Graph Cut Segmentation of Range Images into Planar Regions

Simon Muller and Willie Brink
Applied Mathematics
Department of Mathematical Sciences
University of Stellenbosch, South Africa
Email: {samuller, wbrink}@sun.ac.za

***Abstract*—We present a method for segmenting range images into separate planar surfaces, through a novel use of graph cuts. Our method attempts to assign an element from a discrete set of normal vectors to every pixel in the image, in an optimal way that considers the observed data as well as a smoothness prior. We show results from executing our method on a standard range image dataset and evaluate performance quantitatively. We find that our method's accuracy compares well with other existing methods. Moreover, sensitivity to the single parameter that needs to be specified appears to be quite low.**

I. Introduction

Image segmentation is a common problem in the field of computer vision. If successfully achieved, the segments can be used as symbols in higher-level description or interpretation tasks. Unfortunately the segmentation problem can be extremely difficult, partially because successful (or accurate) segmentation is not a well-defined concept and may have more than one "correct" answer for a given image. The significance of this issue is substantiated by the large variations found when humans are tasked to segment images manually [1]. In the case of range images the problem becomes one of segmenting geometry instead of intensities and, as such, becomes somewhat less subjective.

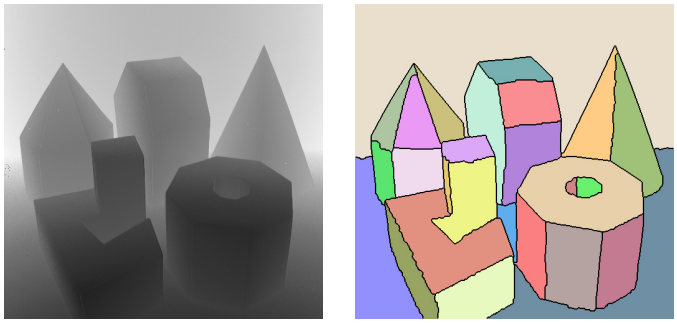
Range images, also referred to as depth images, are images in which every pixel's value represents the distance between the sensor and a 3D object or scene point. Fig. 1(a) shows an example, where distances are depicted as greyscale intensities. We can regard range images to be much less ambiguous than colour images, since the colour of a pixel can be influenced by an intricate combination of lighting conditions, texture and reflection properties of surfaces, the viewing angle, etc. Common methods for creating non-synthetic range images include stereo vision, time-of-flight and structured light. Range imaging has also become more accessible with the recent development of the Kinect (an inexpensive, consumer grade, real-time structured light sensor).

A typical approach to segmentation is to identify object boundaries, or edges. In range images surface edges are commonly categorized into two types: jump edges and crease edges [2]. Jump edges occur at depth discontinuities in the range data. Crease edges occur where differently orientated surfaces meet, and are harder to identify because the surfaces have the same depth along such an edge. The only way to identify crease edges is to consider change in surface normals, which can be problematic due to small amounts of noise in the range image translating to more extreme noise in the normals.

Range image segmentation has the goal of segmenting the discrete points in a range image into regions of points that belong to the same surface. Current approaches seem to either use region growing and clustering, or focus on edge detection [3]. A notable exception is the work by Han et al. [4]. Although they also incorporate edge detection, it is used only to improve execution time. Similar to our work, they set up and minimize an energy function that describes how correct a given segmentation seems. Where we use graph cuts to optimize, they employ simulated annealing.

An automated framework for quantitatively evaluating results from range image segmentation algorithms was developed by Hoover et al. [2], [5]. It provides a methodology for comparing results against ground truth by defining and measuring five performance metrics. A dataset of range images captured by different methods is also provided. All of these images contain scenes that consist only of polyhedral shapes and should therefore segment into planar surfaces (the image in Fig. 1(a) comes from this set). Some follow-up papers build upon their work by doing the same for images containing curved surfaces [6], [7].

The segmentation method we propose in this paper works by labelling every pixel with one of a number of normal vectors.



(a) input range image (b) our segmentation

Fig. 1. A range image visualized as a greyscale image, where lighter values represent larger depths, and a result from applying our proposed segmentation algorithm on this image.

We use graph cuts to find a (near) optimal labelling, in an attempt to simultaneously remain close to the observed normal vectors and enforce smooth label transition between neighbouring pixels. A key contribution is our way of incorporating jump edges in the pixel neighbourhood structure. Finally we segment, by essentially searching for connected components in the labelling.

Brief background on graph cut optimization is provided in section II, followed by a description of our method in section III. Results obtained by our algorithm, also evaluated using the framework mentioned above, are given in section IV and we conclude in section V.

II. Graph cut optimization

Graph cuts as a technique for optimization originated as a method for finding optimal binary segmentations of images, and has since been extended for multi-label optimization in a wide variety of computer vision problems ranging from image restoration to stereo correspondence. We chose to utilize graph cuts in our range image segmentation method. Our work is based on the theory and implementation developed by Veksler, Boykov, Zabih and Kolmogorov [8], [9], [10].

A. The labelling problem

Many computer vision problems, including image restoration, stereo correspondence, segmentation and multiview reconstruction, can be stated as labelling problems. For a set of sites (usually pixels) $\mathcal{P} = \{1, 2, \ldots, n\}$ and a set of labels $\mathcal{L} = \{\ell_1, \ldots, \ell_k\}$, the labelling problem is that of assigning a label from $\mathcal{L}$ to every site in the set $\mathcal{P}$. This mapping from $\mathcal{P}$ to $\mathcal{L}$ is called a labelling, or configuration, and is denoted by $f = \{f_1, \ldots, f_n\}$ where $f_p \in \mathcal{L}$ for every $p \in \mathcal{P}$. The set $\mathcal{F}$ consists of all k^n possible configurations.

An energy function $E(f)$ can be used to evaluate a particular labelling f, such that the problem of finding some optimal labelling then becomes one of minimizing the energy function over $\mathcal{F}$.

B. Energy functions

Energy functions of the form

$$E(f) = E_{\text{data}}(f) + \lambda \cdot E_{\text{prior}}(f) \tag{1}$$

are popular for computer vision problems. The terms encode information about how closely a configuration matches the observed data, and how closely it matches prior assumptions about the dependency between sites. The constant λ allows for the adjustment of relative importance carried by each term.

The minimization of E requires a global optimization which, as it stands, is a significant problem for such a general formula. Moreover, due to the nature of $\mathcal{P}$ and $\mathcal{L}$, the search space is typically extremely large and high dimensional. While using a more specific function limits its possible uses, it may enable the use of efficient methods for finding a global minimum (or "good" local minimum).

C. Minimization by graph cuts

Graph cut optimization involves setting up a graph structure whose minimum cut (a subset of the edges, with minimum total edge weight, that if removed splits the graph into two disjoint components) coincides with the global minimum of a specific energy function [11]. The real strength of this approach lies in the fact that the minimum cut problem can be solved fast and efficiently, by solving a dual maximum flow problem, hence the minimization of the energy function becomes practically feasible.

A single cut optimizes a binary labelling. When $\mathcal{L}$ consists of more than two labels it becomes necessary to perform multiple graph cuts iteratively, traversing the solution space in some way, and global optima are no longer guaranteed. However the work of Veksler [8] assures that provably good local optima, with respect to large label swap or expansion moves, are attainable with a relatively small number of iterations.

The graph cuts framework we are using considers energy functions of the form

$$E(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \sum_{\{p,q\} \in \mathcal{N}} V_{\{p,q\}}(f_p, f_q). \tag{2}$$

We assume here that λ is included in the second term. $D_p(f_p)$ determines the data cost for how much the assigning of label f_p to pixel p disagrees with the observed data. The prior term assumes that the structure of the scene has some uniformity or smoothness so that $V_{\{p,q\}}(f_p, f_q)$ is now a smoothness cost determined for pairs of neighbouring pixels, as defined by the neighbourhood system $\mathcal{N}$. If the smoothness cost is a semi-metric function, α-β swap moves can be performed in attempting to optimize the energy function. If the smoothness cost is strictly metric, the more efficient α-expansion move, with better optimality properties, can be used [8].

III. Our method

The essence of our range image segmentation method consists of a multi-label optimization by graph cuts. The energy function E is set up specifically such that its minimum is attained at a configuration that will segment the image into planar surfaces. The set of labels will be a discretization of possible surface normals. For each pixel in the range image we calculate a normal vector from the depth data, in order to define the data term in E. With the assumption that neighbouring pixels are likely to be on the same planar surface, and the fact that points on a plane have the same normal vector, we can set up the smoothness term accordingly.

The energy function thus specified enables us to segment the range image by surface normals, making our algorithm theoretically capable of handling all crease edges as well as jump edges between surfaces with different orientations. In order to also include jump edges between surfaces with approximately the same orientation, but at different depths, we modify the neighbourhood structure $\mathcal{N}$ that will be used in the graph cut optimization.

We proceed with a more detailed explanation of the various components of our method.

A. Calculating surface normals from the data

We calculate the local surface normal at each pixel of the range image in a simple way. A pixel's 4-connected neighbours are used to create two vectors, and their normalized cross product gives a normal vector associated with that pixel. If necessary we flip the vector to ensure that all normal vectors have the same sign along the depth axis (i.e. they all point towards, or they all point away from, the sensor). Note that here we need to use metric 3D coordinates associated with the pixels which should be available under the assumption that the range sensor is properly calibrated.

While more complex and noise resistant methods can be used, such as fitting least squares planes over larger neighbourhoods, we found that it is not necessary. The graph cut optimization step should be able to handle noise in the data quite well and, in fact, fitting normal vectors to larger neighbourhoods may lead to excessive and unwanted smoothing particularly around edges in the data.

B. Discretizing normal vectors and defining labels

In our case the labelling problem becomes one of assigning a normal vector to every pixel. This idea presents two problems within the framework of graph cuts. Firstly, the continuous space of possible normal vectors needs to be discretized so that it can be mapped to a finite number of labels. Secondly, every label should be associated with a single index so that we can write the label set as $\{\ell_1, \ldots, \ell_k\}$, however the space of possible normal vectors, in our case points on the unit semi-sphere, is described by two parameters (e.g. elevation and azimuth angles).

The first of these problems is dealt with by distributing k points on the surface of the semi-sphere. There are many different existing ways in which this distribution can be performed [12]. For the purposes of illustration we opt for a simple distribution that is regular in the elevation-azimuth parameter space. Fig. 2 illustrates. Note that our core method permits any discretization of the normal vector space (some may be more sensible than others) and not specifically the one mentioned and used here.

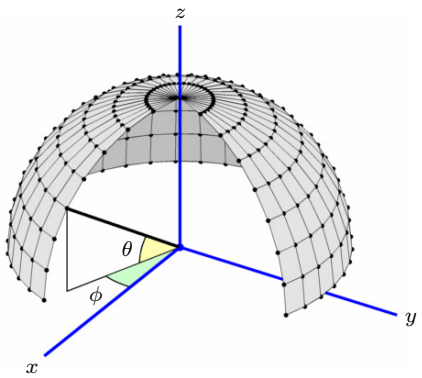


Fig. 2. A possible distribution of points on the semi-sphere is to space them regularly in the elevation (θ) and azimuth (ϕ) domain. Normal vectors, which will serve as labels, are described by (θ, ϕ) pairs.

It should also be noted that some normal vectors may be more likely to occur in range data than others. Our elevation-azimuth discretization produces vectors more closely spaced at $\theta \approx \pi/2$, which may be desirable. We also realize that surface normals perpendicular to the principal axis of the range sensor will be difficult, if not impossible, to measure and we may therefore omit normal vectors for which $\theta = 0$ from further consideration.

It remains to define an ordering of the discrete set of normal vectors. In our implementation we decided to resolve ϕ, which can be an element of $[0, 2\pi)$, into 32 equally spaced nodes and θ, which can be in $[0, \pi/2]$, into 8. At $\theta = \pi/2$ we need only one vector (not 32) so that we end up with $8 \times 32 - 31 = 225$ normal vectors. We now simply map these 225 vectors to the integers $1, 2, \ldots, 225$, in some specific order, to arrive at our label set. It should be stressed that such a mapping does not necessarily imply that proximity in the labels will mean that those normal vectors are close to one another.

C. Setting up an energy function

Next we define an energy function, which will be of the form given in equation 2, such that its minimum will coincide with our goal of segmenting the range image into planar surface regions. In order to set up the function we must specify a data term and a smoothness term, consider the relative importance between these two terms, and also build a neighbourhood structure over the pixels.

1) Data term: The data term in the energy function determines penalties for assigning certain labels to pixels. In our case it must encapsulate, for every pixel p, the difference between the proposed label f_p, and the normal vector calculated directly from the data. We define $D_p(f_p)$, i.e. the cost of assigning label f_p to pixel p, to be the positive angle between the normal vector associated with label f_p and the one given by the data. We represent this angle in degrees, but we may just as well have chosen radians. Any scale change in either the data term or the smoothness term can always be compensated for by scaling the λ parameter accordingly.

2) Smoothness term: We have defined our labels to correspond to surface normals and, if we limit our input data to range images containing only planar or approximately planar surfaces, it is reasonable to assume that neighbouring pixels are likely to have the same normal vector and therefore the same label. Furthermore we were forced to map a 2-parameter set of normal vectors to a 1-dimensional set of indices which makes it hard to define a metric function that varies smoothly over those indices. In light of these arguments we opt for a piecewise constant prior,

$$V_{\{p,q\}} = \lambda \cdot u_{\{p,q\}} \cdot \delta(f_p \neq f_q), \tag{3}$$

where $u_{\{p,q\}}$ is the edge weight between two neighbouring pixels (dealt with in the next section) and

$$\delta(f_p \neq f_q) = \begin{cases} 1, & \text{if } f_p \neq f_q, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

This smoothness function will assign a penalty to neighbouring pixels which have different labels, and that penalty is constant for any two different labels, which amounts to a prior assumption that the scene contains significant regions over which the labelling is constant (i.e. planar surfaces). It is important to note also that this prior is invariant with regards to the specific ordering of labels, which is preferable because the assigning of 1-dimensional indices to the 2-parameter set of normal vectors will always be somewhat arbitrary.

Finally we note that our smoothness cost is a metric function, allowing us to perform α-expansion moves [8] iteratively in our graph cuts optimization of the energy function.

3) The value of λ: An ideal value for λ, the parameter that adjusts relative importance between the data term and smoothness term, is related to how well the normal vectors obtained purely from the data fit our prior assumptions about smoothness. Factors to take into consideration when selecting λ include the level of detail in the scene, i.e. the amount and size of the scene's planar constituents, as well as the severity of noise to be expected in the captured data. In our experiments λ will be the only parameter that we change, and we will investigate its effect on the results obtained.

D. Defining a neighbourhood structure

The only issue remaining in our graph cuts formulation is that of defining a neighbourhood structure over the pixels in the range image, meaning that we need to specify which pairs of pixels $\{p, q\}$ belong to $\mathcal{N}$.

We can choose $\mathcal{N}$ to be the commonly used 4-connected neighbourhood structure, where every (interior) pixel is connected to the pixels immediately above and below, to the left and to the right of it (pixels on the boundary will obviously have less than 4 neighbours). Here all edges carry equal weight, meaning that we assign a constant value to each $u_{p,q}$. This structure may already be sufficient and, coupled with our smoothness term, is capable of capturing our assumption about labels likely to be the same for pixels in close proximity to one another.

The simple 4-connected neighbourhood structure can, in theory, correctly handle crease edges where the change in surface orientation is large enough for the smoothness term to lose its dominance over the data term. The same applies for jump edges between surfaces with different normals. However, situations may arise where a jump edge occurs between two parallel (or nearly parallel) surfaces separated only in depth. Those surfaces have the same (or approximately the same) normal vector and our optimization procedure will combine them into one region unknowingly, thereby failing to detect the jump edge.

Luckily it turns out that jump edges are quite easy to detect in range images. All the significant discontinuities in a range image represent valid jump edges, and we found that a simple edge detection technique, such as convolution with the Sobel masks followed by morphological thinning and bridging, has the ability to deliver crystal clear jump edges (it may be necessary to suppress noise first with an edge preserving median filter [13]).

We now simply remove from $\mathcal{N}$ all the connections to every jump edge pixel found (or, equivalently, we set $u_{p,q} = 0$ for any identified jump edge pixel p or q). Note that this will have the effect of also breaking connections between pixels on either side of a jump edge, thus no longer enforcing smoothness across that edge. Every jump edge pixel itself will subsequently be labelled with the normal vector closest to the one calculated from the data because such a pixel will not have any effect on the smoothness term.

We have now defined a set of labels, an energy function and a neighbourhood system and have defined everything necessary to run the graph cut optimization procedure. It gives as output an optimal labelling of the pixels, which we can convert into a segmentation of the range image.

E. Relabelling connected components

The graph cut optimization process produces a labelled image that should, depending on the scene and the data, contain large regions of identically labelled pixels. This is almost a segmentation, except that disconnected regions coinciding with surfaces that happen to be (near) parallel may have the same labels. We therefore perform a connected component labelling, on the labels returned by the optimization step, to arrive at a segmentation in which every segment is a region of connected pixels with the same label.

This step is necessary because our graph cut optimization is not actually a segmentation method, but more a method of restoration that tries to find original normal vectors before they were "corrupted" by sensor noise. However since crease edges should appear in this restoration, as well as jump edges due to our handling of the neighbourhood structure, it is natural to assume that a sensible segmentation can be obtained from a good labelling found by our graph cut optimization.

F. Merging thin regions

As a final step we may identify regions likely to be incorrect and, in an effort to "clean up" our segmentation, merge them with surrounding regions.

We note that incorrectly segmented regions are typically thin and line-like, occurring along jump edges or creases edges. We identify regions with these shape properties by calculating the ratio between the total number of pixels contained in a region (the area) and the number of pixels on the perimeter of the region (the circumference). If this ratio is below some threshold we iteratively erode that region, by changing the labels of eroded pixels to those of surrounding regions, until the region disappears. In the experiments that follow we chose a threshold ratio of 2.

IV. Results

We ran our segmentation algorithm on the *Perceptron-Test* dataset created by Hoover et al. [2] and then used their comparison tool to evaluate our results against the ground truth data they provide. The dataset consists of 30 range images

obtained using a Perceptron laser range finder. Every image is a 512×512 array of 12-bit depth data, and contains a scene of "up to five polyhedral objects placed in a variety of poses and with varying degrees of inter-object spacing" [2], which means that the surfaces in the scene are all planar.

Fig. 3 shows some results of our graph cut optimization step, applied to one of the images from the dataset (the one shown in Fig. 1 of this paper) for different choices of λ. These are results before connected component relabelling and region merging, so that different shades of grey represent different labels. It is clear that very large values of λ lead to under-segmentation (or over-smoothing) as too much emphasis is given to the smoothness prior, while very small λ values may lead to over-segmentation. In the case of $\lambda = 0$ there is absolutely no smoothing, and the result is identical to the labels found when calculating normals directly from the data which is noticeably noisy. We see that graph cut optimization can successfully deal with this noise. It seems that for this image $\lambda = 75$ might be close to an optimal choice (that result was subjected to connected component relabelling and region merging to create the segmentation in Fig. 1(b)).

In setting up a neighbourhood structure for graph cuts we decided to break connections to detected jump edges, thereby removing the smoothness constraints on labels of pixels on either side of such an edge. This breaking has another important benefit: it tends to produce better (less noisy) edges between regions separated by jump edges. Some examples are shown in Fig. 4 where we compare segmentations obtained with and without jump edge disconnection (JED).

We evaluated our segmentations of the dataset images using the comparison tool of Hoover et al. [2]. It compares a submitted machine segmentation with a ground truth segmentation, and classifies regions as one of the following: *correct*, *over-segmented*, *under-segmented*, *missing* and *noise*. Regions that

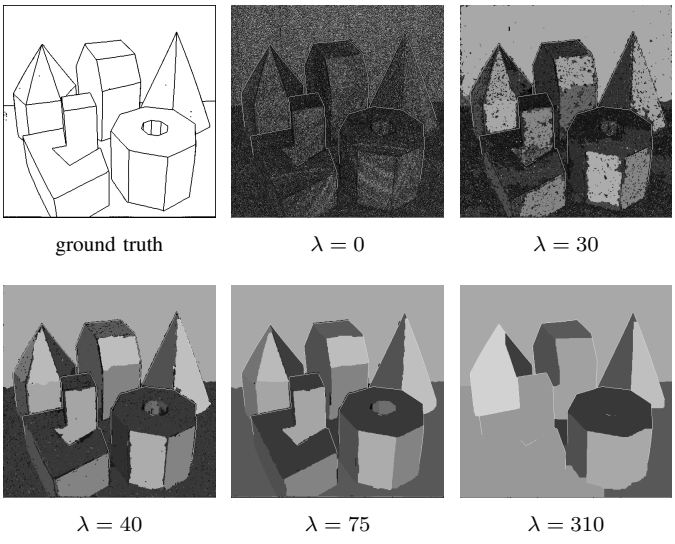
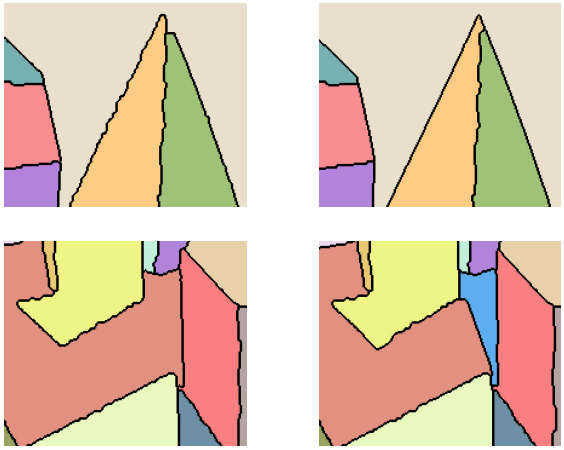


Fig. 3. A comparison of output results from our graph cut optimization procedure (before connected component relabelling and region merging), for a few choices of λ. Detected jump edges are visible as white lines.



(a) without JED (b) with JED

Fig. 4. Close-ups of a segmentation we obtain without jump edge disconnection (JED), compared to the segmentation we obtain with JED. For the latter the segmentation is less noisy around detected jump edges and parallel surfaces at different depths are separated successfully.

are *missing* are those that appear in the ground truth but not in the machine segmentation. Conversely, *noise* regions are regions that are found only in the machine segmentation and not in the ground truth. A tolerance threshold T, chosen between 50% and 100% where 100% is the strictest, is used to determine region classification.

Fig. 5 shows the number of correct, over-segmented, under-segmented, missed and noisy regions in our segmentations, for various values of λ (the one parameter in our method that needs to be specified), averaged over the entire dataset. A rather strict threshold value of $T = 80\%$ was used, because it is the value in [2] for which more results from other methods are available. Note also that results are shown for $\lambda \geq 40$, due to the fact that the comparison tool allows segmentations containing at most 256 regions and smaller λ values will typically produce more regions than that.

We note that the accuracy of our algorithm is quite strongly related to the value of λ but, fortunately, there seems to be a large range of λ values for which the algorithm performs well. This implies that the method is quite stable, and not extremely sensitive to a particular choice for λ.

In Fig. 6 we show the average number of regions correctly identified, when tweaking T, for a fixed value of $\lambda = 45$ (which, from Fig. 5, appears to be a possibly optimal choice). The figure compares our results, indicated by GC (for "graph cuts"), with those obtained by the four algorithms evaluated by Hoover et al. in [2].

Overall our method compares well with the others, and even slightly outperforms all of them. However we should be careful of reading too much into these results. Output from the comparison tool can be rather misleading since quantitative accuracy does not necessarily agree with a qualitative evaluation of segmentations obtained. For example, the tool considers regions of any size to be equally important.

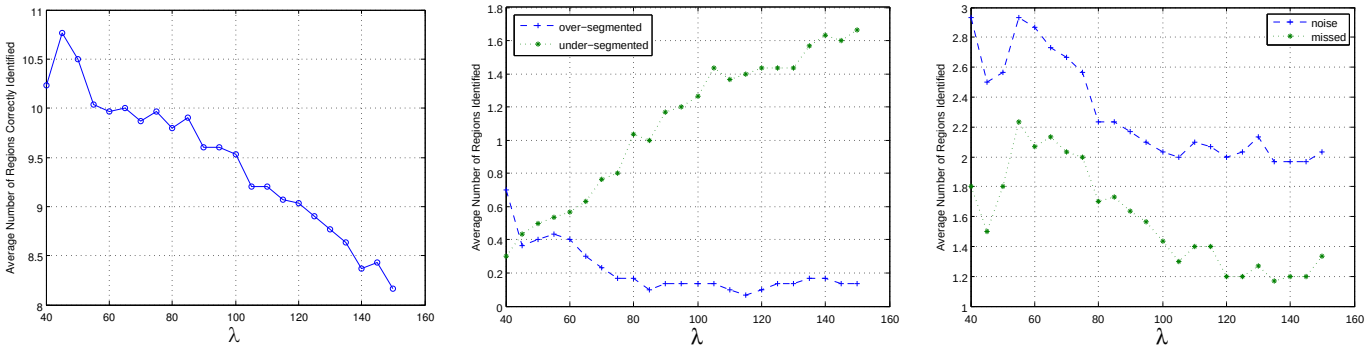


Fig. 5. Results from the comparison tool of Hoover et al. [2], indicating how the various performance metrics change for our method using different choices of λ (the parameter that controls the relative interplay between data and smoothness in the graph cut optimization). The tool's threshold was set to 80%.

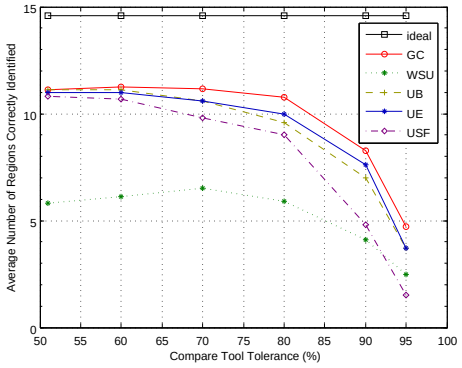


Fig. 6. A comparison of our algorithm with $\lambda = 45$ (GC) with the four methods investigated in [2], for different threshold settings used in the comparison tool.

V. Conclusions and Future Work

We proposed a method for segmenting range images into planar regions, which performs graph cuts in order to find an optimal configuration of normal vectors at all pixels. We introduced jump edge disconnection, and found that it can produce cleaner edges between segments and separate adjacent planar surfaces at different depths.

Our current implementation requires on average about 3 minutes to segment a 512×512 image. While that could be considered quite fast, considering the size of the search space (there are $k^{512\times512}$ possible labellings where k is the number of labels) over which the energy function needs to be minimized, for most applications it would be much too slow. The number of labels we consider is quite high (in this paper we chose $k = 225$), making it a significant reason for the slow execution, yet it still yields a fairly sparse distribution of points on the sphere (recall Fig. 2). Possible future work includes an investigation into other ways of placing points on the surface of the sphere.

The concept of optimizing energy functions by graph cuts is still quite actively researched, and in the near future we hope to consider the incorporation of label costs [14]. This extra term in the energy function penalizes segmentations according to number of regions — an idea already found by Han et al. [4] to have potential in a simulated annealing framework.

Lastly, an important consideration for future work would be to investigate the performance of our method on range images containing non-planar surfaces, and to expand on our ideas to better handle such cases.

References

[1] D. Martin, C. Fowlkes, D. Tal, J. Malik, *A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics*, IEEE International Conference on Computer Vision, 2:416–423, 2001.

[2] A. Hoover, G. Jean-Baptiste, X. Jiang, P. Flynn, H. Bunke, D. Goldof, K. Bowyer, D. Eggert, A. Fitzgibbon, R. Fisher, *An experimental comparison of range image segmentation algorithms*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 18(7):673–689, 1996.

[3] R. Hoffman, A. Jain, *Segmentation and Classification of Range Images*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 9(5):608–620, 1987.

[4] F. Han, Z. Tu, S.-C. Zhu, *Range image segmentation by an effective jump-diffusion method*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(9), 1138–1153, 2004.

[5] X. Jiang, K. Bowyer, Y. Morioka, S. Hiura, K. Sato, S. Inokuchi, M. Bock, C. Guerra, R. Loke, J. du Buf, *Some further results of experimental comparison of range image segmentation algorithms*, International Conference on Pattern Recognition, 4877–4882, 2000.

[6] M. Powell, *Comparing curved-surface range image segmentors*, MSc thesis, University of South Florida, 1997.

[7] J. Min, M. Powell, K. Bowyer, *Progress in automated evaluation of curved surface range image segmentation*, International Conference on Pattern Recognition, 1640–1644, 2000.

[8] O. Veksler, *Efficient graph-based energy minimization methods in computer vision*, PhD dissertation, Cornell University, 1999.

[9] Y. Boykov, O. Veksler, R. Zabih, *Efficient approximate energy minimization via graph cuts*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(12), 1222–1239, 2001.

[10] Y. Boykov, V. Kolmogorov, *An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(9):1124–1137, 2004.

[11] V. Kolmogorov, R. Zabih, *What energy functions can be minimized via graph cuts?*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 26(2):147–159, 2004.

[12] E. Saff, A. Kuijlaars, *Distributing many points on a sphere*, The Mathematical Intelligencer, 19(1):5–11, 1997.

[13] V. Koivunen, M. Pietikäinen, *Experiments with combined edge and region-based range image segmentation*, Theory and Applications of Image Analysis: 7th Scandinavian Conference on Image Analysis, 162–176, 1991.

[14] A. Delong, A. Osokin, H. Isack, Y. Boykov, *Fast approximate energy minimization with label costs*, IEEE Conference on Computer Vision and Pattern Recognition, 2173–2180, 2010.