Robust Edge Map Generation From RGB-D Data using PGM and Structured SVM

Abhishek Bafna, Aniket Deshmukh
Department of EECS
University of Michigan
Ann Arbor, MI 48105
{abafna, aniketde} @umich.edu

Abstract

Edge detection plays an important role in the problems of Image Processing and Computer Vision. Recently, there has been some interest in the 3-D edge detection using RGB-Depth (RGB-D) data. With the increasing development and usage of better depth sensors, leveraging this information in an efficient way has become a challenging and important task. In this project we propose a novel approach for robust edge map generation using probabilistic graphical model that captures information from both RGB and depth data. We define two unary potentials, a smoothness potential and a competing lineness potential. We learn parameter weights using a structured support vector machine and perform inference using belief propagation. Results of this novel approach are promising on synthetic and real datasets.

1 Introduction

1.1 Motivation and Problem Statement

Understanding of scenes is a key aspect of computer vision. Edge detection helps us to understand more about the scene structure since the edges mark a clear distinction for a transition from one region with similar properties to another one. This has been a well researched and challenging problem since several decades. There has been some recent interest in exploring edge map generation using RGB-D data. With the increasing development and usage of better depth sensors, leveraging this information in an efficient way has become a challenging and important task. This information should be intended to aid the information provided by RGB camera images thereby creating a holistic understanding of the environment. An example for this is images taken in dark, where RGB images provide little or no information about the surroundings. In this case, the depth map should provide most of the information about the surrounding objects. In [1], a method for fast edge detection in RGB-D images is described that makes use of smooth constraints in orientation. It uses the implicit order of pixel coordinates in RGB space over the point cloud in RGB-D space. In [6], the authors train a deep convolutional neural network (CNN) to identify occlusion edges in images and videos with both RGB and depth inputs.

In this project we create a robust edge map incorporating relevant information from both RGB and Depth maps of an image. We also create a labeled map showing Depth edges specifically within this robust edge map. We describe a novel approach for robust and holistic edge detection from RBG and Depth maps by constructing a probabilistic graphical model between the pixels of the image. We use a modified structured support vector machine (StructSVM) to learn the parameters of the model and perform inference using

belief propagation to generate the edge map. We test the algorithm on synthetic and real data to show promising results. Specifically, we demonstrate the significant usage of the robust edge map in helping visually impaired to detect obstacles like staircases without dependence on ambient light [8, 2, 4].

The rest of the report is organized as follows. Section 2 gives a detailed description of the initial and the improved probabilistic models constructed. Section 3 explains the training of the weights for the potential functions using a StructSVM. Sections 4 describes the interference performed on the graphical model using Belief Propagation. Section 5 describes the experiment setup, datasets used and the results on synthetic and real world data. Finally, we conclude in section 6.

2 Probabilistic Graphical Model

2.1 Initial Model on RGB and Depth edges

Figure 1 shows the proposed flowchart of the algorithm to detect stairs and a initial probabilistic graphical model (PGM) to link the edges between RGB and depth. In this section we describe an initial PGM and in next subsection we explain drawbacks of this model. We define 3 unary potential functions corresponding to link between RGB edge and depth edge - ϕ_o for edge orientation, ϕ_s for edge size and ϕ_l for edge location. We define an indicator random variable δ_{ij} such that $\delta^l_{ij} = 1$ if i^{th} edge in RGB is linked to j^{th} edge in depth. Let's start with the edge i in RGB and define the bounding box around it. All the edges in the corresponding bounding box that will be in depth image are in set jb. These are the possible candidate edges that may be linked. Just using these unary potentials we get the following total energy function between each node i in RGB and each node $j \in jb$ in depth:

$$\phi_{total} = w_l\phi_l + w_o\phi_o + w_s\phi_s \tag{1}$$

where,

$$\phi_l = \frac{(bBoxArea(i) \cap bBoxArea(j))}{(max(bBoxArea(i), bBoxArea(j))}, \tag{2}$$

$$\phi_o = cos(\theta(i) - \theta(j)) \tag{3}$$

$$\phi_s = \frac{(length(i) \cap length(j))}{max(length(i), length(j))}, \tag{4}$$

We collect the training dataset and label the RGB depth map if the particular edge is linked to depth (ie. equal to 1 or is a depth edge) or not (ie. equal to 0 or is a texture/shadow edge). Now let's define the 0-1 loss function to train the weights w_l, w_o and w_s. Instead we use surrogate loss function - hinge loss which is known to be calibrated. Then we use SVM to learn these weights.

2.2 Improved Model on Lattice of Pixels

In the former model, our graph comprised of edges as nodes (random variables). This was a very strong assumption and the properties of the edges could not be extracted with precision. Moreover, there was no linkage between nodes(edges) in the same channel, leading to the creation of several nodal graphs. The only links were between the edges in RGB and depth channels and this graphical structure did not yield good results. Also the model was not scalable and relied heavily upon the reference edge maps.

In this subsection, we propose an improvised graphical model constructed using the pixels in the image as nodes. We define a random variable for each node with states 1 and 0 corresponding to that pixel being an edge point or not. We define two unary and two higher order potentials for each node as described below. Figure [3] shows the new graphical model defined and in the next subsection all potentials are defined.

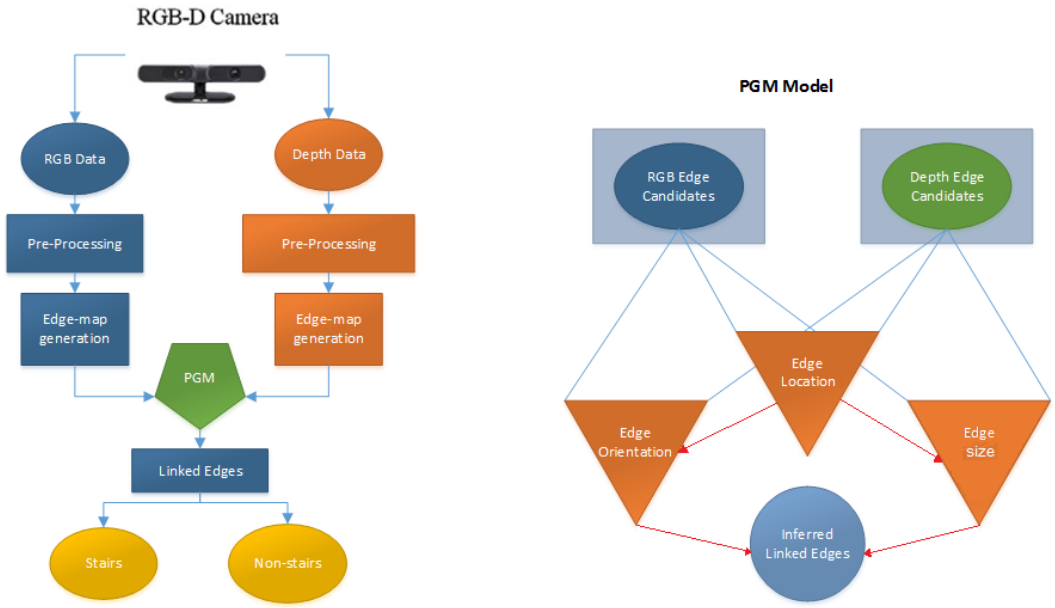

Figure 1: a) left - flowchart, b) right - Initial Graphical Model

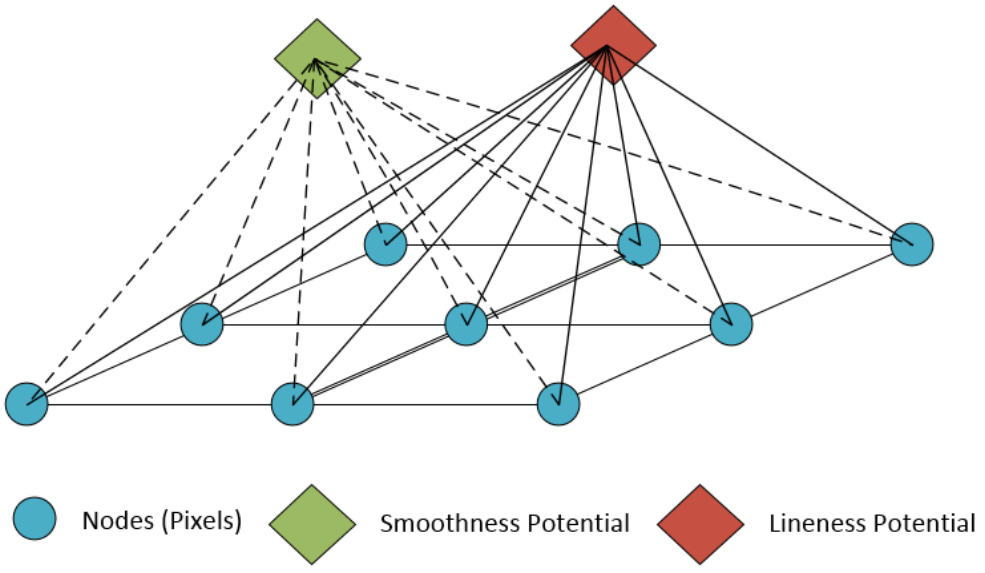

Figure 2: Improved Graphical Model

2.2.1 Unary Potential

Let's take an image of size $n \times m$. So there are total $n \times m$ binary random variables. Let y_i, $\forall i = 1, ..., nm$ be those random variables. We define two unary potentials - one with respect to RGB edge map and one with respect to depth edge map. Let's say x_i represents the information of RGB and depth edge map for node(pixel) i. If the variable takes value 1 then the penalty will be directly proportional to the distance of the that point from the nearest edge point in the edge map. If the variable takes value 0 then the penalty is inversely proportional to the distance of the that point from the nearest edge point in the edge map (defined for both the edge maps). Formally it is defined as follows - Let y_i be a some particular pixel (binary random variable) and d be the distance of the that point from the nearest edge point in the edge map. Let $dmax$ be some threshold (we don't want to penalize too much if nearest edge point is too far in one of the RGB or depth map). The potential of node i with respect to the RGB depth map is thus defined as follows :
If y_i takes value 1 -

$$\phi_{ur}(x_i, y_i) = \begin{cases} d & \text{if } d \leq dmax \\ dmax & \text{if } d > dmax \end{cases} \tag{5}$$

If y_i takes value 0 -

$$\phi_{ur}(x_i, y_i) = \begin{cases} \frac{1}{1+d} & \text{if } d \leq dmax \\ 0 & \text{if } d > dmax \end{cases} \tag{6}$$

The unary potentials with respect to the depth map, $\phi_{ud}(x_i, y_i), i = 1 \dots nm$ are defined similarly.

2.2.2 Smoothness Potential

If the binary random variable y_i is taking a particular value then there is a high probability that random variables in the 8-connected neighbourhood around y_i take the same value. Formally smoothness potential is defined as follows - Let $(\bar{y}_i)$ be the 3X3 lattice structure containing y_i.

$$\phi_s(\bar{y}_i) = \sum_j y_i - y_j \tag{7}$$

2.2.3 Lineness Potential

As the condition for smoothness potential is not always true, we define a competing potential called Lineness potential. Lineness potential discourages blob like structures and encourages line like structures in the 8 point neighbourhood of every pixel. It is defined in terms of the eigenvalues (λ_{max}, λ_{min}) of the covariance matrix formed from the co-ordinates of the edge points(n in number) in the 3X3 lattice $(\bar{y}_i)$. If there are no edge points in the lattice then there is no penalty as this is a valid configuration. If there is only one edge point in the lattice then we incur the highest penalty of 1. If there are two edge points in the lattice then they will always form a line (with one non zero eigenvalue and one zero eigenvalue). So we incur a minor penalty (0.1) if the two edge points are next to each other and a moderate penalty (0.5) if those two points are not consecutive. If there are three or more than three edge points in the lattice and if both of their eigenvalues are not zero then they form a blob like structure (and not a line). So if they have only one non-zero eigenvalue then there is no penalty, otherwise highest penalty of 1. Thus the Lineness potential ϕ_l is defined.

$$\phi_l(\bar{y}_i) = f_l(\lambda_{max}, \lambda_{min}, n) \tag{8}$$

2.2.4 Final Model

Let x_i, y_i be as defined in previous subsections. x_i is the information that we know and we need to infer y_i. We have 4 potential functions - $\phi(x_i, y_i) = [\phi_{ur}(x_i, y_i)\, \phi_{ud}(x_i, y_i)\, \phi_{us}(\bar{y}_i)\, \phi_{ul}(\bar{y}_i)] \in R^4$, where $\phi_{ur}(x_i, y_i)$ is a potential function with respect to RGB edge map, $\phi_{ud}(x_i, y_i)$ is with respect to depth map, $\phi_{us}(\bar{y}_i)$ is smoothness potential and $\phi_{ul}(\bar{y}_i)$ is a lineness potential. Let's

define 4 dimensional column vector w as weights for each potential. Finally, combining all potential functions we need to minimize the following function -

$$\sum_i \min_{y_i} w_1\phi_{ur}(x_i, y_i) + w_2\phi_{ud}(x_i, y_i) + w_3\phi_{us}(\bar{y}_i) + w_4\phi_{ul}(\bar{y}_i) \tag{9}$$

In more compact form -

$$\sum_i \min_{y_i} w^T\phi(x_i, y_i) \tag{10}$$

To solve this optimization function we need to train the weights w and do inference to get y and next 2 sections describe our approach.

3 Training of Weights - Structured SVM

Let x, y, w, and $\phi(x, y)$ be as defined in the previous section. Further let Y_i be the ground truth and y_i be the variable. To train weights w we can either solve max margin SVM type optimization or use some heuristics like cross validation. For the purpose of this project we use SVM to train the weights. Standard binary SVM is used when we want to learn a mapping from flat input space to flat output space. It solves the following optimization -

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + c\sum_i \epsilon_i \\ \text{subject to} \quad & Y_i(w^T x_i) \geq 1 - \epsilon_i, \qquad \epsilon_i \geq 0. \end{aligned} \tag{11}$$

ϵ is a slack variable, which allows the misclassified example (soft margin). This method finds the maximum margin hyperplane while still splitting the two labels as much as possible. Hence the optimization becomes a trade off between a large margin and a small error penalty. But in the current case we have structured input and a structured output. Same ideas of binary SVM are extended for such a structured case and we train weights using StructSVM [7]. It solves the following optimization -

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2}\|w\|^2 + c\sum_i \epsilon_i \\ \text{subject to} \quad & w^T\phi(x_i, Y_i) - w^T\phi(x_i, y_i) \geq \triangle(Y_i, y_i) - \epsilon_i, \qquad \epsilon_i \geq 0. \end{aligned} \tag{12}$$

Where, $\triangle(Y_i, y_i)$ is a loss function between true labels Y_i and predicted label y_i. It's very difficult to solve this problem using standard binary SVM technique as constraints here can be potentially infinite. So, instead [3, 9] gives the cutting plane algorithm, which solves only for most violated constraints. To get such constraints, we solve following optimization for y_i given w -

$$\underset{y_i}{argmax}\ \triangle(Y_i, y_i) - (w^T\phi(x_i, Y_i) - w^T\phi(x_i, y_i)) \tag{13}$$

This is exactly a trade off like binary SVM. We need w which agrees with the minimum margin and have a biggest loss i.e. most wrong prediction. We check if the following equation is satisfied or not. We add it to constraints if it's satisfied.

$$\triangle(Y_i, y_i)(1 - (w^T\phi(x_i, Y_i) + w^T\phi(x_i, y_i)) > \epsilon_i \tag{14}$$

We model this problem as an inference problem by including $\triangle(Y_i, y_i)$ as one of the factor and solve using maximum a posteriori (MAP). We can also use belief propagation here.

4 Inference - Belief Propagation

After learning weights, we perform inference using Belief Propagation or Sum-Product message passing. This results in the calculation of the marginal distribution for each node, (here pixel) in our graph. To calculate the marginal of a single random variable y_i, instead of summing over the other random variables, which becomes computationally prohibitive

Algorithm 1 Training w using StructSVM

Input: $c, \{x_i, Y_i\}, \forall i = 1, ..., n$.
Step 0: Initialize $w, \Omega_i \leftarrow 0, \epsilon \leftarrow 0, \forall i = 1, ..., n$
for until no change in Ω_i: **do**
    for $i = 1, ..., n$: **do**
        $\hat{y} \leftarrow$ solution of $P(13)$
        if $P(14) == 1$ **then**
            $\Omega_i \leftarrow \Omega_i \cup \{\hat{y}\}$
            $(w, \epsilon) \leftarrow$ solution of $P(12)$ using Ω_i constraints
        end if
    end for
end for
return w

very fast, belief propagation allows the marginals to be computed more efficiently exploiting the conditional independencies and graphical structure. Let our factor graph be a bipartite graph containing nodes corresponding to variables V and factors F. The algorithm works by passing real valued functions (messages) along edges between the variables and factors. Let v represent a variable node and a a factor node, y_v be a value taken by the random variable associated with v, X_a be the vector of neighbouring variable nodes of factor a. The message $\mu_{v->a}$ is the product of the messages from all other neighbouring factor nodes $N(v)$.

$$\mu_{v \to a}(y_v) = \prod_{a^* \in N(v) \setminus a} \mu_{a^* \to v}(y_v), \forall y_v \tag{15}$$

A message from a factor node a to a variable node v is the product of the messages received by that factor from all other neighbouring variable nodes $N(a)$ marginalized over the variables except those associated with v.

$$\mu_{a \to v}(y_v) = \sum_{X'_a : y'_v = y_v} f_a(X'_a) \prod_{v^* \in N(a) \setminus v} \mu_{v^* \to a}(y_v) \tag{16}$$

Upon convergence, the estimated marginal distribution of each node is proportional to the product of all messages from adjoining factors.

$$p_{y_v}(y_v) \propto \prod_{a \in N(v)} \mu_{a \to v}(y_v) \tag{17}$$

And the joint marginal distribution of variables in one factor is proportional to the product of the factor and messages from the variables.

$$p_{y_a}(X_a) \propto f_a(X_a) \prod_{v \in N(a)} \mu_{v \to a}(y_v) \tag{18}$$

We combine all modules defined till now in the algorithm 2.

5 Results and Experiments

We first validated our graphical model on a set of manually created RGB-D edge maps reflecting realistic scenarios. Later we proceeded to test it on RGB data collected from an Asus Xtion Pro sensor, for which we developed a extraction code in Python. The images were saved at the rate 25 frames per second for both RGB and Depth images. We use PMTK toolbox [5] for inference and OnlineStructSVM [9] for the implementation of structSVM. Experimental setup is same for all the results below. We first train the parameters for each scenario and and perform inference on that image itself. We do this because our previous model failed even with respect to training error. If this is successful (which it is now) we plan to create extensive training and testing datasets and then train/test on them.

Algorithm 2 Generation of Edge Map

Input: I RGB image, D depth image, y Ground truth edge map for training
$x = \{E_I, E_D\} \leftarrow$ Run Canny edge detector on I, D to get edge maps
for $i = 1, ..., n$: **do**
    $\phi(x_i, y) \leftarrow$ Compute all four potentials using the improved model
end for
Create a factor graph using $\phi(x_i, y), \forall i = 1, ..., n$
$w \leftarrow$ train weights using algorithm 1
$\hat{y} \leftarrow$ solution of P(10) using belief propagation
return $\hat{y}$

5.1 Synthetic Data

Results on synthetic data are shown in figure [3]. Here we created different test cases to see if proposed method works in all cases.

Test Cases:

- There is an edge in RGB but no edge in depth. Ground truth is RGB.
- RGB doesn't have a complete edge but depth has a complete edge. Ground truth is depth.
- RGB has an accurate edge but depth has erroneous edge. Ground truth is RGB.
- Both RGB and depth have edge but the edge in RGB is not in depth and edge in depth is not in RGB. Both of these edges are far away. Ground truth is logical operation - RGB or depth.
- Depth has an accurate edge but RGB has erroneous edge. Ground truth is depth.
- Depth has an accurate edge but RGB has erroneous edge. Ground truth is depth.
- RGB edge and depth edge are adjacent. Ground truth is either RGB or depth but we have a preference to RGB.
- RGB has an extra edge. Ground truth is RGB.
- RGB has some part of the edge and depth has some part of the edge. edge is vertical. Ground truth is logical operation - RGB or depth.
- RGB has some part of the edge and depth has some part of the edge. edges are at some angle. Ground truth is logical operation - RGB or depth.

From figure [3], we can see that proposed method works for all the cases but case number 4. Though this case is going to be rare we need to modify a unary potential so that it works on this case too.

5.2 Real Data

We performed experiments on the dataset from the Xtion. We need ground truth to learn the weight parameters and it's very time consuming to label each image in the dataset. So to experiment for now, we just gave RGB as ground truth. Using the trained weights from StructSVM, we could infer edge map accurately. Then based on this new edge map and depth edge, we labelled the edges in new edge map as "depth" or "no depth".

5.3 Application - Navigation System for the visually impaired

We test our robust labelled edge map for detecting stairs. This will be helpful for navigation for visually impaired people. After obtaining the holistic labeled edge map using algorithm 2, features are extracted from each edge namely location and slope for both depth-edges and non-depth edges. The presence of consecutive parallel depth-edges alerts the user of a staircase. The detection of the same with non depth images would suggest the presence

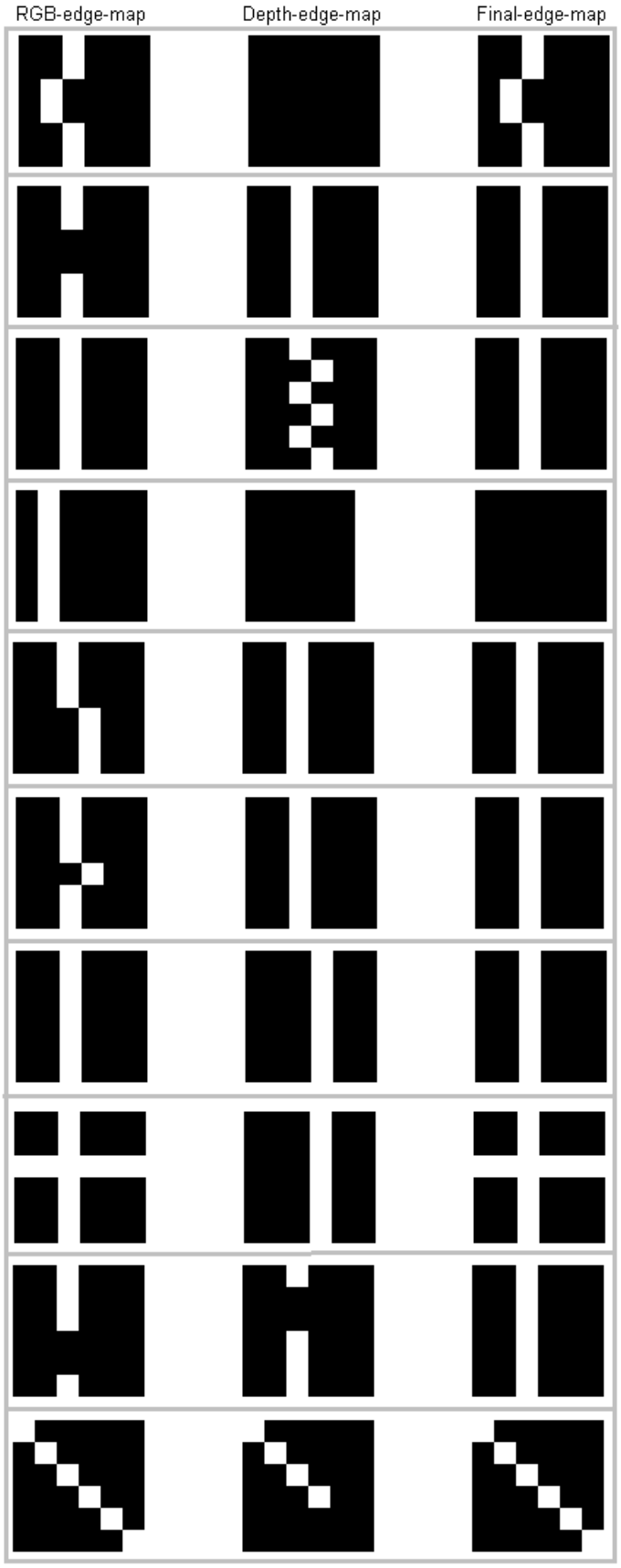

Figure 3: Results on Syntehtic Data

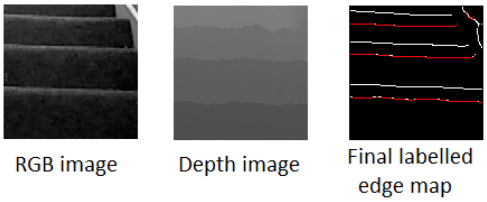

Figure 4: Results on Syntehtic Data

of a textured pattern, in our case a cross-walk. Figure [4] shows the result for one instance for the same.

6 Conclusions and future work

We proposed a novel approach for detecting edge map of RGB-D data using a probabilistic graphical model. We use two unary potentials and two higher order potentials to capture the information of given data and dependencies respectively. We then use StructSVM to learn the parameters and perform inference by belief propagation to get the final edge map. Right now we do this on single images separately and see if it works. We show the effectiveness and robustness of our method on synthetic as well as real life data with respect to training error. We also show result on the navigation application to help visually impaired person to detect staircases. Our method works on most of the cases as shown in synthetic dataset but fails for a particular case when all RGB edges are absent in depth edge map as well as all depth edges are absent in the RGB edge map. Also, as our method is applied to each pixel in the image, it's computationally very expensive. We need to figure out the way to reduce the computational complexity. From the best of our knowledge our method is the first to use probabilistic graphical model for the generation of robust edge map and it has shown very promising results. So there are 3 things that we need to do to give very strong results - 1) Apply it on a bigger and broad spectrum of datasets. Build a rich training and testing dataset and then report the error. 2) Modify the potential function so that we even get the correct result for the case 4 of synthetic dataset, 3) Make it computationally efficient.

7 Acknowledgement

We would like to thank Vikas Dhiman for his useful inputs while defining the PGM.

References

[1] H. Casarrubias-Vargas, A. Petrilli-Barceló, and E. Bayro-Corrochano. Fast edge detection in rgb-d images. In *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*, pages 868–875. Springer, 2014.

[2] L. A. Guerrero, F. Vasquez, and S. F. Ochoa. An indoor navigation system for the visually impaired. *Sensors*, 12(6):8236–8258, 2012.

[3] T. Joachims, T. Finley, and C.-N. J. Yu. Cutting-plane training of structural svms. *Machine Learning*, 77(1):27–59, 2009.

[4] Y. H. Lee and G. Medioni. A rgb-d camera based navigation for the visually impaired. In *RSS 2011 RGBD: Advanced Reasoning with Depth Camera Workshop*, pages 1–6, 2011.

[5] K. Murphy and M. Dunham. Pmtk: Probabilistic modeling toolkit. In *Neural Information Processing Systems (NIPS) Workshop on Probabilistic Programming*, 2008.

[6] S. Sarkar, V. Venugopalan, K. Reddy, M. Giering, J. Ryde, and N. Jaitly. Occlusion edge detection in rgb-d frames using deep convolutional networks. *arXiv preprint arXiv:1412.7007*, 2014.

[7] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, page 104. ACM, 2004.

[8] S. Wang, H. Pan, C. Zhang, and Y. Tian. Rgb-d image-based detection of stairs, pedestrian crosswalks and traffic signs. *Journal of Visual Communication and Image Representation*, 25(2):263–272, 2014.

[9] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba. Recognizing scene viewpoint using panoramic place representation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2695–2702. IEEE, 2012.