ECSE 626 Course Project : A Study in the Efficient Graph-Based Image Segmentation

Chu Wang
Center for Intelligent Machines
chu.wang@mail.mcgill.ca

Abstract

In this course project, I will investigate into the Efficient Graph-based Image Segmentation [2]. This segmentation method is very fast and achieved good result as a raw segmentation. The superpixels it acquired can be further applied to refine the segmentation by using more sophisticated techniques [4]. As for implementation, I utilized P. Felzenszwalb's code and added new functionalities to it. Specifically, I added the ability to incorporate depth data from Microsoft Kinect into the segmentation algorithm and demonstrated that the additional channel of depth map in the RGB-D point cloud data can actually boost the segmentation performance compared to using standalone RGB image. Meanwhile, different weight functions characterizing the difference between neighbourhood pixels are tested and a reverse Gaussian weighting function designed by myself achieved better result for segmenting point cloud data. Finally, segmented results are provided on two different point cloud datasets, which are UW RGB-D dataset [3] and NYU scene v2 dataset [4].

1. Introduction

Image segmentation can be viewed as equivalently a pixel labelling problem. Each pixel in the image can be assigned with a label and the label set is finite. In [2], the image is viewed as a Graph where pixels are the vertices and edges are constructed by connecting neighbouring pixels. The graph based approach creates a segmentation by partitioning the vertices into components where each component is connected and pixels in one component share the same label. An iterative algorithm loops through all edges in the graph, and examines conditions based on energy functions to determine whether to merge the two components connected by this edge or not. The details of the energy functions concerning the two components connected by the edge of current iteration will be further discussed in section (2). The nature of the algorithm is a greedy algorithm and it is very efficient. However, it possesses good convergence properties [2] and gives nice segmentation results in practice.

To make the efficient graph based segmentation algorithm more versatile, the ability to segment point cloud data gathered from Microsoft Kinect is built as the first contribution of this project. Kinect's RGB-D point cloud data provides much richer information than standard color camera's RGB data by adding one dimension of depth map. Unlike color images, the additional dimension of depth map is robust to the variations of lighting or colors, provides shape cues and separates background and foreground by depth discontinuities. As we expected, the informative depth map boosted the segmentation performance when keeps other parameters of the algorithm the same as using standalone RGB data. Further experimental results can be found in section (5)

The energy functions applied in [2] is based on weights of edges in the graph. Therefore one key design perspective, which is another contribution of this project, is to choose a proper weight function of an edge $e = (v_i, v_j)$ in the graph V to segment point cloud data. One naive idea would be simply add depth as another channel and calculate the Euclidean distance between the neighbouring pixels $p_1 = (r_1, g_1, b_1, d_1)$ and $p_2 = (r_2, g_2, b_2, d_2)$. However, a more reasonable idea would be utilizing the depth discontinuity to emphasize the difference between p_1 and p_2. A reverse Gaussian function is applied to emphasize edge weight when the depth discontinuity between the two pixels is large and penalize the weight when the depth discontinuity is small. We will address this issue in full detail in section (4).

The rest of the report is organized as follows. In section (2), we will introduce the theories behind the efficient graph-based method and the outline of the algorithm. In section (3), we will provide analysis over the algorithm's probabilistic aspect and how it is related to Maximum A Posteriori. Then the improvement of the algorithm and main contribution of this project will be illustrated in section (4). Experimental results of the improved efficient

graph based segmentation algorithm are provided in section (5). Some acknowledgement and future work are discussed in section (7).

2. Graph-Based Segmentation Algorithm

In order to label each pixel in a 2D image, Felzenszwalb constructed a graph $G = (V, E)$ where the vertices V are composed of pixels $\mathcal{P}$ and edges $(v_i, v_j) \in E$ correspond to pairs of neighbouring vertices. Associated with each edge (v_i, v_j), an energy weight $w(v_i, v_j)$ is assigned which measures the disagreement between neighbouring vertices. A component C of vertices form a group of vertices and they have a internal energy which is defined as the largest weight w in the minimum spanning tree $MST(C, E)$ of the component, which is

$$Int(C) = \max_{e \in MST(C,E)} w(e). \tag{1}$$

Intuitively, by comparing the Internal energy with the disagreement measurement

$$Dif(C_1, C_2) = \min_{v_i \in C_1, v_j \in C_2, (v_i, v_j) \in E} w(v_i, v_j) \tag{2}$$

between components C_1 and C_2 ,which is the minimum weight of any edge connecting the two, one can decide whether to merge the two component or not. The key element in both internal energy and external difference measurement is the edge weight function $w(\cdot)$. The edge weight measures the difference between neighbouring pixels. The merging between two components C_1 and C_2 will take place if the minimum internal energy $MInt(C_1, C_2)$ defined as

$$MInt(C_1, C_2) = \min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2)) \tag{3}$$

is larger than the external difference $Dif(C_1, C_2)$ defined in equation (2). A threshold function is applied as

$$\tau(C) = \frac{k}{|C|} \tag{4}$$

in the minimum internal energy where $|C|$ denotes the component size of C. This threshold serves as a control measurement for the merging process. For small components, they require a stronger evidence for a boundary to stop the merging. If the external difference is not large between two small components, it will be more likely that

$$Dif(C_1, C_2) < MInt(C_1, C_2) \tag{5}$$

because $\tau(C) = \frac{k}{|C|}$ can be pretty large for small components. Therefore small components tend to merge with each other until it reach a point that the boundary evidence is high enough to stop merging.

Formally, Felzenszwalb applied a algorithm similar to Krushal's algorithm(the famous greedy algorithm for constructing minimum spanning tree of a graph) to iteratively merge the components after a initial assignment. The algorithm checks if the difference measurement $Dif(C_1, C_2)$ between two components is smaller than both internal energies, then merge C_1 and C_2 else do nothing. By acquisition of component pairs using edge pairs with non decreasing weights, the algorithm works in $O(|E| \log |E|)$ runtime complexity where $|E|$ is the number of edges in the graph. The detail processes of the efficient graph-based segmentation method is presented in algorithm (1).

Algorithm 1 Efficient Graph Based Algorithm

procedure EFFICIENTSEG
  input ← Graph = (V,E)
  $\pi = (o_1, ..., o_m)$ ← Sort E
    by non-decreasing edge weight
  S^0 ← *a segmentation where each vertex*
    v_i *is in its own component*
  for $q = 1, ..., m$ **do** :
    $S^q \leftarrow S^{q-1}$
    $v_1 \leftarrow o_q.start$
    $v_2 \leftarrow o_q.end$
    $C_1 \leftarrow v_1$'s Component in S^{q-1}
    $C_2 \leftarrow v_2$'s Component in S^{q-1}
    if $w(o_q) \leq MInt(C_1, C_2)$ **then**
      $S^q \leftarrow$ *merge(* C_1, C_2,S^{q-1}*).*
  return $S \leftarrow S^m$

3. Statistical Theory for the Graph-Based Segmentation Algorithm

The probabilistic theory behind the algorithm is actually Maximum A Posteriori and we have the following lemma.

Lemma 3.1. *The iterative greedy algorithm in section (2) IS minimizing a energy term like*

$$E = E_{data} + E_{smooth}. \tag{6}$$

Here we have

$$E_{data} = \sum_{i \in \mathcal{C}} Int(C_i) \tag{7}$$

$$E_{smooth} = \sum_{i,j \in \mathcal{C}} Dif(C_i, C_j) + \sum_{i \in \mathcal{C}} \frac{k}{|C_i|}. \tag{8}$$

Proof. We provide the reasoning over this as follows. The iterative algorithm loops through all the edges in the graph by descendent order, and merge the components C_1, C_2 connected by the edge in current iteration if the internal difference of one component is smaller than the edge's weight

$w(v_1, v_2)$, which is

$$w(v_1, v_2) \leq \min(Int(C_1) + \frac{k}{|C_1|}, Int(C_2) + \frac{k}{|C_2|}). \quad (9)$$

The energy concerning only C_1 and C_2 before merging in this iteration is

$$E_{old} = Int(C_1) + Int(C_2) + Dif(C_1, C_2) + \frac{k}{|C_1|} + \frac{k}{|C_2|} \quad (10)$$

and the energy after merging is

$$E_{new} = Int(C_1 \cup C_2) + \frac{k}{|C_1| + |C_2|}. \quad (11)$$

Then if

$$w(v_1, v_2) \leq \min(Int(C_1) + \frac{k}{|C_1|}, Int(C_2) + \frac{k}{|C_2|}), \quad (12)$$

and suppose without loss of generality $Int(C_1) + \frac{k}{|C_1|} > Int(C_2) + \frac{k}{|C_2|}$, we are guaranteed from the properties of Minimum Spanning Tree that

$$Int(C_1 \cup C_2) = \max(Int(C_1), Int(C_2), w(v_1, v_2)) \quad (13)$$

$$\leq Int(C_1) + \frac{k}{|C_1|}. \quad (14)$$

Together with

$$\frac{k}{|C_1| + |C_2|} < \frac{k}{|C_2|}, \quad (15)$$

we have

$$E_{new} = Int(C_1 \cup C_2) + \frac{k}{|C_1| + |C_2|} \quad (16)$$

$$< Int(C_1) + \frac{k}{|C_1|} + \frac{k}{|C_1| + |C_2|} \quad (17)$$

$$< Int(C_1) + \frac{k}{|C_1|} + \frac{k}{|C_2|} \quad (18)$$

$$< E_{old}. \quad (19)$$

Therefore by satisfying the condition (12), we are guaranteed the energy concerning only C_1 and C_2 will decrease through merging the two components. The global energy E is still decreased because of the following reasons.

1. The difference between the new global energy and the old global energy lies in the **local difference** concerning C_1 and C_2 as well as **the external difference** term $Dif(C_i, C_j)$ concerning remaining components with C_1, C_2 and $C_1 \cup C_2$.

2. We already showed the local difference is decreased by merging. We only need to show the external difference term is non-increasing. We have

$$\Delta E_{ext} = E_{ext}^{new} - E_{ext}^{old} \quad (20)$$

and

$$E_{ext}^{new} = \sum_{i \in \mathcal{C}_{new}} Dif(C_i, C_1 \cup C_2) \quad (21)$$

$$E_{ext}^{old} = Dif(C_1, C_2) \quad (22)$$

$$+ \sum_{i \in \mathcal{C}_{old}} Dif(C_i, C_1) + Dif(C_i, C_2). \quad (23)$$

3. The edge connecting C_i and $C_1 \cup C_2$ is unchanged after merging C_1, C_2. Therefore we have

$$\sum_{i \in \mathcal{C}_{new}} Dif(C_i, C_1 \cup C_2) = \quad (24)$$

$$\sum_{i \in \mathcal{C}_{old}} Dif(C_i, C_1) + Dif(C_i, C_2). \quad (25)$$

This gives us

$$\Delta E_{ext} \leq 0. \quad (26)$$

Therefore the global energy E is decreased after merging with condition (12). Proof finished. □

Theorem 3.2. *Minimizing the energy term as defined in equation (6) is equivalent to maximizing the posterior probability of the pixel labelling assigned by components set $\mathcal{C}$.*

Proof. The posteriori probability of a labelling

$$P(Label|Data) \propto P(Data|Label)P(Label) \quad (27)$$

can be modelled by defining $P(Data|Label)$ and $P(Label)$ as an exponential form like $e^{-E_{data}}$ and $e^{-E_{smooth}}$ by assuming they follow Gibbs distribution.

Then the logarithm of the posteriori (27) becomes a negative form of the energy term (6). Therefore minimizing the energy function IS actually maximizing the Posterior probability of our labelling. □

Based on lemma (3.1) and theorem (3.2) , the efficient graph-based algorithm [2] is iteratively maximizing the posterior probability of the labelling assigned by components set $\mathcal{C}$. Therefore it follows the idea of Maximum A Posteriori and has its soundness in statistical aspect.

4. My Improvements

The efficient graph based algorithm (1) was once the state of the art segmentation algorithm. And now it is still very viable to provide a raw segmentation for more sophisticated segmentation methods, like support inference based method in [4]. **The main contribution of this project is to investigate into the potential improvements of the efficient graph-based algorithm.**

The first potential will be incorporating 3D information from point cloud data to the graph-based segmentation pipeline. The emergence of Microsoft Kinect brings a revolution in computer vision community. RGBD data becomes much easier to collect than using a laser range finder and has much less noise in depth map. Recent works has demonstrated the power of depth data to boost segmentation performance [4] and recognition performance [3]. The point cloud data from Microsoft Kinect contains standard RGB and depth data. The additional dimension of depth map, which is robust to the variations of lighting or colors, provides 3D shape cues and separates background and foreground by depth discontinuities. Therefore I have confidence that if appropriately combine the RGB data with depth information in the Graph based segmentation method [2], we can improve the performance of the algorithm. We provide a illustration of RGB and depth data in UW dataset [3] in Figure (1).

(a) RGB image

(b) Depth map

Figure 1: RGB images and depth map in UW RGBD dataset. The example is a banana.

An example of NYU scene v2 dataset is provided in Figure (2).

The second potential of improvement is a proper design of edge weight function $w(v_i, v_j)$ to smartly incorporate point cloud data. The energy functions applied in section (2) is highly based on weights of edges in the graph $G = (V, E)$ and the algorithm's merging process

(a) RGB image

(b) Depth map

Figure 2: RGB images and depth map in NYU scene v2 dataset. The example is a scene of a classroom.

is based on comparing these energies. Therefore the edge weight function is a fundamental element in the segmentation method and a good choice with proper reasoning will provide a better segmentation result. A naive edge weight function will be simply add depth as another channel and calculate the Euclidean distance between the neighbouring pixels $v_i = (r_i, g_i, b_i, d_i)$ and $v_j = (r_j, g_j, b_j, d_j)$. We define it as

$$w_E(v_i, v_j) = \sqrt{(v_i - v_j)(v_i - v_j)^T} \tag{28}$$

The Euclidean distance edge weight function does incorporate depth map into the segmentation criteria, however the emphasize on depth discontinuity is not enough. Wherever there is a large depth discontinuity, the edge weight should be always large. If the RGB value between v_i and v_j is the same while the depth change is pretty significant, the edge weight (28) could be not large enough to justify an existence of a boundary. Therefore, a more reasonable idea would be further utilizing the depth discontinuity to emphasize the Euclidean difference between v_i and v_j. A reverse Gaussian function is applied to emphasize edge weight when the depth discontinuity between the two pixels is large and penalize the weight when the depth discontinuity is small. We define the Gaussian weighted edge weight

as

$$w_G(v_i, v_j) = (\tau - \mathcal{G}(d_i - d_j))\sqrt{(p_i - p_j)(p_i - p_j)^T} \tag{29}$$

$$\mathcal{G}(d_i - d_j) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(d_i - d_j)^2}{2\sigma^2}} \tag{30}$$

$$p_i = (r_i, g_i, b_i) \tag{31}$$

$$p_j = (r_j, g_j, b_j). \tag{32}$$

where τ and σ are parameters for reverse Gaussian weighting. A show case for reverse Gaussian mask

$$\mathcal{M} = \tau - \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x)^2}{2\sigma^2}} \tag{33}$$

is presented in Figure (3).

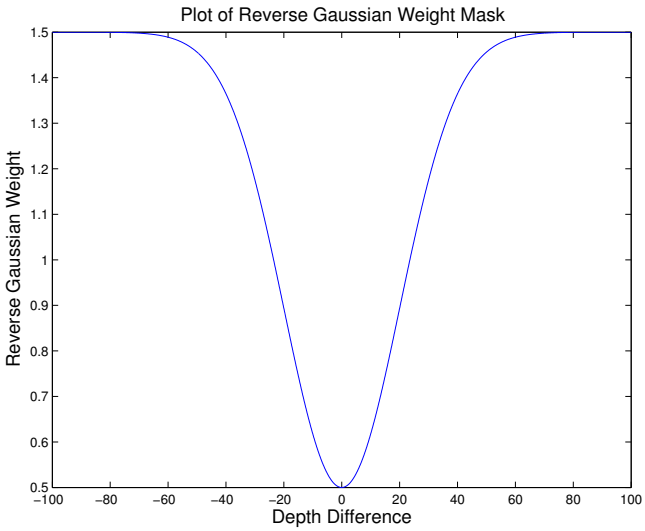


Figure 3: Plot of the Reverse Gaussian Mask applied in edge weight function w_G. $\sigma = 20$ and $\tau = 1.5$.

In [2], $w(v_i, v_j)$ are simply modelled as the difference between pixel RGB intensities which is similar to the Euclidean edge weight defined as (28). A more sophisticated way like reverse Gaussian edge weight should have the potential to improve segmentation performance since it better utilized depth data, but the more complicated the weight function be, the more runtime the algorithm may require.

5. Results

In the implementation of this project, I utilized Felzenszwalb's code available on his website [1] and modified it for processing point cloud data. Details of my modification to the code can be found in section (4) as well as my code submission for this project. In this section, we will present and compare the segmentation results by using standalone RGB data, point cloud with Euclidean distance edge weight w_E and point cloud with reverse Gaussian edge weight w_G. We will present the segmentation results with different settings in a figure array with proper captions and titles. We first present the segmentation results on UW RGBD dataset [3] in Figure (4). We then present the segmentation results on NYU scene v2 dataset [4] in Figure (5).

Due to the size of the figure arrays, Figure (4) and (5) will appear on the next page.

6. Discussion and Conclusion

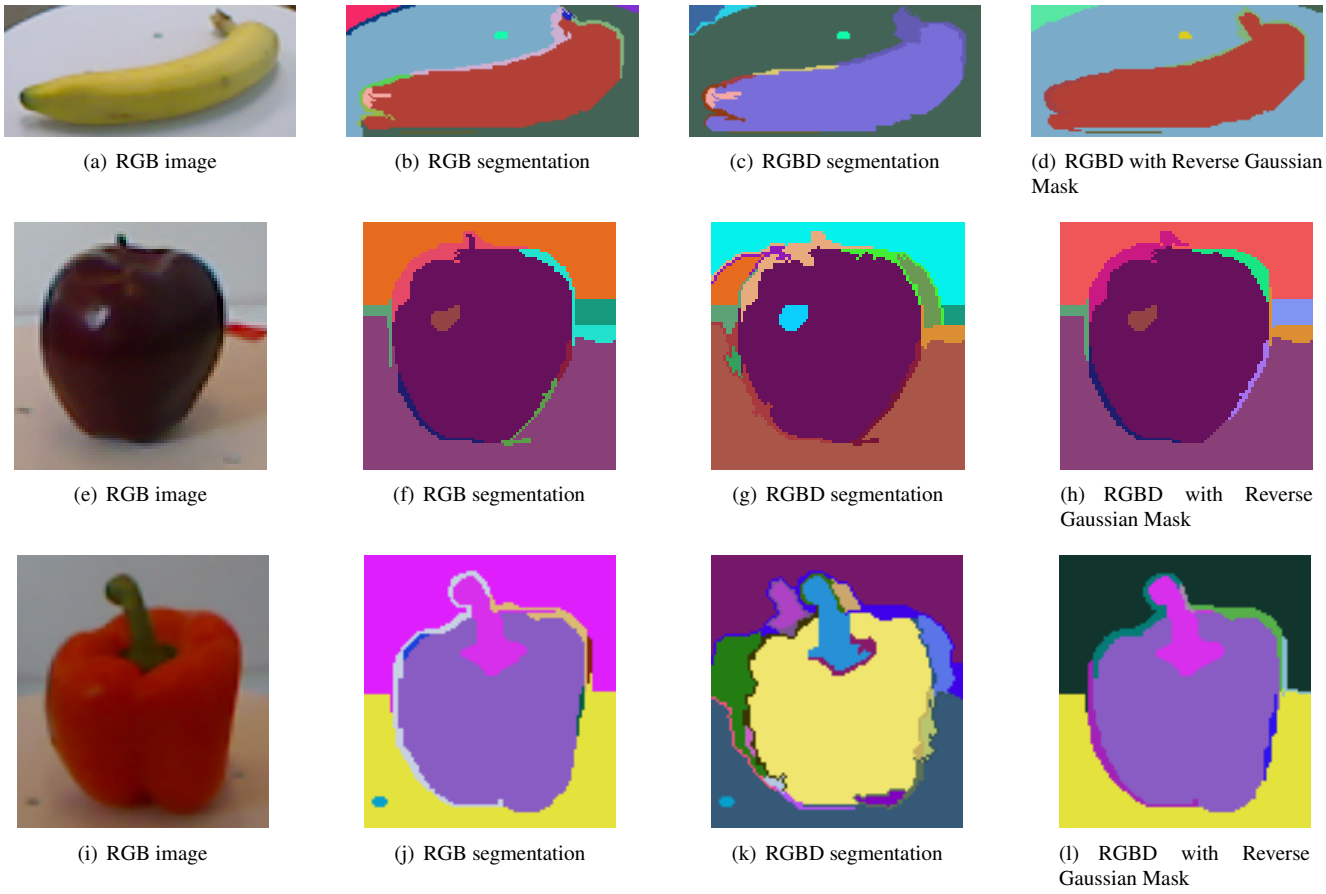
In this section, we will discuss over the comparison results in Figure (4) and (5). We have the following observations and conclusions

1. **Segmentation with Euclidean distance edge weight w_E usually generates noisy segmentations by using RGBD point cloud.** In this case, we get low segmentation improvement from depth map information. The result of using Euclidean distance edge weight is sometimes even worse than using standalone RGB data. This fact is clearly supported by comparing Figure (4(f)) and (4(g)). There are a lot of noisy regions surrounding the apple in Figure (4(g)) which uses Euclidean distance edge weight in the segmentation method. The similar situation holds for bell pepper case by inspecting Figure (4(k)). This is due to the fact that the depth data is pretty noisy and there are a lot of black holes in the Kinect depth data, which is quite a common issue of the point clouds. To illustrate the quality of the depth map, we present the depth map extracted from the point cloud data of apple and bell pepper in Figure (6).

(a) Apple (b) Bell Pepper

Figure 6: Noisy Depth Maps acquired by Kinect.

2. **The reverse Gaussian weighted edge energy w_G gives better segmentation result.** In the UW dataset, we found the segmentation results with reverse Gaussian weighted edge energy always outperforms the other two settings. The w_G energy clearly succeeded in emphasizing the depth discontinuities while still

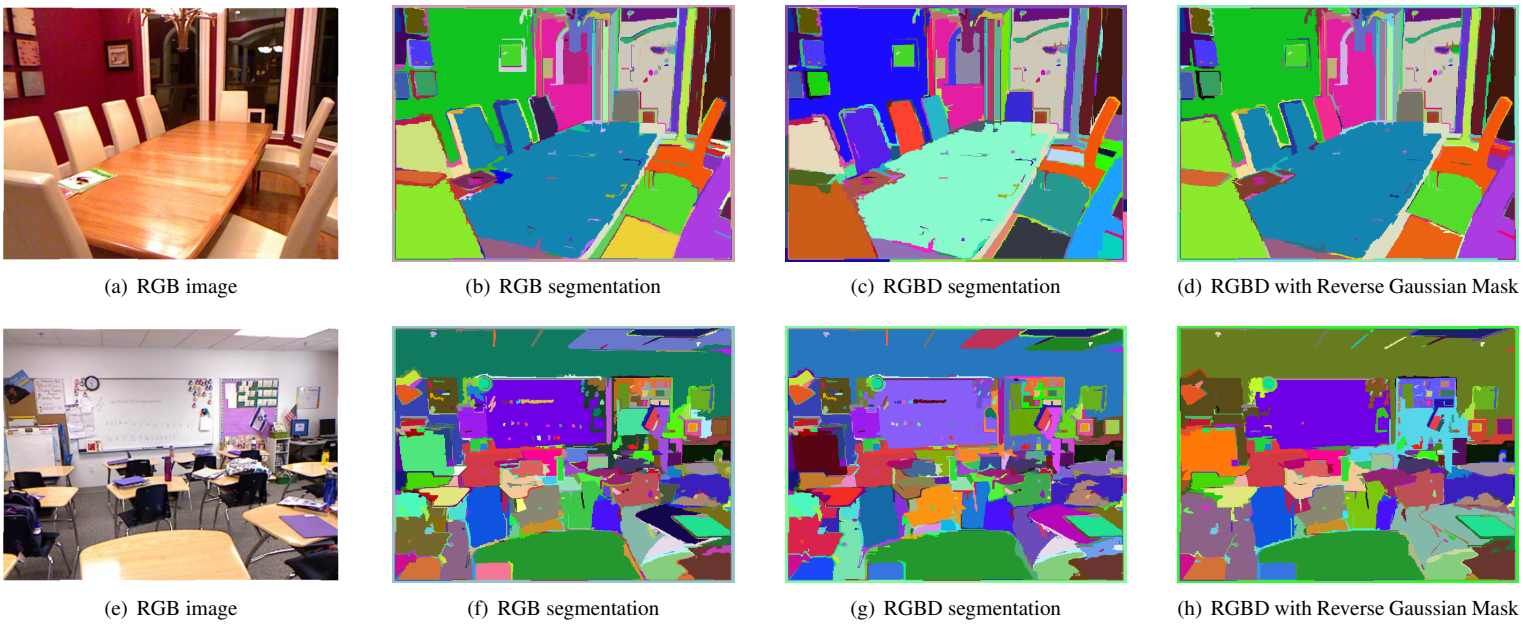


(a) RGB image (b) RGB segmentation (c) RGBD segmentation (d) RGBD with Reverse Gaussian Mask

(e) RGB image (f) RGB segmentation (g) RGBD segmentation (h) RGBD with Reverse Gaussian Mask

(i) RGB image (j) RGB segmentation (k) RGBD segmentation (l) RGBD with Reverse Gaussian Mask

Figure 4: Segmentation results comparison for UW RGBD dataset.



(a) RGB image (b) RGB segmentation (c) RGBD segmentation (d) RGBD with Reverse Gaussian Mask

(e) RGB image (f) RGB segmentation (g) RGBD segmentation (h) RGBD with Reverse Gaussian Mask

Figure 5: Segmentation results comparison for NYU scene v2 dataset.

preserves necessary difference between neighbourhood pixels when depth change is small. By comparing Figure (4(l)) with Figure (4(j)), we notice the noisy point in the left corner disappeared by using the reverse Gaussian weighting. The boundaries around the apple in Figure (4(h)) is also smoother and less noisy than its RGB segmentation counter part.

3. **Segmentation with reverse Gaussian weighted edge energy w_G will perform worse when the size of the point cloud is large and the scene is complex.** By inspecting the segmentation results in the first example of NYU scene comparison in Figure (5), we found we are getting improvements by using reverse Gaussian weighting by eliminating the noise on the table. However, when the scene becomes more complicated where there are multiple occlusions, the reverse Gaussian weighting segmentation tends to "coarse-segment" the point cloud by merging many distinct regions. In Figure (5(d)), we found that a lot of tables are connected with the ground as one large components and objects on the chairs are merged with the chair themselves.

7. Critiques and Future Work

While the success on UW dataset demonstrated the contribution of my improvement to the efficient graph-based algorithm, the problematic performance in the second scene of NYU dataset clearly indicates we still need improvements to make the reverse Gaussian weighting idea to work on complex scenes. The first potential of future work would be using adaptive Gaussian functions in reverse Gaussian weighting Mask. The Gaussian function applied in the weighting mask (29) of w_G is uniform throughout the point cloud, and an intuitive idea would be adapting the σ and τ parameter based on the local information of the point cloud. A large $sigma$ would make no sense if there is a lot of subtle yet important depth changes in a local area, say a table. The adaptive scale of the reverse Gaussian mask would have the potential to capture the details on the small class tables in Figure (5(a)) and we would be able to isolate these objects instead of merging them. The second future improvement would be incorporating higher order shape information in the segmentation algorithm. The surface normals and curvature information are very distinctive between different parts of a scene and they possess the potential to further boost the segmentation performance.

By the end of the report, I would give sincere thanks to Prof. Tal Arbel for her kind help on my questions and her excellent lectures. I am also very grateful to Dr. Meltem Demirkus who helped me on my assignments and provided valuable feedbacks on my course works.

References

[1] P. F. Felzenszwalb. Efficient Graph-based Image segmentation C++ Code. http://cs.brown.edu/~pff/, 2007. [Online]. 5

[2] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004. 1, 3, 4, 5

[3] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011. 1, 4, 5

[4] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1, 4, 5