Max-flow Min-cut Graph Based Image Segmentation

Abhishek Damle
Electical Engineering
Virginia Tech
Blacksburg, Virginia 24060
Email: adamle@vt.edu

Cody Crofford
Computer Engineering
Virginia Tech
Blacksburg, Virginia 24060
Email: ccody7@vt.edu

***Abstract*: Image Segmentation is a process used for dividing images into different regions to allow for simplification of image analysis techniques. We formulate image segmentation as a max-flow, min-cut problem and develop a graph cut based method of interactive image segmentation that segments the foreground and background from images. The performance of our method is evaluated using a subset of images from the Berkeley Segmentation Dataset and we track the performance of our method with respect to the number of prelabelled pixels. We find that the performance of our method is dependent on the probability and spatial distributions of the pixels in the foreground and background of an image and suggest ways to further improve the performance of our method.**

Index terms: Segmentation, Graph, minimum-cut, maxflow

I. Introduction

Image Segmentation is a process of dividing an image into different regions based on the characteristics of the pixels in the image[1]. It can be used to partition an image's background and foreground or to determine object boundaries within an image. This simplification of images, through segmentation, allows the analysis of areas of interest for applications such as medical imaging diagnosis or object detection for self-driving vehicles.

Interactive segmentation is the process of partitioning images where some pixels and the segments they belong to, have been identified beforehand. These pixels are then segmented into their prelabelled segments and act as hard constraints that allow a user to provide clues on what they want to segment [2]. This characteristic makes interactive segmentation ideal for applications where human guidance is available or preferred to help segment an image such as photo editing and medical image segmentation.

Separating Images by contextual regions such as background and foreground can be performed with clustering or Graph-based segmentation approaches. Segmentation by clustering is done by clustering pixels based on selected pixel features such as intensity, color, and texture. While this method is simple, it has the drawback of being subjective to initial settings for cluster centers, and being highly sensitive to outliers.

Graph based segmentation methods are executed by converting the pixels in an image to fully connected graphs that perform segmentation through finding the maximum-flow minimum-cut of the graph. Although these methods are computationally complex, they have the advantage of not requiring training.

II. Problem Formulation

To formulate image segmentation as an optimization problem, we consider a graph representation, *G = (V,E),* of an image. In the graph representation, the vertices, *V*, represent the pixels while the edges, *E*, represent neighboring pairs of pixels. Each pixel *i* in the graph also has a probability a_i and b_i that it belongs to the foreground or background, respectively. Next, we introduce a separation penalty, p_{ij}, that makes the segment boundaries smoother by penalizing neighboring pixels *i* and *j* for being placed in different segments. Considering a_i, b_i, and p_{ij}, the image segmentation problem can be formulated as finding a partition *(A, B)* that splits the pixels into foreground and background sets respectively, such that the objective function, $q(A,B)$, as shown below, is maximized[3].

$$q(A,B) = \sum_{i\in A} a_i + \sum_{j\in B} b_j - \sum_{\substack{(i,j)\in E \\ |A\cap\{i,j\}|=1}} p_{ij} \qquad (1)$$

A. Formulation as a Minimum-cut Problem

The image segmentation problem can be equated to a minimum-cut problem. The original objective function can be transformed from a maximization function to a minimization function, as shown below.

$$q(A,B) = \sum_{i\in A} a_i + \sum_{j\in B} b_j - \sum_{\substack{(i,j)\in E \\ |A\cap\{i,j\}|=1}} p_{ij} \qquad (2)$$

$$q(A,B) = \sum_i a_i + b_i - \sum_{i\in A} b_i - \sum_{j\in B} a_j - \sum_{\substack{(i,j)\in E \\ |A\cap\{i,j\}|=1}} p_{ij} \qquad (3)$$

thus maximizing $q(A,B)$is equivalent to minimizing $q'(A,B)$,

$$q'(A,B) = \sum_{i\in A} b_i + \sum_{j\in B} a_j + \sum_{\substack{(i,j)\in E \\ |A\cap\{i,j\}|=1}} p_{ij} \qquad (4)$$

A source node (s) and a sink node (t) can then be added to the original graph to represent the foreground and background regions of the image, respectively. All other nodes in the graph are connected to the source and sink. a_i and b_i, then represent the capacities of the edges between the nodes and source and sink respectively. Lastly, all undirected edges between pairs of nodes are replaced with two directed edges with opposite directions.

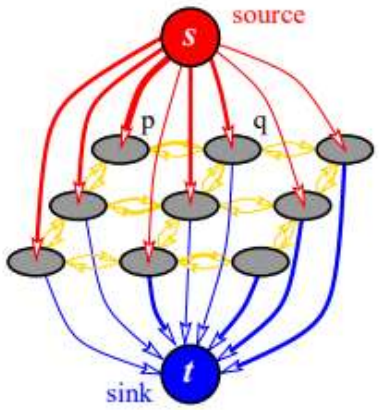

Fig. 1. Resulting directed graph with addition of source (s) and sink (t) nodes. [9]

The capacity of the s-t cut, $c(A, B)$, is defined below and consists of three terms. The *K* and *L* terms account for cuts to edges connecting the source and sink nodes respectively, to other nodes. Meanwhile, the M term accounts for cuts to edges that connect neighboring pixels.

$$c(A,B) = K + L + M,$$
$$K = \sum_{i \in A} b_i \, , L = \sum_{j \in B} a_j \, , \; M = \sum_{\substack{(i,j) \in E \\ |A \cap \{i,j\}|=1}} p_{ij} \qquad (5)$$

Since *c(A, B)* is equivalent to *q'(A, B)*, the image segmentation problem can be solved through minimizing *c(A, B)* by finding the minimum-cut that separates the source and sink.

III. Methods

Since image segmentation has been posed as a minimum-cut problem, it can be solved through well-established optimization algorithms utilizing the max-flow min-cut theorem. In order to use this method, the image is first preprocessed and converted to a graph.

A. Preprocessing

The first step in converting the image to a graph was transforming it from RGB to grayscale. This step, while not strictly necessary, helped simplify the conversion to a graph. The cvtColor() function from the OpenCV python library maps red, green, and blue pixel values to a single intensity value and was used to transform the input RGB image to grayscale. The equation used for the mapping is shown below [4].

$$I = 0.299 \times R + \; 0.587 \times G + 0.114 \times B \qquad (5)$$

In the preprocessing stage, the foreground and background pixels were also manually selected. This was done iteratively in three stages with increasing coverage so that the performance of our method could be tracked with respect to the number of pixels manually prelabelled. The ranges of coverages of the manual selections as percents of the entire image are shown in table 1.

TABLE I. Coverage of the Manual Selections as Percent of the Entire Image

Amount of Prelabelled Pixels	Range of Coverage
Minimal	.36% - .74%
Moderate	1.70% - 3.61%
High	17.50% - 39.68%

B. Source and Sing Edge Capacities

The capacities of the edges between the nodes and the source and sink represent the probability that the node belongs in the foreground or background segment. These probabilities were estimated from probability density functions based on the labeled background and foreground nodes. As suggested by [3], the capacities of the edges were calculated using equations 6 and 7, as shown below, where f_A and f_B are the gaussian distributions generated from the labeled foreground and background nodes.

$$a_i = -\ln\left(f_B(I_i)\right) \qquad (6)$$
$$b_i = -\ln\left(f_A(I_i)\right) \qquad (7)$$

The labeled background, and foreground nodes were assigned maximum sink, and source edge capacities to force the graph cut algorithm to categorize the prelabelled pixels in the correct segment. These maximum capacity values were calculated based on the set of a_i and b_i for all the pixels in the image.

C. Neighboring Node Edge Capacities

The edge capacity of neighboring nodes is related to the similarity of their intensity values. The equation of the edge capacity of neighboring nodes is given by equation (8) where the σ scales the penalization term $(I_i - I_j)$.

$$p_{ij} = \exp\left(-\frac{(I_i - I_j)^2}{2\sigma^2}\right) \qquad (8)$$

Using the resulting graph obtained from the image, and the capacities for our nodes and edges, we calculated the minimum cut of the directed graph.

D. Finding the Minimum Cut

The NetworkX python library was used to find the maximum flow of the image graph. The NetworkX library provides several algorithms for finding the maximum flow of a graph including Edmonds Karp, Preflow Push, and Shortest Augmenting Path [6]. The time complexity of these algorithms, given the number of pixels, p, in a graph representation of an image is given in table 2.

TABLE II. Time Complexity of Maximum Flow Algorithms from the NetworkX Library.

Algorithm	Time Complexity
Edmonds Karp	$O(p^3)$
Preflow Push	$O(p^{2.5})$
Shortest Augmenting Path	$O(p^3)$

The Preflow Push algorithm for finding the maximum flow was used in this work since it has the smallest time complexity given the number of pixels in an image.

Lastly, the nodes in the graph were partitioned by cutting the saturated edges from the residual graph of the maximum flow graph [7].

IV. Performance Evaluation

Once the image was segmented, the performance of the proposed method was evaluated by comparing the method's output with the ground truth for a set of images. The intersection over union (IoU) and F1 score metrics were used to measure the performance of the proposed method.

The IoU is a measure of overlap between the predicted segmentation and the ground truth and is calculated using equation (9) where A is the set of pixels in the predicted segment and B is the set of pixels in the ground truth[5]. Equation (9) also restates IoU in terms of true positives (TP), false positives (FP), and false negatives (FN).

$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{TP}{TP+FP+FN} \qquad (9)$$

One key feature of the IoU metric is that the size of a segment has no effect on its IoU score. This made the IoU metric ideal since it provides a way to measure the performance of the proposed method on a diverse set of images where the foreground and background vary in size.

F1 score is another common performance metric used to measure segmentation performance. F1 score in terms of true positives (TP), false positives (FP), and false negatives (FN) as well as precision (P) and recall (R) is given below.

$$F1 = 2 * \frac{P \cdot R}{P+R} = \frac{TP}{TP+.5 \cdot (FP+F\ )} \qquad (10)$$

Although the F1 score is very similar to IoU, it was used in this work primarily to compare the performance of our method to other works.

V. Dataset

A set of images from the Berkeley Segmentation dataset were used to test and evaluate our method. This dataset contains both grayscale and color images with accompanying hand label segmentation labels. The dataset also has accompanying benchmarks for different image segmentation methods that could be used for comparison with our minimum-cut graph segmentation method [8].

The Berkeley Segmentation contains images with a varying number of segments and the four images with two segments were all used to evaluate the graph-based segmentation method. To increase the number of images used for evaluating our segmentation method, the segments from the ground truth of 12 additional images from the Berkeley Segmentation dataset were reduced through combining them together. The set of images used to measure the performance of our segmentation method are shown in figure 2.

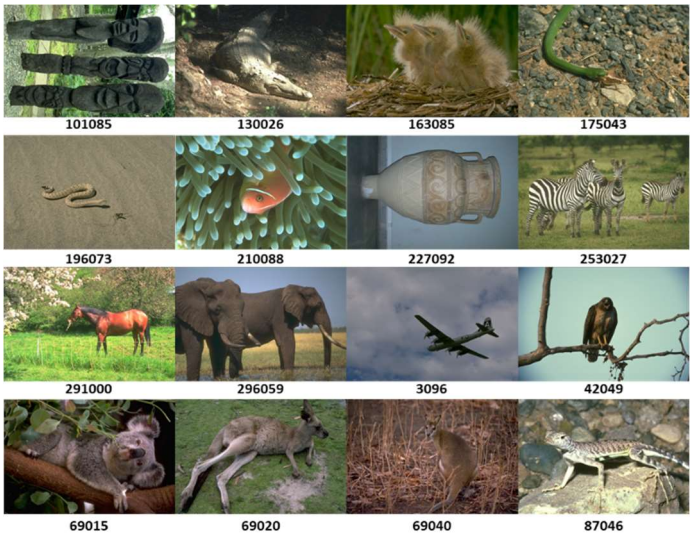

Fig. 2. Subset of Images from the Berkeley Segmentation Dataset Used to Evaluate Performance.

VI. Results

The F1 scores of our method for various images from the Berkeley Segmentation dataset with minimal, moderate, and high numbers of prelabelled pixels are shown in figure 3. Figure 3 also contains the best F1 scores of other works for the images used to evaluate our method. Figure 4 contains similar information to figure 3 and uses IoU scores to show the performance of our method.

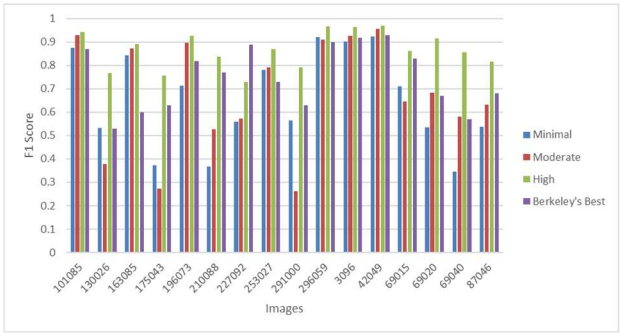

Fig. 3. F1 Scores of the Max-flow Min-cut Graph Based Image Segmentation for various images from the Berkeley Image Segmentation dataset.

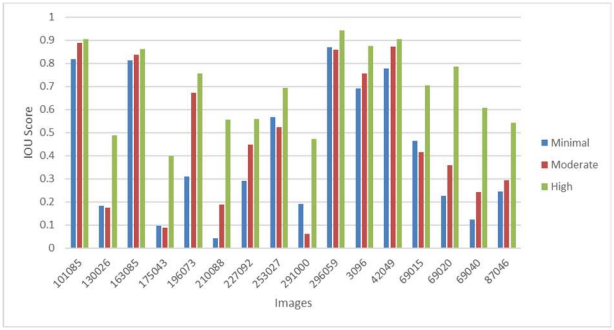

Fig. 4. IoU Scores of the Max-flow Min-cut Graph Based Image Segmentation for various images from the Berkeley Image Segmentation dataset.

Amongst the 16 images used to measure the performance of our segmentation method, images 296059 and 42049 had the highest average IoU, and F1 scores. Meanwhile, images 175043, and 291000 had the lowest average IoU, and F1 scores.

Adding more prelabelled pixels generally increased the performance of our segmentation method. However, the performance of our segmentation method was lower for moderate than minimal numbers of prelabelled pixels in the case of certain images. These include images 291000, 69015, 175043, and 130026. Conversely images 101085, 163085, 296059, 3096, and 42049, saw a very small change in segmentation performance based on the number of hand labelled pixels.

VII. Discussion

A. *Performance Differences with Respect to Images*

Our segmentation method had a large variance in its performance based on the image it attempted to segment. This was largely due to how well the image fit our assumptions for estimating the probability that a pixel belongs to the foreground or background in the process of calculating the source and sink edge capacities of the graph representation of the image. Namely that the distribution of the foreground and background are normal and distinct from one another.

Our segmentation method was able to segment image 296059 with a high level of accuracy. The actual distributions of the pixel values in the ground truth foreground and background segments are shown in figure 4 as histograms. Additionally, figure 4 also has Gaussian distributions fit over the foreground and background distributions. Assuming that the hand labelled pixels have the same distribution as the segments they belong to, figure 5 shows the probability density functions used to calculate the source and sink edge capacities in an ideal case.

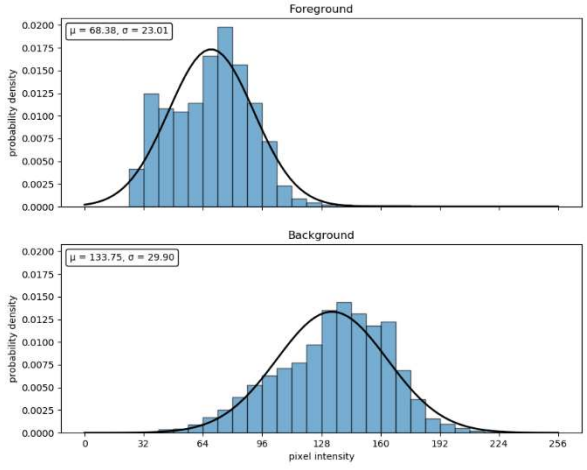

Fig. 5. Distribution of Foreground and Background Pixels in Image 296059

Based on figure 5, the foreground and background pixels of image 296059 are distributed normally and have distinct Gaussian distributions. The probabilities that pixels belong to the foreground, and background based on the Gaussian distributions from figure 5 are shown as heat maps in figure 6. The ground truth segmentation is also shown as a background in figure 6 to help discern the location of pixels.

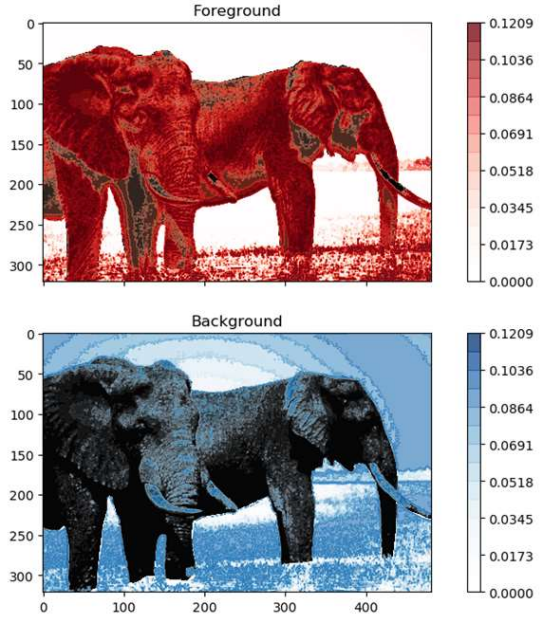

Fig. 6. Heatmaps of the Probability that Pixels Belong in the Foreground and Background for Image 296059

Figure 6 confirms that, given the hand labelled pixels accurately approximate the distributions of the segments, the sink, and source capacities of the graph of image 296059 will have different values for each node. This will allow our segmentation method to accurately segment image 296059 as confirmed by our results shown in figure 3 and 4.

Our segmentation method had difficulties segmenting image 175043 with a high level of accuracy. The distribution of pixels in the foreground and background segments along with the estimated normal distribution of segments is shown in figure 7. The heat map of image 175043 based on estimated probability density functions from figure 7 is shown in figure 8.

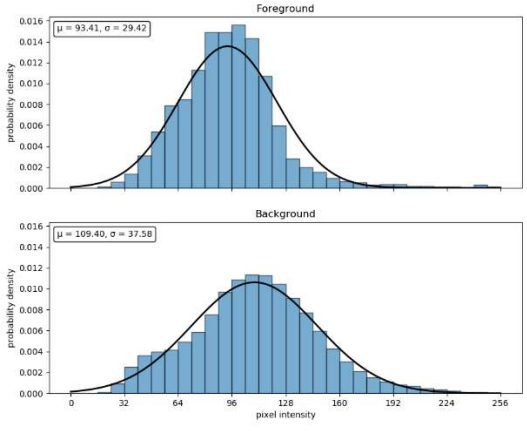

Fig. 7. Distribution of Foreground and Background Pixels in Image 175043

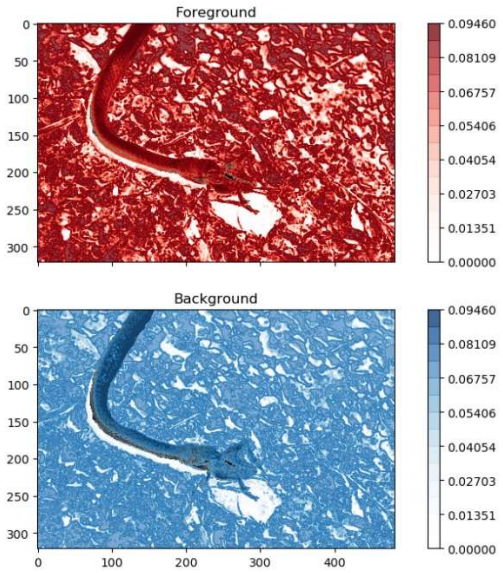

Fig. 8. Heatmaps of the Probability that Pixels Belong in the Foreground and Background for Image 175043

Based on figure 7, even though a Gaussian distribution matches the actual distribution of the pixels closely, the estimated foreground and background distributions are not distinct from one another. As shown in figure 8, this results in pixels having similar probabilities of belonging in the foreground, and background. Thus our segmentation method has poor performance for image 175043 and similar images where our assumption that the foreground and background segments have distinct distributions is incorrect.

Our segmentation method also has low performance for images where the distribution of the foreground and background is not gaussian as is the case for image 130026. The distributions of the foreground and background of image 130026 are shown in figure 9. Since neither the foreground nor the background distributions are normal, the standard deviations of the gaussian distribution estimates are relatively high. This caused the pixels to have similar low probabilities of belonging in the foreground, and background which leads to low segmentation performance.

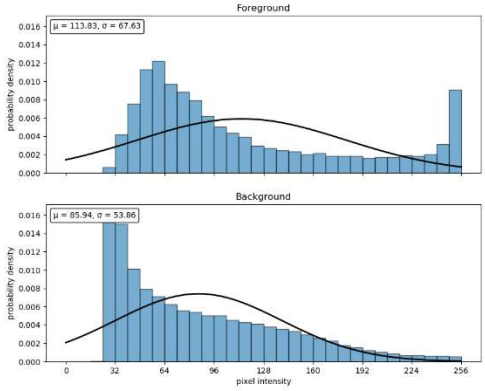

Fig. 9. Distribution of Foreground and Background Pixels in Image 175043

B. *Performance Difference between Minimally, Moderately, and Highly Prelabelled Images*

Our segmentation method, when provided with a high number of prelabelled pixels, produced the best output since the probability that the distribution of the prelabelled pixels matches the actual distributions of the foreground and background increases with the number of prelabelled pixels.

However, with an increase from a minimal number of prelabelled pixels to a moderate number of prelabelled pixels, an improvement in performance was not observed for all images. Image 291000 of a horse is such an image where adding more prelabelled pixels did not improve the performance of our method. Figure 10 shows the locations of the prelabelled pixels superimposed over image 291000. As can be observed in the image, the foreground and background are very heterogeneous. For example, the horse has a local dark area on its mane and a local bright red area on its rear. Therefore, it is difficult to replicate the distributions of the foreground and background with prelabelled pixels that consist of groups of neighboring pixels, leading to the observed lowered performance in heterogeneous images.

Fig. 10. Left: minimal density image marking. Right: Moderate density image marking.

When the features in different segments, foreground and background, are homogenous or nearly homogeneous, the initial selection does not have as great of an effect. An example of this can be seen in Figure 11, with the bird on the branch against the sky as a background. This is because there are smaller pixel gradients with-in the foreground and background, but significantly large one separating the segments. The boundary penalty factor here will be more pronounced in the locations with large pixel intensity differences.

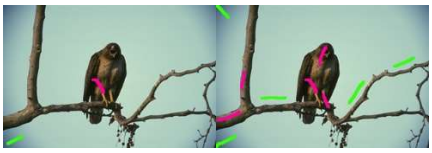

Fig. 11. Left: Minimal density image marking. Right: Moderate density image marking.

C. *Comparison to Other Segmentation Methods*

The Best Berkeley benchmark algorithms results on the images we selected were primarily the gPb-ucm (gray), Ren et al. NIPS2012 (gray), and xren (gray) algorithms. The "gPb-ucm (gray)" algorithm is from the paper "Contour Detection and Hierarchical Image Segmentation" and uses local contour detection cues to perform global spectral clustering.[10]

The best result for our segmentation method was image 296059 of an elephant that was segmented using a high number of prelabelled pixels. We achieved a IOU of 0.967 and an F1

Score of 0.942. The best result from the Berkeley dataset bench marks displayed with the segmentation outline resulted in an F1 score of 0.9 through the use of Boundary Detection Benchmark: Algorithm "gPb-ucm (gray)"[10]. The elephant image has distinct feature characteristics on the sky, grass and elephant regions of this image, which lead to good F1 scores in the two techniques.

The Ren et al. NIPS2012 (gray) algorithm is introduced in "Discriminatively Trained Sparse Code Gradients for Contour Detection"[12]. This method uses a clustering method to group and classify pixel neighborhoods to identify contours in images.

Our worst result for image 210088 of a fish in a plant with the segmentation using a low number of prelabelled pixels. This segmentation resulted in an IOU of 0.042 and an F1 score of 0.373. The best segmentation of this image in the Berkeley benchmarks reported an F1 score of 0.77, through the use of the Boundary Detection Benchmark: Algorithm "Ren et al. NIPS2012 (gray)"[12].

In Contrast to our low F1 score for our low marking method, our F1 score for this image on our high density marking image resulted in an F1 of 0.838, out pperforming the boundary detection segmentation method used in the benchmarks.

Additionally the human segmentation of this image resulted in a F1 of 0.28. The reason for failures on this image are due to the image being divided into fish and plant, according to the ground truth image segmentation. However the plant and the fish appear to both be in the focus and in the foreground of this image.

The "xren (gray)" algorithm was introduced in "Multi-Scale Improves Boundary Detection in Natural Images"[11]. This method is a boundary detection technique that uses local boundary cues like contrast, this method is effective with large scale detection and small-scale detection, but is sensitive to cluttered images. The boundary detection algorithm "xren(gray)" for image segmentation was another high performance algorithm on the group of images we segmented. The F1 score for xren on the image 196073 of a snake in the sand was 0.82. With our graph-based image segmentation using the low, medium, and high density marking methods we achieved F1 scores of 0.7123, 0.897, and 0.927 respectively.

VIII. Future Work

A significant source of error in our method was due to inadequately estimating the probability density functions of foreground and background pixels. The probability density function estimations can be improved through utilizing a series of different probability density functions and choosing the one that best fits the prelabeled pixels. Gaussian mixture model probability distributions could also be explored in the case of complex foreground and background probability distributions. Furthermore, the estimated probability density functions could be based on the red, blue, and green values of the pixels rather than their grayscale values. This would decrease the chances of the distributions of the foreground and background being similar and would increase the segmentation performance of our method.

Future improvements would be to replace the interactive marking step with another method for boundary detection, such as connected components or contour detection methods. The graph-based method to determine segmentation boundaries could be applied to the contour detection methods to modify the boundaries in a way that satisfies our objective function. Image thresholding can also be done to split the image into two groups based on an intensity threshold. Also a k-means pixel clustering segmentation could be used as a replacement for the image marking, with the graph method applied to the image with the k-means cluster assigned labels. Methods for initial foreground and background estimations would complement our graph based method, by improving the initial foreground and background labels.

References

[1] "What Is Image Segmentation?," Image Segmentation. [Online]. Available:https://www.mathworks.com/discovery/image-segmentation.html. [Accessed: 14-Apr-2021].

[2] Y. Y. Boykov and M. -. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001, Vancouver, BC, Canada, 2001, pp. 105-112 vol.1, doi: 10.1109/ICCV.2001.937505.

[3] J. Kleinberg and Tardos Éva, "Image Segmentation," in Algorithm design, Boston, Massachusetts: Pearson/Addison-Wesley, 2014, pp. 391–395.

[4] "cvtColor," Miscellaneous Image Transformations, 2019. [Online]. Available: https://docs.opencv.org/2.4/modules/imgproc/doc/miscellaneous_transformations.html. [Accessed: 14-Apr-2021].

[5] E. Tiu, "Metrics to Evaluate your Semantic Segmentation Model," Medium, 03-Oct-2020. [Online]. Available: https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2. [Accessed: 14-Apr-2021].

[6] A. Hagberg, "networkx.algorithms.flow.minimum_cut," NetworkX Documentation, 22-Aug-2020. [Online]. Available: https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.flow.minimum_cut.html#networkx.algorithms.flow.minimum_cut. [Accessed: 13-May-2021].

[7] A. Hagberg, "Source code for networkx.algorithms.flow.maxflow," NetworkX Source Code, 22-Aug-2020. [Online]. Available: https://networkx.org/documentation/stable/_modules/networkx/algorithms/flow/maxflow.html#minimum_cut. [Accessed: 13-May-2021].

[8] P. Arbelaez, The Berkeley Segmentation Dataset and Benchmark, Jun-2007.[Online].Available: https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/bench/html/algorithms.html. [Accessed: 14-Apr-2021].

[9] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 26, no. 9, pp. 1124-1137, Sept. 2004, doi: 10.1109/TPAMI.2004.60

[10] Contour Detection and Hierarchical Image Segmentation. P. Arbelaez, M. Maire, C. Fowlkes and J. Malik. IEEE TPAMI, Vol. 33, No. 5, pp. 898-916, May 2011.

[11] Xiaofeng Ren, "Multi-Scale Improves Boundary Detection in Natural Images", ECCV, 2008.

[12] Xiaofeng Ren and Liefeng Bo, "Discriminatively Trained Sparse Code Gradients for Contour Detection.", NIPS 2012.

[13] https://networkx.org/documentation/stable/_modules/networkx/algorithms/flow/maxflow.html#minimum_cut