6.865 Final Project: Graph Cuts and GrabCut for Image Segmentation

Gregory Izatt
gizatt@mit.edu
MIT

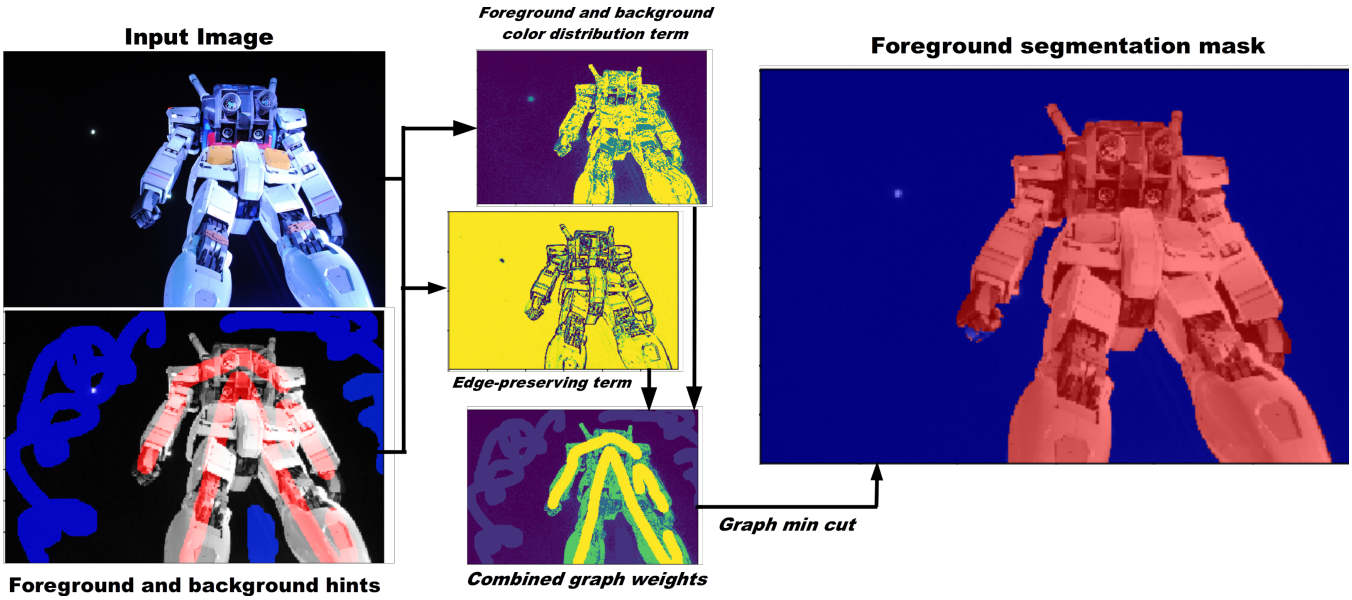

Figure 1: My GraphCut implementation applied to a real-world image. An input image, along with background and (optional) foreground segmentation hints, is converted to a graph representation that is constructed such that a minimum edge cut over the graph corresponds to a foreground / background segmentation.

ABSTRACT

I implemented classic graph-cut image segmentation [Boykov and Jolly 2001] (with GrabCut-style cropping [Rother et al. 2004]) in Python, carefully leveraging sparse matrix libraries and vectorized operations to maintain speed. I tested graph-cut foreground segmentation quality on an image segmentation dataset, comparing performance across a handful of parameter settings and labeling decisions. My results confirm that these classic techniques perform admirably on many images, but are relatively easy to break.

1 INTRODUCTION

Graph-cut based techniques for image segmentation operate by forming an undirected weighted graph over image pixels and foreground/background source nodes, defined such that taking a minimum cut over the graph corresponds to finding an optimal segmentation. (See Figure 2.) This technique can be seen as a special case of a Markov Random Field approach [Boykov et al. 1998] (connection discussed in [Hoiem 2008]). Via that same connection to MRFs, graph-cut techniques also strong theoretical connections to spectral matting methods, e.g. [Levin et al. 2008], in that spectral methods approximate the min-cut operation with a generalized eigenvalue problem [Shi and Malik 2000]. This broad class of graph-based image segmentation continues to evolve, with recent work using deep learning to build better affinity matrices, while still using reliable eigendecomposition methods for the final solve (e.g. [Aksoy et al. 2018]).

In this report, I review results from implementing a basic version of graph-cut, which follows closely along with Boykov and Jolly's 2001 implementation [Boykov and Jolly 2001] and GrabCut [Rother et al. 2004], and testing it on a number of real-world test images from [Rubinstein et al. 2013] (which itself is an eclectic mix of internet images, as well as images from iCoseg [Batra et al. 2010] and MSRC [Shotton et al. 2006]). I compare performance across a small number of parameter choices to evaluate the original author's suggestions, and find that these classic techniques perform admirably on many images, but fail in low-contrast, cluttered, or otherwise complex scenes. *Critically, I do not evaluate the performance of these techniques as used in a closed loop with an artist, as they were originally intended – I instead evaluate their single-shot segmentation quality given either a degraded foreground segment hint, or no foreground hint at all.* This analysis provides a benchmark for what performance baseline methods can reasonably achieve on realistic images, which is useful when seeking to understand the complexity-performance tradeoff inherent in cutting-edge approaches.

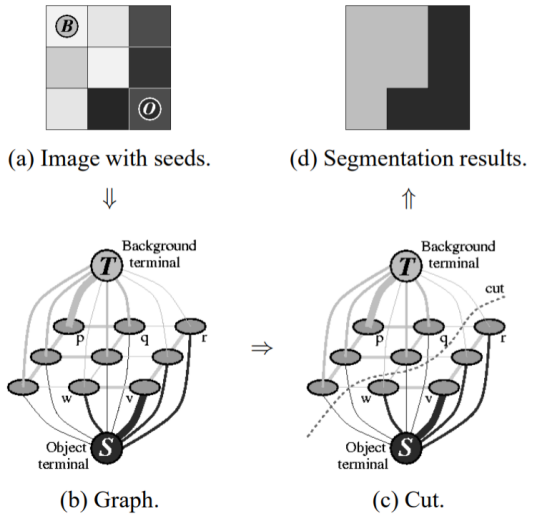

Figure 2: Overview of graph-cut segmentation. (a) An image with optional foreground object (O) and background (b) labels is supplied. (b) A sparse undirected weighted graph is formed from the image. Thicker edges indicate stronger weights. (c. d) A minimum cut, separating the foreground and background terminals, defines a segmentation of pixels. Figure from [Boykov and Jolly 2001].

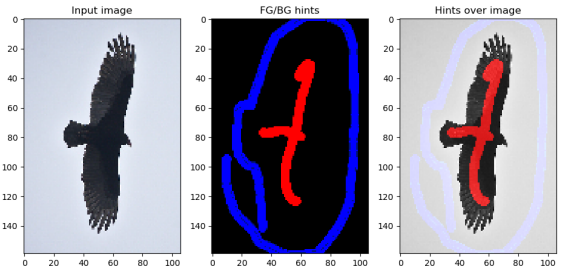

Figure 3: The graph-cut segmentation system operates on an RGB or grayscale image $\mathcal{I}$ and a set of foreground and background mask hints. *Left:* An example RGB image. *Middle:* A set of foreground (red) and background (blue) hints. *Right:* The hints overlayed on the original image. *(In my implementation, the FG/BG hints are supplied by the user in the form of a color image matching the target image size, with FG hints in the red channel and BG hints in the blue channel.)*

2 SEGMENTATION METHOD

All of the following description follows directly from [Boykov and Jolly 2001] and [Rother et al. 2004]. The basic graph-cut segmentation method takes as input an RGB image, a binary foreground mask hint, and a binary background mask hint (Figure 3), and produces a binary background/foreground classification for each pixel (e.g. Figure 4).

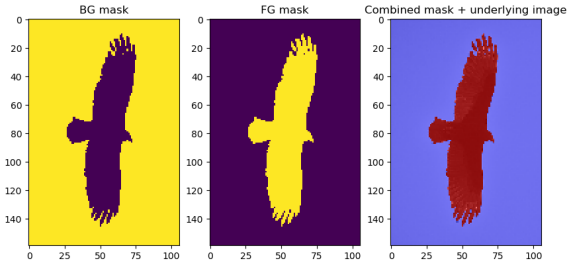

Figure 4: An example segmentation performed by my graph-cut implementation. *Left and middle:* The background and foreground segmentation masks (yellow = 1, purple = 0). *Right:* The masks overlayed on the original image, with increased red component in the foreground and increased blue component in the background.

The essence of this foreground-background segmentation approach is to form a sparse, weighted, undirected graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$ with nodes $\mathcal{V}$ and edges $\mathcal{E}$. The node set $\mathcal{V}$ contains a node for each pixel of the input image, as well as one source (s) node and one sink (t) node. The min s-t cut over this graph will create two partitions – the one containing the source will be the foreground, and the one containins the sink will be the background. To make this happen, each pixel is connected to its 8 neighbors (assuming a 2D image), the sink node, and the source node (Figure 2::b).The weight of these connections varies:

- Edge connections to neighbors are weighted more heavily if the nodes have similar colors, and less heavily if the nodes have dissimilar colors. Specifically, for a neighbor pair $\{p, q\}$ with color or luminance values $\{I_p, I_q\}$, the weight is

$$\exp\left(-\frac{||I_p - I_q||_2^2}{2\sigma^2}\right) \cdot \frac{1}{\text{dist}(p,q)},$$

  where the distance is taken in pixel space, and sigma is chosen based on the average difference across the image:

$$\sigma = K_\sigma \times mean\left(||I_p - I_q||_2^2\right).$$

  In my implementation, a different σ is calculated for each of the 8 directions, and K_σ is a hyperparameter near 1.
- Connections to the sink and source nodes are based on how the pixel is labeled:
  - *Foreground-labeled pixels* are connected to the sink with weight K_{big} and the source with weight 0.
  - *Background-labeled pixels* are connected to the source with weight K_{big} and the sink with weight 0.
  - *Unlabeled pixels* are connected to the source with weights $-K_\lambda * \log(P(I_p|O))$, and likewise to the sink with weights $-K_\lambda * \log(P(I_p|O))$, with $P(I_p|O)$ and $P(I_o|B)$ being calculated by taking a normalized histogram over the luminance intensities of labeled foreground and background pixels respectively. Histograms are taken with 50 bins in this

work, and if no labeled pixels are available, connections of weight 0 are added[1]

K_{big} is chosen to be sufficiently big such that those connections will never be cut in a minimal cut (see [Boykov and Jolly 2001]). K_λ is a hyperparameter around 1/50.

2.1 Implementation Notes

I construct this graph using the scipy sparse matrix library, and take a minimum cut over it with the networkx library using the preflow-push algorithm. [Boykov and Jolly 2001] advocates for the min-cut algorithm detailed in [Boykov and Kolmogorov 2004], which uses a different heuristic that terminates empirically faster for the wide-and-shallow sparse graphes used in computer vision; switching to that algorithm (via e.g. creating Python bindings for the code at https://vision.cs.uwaterloo.ca/code/) would benefit my implementation a lot. As it stands, forming the sparse graph takes on the order of 1 second, and performing the cut 10 seconds, for a 150x150 pixel image. (These numbers vary based on weight values, as they influence the search path taken by the max-flow solver.)

3 BENCHMARKING RESULTS

I tested this algorithm on a handful of my own images (see Figures 1, 4, and 8), but in the name of science, also tested the algorithm across a 450-image randomly-selected subset of the segmentation dataset provided by [Rubinstein et al. 2013], for each of the following parameter choices:

- Taking the contrast value in color or **luminance** space.
- $K_\sigma = \{0.5, \mathbf{1}, 2\}$
- $K_\lambda = \{1/10, \mathbf{1/50}, 1/250\}$
- Providing a foreground hint, or **not providing a foreground hint**.

with the nominal values (in bold) being the values from GrabCut [Rother et al. 2004]. Each image was scored against human-labeled ground truth via pixelwise intersection-over-union (IOU). Foreground hints were provided by pixelwise eroding the ground truth foreground segment by 10% the foreground segment bounding box size (see Figure 5).[2] Images were cropped around their subjects according to the foreground mask in cases where the subject occupied a sufficiently small amount of the image.

Performance across conditions is reported in Figure 7. The best condition appears to be $K_\sigma = 0.5, K_\lambda = 0.1$, which weights the pixelwise foreground and background color distribution log-probability higher (by 5x) than [Rother et al. 2004] suggested. A set of example segmentations from the test dataset and my own images is presented in 1.

[1]This isn't exactly the right thing to do, but seemed to be a reasonable compromise – a weight of 0 is approximtely in the middle of the range of typical log density values for pixel colors. In [Rother et al. 2004], the authors employ an iterative EM procedure in which they alternate taking min cuts to find candidate foreground regions, and fitting GMM color models to those candidate foreground regions. I suspect this results in a sort of region growing in which the first iteration will grab obvious parts of the foreground object using primarily edge information (and negative information about what doesn't match the background), and then future iterations find more similar pixels to those obvious ones to complete the foreground.

[2]This definitely wasn't perfect / hurt performance slightly, as subjects that were thin but long might have their labels eroded entirely. It was the best I could come up with and implement quickly...

Figure 5: *Left:* An example image from the test dataset, cropped around the subject. *Middle:* The human-annotated ground truth foreground mask. *Right:* An eroded version of the foreground mask, used as a foreground hint for one condition of the benchmark process.

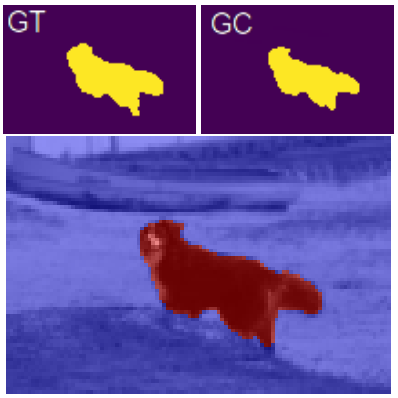

Figure 6: GraphCut applied to the test image from Figure 5. *Top Left:* Ground truth segmentation. *Top right:* Segmentation from GraphCut ($K_\sigma = 1.0$, $K_\lambda = 0.02$, using color and foreground hint). *Bottom:* Original image colorized with GraphCut segmentation.

4 CONCLUSIONS

GraphCut works adequately, *even with no interaction*, on images with obvious foreground objects, but struggles in cases where the foreground "blends in" in terms of color. Naturally, its performance dramatically improves given human interaction. The dataset I used here is hard![3]

My implementation has plenty of issues, too! As is relatively clear from Figure 9, my technique of eroding the ground truth mask to produce "fake" foreground hints created bad foreground hints in many cases, which skewed my results negatively. (Even running these tests again against *uneroded* masks would be informative: in what cases does GraphCut overstep its boundaries?) The min-cut algorithm I'm using is slow: I did not use the version of [Boykov and Kolmogorov 2004], which is specialized for computer-vision problems. Speeding up the cut algorithm would allow me to run on higher-res images and experiment with interactivity, which would be fun and probably informative. I used the histogram model over luminance values (of [Boykov and Jolly 2001]) instead of the GMM formulation from GrabCut [Rother et al. 2004] for representing

[3]The authors of the dataset [Rubinstein et al. 2013] used it as part of a study on segmentation in "noisy" datasets where the images were relatively unfiltered image search results, in which some images were bad, and some didn't even include the target object at all. I threw out images where the ground truth masks were empty (the "noise" images in the dataset) but there still remained images with multiple targets (I picked the "first" one).

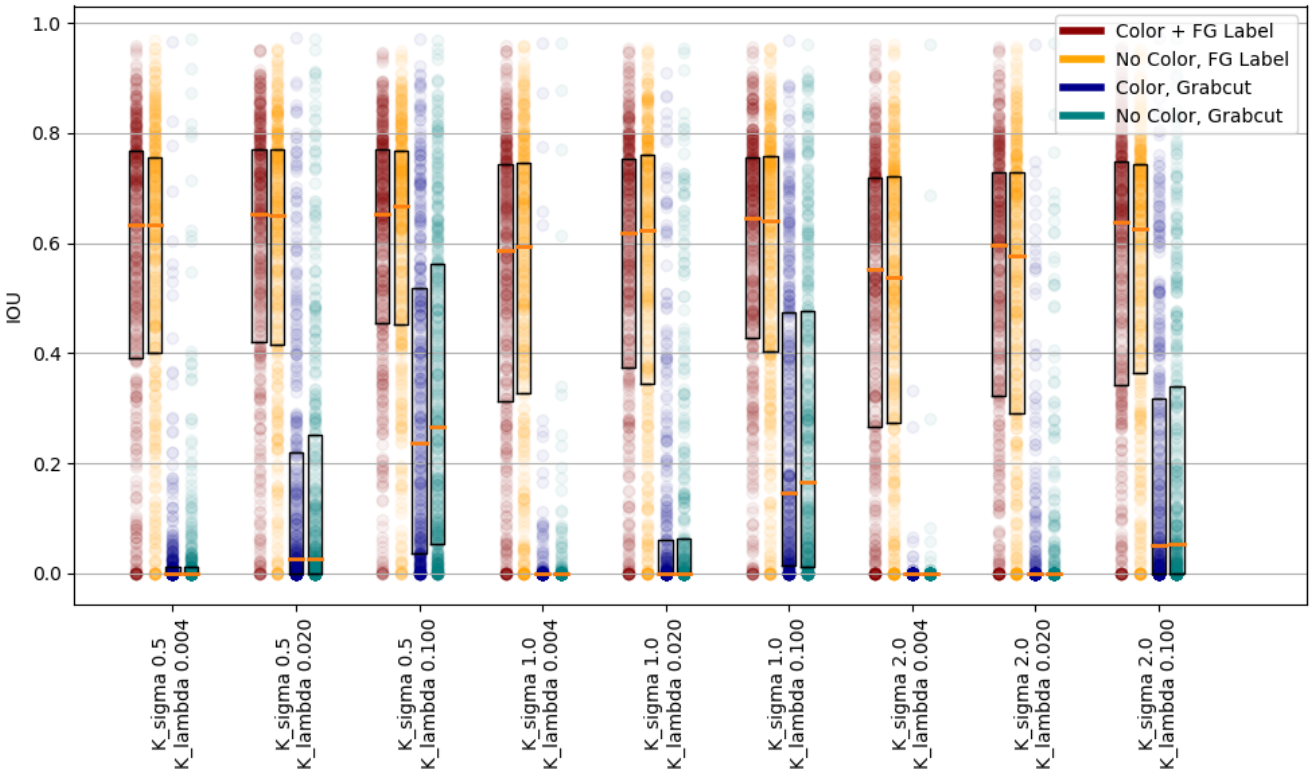

Figure 7: Performance of the GraphCut method across parameter settings. Box-and-whisker plots show medians (orange horizontal lines) and upper and lower quartiles (box bounds) per condition. Individual scores for image (N=450) at each condition are shown as dots. Yeah, it's a mess. "Grabcut" refers to the no-foreground-hint condition. Color vs no color (red vs orange and dark blue vs teal) had little effect. While overall performance was similar across almost all tested settings (within the Grabcut / not-Grabcub conditions), wildly varying image content led to massive variations in performance between images.

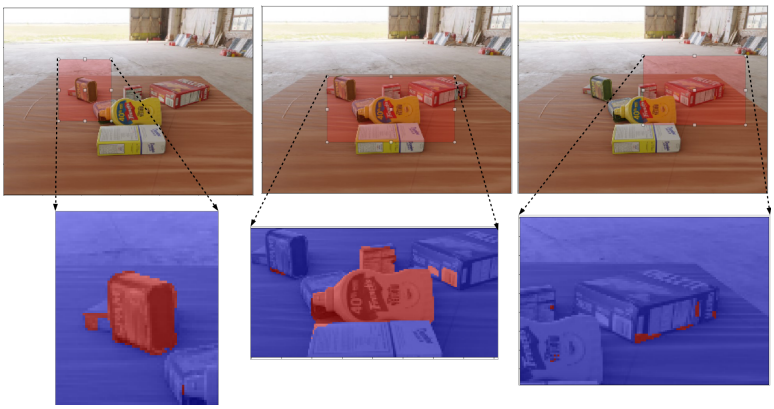

Figure 8: I include a GrabCut-style [Rother et al. 2004] interface for specifying object regions of interest to segment: instead of providing an explicit foreground label, the user specifies a bounding box. The outer 10 pixels of the bounding box are labeled background, and no foreground labels are supplied. Resulting segmentations are shown in the bottom row, with the background colored blue and the foreground colored red. For subjects that are well-separated in terms of edges and color content, this method works surprisingly well (e.g. left). However, objects that are similar intensity and color to the background are less likely to be picked out without additional information (right).

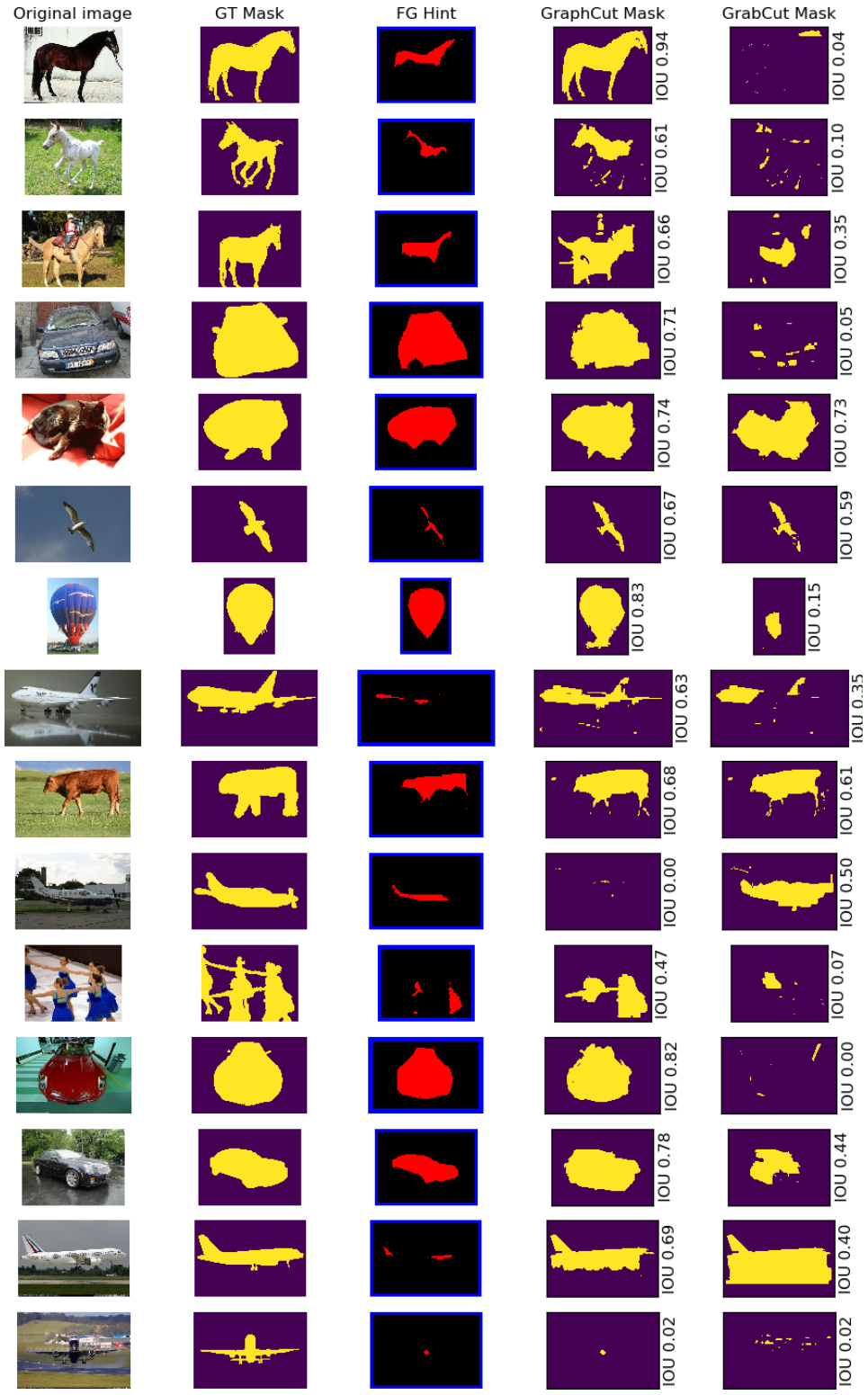

Figure 9: Grabcut (no foreground hint) and GraphCut (foreground hint as shown in column 3) results for random images from the test set, using the "best" conditions on the test set $K_\sigma = 0.5, K_\lambda = 0.1$. IOU is labeled to the right of the respective foreground mask image.

the density of a pixel given the distribution of labeled pixels; it's hard to tell how limiting that is, though an enormous amount of the contrast of *any* image comes from the luminance channel, so I hope this choice was not seriously limiting. As mentioned in the caption of Figure 8, in the case when no foreground pixels are labeled, no foreground distribution is *ever* defined, whereas[Rother et al. 2004] uses an EM algorithm to gradually grow a candidate foreground region according to the foreground distribution of the last iteration. Implementing that blocks on having a more efficient min-cut algorithm.

GraphCut (and the related line of MRF and graph-spectral methods) provide an interpretable and flexible – but powerful – backbone for image segmentation. I hope I get to play with it more in the future!

ACKNOWLEDGMENTS

Thanks to Fredo and the TA's for running an excellent class! :)

REFERENCES

Yağiz Aksoy, Tae-Hyun Oh, Sylvain Paris, Marc Pollefeys, and Wojciech Matusik. 2018. Semantic soft segmentation. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 72.

Dhruv Batra, Adarsh Kowdle, Devi Parikh, Jiebo Luo, and Tsuhan Chen. 2010. icoseg: Interactive co-segmentation with intelligent scribble guidance. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 3169–3176.

Yuri Boykov and Vladimir Kolmogorov. 2004. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 9 (2004), 1124–1137.

Yuri Boykov, Olga Veksler, and Ramin Zabih. 1998. Markov random fields with efficient approximations. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*. IEEE, 648–655.

Yuri Y Boykov and M-P Jolly. 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in ND images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, Vol. 1. IEEE, 105–112.

Derek Hoiem. 2008. MRFs and Segmentation with GraphCuts. https://tinyurl.com/y3d692lj.

Anat Levin, Alex Rav-Acha, and Dani Lischinski. 2008. Spectral matting. *IEEE transactions on pattern analysis and machine intelligence* 30, 10 (2008), 1699–1712.

Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM transactions on graphics (TOG)*, Vol. 23. ACM, 309–314.

Michael Rubinstein, Armand Joulin, Johannes Kopf, and Ce Liu. 2013. Unsupervised joint object discovery and segmentation in internet images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1939–1946.

Jianbo Shi and Jitendra Malik. 2000. Normalized cuts and image segmentation. *Departmental Papers (CIS)* (2000), 107.

Jamie Shotton, John Winn, Carsten Rother, and Antonio Criminisi. 2006. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *European conference on computer vision*. Springer, 1–15.