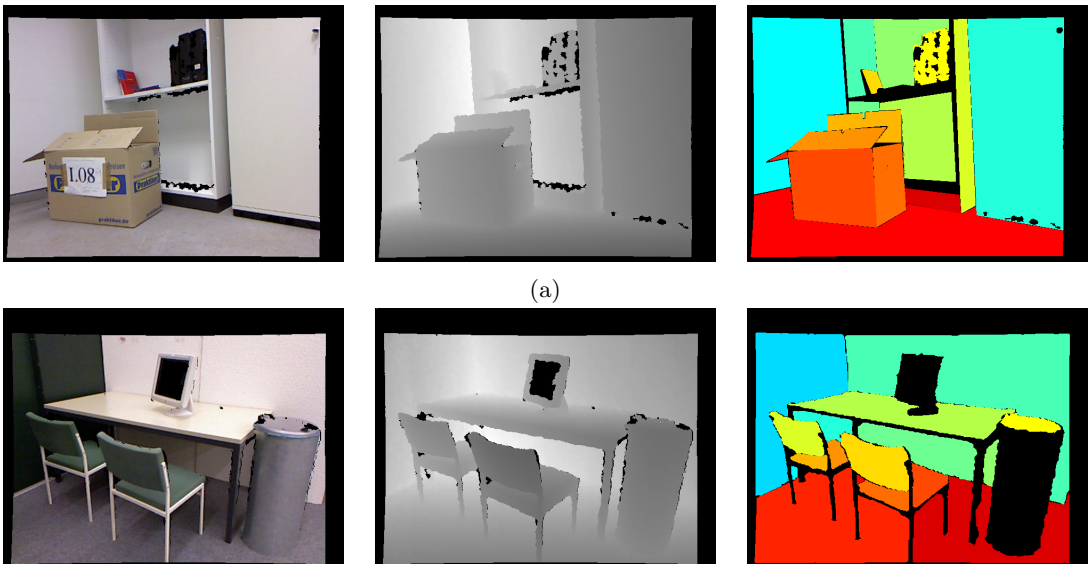
4.3.2 Qualitative Evaluation

Qualitative results over sample images from these datasets are shown in Fig. 4.10 - 4.12. Each column in the figure shows the results of our method on a different image from the corresponding dataset. Fig. 4.10 shows results on images from the NYU dataset,



(a) (b) (c)

Figure 4.8: Real indoor images (SCT Depth Planar dataset). (a) RGB images (sct-1, sct-3). (b) Corresponding depth images. (c) Corresponding ground truth segmentation.



(a)

(b)

Figure 4.9: Kinect Segmentation dataset. (a) RGB images (ks-003, ks-013). (b) Corresponding depth images. (c) Corresponding ground truth segmentation.

sample results from the SCT dataset are shown in Fig. 4.11, and those from the Kinect Segmentation dataset are in Fig. 4.12.

The results are organized as follows. The depth image is shown in the first row. The second row shows the normal image computed using the SVD method described in Section 4.2.1. The third row shows the final segmentation with the identically colored regions representing the planar fragments. Finally, we show the RGB image in the last row for reference.

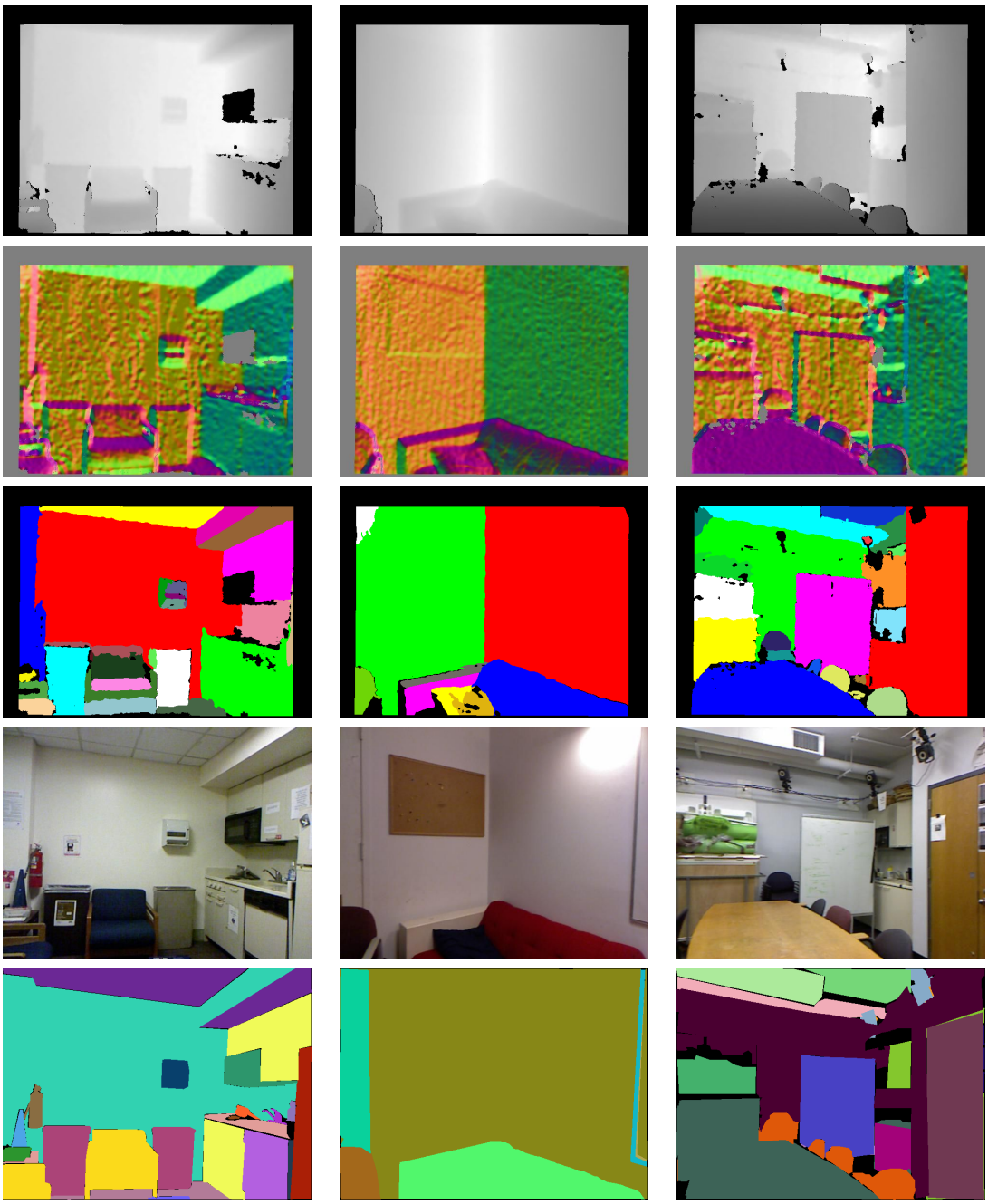
4.3.3 Quantitative Evaluation

We perform a quantitative evaluation of the proposed method using the Kinect Segmetnation data set. We use the SegComp tool [50] to evaluate the resulting planar segmentation and briefly summarize the evaluation mechanism. The details and related proofs are provided in [50]. We denote $\mathcal{P}_i,\ i = 1\ldots,n$ as the set of pixels corresponding to the n detected segments. The m ground truth segments in the image are denoted by $\mathcal{S}_j,\ j = 1,\ldots,m$. The SegComp evaluation tool uses a threshold t, as fraction of overlapping pixels between the ground truth and the detected segments. For a given threshold t, it counts the number of planes contributing to each of the following categories of segmentation

- Correct Detection - When the fraction of overlap between segments $\mathcal{P}_i$ and $\mathcal{S}_j$ are mutually greater than t, i.e., when both conditions are satisfied $|\mathcal{P}_i \cap \mathcal{S}_j| > t|\mathcal{S}_j|$ and $|\mathcal{P}_i \cap \mathcal{S}_j| > t|\mathcal{P}_i|$, then a correct detection is counted.

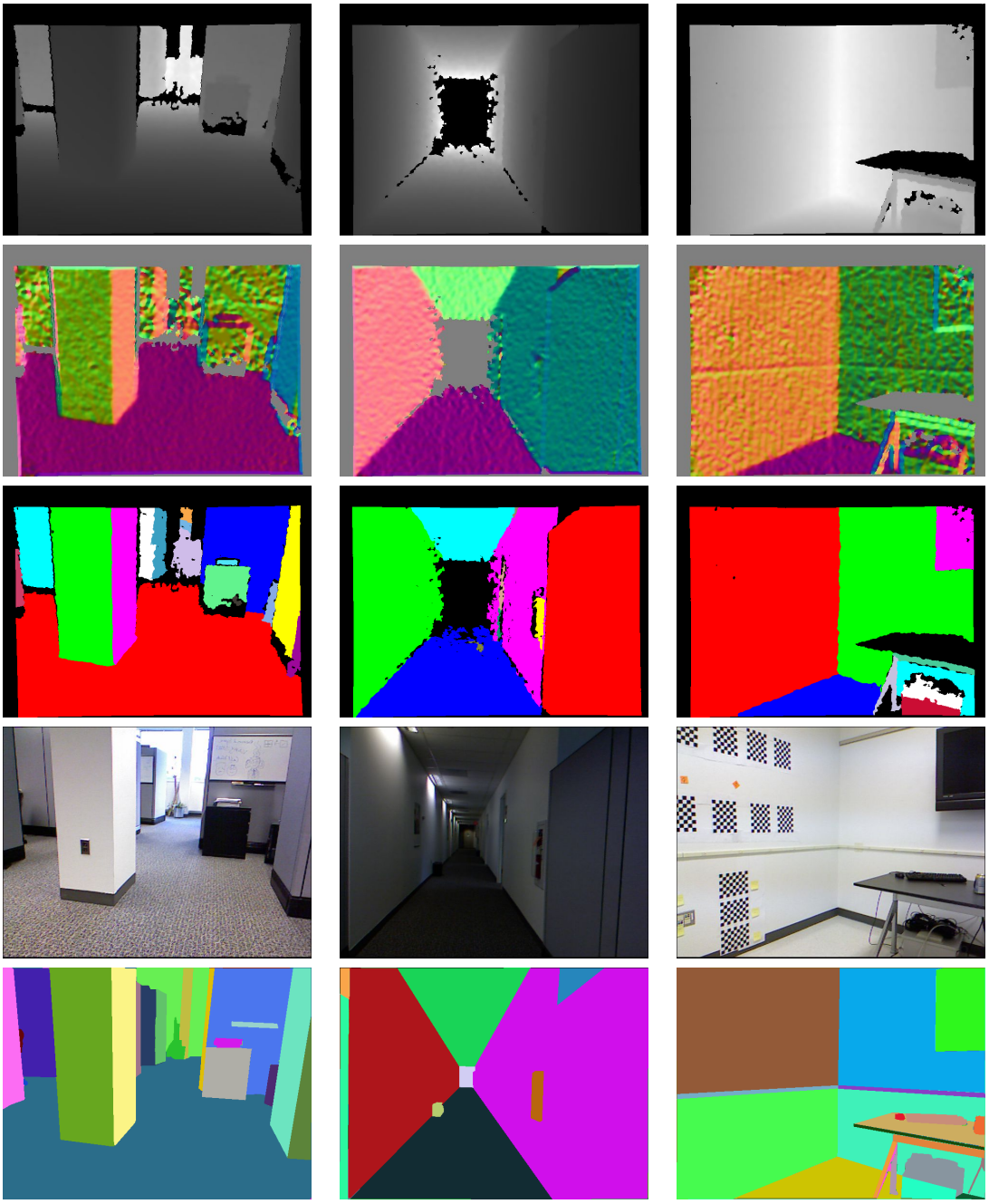
- Oversegmentation - When two or more detected segments $\{\mathcal{P}_{i_k}\}, k \geq 2$ overlap with $\mathcal{S}_j$ such that for each $\mathcal{P}_{i_k}$, the fraction of overlapping pixels exceeds the threshold t, and if the fraction of overlap between $\mathcal{S}_j$ and $\cup_k \mathcal{P}_{i_k}$ is larger than t, then an oversegmentation is recorded.

- Undersegmentation - When two or more ground truth segments $\{\mathcal{S}_{j_k}\}, k \geq 2$ overlap with $\mathcal{P}_i$ such that for each $\mathcal{S}_{j_k}$, the fraction of overlapping pixels exceeds the threshold t, and if the fraction of overlap between $\mathcal{P}_i$ and $\cup_k \mathcal{S}_{j_k}$ is larger than t, then an undersegmentation is recorded.



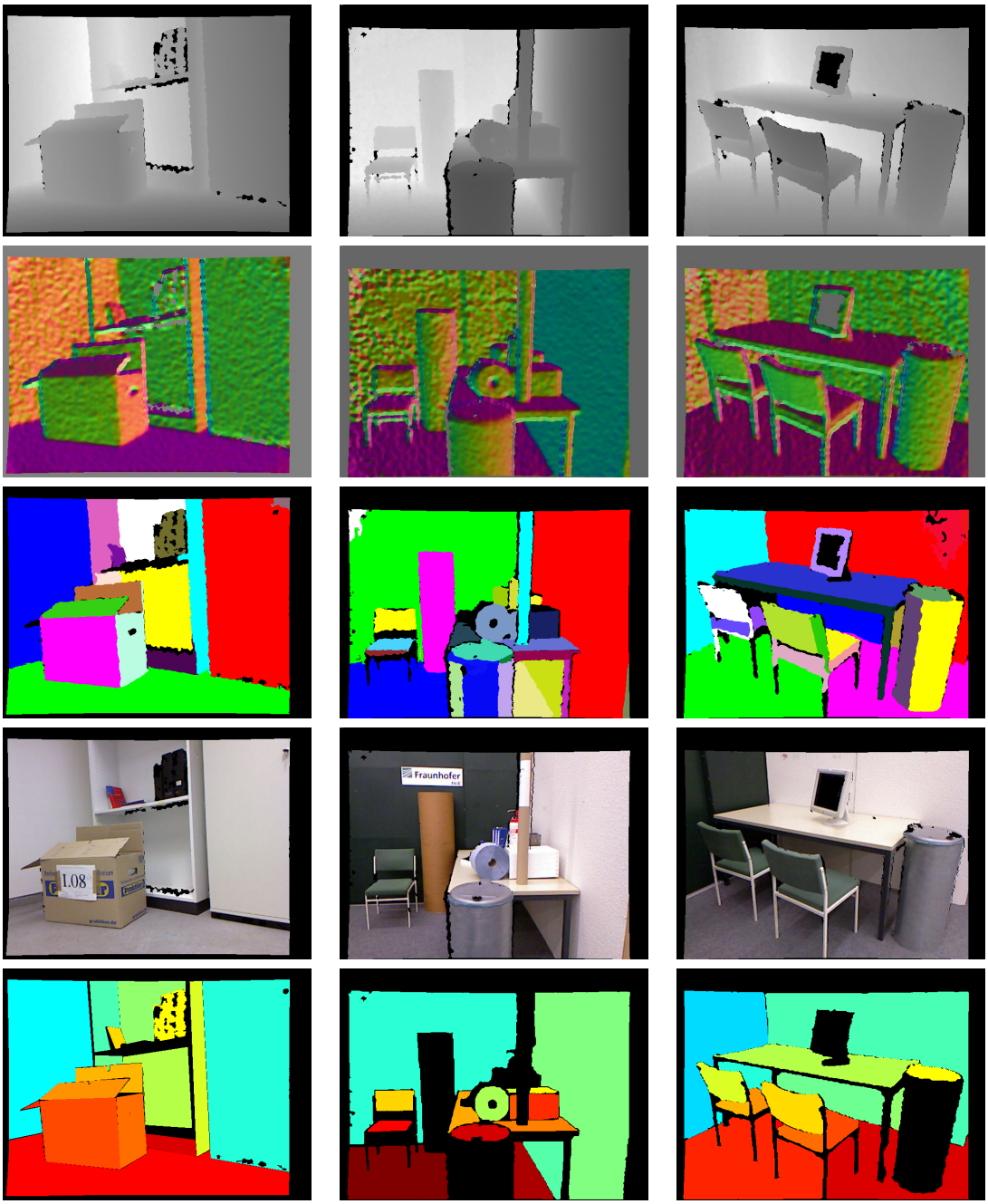
(a) (b) (c)

Figure 4.10: Results on NYU depth planar dataset. (a) nyu-0001; (b) nyu-0017; and, (c) nyu-0031 images. Rows top to bottom: depth, surface-normals, final segmentation, RGB, ground truth segmentation. The ground truth segmentation for NYU is only given for completeness as the segments do not follow plane boundaries.



(a) (b) (c)

Figure 4.11: Results on the SCT depth planar dataset. (a) SCT-1; (b) SCT-2; and, (c) SCT-3 images. Rows top to bottom: depth, surface-normals, final segmentation, RGB, ground truth segmentation.



(a) (b) (c)

Figure 4.12: Results on the Kinect Segmentation depth planar dataset. (a) ks-003; (b) ks-013; and, (c) ks-021 images. Rows top to bottom: depth, surface-normals, final segmentation, RGB, ground truth segmentation. The nonplanar regions are labeled with black pixels in the ground truth segmentation.

Table 4.1: Quantitative evaluation of the planar segmentation using the SegComp tool [50]. Results are compared with [87]. The numbers are fractions of planes contributing to each category of segmentation, averaged over 30 images of the Kinect Segmentation Dataset.

Overlap	Correct		Overseg		Underseg		Missed	
	Ours	[87]	Ours	[87]	Ours	[87]	Ours	[87]
0.51	0.5903	0.5491	0.1256	0.1484	0.0469	0.0586	0.1839	0.2424
0.6	0.5692	0.4844	0.1132	0.1406	0.0415	0.0547	0.2281	0.3125
0.7	0.5267	0.4531	0.1132	0.1250	0.0276	0.0391	0.2985	0.4063
0.8	0.4349	0.3281	0.0989	0.1016	0.0212	0.0234	0.4186	0.5625
0.9	0.2579	0.1563	0.0735	0.0547	0.0012	0.0078	0.6637	0.7891

- Missed classification - When a region $\mathcal{S}_j$ is not a part of a correct detection, oversegmentation or undersegmentation incident, it is counted as a missed detection.

Note that these categories are not unique, but it was shown in [50], that for $0.5 < t < 1.0$, a segment can at most contribute to three instances, one each of correct detection, oversegmentation and undersegmentation. Therefore the evaluation of segmentation is done by varying the threshold t and counting the instances of these classes for each image.

We present the evaluation in Table 4.1 and compare with the results reported in [87] on the Kinect Segmentation dataset. The method proposed in [87] is a combination of RANSAC and Hough transforms to detect multiple planes in a range image. Clearly, a better segmentation is expected to achieve a high correct detection rate and low oversegmentation, undersegmentation and missed detection rates. It can be seen that our method almost always performs better compared to [87]. The numbers shown in the table are the fraction of planes contributing to the four categories of segmentations averaged over all the 30 images in the Kinect Segmentation dataset. For example, for $t = 0.51$, on average, our method correctly detects about 59 percent of the planes, while misses about 18 percent of the planes in an image.

4.4 Discussion

We presented a bottom-up, scalable procedure for detecting planar structures in a single RGB-D image from Microsoft Kinect. The proposed method generates a large number of planar model hypotheses by computing surface normals at each 3D point. A superpixel segmentation algorithm is used to prune these hypotheses to a much smaller set. We combined superpixels to generate a unique set of planar models by employing a region growing algorithm that exploits the underlying heteroscedasticity in the data. The pixels were then competitively assigned to the set of planar models to generate a parametric representation of the depth image. We showed qualitative results on different datasets with a varying degree of scene complexity and clutter. We also compared the quantitative performance of our algorithms with a recent planar segmentation method [87] using the Kinect Segmentataion Dataset.

This representation of depth images can be viewed as a planar segmentation of the scene, which could be useful for robotics applications such as object detection, recognition, path planning etc. It can also be used to compress a depth image before transmission, where the boundary contour and the planar parameters characterize a segment. The depth of its constituent pixels can be recovered using the intrinsic parameters and the planar equation. Finally, this set of 3D planes can be viewed as a 3D model of the scene. Although for building a complete parametric, watertight 3D model, it is necessary to parametrize the boundaries of the planes.

Intersecting boundaries between planes are always linear and can be easily parametrized. Occluding boundaries, i.e., boundaries caused by a plane occluding another plane, are more complicated as they can take an arbitrary shape. The occasional occurrences of holes, i.e., missing depth data makes the problem of occluding boundaries harder. The depth discontinuity at the occluding boundaries can vary significantly, reliable detection using depth alone may be difficult. Besides, depth sensors like Microsoft Kinect also has certain systematic errors, which were not modeled in our noise parametrization. For example, in the third row of Fig. 4.10, the top-left corner has a small segment detected in nyu-0017 and nyu-0031. This can be seen in the top right and left corners

of the segmented images in Fig. 4.12 too. This is a common artifact in the Kinect depth images, where the corners are 'dog-eared', possibly because of the interpolation during the depth estimation process.

Some of these issues can be resolved by leveraging from the RGB channel of Microsoft Kinect. For example, the occluding object may have a different color or texture than the background plane. The color channel may contain complementary information to the detected planar regions. We plan to use this information in future.

Chapter 5

Semi-supervised Kernel Mean Shift Clustering

Mean shift is a popular mode seeking algorithm, which iteratively locates the modes in the data by maximizing the kernel density estimate (KDE). Although the procedure was initially described decades ago [38], it was not popular in the vision community until its potential uses for feature space analysis and optimization were understood [16, 24]. The nonparametric nature of mean shift makes it a powerful tool to discover arbitrarily shaped clusters present in the data. Additionally, the number of clusters is automatically determined by the number of discovered modes. Due to these properties, mean shift has been applied to solve several computer vision problems, e.g., image smoothing and segmentation [23, 119], visual tracking [21, 45] and information fusion [15, 22]. Mean shift clustering was also extended to nonlinear spaces, for example, to perform clustering on analytic manifolds [107, 114] and kernel induced feature space [102, 117] under an unsupervised setting.

In many clustering problems, in addition to the unlabeled data, often some additional information is also easily available. Depending on a particular problem, this additional information could be available in different forms. For example, the number of clusters or a few *must-link* (similarity) and *cannot-link* (dissimilarity) pairs could be known a-priori. In the last decade, semi-supervised clustering methods that aim to incorporate prior information into the clustering algorithm as a guide, have received considerable attention in machine learning and computer vision [6, 13, 42]. Increasing interest in this area has triggered the adaptation of several popular unsupervised clustering algorithms into a semi-supervised framework, e.g., background constrained k-means [118], constrained spectral clustering [56, 73] and kernel graph clustering [62]. It is shown that unlabeled data, when used in conjunction with a small amount of

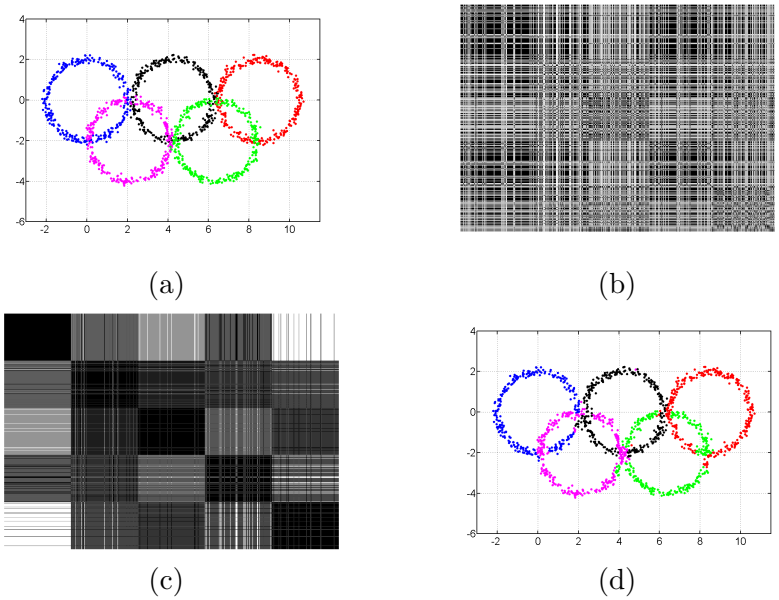


Figure 5.1: Olympic Circles. (a) Original data in the input space with five different clusters. (b) Pairwise distance matrix (PDM) using the modes discovered by unsupervised mean shift clustering performed in the kernel space. The points are ordered according to class. (c) PDM using the modes found after supervision was added. (d) Clustering results shown in the input space.

labeled data, can produce significant improvement in clustering accuracy. However, despite these advances, mean shift clustering has largely been ignored under the semi-supervised learning framework. To leverage all the useful properties of mean shift, we adapt the original unsupervised algorithm into a semi-supervised clustering technique.

The work in [113] was the only attempt to introduce weak supervision into the kernel mean shift algorithm where the additional information was provided through a few pairwise must-link constraints. In that framework, each pair of must-link points was collapsed to a single point through a linear projection operation, guaranteeing their association with the same cluster. In this chapter, we extend that work by generalizing the linear projection operation to a linear *transformation* of the kernel space that permits us to scale the distance between the constraint points. Using this transformation, the must-link points are moved closer to each other, while the cannot-link points are moved farther apart. We show that this transformation can be achieved implicitly by

modifying the kernel matrix. We also show that given one constraint pair, the corresponding kernel update is equivalent to minimizing the log det divergence between the updated and the initial kernel matrix. For multiple constraints, this result proves to be very useful since we can learn the kernel matrix by solving a constrained log det minimization problem similar to [55].

Fig. 5.1 illustrates the basic approach. The original data with 300 points along each of the five circles is shown in Fig. 5.1a. The data is first mapped to a kernel space using a Gaussian kernel ($\sigma = 0.5$). In the first case, kernel mean shift is directly applied to the data points in the absence of any supervision. Fig. 5.1b shows the corresponding 1500×1500 pairwise distance matrix (PDM) obtained using modes recovered by mean-shift. The lack of block diagonal structure implies the absence of meaningful clusters discovered in the data. In the second case, we assume that only 20 points from each cluster are labeled at random (6.7% of the data) to generate pairwise must-link and cannot-link constraints. These constraints are then used to transform the initial kernel space by learning an appropriate kernel matrix. Note that, although we generate pairwise constraints using a small fraction of labeled points, the algorithm *does not* require the explicit knowledge of class labels. Fig. 5.1c shows the PDM between the modes obtained when kernel mean shift is applied to the transformed feature points. In this case, the five-cluster structure is clearly visible in the PDM. Finally, Fig. 5.1d shows the corresponding results in the original space. Through this example, it becomes evident that a small amount of supervision can improve the clustering performance significantly.

The main contributions of our work are summarized below.

- We develop the variable bandwidth kernel mean shift algorithm to perform clustering in high-dimensional kernel spaces.

- We introduce supervision into the kernel mean shift algorithm by imposing pairwise must-link and cannot-link constraints via a linear transformation of the kernel space.

- Given a constraint pair, we show that the corresponding kernel update is equivalent to minimizing the log det divergence between the updated and the initial

kernel matrix.

- Using the must-link constraint points, we propose a method to select the bandwidth parameter for kernel mean shift.
- For the initial choice of a Gaussian kernel, we also select the scale parameter automatically using the pairwise constraints.

The chapter is organized as follows. In Section 5.1 we present some related work. In Section 5.2, we briefly review the Euclidean mean shift algorithm and its extension to kernel spaces. In Section 5.3 we derive the expression for kernel updates by applying a linear transformation to the kernel space that satisifies the specified constraints. We also motivate the formulation of the kernel learning as a Bregman divergence minimization problem. The constrained $\log\det$ divergence minimization formulation is treated in Section 5.4. We describe the complete algorithm in Section 5.6. In Section 5.7, we show experimental results on synthetic and real data and conclude with a discussion in Section 5.8.

5.1 Related Work

Semi-supervised clustering has received a lot of attention in the past few years due to its highly improved performance over traditional unsupervised clustering methods [7, 13]. As compared to fully-supervised learning algorithms, these methods require a weaker form of supervision in terms of both the amount and the form of labeled data used. In clustering, this is usually achieved by using only a few pairwise must-link and cannot-link constraints. Since generating pairwise constraints does not require the knowledge of class labels or the number of clusters, they can easily be generated using supplementary data. For example, while clustering images of objects, two images with similar text annotations could be used as a must-link pair. We discuss some of the traditional unsupervised clustering methods and their semi-supervised variants below.

Partitioning based clustering: k-means is one of the oldest and most popular clustering algorithm. See [54] for an extensive review of all the variants of k-means algorithm.

One of its popular semi-supervised extensions using pairwise constraints was proposed in [118]. The method partitions the data into k non-overlapping regions such that must-link pairs are enforced to lie in the same cluster while the cannot-link pairs are enforced to lie in different clusters. However, since the method enforces all constraints rigidly, it is sensitive to labeling errors in pairwise constraints. Basu et al. [6] proposed a more robust, probabilistic model by explicitly allowing relaxation for a few constraints in k-means clustering. Similarly, Bilenko et al. [9] proposed a metric learning based approach to incorporate constraints into k-means clustering framework. Since the clustering in both these methods is performed in the input space, these methods fail to handle non-linear cluster boundaries. More recently, Kulis et al. proposed semi-supervised kernel k-means (SSKK) algorithm [62], that constructs a kernel matrix by adding the input similarity matrix to the constraint penalty matrix. This kernel matrix is then used to perform k-means clustering.

Graph based clustering: Spectral clustering [86] is another very popular technique that can also cluster data points with non-linear cluster boundaries. In [56], this method was extended to incorporate weak supervision by updating the computed affinity matrix for the specified constraint points. Later, Lu et al. [73] further modified the algorithm by propagating the specified constraints to the remaining points using a Gaussian process. More recently, Lu and Ip [74] showed improved clustering performances by applying exhaustive constraint propagation and handling soft constraints (E2CP). One of the fundamental problems with this method is that it can be sensitive to labeling noise since the effect of a mislabeled data point pair could easily get propagated throughout the affinity matrix. Moreover, in general, the spectral clustering methods suffer when the clusters have very different scales and densities [84].

Density based clustering: Clustering methods in this category make use of the estimated local density of the data to discover clusters. Gaussian Mixture Models (GMM) are often used for soft clustering where each mixture represents a cluster distribution [70]. Mean shift [23, 24, 113] was employed in computer vision for clustering data in the feature space by locating the modes of the nonparametric estimate of the underlying density. There exist other density based clustering methods that are less known

in the vision community, but have been applied to data mining applications. For example, DBSCAN [32] groups together points that are connected through a chain of high-density neighbors that are determined by the two parameters: neighborhood size and minimum allowable density. SSDBSCAN [68] is a semi-supervised variant of DBSCAN that explicitly uses the labels of a few points to determine the neighborhood parameters. C-DBSCAN [95] performs a hierarchical density based clustering while enforcing the must-link and cannot-link constraints.

5.2 Kernel Mean Shift Clustering

First, we briefly describe the Euclidean mean shift clustering algorithm in Section 5.2.1 and then derive the kernel mean shift algorithm in Section 5.2.2.

5.2.1 Euclidean Mean Shift Clustering

Given n data points $\mathbf{x}_i$ on a d-dimensional space $\mathbb{R}^d$ and the associated diagonal bandwidth matrices $h_i\mathbf{I}_{d\times d}$, $i = 1, ..., n$, the sample point density estimator obtained with the kernel profile $k(\mathbf{x})$ is given by

$$f(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{h_i^d}\,k\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right). \tag{5.1}$$

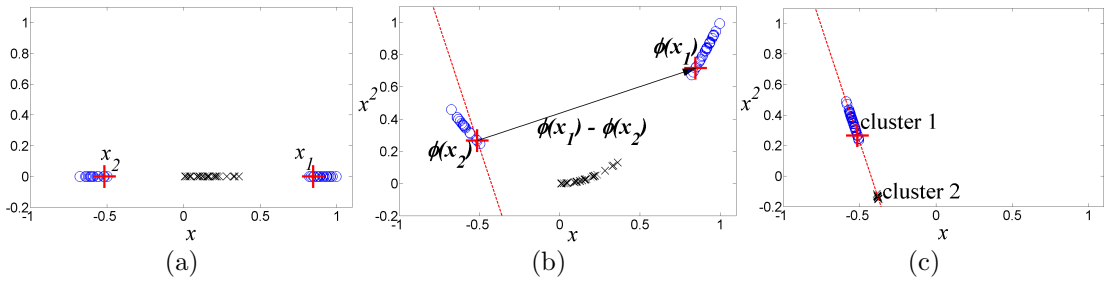
We utilize multivariate normal profile

$$k(x) = e^{-\frac{1}{2}x} \qquad x \geq 0. \tag{5.2}$$

Taking the gradient of (5.1), we observe that the stationary points of the density function satisfy

$$\frac{2}{n}\sum_{i=1}^{n}\frac{1}{h_i^{d+2}}(\mathbf{x}_i-\mathbf{x})\,g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right) = 0 \tag{5.3}$$

where $g(x) = -k'(x)$. The solution is a local maximum of the density function which



(a) (b) (c)

Figure 5.2: Clustering with must-link constraints. (a) Input space. Red markers are the constraint pair (x_1, x_2). (b) The input space is mapped to the feature space via quadratic mapping $\boldsymbol{\phi}(x) = [x \;\; x^2]^\top$. The black arrow is the constraint vector $(\boldsymbol{\phi}(x_1) - \boldsymbol{\phi}(x_2))^\top$, and the red dashed line is its null space. (c) The feature space is projected to the null space of the constraint vector. Constraint points collapse to a single point and are guaranteed to be clustered together. Two clusters can be easily identified.

can be iteratively reached using mean shift procedure

$$\delta\mathbf{x} = \frac{\sum_{i=1}^{n} \frac{\mathbf{x}_i}{h_i^{d+2}}\, g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)}{\sum_{i=1}^{n} \frac{1}{h_i^{d+2}}\, g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h_i}\right\|^2\right)} - \mathbf{x} \tag{5.4}$$

where $\mathbf{x}$ is the current mean and $\delta\mathbf{x}$ is the mean shift vector. To recover from saddle points adding a small perturbation to the current mean vector is usually sufficient. Comaniciu and Meer [23] showed that the convergence to a local mode of the distribution is guaranteed.

5.2.2 Mean Shift Clustering in Kernel Spaces

We extend the original mean shift algorithm from the Euclidean space to a general inner product space. This makes it possible to apply the algorithm to a larger class of nonlinear problems, such as clustering on manifolds [114]. We also note that a similar derivation for fixed bandwidth mean shift algorithm was given in [117].

Let $\mathcal{X}$ be the input space such that the data points $\mathbf{x}_i \in \mathcal{X}$, $i = 1, ..., n$. Although, in general, $\mathcal{X}$ may not necessarily be a Euclidean space, for sake of simplicity, we assume $\mathcal{X}$ corresponds to $\mathbb{R}^d$. Every point $\mathbf{x}$ is then mapped to a d_ϕ-dimensional feature space

$\mathcal{H}$ by applying the mapping functions $\phi_l, l = 1, \ldots, d_\phi$, where

$$\boldsymbol{\phi}(\mathbf{x}) = [\phi_1(\mathbf{x}) \quad \phi_2(\mathbf{x}) \quad \ldots \quad \phi_{d_\phi}(\mathbf{x})]^\top. \tag{5.5}$$

Note that in many problems, this mapping is sufficient to achieve the desired separability between different clusters.

We first derive the mean shift procedure on the feature space $\mathcal{H}$ in terms of the explicit representation of the mapping $\boldsymbol{\phi}$. The point sample density estimator at $\mathbf{y} \in \mathcal{H}$, with the diagonal bandwidth matrices $h_i \mathbf{I}_{d_\phi \times d_\phi}$ is

$$f_{\mathcal{H}}(\mathbf{y}) = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{h_i^{d_\phi}} k\left(\left\| \frac{\mathbf{y} - \boldsymbol{\phi}(\mathbf{x}_i)}{h_i} \right\|^2 \right). \tag{5.6}$$

Taking the gradient of (5.6) w.r.t. $\boldsymbol{\phi}$, like (5.4), the solution can be found iteratively using the mean shift procedure

$$\delta \mathbf{y} = \frac{\sum_{i=1}^{n} \frac{\boldsymbol{\phi}(\mathbf{x}_i)}{h_i^{d_\phi+2}} g\left(\left\| \frac{\mathbf{y} - \boldsymbol{\phi}(\mathbf{x}_i)}{h_i} \right\|^2 \right)}{\sum_{i=1}^{n} \frac{1}{h_i^{d_\phi+2}} g\left(\left\| \frac{\mathbf{y} - \boldsymbol{\phi}(\mathbf{x}_i)}{h_i} \right\|^2 \right)} - \mathbf{y}. \tag{5.7}$$

By employing the kernel trick, we now derive the implicit formulation of the kernel mean shift algorithm. We define $K : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, a positive semidefinite, scalar kernel function satisfying for all $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$

$$K(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \boldsymbol{\phi}(\mathbf{x}'). \tag{5.8}$$

$K(\cdot)$ defines an inner product which makes it possible to map the data implicitly to a high-dimensional kernel space. Let

$$\boldsymbol{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1) \quad \boldsymbol{\phi}(\mathbf{x}_2) \quad \ldots \quad \boldsymbol{\phi}(\mathbf{x}_n)] \tag{5.9}$$

be the $d_\phi \times n$ matrix of the mapped points and $\mathbf{K} = \boldsymbol{\Phi}^T \boldsymbol{\Phi}$ be the $n \times n$ kernel (Gram) matrix. We observe that at each iteration of the mean shift procedure (5.7), the estimate

$\bar{\mathbf{y}} = \mathbf{y} + \delta\mathbf{y}$ always lies in the column space of $\mathbf{\Phi}$. Any such point $\bar{\mathbf{y}}$ can be written as

$$\bar{\mathbf{y}} = \mathbf{\Phi}\boldsymbol{\alpha}_{\bar{\mathbf{y}}} \tag{5.10}$$

where $\boldsymbol{\alpha}_{\bar{\mathbf{y}}}$ is an n-dimensional weighting vector. The distance between two points $\mathbf{y}$ and $\mathbf{y}'$ in this space can be expressed in terms of their inner product and their respective weighting vectors

$$\begin{aligned} \|\mathbf{y} - \mathbf{y}'\|^2 &= \|\mathbf{\Phi}\boldsymbol{\alpha}_{\mathbf{y}} - \mathbf{\Phi}\boldsymbol{\alpha}_{\mathbf{y}'}\|^2 \\ &= \boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{\Phi}^\top \mathbf{\Phi}\boldsymbol{\alpha}_{\mathbf{y}} + \boldsymbol{\alpha}_{\mathbf{y}'}^\top \mathbf{\Phi}^\top \mathbf{\Phi}\boldsymbol{\alpha}_{\mathbf{y}'} - 2\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{\Phi}^\top \mathbf{\Phi}\boldsymbol{\alpha}_{\mathbf{y}'} \\ &= \boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}} + \boldsymbol{\alpha}_{\mathbf{y}'}^\top \mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}'} - 2\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}'}. \end{aligned} \tag{5.11}$$

Let $\mathbf{e}_i$ denote the i-th canonical basis vector for $\mathbb{R}^n$. Applying (5.11) to compute distances in (5.7) by using the equivalence $\boldsymbol{\phi}(\mathbf{x}_i) = \mathbf{\Phi}\mathbf{e}_i$, the mean shift algorithm iteratively updates the weighting vector $\boldsymbol{\alpha}_{\mathbf{y}}$

$$\boldsymbol{\alpha}_{\bar{\mathbf{y}}} = \frac{\sum_{i=1}^{n} \frac{\mathbf{e}_i}{h_i^{d_\phi+2}}\, g\left(\frac{\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}} + \mathbf{e}_i^\top \mathbf{K}\mathbf{e}_i - 2\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K}\mathbf{e}_i}{h_i^2}\right)}{\sum_{i=1}^{n} \frac{1}{h_i^{d_\phi+2}}\, g\left(\frac{\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K}\boldsymbol{\alpha}_{\mathbf{y}} + \mathbf{e}_i^\top \mathbf{K}\mathbf{e}_i - 2\boldsymbol{\alpha}_{\mathbf{y}}^\top \mathbf{K}\mathbf{e}_i}{h_i^2}\right)}. \tag{5.12}$$

The clustering starts on the data points on $\mathcal{H}$, therefore the initial weighting vectors are given by $\boldsymbol{\alpha}_{\mathbf{y}_i} = \mathbf{e}_i$, such that, $\mathbf{y}_i = \mathbf{\Phi}\boldsymbol{\alpha}_{\mathbf{y}_i} = \boldsymbol{\phi}(\mathbf{x}_i)$, $i = 1 \ldots n$. At convergence, the mode $\bar{\mathbf{y}}$ can be recovered using (5.10) as $\mathbf{\Phi}\bar{\boldsymbol{\alpha}}_{\mathbf{y}}$. The points converging close to the same mode are clustered together, following the original proof [23]. Since any positive semidefinite matrix $\mathbf{K}$ is a kernel for some feature space [25], the derived method implicitly applies mean shift on the feature space induced by $\mathbf{K}$. Note that under this formulation, mean shift in the input space can be implemented by simply choosing the mapping function $\boldsymbol{\phi}(\cdot)$ to be identity, i.e. $\boldsymbol{\phi}(\mathbf{x}) = \mathbf{x}$.

An important point is that the dimensionality of the feature space d_ϕ can be very large, for example it is infinite in case of the Gaussian kernel function. In such cases it may not be possible to explicitly compute the point sample density estimator (5.6) and consequently the mean shift vectors (5.7). However, since the procedure is restricted to

the subspace spanned by the feature points, the mean shift procedure can be executed using (5.12) via the kernel matrix $\mathbf{K}$. The dimensionality of the subspace can be estimated from the rank of the kernel matrix $\mathbf{K}$.

5.3 Kernel Learning Using Linear Transformations

A nonlinear mapping of the input data to a higher-dimensional kernel space often improves cluster separability. By effectively enforcing the available constraints, it is possible to transform the entire space and guide the clustering to discover the desired structure in the data.

To illustrate this intuition, we present a simple two class example from [113] in Fig. 5.2. The original data lies along the x-axis, with the blue points associated with one class and the black points with the second class (Fig. 5.2a). This data appears to have originated from three clusters. Let (x_1, x_2) be the pair of points marked in red which are constrained to be clustered together. We map the data explicitly to a two-dimensional feature space via a simple quadratic mapping $\boldsymbol{\phi}(x) = [x \;\; x^2]^\top$ (Fig. 5.2b). Although the data is linearly separable, it still appears to form three clusters. Using the must-link constraint pair, we enforce the two red points to be clustered together. The black arrow denotes the constraint vector $\boldsymbol{\phi}(x_1) - \boldsymbol{\phi}(x_2)$ and the dashed red line is its null space. By projecting the feature points to the null space of the constraint vector, the points $\boldsymbol{\phi}(x_1)$ and $\boldsymbol{\phi}(x_2)$ are collapsed to the same point, guaranteeing their association with the same mode. From Fig.5.2c, we can see that in the projected space, the data has the desired cluster structure which is consistent with its class association.

This approach, although simple, does not scale well with increasing number of constraints for a simple nonlinear mapping like above. Given m linearly independent constraint vectors in a d_ϕ-dimensional space, the dimensionality of the null space of the constraint matrix is $d_\phi - m$. This implies that if d_ϕ or more constraints are specified, *all the points* collapse to a single point and are therefore grouped together in one cluster. This problem can be alleviated if a mapping function $\boldsymbol{\phi}(\cdot)$ is chosen such that d_ϕ is very large. Since explicitly designing the mapping function $\phi(\mathbf{x})$ is not always

practical, we use a kernel function $K(\mathbf{x}, \mathbf{x}')$ to implicitly map the input data to a very high-dimensional kernel space. As we show in Section 5.3.1, the subsequent projection to the null-space of the constraint vectors can also be achieved implicitly by appropriately updating the kernel matrix. In Section 5.3.2, we further generalize the projection operation to a linear transformation that also utilizes cannot-link constraints.

5.3.1 Kernel Updates Using Orthogonal Projection

Recall the matrix $\mathbf{\Phi}$ in (5.9), obtained by mapping the input points to $\mathcal{H}$ via the nonlinear function $\boldsymbol{\phi}$. Let (j_1, j_2) be a must-link constraint pair such that $\boldsymbol{\phi}(\mathbf{x}_{j_1}) = \mathbf{\Phi}\mathbf{e}_{j_1}$ and $\boldsymbol{\phi}(\mathbf{x}_{j_2}) = \mathbf{\Phi}\mathbf{e}_{j_2}$ are to be clustered together. Given a set of m such must-link constraint pairs $\mathcal{M}$, for every $(j_1, j_2) \in \mathcal{M}$, the d_ϕ-dimensional constraint vector can be written as $\mathbf{a}_j = \mathbf{\Phi}\left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) = \mathbf{\Phi}\mathbf{z}_j$. We refer to the n-dimensional vector $\mathbf{z}_j$ as the *indicator* vector for the jth constraint. The $d_\phi \times m$ dimensional constraint matrix $\mathbf{A}$ can be obtained by column stacking all the m constraint vectors, i.e., $\mathbf{A} = \mathbf{\Phi}\mathbf{Z}$, where $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_m]$ is the $n \times m$ matrix of indicator vectors. Similar to the example in Fig. 5.2, we impose the constraints by projecting the matrix $\mathbf{\Phi}$ to the null space of $\mathbf{A}$ using the the projection matrix

$$\mathbf{P} = \mathbf{I}_{d_\phi} - \mathbf{A}\left(\mathbf{A}^\top\mathbf{A}\right)^+\mathbf{A}^\top \tag{5.13}$$

where $\mathbf{I}_{d_\phi}$ is the d_ϕ-dimensional identity matrix and '+' denotes the matrix pseudoinverse operation. Let $\mathbf{S} = \mathbf{A}^\top\mathbf{A}$ be the $m \times m$ scaling matrix. The matrix $\mathbf{S}$ can be computed using the indicator vectors and the initial kernel matrix $\mathbf{K}$ without knowing the mapping $\boldsymbol{\phi}$ as

$$\mathbf{S} = \mathbf{A}^\top\mathbf{A} = \mathbf{Z}^\top\mathbf{\Phi}^\top\mathbf{\Phi}\mathbf{Z} = \mathbf{Z}^\top\mathbf{K}\mathbf{Z}. \tag{5.14}$$

Given the constraint set, the new mapping function $\widehat{\boldsymbol{\phi}}(\mathbf{x})$ is computed as $\widehat{\boldsymbol{\phi}}(\mathbf{x}) = \mathbf{P}\boldsymbol{\phi}(\mathbf{x})$. Since all the constraints in $\mathcal{M}$ are satisfied in the projected subspace, we have

$$||\mathbf{P}\mathbf{\Phi}\mathbf{Z}||_F^2 = 0 \tag{5.15}$$

where $||\cdot||_F$ denotes the Frobenius norm. The initial kernel function (5.8) correspondingly transforms to the projected kernel function $\widehat{K}(\mathbf{x},\mathbf{x}') = \widehat{\boldsymbol{\phi}}(\mathbf{x})^\top \widehat{\boldsymbol{\phi}}(\mathbf{x}')$. Using the projection matrix $\mathbf{P}$, it can be rewritten in terms of the initial kernel function $K(\mathbf{x},\mathbf{x}')$ and the constraint matrix $\mathbf{A}$

$$\begin{aligned}\widehat{K}(\mathbf{x},\mathbf{x}') &= \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{P}^\top \mathbf{P} \boldsymbol{\phi}(\mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{P} \boldsymbol{\phi}(\mathbf{x}') \\ &= \boldsymbol{\phi}(\mathbf{x})^\top (\mathbf{I}_{d_\phi} - \mathbf{A}\mathbf{S}^+\mathbf{A}^\top)\boldsymbol{\phi}(\mathbf{x}') \\ &= K(\mathbf{x},\mathbf{x}') - \boldsymbol{\phi}(\mathbf{x})^\top \mathbf{A}\mathbf{S}^+\mathbf{A}^\top \boldsymbol{\phi}(\mathbf{x}'). \end{aligned} \tag{5.16}$$

Note that the identity $\mathbf{P}^\top\mathbf{P} = \mathbf{P}$ follows from the fact that $\mathbf{P}$ is a symmetric projection matrix. The m-dimensional vector $\mathbf{A}^\top\boldsymbol{\phi}(\mathbf{x})$ can also be written in terms of the initial kernel function as

$$\begin{aligned}\mathbf{A}^\top \boldsymbol{\phi}(\mathbf{x}) &= \mathbf{Z}^\top \boldsymbol{\Phi}^\top \boldsymbol{\phi}(\mathbf{x}) \\ &= \mathbf{Z}^\top \left[K(\mathbf{x}_1,\mathbf{x}),\ldots,K(\mathbf{x}_n,\mathbf{x})\right]^\top. \end{aligned} \tag{5.17}$$

Let the vector $\mathbf{k}_\mathbf{x} = \left[K(\mathbf{x}_1,\mathbf{x}),\ldots,K(\mathbf{x}_n,\mathbf{x})\right]^\top$. From (5.16), the projected kernel function can be written as

$$\widehat{K}(\mathbf{x},\mathbf{x}') = K(\mathbf{x},\mathbf{x}') - \mathbf{k}_\mathbf{x}^\top \mathbf{Z}\mathbf{S}^+\mathbf{Z}^\top \mathbf{k}_{\mathbf{x}'} \tag{5.18}$$

The projected kernel matrix can be directly computed as

$$\widehat{\mathbf{K}} = \mathbf{K} - \mathbf{K}\mathbf{Z}\mathbf{S}^+\mathbf{Z}^\top\mathbf{K} \tag{5.19}$$

where the symmetric $n \times n$ initial kernel matrix $\mathbf{K}$ has rank $r \leq n$. The rank of the matrix $\mathbf{S}$ is equal to the number of linearly independent constraints, and the rank of the projected kernel matrix $\widehat{\mathbf{K}}$ is $r - \text{rank } \mathbf{S}$.

5.3.2 Kernel Updates Using Linear Transformation

By projecting the feature points to the null space of the constraint vector $\mathbf{a}$, their components along the $\mathbf{a}$ are fully eliminated. This operation guarantees that the two points belong to the same cluster. As proposed earlier, by appropriately choosing the kernel function $K(\cdot)$ (such that d_ϕ is very large), we can make the initial kernel matrix $\mathbf{K}$ full-rank. However, a sufficiently large number of such linearly independent constraints could still result in a projected space with dimensionality that is too small for meaningful clustering.

From a clustering perspective, it might suffice to bring the two must-link constraint points sufficiently close to each other. This can be achieved through a linear transformation of the kernel space that *only* scales down the component along the constraint vector. Such a linear transformation would preserve the rank of the kernel matrix and is potentially capable of handling a very large number of must-link constraints. A similar transformation that *increases* the distance between two points also enables us to incorporate cannot-link constraints.

Given a constraint vector $\mathbf{a} = \mathbf{\Phi z}$, a symmetric transformation matrix of the form

$$\mathbf{T} = \mathbf{I}_{d_\phi} - s\left(\mathbf{a}\mathbf{a}^\top\right) \tag{5.20}$$

allows us to control the scaling along the vector $\mathbf{a}$ using the scaling factor s. When $s = \frac{1}{\mathbf{a}^\top\mathbf{a}}$ the transformation becomes a projection to the null space of $\mathbf{a}$. The transformation decreases distances along $\mathbf{a}$ for $0 < s < \frac{2}{\mathbf{a}^\top\mathbf{a}}$, while it is increased for $s < 0$ or $s > \frac{2}{\mathbf{a}^\top\mathbf{a}}$.

We can set a target distance $d > 0$ for the constraint point pair by applying an appropriate transformation $\mathbf{T}$. The constraint equation can be written as

$$||\mathbf{T\Phi z}||_F^2 = \mathbf{z}^\top\widehat{\mathbf{K}}\mathbf{z} = d \tag{5.21}$$

where $\widehat{\mathbf{K}} = \mathbf{\Phi}^\top\mathbf{T}^\top\mathbf{T\Phi} = \mathbf{\Phi}^\top\mathbf{T}^2\mathbf{\Phi}$ is the corresponding transformed kernel matrix. To handle must-link constraints, d should be small, while it should be large for cannot-link constraint pairs. Using the specified d, we can compute s and therefore the transformed

kernel matrix

$$\widehat{\mathbf{K}} = \mathbf{\Phi}^\top \left(\mathbf{I}_{d_\phi} - s\,\mathbf{a}\mathbf{a}^\top\right)^2 \mathbf{\Phi}. \tag{5.22}$$

Substituting $\mathbf{a} = \mathbf{\Phi}\mathbf{z}$ the expression for $\widehat{\mathbf{K}}$ is

$$\widehat{\mathbf{K}} = \mathbf{K} - 2s\,\mathbf{K}\mathbf{z}\mathbf{z}^\top\mathbf{K} + s^2\left(\mathbf{z}^\top\mathbf{K}\mathbf{z}\right)\mathbf{K}\mathbf{z}\mathbf{z}^\top\mathbf{K}. \tag{5.23}$$

From (5.21), we can solve for s

$$s = \frac{1}{p}\left(1 \pm \sqrt{\frac{d}{p}}\right) \quad \text{where} \quad p = \mathbf{z}^\top\mathbf{K}\mathbf{z} > 0 \tag{5.24}$$

and the choice of s does not affect the following kernel update[1] such that the constraint (5.21) is satisfied

$$\widehat{\mathbf{K}} = \mathbf{K} + \beta\mathbf{K}\mathbf{z}\mathbf{z}^\top\mathbf{K}, \qquad \beta = \left(\frac{d}{p^2} - \frac{1}{p}\right). \tag{5.25}$$

The case when $p = 0$ implies that the constraint vector is a zero vector and $\mathbf{T} = \mathbf{I}_{d_\phi}$ and a value of $\beta = 0$ should be used.

When multiple must-link and cannot-link constraints are available, the kernel matrix can be updated iteratively for each constraint by computing the update parameter β_j using corresponding d_j and p_j. However, by enforcing the distance between constraint pairs to be *exactly* d_j, the linear transformation imposes *hard* constraints for learning the kernel matrix which could potentially have two adverse consequences:

1. The hard constraints could make the algorithm difficult (or even impossible) to converge, since previously satisfied constraints could easily get violated during subsequent updates of the kernel matrix.

2. Even when the constraints are non-conflicting in nature, in the absence of any relaxation, the method becomes sensitive to labeling errors. This is illustrated through the following example.

The input data consists of points along five concentric circles as shown in Fig.5.3a. Each

[1]This update rule is equivalent to minimizing the $\log\det$ divergence (5.29) for a single equality constraint using Bregman projections (5.31).

cluster comprises of 150 noisy points along a circle. Four labeled points per class (shown by square markers) are used to generate a set of $\binom{4}{2} \times 5 = 30$ must-link constraints. The initial kernel matrix $\mathbf{K}$ is computed using a Gaussian kernel with $\sigma = 5$, and updated by imposing the provided constraints (5.19). The updated kernel matrix $\widehat{\mathbf{K}}$ is used for kernel mean shift clustering (Section 5.2.2) and the corresponding results are shown in Fig. 5.3b. To test the performance of the method under labeling errors, we also add one mislabeled must-link constraint (shown in black line). The clustering performance deteriorates drastically with just one mislabeled constraint as shown in Fig. 5.3c.

In order to overcome these limitations, the learning algorithm must accommodate for systematic relaxation of constraints. This can be achieved by observing that the kernel update in (5.25) is equivalent to minimizing the log det divergence between $\widehat{\mathbf{K}}$ and $\mathbf{K}$ subject to constraint (5.21) [63, Sec. 5.1.1]. In the following section, we formulate the kernel learning algorithm into a log det minimization problem with *soft* constraints.

5.4 Kernel Learning Using Bregman Divergence

Bregman divergences have been studied in the context of clustering, matrix nearness and metric and kernel learning [55, 63, 5]. We briefly discuss the log det Bregman divergence and its properties in Section 5.4.1. We formulate a constrained log det divergence minimization problem similar to [55] in Section 5.4.2.

5.4.1 The LogDet Bregman Divergence

The Bregman divergence [11] between real, symmetric $n \times n$ matrices $\mathbf{X}$ and $\mathbf{Y}$ is a scalar function given by

$$D_{\varphi}(\mathbf{X}, \mathbf{Y}) = \varphi(\mathbf{X}) - \varphi(\mathbf{Y}) - \mathrm{tr}\Big(\nabla\varphi\,(\mathbf{Y})^{\top}(\mathbf{X} - \mathbf{Y})\Big) \tag{5.26}$$

where φ is a strictly convex function and ∇ denotes the gradient operator. For $\varphi(\mathbf{X}) = -\log\left(\det(\mathbf{X})\right)$, the resulting divergence is called the log det divergence

$$D_{ld}\left(\mathbf{X}, \mathbf{Y}\right) = \mathrm{tr}\left(\mathbf{X}\mathbf{Y}^{-1}\right) - \log\det\left(\mathbf{X}\mathbf{Y}^{-1}\right) - n \tag{5.27}$$

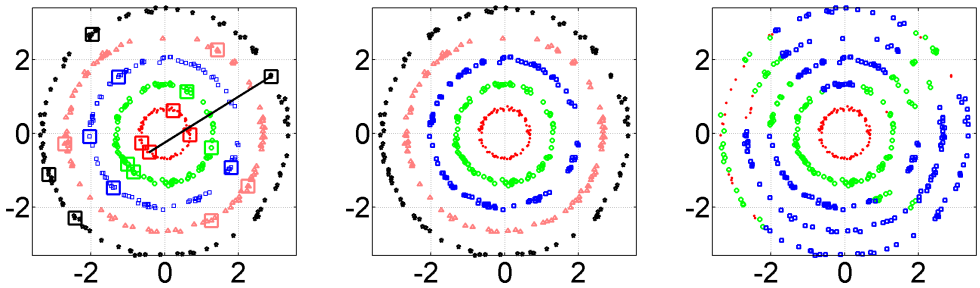


Figure 5.3: Five concentric circles. Kernel learning without relaxation (a) Input data. The square markers indicate the data points used to generate pairwise constraints. The black line shows the only mislabeled similarity constraint used. (b) Clustering results when only the correctly labeled similarity constraints were used. (c) Clustering results when the mislabeled constraint was also used.

and is defined when $\mathbf{X}, \mathbf{Y}$ are positive definite. In [63], this definition was extended to rank deficient (positive semidefinite) matrices by restricting the convex function to the range spaces of the matrices. For positive semidefinite matrices $\mathbf{X}$ and $\mathbf{Y}$, both having rank $r \leq n$ and singular value decomposition $\mathbf{X} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^\top$ and $\mathbf{Y} = \mathbf{U}\mathbf{\Theta}\mathbf{U}^\top$, the log det divergence is defined as

$$D_{ld}\left(\mathbf{X}, \mathbf{Y}\right) = \sum_{i,j \leq r} \left(\mathbf{v}_i^\top \mathbf{u}_j\right)^2 \left(\frac{\lambda_i}{\theta_j} - \log \frac{\lambda_i}{\theta_j} - 1\right). \tag{5.28}$$

Moreover, the log det divergence, like any Bregman divergence, is convex with respect to its first argument $\mathbf{X}$ [5]. This property is useful in formulating the kernel learning as a convex minimization problem in the presence of multiple constraints.

5.4.2 Kernel Learning with LogDet Divergences

Let $\mathcal{M}$ and $\mathcal{C}$ denote the sets of m must-link and c cannot-link pairs respectively, such that $m + c = n_c$. Let d_m and d_c be the target squared distance thresholds for must-link and cannot-link constraints respectively. The problem of learning a kernel matrix using linear transformations for multiple constraints, discussed in Section 5.3.2, can be

equivalently formulated as the following constrained log det minimization problem

$$\begin{aligned} \min_{\widehat{\mathbf{K}}} \quad & D_{ld}\left(\widehat{\mathbf{K}}, \mathbf{K}\right) & & (5.29) \\ \text{s.t.} \quad & \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right)^\top \widehat{\mathbf{K}} \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) = d_m & \forall (j_1, j_2) \in \mathcal{M} \\ & \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right)^\top \widehat{\mathbf{K}} \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) = d_c & \forall (j_1, j_2) \in \mathcal{C}. \end{aligned}$$

The final kernel matrix $\widehat{\mathbf{K}}$ is obtained by iteratively updating the initial kernel matrix $\mathbf{K}$.

In order to permit relaxation of constraints, we rewrite the learning problem using a *soft margin* formulation similar to [55], where each constraint pair (j_1, j_2) is associated with a slack variable $\widehat{\xi}_j,\ j = 1, \ldots, n_c$

$$\begin{aligned} \min_{\widehat{\mathbf{K}}, \widehat{\boldsymbol{\xi}}} \quad & D_{ld}\left(\widehat{\mathbf{K}}, \mathbf{K}\right) + \gamma D_{ld}\left(\operatorname{diag}\left(\widehat{\boldsymbol{\xi}}\right), \operatorname{diag}\left(\boldsymbol{\xi}\right)\right) & & (5.30) \\ \text{s.t.} \quad & \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right)^\top \widehat{\mathbf{K}} \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) \leq \widehat{\xi}_j & \forall (j_1, j_2) \in \mathcal{M} \\ & \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right)^\top \widehat{\mathbf{K}} \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) \geq \widehat{\xi}_j & \forall (j_1, j_2) \in \mathcal{C}. \end{aligned}$$

The n_c-dimensional vector $\widehat{\boldsymbol{\xi}}$ is the vector of slack variables and $\boldsymbol{\xi}$ is the vector of target distance thresholds d_m and d_c. The regularization parameter γ controls the trade-off between fidelity to the original kernel and the training error. Note that by changing the equality constraints to inequality constraints, the algorithm allows must-link pairs to be closer and cannot-link pairs to be farther than their corresponding distance thresholds.

The optimization problem in (5.30) is solved using the method of Bregman projections [11]. A Bregman projection (not necessarily orthogonal) is performed to update the current matrix, such that the updated matrix satisfies that constraint. In each iteration, it is possible that the current update violates a previously satisfied constraint. Since the problem in (5.30) is convex [63], the algorithm converges to the global minimum after repeatedly updating the kernel matrix for each constraint in the set $\mathcal{M} \cup \mathcal{C}$.

For the log det divergence, the Bregman projection that minimizes the objective function in (5.30) for a given constraint $(j_1, j_2) \in \mathcal{M} \cup \mathcal{C}$, can be written as derived in

[63]

$$\widehat{\mathbf{K}}_{t+1} = \widehat{\mathbf{K}}_t + \beta_t \widehat{\mathbf{K}}_t \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right) \left(\mathbf{e}_{j_1} - \mathbf{e}_{j_2}\right)^\top \widehat{\mathbf{K}}_t. \tag{5.31}$$

For the tth iteration the parameter β_t is computed in closed form and is explained in Section 5.5 along with the complete low-rank kernel learning algorithm. The algorithm converges when β_t approaches zero for all $(j_1, j_2) \in \mathcal{M} \cup \mathcal{C}$ with the final learned kernel matrix $\widehat{\mathbf{K}} = \widehat{\mathbf{\Phi}}^\top \widehat{\mathbf{\Phi}}$.

Using $\widehat{\mathbf{K}}$, the kernel function that defines the inner product in the corresponding transformed kernel space can be written as

$$\widehat{K}\left(\mathbf{x}, \mathbf{y}\right) = K\left(\mathbf{x}, \mathbf{y}\right) + \mathbf{k}_{\mathbf{x}}^\top \left(\mathbf{K}^+ \left(\widehat{\mathbf{K}} - \mathbf{K}\right) \mathbf{K}^+\right) \mathbf{k}_{\mathbf{y}} \tag{5.32}$$

where $K(\cdot)$ is the scalar initial kernel function (5.8), and the vectors $\mathbf{k}_{\mathbf{x}}$ and $\mathbf{k}_{\mathbf{y}}$ are $[K(\mathbf{x}, \mathbf{x}_1), \ldots, K(\mathbf{x}, \mathbf{x}_n)]^\top$ and $[K(\mathbf{y}, \mathbf{x}_1), \ldots, K(\mathbf{y}, \mathbf{x}_n)]^\top$ respectively [55]. The points $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ could be *out of sample* points, i.e., points that are not in the sample set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ used to learn $\widehat{\mathbf{K}}$. Note that (5.18) also generalizes the inner product in the projected kernel space to out of sample points.

5.5 Low Rank Kernel Updates

The kernel update presented in Jain et al. [55] for solving (5.30) involves the full $n \times n$ kernel matrix. Since in all our applications the kernel matrix is always low-rank, we adapt the low-rank kernel learning algorithm in [63] to solve (5.30).

The initial kernel matrix $\mathbf{K}$ with rank $r \leq n$, can be decomposed as $\mathbf{K} = \mathbf{G}\mathbf{G}^\top$, where $\mathbf{G}$ is the $n \times r$ square root matrix. If $r \ll n$, the complexity of the learning algorithm can be significantly reduced by applying fast *Cholesky* updates [63, Section 5.1.3]. The update equation (5.31) can be rewritten as

$$\widehat{\mathbf{K}}_{t+1} = \widehat{\mathbf{G}}_t \Big(\mathbf{I}_r + \beta_t \widehat{\mathbf{G}}_t^\top (\mathbf{e}_{j_1} - \mathbf{e}_{j_2})(\mathbf{e}_{j_1} - \mathbf{e}_{j_2})^\top \widehat{\mathbf{G}}_t\Big) \widehat{\mathbf{G}}_t^\top \tag{5.33}$$

where $\widehat{\mathbf{G}}_t$ is the $n \times r$ updated square root matrix in the tth iteration. The term within

Input:
$\mathbf{G}$ - $n \times r$ initial square root matrix $\mathbf{K} = \mathbf{G}\mathbf{G}^\top$
$\mathcal{M}, \mathcal{C}$ - similarity and dissimilarity constraint sets
d_m, d_c - the distance thresholds.
γ - the trade-off parameter
Output:
$\widehat{\mathbf{G}}$ - the $n \times r$ learned square root matrix.

- Initialize $t = 0$, $\mathbf{B}_0 = \mathbf{I}_{r\times r}$, $\lambda_j = 0 \quad j = 1, \ldots, n_c$
- Initialize slack variables
  $\widehat{\xi}_j = \xi_j = d_m$ for $(j_1, j_2) \in \mathcal{M}$
  $\widehat{\xi}_j = \xi_j = d_c$ for $(j_1, j_2) \in \mathcal{C}$
- Repeat until convergence
  - Pick a constraint $(j_1, j_2) \in \mathcal{M}$ or $\mathcal{C}$.
  - $\delta = 1$ if $(j_1, j_2) \in \mathcal{M}$ or -1 if $(j_1, j_2) \in \mathcal{C}$.
  - $\mathbf{w} = \mathbf{B}_t \left[\mathbf{G}(j_1, \cdot) - \mathbf{G}(j_2, \cdot)\right]^\top$
  - $\alpha = \min\left(\lambda_j, \frac{\delta\gamma}{\gamma+1}\left(\frac{1}{||\mathbf{w}||_2^2} - \frac{1}{\widehat{\xi}_j}\right)\right)$.
  - $\widehat{\xi}_j \leftarrow \gamma\widehat{\xi}_j / \left(\gamma + \delta\alpha\widehat{\xi}_j\right)$.
  - $\lambda_j \leftarrow \lambda_j - \alpha$.
  - $\beta = \delta\alpha / \left(1 - \delta\alpha||\mathbf{w}||_2^2\right)$.
  - Factorize $\left(\mathbf{I} + \beta\mathbf{w}\mathbf{w}^\top\right)$ into $\mathbf{L}_t\mathbf{L}_t^\top$ using Cholesky decomposition. Update $\mathbf{B}_{t+1} = \mathbf{B}_t\mathbf{L}_t$.
  - $t \leftarrow t + 1$.
- return $\widehat{\mathbf{G}} = \mathbf{G}\mathbf{B}_{t_{conv}}$

Figure 5.4: Low rank kernel learning algorithm.

parentheses is a rank one perturbation of the $r \times r$ identity matrix with the Cholesky decomposition $\mathbf{L}_t\mathbf{L}_t^\top$. The matrix $\widehat{\mathbf{G}}_t$ is initialized as $\widehat{\mathbf{G}}_0 = \mathbf{G}$ and it can be seen that $\widehat{\mathbf{G}}_{t+1} = \widehat{\mathbf{G}}_t\mathbf{L}_t = \mathbf{G}\mathbf{L}_1 \cdots \mathbf{L}_t$. At convergence, we have $\widehat{\mathbf{G}} = \mathbf{G}\prod_i \mathbf{L}_i,\ i = 1, \ldots, t_{conv}$ and the corresponding learned kernel is $\widehat{\mathbf{K}} = \widehat{\mathbf{G}}\widehat{\mathbf{G}}^\top$.

The algorithm for low-rank kernel learning is summarized in Fig. 5.4. The input is the $n \times r$ square root matrix $\mathbf{G}$ and the sets of pairwise constraints $\mathcal{M}$ and $\mathcal{C}$ along with their corresponding distance thresholds d_m and d_c.

Let $\mathbf{G}(i, \cdot)$ represent the ith row of the square root matrix $\mathbf{G}$. In the feature space induced by $\widehat{\mathbf{K}}_t$, the squared distance between a pair of constraint points (j_1, j_2) is $\mathbf{w}^\top\mathbf{w}$ computed using the $\mathbf{G}$ and the accumulator matrix $\mathbf{B}_t = \prod_i \mathbf{L}_i,\ i = 1, \ldots, t-1$.

The parameter α is used to relax the slack variables $\widehat{\xi}_j$ and update β. The n_c-dimensional vector $\boldsymbol{\lambda}$ is initialized to zero and updated in each iteration such that $\lambda_j \geq 0$ for a constraint $(j_1, j_2) \in \mathcal{M} \cup \mathcal{C}$. The algorithm converges when $||\boldsymbol{\lambda}_t - \boldsymbol{\lambda}_{t-1}||$ is below a small threshold set to 0.01.

5.6 Semi-supervised Kernel Mean-Shift Clustering Algorithm

We present the complete algorithm for semi-supervised kernel mean shift clustering (SKMS) in this section. Fig. 5.5 shows a block diagram for the overall algorithm. We explain each of the modules using two examples: *Olympic circles*, a synthetic data set and a 1000 sample subset of *USPS digits*, a real data set with images of handwritten digits. In Section 5.6.1, we propose a method to select the scale parameter σ for the initial Gaussian kernel function using the sets $\mathcal{M}, \mathcal{C}$ and target distances d_m, d_c. We show the speed-up achieved by performing the low-rank kernel matrix updates (as opposed to updating the full kernel matrix) in Section 5.6.2. For mean shift clustering, we present a strategy to automatically select the bandwidth parameter using the pairwise must-link constraints in Section 5.6.3. Finally, in Section 5.6.4, we discuss the selection of the trade-off parameter γ.

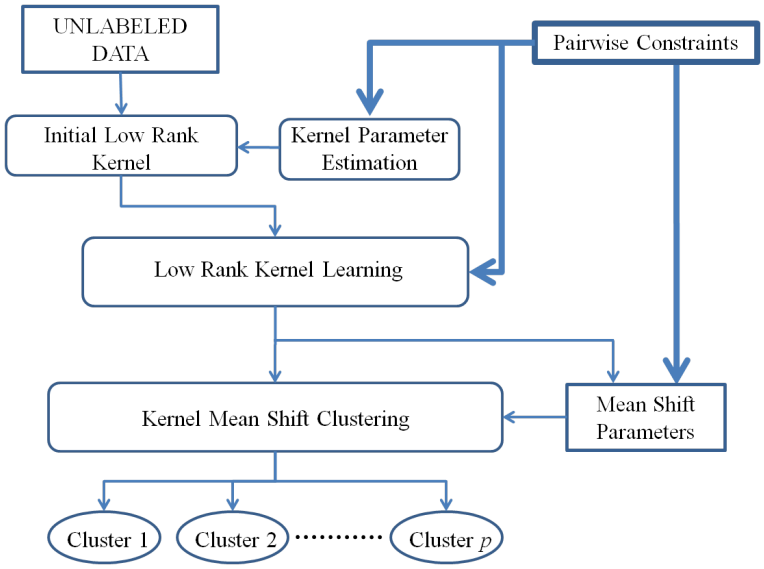


Figure 5.5: Block diagram describing the semi-supervised kernel mean shift clustering algorithm. The bold boxes indicate the user input to the algorithm.

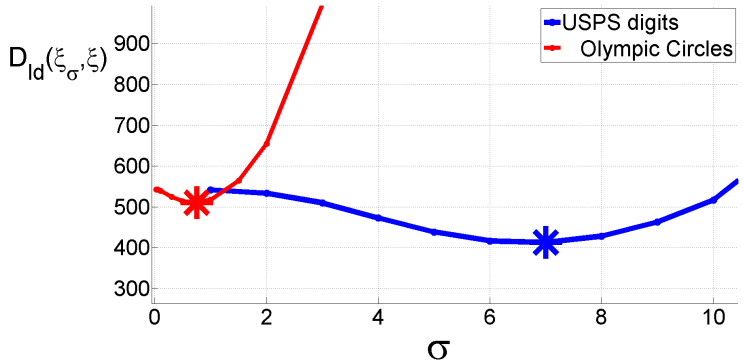


Figure 5.6: Selecting the scale parameter $\hat{\sigma}$ that minimizes $D_{ld}(\boldsymbol{\xi}, \boldsymbol{\xi}_\sigma)$ using grid search. Selected σ values: Olympic circles, $\hat{\sigma} = 0.75$; USPS digits, $\hat{\sigma} = 7$.

5.6.1 Initial Parameter Selection

Given two points $\mathbf{x}_i, \mathbf{x}_j$, the Gaussian kernel function is given by

$$K_\sigma(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) \in [0, 1] \tag{5.34}$$

where σ is the scale parameter. From (5.34) and (5.11) it is evident that pairwise distances between sample points in the feature space induced by $K_\sigma(\cdot)$ lies in the interval $[0, 2]$. This provides us with an effective way of setting up the target distances $d_m = \min(d_1, 0.05)$ and $d_c = \max(d_{99}, 1.95)$, where d_1 and d_{99} are the 1st and 99th percentile of distances between *all* pairs of points in the kernel space. We select the scale parameter σ such that, in the initial kernel space, distances between must-link points are small while those between cannot-link points are large. This results in good regularization and faster convergence of the learning algorithm.

Recall that $\boldsymbol{\xi}$ is the n_c-dimensional vector of target squared distances between the constraint pairs

$$\xi_j = \begin{cases} d_m & \forall (j_1, j_2) \in \mathcal{M} \\ d_c & \forall (j_1, j_2) \in \mathcal{C}. \end{cases} \tag{5.35}$$

Let $\boldsymbol{\xi}_\sigma$ be the vector of distances computed for the n_c constraint pairs using the kernel matrix $\mathbf{K}_\sigma$. The scale parameter that minimizes the $\log\det$ divergence between the

Table 5.1: Comparison of execution times of kernel learning using the low-rank updates (5.33) vs. full matrix updates (5.31).

Data set	**Size(K)** (n)	rank $\mathbf{K}$ (r)	**Time** (s)	
			low-rank	full matrix
Olympic Circles	1500	58	0.92	323.32
USPS Digits	1000	499	68.19	151.09

vectors $\boldsymbol{\xi}$ and $\boldsymbol{\xi}_\sigma$ is

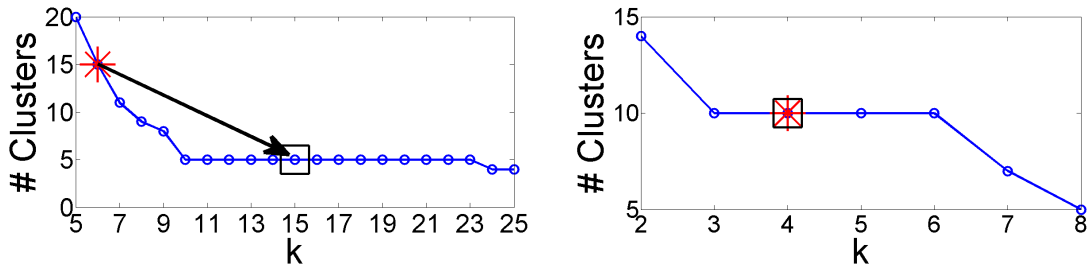
$$\hat{\sigma} = \arg\min_{\sigma} \quad D_{ld}\left(\operatorname{diag}\left(\boldsymbol{\xi}\right), \operatorname{diag}\left(\boldsymbol{\xi}_\sigma\right)\right) \tag{5.36}$$

and the corresponding kernel matrix is $\mathbf{K}_{\hat{\sigma}}$. The kernel learning is insensitive to small variations in σ, so it is sufficient to do a grid search over a range of σ values. Fig. 5.6 shows the plot of the objective function in (5.36) against different values of σ for the *Olympic circles* data set and the *USPS digits* data set.

5.6.2 Low Rank Kernel Learning

When the initial kernel matrix has rank $r \leq n$, the $n \times n$ matrix updates (5.31) can be modified to achieve a significant computational speed-up [63]. We learn a kernel matrix for clustering the two example data sets: Olympic circles (5 classes, $n = 1500$) and USPS digits (10 classes, $n = 1000$). The must-link constraints are generated using 15 labeled points from each class: 525 for Olympic circles and 1050 for USPS digits, while an equal number of cannot-link constraints is used.

The $n \times n$ initial kernel matrix $\mathbf{K}_{\hat{\sigma}}$ is computed as described in the previous section. Using singular value decomposition, we compute an $n \times n$ low-rank kernel matrix $\mathbf{K}$ such that $\operatorname{rank} \mathbf{K} = r \leq n$ and $\frac{\|\mathbf{K}\|_F}{\|\mathbf{K}_{\hat{\sigma}}\|_F} \geq 0.99$. Given pairwise constraints and the initial matrix $\mathbf{K}$, the kernel learning is performed using both, full matrix updates (5.31) and low-rank kernel updates (5.33). Table 5.1 shows significant improvements in computational time when the low-rank learning algorithm is used (see Appendix for details). We



(a) (b)

Figure 5.7: Selecting the mean shift bandwidth parameter k, given five labeled points per class. (a) Olympic circles. Number of recovered clusters is sensitive at the median based estimate $k = 6$, but not at $k = 15$ (see text). (b) USPS digits. Number of recovered clusters is not sensitive at the median based estimate $k = 4$. The asterisk indicates the median based estimate, while the square marker shows a good estimate of k.

also observed that in all our experiments, clustering performance did not deteriorate significantly when the low-rank approximation of the kernel matrix was used.

5.6.3 Setting the Mean Shift Parameters

For kernel mean shift clustering, we define the bandwidth for each point as the distance to its kth nearest neighbor. In general, clustering is an interactive process where the bandwidth parameter for mean shift is provided by the user, or is estimated based on the desired number of clusters. In this section we propose an automatic recommendation method for the bandwidth parameter k by using *only* the must-link constraint pairs.

We build upon the intuition that in the transformed kernel space, the neighborhood of a must-link pair comprises of points similar to the constraint points. Given the jth constraint pair $(j_1, j_2) \in \mathcal{M}$, we compute the pairwise distances in the transformed kernel space between the first constraint point $\widehat{\mathbf{\Phi}}\mathbf{e}_{j_1}$ and all other feature points as $d_i = (\mathbf{e}_{j_1} - \mathbf{e}_i)^\top \widehat{\mathbf{K}}(\mathbf{e}_{j_1} - \mathbf{e}_i),\ i = 1, \ldots, n,\ i \neq j_1$. These points are then sorted in the increasing order of d_i. The bandwidth parameter k_j for the jth constraint corresponds to the index of j_2 in this sorted list. Therefore, the point $\widehat{\mathbf{\Phi}}\mathbf{e}_{j_2}$ is the k_j^{th} neighbor of $\widehat{\mathbf{\Phi}}\mathbf{e}_{j_1}$. Finally, the value of k is selected as the median over the set $\{k_j, j = 1, \ldots, m\}$.

For a correct choice of k, we expect the performance of mean shift to be insensitive to small changes in the value of k. In Fig. 5.7, we plot the number of clusters recovered by kernel mean shift as the bandwidth parameter is varied. For learning the kernel,

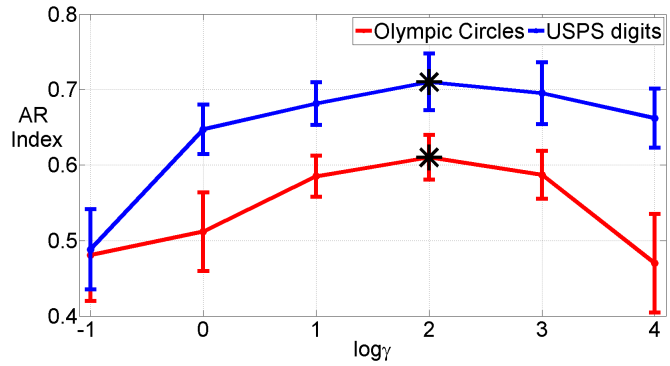


Figure 5.8: Selecting the trade-off parameter γ. The AR index vs $\log \gamma$ for 50 cross-validation runs. The asterisks mark the selected value of $\gamma = 100$.

we used 5 points from each class to generate must-link constraints: 50 for the Olympic circles and 100 for USPS digits. An equal number of cannot-link constraints is used. Fig. 5.7a shows the plot for Olympic circles (5 classes), where the median based value ($k = 6$) underestimates the bandwidth parameter. A good choice is $k = 15$, which lies in the range $(10 - 23)$ where the clustering output is insensitive to changes in k.

As seen in Fig. 5.7b, in case of USPS digits (10 classes), the number of clusters recovered by mean shift is insensitive to the median based estimate ($k = 4$). For all other data sets we used in our experiments, the median based estimates produced good clustering output. Therefore, with the exception of Olympic circles, we used the bandwidth parameter obtained using the median based approach. However, for completeness, the choice of k should be verified, and corrected if necessary, by analyzing the sensitivity of the clustering output to small perturbations in k.

For computational efficiency, we run the mean shift algorithm in a lower dimensional subspace spanned by the singular vectors corresponding to the 25 largest singular values of $\widehat{\mathbf{K}}$. For all the experiments in Section 5.7, a 25-dimensional representation of the kernel matrix was large enough to represent the data in the kernel space.

5.6.4 Selecting the Trade-off Parameter

The trade-off parameter γ is used to weight the objective function for the kernel learning. We select γ by performing a two-fold cross-validation over different values of γ

and the clustering performance is evaluated using the scalar measure Adjusted Rand (AR) index [51]. The kernel matrix is learned using half of the data with 15 points used to generate the pairwise constraints, 525 must-link constraints for the Olympic circles and 1050 for the USPS digits. An equal number of cannot-link constraint pairs is also generated.

Each cross-validation step involves learning the kernel matrix with the specified γ and clustering the testing subset using the kernel mean shift algorithm and the transformed kernel function(5.32). Fig. 5.8 shows the average AR index plotted against log γ for the two data sets. In both the examples an optimum value of $\gamma = 100$ was obtained. In general, this value may not be optimum for other applications. However, in all our experiments in Section 5.7, we use $\gamma = 100$, since we obtained similar curves with small variations in the AR index in the vicinity of $\gamma = 100$.

5.7 Experiments

We show the performance of semi-supervised kernel mean shift algorithm (SKMS) on two synthetic examples and four real-world examples. We also compare our method with two state-of-the-art methods: the semi-supervised kernel k-means (SSKK) [62] and the constrained spectral clustering (E2CP) [74]. In the past, the superior performance of E2CP over other recent methods has been demonstrated. Please see [74] for more details. In addition to this, we also compare SKMS with the kernelized k-means algorithm (Kkm). By providing both SKMS and Kkm the same learned kernel matrix, we compare the clustering performance of mean shift with that of k-means. Note that, SSKK uses a different strategy to learn the kernel matrix using pairwise constraints. With the exception of SKMS, all the methods require the user to provide the correct number of clusters.

Comparison metric. The clustering performance of different algorithms is compared using the Adjusted Rand (AR) index [51]. It is an adaptation of the rand index that penalizes random cluster assignments, while measuring the agreement between the clustering output and the *ground truth* labels. Using the contingency table t, the AR

index can be computed as [20]

$$
\begin{aligned}
AR &= \frac{RI - \mathbb{E}(RI)}{max\_RI - \mathbb{E}(RI)} \\
&= \frac{N\,(a+d) - [(a+b)\,(a+c) + (d+b)\,(d+c)]}{N^2 - [(a+b)\,(a+c) + (d+b)\,(d+c)]} \\
a &= \sum_{r=1}^{R}\sum_{c=1}^{C}\binom{t_{rc}}{2}, \qquad b = \sum_{r=1}^{R}\binom{t_{r\cdot}}{2} - a \\
c &= \sum_{c=1}^{C}\binom{t_{\cdot c}}{2} - a, \qquad d = \binom{n}{2} - a - b - c
\end{aligned}
\tag{5.37}
$$

where $t_{rc} = t(r,c)$, $t_{r\cdot} = \sum_c t(r,c)$ and $t_{\cdot c} = \sum_r t(r,c)$ and N is the sum of all entries in the contingency table, and $\mathbb{E}(\cdot)$ is the expectation operator. The AR index is a scalar and takes values between zero and one, with perfect clustering yielding a value of one.

Experimental setup. To generate pairwise constraints, we randomly select b labeled points from each class. All possible must-link constraints are generated for each class such that $m = \binom{b}{2}$ and a subset of all cannot-link constraint pairs is selected at random such that $c{=}m{=}\frac{n_c}{2}$. For each experimental setting, we average the clustering results over 50 independent runs, each with randomly selected pairwise constraints.

The initial kernel matrix is computed using a Gaussian kernel (5.34) for all the methods. We hand-picked the scale parameter σ for SSKK and E2CP from a wide range of values such that their final clustering performance on each data set was maximized. For SKMS and Kkm, the values of σ and the target distances d_m and d_c are estimated as described in Section 5.6.1. Finally, the bandwidth parameter for mean shift k was estimated as described in Section 5.6.3.

For each experiment, we specify the scale parameter σ we used for SSKK and E2CP, while for SKMS and Kkm, we report the value of σ chosen most often using (5.36). We also list the range of k, the bandwidth parameter for mean shift for each application. For the $\log\det$ divergence based kernel learning, we set the maximum number of iterations to 100000 and the trade-off parameter γ to 100.

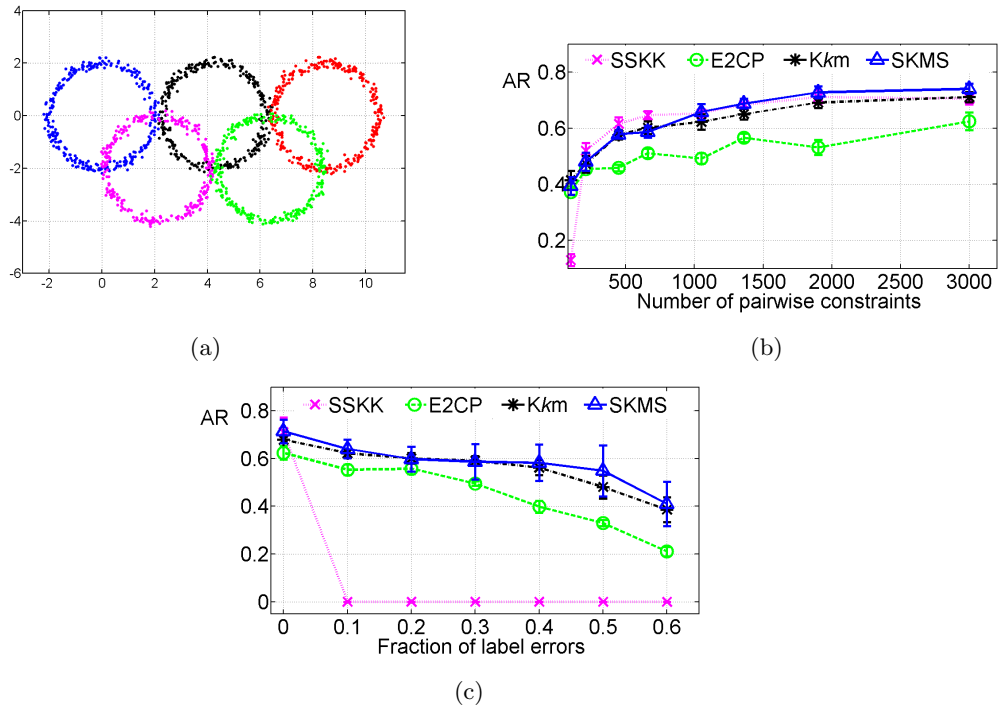


Figure 5.9: Olympic circles. (a) Input data. (b) AR index as the number of pairwise constraints is varied. (c) AR index as the fraction of mislabeled constraints is varied.

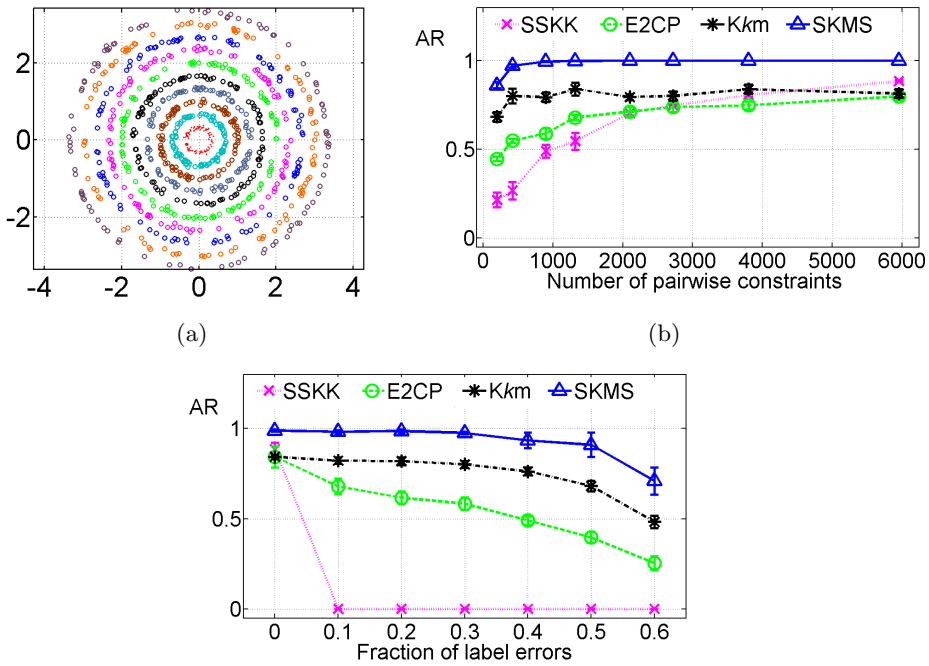
5.7.1 Synthetic Examples

Olympic Circles ($\sigma = 0.75$, $k = 15-35$)

As shown in Fig. 5.9a, the data consists of noisy points along five intersecting circles each comprising 300 points. For SSKK and E2CP algorithms, the value of the initial kernel parameter σ was 0.5.

We performed two sets of experiments. In the first experiment, the performance of all the algorithms is compared by varying the total number of pairwise constraints. The number of labeled points per class vary as $\{5, 7, 10, 12, 15, 17, 20, 25\}$ and are used to generate must-link and cannot-link constraints that vary between 100 and 3000. Fig. 5.9b demonstrates SKMS performs better than E2CP while its performance is similar to SSKK. For 100 constraints, SKMS recovered $5-8$ clusters, making a mistake 22% of the times. For all other settings together, it recovered an incorrect number of clusters $(4-6)$ only 6.3% of the times.

In the second experiment, we introduced labeling errors by randomly swapping the



(a) (b)

(c)

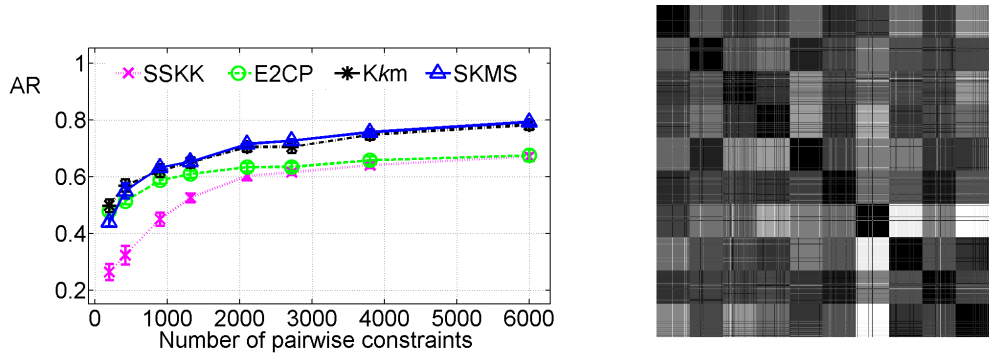
Figure 5.10: Ten concentric circles. (a) Input data. (b) AR index as the number of pairwise constraints is varied. (c) AR index as the fraction of mislabeled constraints is varied.

labels of a fraction of the pairwise constraints. We use 20 labeled sample points per class to generate 1900 pairwise constraints and vary the fraction of mislabeled constraints between 0 and 0.6 in steps of 0.1. Fig. 5.9c shows the clustering performance of all the methods. The performance of SKMS degrades only slightly even when half the constraint points are labeled wrongly, but it starts deteriorating when 60% of the constraint pairs are mislabeled.

Concentric Circles ($\sigma = 1$, $k = 25 - 45$)

The data consists of ten concentric circles each comprising 100 noisy points (Fig. 5.10a). We vary the number of labeled points per class as $\{5, 7, 10, 12, 15, 17, 20, 25\}$ and generate pairwise constraints between 200 and 6000. The other two algorithms have the value of $\sigma = 0.2$.

In the first experiment, the number of pairwise constraints are varied between 200 and 6000. For 200 constraints, SKMS incorrectly recovered nine clusters 50% of the times, while for all the other settings it correctly detected 10 clusters every time. In



(a) (b)

Figure 5.11: USPS digits. (a) AR index as the number of pairwise constraints is varied. (b) The 11000×11000 pairwise distance matrix (PDM) after performing mean shift clustering.

the second experiment, we use 6000 pairwise constraints and mislabeled a fraction of randomly selected constraints. This mislabeled fraction was varied between 0 to 0.6 in steps of 0.1 and the performance of all the algorithms is shown in Fig. 5.10b-c.

5.7.2 Real-World Applications

In this section, we demonstrate the performance of our method on two real applications having a small number of classes; USPS digits: 10 classes and MIT scene: 8 classes; and two real applications with a large number of clusters; PIE faces: 68 classes and Caltech-101 subset: 50 classes. We also show the performance of SKMS while clustering out of sample points using (5.32) on the USPS digits and the MIT scene data sets. Since the sample size per class is much smaller for PIE faces and Caltech-101 subset, results for generalization are not shown. We observe that the superiority of SKMS over other competing algorithms is clearer when the number of clusters is large.

USPS Digits ($\sigma = 7$, $k = 4-14$)

The USPS digits data set is a collection of 16×16 grayscale images of natural handwritten digits and is available from `http://cs.nyu.edu/ roweis/data.html`. Each class contains 1100 images of one of the ten digits. Fig. 5.12 shows sample images from each class. Each image is then represented with a 256-dimensional vector where the columns of the image are concatenated. We vary the number of labeled points per class as $\{5, 7, 10, 12, 15, 17, 20, 25\}$ to generate pairwise constraints from 200 to 6000. The

maximum number of labeled points per class used comprises only 2.27% of the total data.

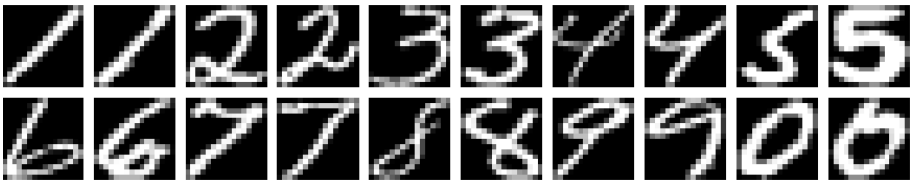


Figure 5.12: Sample images from the USPS digits data set.

Since the whole data set has 11000 points, we select 100 points from each class at random to generate a 1000 sample subset. The labeled points are selected at random from this subset for learning the 1000×1000 kernel matrix. The value of σ used was 5 for SSKK and 2 for E2CP.

In the first experiment we compare the performance of all the algorithms on this subset of 1000 points. For each algorithm, the results were averaged over 50 independent runs. Fig. 5.11a shows the clustering performance of all the methods. Once the number of constraints were increased beyond 500, SKMS outperformed the other algorithms. For 200 constraints, the SKMS discovered $9-11$ clusters, making a mistake 18% of the times, while it recovered exactly 10 clusters in all the other cases.

In the second experiment, we evaluated the performance of SKMS for the entire data set using 25 labeled points per class. The AR index averaged over 50 runs was 0.7529 ± 0.0510. Note that from Fig. 5.11a it can be observed that there is only a marginal decrease in clustering accuracy, showing that SKMS generalizes well over out of sample points. The pairwise distance matrix (PDM) after performing mean shift clustering is shown in Fig. 5.11b. For illustration purpose, the data is ordered such that the images belonging to the same class appear together. The block diagonal structure indicates good generalization of SKMS for out of sample points with little confusion between classes. Neither SSKK nor E2CP could be generalized to out of sample points because these methods need to learn a new kernel or affinity matrix (11000×11000).

MIT Scene ($\sigma = 1.75$, $k = 4-14$)

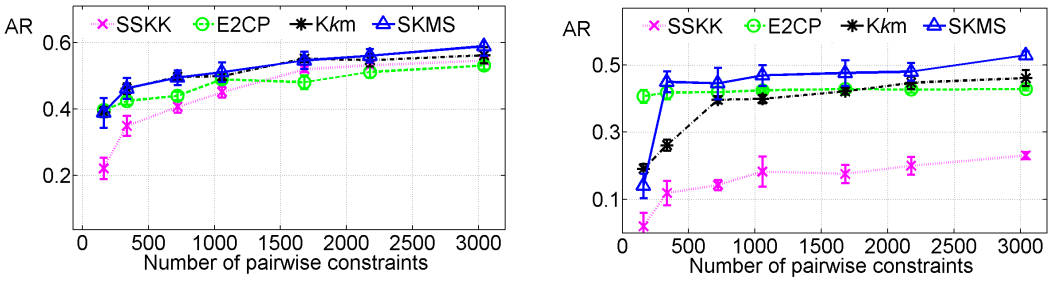
The data set is available from MIT `http://people.csail.mit.edu/torralba/code/spatialenvelope/` and contains 2688 labeled images. Each image is 256×256 pixels

in size and belongs to one of the eight outdoor scene categories, four natural and four man-made. Fig. 5.13 shows one example image from each of the eight categories. Using the code provided with the data, we extracted the GIST descriptors [88] which were then used to test all the algorithms. We vary the number of labeled points per class as $\{5, 7, 10, 12, 15, 17, 20\}$ to generate the pairwise constraints from 160 to 3040. The maximum number of labeled points used comprises only 7.44% of the total data.

Figure 5.13: Sample images from each of the eight categories of the MIT scene data set.

We select 100 points from each class at random to generate an 800 sample subset. The labeled points are selected at random from this subset for learning the 800×800 kernel matrix. The value of σ used was 1 for SSKK and 0.5 for E2CP. Fig. 5.14a shows the clustering performance of all the algorithms on the 800 sample subset as the number of constraint points are varied. For 160 constraint pairs, SKMS incorrectly discovered $7-10$ clusters about 22% of the time, while it correctly recovered eight clusters in all other settings.

In the second experiment, the whole data set was clustered using the 800×800 learned kernel matrix and generalizing to the out of sample points (5.32). Both SSKK and E2CP were used to learn the full 2688×2688 kernel and affinity matrix respectively. Fig. 5.14b shows the performance of all the algorithms as the number of pairwise constraints was varied. Note that in [74] for similar experiments, the superior performance of E2CP was probably because of the use of spatial Markov kernel instead of the Gaussian kernel.



(a) (b)

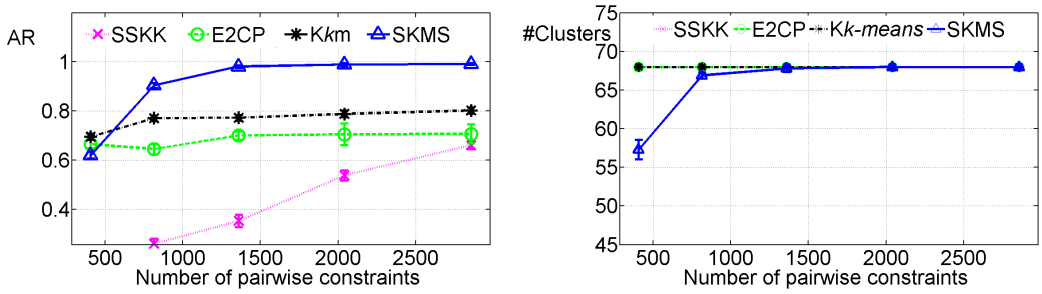
Figure 5.14: MIT Scene data set. (a) AR index on the 800×800 kernel matrix as the number of pairwise constraints is varied. (b) AR index on *entire data* as the number of pairwise constraints is varied.

PIE Faces ($\sigma = 25$, $k = 4 - 7$)

From the CMU PIE face data set [105], we use only the frontal pose and neutral expression of all 68 subjects under 21 different lighting conditions. We coarsely aligned the images with respect to eye and mouth locations and resized them to be 128×128. In Fig.5.15, we show eight illumination conditions for three different subjects. Due to significant illumination variation, interclass variability is very large and some of the samples from different subjects appear to be much closer to each other than within classes.

Figure 5.15: PIE faces data set. Sample images showing eight different illuminations for three subjects.

We convert the images from color to gray scale and normalize the intensities between zero and one. Each image is then represented with a 16384-dimensional vector where the columns of the image are concatenated. We vary the number of labeled points per class as $\{3, 4, 5, 6, 7\}$ to generate pairwise constraints between 408 and 2856. The



(a) (b)

Figure 5.16: PIE Faces data set. (a) AR index as the number of pairwise constraints is varied. (b) Number of clusters discovered by SKMS as number of pairwise constraints is varied.

maximum number of labeled points comprises 30% of the total data.

We generate the 1428×1428 initial kernel matrix using all the data. The value of σ used was 10 for SSKK and 22 for E2CP. Fig. 5.16 shows the performance of all the algorithms using the AR index in this experiment and it can be observed that SKMS outperforms all the other algorithms. Note that SKMS approaches near perfect clustering for more than 5 labeled points per class. When three labeled points per class were used, the SKMS discovered $61-71$ clusters, making a mistake about 84% of the time. For all other settings, SKMS correctly recovered 68 clusters about 62% of the times, while it recovered 67 clusters about 32% of the time and between $69-71$ clusters in the remaining runs. Note that the kernel k-means (Kkm) method performs poorly inspite of explicitly using the number of clusters and the same learned kernel matrix as SKMS.

Caltech-101 Objects ($\sigma = 0.5$, $k = 4 - 11$)

The Caltech-101 data set [33] is a collection of variable sized images across 101 object categories. This is a particularly hard data set with large intraclass variability. Fig. 5.17 shows sample images from eight different categories.

We randomly sampled a subset of 50 categories, as listed in Table 5.2, with each class containing 31 to 40 samples. For each sample image, we extract GIST descriptors [88] and use them for evaluation of all the competing clustering algorithms. We vary the number of labeled points per class as $\{5, 7, 10, 12, 15\}$ to generate pairwise constraints

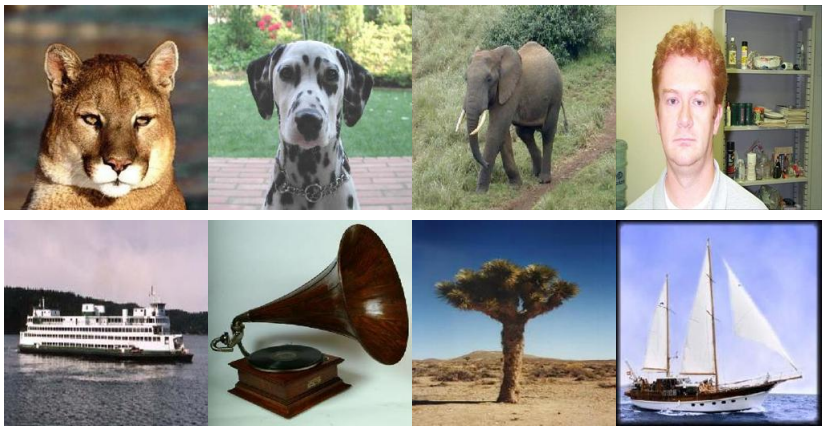


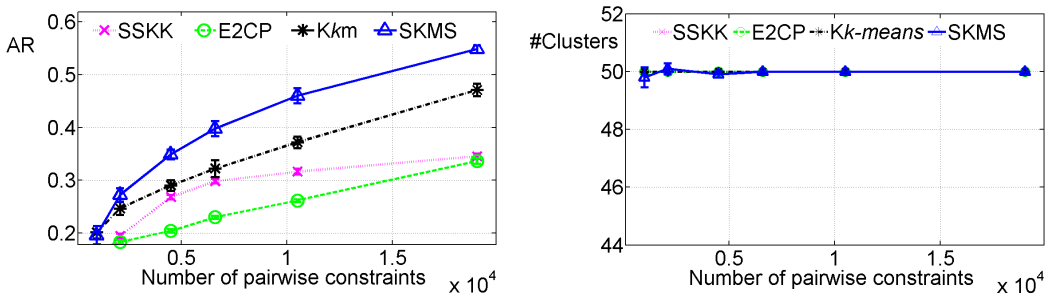
Figure 5.17: Sample images from eight of the 50 classes used from Caltech-101 data set.

between 500 and 10500. The maximum number of labeled points comprises 38% of the total data. We use a larger number of constraints in order to overcome the large variability in the data set.

We generate the 1959×1959 initial kernel matrix using all the data. The value of σ used is 0.2 for E2CP and 0.3 for SSKK. Fig. 5.18 shows the comparison of SKMS with the other competing algorithms. It can be seen that SKMS outperforms all the other methods. For five labeled points per class, SKMS detected $50 - 52$ clusters, making mistakes 75% of the times. For all the other settings together, SKMS recovered the incorrect number $(48 - 51)$ of clusters only 9% of the times.

Table 5.2: Object classes used from Caltech-101 data set.

elephant	flamingo_head	emu	faces	gerenuk	stegosaurus
accordion	ferry	cougar_face	mayfly	chair	scissors
menorah	platypus	butterfly	tick	metronome	inline_skate
bass	pyramid	leopards	sea_horse	cougar_body	stop_sign
lotus	dalmatian	gramophone	camera	trilobite	dragonfly
grand_piano	headphone	sunflower	ketch	wild_cat	crayfish
nautilus	buddha	yin_yang	dolphin	minaret	anchor
car_side	rooster	wheelchair	octopus	joshua_tree	ant
umbrella	crocodile				



(a) (b)

Figure 5.18: Caltech-101 data set. (a) AR index as the number of pairwise constraints is varied. (b) Number of clusters discovered by SKMS as number of pairwise constraints is varied.

5.8 Discussion

The performance of unsupervised clustering methods is directly impacted by the ability of the underlying metric to find meaningful structures in data. We described a semi-supervised kernel mean shift (SKMS) clustering algorithm using pairwise must-link and cannot-link constraints. We showed the superior performance of our algorithm compared to state-of-the-art using challenging synthetic and real data.

However, the clustering problem itself becomes very difficult when the data has large intra-class variability. For example, Fig. 5.19 shows three images from the *highway* category of the MIT scene data set that were misclassified. The misclassification error rate in Caltech-101 data set was even higher. On large scale data sets with over 10000 categories [28], all classification methods will perform poorly, as an image may qualify for multiple categories. For example, the images in Fig. 5.19 were classified in the *street* category, which is semantically correct. To deal with a large number of categories,

Figure 5.19: Three images from the class *highway* of the MIT Scene data set that were misclassified.

clustering (classification) algorithms should incorporate the ability to use higher level semantic features that connect an image to its possible categories.

We did not address the issue of clustering in presence of outliers. In many practical scenarios, the outlying points should be detected and eliminated before the clustering procedure begins.

Chapter 6

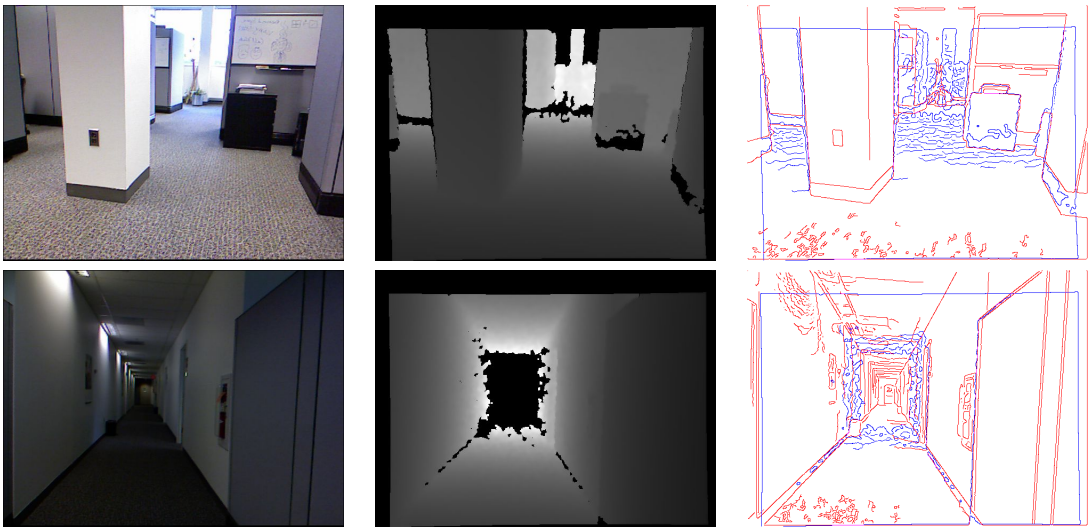
Future Work

In this chapter, we briefly give an overview of possible directions for future work. In Section 6.1, we discuss the framework in which a sequence of RGB-D images can be processed in order to build a dynamic 3D model. We discuss the reformulation of the kernel learning problem in Section 6.2 and propose that a more efficient algorithm can be devised by combining ideas from group theory and semidefinite programming.

6.1 Dynamic Scene Modeling Using RGB-D Sequences

Our work on planar modeling from a single RGB-D image can be extended in three directions. In Chapter 4, we only used the depth image to detect the planar segments in the scene. We used surface normals and their properties to identify these planar regions. In the future, we would like to generalize this approach to identify other common geometric shapes, like cylinders, cones and spheres while maintaining the scalability of the complete system.

The Microsoft Kinect can have 'holes' in the image, i.e. group of pixels where the depth estimate was unavailable. Fig. 6.1 shows edges extracted from the RGB (red) and the depth (blue) images. It is clear that the RGB image has additional information that can be used, e.g., to fill up some of the holes in the depth image. For example, the linear edges can provide crucial information about intersection of planes even at points where depth data is missing. Moreover, using the 3D information in the depth image, we can robustly identify the vanishing points and vanishing lines of the image. Methods for scene reconstruction from a single 2D image [89, 100], usually are sensitive to estimates of plane at infinity. By using the depth information and the detected planes, we can accurately estimate the plane at infinity using 3D data. This additional



(a) (b) (c)

Figure 6.1: Edge image of RGB and depth images for SCT-1 and SCT-2. RGB images can provide information about regions of missing depth. (a) RGB image. (b) Depth image. (c) Edge image - Red edges correspond to the RGB image, while the blue edge correspond to the depth image.

information can be used to overcome some limitations of depth data like, range of the depth sensor, noisy measurements at depth discontinuities etc.

Another proposed extension of this work is processing a sequence of RGB-D frames from Kinect. Performing the planar segmentation in the first frame, we can compute the covariances of each of the planar parameters. We can compute the direction of the intersecting edge using the normals as $\mathbf{l} = \widehat{\mathbf{n}}_1 \times \widehat{\mathbf{n}}_2$ and the corresponding covariance by error propagation. Using these parameters and their uncertainties as initializations, a particle filtering framework can be used to estimate the planar regions in the subsequent frames. An interesting question would be how to optimally use the information from the RGB channel in a unified framework.

6.2 Kernel Learning Using Independent Constraints

The method of learning a positive semidefinite kernel matrix described in Section 5.4.2 is equivalent to a dual coordinate descent for optimization of the $\log\det$ Bregman divergence. We present only the reformulation of the kernel learning problem. The development of the algorithm involves combining ideas from group theory and semidefinite

programming [3] and is proposed as future work.

Let $\mathbf{\Phi} = [\boldsymbol{\phi}(\mathbf{x}_1), \boldsymbol{\phi}(\mathbf{x}_2), \ldots, \boldsymbol{\phi}(\mathbf{x}_n)]$ be the $d_{\boldsymbol{\phi}} \times n$ matrix of data points in an input space $\mathbb{R}^d_{\boldsymbol{\phi}}$. We use notation similar to Section 5.3.1 and define $\mathbf{Z} = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_{n_c}]$ as the $n \times n_c$ matrix of constraint pair indicator vectors of the form $\mathbf{z}_j = \mathbf{e}_{j_1} - \mathbf{e}_{j_2},\ (j_1, j_2) \in \mathcal{M} \cup \mathcal{C}$, where $\mathcal{M}$ and $\mathcal{C}$ are the sets of must-link and cannot-link pairs.

Let $\mathbf{Y} = \mathbf{\Phi}\mathbf{Z}$ be the $d_{\boldsymbol{\phi}} \times n_c$ matrix of constraints, with column rank $r \leq \min(d_{\boldsymbol{\phi}}, n_c)$, then the constraint inequalities can be written as

$$\begin{aligned} \mathbf{y}_j^\top \mathbf{T} \mathbf{y}_j &\leq \xi_j \qquad \forall (j_1, j_2) \in \mathcal{M} \\ \mathbf{y}_j^\top \mathbf{T} \mathbf{y}_j &\geq \xi_j \qquad \forall (j_1, j_2) \in \mathcal{C} \end{aligned} \tag{6.1}$$

where the elements ξ_j of the vector $\boldsymbol{\xi}$ are desired distance values corresponding to must-link and cannot-link constraints. $\mathbf{T}$ is the desired symmetric transformation matrix to be learned. Let $\mathbf{U_Y \Sigma_Y V_Y^\top}$ be the SVD of $\mathbf{Y}$, with $\mathbf{\Sigma_Y}$ as the $r \times r$ diagonal matrix and $\mathbf{U_Y}$ and $\mathbf{V_Y}$ as the $d_{\boldsymbol{\phi}} \times r$ and $n_c \times r$ matrices respectively such that $\mathbf{U_Y^\top U_Y} = \mathbf{V_Y^\top V_Y} = \mathbf{I}_r$.

To satisfy (6.1), $\mathbf{T}$ should be of the following form

$$\mathbf{T} = \mathbf{I}_{d_{\boldsymbol{\phi}}} + \mathbf{Y} \mathbf{M_Y} \mathbf{Y}^\top \tag{6.2}$$

where the second term of the transformation modifies only the r-dimensional column space of $\mathbf{Y}$ through the $n_c \times n_c$ symmetric matrix $\mathbf{M_Y}$. We can chose to write $\mathbf{M_Y} = \mathbf{V_Y W V_Y^\top}$, where $\mathbf{W}$ is an $r \times r$ symmetric and positive (semi)definite matrix. In the kernel space, $\mathbf{U_Y}$ cannot be computed explicitly, so from (6.2) we have

$$\begin{aligned} \mathbf{T} &= \mathbf{I}_{d_{\boldsymbol{\phi}}} + \mathbf{Y} \left(\mathbf{V_Y W V_Y^\top} \right) \mathbf{Y}^\top \\ &= \left(\mathbf{I}_{d_{\boldsymbol{\phi}}} - \mathbf{Y} \left(\mathbf{Y}^\top \mathbf{Y} \right)^+ \mathbf{Y}^\top \right) + \mathbf{Y} \left(\left(\mathbf{Y}^\top \mathbf{Y} \right)^+ + \mathbf{V_Y W V_Y^\top} \right) \mathbf{Y}^\top \\ &= \mathbf{P}_{\mathcal{N}(\mathbf{Y})} + \mathbf{Y V_Y} \left(\left(\mathbf{\Sigma_Y^\top \Sigma_Y} \right)^+ + \mathbf{W} \right) \mathbf{V_Y^\top Y}^\top. \end{aligned} \tag{6.3}$$

The final expression (6.3) represents $\mathbf{T}$ as the sum of a projection matrix to $\mathcal{N}(\mathbf{Y})$, the

null space of $\mathbf{Y}$ and the linear transformation of the subspace spanned by the columns of $\mathbf{Y}$. Let $\mathbf{S} = \left(\mathbf{\Sigma}_{\mathbf{Y}}^{\top}\mathbf{\Sigma}_{\mathbf{Y}}\right)^{+} + \mathbf{W} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top}$ be the $r \times r$ matrix corresponding to the term within the parantheses. Therefore the transformation $\mathbf{T}$ becomes

$$\mathbf{T} = \mathbf{P}_{\mathcal{N}(\mathbf{Y})} + \mathbf{Y}\mathbf{V}_{\mathbf{Y}}\mathbf{S}\mathbf{V}_{\mathbf{Y}}^{\top}\mathbf{Y}^{\top}. \tag{6.4}$$

The matrix $\mathbf{V}_{\mathbf{Y}}$ can be computed using the SVD of the matrix $\mathbf{Y}^{\top}\mathbf{Y} = \mathbf{Z}^{\top}\mathbf{K}\mathbf{Z}$, where $\mathbf{K} = \mathbf{\Phi}^{\top}\mathbf{\Phi}$ is the kernel matrix. Recalling that $\mathbf{y}_j = \mathbf{\Phi}\mathbf{z}_j$, the distance between the jth constraint pair of points can now be written in terms of the kernel matrix $\mathbf{K}$ as

$$\begin{aligned}\mathbf{y}_j^{\top}\mathbf{T}\mathbf{y}_j &= \mathbf{y}_j^{\top}\mathbf{P}_{\mathcal{N}(\mathbf{Y})}\mathbf{y}_j + \mathbf{y}_j^{\top}\mathbf{Y}\mathbf{V}_{\mathbf{Y}}\mathbf{S}\mathbf{V}_{\mathbf{Y}}^{\top}\mathbf{Y}^{\top}\mathbf{y}_j \\ &= \mathbf{z}_j^{\top}\mathbf{K}\mathbf{Z}\mathbf{V}_{\mathbf{Y}}\mathbf{S}\mathbf{V}_{\mathbf{Y}}^{\top}\mathbf{Z}^{\top}\mathbf{K}\mathbf{z}_j \end{aligned} \tag{6.5}$$

In (6.5), if the only unknown matrix $\mathbf{S}$ satisfies all the constraints, then the learned kernel is computed as

$$\begin{aligned}\widehat{\mathbf{K}} &= \mathbf{X}^{\top}\mathbf{T}\mathbf{X} = \mathbf{X}^{\top}\left(\mathbf{P}_{\mathcal{N}(\mathbf{Y})} + \mathbf{Y}\mathbf{V}_{\mathbf{Y}}\mathbf{S}\mathbf{V}_{\mathbf{Y}}^{\top}\mathbf{Y}^{\top}\right)\mathbf{X} \\ \widehat{\mathbf{K}} &= \left(\mathbf{K} - \mathbf{K}\mathbf{Z}\left(\mathbf{Z}^{\top}\mathbf{K}\mathbf{Z}\right)^{+}\mathbf{Z}^{\top}\mathbf{K}\right) + \mathbf{K}\mathbf{Z}\mathbf{V}_{\mathbf{Y}}\mathbf{S}\mathbf{V}_{\mathbf{Y}}^{\top}\mathbf{Z}^{\top}\mathbf{K}. \end{aligned} \tag{6.6}$$

The first term of (6.6) is the kernel matrix projected to the null space of $\mathbf{Y}$ [113], while the matrix $\mathbf{S}$ needs to be learned in order to compute the second term.

Learning the Transformation Matrix

In order to learn the transformation matrix $\mathbf{T}$, we want to minimize some divergence between $\mathbf{T}$ and $\mathbf{I}_{d_{\phi}}$ while imposing the constraints. Since $\mathbf{T}$ can be written as a sum of matrices spanning complementary subspaces (6.4), it can be shown that $D_{ld}(\mathbf{T}, \mathbf{I}_{d_{\phi}}) = D_{ld}(\mathbf{S}, \mathbf{I}_r)$ [63]. The $\log\det$ divergence $D_{ld}(\cdot)$ depends only on the eigenvectors and eigenvalues of the arguments. The intuition here is that by modifying $\mathbf{S}$, we only modify the eigenvalues and eigenvectors lying in the subspace spanned by the columns of $\mathbf{Y}$, while the null space of $\mathbf{Y}$ remains unchanged.

For two $n \times n$ symmetric positive definite matrices $\mathbf{K} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^{\top}$ and $\mathbf{K}_0 = \mathbf{V}\mathbf{\Theta}\mathbf{V}^{\top}$,

recall that the log det divergence between $\mathbf{K}$ and $\mathbf{K}_0$ is computed as

$$D_{ld}(\mathbf{K}, \mathbf{K}_0) = \sum_{i,j=1}^{n} \left(\mathbf{u}_i^\top \mathbf{v}_j\right)^2 \left(\frac{\lambda_i}{\theta_j} - \log \frac{\lambda_i}{\theta_j} - 1\right). \tag{6.7}$$

Therefore the log det divergence between $\mathbf{S} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ and $\mathbf{I}_r$ is

$$\begin{aligned} D_{ld}(\mathbf{S}, \mathbf{I}_r) &= \sum_{i,j=1}^{r} \left(\mathbf{u}_i^\top \mathbf{e}_j\right)^2 (\lambda_i - \log \lambda_i - 1) \\ &= \sum_{i=1}^{r} (\lambda_i - \log \lambda_i - 1) \left(\sum_{j=1}^{r} \left(\mathbf{u}_i^\top \mathbf{e}_j\right)^2\right) \\ &= \sum_{i=1}^{r} (\lambda_i - \log \lambda_i - 1) \, ||\mathbf{u}_i||_2^2 \\ &= \sum_{i=1}^{r} (\lambda_i - \log \lambda_i - 1) \end{aligned}$$

where $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \ldots \mathbf{u}_r]$ is the $r \times r$ orthogonal matrix and $\mathbf{\Lambda} = \mathbf{diag}([\lambda_1, \lambda_2, \ldots, \lambda_r])$ is the diagonal matrix of eigenvalues of $\mathbf{S}$. Since the matrices $\mathbf{K}, \mathbf{Z}$ and $\mathbf{V_Y}$ are known and do not change, for convenience of notation, we define $\widetilde{\mathbf{y}}_j = \mathbf{V}_\mathbf{Y}^\top \mathbf{Z}^\top \mathbf{K}\mathbf{z}_j$. The constraint inequalities in (6.1) can now be written as

$$\begin{aligned} \widetilde{\mathbf{y}}_j^\top \mathbf{S} \widetilde{\mathbf{y}}_j &\leq \xi_j \qquad \forall (j_1, j_2) \in \mathcal{M} \\ \widetilde{\mathbf{y}}_j^\top \mathbf{S} \widetilde{\mathbf{y}}_j &\geq \xi_j \qquad \forall (j_1, j_2) \in \mathcal{C} \end{aligned}$$

The minimization problem can be rewritten as

$$\begin{aligned} \min_{\mathbf{S}} \quad & D_{ld}(\mathbf{S}, \mathbf{I}_r) \\ \text{s.t.} \quad & \widetilde{\mathbf{y}}_j^\top \mathbf{S} \widetilde{\mathbf{y}}_j \leq \xi_j \qquad \forall (j_1, j_2) \in \mathcal{M} \\ & \widetilde{\mathbf{y}}_j^\top \mathbf{S} \widetilde{\mathbf{y}}_j \geq \xi_j \qquad \forall (j_1, j_2) \in \mathcal{C} \end{aligned} \tag{6.8}$$

Writing the optimization problem by separating the diagonal eigenvalue matrix $\mathbf{\Lambda}$ and

the orthogonal eigenvector matrix $\mathbf{U}$, we get the following

$$
\begin{aligned}
\min_{\mathbf{U},\mathbf{\Lambda}} \quad & \operatorname{tr}\left(\mathbf{\Lambda} - \log \mathbf{\Lambda} - \mathbf{I}_r\right) \\
\text{s.t.} \quad & \widetilde{\mathbf{y}}_j^\top \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top \widetilde{\mathbf{y}}_j \leq \xi_j \qquad \forall (j_1, j_2) \in \mathcal{M} \\
& \widetilde{\mathbf{y}}_j^\top \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top \widetilde{\mathbf{y}}_j \geq \xi_j \qquad \forall (j_1, j_2) \in \mathcal{C} \\
& \mathbf{\Lambda} \succ 0 \\
& \mathbf{U}^\top \mathbf{U} = \mathbf{I}_r
\end{aligned}
\tag{6.9}
$$

The algorithm for solving this problem can be developed by taking ideas from group theory and semidefinite programming, e.g., the Q-method [3]. This approach will remove the redundancy in the kernel learning by using only the linearly independent constraints. However, designing the complete algorithm was out of the scope of this thesis and will be pursued in future.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34:2274–2282, 2012.

[2] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Commun. ACM*, 54:105–112, 2011.

[3] F. Alizadeh, J.-P. A. Haeberly, and M. L. Overton. Primal-dual interior-point methods for semidefinite programming: Convergence rates, stability and numerical results. *SIAM J. on Optimization*, 8:746–768, 1998.

[4] D. Anderson, H. Herman, and A. Kelly. Experimental characterization of commercial flash ladar devices. In *Int. Conf. of Sensing and Technology*, Nov. 2005.

[5] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

[6] S. Basu, M. Bilenko, and R. Mooney. A probabilistic framework for semi-supervised clustering. In *Proc. 10th Intl. Conf. on Knowledge Discovery and Data Mining*, 2004.

[7] S. Basu, I. Davidson, and K. Wagstaff. *Constrained Clustering: Advances in Algorithms, Theory, and Applications.* Chapman & Hall/CRC Press, 2008.

[8] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (SURF). *Comput. Vis. Image Underst.*, 110:346–359, June 2008.

[9] M. Bilenko, S. Basu, and R. J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *Int. Conf. Machine Learning*, pages 81–88, 2004.

[10] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23:1222–1239, 2001.

[11] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *USSR Computational Mathematics and Mathematical Physics*, 7:200–217, 1967.

[12] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *J. ACM*, 58:11:1–11:37, 2011.

[13] O. Chapelle, B. Schlkopf, and A. Zien. *Semi-Supervised Learning.* MIT Press, 2006.

[14] A.-L. Chauve, P. Labatut, and J.-P. Pons. Robust piecewise-planar 3d reconstruction and completion from large-scale unstructured point data. In *CVPR*, pages 1261–1268, 2010.

[15] H. Chen and P. Meer. Robust fusion of uncertain information. *IEEE Sys. Man Cyber. Part B*, 35:578–586, 2005.

[16] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE PAMI*, 17:790–799, 1995.

[17] T.-J. Chin, H. Wang, and D. Suter. The ordered residual kernel for robust motion subspace clustering. In *NIPS*, pages 333–341. 2009.

[18] J. Choi and G. Medioni. Starsac: Stable random sample consensus for parameter estimation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 675–682, 2009.

[19] J. C. K. Chow, K. D. Ang, D. D. Lichti, and W. F. Teskey. Performance analysis of a low-cost triangulation-based 3D camera: Microsoft Kinect system. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XXXIX-B5:175–180, 2012.

[20] L. M. Collins and C. W. Dent. Omega: A general formulation of the rand index of cluster recovery suitable for non-disjoint solutions. *Multivariate Behavioral Research*, 23(2):231–242, 1988.

[21] R. Collins. Mean shift blob tracking through scale space. In *CVPR*, volume 2, pages 234–240, 2003.

[22] D. Comaniciu. Variable bandwidth density-based fusion. In *CVPR*, volume 1, pages 56–66, 2003.

[23] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24:603–619, 2002.

[24] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR*, volume 1, pages 142–149, 2000.

[25] N. Cristianini and J. Shawe-Taylor. *Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge U. Press, 2000.

[26] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. volume 1, pages 886–893, 2005.

[27] A. Delong, A. Osokin, H. Isack, and Y. Boykov. Fast approximate energy minimization with label costs. *International Journal of Computer Vision*, 96:1–27, 2012.

[28] J. Deng, A. C. Berg, K. Li, and L. Fei-Fei. What does classifying more than 10, 000 image categories tell us? In *ECCV*, pages 71–84, 2010.

[29] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. Wiley, second edition, 2001.

[30] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *CVPR*, pages 2790–2797, 2009.

[31] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the RGB-D SLAM system. In *Proc. of Int. Conf. on Robotics and Automation (ICRA)*, May 2012.

[32] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Int. Conf. on Knowledge Discovery and Data Mining*, pages 226–231, 1996.

[33] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE CVPR Workshop of Generative Model Based Vision (WGMBV)*, 2004.

[34] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59:167–181, Sept. 2004.

[35] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Assoc. Comp. Mach*, 24(6):381–395, 1981.

[36] M. Franaszek, G. Cheok, and K. Saidi. Experimental verification of formulas for variances of plane parameters fitted to three-dimensional imaging data. *IEEE Trans. on Instrumentation and Measurement*, 61(1):103 –110, jan. 2012.

[37] B. J. Frey and D. Dueck. Clustering by passing messages between data points. *Science*, 315:972–976, 2007.

[38] K. Fukunaga and L. D. Hostetler. The estimation of the gradient of a density function with applications in pattern recognition. *IEEE Information Theory*, 21:32–40, 1975.

[39] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Reconstructing building interiors from images. In *ICCV*, pages 80–87, 2009.

[40] D. Gallup, J.-M. Frahm, and M. Pollefeys. Piecewise planar and non-planar stereo for urban scene reconstruction. In *CVPR*, pages 1418–1425, 2010.

[41] A. Geiger, J. Ziegler, and C. Stiller. Stereoscan: Dense 3d reconstruction in real-time. In *IEEE Intelligent Vehicles Symposium*, pages 963–968, June 2011.

[42] R. Ghani. Combining labeled and unlabeled data for multiclass text categorization. In *Proc. of Int. Conf. on Machine Learning*, pages 187–194, 2002.

[43] I. E. Givoni and B. J. Frey. Semi-supervised affinity propagation with instance-level constraints. *Journal of Machine Learning Research - Proceedings Track*, 5:161–168, 2009.

[44] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3D scene geometry to human workspace. In *CVPR*, pages 1961–1968, 2011.

[45] G. Hager, M. Dewan, and C. Stewart. Multiple kernel tracking with SSD. In *CVPR*, pages 790–797, 2004.

[46] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[47] V. Hedau, D. Hoiem, and D. A. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, pages 1849–1856, 2009.

[48] V. Hedau, D. Hoiem, and D. A. Forsyth. Thinking inside the box: Using appearance models and context based on room geometry. In *ECCV*, pages 224–237, 2010.

[49] D. C. Herrera, J. Kannala, and J. Heikkila. Joint depth and color camera calibration with distortion correction. *IEEE PAMI*, 34:2058–2064, 2012.

[50] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher. An experimental

comparison of range image segmentation algorithms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18:673–689, July 1996.

[51] L. Hubert and P. Arabie. Comparing partitions. *Journal of Classification*, 2:193–218, 1985.

[52] H. Isack and Y. Boykov. Energy-based geometric multi-model fitting. *International Journal of Computer Vision*, 97:123–147, 2012.

[53] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. of ACM Symp. on User Interface Software and Technology*, pages 559–568, 2011.

[54] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern Recognition Letters*, 31:651–666, 2010.

[55] P. Jain, B. Kulis, J. V. Davis, and I. S. Dhillon. Metric and kernel learning using a linear transformation. *Journal of Machine Learning Research*, 13:519–547, 2012.

[56] S. D. Kamvar, D. Klein, and C. D. Manning. Spectral learning. In *Intl. Conf. on Artificial Intelligence*, pages 561–566, 2003.

[57] K. Khoshelham and S. O. Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[58] K. Klasing, D. Althoff, D. Wollherr, and M. Buss. Comparison of surface normal estimation methods for range sensing applications. In *ICRA*, pages 3206–3211, 2009.

[59] S. Kluckner and H. Bischof. Image based building classification and 3d modeling with super-pixels. *Proc. of Int. Soc. for Photogrammetry and Remote Sensing, Photogrammetric Computer Vision and Image Analysis*, 2010.

[60] H. S. Koppula, A. Anand, T. Joachims, and A. Saxena. Semantic labeling of 3D point clouds for indoor scenes. In *NIPS*, pages 244–252. 2011.

[61] B. Kulis, S. Basu, I. Dhillon, and R. Mooney. Semi-supervised graph clustering: a kernel approach. In *Proc. 22nd Intl. Conf. on Machine Learning*, 2005.

[62] B. Kulis, S. Basu, I. S. Dhillon, and R. J. Mooney. Semi-supervised graph clustering: A kernel approach. *Machine Learning*, 74:1–22, 2009.

[63] B. Kulis, M. A. Sustik, and I. S. Dhillon. Low-rank kernel learning with Bregman matrix divergences. *Journal of Machine Learning Research*, 10:341–376, 2009.

[64] F. Lafarge and C. Mallet. Building large urban environments from unstructured point data. In *ICCV*, pages 1068–1075, 2011.

[65] F. Lafarge and C. Mallet. Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. *IJCV*, 99:69–85, 2012.

[66] D. C. Lee, A. Gupta, M. Hebert, and T. Kanade. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, pages 1288–1296, 2010.

[67] D. C. Lee, M. Hebert, and T. Kanade. Geometric reasoning for single image structure recovery. In *CVPR*, pages 2136–2143, 2009.

[68] L. Lelis and J. Sander. Semi-supervised density-based clustering. In *IEEE Proc. Int. Conf. of Data Mining*, pages 842–847, 2009.

[69] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. Turbopixels: Fast superpixels using geometric flows. *PAMI*, 31:2290–2297, Dec. 2009.

[70] M. Liu, B. Vemuri, S.-I. Amari, and F. Nielsen. Shape retrieval using hierarchical total bregman soft clustering. *PAMI*, 34:2407 –2419, 2012.

[71] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *CVPR*, pages 2097–2104, 2011.

[72] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

[73] Lu Zhengdong and M. A. Carreira-Perpiñán. Constrained spectral clustering through affinity propagation. In *CVPR*, 2008.

[74] Lu Zhiwu and H. H.-S. Ip. Constrained spectral clustering via exhaustive and efficient constraint propagation. In *ECCV*, pages 1–14, 2010.

[75] U. Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17:395–416, 2007.

[76] MANCTL. Rgbdemo. *Available online: http://labs.manctl.com/rgbdemo*, 2012 (accessed Sept 24, 2012).

[77] B. C. Matei and P. Meer. Estimation of nonlinear errors-in-variables models for computer vision applications. *IEEE PAMI*, 28:1537–1552, 2006.

[78] D. Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, June 1982.

[79] G. Medioni and S. B. Kang. *Emerging Topics in Computer Vision*. Prentice Hall PTR, 2004.

[80] Microsoft. Kinect. *Available online: http://www.xbox.com/en-US/kinect*, 2012 (accessed Sept 24, 2012).

[81] Microsoft. Photosynth. *Available online: http://photosynth.net*, 2012 (accessed Sept 24, 2012).

[82] S. Mittal, S. Anand, and P. Meer. Generalized projection based m-estimator. *PAMI*, 99, 2012.

[83] A. P. Moore, S. J. D. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel Lattices. *CVPR*, 2008.

[84] B. Nadler and M. Galun. Fundamental limitations of spectral clustering. In *NIPS*, pages 1017–1024, 2007.

[85] R. A. Newcombe, A. J. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR)*, pages 127–136, Oct. 2011.

[86] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, pages 849–856, 2001.

[87] B. Oehler, J. Stueckler, J. Welle, D. Schulz, and S. Behnke. Efficient multi-resolution plane segmentation of 3D point clouds. In *Proc. of Int. Conf. on Intelligent Robotics and Applications*, pages 145–156, 2011.

[88] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42:145–175, 2001.

[89] L. D. Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard. Bayesian geometric modeling of indoor scenes. In *CVPR*, pages 2719–2726, 2012.

[90] J. Poppinga, A. Birk, and K. Pathak. A characterization of 3D sensors for response robots. *RoboCup 2009: Robot WorldCup XIII*, 5949:264–275, 2010.

[91] C. Poullis and S. You. 3d reconstruction of urban areas. In *Int. Conf. on 3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 33 –40, May 2011.

[92] L. Rabiner and R. Schafer. *Theory and Applications of Digital Speech Processing.* Prentice Hall Press, 1st edition, 2010.

[93] R. Raguram and J.-M. Frahm. RECON: Scale-adaptive robust estimation via residual consensus. In *ICCV*, pages 1299–1306, 2011.

[94] R. Raguram, J.-M. Frahm, and M. Pollefeys. A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus. In *Proceedings of the 10th European Conference on Computer Vision: Part II*, ECCV '08, pages 500–513, 2008.

[95] C. Ruiz, M. Spiliopoulou, and E. M. Ruiz. Density-based semi-supervised clustering. *Data Min. Knowl. Discov.*, 21:345–370, 2010.

[96] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *Int. Conf. on 3D Digital Imaging and Modeling (3DIM)*, June 2001.

[97] R. B. Rusu and S. Cousins. 3d is here: Point cloud library (PCL). In *Int. Conf. on Robotics and Automation (ICRA)*, 2011.

[98] F. Schindler, W. Förstner, and J.-M. Frahm. Classification and reconstruction of surfaces from point clouds of man-made objects. In *ICCV Workshop on Computer Vision for Remote Sensing of the Environment*, pages 257–263, 2011.

[99] R. Schnabel, R. Wahl, and R. Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007.

[100] A. G. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun. Efficient Structured Prediction for 3D Indoor Scene Understanding. In *CVPR*, 2012.

[101] A. G. Schwing and R. Urtasun. Efficient exact inference for 3d indoor scene understanding. In *ECCV (6)*, pages 299–313, 2012.

[102] Y. Sheikh, E. Khan, and T. Kanade. Mode-seeking by medoidshifts. In *ICCV*, pages 1–8, 2007.

[103] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22:888–905, 2000.

[104] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *ICCV Workshop on 3D Representation and Recognition*, pages 601–608.

[105] T. Sim, S. Baker, and M. Bsat. The CMU Pose, Illumination, and Expression (PIE) Database. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*, May 2002.

[106] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige. Double window optimisation for constant time visual SLAM. In *ICCV*, pages 2352–2359, 2011.

[107] R. Subbarao and P. Meer. Nonlinear mean shift for clustering over analytic manifolds. In *CVPR*, pages 1168–1175, 2006.

[108] M. Sun, G. R. Bradski, B.-X. Xu, and S. Savarese. Depth-encoded hough voting for joint object detection and shape recovery. In *ECCV (5)*, pages 658–671, 2010.

[109] M. Sushil. *User Independent Robust Statistics for Computer Vision*. PhD thesis, Rutgers, The State University of New Jersey, Sept. 2011.

[110] G. Takacs, M. El Choubassi, Y. Wu, and I. Kozintsev. 3D mobile augmented reality in urban scenes. In *IEEE Int. Conf. on Multimedia and Expo (ICME)*, pages 1–4, July 2011.

[111] R. Toldo and A. Fusiello. Robust multiple structures estimation with J-Linkage. In *ECCV*, pages 537–547, 2008.

[112] C. Tomasi and R. Manduchi. Bilateral filtering for gray and color images. In *ICCV*, pages 839–846, 1998.

[113] O. Tuzel, F. Porikli, and P. Meer. Kernel methods for weakly supervised mean shift clustering. In *ICCV*, pages 48–55, 2009.

[114] O. Tuzel, R. Subbarao, and P. Meer. Simultaneous multiple 3D motion estimation via mode finding on Lie groups. *ICCV*, pages 18–25, 2005.

[115] C. J. van Rijsbergen. *Information Retrieval*. Butterworth, 1979.

[116] N. Vaskevicius, A. Birk, K. Pathak, and S. Schwertfeger. Efficient representation in three-dimensional environment modeling for planetary robotic exploration. *Advanced Robotics*, 24(8-9):1169–1197, 2010.

[117] A. Vedaldi and S. Soatto. Quick shift and kernel methods for mode seeking. In *ECCV*, pages 705–718, 2008.

[118] K. Wagstaff, C. Cardie, S. Rogers, and S. Schrödl. Constrained k-means clustering with background knowledge. In *Intl. Conf. Machine Learning*, pages 577–584, 2001.

[119] J. Wang, B. Thiesson, Y. Xu, and M. Cohen. Image and video segmentation by anisotropic kernel mean shift. In *ECCV*, volume 2, pages 238–249, 2004.

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

[121] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, pages 1601–1608, 2004.

Vita

Saket Anand

Education

2010-2013 Ph.D., Department of Electrical and Computer Engineering, Rutgers University
Advisor: Prof. Peter Meer

2004-2006 M.S., Department of Electrical and Computer Engineering, Rutgers University
Advisor: Prof. Richard Mammone

1999-2003 Bachelor of Engineering, Department of Electronics Engineering, University of Pune, Pune, India

Professional Experience

2007-2010 Research Engineer, Read-Ink Technologies Pvt. Ltd., Bangalore, India

2006-2007 Research Engineer, STAR Technologies, New Brunswick, NJ

Publications

- **S. Anand**, S. Mittal, O. Tuzel, P. Meer, "Semi-Supervised Kernel Mean Shift Clustering", submitted to IEEE Trans. PAMI (submitted Dec.'12).
- **S. Anand**, M. Singh, V. Singh, P. Meer, "Generating 3D models from a Single RGB-D Image", (in preparation)
- S. Mittal, **S. Anand** and P. Meer, "Generalized Projection based M-Estimator,". In preparation for submission to *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.
- S. Mittal, **S. Anand**, P. Meer, "Generalized Projection based M-Estimator: Theory and Applications". In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Colorado Springs, CO*, pages 2689 - 2696, 2011.
- **S. Anand**, D. Madigan, R. Mammone, S. Pathak and F. Roberts, Experimental Analysis of Sequential Decision Making Algorithms for Port of Entry Inspection Procedures, in S. Mehrotra, et al (eds.), Proceedings of Intelligence and Security Informatics, LNCS 3975, Springer-Verlag, New York, 2006.
- C. Podilchuk, A. Patel, A. Harthattu, **S. Anand** and R. Mammone, A New Face Recognition Algorithm based on Bijective Mappings, IEEE Workshop on FRGC Experiments, CVPR 2005.