# TITLE

Thesis submitted in partial fulfillment
of the requirements for the degree of

*DEGREE*
*in*
*COURSE*

by

NAME
ROLL NUMBER
`EMAIL ID`

CENTER NAME
International Institute of Information Technology
Hyderabad - 500 032, INDIA
MONTH YEAR

International Institute of Information Technology
Hyderabad, India

**CERTIFICATE**

It is certified that the work contained in this thesis, titled " " by NAME, has been carried out under my supervision and is not submitted elsewhere for a degree.

| | |
|---|---|
| Date | Adviser: Prof. NAME |

To SOMEONE

# Acknowledgments

ionAcknowledgements goes here ...

# Abstract

Abstract goes here ...

# Contents

# List of Figures

# List of Tables

# *Chapter 1*

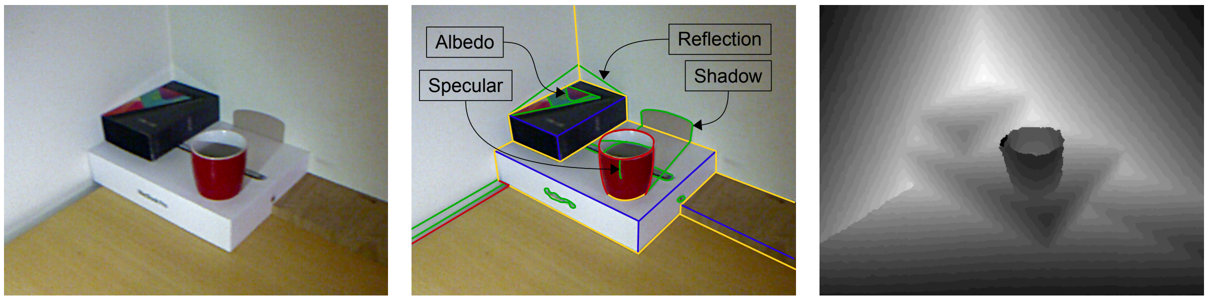
# Introduction

## 1.1 Paper's Content

**Classical problem, history** The task of labeling edges as convex, concave, and occluding entities [12, 13, 16, 25] is one of the classical problems in computer vision. This problem was first addressed on synthetic drawings, where several constraint satisfaction algorithms were proposed [25]. On real image data, the problem of labeling line-drawings is a very hard problem. Furthermore, this problem was shown to be challenging even on RGBD data obtained using commodity sensors like Kinect. This paper studies the problem of classifying boundaries from RGBD data. One could also apply such techniques to dense point clouds computed using multiview techniques [5, 24].

**Edges** Edges in an image often correspond to depth discontinuities at object boundaries (occlusion edges) or normal discontinuities (convex or concave edges). In addition, there could be planar edges that are within planar regions. Figure 3.1 shows an image containing different types of edges. Note that planar edges may result from shadows, reflection, specularities and albedo variations. In classical line labeling with synthetic line drawings, we do not have any planar edges as the purpose of edge labeling has always been to classify the depth edges into occluding, convex and concave entities. However, in real images planar edges occur more frequently than others.

**importance, applications (reconstruction, segmentation and recognition, challenges** Contours are critical to the human perception of scenes. Beyond localizing the discontinuities in the color distribution, edges provide cues related to the surface orientation and depth discontinuities. The importance of edges in understanding of structure was realized quite early with works on recovering 3D structure from single images [17, 12]. Hoiem *et al.* [9] showed that occlusion boundaries provide very useful cues to estimate the depth of a scene from a single image. It was shown that PMVS point cloud [5] can be densified and improved if occluding contour information is available [22]. In our work, we specifically focus on providing this information. Two important challenges need to be addressed in order to use this information in a recognition or reasoning algorithm. The first arises from the incompleteness of edge information derived from real-world images as well as noisy edges from lighting and other imaging conditions. The second is due to inherent ambiguities in mapping from edges to structure, where the



(a) RGB Image (b) Edge Types (c) Depth Map

**Figure 1.1** *The edges in (a) are marked in (b) with the color code: red (occluding), green (planar), blue (convex), and yellow (concave). Planar edges are caused by different phenomena as marked. (c) shows the Kinect depth map (note the depth quantization artifacts).*

same edge map can result from multiple object configurations [25]. As a result, many existing algorithms [8, 7, 21, 20, 15] use the boundary pixels for semantic labeling and depth estimation without considering the class labels for these edge pixels.

**Close relation with Jia et al** Our approach is closely related to the work of Jia *et al.* [11]. The main similarity is the use of a 2D MRF to infer the edge labels. However, there are many other differences. Jia *et al.* [11] showed three-class labeling of occluding, connecting and homogeneous boundaries from RGBD data. On the contrary, we show four class edge labeling. In their approach, plane-fitting near edges are used to compute the features for the task of edge-labeling. In their approach, SVM regression is used for unary feature computation, whereas we use Random forest. In our work, we rely on simple depth comparison features for the classification and such simple features are more robust to missing data and noise in comparison to plane fitting techniques. Note that such simple edge comparison features are shown be useful in Kinect human pose regression work of Shotton *et al.* [19].

**Gupta et al** Gupta *et al.* [6] addressed the problem of indoor scene understanding from a single RGBD image, where they segment and classify image regions. In the process, they also label the edges to be convex, concave or occluding, and provide some qualitative results of the same. We are interested in classifying every edge pixel as one of the four types: convex, concave, occluding and planar. We use the local information available from color, depth and surface normals for this purpose. Our experiments show that we perform better than Gupta *et al.* [6].

**RGB is being explored. Why RGBD ?** There has been a few methods that address the edge labeling from single RGB images. Fouhey *et al.* [4] addressed this problem in an indoor Manhattan world setting, while Gupta *et al.* [4] and Eigen *et al.* [3] predict normal and depth maps respectively from a single view using deep networks, which can then be used for detecting and labeling edges. We solve the problem using coarse depth information. In many 3D models obtained using RGBD sensors or multi-view reconstruction techniques, we typically have very noisy 3D point cloud near the boundaries. This is because most stereo reconstruction algorithms and structured light techniques are known to provide noisy reconstruction near the boundaries. For example, the accuracy of 3D points obtained from a

Kinect sensor is extremely noisy near the boundaries. Thus the labeling problem that we address in this paper is not straightforward. We believe that by first solving this problem in RGBD settings, we will get the necessary insights to solve the more challenging problem of labeling edges from a single RGB image.

## 1.2 Edges in scene understanding

History of edge labeling; applications like segmentation, reconstruction and recognition; importance of edge labels in understanding the scene; some papers;
[**?**]

## 1.3 Depth estimation techniques

### 1.3.1 Monocular

### 1.3.2 Stereo

### 1.3.3 3D scanner

In many 3D models obtained using RGBD sensors or multi-view reconstruction techniques, we typically have very noisy 3D point cloud near the boundaries. This is because most stereo reconstruction algorithms and structured light techniques are known to provide noisy reconstruction near the boundaries.

[**?**]

*Chapter 2*

# Related Work

- Explain works with edge labeling. How our bottom-up approach is new. - Works with repairing of 3D models using edge labels.

*Chapter 3*

# Edge labeling using point cloud

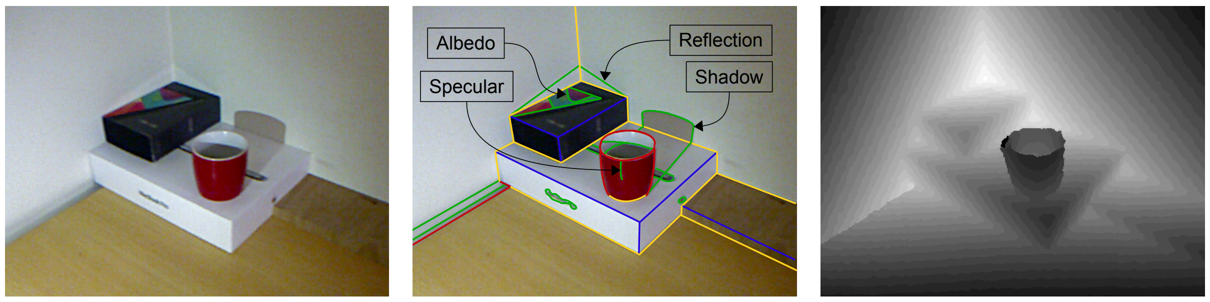
**Abstract:** The problem of labeling the edges present in a single color image as convex, concave, and occluding entities is one of the fundamental problems in computer vision [25]. It has been shown that this information can contribute to segmentation, reconstruction and recognition problems. Recently, it has been shown that this classification is not straightforward even using RGBD data. This makes us wonder whether this apparent simple cue has more information than a depth map? In this paper, we propose a novel algorithm using random forest for classifying edges into convex, concave and occluding entities. We release a data set with more than 500 RGBD images with pixel-wise ground labels. Our method produces promising results and achieves an F-score of $0.84$ on the data set. [We first solve the problem using scanned 3D models. Then we move on to use stereo images which is more difficult.]

**Introduction:** Edges in an image often correspond to depth discontinuities at object boundaries (occlusion edges) or normal discontinuities (convex or concave edges). In addition, there could be planar edges that are within planar regions. Figure 3.1 shows an image containing different types of edges. Note that planar edges may result from shadows, reflection, specularities and albedo variations. In classical line labeling with synthetic line drawings, we do not have any planar edges as the purpose of edge labeling has always been to classify the depth edges as occluding, convex and concave. However, in real images planar edges occur more frequently than others. This paper studies the problem of classifying boundaries from RGBD data. In many 3D models obtained using RGBD sensors or multi-view reconstruction techniques, we typically have very noisy 3D point cloud near the boundaries. This is because most stereo reconstruction algorithms and structured light techniques are known to provide noisy reconstruction near the boundaries. This makes the labeling problem challenging.

## 3.1 Method

## 3.2 Contour Labeling

We use both image and depth cues to infer the labels of edge pixels. We start with a set of edge pixels obtained from an edge detection algorithm and the goal is to assign one of the four labels to each of these



(a) RGB Image (b) Edge Types (c) Depth Map

**Figure 3.1** *The edges in (a) are marked in (b) with the color code: red (occluding), green (planar), blue (convex), and yellow (concave). Planar edges are caused by different phenomena as marked. (c) shows the Kinect depth map (note the depth quantization artifacts).*

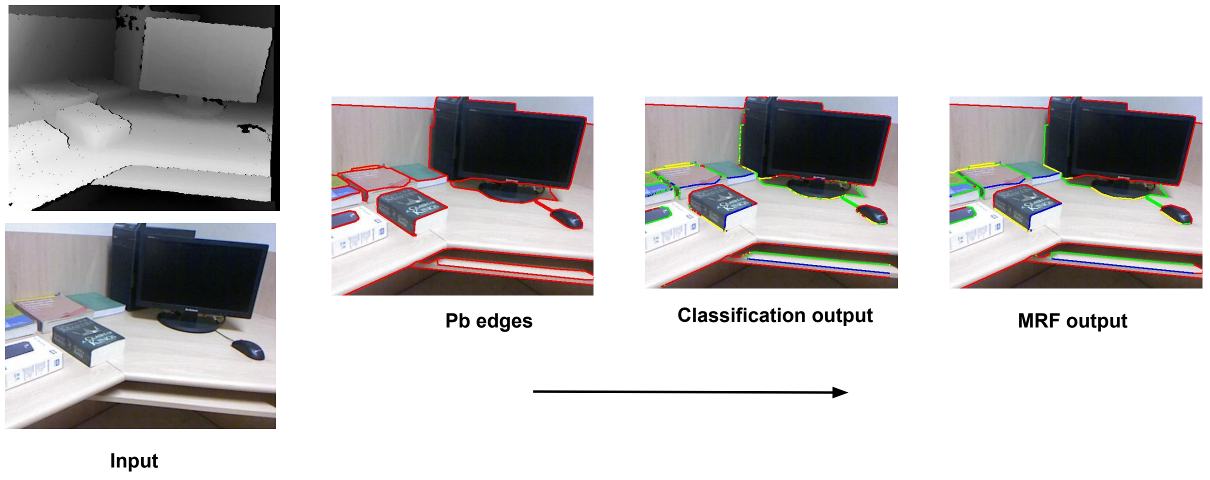
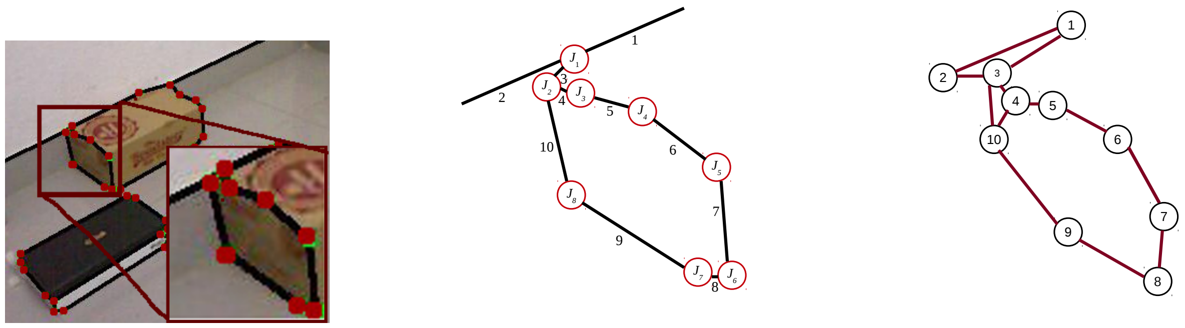


**Figure 3.2** *This figure summarizes the pipeline of our approach. It shows RGB and depth maps as input (1st image set), with Pb edge detection [18] (2nd image). The classification and MRF outputs are shown in the last two images respectively. Color code: red (occ), green (pln), blue (cvx), yellow (ccv).*

edge pixels. However, to improve computational efficiency and to overcome noise in the data, we link similar connected edge pixels in a neighborhood into contour segments. The process is carried out using an edge linking algorithm that combines connected edge pixels into a link as long as the curvature of the link is within a threshold. Each edge pixel is uniquely mapped to one of the contour segments. Each contour segment $c_i$ corresponds to a set of edge pixels and labeling the contour segments will uniquely label the edge pixels as well. The problem is thus reduced to computing an optimal labeling of all the contour segments. The individual steps in the algorithm is shown in Figure 3.2.

### 3.2.1 Contour graph

Each contour segment is considered as a single entity for labeling and is represented as a node in a graph. The junctions between the contour segments provide the connectivity information for the graph.



(a) Edge Links (b) Junction Graph (c) Contour Graph

**Figure 3.3** *Contour segments are part of edge links that is bounded by two junctions as shown in (a). In (b), we show a graph where edge junctions are nodes and edge links are edges. In (c), we show a graph with contour segments $c_i$ as nodes and junctions lead to edges between nodes.*

Labeling of an object boundary is then reduced to labeling of all the nodes in the graph that correspond to that specific boundary. Figure 3.3 shows an example, where the edge map from a portion of an image is converted into the corresponding contour graph representation.

We formulate the edge labeling problem as an inference in a graph where the nodes take different labels or states. The optimum labeling is achieved by minimizing an energy function. Each node can take one of the four labels given by occluding, planar, convex, and concave.

Let us consider a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where the vertices of the graph correspond to the set of $n$ contour segments; i.e., $c_i \in \mathcal{V}, i = \{1, ..., n\}$. The edges in the graph are pairs of contour segments. For every junction $J_k$ that falls on two contour segments $c_i$ and $c_j$, we have an edge $(c_i, c_j) \in \mathcal{E}$. Each vertex $c_i$ can take four possible states given by $\mathcal{L} = \{\text{occ,pln,cvx,ccv}\}$.

To formulate the problem as a labeling problem on an MRF, we need to define unary potentials for each node $c_i$, as well as the pairwise potentials for every edge in the graph $\mathcal{G}$.

### 3.2.2 Unary Potentials

To define the unary potentials of an edge contour $c_i$, we consider the corresponding set of edge pixels in $c_i$. We define a feature vector for each edge pixel and use a classifier to predict its class label. The unary potential of edge contour $c_i$ for label $l$ is defined as:

$$U_l(c_i) = 1 - \frac{N_l}{N_c} \tag{3.1}$$

Here, $N_l$ is the total number of edge pixels on the contour $c_i$ belonging to label $l$. $N_c$ is the total number of edge pixels on contour $c_i$. The computation of the feature vector and the classification process is described in the following section.

**The Pixel Classifier:** Given a contour segment $c_i$, we define its neighborhood properties based on a set of pixels on either side of the edge as shown in Figure 3.4. We denote a pixel on the contour

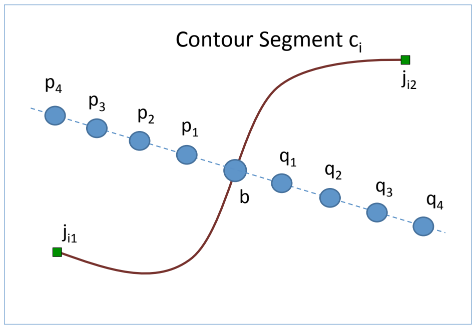


**Figure 3.4** *Edge Pixel Neighborhood: The graph potentials of an edge pixel $b$ are defined based on a neighborhood consisting of four pixel on either side of the edge ($p_i$'s and $q_i$'s) on a line perpendicular to the gradient edge direction at $b$.*

segment in the image as $b$. We consider $4$ points on either side of the edge pixel. These points lie on a line perpendicular to the gradient edge direction at $b$. Let the four points on one side be denoted by $(p_1, p_2, p_3, p_4)$ and the other side be $(q_1, q_2, q_3, q_4)$. Let $I(p_i)$ denote the RGB color vector at pixel $p_i$. The corresponding 3D points in the world are denoted using upper case letters, $P_i$, $Q_i$ and $B$. The 3D points are obtained using RGBD data. Let $O$ be the center of the camera. For a vector $V$, let $< V >= \frac{V}{|V|}$ denote the corresponding normalized vector. Let $A.B$ denote the dot product of vectors $A$ and $B$. Let $\mathcal{L}(Q_i, P_j, P_k)$ denote the distance of a point $Q_i$ to the 3D line joining points $P_j$ and $P_k$. We denote the number of pixels in the neighborhood of $b$ with unknown depth values as $\mathcal{U}(b)$.

We briefly describe the role of different elements of the feature vector from the Table 3.1.

**Table 3.1** *Elements of the feature vector used in pixel-wise edge classifier. Here, the indices $i$, $j$ and $k$ vary from 1 to 4.*

| Set Index | Description |
|---|---|
| 1 | $< P_i - P_j > . < Q_i - Q_j >$ |
| 2 | $\frac{\lvert P_i - Q_i \rvert}{min(\lvert P_i - B \rvert, \lvert Q_i - B \rvert)}$ |
| 3 | $\lvert I(p_i) - I(q_i) \rvert$ |
| 4 | $< (P_4 - B) + (Q_4 - B) > . < B - O >$ |
| 5 | $\frac{\lvert P_i - O \rvert}{\lvert P_{i-1} - O \rvert}$ |
| 6 | $\mathcal{L}(Q_i, P_j, P_k)$ |
| 7 | $\mathcal{U}(b)$ |
| 8 | $\lvert A - B \rvert, A, B \in \{P_1, .., P_4, B, Q_1, .., Q_4\}$ |
| 9 | $< P_i - B > . < Q_i - B >$ |

1. The first set of features denote the dot products based on two vectors on either side of an edge pixel. This captures the surface planarity in the neighborhood of an edge pixel. There are six distinct features of this category and are computed from the edge pixel neighborhood (see Figure 3.4). This set of features have high values for planar edge pixels.

2. The second set of features try to capture the depth difference on either side of the edge, which helps in finding occlusion edges. The value is expected to be high for occluding edges.

3. The third set captures the normalized color difference between pixels on either side. This value is likely to be high for occluding edges.

4. The dot products in the fourth set differentiates between convex and concave labels.

5. The ratios of distances in the fifth set captures the slopes of surfaces on either side from the view point of the camera. This helps in separating convex and concave edges. This value would be greater than one for concave and less than one for convex.

6. The sixth set captures the distances from a point to a line. These values are close to zero for planar, and nonzero for concave, convex and occluding. For occluding these values are very large.

7. The number of pixels in the neighborhood of an edge with unknown depth values. This number is high for occluding edge pixels.

8. The eighth set contains the depth differences between all pairs of points in the set.

9. The ninth set contains dot products between the pair of vectors originating from $B$ and ending at $P_i$ and $Q_i$ respectively.

Given the feature representations of all edge pixels in an image, we train a random forest classifier with 30 trees that outputs the likelihood for each label. Based on the likelihoods, we identify the class label for each pixel.

### 3.2.3 Inference using graph cuts

Given the unary and pairwise potentials of each contour segment (node), we will pose the problem of finding the most likely labels as that of minimizing the total energy over an MRF. A labeling of the graph, $L$ is defined as an assignment of labels $l_p$ to each node $c_p \in \mathcal{V}$ in the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. The data term $\mathcal{D}(L)$ is a sum of the unary potentials over all the nodes with respect to the labeling $L$ as shown below:

$$\mathcal{D}(L) = \sum_{p=1}^{n} U_{l_p}(c_p) \tag{3.2}$$

The smoothness term $\mathcal{S}(L)$ is the sum of pairwise potentials over all the neighbors $(c_p, c_q) \in \mathcal{E}$ in the graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. We use Potts model to define the pairwise potential $P_{l_p,l_q}$, which takes a value of 0 for same labels ($l_p = l_q$) and 1 for dissimilar ones ($l_p \neq l_q$). The weight factor $W_{l_p,l_q}$ is defined as the cost of assigning labels $l_p$ and $l_q$ to any two neighboring nodes $c_p$ and $c_q$. The smoothness terms is shown below:

$$\mathcal{S}(L) = \sum_{p,q=1,p\neq q}^{n} W_{l_p,l_q} P_{c_p,c_q} \tag{3.3}$$

The total energy $E(L)$ is defined in equation 3.4. The energy function is given by the sum of unary and pairwise terms:

$$E(L) = \lambda_1 \mathcal{D}(L) + \lambda_2 \mathcal{S}(L) \tag{3.4}$$

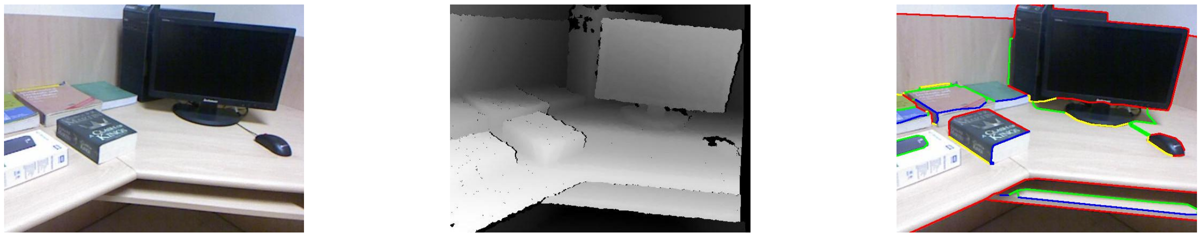
Here, $n$ is the total number of contour segments (nodes) in the graph. The parameters $\lambda_1$ and $\lambda_2$ are positive values that control the relative importance of the two terms. For all experiments, we use $\lambda_1 = 1000.0$ and $\lambda_2 = 12.5$ respectively. The multi-label MRF problem is solved using alpha-beta swap [1].

[Contour graph, unary potential (classifier), graph cut..]

## 3.3 Experiments

[Dataset and annotation, evaluation and numerical results]

**Dataset and Annotation:** We have experimented with multiple edge detectors including Canny [2], Pb [18] and Crisp boundary [10] edge detectors, and finally settled down on the Pb edge detector for our purposes. As noted before, the choice of edge detector does not significantly affect the computation of unary potentials and the label assignment. However, the connected contour nature of Pb edges makes it easier to define the graph connectivity for MRF formulation. Edge detection is followed by edge linking to obtain a set of possible edge contour segments in the image. While this approach provides good results, it can be readily replaced by any edge detection algorithm that performs well for the given class of images [14].



(a) RGB (b) Depth Map (c) Ground truth

**Figure 3.5** *Sample image from our dataset showing the RGB image, depth map and edge groundtruth. Colors: red (occ), green (pln), blue (cvx), yellow (ccv).*

For quantitative evaluation of the method, we have created an annotated dataset of 500 RGBD images of varying complexity. Train to test ratio is 3:2. Though in principle we could use a variety of methods to create 3D cues, we use Kinect based images and depth maps in this work. Kinect does not work beyond a distance of around four meters. Therefore, our dataset consists of indoor scenes where the objects lie within this distance. Our dataset consists of objects such as tables, chairs, cupboard shelves, boxes and household objects in addition to walls and floors. We also annotate 100 images from NYU [23] dataset, which include varying scenes from bed-room, living-room, kitchen, bathroom and so on with different complexities.

For each edge contour, we annotate each of its contour segments with one of the four labels: occlusion, convex, concave and planar. The average number of occlusion, planar, convex and concave edges pixels in an image in the annotated dataset are $953, 1324, 304$ and $468$ respectively. The corresponding numbers for NYU dataset are $1645, 445, 325$ and $399$ respectively. Figure 3.5 shows an annotated example from our dataset.

### 3.3.1 Evaluation and Numerical Results

We use recall, precision and F-measure in order to evaluate the performance of the labeling algorithms. The average precision, recall and F-measure for each of the edge labels over all the images in the dataset is given in Table 3.2 in addition to results on the NYU dataset. Results for different images using the algorithm are given in Figure 3.7 and Figure 3.8.

**Table 3.2** *Precision, Recall and F-measure for each edge type on our and NYU datasets.* $1^{st}$ *and* $2^{nd}$ *rows of each set gives the results of our approach and comparison with [6]. The* $3^{rd}$ *row in each set shows the results of our approach on NYU dataset.*

| | Occluding | Planar | Convex | Concave |
|---|---|---|---|---|
| Recall | **0.85** | **0.92** | **0.70** | **0.78** |
| Gupta *et al.* [6] Recall | 0.70 | 0.84 | 0.52 | 0.67 |
| Our Recall on NYU | 0.76 | 0.85 | 0.56 | 0.69 |
| Precision | **0.86** | **0.81** | **0.93** | **0.89** |
| Gupta *et al.* [6] Precision | 0.71 | 0.75 | 0.72 | 0.71 |
| Our Precision on NYU | 0.79 | 0.80 | 0.77 | 0.71 |
| F-measure | **0.86** | **0.86** | **0.80** | **0.83** |
| Gupta *et al.* [6] F-measure | 0.71 | 0.79 | 0.61 | 0.69 |
| Our F-measure on NYU | 0.77 | 0.83 | 0.65 | 0.70 |

**Table 3.3** *Confusion matrix across the four classes. The numbers given are the average number of edge pixels per image.*

| | Occ | Pln | Cvx | Ccv |
|---|---|---|---|---|
| Occ | 677 | 100 | 5 | 11 |
| Pln | 53 | 993 | 10 | 27 |
| Cvx | 27 | 58 | 201 | 2 |
| Ccv | 28 | 70 | 1 | 344 |

**Table 3.4** *Precision, recall and F-measure for each edge type without and with pairwise potentials.*

| | Occ | Pln | Cvx | Ccv |
|---|---|---|---|---|
| Pixel Recall | 0.82 | 0.87 | 0.69 | 0.75 |
| Final Recall | **0.85** | **0.92** | **0.70** | **0.78** |
| Pixel Precision | 0.84 | **0.85** | 0.90 | 0.86 |
| Final Precision | **0.86** | 0.81 | **0.93** | **0.89** |
| Pixel F-measure | 0.83 | 0.86 | 0.78 | 0.80 |
| Final F-measure | **0.86** | **0.86** | **0.80** | **0.83** |

To understand the effect of unary versus pairwise potentials on the precision and recall measures, we look at the effect of classification of edge pixels and edge contour segments using only the unary potentials, without the pairwise terms. Table 3.4 shows the resulting precision, recall and F-score values over the whole database. For convenience of comparison, we have also included the results using the pairwise terms. We get an average F-score of $0.82$ on the classification results for our data set. The use of smoothness constraints in the MRF achieves an F-score of $0.84$.

We compare the proposed approach with the semantic labeling of edges obtained by Gupta *et al.* [6], by computing their results on our dataset of annotated edges. Note that their work provides three labels (occluding, convex and concave). Since our dataset contains planar edges also, we perform a straightforward extension of their approach to $4$ labels for a fair comparison.

For evaluation purposes, we consider only those pixels in the cropped range $41 - 600 \times 46 - 470$ to remove the area where depth information is mostly missing in the Kinect depth map. This was done to be consistent with [6] in the data used for comparison. Figure 3.7, Figure 3.8 and table 3.2 provide qualitative and quantitative comparisons of the outputs of the two approaches. It can be seen that our approach produces superior results. We see that our approach labels the edges correctly even when they occur extremely close to another type of edge. For *e.g.*, in result (d) of figure 3.8, the rightmost convex edge got correctly classified while the approach by Gupta *et al.* [6] fails to do so. Similar results can be seen in much of the complex images in the NYU dataset (see figure 3.6).

We see that the algorithm achieves high precision for each of the edge types. The recalls are also high except for convex and concave edges. This is primarily due to the fact that we have several complex scenes in our dataset where a convex/concave edge does not have good depth quantization or do not have enough depth values registered around it. We are able to correctly classify complex convex/concave edges even with narrow regions having steep slope on either sides of the edge, provided the depth map is good. The NYU dataset contains complex scenes with glass windows and table heads for which Kinect fails to register the depth accurately. While this results in lower recall for convex and concave edges, we achieve an average F-score of $0.74$ for the NYU dataset.

On detailed analysis, the primary causes of errors in our approach were found to be: i) missing depth values from Kinect and ii) very small depth differences for occluding edges. While the first problem may be solved using better sensors and using image based potentials, the second would require a higher level understanding of the scene and objects. The proposed algorithm can be extended to work with SFM point clouds as well. The depth map in SFM can be easily created using the camera matrices of images and the point cloud. The main challenge here is that SFM point cloud is sparser than Kinect. Therefore, the depth and normal information may not be as reliable as that of Kinect.

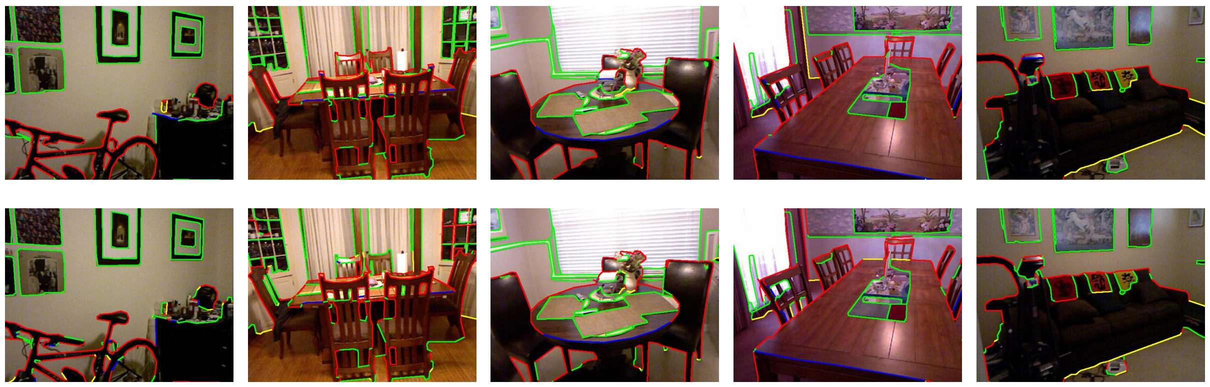


**Figure 3.6** *NYU dataset results : Ground truths (above) and the corresponding results from our approach (below). Color code: red (occ), green (pln), blue (cvx), yellow (ccv).*

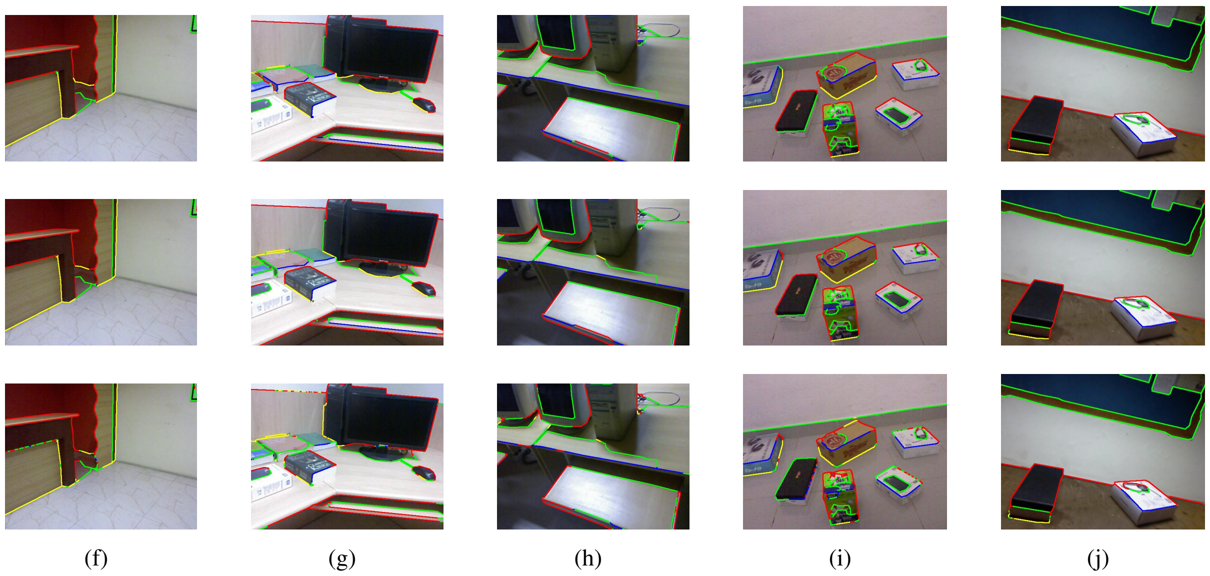


**Figure 3.7** *Ground truths (above) and the corresponding results from our approach ($2^{nd}$ row) and Gupta* et al. *[6] ($3^{rd}$ row). Color code: red (occ), green (pln), blue (cvx), yellow (ccv).*

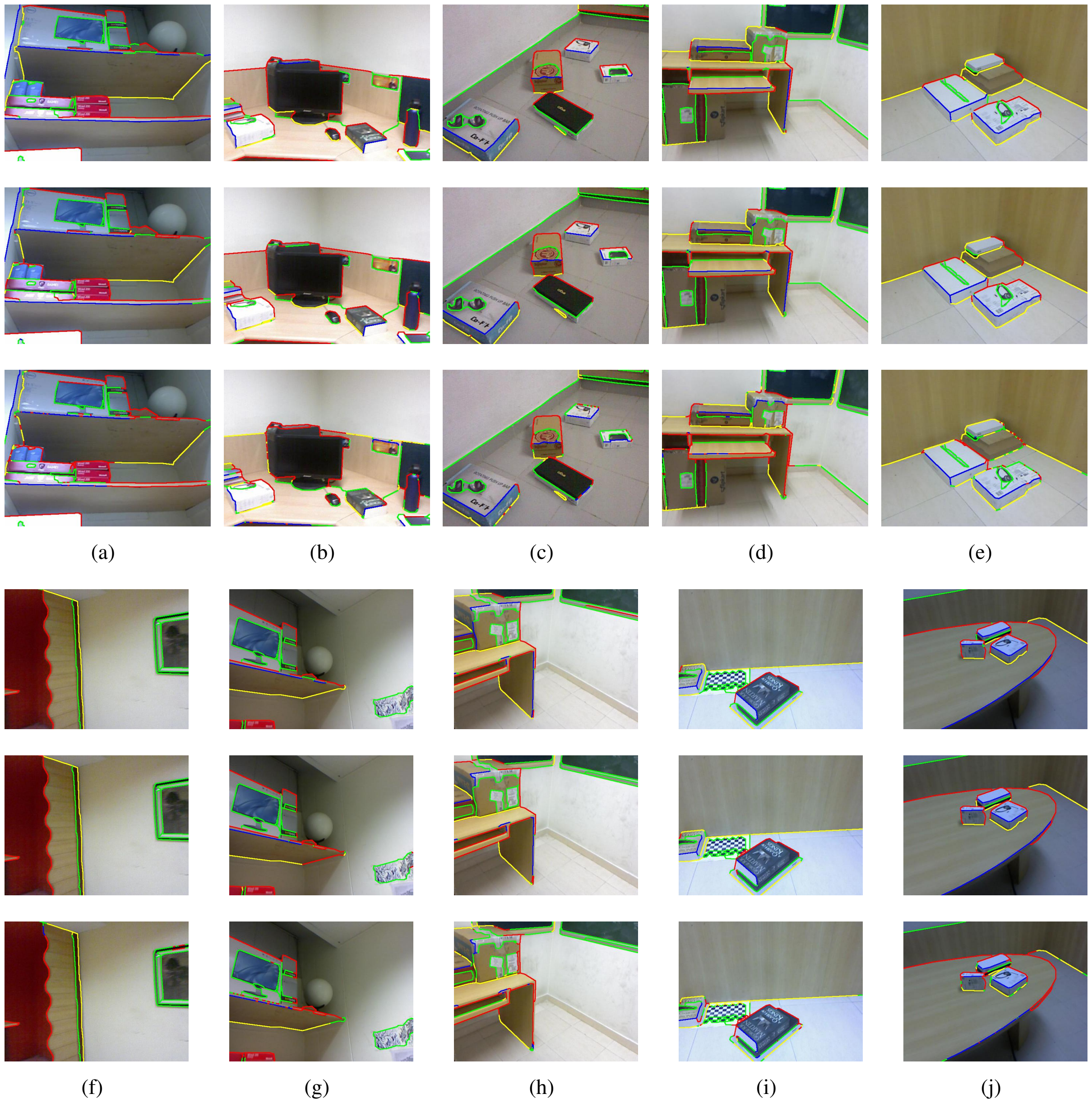


**Figure 3.8** *Ground truths (above) and the corresponding results from our approach* ($2^{nd}$ *row) and Gupta* et al. *[6]* ($3^{rd}$ *row). Color code: red (occ), green (pln), blue (cvx), yellow (ccv).*

*Chapter 4*

# Edge labeling using Stereo

*Chapter 5*

# Conclusion

## Related Publications

# Bibliography

[1] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *PAMI*, 2001.

[2] J. F. Canny. Finding edges and lines in images. Technical report, MIT, Cambridge, MA, 1983.

[3] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[4] D. F. Fouhey, A. Gupta, and M. Hebert. Unfolding an indoor origami world. In *ECCV*, 2014.

[5] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 2010.

[6] S. Gupta, P. Arbelaez, and J. Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *CVPR*, 2013.

[7] V. Hedau, D. Hoiem, and D. Forsyth. Recovering free space of indoor scenes from a single image. In *CVPR*, 2012.

[8] D. Hoiem, A. Efros, and M. Hebert. Geometric context from a single image. In *ICCV*, 2005.

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

[10] P. Isola, D. Zoran, D. Krishnan, and T. Adelson. Crisp boundary detection using pointwise mutual information. In *ECCV*, 2014.

[11] Z. Jia, A. Gallagher, and T. Chen. Learning boundaries with color and depth. In *ICIP*, 2013.

[12] T. Kanade. Recovery of the three-dimensional shape of an object from a single view. *AI*, 1981.

[13] J. Koenderink. Perception. In *What does the occluding contour tell us about solid shape*, 1984.

[14] A. Koschan and M. Abidi. Detection and classification of edges in color images: A review of vector-valued techniques. *Signal Processing Magazine, Special Issue on Color Image Processing*, 2005.

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

[16] J. Malik. Interpreting line drawings of curved objects. *IJCV*, 1987.

[17] J. Malik and D. Maydan. Recovering three-dimensional shape from a single image of curved objects. *PAMI*, 1989.

[18] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 2004.

[19] R. Newcombe, A. 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 *ISMAR*, 2011.

[20] S. Ramalingam and M. Brand. Lifting 3d manhattan lines from a single image. In *ICCV*, 2013.

[21] A. G. Schwing and R. Urtasun. Efficient exact inference for 3D indoor scene understanding. In *ECCV*, 2012.

[22] Q. Shan, B. Curless, Y. Furukawa, C. Hernandez, and S. Seitz. Occluding contours for multi-view stereo. In *IEEE Conference on Computer Vision and Pattern Recognition*, jun 2014.

[23] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

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

[25] K. Sugihara. *Machine Interpretation of Line Drawings*. MIT Press, 1986.