Interactive Image Segmentation with GrabCut

Bryan Anenberg
Stanford University
anenberg@stanford.edu

Michela Meister
Stanford University
mmeister@stanford.edu

Abstract

We implement GrabCut and experiment with three extensions. We try varying the number of GMM components used, reinitializing GMM components after a few iterations, and restricting the background Gaussian Mixture Model to pixels to be within the bounding box.

1. Introduction

The goal of foreground-background segmentation is to divide the pixels of an image into exactly two sections, one foreground and one background. Foreground-background segmentation is a challenging task, and has many applications in object recognition and classification [3].

A successful segmentation requires knowledge of both local features and global features in an image. GrabCut, which we will discuss in more detail in the following section, minimizes an energy equation that balances both local and global relationships between pixels [9]. Since its development in 2004, GrabCut has been applied to many different segmentation problems. For example, Lempitsky et al of Microsoft Research modified GrabCut to favor foreground segmentations that are tight to the given bounding box [6]. Of course, for many images a bounding box is not available. However, Guillaumin et al developed a method that propagates known segmentations across ImageNet to initialize GrabCut on similar images that have no annotated bounding box [3]. When no bounding box in a set of images is available, another option is cosegmentation, which analyzes similar images together to identify a foreground segmentation [5].

2. Algorithm

2.1. GrabCut

GrabCut chooses a segmentation by iteratively revising foreground and background pixel assignments. At the start, GrabCut initializes a foreground and background model using the bounding box. Any pixel outside the bounding box can be confidently assigned to the background. Pixels within the bounding box are initially assigned to foreground. Using these initial assignments, GrabCut then develops a Gaussian mixture model (GMM) for the foreground and background respectively. Each GMM is made up of multiple components, which represent clusters of similar pixels. GrabCut then defines a distance measure between each pixel and the foreground and background models, based on the component most similar to the pixel in each model. GrabCut also calculates the color distance between each pixel and its neighbors.

Using these two distance measures, GrabCut models the entire image as a directed, weighted graph, where each pixel has an edge to a foreground source node, a background sink node, and each of its eight neighbors. The distance measures (D) to the foreground and background models weight the edges from a pixel to the source and sink nodes, respectively. These edges represent the cost of assigning an individual pixel to the foreground or background.

$$U(\alpha, k, \theta, z) = \sum_n D(\alpha_n, k_n, \theta, z_n)$$

$$D(\alpha_n, k_n, \theta, z_n) = -\log \pi(\alpha_n, k_n) + \frac{1}{2}\log\det\Sigma(\alpha_n, k_n) + \frac{1}{2}\left[z_n - \mu(\alpha_n, k_n)\right]^T \Sigma(\alpha_n k_n)^{-1}\left[z_n - \mu(\alpha_n, k_n)\right]$$

Edges from a pixel to its neighbors have weights that consider both the color distance between the two pixels and the landscape of the pixel neighborhood. These pairwise edges (V) represent the cost of assigning neighboring pixels to different segments and are engineered with β such that neighboring pixels are only given opposite assignments when there is a significant change in color over multiple pixels.

$$V(\alpha, z) = \gamma \sum_{(m,n)\in C} [\alpha \not= \alpha_m] \exp -\beta||z_m - z_n||^2$$

$$\beta = \left(2\langle(z_m - z_n)^2\rangle\right)^{-1}$$

This helps encourage smooth segmentations - one outlier pixel in an otherwise homogenous segment will have the same assignment as its neighbors even if it is a very different color.

GrabCut then calculates the minimum cut of the constructed graph to find the minimum-cost segmentation (E) and re-assigns pixels to the foreground and background ac-

cordingly. The entire process then repeats until convergence by relearning the GMM models and constructing another graph, etc [9].

$E(\alpha, k, \theta, z) = U(\alpha, k, \theta, z) + V(\alpha, z)$

2.2. Our Algorithm: Standard Implementation

Our standard GrabCut implementation uses five components for the GMMs and an 8-connected graph, just as in the original implementation [9]. We initialize the GMMs on the first iteration using kmeans. The original only calculates assignments for pixels that were in the foreground on the previous iteration. We re-calculate foreground and background GMM assignments for every pixel on each iteration in order to update the GMM parameters. We artificially keep pixels previously assigned to background in the background during reassignment by setting high weights to the edges that connect these pixels to the foreground source.

2.3. Our Algorithm: Extensions

Motivated by the subpar segmentation of the banana in figures 2 1 , we experiment with using different numbers of GMM components. Our hypothesis is that a system with five GMM components overrepresents the color distribution of the foreground, resulting in foreground GMM components with color means similar to the background. As a result, pixels that should be assigned to background are instead assigned to foreground.

In addition, our original segmentation of the elephant in figure 1 inspires us to reset the number of GMM components after a few iterations. Each trial begins with a constant 5 GMM components, however after 3-4 iterations we reset the number of components and reinitialize using kmeans. Our hypothesis here is that the foreground is mostly stabilized after a few iterations and so reducing the number of GMM components at this point could help shed some remaining background material from the foreground segmentation.

In our final extension we experiment with a different background model. Instead of generating the background GMM using all the pixels assigned to the background, we only use the pixels assigned to background that fall within the bounding box (as introduced in [2]). Our idea is that this might produce better segmentations, because the background GMM will better represent the local variation in background surrounding the foreground object (figure 8). To test this technique we also reinitialize the GMM components after 4 iterations, as in the preceding extension.

3. Code

A README is included in the code. File Overview:

1. grab_cut_fcn.py
   Implementation of GrabCut with various options to implement the 3 extensions. [7; 4; 1; 8]

2. grab_cut_script.py The script used to compute all of the segmentations featured in the paper.

3. simple_script.py Simple script to run the GrabCut segmentation with the experimentally discovered best settings. e.g, initialize foreground and background GMMs with 5 components, restrict the background GMM to pixels within the original bounding box, and reduce the number of GMM componenets to 3 after the 4th iteration of energy minimization.

4. Results

4.1. Standard Implementation

Our standard implementation segments the dataset with 95.44% accuracy and 84.12% Jaccard similarity. As seen in figure 2, some images segment better than others. The llama, teddy bear, and banana on apple background images (figure 1) segment well, whereas the scissors and banana on yellow background segment poorly. These two cases are examples of overfitting, where the background sections above the banana and inside the thumbholes of the scissors are assigned to foreground since they identify with one of the foreground GMM components. The elephant image (figure 1) segments moderately well, although the section between the elephants trunk and legs is mislabeled as foreground.

4.2. Extensions

4.2.1 Vary the number of GMM components

Varying the number of GMM components used (table 1) improves segmentations for some images, as seen in figure 3. Specifically, the banana on yellow background segments very well with one to three GMM components, after which there is a steep drop in Jaccard similarity. The cross and scissors also have their best segmentations with one GMM component. We see in Figure 4 that the segmentation of the scissors with only one GMM component no longer overfits the area inside the thumbholes, and these sections are correctly assigned to background. Segmentations of both the cross and the scissors degrade as the number of GMM components increases, with the cross classified entirely as background when eight components are used. (This case occurs since at initialization the foreground and background GMMs are likely very similar and by chance the pixels in the true foreground as assigned to the background.) The grave segmentation, however, improves as the number of GMM components increases, likely because its background is so varied. From these results we can see that images with fairly constant backgrounds segment better with fewer GMM components, whereas images with more variation in

K	Accuracy	Jaccard
1	97.01	82.76
2	96.29	84.34
3	96.41	85.36
4	95.56	84.64
5	95.44	84.12
6	95.54	84.52
7	95.67	84.16
8	94.60	82.22
9	95.47	83.74
10	95.47	83.72

Table 1: Average Accuracy and Jaccard Similarity as a function of number of Gaussian components.

the background, such as the grave, segment better with more GMM components.

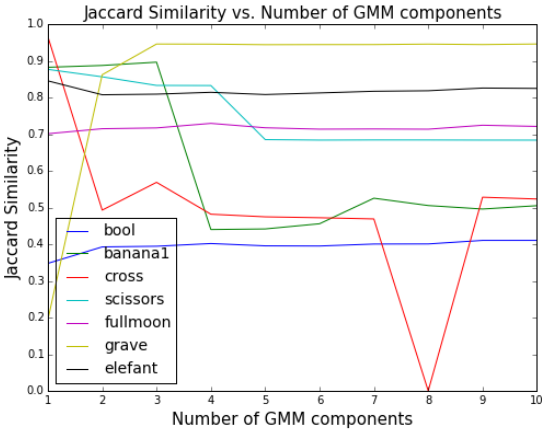

Figure 3: This figure displays the Jaccard Similarity as a function of the number of foreground and background GMM componenets. Jaccard similarity is the intersection over the union of the predicted foreground region with the ground truth.

Figure 4: banana1 and scissors.

4.2.2 Reinitialize GMMs after 3rd iteration

Reinitializing the number of GMM components after the third iteration (table 2) tends to stabilize segmentations as seen in figure 5. For example, the cross segmentation does not experience such a drastic dip in Jaccard similarity at 8 components and the grave at 1 component. The elephant segments well with 6 components after reinitialization, and we see in figure 6 that the segmentation no longer mislabels the sections of background next to the elephant as foreground. This is likely because the foreground is mostly localized after a few iterations, which means that, at this point, we only need a few components to model it well, and so including more components in our GMM only leads to overfitting.

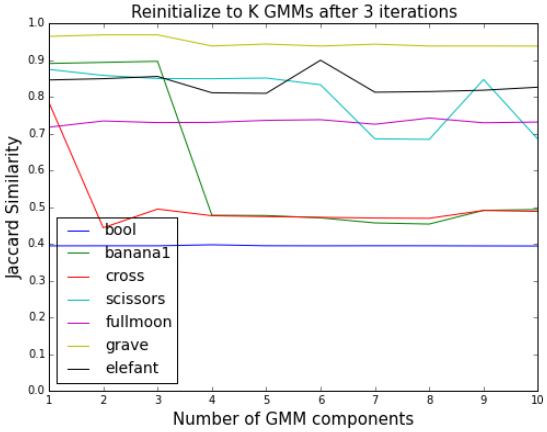

Figure 5: This figure displays the Jaccard Similarity as a function of the the number of GMM components after reinitialization on the 3rd iteration. The performance is similar to that of figure 3, but much more stable. For example, the initial 5 component GMM yields a rough segmentation of the grave. However, by reinitialzing to 1 component, the holes at the top of the grave are assigned to background since the majority of the foreground contains the gray grave color.

Figure 6: Left: original elefant image. Middle: Segmentation using standard GrabCut. Right: Segmentation by reinitialzing GMM componenets after 3rd iteration.

K	Accuracy	Jaccard
1	97.27	85.48
2	96.49	85.97
3	96.72	86.70
4	95.68	84.72
5	95.68	84.79
6	95.82	85.12
7	95.50	84.15
8	95.47	84.11
9	95.82	84.87
10	95.50	83.61

Table 2: Average Accuracy and Jaccard Similarity as a result of reinitializing after 3 iterations to K Gaussian components.

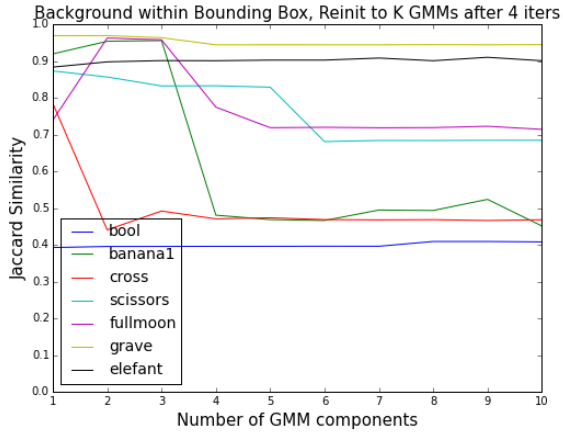

Figure 7: This figure displays the Jaccard Similarity as a function of the the number of GMM components when modeling the background only using pixels from within the original bounding box and reinitializing on the 4rd iteration. 11 iterations are run in total. Notice the improvement in the fullmoon segmentation.

4.2.3 Constrain background pixels to within original bounding box

Restricting the background model to only include pixels within the bounding box (figure 7, 3) works especially well for the fullmoon image (figure 8) when the number of GMM components is reset to two or three. This is likely because the black background is not uniform over the entire image and has slightly different color in the area bordering the moon. By only including pixels originally in the bounding box in our background model, we create a better local model for the background that borders the moon and therefore can produce a finer segmentation. While this works well for the moon, this extension still does not solve the segmentation for the "bool" image (figure 9). Unlike the moon, the bool foreground has very similar color to the background, as the mans vest is nearly the same color of green as the grass. Because of this, using a more local background model still does not help us differentiate between foreground and background. This extension yields some of the best segmentations (table 3) and strongest overall result (figure 10).

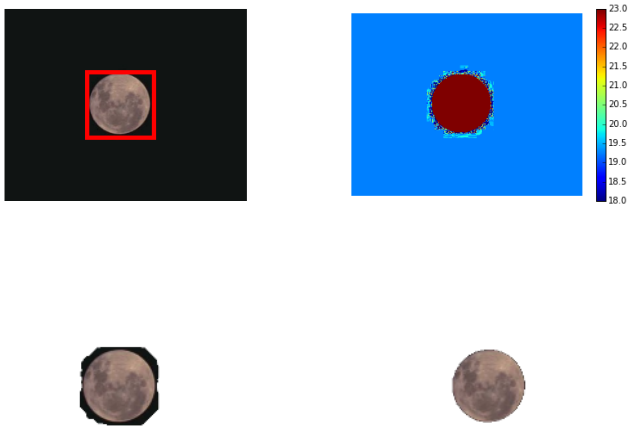

Figure 8:

- Top left: original fullmoon image with bounding box.
- Top right: original fullmoon image transformed into grayscale and then displayed between the intensities between the range 18 to 23. The majority of the background pixels have exactly the same value: 19.26. The background pixels immediately surrounding the fullmoon vary between 18 and 20. The pixels interior to the fullmoon have values greater than 23.
- Bottom left: The fullmoon segmentation using the standard GrabCut algorithm.
- Bottom right: The fullmoon segmented by constraining the background to within the bounding box.

5. Conclusion

From our three experiments we see that the number of GMM components used can have a large effect on the final segmentation for certain images. Moreover, reinitializing GMM components after a few iterations can stabilize these segmentations. Finally, we achieve our best result of 96.94% accuracy and 87.71% Jaccard similarity when we constrain the background model to only the pixels within the bounding box on all but the initial iteration. This approach allows the model to better represent the local color distribution around the object. Nevertheless, some images still may not segment well, because of similarities between

K	Accuracy	Jaccard
1	97.11	84.61
2	96.77	87.53
3	96.94	87.71
4	95.93	85.53
5	95.87	85.16
6	95.74	84.57
7	95.88	84.95
8	95.87	84.88
9	95.97	85.05
10	95.71	84.74

Table 3: Average Accuracy and Jaccard Similarity after sampling background from within bounding box and still reinitializing to K Gaussian components.

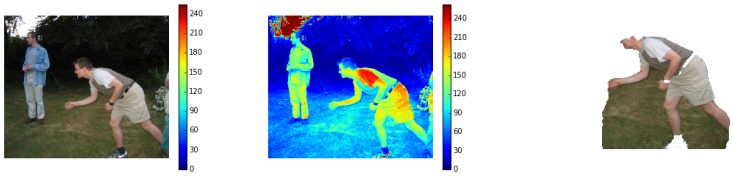

Figure 9: Left: original "bool" image. Notice the grass is a similar color to the vest of the man. Middle: original "bool" image as displayed using grayscale intensity. Right: "bool" segmented by constraining the background to within the bounding box.

the foreground and background. In these cases, we could further improve our segmentations by including user interaction.

References

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

[2] D. Chen, B. Chen, G. Mamic, C. Fookes, and S. Sridharan. Improved grabcut segmentation via gmm optimisation. In *Digital Image Computing: Techniques and Applications (DICTA), 2008*, pages 39–45, Dec 2008.

[3] Matthieu Guillaumin, Daniel Kttel, and Vittorio Ferrari. Imagenet auto-annotation with segmentation propagation. *International Journal of Computer Vision*, 110(3):328–348, 2014.

[4] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[5] A. Joulin, F. Bach, and J. Ponce. Discriminative clustering for image co-segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1943–1950, June 2010.

[6] Victor Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. Image segmentation with a bounding box prior. In *ICCV*, number MSR-TR-2009-85, 2009.

[7] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[8] pmneila. Pymaxflow: Python library for creating flow networks and computing the maxflow/mincut. *GitHub repository*, 2015.

[9] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.

Figure 1: Top row are the ground truth original images, bottom row are the segmentations using the standard GrabCut algorithm (5 component GMMs). Images towards the left segment well, whereas images towards the right segment poorly.

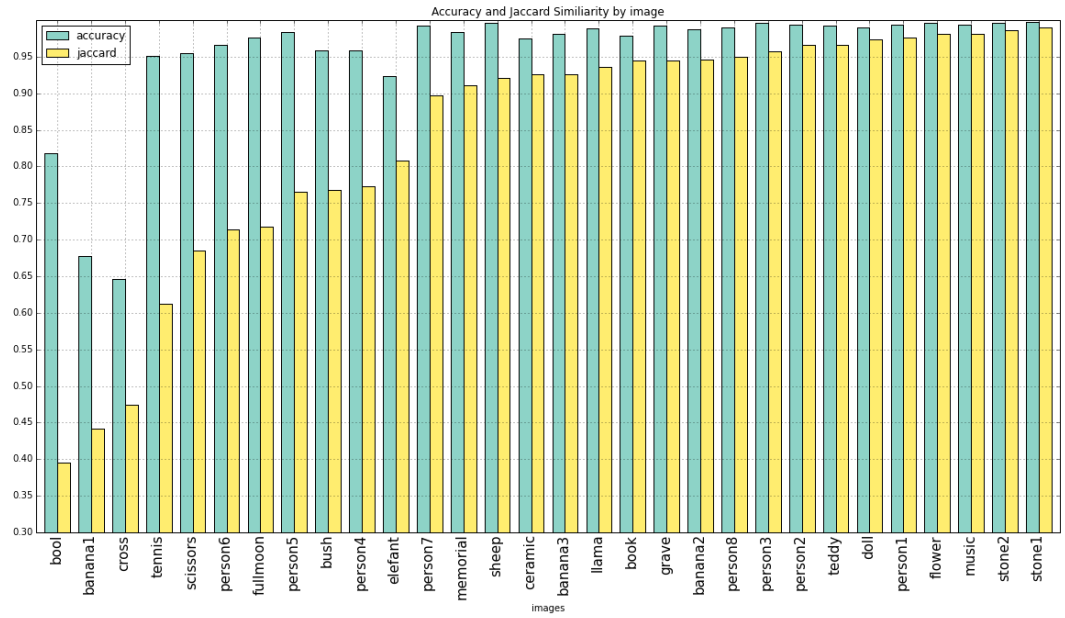

Figure 2: Accuracy and Jaccard similarity across all images for the standard GrabCut algorithm (5 component GMMs). Average Accuracy: 95.44% Average Jaccard Similarity: 84.12%

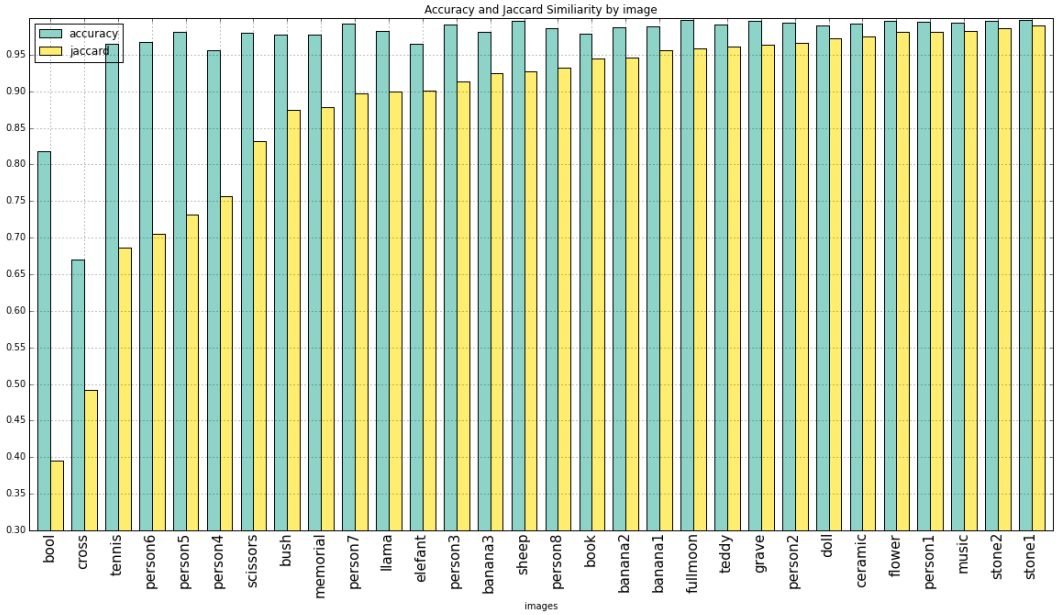

Figure 10: Best settings. Average accuracy: 96.94% Jaccard similarity: 87.71%