IMAGE SEGMENTATION USING MARKOV RANDOM FIELD

by

Aparna Tatavarti

A thesis submitted to the faculty of
The University of North Carolina at Charlotte
in partial fulfillment of the requirements
for the degree of Master of Science in
Electrical Engineering

Charlotte

2017

ABSTRACT

APARNA TATAVARTI. Image segmentation using markov random field. (Under the direction of DR. ANDREW WILLIS)

This thesis describes an algorithm for efficient segmentation of point cloud data into local planar surface regions. This is a problem of generic interest to researchers in the computer graphics, computer vision, artificial intelligence and robotics community where it plays an important role in applications such as object recognition, mapping, navigation and conversion from point clouds representations to 3D surface models. Prior work on the subject is either computationally burdensome, precluding real time applications such as robotic navigation and mapping, prone to error for noisy measurements commonly found at long range or requires availability of co-registered color imagery. The approach we describe consists of 3 steps: (1) detect a set of candidate planar surfaces, (2) cluster the planar surfaces merging redundant plane models, and (3) segment the point clouds by imposing a Markov Random Field (MRF) on the data and planar models and computing the Maximum A-Posteriori (MAP) of the segmentation labels using Bayesian Belief Propagation (BBP). In contrast to prior work which relies on color information for geometric segmentation, our implementation performs detection, clustering and estimation using only geometric data. Novelty is found in the fast clustering technique and new MRF clique potentials that are heretofore unexplored in the literature. The clustering procedure removes redundant detections of planes in the scene prior to segmentation using BBP optimization of the MRF to improve performance. The MRF clique potentials dynamically change to encourage distinct labels across depth discontinuities. These modifications provide improved segmentations for geometry-only depth images while simultaneously controlling the computational cost. Algorithm parameters are tunable to enable researchers to strike a compromise between segmentation detail and computational performance. Experimental results apply the algorithm to depth images from the NYU depth dataset which indicate that the algorithm can accurately extract large planar surfaces from depth sensor data.

ACKNOWLEDGEMENTS

With all humility I would like to express my sincere gratitude to my guide Dr. Andrew Willis who showed the way and direction to complete my thesis. This thesis wouldn't have been possible without the help of Dr. Willis. I would also like to thank my co-worker John Papadakis for helping me throughout my journey as a thesis student. Also, I would like to thank the committee members Dr. Hamed Tabhki and Dr. Tao Han for their encouragement.

I would like to thank my family and friends for their support, especially Telwin George.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

BBP	An acronym for Bayesian Belief Propagation
ECE	An acronym for Electrical and Computer Engineering.
ICA	An acronym for Independnet Component Analysis
ICM	An acronym for Iterative Conditional Modes
ICM	An acronym for Iterative Conditional Modes
MAP	An acronym for Maximum A-Posteriori
MRF	An acronym for Markov Random Field
PCA	An acronym for Principal Component Analysis

CHAPTER 1: INTRODUCTION

Images provide richly detailed summaries of complex, dynamic environments. Scene understanding is one of the most essential functionalities of human vision and also a major goal of computer vision research. Using computer vision systems, we can detect and recognize objects, track their motion, or infer three–dimensional (3D) scene geometry. Due to the wide availability of digital cameras, these methods are used in a huge range of applications, including human–computer interfaces, robot navigation, medical diagnosis, visual effects, multimedia retrieval, and remote sensing.

While humans can perform complex tasks such as to recognizing and distinguishing objects of different size, color and shape from images with ease, doing the same task on a robot is quite challenging. Robots use sensors to measure environment. Programs estimate scene structures from these measurements. Factors such as illumination, background, viewpoint, camera parameters, and camera location effects the image formation process which makes the recognition task by the computers difficult. Within the image the task becomes quickly intractable for modern computers when there are many objects present in the scene of interest. Hence, efficient algorithms are needed to perform complex tasks of object recognition from sensor data with greater accuracy.

Most robotic object recognition algorithms label image data to a collection of object classes. An object model database contains a collection of object classes. The algorithms recognize objects using the object model database. Feature based approaches extract features from images of objects and uses these features as attributes to describe and recognize each object of interest. Common features for recognition are edges, contours of the object, surfaces, and corners. This step seeks to reduce the complexity of the object representation which reduces dimension of the search space while simultaneously preserving the most distinctive attributes needed for recognizing the object. Searching feature space is less computationally complex. Each collection of features must be compared with representative values that the feature assumes for each object class to search for the correct label to assign for the observed feature collection.

This thesis describes about a preprocessing technique (segmentation) which has the ability to simplify, interpret and analyze information from images. The goal of the thesis is to perform segmentation of point cloud data into local planar surface regions. This is a problem of generic interest to researchers in the computer graphics, computer vision, artificial intelligence and robotics community where it plays an important role in applications such as object recognition, mapping, navigation and conversion from point clouds representations to 3D surface models. Prior work on the subject is either computationally burdensome, precluding real time applications such as robotic navigation and mapping, prone to error for noisy measurements commonly found at long range or requires availability of co- registered color imagery. The approach we describe consists of 3 steps: (1) detect a set of candidate planar surfaces, (2) cluster the planar surfaces merging redundant plane models, and (3) segment the point clouds by imposing a Markov Random Field (MRF) on the data and planar models and computing the Maximum A-Posteriori (MAP) of the segmentation labels using Bayesian Belief Propagation (BBP). In contrast to prior work which relies on color information for geometric segmentation, our implementation performs detection, clustering and estimation using only geometric data. Novelty is found in the fast clustering technique and new MRF clique potentials that are heretofore unexplored in the literature. The clustering procedure removes redundant detections of planes in the scene prior to segmentation using BBP optimization of the MRF to improve performance. The MRF clique potentials dynamically change to encourage distinct labels across depth discontinuities. These modifications provide improved

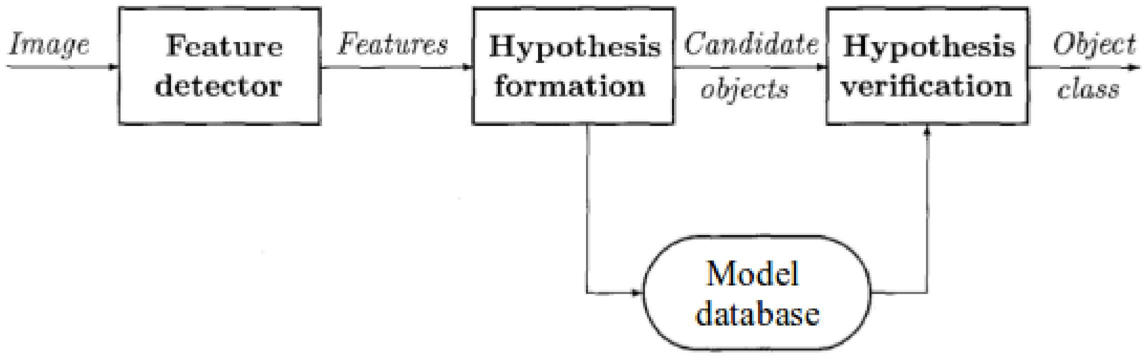


Figure 1.1: Object recognition system

segmentations for geometry-only depth images while simultaneously controlling the computational cost.

The following section discusses the working of an object recognition system and it's relation with our segmentation algorithm.

1.1 Object Recognition System

An object recognition system consists of the following components as discussed in [4]:

- Model database
- Feature detector
- Hypothesizer
- Hypothesis verifier

Figure 1.1 shows the block diagram of object recognition system. The goal of object recognition system is to identify a group of pixels or a region in an image that correspond to an object. A feature contains information that can be helpful in identifying and classifying a group of pixels as an object. It also plays a key role in distinguishing, describing and recognizing an object in relation to other objects. Size, color, and shape are some commonly used features. The feature detector identifies locations of features of the object in the image/scene of interest that help in forming object hypotheses. Using the detected features, each object present in the scene are assigned a likelihood value based on the objects present in the model database. Initially, model database contains all the possible objects in the input image. After feature detection step, the model database consists of only objects that match the feature detection criteria. The verifier then uses object models to verify the hypotheses and refines the likelihood of objects. The system then selects the object with the highest likelihood as the correct object. Many vision problems can be posed as a labeling problem in which the solution to a problem is a set of labels assigned to image pixels or features. In our project, the segmentation problem can be stated as a labeling problem. In the case of an object recognition system we solve the labeling problem by assigning a label to every object present in the image. Our project model database consists of distinct planes having different orientations. As any indoor scene can be best represented with a set of planes, we assign a label (which is a plane) from the model database to each surface data in a point cloud which is acquired by an RGBD sensor. The following section discusses the various components in the object recognition system.

1.1.1 Feature detection:

We assume that a region or a closed boundary corresponds to an entity that is either an object or a part of an object. Feature detection using features based on regions or boundaries in an image, have the higher chance of recognizing the object. In our project, the features are the set of planes obtained after clustering process.

1.1.1.1 Global features

Global features usually are some characteristics of regions in images such as area, perimeter, Fourier descriptors, and moments. Global features can be obtained either for a region by considering all points within a region, or only for those points on the boundary of a region. The characteristics that represent global features can be all the points, their locations, intensity and spatial relations [5]. The most important global features are those extracted by dimension reduction approaches such as Principle Component Analysis (PCA) [6], Independent Component Analysis (ICA) [7]. Global features have the advantages of being able to capture the holistic information of the object/image and usually being computationally efficient, but they are weak for representing objects with significant background clutters, occlusions and intra-class variations. The global features in our project are all the planes that share many surface points.

1.1.1.2 Local features

Local features are powerful as the representation is invariant to image transformations and viewpoint changes while their locality ensures a degree of robustness to occlusion, non uniform illumination and object deformations. Local features are usually on the boundary of an object or represent a distinguishable small area of a region. Curvature and related properties are commonly used as local features. The curvature may be the curvature on a boundary or may be computed on a surface. Local features can contain a specific shape of a small boundary segment or a surface patch. Common local features are curvature, boundary segments, and corners. In our project, the local features are the planes assigned to each surface point.

1.1.1.3 Relational features

Relational features are based on the relative positions of different entities, either regions, closed contours, or local features. These features usually include distance between features and relative orientation measurements. These features are very useful in defining composite objects using many regions or local features in images. In most cases, the relative position of entities is what defines objects [4].

1.1.2 Recognition methods

The next step in the process of object recognition is to apply recognition methods on the detected features in an image. The common recognition methods are classification approach, matching and indexing. For our project, the recognition method groups all the surface points that have similar depth values and classifies the group to a plane.

1.1.2.1 Classification

The basic idea in classification is to identify all pixels that share group similar characteristics such as intensity, texture , color and group them into classes.

Nearest neighbor classifiers

Similar detected features are grouped into classes. To decide the class of the object, we measure its similarity with each class by computing its distance from the points representing each class in the feature space and assign it to the nearest class.

Bayesian approach

A Bayesian approach is efficient in recognizing objects when there is a significant overlap in feature values of different objects. The Bayesian approach uses probabilistic knowledge about the features for objects and the frequency of the objects. Suppose that we know that the probability of objects of class j is $P(w_j)$ [8], which is called the prior term. Decisions about the class of an object are usually made based on feature observations. Suppose that the probability $P(x|w_j)$ is given and is called the likelihood term. Based on this knowledge, we can compute the posteriori probability for the object. The a-posteriori probability is the probability $P(w_j|x)$ defined as a product of likelihood and prior.

$$P(w_j|x) = \frac{P(x|w_j)P(w_j)}{P(x)} \tag{1.1}$$

1.1.2.2 Matching

Classification approaches use effective features and knowledge of the application. In most of the applications priori knowledge of the features is not known before hand to design a classifier. Direct matching of the model to an unknown object can be effective method. Matching is done between objects present in the scene of the image and the object models available in the model database. Each model in the database are fit to the objects present in the image. In our project, we decompose the image into tiles. We fit all the surface data points in each tile with a plane that best minimizes the fitting error. The fitting error is measured as the perpendicular squared Euclidean distance between the measurements and the plane.

1.1.2.3 Feature indexing

The indexing technique is efficient when the number of objects in the image is large. In contrast, the matching approach is a sequential approach and requires to compare the object in the image with the models. This sequential nature of the approach makes it unsuitable with a large number of objects. Feature indexing approaches use features of objects to structure the model database. When a feature from the indexing set is detected in an image, this feature is used to reduce the search space. More than one feature from the indexing set may be detected and used to reduce the search space and in turn reduce the total time spent on object recognition.

1.1.3 Verification methods

The verification step in the object recognition task finds how many times and where a given object appears in the image of interest. In our project, the verification step is to re-validate each surface data point that lies on a plane by looking at the likelihood of the surface data point associated with the plane.

1.1.3.1 Template matching

A template is an image of an object in the model database. Template matching detects the presence of given object in an image by placing the template at a location in an image and comparing intensity values in the template with the corresponding values in the image. A measure of dissimilarity between the intensity values of the template and the corresponding values of the image is given by the sum of squared errors difference. Let $f(i,j)$ be the template and $g(i,j)$ be the image, sum of squared difference. A reasonable strategy for obtaining all locations and instances of the template is to shift the template and use the match measure at every point in the image [4]. Thus, for an m xn template, match measure M is given by

$$M\left[i,j\right] = \sum\sum g\left[k,l\right] f\left[i+k,j+l\right] \tag{1.2}$$

where k and l are the displacements with respect to the template of the image.

CHAPTER 2: Camera Calibration

The primary role of a camera is to produce a digital image of a captured real world scene that can be stored on a computer. Recently developed RGBD sensors have become popular for measuring both scene appearance and geometry. These sensors provide high resolution visual data in a low cost (~$200USD) and compact package. RGBD sensors combine a traditional color camera (RGB) with an infrared depth sensor (D), and merge this data to produce HD color+range images at real-time frame rates (~30 fps). Unlike traditional 2D cameras, RGBD sensors provide depth measurements that directly impart a sense of scene geometry, without the use of techniques such as stereoscopic reconstruction. These sensors have seen increased use as research tools for many computer vision related problem. In our project, we wish to extract information from depth images captured by RGBD sensors.

The volume of data produced by such sensors is quite large. As such, real-time systems must carefully consider the computational cost of algorithms that process this data. This is particularly important for time sensitive tasks such as visual odometry-based navigation, which relies on incoming sensor data to navigate geometrically complex scenes.

With more complex camera systems, errors resulting from misaligned lenses and deformations in their structures can result in more complex distortions in the final image. Hence, camera calibration is an important step in correcting the distortions to achieve accurate representation of the real world scene in captured images. It's objective is to determine a set of camera parameters that describe the mapping between 3-D surface points viewed by the camera and their projection into the sensed 2D image. The mathematical model that characterizes this process uses a collection

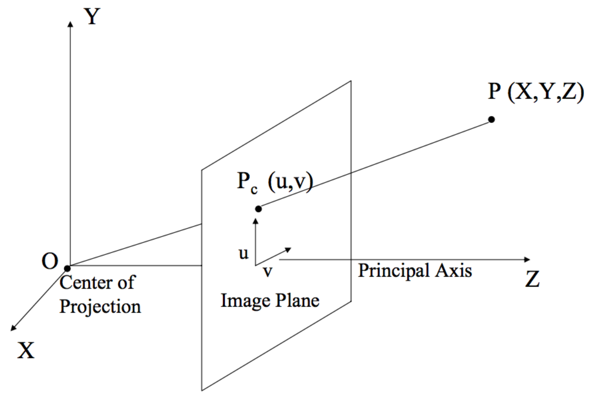


Figure 2.1: Pinhole Camera Model [1]

of parameters that are subdivided into internal and external parameters of the camera [9].

Intrinsic parameters define the internal geometric and optical characteristics of the camera. These parameters encompass focal length, image sensor format, and principal point.

Extrinsic parameters which denote the coordinate system transformations from 3D world coordinates to 3D camera coordinates. The extrinsic parameters define the position and orientation of the camera within an arbitrary defined 3D coordinate system.

2.1 Pinhole Camera

A pinhole camera is the simplest, and the ideal camera model without a lens but with a tiny aperture, a pinhole. When a light from a scene passes through the aperture, projects an inverted image on the opposite side of the box, which is known as the camera obscura effect. This process reduces the dimensions of the data taken in by the camera from three to two (light from a 3D scene is stored on a 2D image). In an ideal pinhole camera, a simple projection matrix is enough to represent the scene on image plane.

Figure 2.1 depicts the pinhole camera model [1] . The camera is placed at the origin O. A 2D point is denoted by$P_c\,(u,v)$,$m = [u,v]^T$. A 3D point is denoted by

$P\left(X,Y,Z\right)$, $M = [X,Y,Z]^T$.

In the figure 2.1, we want to estimate $P_c\left(u,v\right)$ from $P\left(X,Y,Z\right)$ where f is the focal length and translation of origin is defined by (t_u, t_v)

$$u = \frac{fX}{Z} + t_u \tag{2.1}$$

$$v = \frac{fY}{Z} + t_v \tag{2.2}$$

For a rectangle pixels with resolution m_u and m_v pixels/inch in u and v direction respectively. Therefore, to measure Pc in pixels, its u and v coordinates should be multiplied by m_u and m_v respectively

$$u = m_u\frac{fX}{Z} + m_u t_u$$

$$v = m_v\frac{fY}{Z} + m_v t_v$$

This can be expressed as:

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} m_u f & 0 & m_u t_u \\ 0 & m_v f & m_v t_v \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{2.3}$$

$$K = \begin{pmatrix} \alpha & \gamma & u_0 \\ 0 & \beta & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

K is called the camera intrinsic matrix, with u_0, v_0 the coordinates of the principal point, α, β are the scale factors in image u and v axes, and γ the parameter describing the skew of the two image axes.

When the camera is oriented in an arbitrary fashion (not necessarily z perpendicular to the image plane), then we need a rotation and translation to make the camera coordinate system coincide with the configuration in Figure 2.1. The camera translation to origin of the (X, Y, Z) coordinate be given by $T = (T_X, T_Y, T_Z)$. The rotation applied to coincide the principal axis with Z axis be given by a 3×3 rotation matrix R. Then the matrix formed by first applying the translation followed by the rotation is given by the 3×4 matrix is shown in equation (2.5)

$$E = (R|RT) \tag{2.5}$$

where$E$ is the extrinsic parameter matrix.

A camera is modeled by the usual pinhole: the relationship between a 3D point $P = (X, Y, Z)$ and its image projection P_c

$$P_c = CP \tag{2.6}$$

where $C = KE$

2.2 Experimental Procedure and Results

Calibration for the color camera of the Microsoft Kinect v2 was performed using openCV [10]. A calibration board with an image of checkerboard pattern was used as a calibration target. The positions of pattern corners with respect to the coordinate system of the target are known. An RGB camera was positioned in front of the board which captures and finds the locations of the corners on the calibration board (checkerboard pattern). For each image, a subset of the checkerboard pattern is extracted by software. Equations for calculation of camera parameters were obtained from extracted patterns. Obtained results are compared to default values. It is observed that the obtained results and the default results are in good correlation.

In our project, using the obtained calibration parameters we calculate the 3D (X, Y) values.

The distortion matrix D is

$$D = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

The computed intrinsic matrix K is

$$K = \begin{pmatrix} 567.837 & 0 & 319.5 \\ 0 & 567.837 & 239.5 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.7}$$

The projection matrix P is

$$P = \begin{pmatrix} 567.837 & 0 & 319.5 & 0 \\ 0 & 567.837 & 239.5 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The (X, Y) values are computed by the substituting the values of $f_x, f_y, u_0, v_0, \delta_x, \delta_y$ in the following equations. Measured 3D (X, Y, Z) positions of sensed surfaces can be directly computed from the intrinsic RGBD camera parameters and the measured depth image values. The Z coordinate is directly taken as the depth value and the (X, Y) coordinates are computed using the pinhole camera model. In a typical pinhole camera model, 3D (X, Y, Z) points are projected to (x, y) image locations, e.g., for the image columns the x image coordinate is $x = f_x \frac{X}{Z} + u_0 - \delta_x$. However, for a depth image, this equation is re-organized to “back-project” the depth into the 3D scene and recover the 3D (X, Y) coordinates as shown by equation (2.8)

$$\begin{aligned} X &= (x + \delta_x - u_0)Z/f_x \\ Y &= (y + \delta_y - v_0)Z/f_y \\ Z &= Z \end{aligned} \tag{2.8}$$

where Z denotes the sensed depth at image position (x, y), (f_x, f_y) denotes the camera focal length (in pixels), (u_0, v_0) denotes the pixel coordinate of the image center, i.e., the principal point, and (δ_x, δ_y) denote adjustments of the projected pixel coordinate to correct for camera lens distortion [11].

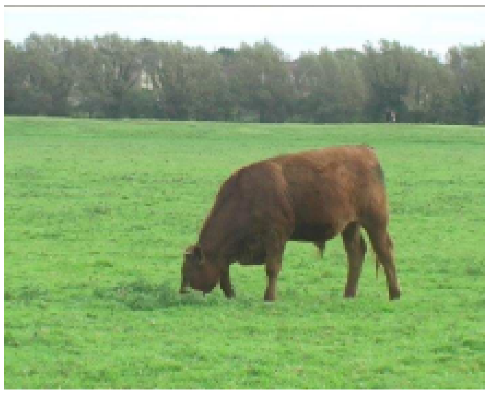
CHAPTER 3: Image Segmentation

Robots and computers use devices such as RGBD sensors, cameras , 3D Laser scanners, etc to capture the real-world scene of interest. Manipulating images obtained from these sources is the next immediate step before applying methods to solve the classical problems in computer vision such as object recognition, image restoration, scene reconstruction etc. Segmentation is one of the commonly used image processing technique which simplifies the representation of a complex raw data into a form that is more meaningful and easier to analyze by a computer.

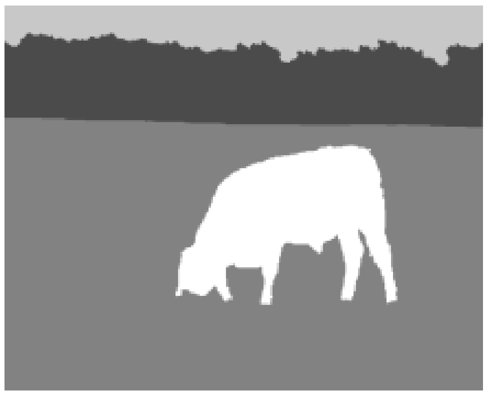
Image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain characteristics such as color, intensity, texture etc.

Figure 3.1 consists of an input color image and output image obtained after applying segmentation algorithm on the given color image. The pixels in the given input color image are classified based on similar intensity values. The output segmented image approximates the input image with 4 different labels.

Figure 3.2 contains gray scale, depth and color images of the captured scene of interest. The image segmentation algorithm groups all the pixels having similar depth values and represents them with a different plane. In this example, all pixels having similar depth values are assigned to a label which is a plane. Reconstructing 3D indoor environments, however, remains challenging due to the cluttered spaces, extensive variability, and the real-time constraint. Furthermore, as an application point of view, perceiving the geometry of surrounding structures is very important for indoor environments. It is a valid assumption to consider, on the average, up to 95% of indoor-environment structures consist of planar surfaces [12]. This thesis discusses

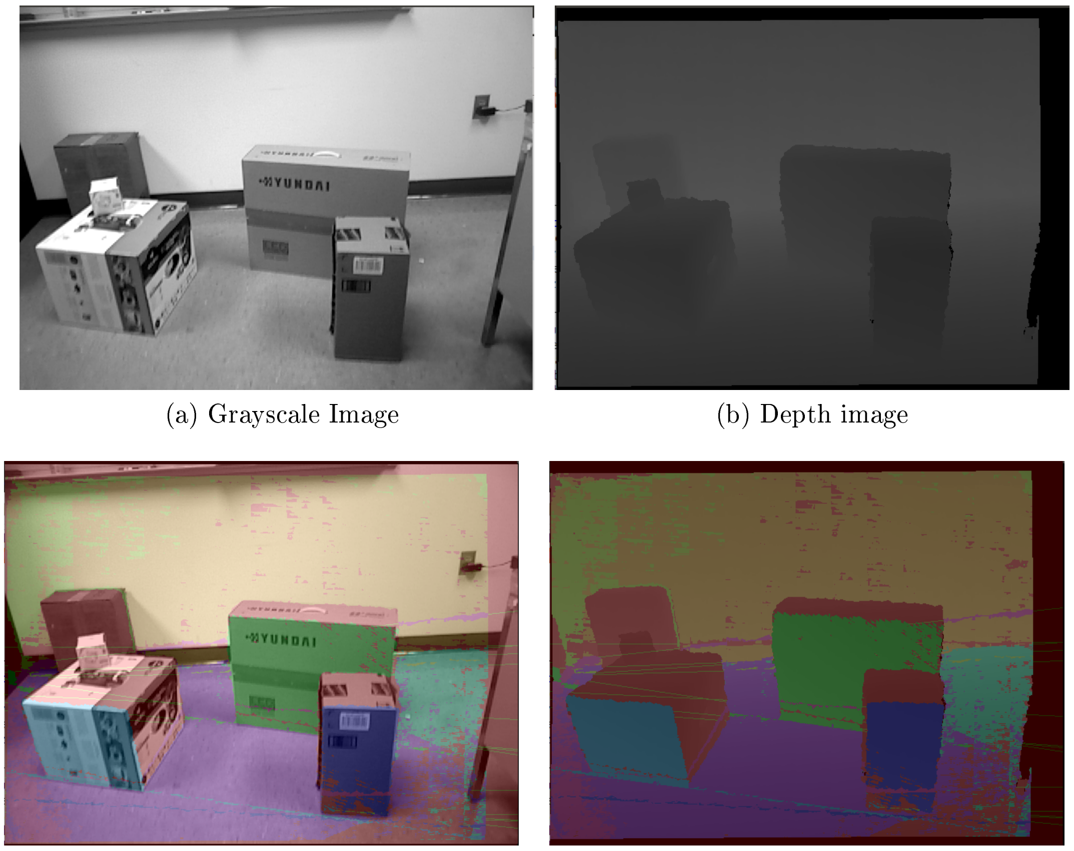


(a) Input color image



(b) Segmented Image

Figure 3.1: Illustration of Image Segmentation with an example [2]



(a) Grayscale Image

(b) Depth image

(c) Color image

(d) Segmented image

Figure 3.2: Illustration of planar segmentation with example [3]

about working of an efficient segmentation algorithm that segments indoor scenes into planes.

The following section discusses different segmentation methods.

3.1 Image Segmentation Methods

There are several methods that perform image segmentation, but the following are the methods of our interest [13] -

- Clustering-based
- Region-growing methods
- Histogram-based methods
- Graph-based

3.1.1 Clustering-based segmentation

Clustering is a process of grouping the pixels in the image based on color, intensity or texture. The K-means algorithm is an iterative technique that partitions the image into K clusters. Initially, the algorithm assigns the labels randomly. The algorithm seeks to minimize the distance between each pixel in the cluster and the cluster center. The algorithm converges after several iterations upon finding the minimum distance between the cluster and the cluster center. The algorithm reassigns the labels in the image, after convergence [14].

3.1.2 Region-growing methods

Region-growing methods rely mainly on the assumption that the neighboring pixels within one region have similar values. The common procedure is to compare one pixel with its neighbors. If a similarity criterion is satisfied, the pixel can be set to belong to the cluster as one or more of its neighbors [15]. The methods to classify a pixel to a region are discussed below.

Seeded region growing method

The segmentation is performed based on considering a set of seeds as input along with the image. The regions are iteratively grown by comparing the pixel's intensity value with the seed. The difference between a pixel's intensity value and the region's mean, δ is used as a measure of similarity. The pixel with the smallest difference measured in this way is assigned to the respective region. This process continues until all pixels are assigned to a region. The segmentation results by this approach are highly dependent on choosing the right set of seeds as input. Poor selection of seeds can result in noisy segmented images.

Unseeded region growing method

It is a modified algorithm that does not require explicit seeds. Initially a single region A_1 is chosen. The region size grows after every iteration similar to seeded region growing method. Let A_i be the region after i^{th} iteration. If the difference between a pixel's intensity value and the region's mean, δ is less than predefined threshold T, the pixel is added to the current region. If not, then the pixel is considered different from all current regions A_i and a new region is created with this pixel.

Split and merge

It is based on a quadtree partition of an image. It is sometimes called quadtree segmentation. This method starts at the root of the tree that represents the whole image. The image is split into four child regions if the pixels in the image are non-uniform (splitting process). Again, each non-uniform region is sub-divided into four child regions. The splitting process stops once all the child regions are uniform. Later, all the uniform child regions are merged as several connected components (the merging process). This process continues recursively until no further splits or merges are possible.

3.1.3 Histogram based method

In this technique, a histogram is computed from all of the pixels in the image, and the peaks and valleys in the histogram are used to locate the clusters in the image. Color or intensity can be used as the measure. A refinement of this technique is to recursively apply the histogram-seeking method to clusters in the image in order to divide them into smaller clusters. This operation is repeated with smaller and smaller clusters until no more clusters are formed. The histogram can also be applied on a per-pixel basis where the resulting information is used to determine the most frequent color for the pixel location [16] [17].

3.1.4 Graph-based segmentation

Graphical models provide a simple way to formulate and solve complex problem statements. These models are powerful tools in analyzing and estimating the relationship between the known and the unknown quantities. A graph comprises of a set of nodes (also called vertices) and a set of links (also known as edges) connecting the nodes. Each node represents a random variable (or group of random variables), and the links express probabilistic relationships between these variables. The graph then captures the way in which the joint distribution over all of the random variables can be decomposed into a product of factors each depending only on a subset of the variables [18].

Types of Graphical models:

- Directed graphical models
- Undirected graphical models

The following section discusses in detail about the different types of graphical models.

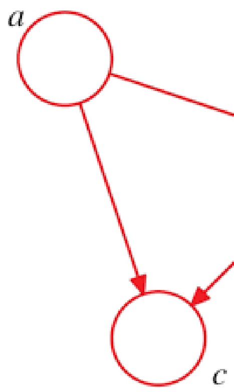


Figure 3.3: A directed graphical model representing the joint probability distribution over three variables a, b, and c

3.1.4.1 Directed graphical models

The links of the graph have a particular directionality which are useful for expressing causal relationships between random variables. Bayesian networks is an example of directed graphical model.

Bayesian networks

Figure 3.3 is a directed graphical model where a, b and c be three random variables having a causal relation [18] . The links in the graph indicate the causal relationship between the random variables. Node a has no incoming links, information at node b depends on node a and information at node c depends on both nodes a, b.

The joint distribution over a, b and c is given by

$$p(a, b, c) = p(c|a, b)p(b|a)p(a) \tag{3.1}$$

Joint distribution over K variables is given by $p(x_1, ...x_k)$ where x is a random variable. By repeated application of the product rule of probability, this joint distribution can be written as a product of conditional distributions, one for each of the variables

$$p(x_1, ..., x_K) = p(x_K|x_1, ...x_{K-1})...p(x_2|x_1)p(x_1) \tag{3.2}$$

3.1.4.1 Undirected graphical models

The undirected graphical model has a set of nodes each of which corresponds to a variable or group of variables, as well as a set of links which are undirected (do not carry arrows) each of which connects a pair of nodes. Markov Random Fields is an example of undirected graphical model. The next chapter discusses about Markov Random Field for images.

CHAPTER 4: Markov Random Field for Images

Bayes networks or, more generally, Markov Random Field (MRF) models, are probabilistic graphical models known for their ability to provide robust and accurate solutions to generic image segmentation problems[19]. MRF are undirected graphical models that can encode spatial dependencies.

A discrete Markov Random Field gives relation between unknown quantities and known quantities. In our project, we use Markov Random Field model to estimate the best label from L set of labels for every surface point.

MRF is defined as a graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. For images we construct a Markov Random Field by associating with each pixel (i, j) with a vertex $(i, j)\epsilon V$ and a random variable $X_{i,j}$, which can take values in the label set L. The sample space X^N is the set of all N-dimensional vectors

$$X = \{X_{i,j}|(i,j)\epsilon V\} \tag{4.1}$$

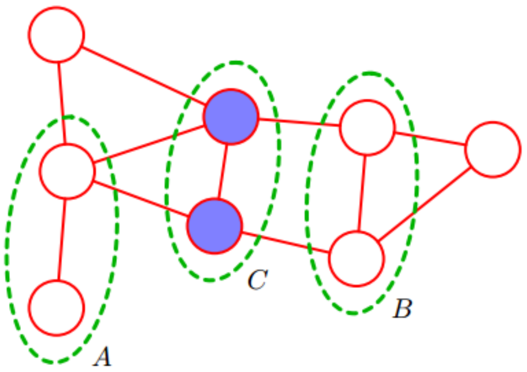


Figure 4.1: An example of an undirected graph in which every path from any node in set A to any node in set B passes through at least one node in set C

with components lying in L and the cardinality of that space is equal to

$$X^N = |L|^N \tag{4.2}$$

Also, the edges E of the underlying graph G represent probabilistic dependencies between random variables. A Markov Random Field thus defines a probability distribution, which assigns to each vector X in the sample space a probability mass $p(X)$. However, not all distributions are allowed by a Markov Random Field. A valid distribution $p(X)$ should respect the probabilistic dependencies implied by the graph edges. More specifically, the following definition holds [20]:

1. The random variables $X = \{X_{ij}\}_{(i,j)\epsilon V}$ are said to form a Markov Random Field with graph G, according to figure 4.1, if whenever the sets A and B are separated in the graph G by a set C then the random variables X_A,X_B are conditionally independent given the variables X_C

$$p(X_A, X_B|X_C) = p(X_A|X_C)p(X_B|X_C) \tag{4.3}$$

In the above definition A, B, C represent arbitrary subsets of nodes in V while the notation X_A denotes all random variables corresponding to nodes included in the set A. The set of nodes C separates the sets A and B, if for any path in the graph G starting from A and ending in B, that path necessarily passes through at least one node belonging to C. This definition of a Markov Random Field is actually a generalization of one dimensional Markov Processes. From the above definition it can be understood that the past and the future observations are conditionally independent given the present observations. Based on the above definition, we also see that the role of the graph G is to act as a kind of filter for the allowed distributions: only those distributions which manage to pass all the conditional independence tests implied by the graph G make up the family

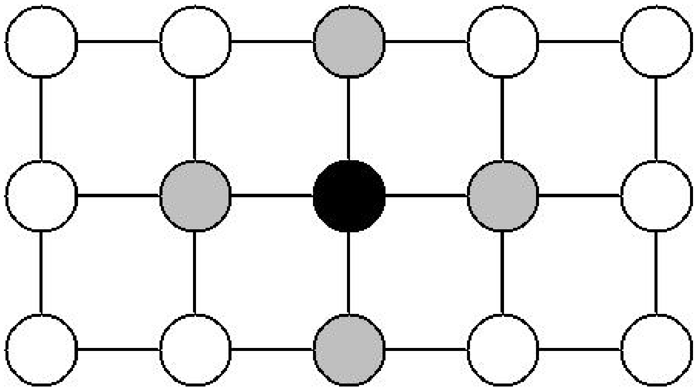


Figure 4.2: A site X_{ij} in a 4-connected MRF lattice is shown (black) and the neighbors of the site, N_{ij}, are also shown (gray). The Markov property states that the distribution on X_{ij} is only a function of those random variables that are in the neighborhood N_{ij}.

of MRF distributions.

2. The Markov property states that the probability distribution of a site given values for the all other elements of the field is equivalent to the probability distribution obtained given only the values of those variables in the neighborhood of the site; expressed mathematically in equation 4.4 [21]. As shown in figure 4.2, $\mathcal{N}_{ij}$ denotes the set of random variables that are members of the 4-connected neighborhood of X_{ij}.

$$p(X_{ij}|X - X_{ij}) = p(X_{ij}|X_{nm} \in \mathcal{N}_{ij}) \tag{4.4}$$

The factorization rule for undirected graphs that will correspond to the conditional independence test is defined by expressing the joint distribution $p(X)$ as a product of functions defined over sets of variables that are local to the graph. If we consider two nodes X_i and X_j that are not connected by a link, then these variables must be conditionally independent given all other nodes in the graph. This follows from the fact that there is no direct path between the two nodes, and all other paths pass through nodes that are observed, and hence those paths are blocked. This conditional independence property can be expressed as

$$p(X_i, X_j|X_{/\{i,j\}}) = p(X_i|X_{/ij})p(X_j|X_{/ij}) \tag{4.5}$$

where $X_{/ij}$ denotes the set X of all variables with X_i and X_j removed. The factorization of the joint distribution must therefore be such that X_i and X_j do not appear in the same factor in order for the conditional independence property to hold for all possible distributions belonging to the graph.

The distribution of entire field of variables, $X = \{\cup_{ij} X_{ij}\}$, represents all possible segmentations of the data and is expressed as shown in equation (4.6)

$$p(X) = \frac{1}{Z}e^{-U(X)} \tag{4.6}$$

where Z denotes the partition function that ensures the probability integrates to 1 and $U(X)$ is referred to as the energy function for the field.

The energy function $U(X)$ generally consists of two parts:

(1) a *data likelihood* term, $D(Z(i,j)|X_{ij} = l)$, and

(2) a *smoothness prior,* $V(X_{ij}|X_{nm} \in \mathcal{N}_{ij})$.

The *data likelihood* term finds out and assigns a label with low data cost value. Smoothness prior term enforces smooth labeling among connected nodes.

The *smoothness prior* term assigns the adjacent labels that are similar to low data cost and a higher value to the adjacent labels that are different.

Three different ways of assigning smoothness cost to the labels as discussed in [22] -

Potts model

If the difference between the labels is zero, the smoothness cost is given a value of zero. If the adjacent labels have a different value, then the smoothness cost is given a value λ. Figure 4.3 shows the graph plotted between the smoothness cost term of neighboring nodes X_i, X_j and corresponding labels l_i, l_j assigned to nodes.

$$V(X_{ij} = l_i | X_{nm} = l_j) = \begin{cases} \lambda & if \quad l_i \neq l_j \\ 0 & if \quad l_i = l_j \end{cases} \tag{4.7}$$

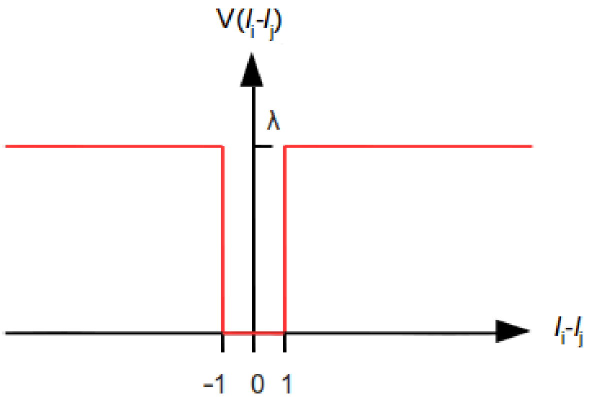


Figure 4.3: Potts model

Truncated Linear Model

The cost function V is based on the magnitude of the difference between two labels. The cost increases linearly based on the distance between the labels l_i, l_j up to some level. In order to allow for large discontinuities in the labeling the cost function stops growing after the difference becomes large. c is the rate of increase in the cost, and d controls when the cost stops increasing. Figure 4.4 depicts behavior of equation 4.8.

$$V(X_{ij} = l_i | X_{nm} = l_j) = min(c|l_i - l_j|, d) \tag{4.8}$$

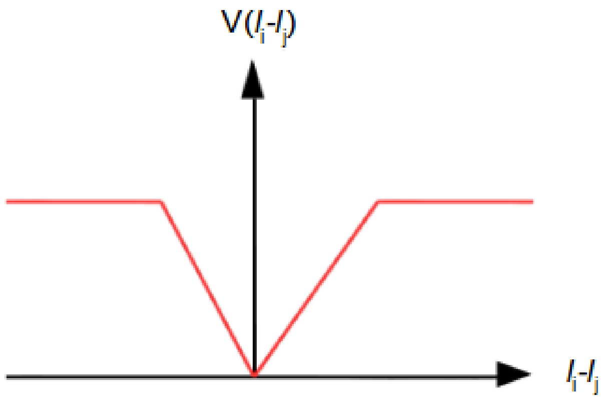


Figure 4.4: Truncated Linear Model

Truncated Quadratic model

The cost grows proportionally to $(l_i - l_j)^2$ up to some level and then becomes a constant thereafter. Figure 4.5 depicts behavior of equation 4.9.

$$V(X_{ij} = l_i | X_{nm} = l_j) = min(c((l_i - l_j)^2, d) \tag{4.9}$$

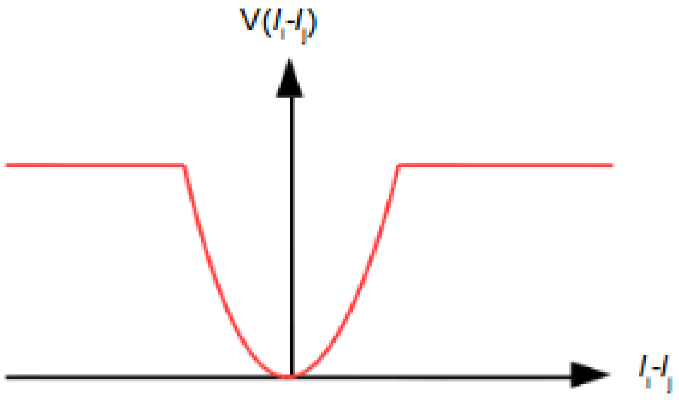


Figure 4.5: Truncated Quadratic Model

Figure 4.6 graphically depicts these relations and makes clear that measured data at each grid location impacts individual sites (in gray) while stochastic relationships between site labels are shown on the lattice (in white). The gray nodes are the observed variables and the white nodes are the latent variables. The links between each node represents dependency. For example, the center white node is connected to 4 latent nodes and one observed node. Hence, calculating the value of the center white node depends only on these 5 nodes that exist in the neighborhood.

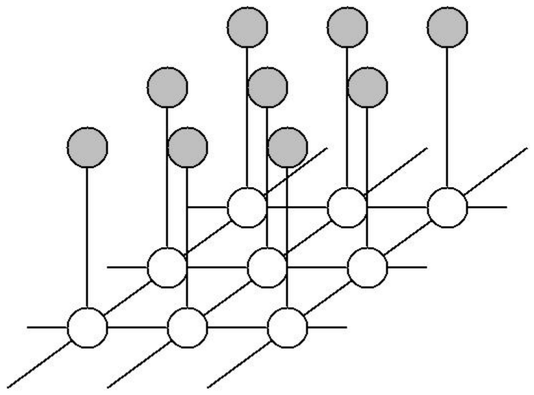


Figure 4.6: Energy in a MRF consists of two stochastic dependencies (shown as edges): (1) those that exist between the site labels and measured data (shown in gray) and (2) those existing between the site and neighboring site on the grid lattice (shown in white).

4.1 Optimization algorithms for Markov Random Field

An optimization problem is one that involves finding the extremum of a quantity or function. Optimization in an MRF problem involves finding the maximum of the joint probability over the graph, usually with some of the variables given by some observed data. MRF segmentation models seek to find the collection of label values that maximize $p(X)$. However, since $p(X)$ is an exponential distribution, the maximizer of $p(X)$ also minimizes the energy function $U(X)$. The optimization algorithms focus on finding a global minimum of the energy function $U(X)$ [23].

List of optimization algorithms as discussed in [24] :

- Iterated Conditional Modes (ICM)
- Graph cuts
- Loopy Bayesian belief propagation
- Tree-reweighted message passing

Each of these methods are discussed in the following subsections.

4.1.1 Iterated Conditional Modes (ICM)

ICM algorithm estimates the labels of each pixel by minimizing the energy value given the current values for all variables in its neighborhood. At the end of an iteration, the new values for each variable become the current values, and the next iteration begins.The algorithm iterates over each node in the graph till convergence.

4.1.2 Graph cuts

The two most popular graph cuts algorithms are

Swap move algorithm

For a pair of labels α, β, a swap move takes some subset of the pixels currently given the label α and assigns them the label β, and vice-versa. The swap move algorithm finds a local minimum such that there is no swap move, for any pair of labels α, β that will produce a lower energy labeling.

Expansion move algorithm

In expansion move algorithm, for a label α the algorithm increases the set of pixels that are given this label. The expansion move algorithm finds a local minimum such that no expansion move, for any label α, yields a labeling with lower energy.

These algorithms rapidly compute a local minimum by trying to minimize the energy.

4.1.3 Loopy Bayesian belief propagation

Belief propagation is an iterative algorithm, which works by continuously propagating local messages between the nodes of the MRF graph. At every iteration, each node sends messages to all of its neighboring nodes, while it also accepts messages from these nodes. This process repeats until all messages stabilize, i.e. they do not change any more. It tries to find a MAP estimate by iteratively solving a finite set of equations until a fixed point is obtained [25][26].

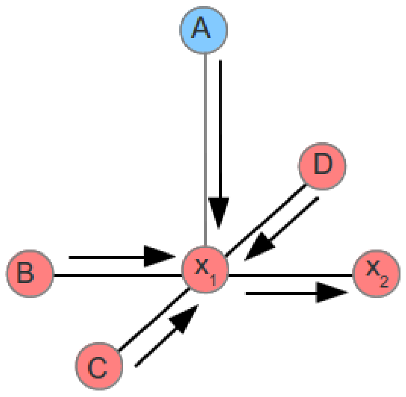


Figure 4.7: Message passing diagram in graph. The blue node A is the observed node , the pink node B, C, D,x_1,x_2 are the hidden nodes.

Figure 4.7 depicts the message passing for a given site [22]. The message from node x_1 is passed to node x_2 , only after node x_1 receives messages from the adjacent nodes A, B, C, D. Each message is a vector of dimension given by the number of possible labels, l. Let $m^t_{x_1 \to x_2}$ be the message that node x_1 sends to a neighboring node x_2 at iteration t. When using negative log probabilities all entries in $m^0_{x_1 \to x_2}$ are initialized to zero, and at each iteration new messages are computed in the following way [27],

$$m^t_{x_1 \to x_2}(l_j) = min\left(V(l_i - l_j) + D_{x_1}(l_i) + \sum m^{t-1}_{s \to x_1}(l_i)\right) \tag{4.10}$$

where $s \epsilon N(x_1)|x_2$ denotes the neighbors of x_1 other than x_2. After T iterations a belief vector is computed for each node by equation 4.11.

$$b_{x_2}(l_j) = D_{x_2}(l_j) + \sum m^T_{x_1 \to x_2}(l_j) \tag{4.11}$$

Finally, the label l^1_j that minimizes $b_{x_2}(l_j)$ individually at each node is selected.

The meaning of belief $b_{x_2}(l_j)$ is that it expresses how likely node x_2 thinks that label l_j should be assigned to x_2. This will depend on two things:

- node x_2 must consider the observed data at that node

- node x_2 must also consider the advice given by all of its neighbors about label

l_j, which is given by the sum term$\sum m_{s\to x_1}^{t-1}\left(l_j\right)$

Based on the above observations, once all beliefs have been computed, each node is then assigned the label having the maximum belief:

$$l_j^1 = argmax\left(b_{x_2}\left(l_j\right)\right)_{x_2} \tag{4.12}$$

Beliefs approximate the max-marginals i.e. each belief l_j approximates the maximum conditional probability that can be obtained given the fact that node x_2 has been already assigned the label l_j.

4.1.4 Tree-reweighted message passing

Tree-reweighted message passing is a message-passing algorithm similar, on the surface, to the loopy belief propagation. Let $m_{x_1\to x_2}^t$be the message that pixel x_1 sends to its neighbor x_2 at iteration t. The message update rule is:

$$m_{x_1\to x_2}^t\left(l_j\right) = min\left(c_{x_1x_2}\left\{D_{x_1}(l_i)\right\} + \sum m_{s\to x_1}^{t-1}\left(l_i\right) - m_{x_2\to x_1}^{t-1}\left(l_i\right) + V(l_i,l_j)\right) \tag{4.13}$$

The coefficients $c_{x_1x_2}$ are determined in the following way. First, a set of trees from the neighborhood graph (a 2D grid in our case) is chosen so that each edge is in at least one tree. A probability distribution p over the set of trees is then chosen. Finally, $c_{x_1x_2}$ is set to $p_{x_1x_2}/p_{x_1}$, i.e. the probability that a tree chosen randomly under p contains edge (x_1, x_2) given that it contains x_1. Note that if $c_{x_1x_2}$ were set to 1, then the update rule would be identical to that of standard belief propagation. An interesting feature of the TRW algorithm is that for any messages it is possible to compute a lower bound on the energy. The original TRW algorithm does not necessarily converge, and does not guarantee that the lower bound always increases with time. An improved version of TRW is called sequential TRW, or TRW-S. In this version, the lower bound estimate is guaranteed not to decrease, which results in certain convergence properties. In TRW-S we first select an arbitrary pixel ordering function S_{x_1}. The messages are updated in order of increasing S_{x_1} and at the next iteration in the reverse order. Trees are constrained to be chains that are monotonic with respect to S_{x_1}.

CHAPTER 5: Methodology

The advent of low-cost real-time depth sensors has attracted multiple studies in 3D perception and reconstruction for a wide range of applications including robotics, health-care, and surveillance. Reconstructing 3D indoor environments, however, remains challenging due to the cluttered spaces, extensive variability, and the real-time constraint. Furthermore, as an application point of view, perceiving the geometry of surrounding structures is very important for indoor environments. It is a valid assumption to consider, on the average, up to 95% of indoor-environment structures consist of planar surfaces [12].

This thesis describes an approach to efficiently partition a dense collection of 3D surface measurements from an RGBD sensor into regions that are locally approximated by a single plane. This problem is generally referred to as the geometric segmentation problem. Solutions to this problem play an important role in the creation of autonomous intelligent systems where these algorithms endows systems, e.g., robotic agents, with the ability to simplify and interpret the geometric scene structure. Applications of these algorithms are often found in object recognition systems [28][29], navigation systems [30][31], and mapping systems [32]. Other applications are found in approaches that simplify point cloud measurements into 3D polygonal models [33][34]. Our application of interest is the use of these algorithms within mobile robots. Here, point cloud data may originate from a pair of stereo cameras , a laser range sensor, or from a depth sensor [10][28]. Current versions of all these sensing systems produce 3D point cloud data at rates that outpace the ability of current state-of-the-art algorithms to segment these data into planar regions.

There are three steps to our approach for planar segmentation of point cloud data

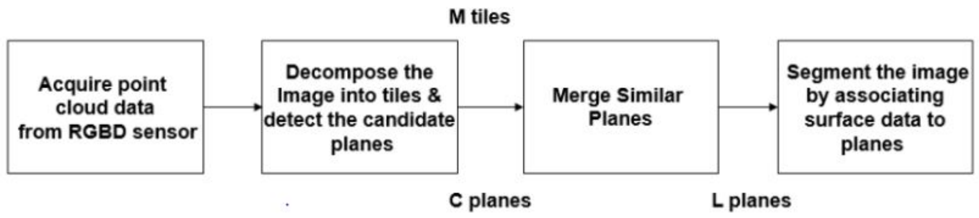

M tiles
Acquire point cloud data from RGBD sensor
Decompose the Image into tiles & detect the candidate planes
Merge Similar Planes
Segment the image by associating surface data to planes
C planes
L planes

Figure 5.1: Block diagram of the segmentation algorithm

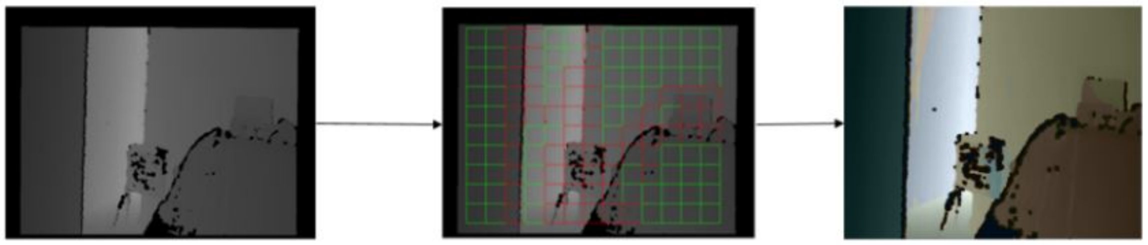


Figure 5.2: Outline of segmentation algorithm with an example

and figure 5.1: gives the outline of segmentation algorithm and figure 5.2 shows the working of algorithm with an example.

1. Detect a set of planes in the scene by fitting planar models to measured 3D point cloud data and storing those having low fit-error.

2. Merge similar planes based on their coefficients to generate a smaller set of L planes.

3. For each measured point, restrict the label set to a small subset of all candidate planes, $L_{(x,y)} \subset L$, by selecting only those planes detected in the vicinity of the point.

4. Perform constrained BBP estimation of the segment labels using the restricted candidate labels for each pixel.

Performance improvements are provided by the merge process of step (2) and the constraint-based estimation of step (4). The following sections detail each step of the algorithm outlined above.

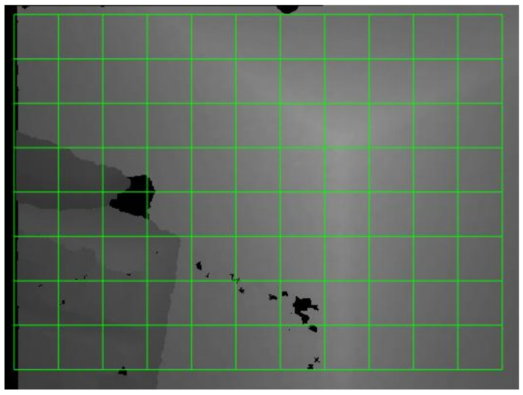


Figure 5.3: A tiling of an RGBD image is shown as a collection of blocks superimposed over the image. The image pixel intensities is proportional to depth from the image plane and the green boxes denote tiles. Within each tile we fit a plane to the sensed (X, Y, Z) surface data to detect planar segments that exist in the scene. Our segmentation associates each measured point to one of the planes detected within the tiles.

5.1 Planar Surface Detection

Detection of planar surfaces is accomplished by decomposing the measured 3D points space into cubical regions and subsequently fitting planes to the data within these regions. In practice, the size of the cubical regions will need to be adjusted to the scale of the object being analyzed. For many sensors, e.g., stereo reconstruction, LiDAR and RGBD camera, measurements are an explicit function of the sensor location. This greatly simplifies the decomposition by tiling the field-of-view of the measurement device to partition the data [35] [36] [37]. Our experiments use an RGBD sensor and tile the measured depth image as shown in Figure 5.3.

Within each tile, we use the standard least-squares method to estimate the unknown plane parameters that determine the plane that minimizes the fitting error which is taken as perpendicular squared Euclidean distance between the measurements and the unknown plane. The implicit formulation seeks to minimize the square of the perpendicular distance between the measured data points and the estimated planar model, i.e.,

$$\epsilon(a,b,c,d) = \min_{a,b,c,d} \sum_{i=1}^{N} \|aX_i + bY_i + cZ_i + d\|^2$$

We re-write this objective function as a quadratic matrix-vector product by defining the vector $\alpha = [\ a \quad b \quad c \quad d\ ]^t$ as the vector of planar coefficients and the matrix $\mathbf{M}$ as the matrix of planar monomials formed from the 3D (X, Y, Z) surface data having i^{th} row $\mathbf{M}_i = [\ X_i \quad Y_i \quad Z_i \quad 1\ ]$ [38]. Using this notation, the optimization function becomes:

$$\epsilon(\alpha) = \min_{\alpha} \alpha^t \mathbf{M}^t \mathbf{M} \alpha$$

As noted in several publications [39] the minimizer is known to be $\widehat{\alpha}$, the eigenvector associated with the smallest eigenvalue of the matrix $\mathbf{M}^t\mathbf{M}$ (also known as the scatter matrix). In general, $\mathbf{M}^t\mathbf{M}$ is a symmetric matrix and, for the monomials $\mathbf{M}_i = [\ X_i \quad Y_i \quad Z_i \quad 1\ ]$, the elements of this matrix are

$$\mathbf{M}^t\mathbf{M} = \sum_{i=1}^{N} \begin{bmatrix} X_i^2 & X_iY_i & X_iZ_i & X_i \\ X_iY_i & Y_i^2 & Y_iZ_i & Y_i \\ X_iZ_i & Y_iZ_i & Z_i^2 & Z_i \\ X_i & Y_i & Z_i & 1 \end{bmatrix} \tag{5.1}$$

The squared Euclidean error between the measured 3D data and the fit plane is obtained by normalizing the coefficients such that the coefficients a, b, c form a vector of unit length. Let $\eta = \sqrt{a^2 + b^2 + c^2}$ denote this normalization constant and we can then write the sum of squared Euclidean errors between the tile points and the fit surface as $\epsilon(\frac{\widehat{\alpha}}{\eta}) = \frac{\sqrt{\lambda_{11}}}{\eta}$.

For each tile, our plane detection algorithm stores the (x, y) position of the tile, the normalized parameters of the fit plane, $\frac{\widehat{\alpha}}{\eta}$, and the error observed between the plane and the data, $\epsilon(\frac{\widehat{\alpha}}{\eta})$. This generates an image of planar fits that, like the image data, is organized on a grid.

5.2 Merging Planes

The next step for our algorithm is to merge estimates of the same plane. This step serves to reduce the number of potential classes that must be considered during the segmentation problem. Standard implementation of BBP algorithms must compute the probability of each candidate classes at each individual pixel. Specifically, the cost of a single iteration of BBP for an $N \times M$ image having L candidate labels is $\mathcal{O}(NML^2)$ [27]. By merging the list of candidate planes we reduce potential over-segmentation of points lying on a same plane, i.e., having multiple instances of the same plane class, while simultaneously reducing the computational cost of the BBP algorithm.

Our merge procedure is a quick algorithm based on a clustering method that uses orthogonal projections of the plane coefficient data into 1-dimensional subspaces [40]. In our application, we cluster planes by projecting the coefficients of the plane models onto each of the 4 plane-parameter axes, i.e., the $a, b, c, d-$axes. In each case, we merge plane models that are adjacent on the given axis and satisfy the relation shown in equation (5.2).

$$\epsilon(\alpha_i, \alpha_j) = 1 - (a_i a_j + b_i b_j + c_i c_j) + \beta \left| d_i - d_j \right| < \upsilon \tag{5.2}$$

where υ denotes a similarity threshold and the plane pair (α_i, α_j) will be merged when $\epsilon(\alpha_i, \alpha_j) < \upsilon$. When two similar plane models are merged we discard the coefficients of the plane having larger fit error, $\epsilon(\frac{\widehat{\alpha}}{\eta})$, and assign the data associated with the discarded plane to reference the remaining plane model.

5.3 Markov Random Field (MRF)

The final step of our algorithm segments the measurement data to one of the classes from the set of L planes remaining after the merging procedure. This is accomplished by writing the segmentation as a MRF and then searching for the MAP estimate of the labels for each pixel.

In practice, the MRF energy is computed as the sum of the site energies. Hence the MRF is determined by defining the energy for site X_{ij} as shown in equation (5.3) and then the total field energy is obtained by summing this energy across all sites [41].

Our site energy function seeks to assign labels to by striking a balance between the error in the planar fit for each measured surface point, $D(Z(i,j)|X_{ij}=l)$ and preserving uniformity, i.e., smoothness in the assigned labels, $V(X_{ij}|X_{nm}\in\mathcal{N}_{ij})$.

$$\begin{aligned} U(X_{ij}) = \textstyle\sum_{l\in L} D(Z(i,j)|X_{ij}=l_i) \quad + \quad ... \\ \textstyle\sum_{(nm)\in\mathcal{N}_{ij}} V(X_{ij}=l_i|X_{nm}=l_j) \end{aligned} \tag{5.3}$$

Segmentation results rely heavily on the form of the data likelihood energy, $D(Z(i,j)|X_{ij}=l)$, and the label smoothness energy, $V(X_{ij}=l_i|X_{nm}=l_j)$.

Our proposed data likelihood energy uses a truncated cost function given in equation (5.4).

$$D(Z(i,j)|X_{ij}=l) = \lambda \min\left(\left|a_l X + b_l Y + c_l Z + d\right|, \tau\right) \tag{5.4}$$

The likelihood energy function encodes the stochastic relationship between measured depth $Z(i,j)$ and the unknown label value, l. Equation (5.4) states that the cost of associating the depth measurement $Z(i,j)$ to the plane having label, l, and plane coefficients $\alpha_l=\{a_l,b_l,c_l,d_l\}$. Note that this cost is proportional to the perpendicular Euclidean distance between the measured (X,Y,Z) point and plane α_l. The likelihood energy function includes two free parameters λ and τ. The λ parameter controls the relative weight of the terms $D(Z(i,j)|X_{ij}=l)$ and $V(X_{ij}|X_{nm}=x_{nm})$ in the total site energy $U(X_{ij})$. The τ parameter is used to restrict the range of the data energy cost which has been found to improve the robustness of MRF estimation procedures and denotes the maximum allowable cost between any given label and a measurement. Limiting the cost in this way reduces the sensitivity of the segmentation result to outliers in the measurement data.

Our proposed smoothness energy differs from typical smoothness constraints which typically rely only on the value of the labels. For example, the Potts model [23] imposes a cost penalty when the label for a given site is different than it's neighbor and zero cost otherwise as shown in equation (5.5).

$$V(X_{ij} = l_i | X_{nm} = l_j) = \begin{cases} \gamma & if \quad l_i \neq l_j \\ 0 & if \quad l_i = l_j \end{cases} \tag{5.5}$$

Since optimization seeks to minimize the cost function, this energy term "smooths" the label assignments by encouraging neighboring locations to share the same label values generating piecewise constant label regions. The cost/penalty, γ, for assigning distinct labels to neighboring MRF sites, i.e., pixel locations, is also referred to as a "discontinuity" cost.

In contrast to typical practice, our smoothness cost is composed of two parts: (1) a cost penalizing distinct neighboring label values (similar to equation (5.5)) and (2) a data likelihood term which serves to modulate the cost by how well the planar model predicts the observed depth difference.

For (1) we use the same plane dissimilarity metric previously applied for clustering as shown in equation (5.5).

$$\epsilon(\alpha_i, \alpha_j) = 1 - (a_i a_j + b_i b_j + c_i c_j) + \beta \, |d_i - d_j| \tag{5.6}$$

Equation (5.2) which has value 0 when $l_i = l_j$ (equivalently $\alpha_i = \alpha_j$) and will evaluate to a value of at least v for all other label pairs $l_i \neq l_j$ due to the clustering stage of §5.2. Note that these costs will increase for label pairs having large dissimilarity and is not constant.

For (2) we use the likelihood of the measured depth difference given the label value, i.e., plane model. Theoretically, we could make use of the plane coefficients to determine how well the depth difference is predicted by the plane coefficients. Consider two neighboring depth measurements Z_1 and Z_2. Hypothesized to lie on the plane $aX + bY + cZ + d = 0$. If this hypothesis is true, the predicted value of the depth at Z_2 will be $\widehat{Z_2} = Z_1 + c(Z_2 - Z_1)$. This can be converted to a cost function $D(Z_1, Z_2|c) = \left|Z_2 - \widehat{Z_2}\right| = |Z_2 - Z_1 - c(Z_2 - Z_1)| = |(1-c)(Z_2 - Z_1)|$ that adds cost when the observed the depth difference is different from that predicted by hypothesized plane model having coefficient c. In practice, we discard the term in c and use the cost $D(Z_1, Z_2) = |Z_2 - Z_1|$ which approximates the theoretical analysis.

The hybrid smoothness energy term resulting from parts (1) and (2) are combined into a single smoothness energy function as shown in equation (5.7) and 5.8.

$$V(X_{ij} = l_i | X_{nm} = l_j) = \max\left(|Z(i,j) - Z(m,n)|, \epsilon(\alpha_i, \alpha_j)\right) \tag{5.7}$$

$$V(X_{ij} = l_i | X_{nm} = l_j) = \begin{cases} \epsilon(\alpha_i, \alpha_j) & if \quad l_i \neq l_j \\ |Z(i,j) - Z(m,n)| & if \quad l_i = l_j \end{cases} \tag{5.8}$$

Where $\epsilon(\alpha_i, \alpha_j)$ is the plane dissimilarity metric of equation (5.6) for the plane pair (α_i, α_j) having labels (l_i, l_j) and $|Z(i,j) - Z(m,n)|$ is the absolute depth difference for the neighboring pixels. Introduction of this term improves segmentation as labels assignments that span large depth discontinuities do not incur the standard label smoothness penalties as is typical to a Potts model and other prior work on geometric segmentation [35].

5.4 MRF Energy Optimization via BBP

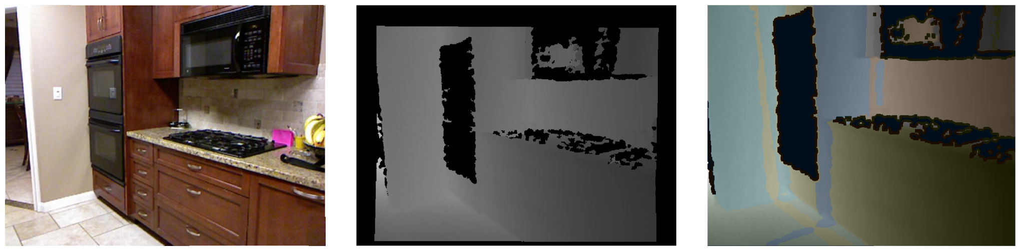
Our optimization of the MRF energy uses the Bayesian Belief Propagation (BBP) algorithm to find the MAP estimate of the segmentation label values. Computational cost is reduced by using the max-sum algorithm and passing messages using the "checkerboard" message passing scheme as described in [27], which takes advantage of regularity of the image grid to formulate the loopy BBP message passing as a two-step message passing process on a bipartite graph. This alternative message passing scheme decreases both the runtime and memory requirements by a factor of two when compared to traditional BBP message passing schemes. After a number of iterations of message passing, the maximum a posteriori label assignment is evaluated [42][43].

As mentioned in [27], several important classical computer vision problems can be formulated with simple smoothness energy functions which often include a single term that depends only on the difference between label values, i.e., $V(X_{ij} = l_i | X_{nm} = l_j,\, X_{nm} \in \mathcal{N}_{ij}) = |l_i - l_j|$. This criteria is the key attribute that allows the computational complexity of an iteration of the BBP algorithm to be reduced from $\mathcal{O}(NML^2)$ to $\mathcal{O}(NML)$, where NM denotes the number of MRF sites/pixels in the image and L denotes the number of segmentation labels. It is important to note that this result also holds for more general MRF label smoothness functions. Specifically, a sufficient condition for this performance boost requires only that the smoothness energy function, $V(X_{ij} = l_i | X_{nm} = l_j,\, X_{nm} \in \mathcal{N}_{ij})$, increases as a monotonic linear or quadratic function of the label difference [44].

Hence, for our algorithm or, more generally, any BBP-based optimization to benefit from the computational gains of standard methods as described in [27] a re-ordering of the labelset must be found that satisfies this monotonicity constraint. Unfortunately, it does not appear that, in general, such an arrangement exists for an arbitrarily large set of plane models using the cost metric of equation (5.7). As such, our BBP algorithm computational complexity remains $\mathcal{O}(NML^2)$ and requires two passes over the labelset to compute BBP messages rather than one. Despite this fact, our merge procedure (§5.2) aggressively clusters planes to generate a small number of candidate plane models. Hence, the constant L is typically small in our implementation which controls the relative impact of the computational complexity. Exploitation of this attribute to improve computational efficiency is a topic of future research.

CHAPTER 6: RESULTS

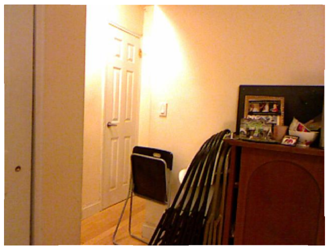
Our experiments use depth data from the Microsoft Kinect sensor provided by the NYU RGBD Dataset which include depth images from multiple indoor scenes. The RGBD sensor has a full-frame resolution of 640x480 pixels and a maximum depth range of ~6m. The standard BBP optimization algorithm and our modifications to this algorithm as described in 5 were implemented as MATLAB programs. Figures 6.1,6.2 6.3, and show the results of the planar segmentation algorithm on multiple indoor scenes including a kitchen area, bedroom, and hallway. Parameters used to generate these segmentations include a blocksize 40x40 px, $\beta = 0.4$, $\upsilon = 0.15$, and 5 iterations of belief propagation with $\lambda = 0.2$, $\tau = 0.5$. For these figures, the first column shows a color image of the scene, the middle column shows the registered depth image, and the last column shows the colored segmentation labels superimposed on the depth image.

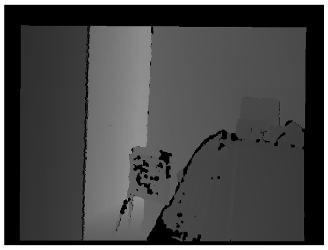


(a) Color image of kitchen (b) Depth image of kitchen (c) Segmented result

Figure 6.1: Planar segmentation algorithm applied to a kitchen area. Regions of similar color have been classified as coplanar. The algorithm performs well, segmenting the image into the planar surfaces such as walls, floors, and cabinetry.

Figure 6.1 shows the segmentation of a kitchen area. The algorithm performed well, separating large scale features such as walls, floor, and cabinetry. The algorithm picked up even the small surface patches below and above the oven. Parameters such


(a) Color Image of hallway


(b) Depth image of hallway

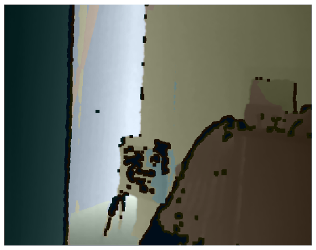
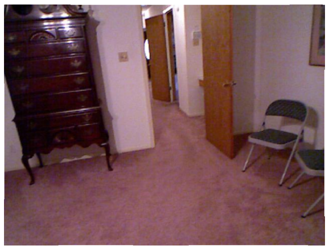
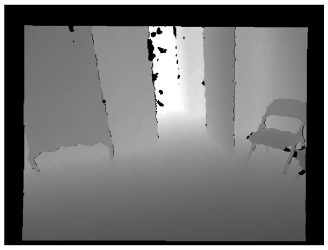

(c) Segmented result

Figure 6.2: Planar segmentation algorithm applied to a portion of a bedroom. Large scale features such as walls and the floor are well segmented.


(a) Color image of bedroom


(b) Depth image of hallway

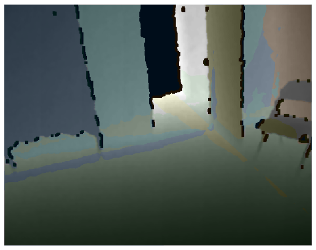

(c) Segmented result

Figure 6.3: Planar segmentation algorithm applied a hallway scene. The data belonging the walls, floors, door, and cupboard in the scene are relatively well classified.

as the blocksize and planar merge threshold can be adjusted to better target planar surfaces of various sizes. Figure 6.2 shows the segmentation of hallway. The segmentation algorithm approximated the image of hallway well with a total of 5 different planes. The larger surfaces such as walls, floor, and the cabinet were well identified and labeled as separate planes. Figure 6.3 shows the segmentation of bedroom. The algorithm segmented large scale planar features such as the walls, cupboard, door and floor particularly well, smaller planar surfaces such as the chair were missed due to the choice in block size. Reliable segmentation and extraction of these large scale planar features is of interest for many vision related problems.

Table 6.1 shows the time taken in milliseconds by each of the input depth image to generate the segmented result by applying our algorithm. The algorithm when run on several set of images took an average of 0.3 seconds to perform segmentation on a given depth image. It can be concluded that the segmentation algorithm is nearly

Table 6.1: Time taken in milliseconds for each image

Input Image	Time (msec)
Kitchen image(Figure 6.1)	372
Hallway image(Figure 6.2)	352
Bedroom image(Figure6.3)	358

real-time.

CHAPTER 7: Conclusion

The thesis discusses about a novel algorithm to segment point cloud data into local planar regions. This is a problem of generic interest to researchers in the computer graphics, computer vision, artificial intelligence and robotics community where it plays an important role in applications such as object recognition, mapping, navigation and conversion from point clouds representations to 3D surface models. In contrast to prior work, this algorithm uses planar models, a Markov Random Field and efficient Bayesian Belief Propagation to segment geometry-only depth images. The fast clustering technique applied removes redundant plane detections prior to optimization to improve performance. A new MRF smoothness energy function dynamically changes to encourage distinct labels across depth discontinuities. These modifications provide improved segmentations for geometry-only depth images while simultaneously controlling the computational cost. We focus on extraction of large scale planes from the measured scene data and detail how our segmentation process efficiently achieves this goal. Our algorithm includes parameters that robotics researchers might employ to strike a compromise between segmentation detail and performance. Experimental results apply the algorithm to the NYU depth dataset and indicate that the algorithm can accurately segment a variety of planar surfaces from depth sensor data.

REFERENCES

[1] P. Joshi, “Understanding camera calibration.”

[2] K. Alahari, *Efficient Inference and Learning for Computer Vision Labelling Problems.* PhD thesis, Oxford Brookes University, 2010.

[3] K. M. B. Andrew R. Willis, “Real-time geometric scene estimation for rgbd images using a 3d box shape grammar,” in *Three-Dimensional Imaging, Visualization, and Display 2016*, 2016.

[4] B. G. S. Ramesh Jain, Rangachar Kasturi, *Machine Vision*, ch. 15, pp. 459–491. McGraw-Hill, 1995.

[5] Y. Wu, *Towards Scene Understanding: Deep and Layered Recognition and Heuristic Parsing of Objects.* PhD thesis, Xian Jiaotong University, 2010.

[6] I. Jolliffe, *Principal Component Analysis.* Springer-Verlag New York, 1986.

[7] A. HyvÃ€rinen, “Independent component analysis by minimization of mutual information,” 1997.

[8] K. P. Murphy, *Machine Learning: A Probabilistic Perspective.* The MIT Press, 2012.

[9] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision.* Upper Saddle River, NJ, USA: Prentice Hall PTR, 1998.

[10] K. Khoshelham and S. O. Elberink, “Accuracy and resolution of kinect depth data for indoor mapping applications,” *Sensors*, vol. 12, no. 2, p. 1437, 2012.

[11] A. R. W. John Papadakis, “Real-time surface fitting to rgbd sensor data,” in *SoutheastCon 2017*, 2017.

[12] H. J. H. . A. P. . E. B. . P. H. N. de, “Fast planar segmentation of depth images,” in *SPIE*, 2015.

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

[14] L. B. J. Sheynin, “Real-world scene perception and perceptual organization: Lessons from computer vision,” *Journal of Vision*, 2013.

[15] R. Nock and F. Nielsen, “Statistical region merging,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, pp. 1452–1458, Nov 2004.

[16] L. G. Shapiro and G. C. Stockman, *Computer Vision.* Prentice Hall, 2001.

[17] R. Ohlander, K. Price, and D. R. Reddy, "Picture segmentation using a recursive region splitting method," *Computer Graphics and Image Processing*, vol. 8, no. 3, pp. 313 – 333, 1978.

[18] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[19] S. Z. Li, *Markov Random Field Modeling in Image Analysis*. Springer Publishing Company, Incorporated, 3rd ed., 2009.

[20] S. A. Barker, *Image Segmentation using Markov Random Field Models*. PhD thesis, University of Cambridge, 1998.

[21] N. Komodakis, *Optimization Algorithms for Discrete Markov Random Fields, with Applications to Computer Vision*. PhD thesis, University of Crete, 2006.

[22] N. Ho, "Loopy belief propagation, markov random field, stereo vision."

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

[24] D. S. O. V. V. K. A. A. M. T. Richard Szeliski, Ramin Zabih and C. Rother, "A comparative study of energy minimization methods for markov random fields," in *Computer Vision - ECCV 2006: 9th European Conference on Computer Vision*, 2006.

[25] R. Timofte and L. Van Gool, *Efficient Loopy Belief Propagation Using the Four Color Theorem*. Advances in Computer Vision and Pattern Recognition, London: Springer, 2013.

[26] A. Ajroud, M. N. Omri, H. Youssef, and S. Benferhat, "Loopy belief propagation in bayesian networks : origin and possibilistic perspectives," *CoRR*, vol. abs/1206.0976, 2012.

[27] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient belief propagation for early vision," *International Journal of Computer Vision*, vol. 70, no. 1, pp. 41–54, 2006.

[28] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments," in *In the 12th International Symposium on Experimental Robotics (ISER*, 2010.

[29] P. K. Nathan Silberman, Derek Hoiem and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012.

[30] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 573–580, Oct 2012.

[31] T. J. J. Tang, W. L. D. Lui, and W. H. Li, "A lightweight approach to 6-dof plane-based egomotion estimation using inverse depth," in *Australasian Conference on Robotics and Automation*, 2011.

[32] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, (Shanghai, China), May 9-13 2011.

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

[34] T. Wiemann, A. NÃŒchter, K. Lingemann, S. Stiene, and J. Hertzberg, "Automatic construction of polygonal maps from point cloud data," in *2010 IEEE Safety Security and Rescue Robotics*, pp. 1–6, July 2010.

[35] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, *Real-Time Plane Segmentation Using RGB-D Cameras*, pp. 306–317. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[36] J. P. Lewis, "Fast template matching," in *Vision interface*, vol. 95, pp. 15–19, 1995.

[37] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-time plane segmentation using rgb-d cameras," in *In Proc. of the 15th RoboCup Int. Symp*, 2011.

[38] G. Taubin, "Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 11, pp. 1115–1138, 1991.

[39] A. Fitzgibbon, M. Pilu, and R. B. Fisher, "Direct least square fitting of ellipses," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, pp. 476–480, May 1999.

[40] D. Han and J. Kim, "Unsupervised simultaneous orthogonal basis clustering feature selection," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5016–5023, June 2015.

[41] stuart geman and C. Graffigne, "Markov random field image models and their applications to computer vision," *International Congress of Mathematicians*, 1986.

[42] K. P. Murphy, Y. Weiss, and M. I. Jordan, "Loopy belief propagation for approximate inference: An empirical study," in *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, (San Francisco, CA, USA), pp. 467–475, Morgan Kaufmann Publishers Inc., 1999.

[43] Y. Weiss and W. T. Freeman, "Correctness of belief propagation in gaussian graphical models of arbitrary topology," *Neural Comput.*, vol. 13, pp. 2173–2200, Oct. 2001.

[44] J. Yedidia, W. Freeman, and Y. Weiss, "Understanding belief propagation and its generalizations," in *Exploring Artificial Intelligence in the New Millennium* (G. Lakemeyer and B. Nebel, eds.), ch. 8, pp. 239–236, Morgan Kaufmann Publishers, Jan. 2003.