Interactive Image Segmentation using Graph Cuts

Mayuresh Kulkarni and Fred Nicolls

Department of Electrical Engineering
University of Cape Town, South Africa
mayuresh.kulkarni@uct.ac.za, fred.nicolls@uct.ac.za

Abstract

This paper presents an accurate interactive image segmentation tool using graph cuts and image properties. Graph cuts is a fast algorithm for performing binary segmentation, used to find the global optimum of a cost function based on the region and boundary properties of the image. The user marks certain pixels as background and foreground, and Gaussian mixture models (GMMs) for these classes are built using the colour and texture features of corresponding pixels. A likelihood ratio is used to calculate the relative probability of each pixel being foreground or background, based on the GMMs. Many features and their combinations are analyzed on images from the Berkeley Segmentation Dataset. Results of different segmentations are compared using precision-recall curves, F-score and accuracy. The average accuracy of the algorithm was 92% over a set of 20 images and the best accuracy was 99.5%.

1. Introduction

There are many methods used to segment images, from simple ones like edge detection and thresholding to complex ones like active shape models, active contour models, clustering, and graph-based segmentation. The simple methods do not need prior knowledge but have limitations, whereas more effective methods are complex and need training data.

Image segmentation based on image properties is an extensively researched problem. Interactive image segmentation is more popular than automatic segmentation because it is an easier problem. Methods based on Gaussian mixture models, Markov random fields, texture and colour-based classification [7, 12], and clustering algorithms like mean shift [5] (or a combination of these) are used to segment images.

The methods developed in this paper are based on work by Boykov and Jolly [3]. Image colour and texture properties are used to build probabilistic models of foreground and background, and a graph cut is used to globally optimize a cost function based on them. Precision-recall curves are used to compare results of segmentation from different methods.

Section 2 gives a brief summary of the previous work done using graph cuts, and other segmentation methods. A detailed description of the process used in this paper is given in Section 3. Experiments and results are explained in Section 4. Section 5 provides suggestions for future work and conclusions.

Images from the Berkeley Segmentation Dataset [9] are used. The images displayed in this paper are named 'eagle', 'plane', 'flower', 'birds', and 'grass'.

2. Related Work

This work is based on the interactive graph cuts method described in Boykov and Jolly [3]. The user marks certain pixels as "background" and "object". The image is segmented by finding the global optimum of a cost function. This cost function is based on the region and boundary properties of the image.

Let $\mathcal{P}$ be the pixels in an image and $\mathcal{N}$ be a set of all unordered pairs of neighbouring pixels. Also, let $A = (A_1, \ldots, A_p \ldots, A_{|\mathcal{P}|})$ be a binary vector where component A_p denotes background or foreground assignment to pixel p in $\mathcal{P}$. Thus *A* defines one possible segmentation out of a set of many segmentations. The cost function *E(A)* is

$$E(A) = \lambda R(A) + B(A) \tag{1}$$

where

$$R(A) = \sum_{p \epsilon \mathcal{P}} R_p(A_p) \tag{2}$$

$$B(A) = \sum_{\{p,q\} \epsilon \mathcal{N}} B_{pq}(A_p, A_q) \tag{3}$$

and

$$B_{pq}(A_p, A_q) = \begin{cases} 1 & \text{if } A_p \neq A_q, \\ 0 & \text{otherwise.} \end{cases}$$

In (1), λ is a coefficient that shows the weight given to region properties $R(A)$ with respect to the boundary properties $B(A)$. The region term $R(A)$ consists of R_p, the penalty of assigning each pixel to background or object. The $B(A)$ term describes the boundary properties of the image: B_{pq} can be interpreted as the evidence of a boundary between two neighbouring pixels p and q. Boykov and Jolly also describe a way to define $R(A)$ and $B(A)$ using variables in their algorithm. The same algorithm is used in this paper, but with $R(A)$ and $B(A)$ calculated differently. Section 3.2 describes and justifies the assignments of $R(A)$ and $B(A)$.

Graphs cuts have been used for 2-D and 3-D segmentation. In 3-D, volumetric graph cuts are used for reconstruction and segmentation of surfaces. In [8] an algorithm is presented to perform segmentation and to estimate the pose of a human body using multiple views. Although this paper covers work done in 2-D, graph cuts can be applied to 3-D.

The graph cuts algorithm is fast and effective. Instead of improving its performance, much work has been done in modifying and setting up the variables differently.

A comparison of the performance of Magic Wand, Intelligent Scissors (Live Wire), Bayes matting, Knockout 2, graph

cut, and level sets is done in [13]. GrabCut, an iterative and interactive image segmentation tool, is introduced in [13]. The main aim of GrabCut is to reduce user interaction by using mechanisms called "iterative estimation" and "incomplete labelling". GrabCut starts with the user drawing a rectangle around the desired object. Foreground is estimated using the pixel data in the rectangle. A segmentation using graph cuts is done and the user is allowed to add background, foreground or matting information to improve the segmentation. Matting information is border information that is used to recover foreground colour information, free of colour bleeding from the background. "Incomplete labelling" enables the user to only mark background pixels. There is no need to mark foreground pixels explicitly because of the rectangle marked by the user. "Iterative estimation" assigns provisional labels to some pixels (in the foreground) that can be retracted subsequently. Border matting is used to overcome the problem of blur and mixed pixels in the segmentation.

In [5], mean shift is used to cluster and segment images. The advantage of mean shift clustering is that the user doesn't have to specify the number of clusters. Graph cuts and mean shift are used in [14]. The image is mean shifted and then a graph cut is used to do the segmentation. This approach was implemented but was considered to be ineffective because graph cuts should be given as much information about the image as possible, rather than clustering and masking the information from the image.

An adaptive Gaussian mixture Markov random field (GMMRF) model is used to segment images in [2]. A "pseudo-likelihood" algorithm is formed after estimating the colour properties of the image. This "pseudo-likelihood" algorithm is based on [1]. A Graph cut is used to segment images using the output of this "pseudo-likelihood" algorithm. The pseudo-likelihood function has limitations as it lacks the complexity for useful models [2].

Gaussian mixture models (GMMs) based on colour and texture features are used in [12] to classify and segment images. In [11] GMMs are used for image retrieval from databases. Although [12, 11] do not use graph cuts, the work done using GMMs based on colour and texture is important for this paper. They use GMMs based on different colour spaces, like RGB and LAB, and texture feature filters like the discrete cosine transform and wavelets. They also compare the performance of GMMs to other statistical models like the 2-D hidden Markov model (HMM), 2-D multi-resolution hidden Markov model (MHMM) and classification and regression trees. It was seen that the average classification error rates were the lowest for GMMs. The decision of using GMMs for this work was based on these results from [12, 11].

Image segmentation is done using texture measures in [7]. These texture measures include maximum response filters, ring/wedge filters, Gabor filters and Berkeley filters.

3. Implementation

Our algorithm works as follows. The user marks certain pixels as foreground and background. GMMs for foreground and background are estimated using these marked pixels. The region properties are set using probability of each pixel being background and foreground from the GMM. The boundary properties are set using the gradient of the image. The region and boundary terms are fed into the graph cut formation for a pixel grid where each pixel is a node in the graph. The details of each step are discussed in the following sections.

3.1. Region and Boundary Properties

The algorithm described in [4] was used to implement the segmentation technique. The pixel grid is used to set up the graph. Each pixel is a node in the graph and the region and boundary properties are derived accordingly. Because the boundary properties are the evidence of an edge in the image, the log of the gradient of the image and the distance transform of the Canny edge detection are used. Experiments show that both methods work equally well. Region properties are derived from the log likelihood ratio in equations (4) and (7). A 8-pixel neighbourhood is used.

3.2. GMMs of Colour and Texture Features

Using the pixels marked by the user, GMMs are set up for foreground and background. Features like colour and texture can be added to the GMM. A Gaussian distribution has the form

$$p(x) = \frac{1}{\sqrt{2\pi\sigma}} \cdot e^{-\frac{1}{2}(x-\mu)^2}, \tag{4}$$

where μ is the mean and σ is the standard deviation. A GMM, with the feature vectors $X^i = \{x_t, 1 \leqslant t \leqslant T_i\}$ for data points, uses *M* Gaussians to model the data as follows:

$$P(X^i|\theta^i_{GMM}) = \prod_{t=1}^{T^i} \sum_{j=1}^{M} P(z_j)\, P_{z_j}(x_t|\mu_j, \Sigma_j) \tag{5}$$

where θ^i_{GMM} includes the model parameters mean, covariance and mixture weights i.e.$\{P(z_j), \mu_j, \Sigma_j, 1 \leq j \leq J\}$. $P_{z_j}(x_t|\mu_j, \Sigma_j)$ is the Gaussian distribution for the j-th class, with a mean vector μ_j and a covariance matrix Σ_j as:

$$P_{z_j}(x_t|\mu_j, \Sigma_j) = \frac{1}{(2\pi)^{D/2} \mid \Sigma_j \mid^{1/2}} e^{-\frac{1}{2}(x_t-\mu_j)^T \Sigma_j^{-1}(x_t-\mu_j)} \tag{6}$$

where D is the dimension of the feature vector x_t. Usually the covariance matrix Σ_j is set to be a diagonal matrix with the covariances of the features as its elements. This is done to reduce the size of the parameter space. The probability of each pixel being in either of the classes is calculated using GMMs.

The logarithmic likelihood ratio, derived from these probabilities, is used to set the region property terms in the graph cut formation:

$$Log\ Likelihood\ Ratio\ (llr_p) = log(\mathcal{K} \cdot\ p_f/p_b) \tag{7}$$

where llr_p is the log likelihood ratio of a pixel *p*, and p_f and p_b are the probabilities of a pixel belonging to the foreground and background respectively. The sensitivity of the segmentation depends on $\mathcal{K}$, the weight given to foreground relative to background. The probability of a pixel being foreground increases with the value of $\mathcal{K}$ and vice versa. Section 4 gives a detailed analysis of the effect of $\mathcal{K}$ on the final segmentation.

For grayscale images, the raw intensity of the pixels, Gabor and MR8 filters were used in the GMMs. For colour images, RGB values, Luv values, gabor filters and MR8 filters were used. The Netlab toolbox [10] is used to set up the GMM

(a) Original image.

(b) Log likelihood ratio image.

Figure 1: Original image and the log likelihood ratio of each pixel in the image based on GMM.

Table 1: Assignment of edge weights

edge	weight (cost)	condition
$\{p,q\}$	B_{pq}	$\{p,q\} \in \mathcal{N}$
$\{p,\mathcal{S}\}$	LLR_p	$p \in \mathcal{P},\ p \notin \mathcal{O} \cup \mathcal{B}$
	$min(LLR_p)$	$p \in \mathcal{O}$
	$max(LLR_p)$	$p \in \mathcal{B}$
$\{p,\mathcal{T}\}$	LLR_p	$p \in \mathcal{P},\ p \notin \mathcal{O} \cup \mathcal{B}$
	$max(LLR_p)$	$p \in \mathcal{O}$
	$min(LLR_p)$	$p \in \mathcal{B}$

and calculate the probabilities. Because GMMs are used in this paper, the weights of the edges are set in a different way to that of [3]. The edge weights are set as follows: where LLR is the log likelihood ratio matrix, and p and q are unordered neighbourhood pixels. $\mathcal{O}$ is "object", $\mathcal{B}$ is "background", $\mathcal{S}$ is the source terminal and $\mathcal{T}$ is the sink terminal of the graph. The boundary property term B_{pq} is calculated using the log of the gradient or the distance transform of the Canny edge detection filter.

4. Results

This section gives a detailed analysis of the effect of different GMM components on the final segmentation. Results of segmentations using various values of λ and $\mathcal{K}$ are compared using precision-recall curves, F-score and accuracy.

4.1. Defining a performance measure

Many papers written on the topic, indicate the performance of their algorithms in the form of the segmented images, as putting a numerical value to the error function is difficult. The obvious performance measure can be the number or percentage of misclassified pixels based on a ground truth image. This may not be an accurate measure as it fails to convey the information of true and false positives and negatives. Thus a more comprehensive measure of performance has to be defined.

Precision and recall measures and F-score are used in this paper to define error. They are also used in [9, 6]. The error measures used are defined as follows:

$$Precision = \frac{tp}{tp + fp}, \tag{8}$$

$$Recall = \frac{tp}{tp + fn}, \tag{9}$$

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn}, \tag{10}$$

$$Fscore = 2 \cdot \frac{Precision \times Recall}{Precision + Recall}, \tag{11}$$

where *tp* is the number of true positives, *tn* is the number of true negatives, *fp* is the number of false positives and *fn* is the number of false negatives.

The output of the segmentation is a binary vector with the same size as the image. A true positive is when output of the segmentation is 1 when ground truth is 1, a true negative is when the output of the segmentation is 0 while the ground truth is 0, a false positive is when the output of the segmentation is 1 when ground truth is 0, and a false negative is when the output of the segmentation is 0 while the ground truth is 1.

Accuracy is the ratio of misclassified pixels to the total number of pixels. Precision is low when there is significant over-segmentation. Low recall is a result of under-segmentation, and indicates failure to retrieve relevant image information. For a perfect segmentation, precision and recall will equal 1. F-score is a combination of precision and recall to provide a single measure for the system, which will be 1 for a perfect segmentation.

4.2. Parameters $\mathcal{K}$ and λ

In Equation (7), $\mathcal{K}$ is a parameter that weighs the foreground and background probabilities. It can be used to adjust the sensitivity of the segmentation. Figure 2 shows the effect of $\mathcal{K}$ on the final segmentation. As $\mathcal{K}$ increases, more pixels are classified as foreground.

(a) $\mathcal{K}$ = 0.01.

(b) $\mathcal{K}$ = 1.

Figure 2: The effect of $\mathcal{K}$ on the final segmentation (λ = 0.02).

The λ in Equation (1) is a coefficient that shows the weight given to the region properties with respect to the boundary properties. When λ is 0, the cost function in Equation (1) consists of boundary properties only. Region and boundary properties will have equal weights when λ is 1. As λ increases, the weight assigned to region properties will increase. The coefficient λ is an important parameter in the graph cut: the value that gives the best segmentation will be different for different images.

Figure 3 shows a family of precision-recall curves for different values of λ and $\mathcal{K}$. The different curves are for different values of λ as $\mathcal{K}$ varies. For the ideal segmentation, precision and recall will be 1. As K increases, it is seen that the precision and recall values vary. For a low λ value (λ = 0.01) the recall is low resulting in under-segmentation. Low precision and over-segmentation is observed when λ increases.

The sensitivity of the segmentation depends on $\mathcal{K}$. When $\mathcal{K}$ is 0, all pixels are classified as background. As $\mathcal{K}$ increases

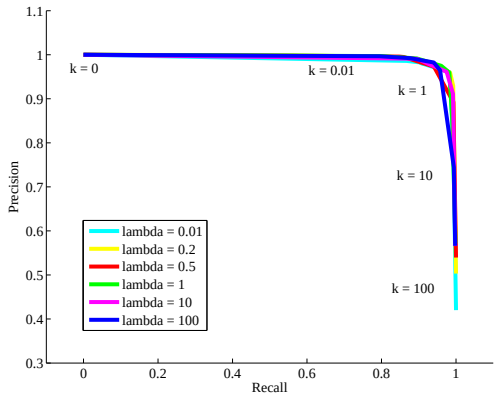


Figure 3: A family of precision-recall curves as $\mathcal{K}$ and λ vary.

the precision starts decreasing and the recall starts increasing. Figure 3 shows that the best segmentation is achieved when $\mathcal{K}$ is 1 and the background and foreground probabilities are equally weighted. Over-segmentation is observed as $\mathcal{K}$ goes above 1. This means that some background pixels are classified as foreground because of an increase in $\mathcal{K}$. This means that all the foreground pixels are in the segmentation i.e. high recall, but some background pixels are also in the segmentation i.e. low precision.

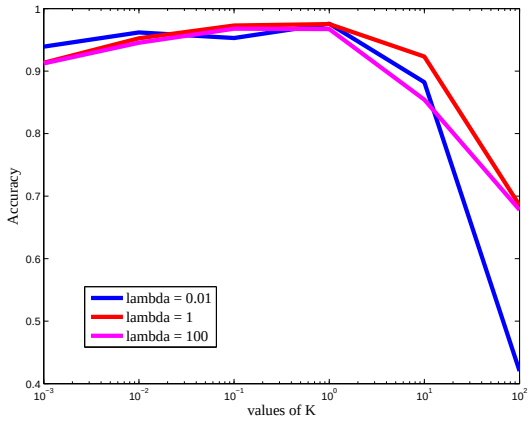


Figure 4: Accuracy versus $\mathcal{K}$. A good value for $\mathcal{K}$ is between 0.1 and 10. ($\mathcal{K}$ is plotted on a log axis.)

Accuracy is the ratio of misclassified pixels to the total number of pixels and F-score is a combination of precision and recall. Figures 4 and 5 show the effect of K on accuracy and F-score. Low and high K values result in under-segmentation and over-segmentation respectively. So accuracy and F-score will be low for very low or very high values of K. As seen in Figures 4 and 5, accuracy and F-score are highest when K is close to 1 and λ varies. High accuracy and F-score are indications of a good segmentation.

In Section 4.1, F-score is considered a better performance measure than accuracy. This is because accuracy is the percent

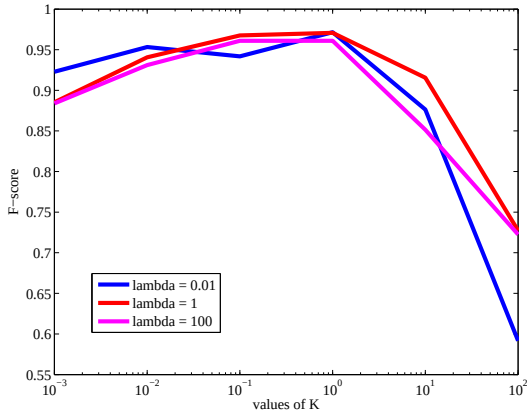


Figure 5: F-score versus $\mathcal{K}$. A good value for $\mathcal{K}$ is between 0.1 and 10. ($\mathcal{K}$ is plotted on a log axis.)

of misclassified pixels where as F-score has information about precision and recall. Figures 4 and 5 demonstrates this characteristic by showing that F-score is more sensitive than accuracy. For a good and bad segmentation, the change in F-score is more than that in accuracy.

Although the actual values of λ that give the best segmentation may differ for different images, the precision-recall curve (in Figure 3) and the F-score and accuracy curves (in Figure 4 and 5) will have the same characteristic. Experiments show that λ values between 1 and 10, and $\mathcal{K}$ values between 0.1 and 10 work well for most segmentations. Both these parameter can be varied according to the sensitivity needed for a particular segmentation.

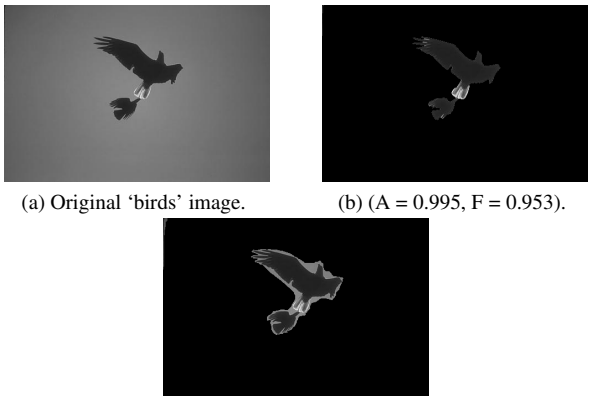
4.3. Results and Interpretation

Experiments were done on grayscale and colour images. Colour and texture measure were used to set up GMMs. The results of the segmentations using different methods were compared using the measures mentioned above. *A* and *F* stand for accuracy and F-score respectively. The performance of MR8 filters was better than Gabor filters, so Gabor filters have been excluded from this report. Experiments were done with the following features of images being selected for the GMMs:

1. Grayscale images :
   - Intenstiy values,
   - Intensity values & MR8 filters.
2. Colour images :
   - R, G, B values,
   - G, (G-R), (G-B) values,
   - L, u, v values,
   - R, G, B, L, u, v values & MR8 filters,
   - G, (G-R), (G-B), L, u, v values & MR8 filters,
   - L, u, v values & MR8 filters, and
   - G, (G-R), (G-B), L, u, v values.

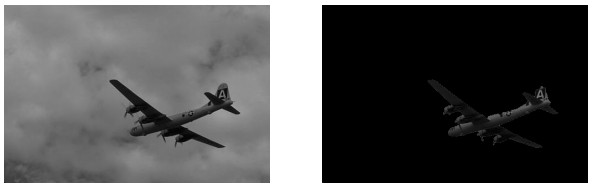
4.3.1. Grayscale images

As shown in Figure 6, the best segmentation was acquired with intensity values. MR8 filters result in over-segmentation which means that the segmentation has the object and some unwanted regions (Figure 6(c)). In the case of the 'birds' image, intensity values give the best segmentation, whereas in an image with texture content MR8 filters will be most useful.



(a) Original 'birds' image. (b) (A = 0.995, F = 0.953). (c) (A = 0.976, F = 0.803).

Figure 6: (a) Original image and segmentation using (b) intensity values and (c) intensity values and MR8 filters.



(a) Original 'plane' image. (b) (A = 0.994, F = 0.946).

Figure 7: (a) Original image and (b) segmentation using intensity values.

4.3.2. Colour Images

The algorithm works better on colour images as they have more information than grayscale images. Many different features were selected in the GMM and different features gave good results for different images.

MR8 filters lead to over-segmentation but were useful in some cases like Figure 12(d).

Luv values worked better than RGB values in images that had brightness differences. For example, the corners of Figure 10(a) are as dark as the eagle so they are classified as foreground in RGB segmentation (Figure 10(b)), but the Luv segmentation in 10(c) gives a better result.

The yellow region in Figure 8(a), which should be in the background, is classified as the object in RGB segmentation because the centers of the flowers are yellow and those values are in the GMM for "object". Combining colour spaces or MR8 filtering solves the problem and gives a better segmentation. Otherwise, any selection of features works equally well.

(a) Original 'flowers' image. (b) (A = 0.967, F = 0.962). (c) (A = 0.973, F = 0.969). (d) (A = 0.976, F = 0.972).

Figure 8: (a) Original image and segmentation using (b) RGB values, (c) Luv, G,(G-R),(G-B) values and (d) RGB, Luv and MR8 filters.

(a) Original 'plane' image. (b) (A = 0.977, F = 0.822). (c) (A = 0.995, F = 0.952). (d) (A = 0.994, F = 0.946).

Figure 9: (a) Original image and segmentation using (b) Luv and MR8 filters, (c) Luv values and (d) RGB values.

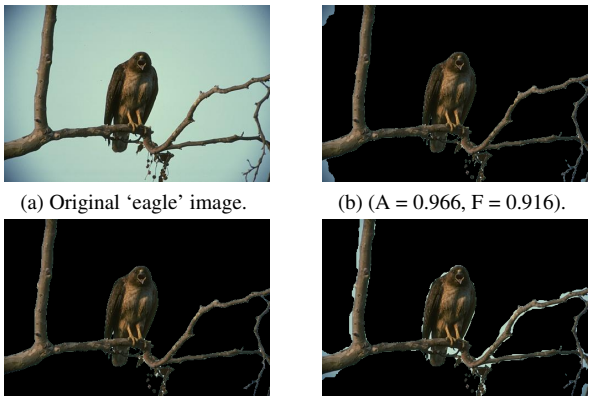
5. Conclusion and Future Work

It can be concluded that a good segmentation can be derived for any image based on colour and texture parameters, using GMMs and graph cuts. The average accuracy of the algorithm was 92% over a set of 20 images and the best accuracy was 99.5%. As shown in figures 6 to 10 the segmentations are logical, as if they were made by a human subject, and they achieve the goal of interactive segmentation within 1 iteration of the algorithm.

Features are selected carefully in this paper, but a formal study in feature selection can be done in future. Automatic feature selection that depends on the image will be effective. This will enable the algorithm to tell which features to select to give the best possible segmentation for any given image.

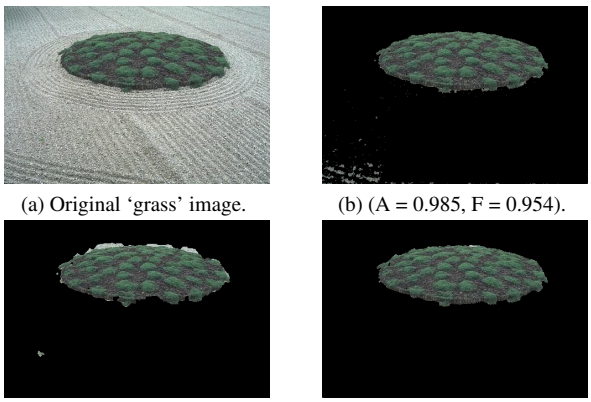
The boundary information can be extracted from the image in a different way. An explicit 'edge' model can be formed from the user-marked pixels and each pixel or regions of pixels can be tested for evidence of boundary using this model.

A different grid, or grouping of regions in the image, can



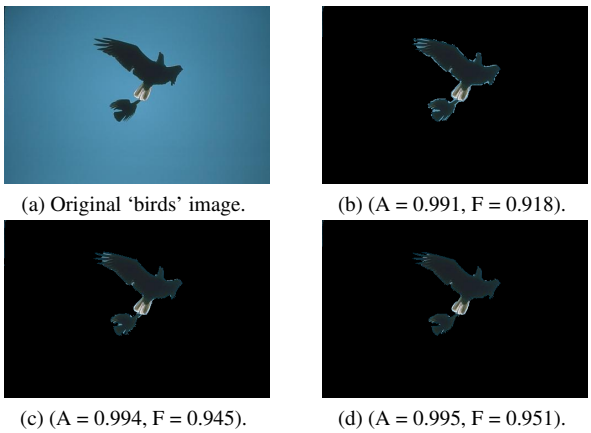
(a) Original 'eagle' image. (b) (A = 0.966, F = 0.916). (c) (A = 0.974, F = 0.934). (d) (A = 0.967, F = 0.921).

Figure 10: (a) Original image and segmentation using (b) RGB values, (c) Luv values and (d) Luv and MR8 filters.



(a) Original 'grass' image. (b) (A = 0.985, F = 0.954). (c) (A = 0.986, F = 0.956). (d) (A = 0.995, F = 0.984).

Figure 11: (a) Original image and segmentation using (b) RGB values, (c) Luv and MR8 filters and (d) G,(G-R),(G-B), Luv and MR8 filters.



(a) Original 'birds' image. (b) (A = 0.991, F = 0.918). (c) (A = 0.994, F = 0.945). (d) (A = 0.995, F = 0.951).

Figure 12: (a) Original image and segmentation using (b) Luv values, (c) RGB filters and (d) G,(G-R),(G-B), Luv filters.

be chosen instead of a pixel grid, and groups of pixels can be classified. This grid or grouping can be derived from clustering algorithms like mean shift, k-means, or nearest neighbour.

6. References

[1] Julian Besag. On the statisical analysis of dirty pictures. *Journal of the Royal Statistical Society B*, 48:259–302, 1986.

[2] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive GMMRF model. In *Lecture notes in computer science*, pages 428–441. Springer-Verlag, May 2004.

[3] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, volume 1, pages 105–112, July 2001.

[4] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004.

[5] Dorin Comaniciu and Peter Meer. Robust analysis of feature spaces: Color image segmentation. *CVPR*, pages 750–755, 1997. San Juan, Puerto Rico.

[6] Francisco J. Estrada and Allan D. Jepson. Quantitative evaluation of a novel image segmentation algorithm. In *CVPR (2)*, pages 1132–1139, 2005.

[7] Stephen Haddad. Texture measures for segmentation. Master's thesis, University of Cape Town, April 2007.

[8] Pushmeet Kohli, Jonathan Rihan, Matthieu Bray, and Philip H. S. Torr. Simultaneous segmentation and pose estimation of humans using dynamic graph cuts. *International Journal of Computer Vision*, 79(3):285–298, 2008.

[9] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[10] Ian Nabney. Netlab: Algorithms for pattern recognition. Springer, 2002.

[11] H. Permuter, J. Francos, and I. Jermyn. Gaussian mixture models of texture and colour for image database. In *ICASSP*, pages 25–88, 2003.

[12] H. Permuter, J. Francos, and I. Jermyn. A study of Gaussian mixture models of color and texture features for image classification and segmentation. In *Pattern Recognition*, volume 39 of *4*, pages 695–706, New York, USA, April 2006. Elsevier Science Inc.

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

[14] Y. Tian, T. Guan, C. Wang, L. Li, and W. Liu. Interactive foreground segmentation method using mean shift and graph cuts. *Sensor Review*, 29(2):157–162, 2009. Emerald Group Publishing Limited.