Interactive Image Segmentation

Mahesh Jagtap

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Interactive Image Segmentation

Dissertation submitted in
May 2014
to the department of
Computer Science and Engineering
of
National Institute of Technology Rourkela
in partial fulfillment of the requirements
for the degree of
Master of Technology
by
Mahesh Jagtap
(Roll no. 212CS1092)
under the supervision of
Dr.Pankaj Kumar Sa

Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela-769 008, India. `www.nitrkl.ac.in`

Dr. Pankaj Kumar Sa
Asst. Professor

Certificate

This is to certify that the work in the thesis entitled *Interactive Image Segmentation* by *Mahesh Jagtap*, bearing roll number 212CS1092, is a record of his research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Master of Technology* in *Computer Science and Engineering*.

Pankaj Kumar Sa

Declaration

I, **Mahesh Jagtap** (Roll **212CS1092**) understand that plagiarism is defined as any one or the combination of the following

1. Uncredited verbatim copying of individual sentences, paragraphs or illustrations (such as graphs, diagrams, etc.) from any source, published or unpublished, including the internet.

2. Uncredited improper paraphrasing of pages or paragraphs (changing a few words or phrases, or rearranging the original sentence order).

3. Credited verbatim copying of a major portion of a paper (or thesis chapter) without clear delineation of who did or wrote what. (Source:IEEE, the Institute, May 2014)

I have made sure that all the ideas, expressions, graphs, diagrams, etc., that are not a result of my work, are properly credited. Long phrases or sentences that had to be used verbatim from published literature have been clearly identified using quotation marks.
I affirm that no portion of my work can be considered as plagiarism and I take full responsibility if such a complaint occurs. I understand fully well that the guide of the thesis may not be in a position to check for the possibility of such incidences of plagiarism in this body of work.

Date:

Mahesh Jagtap

Master of Technology

Computer Science and Engineering

NIT Rourkela

Acknowledgment

I am thankful to various associates who have helped towards molding this thesis. At the beginning, I want to express my true thanks to Dr. Pankaj Kumar Sa for his guidance throughout my thesis work. As my supervisor, he has always swayed me to stay concentrated on accomplishing my objective. His perceptions and remarks have helped me to build the general bearing of the research. He has helped me significantly and been a source of inspiration.

I am really obligated to Dr. S.K Rath, Head-CSE, for his continuous encouragement and support. He is always ready to help with a smile. I am also thankful to all the professors of the department for their support.

My true thanks to my all friends and to everyone who has provided me with kind words and valuable feedback.

Mahesh Jagtap

Abstract

Segmentation of objects from still images has many practical applications. In the past decade, combinatorial graph cut algorithms have been successfully applied to get fairly accurate object segmentation, along with considerable reduction in the amount of user interaction required. In particular, the Grabcut algorithm has been found to provide satisfactory results for a wide variety of images. This work is an extension to the Grabcut algorithm. The Grabcut algorithm uses Gaussian mixture models to fit the color data. The number of Gaussian components used in mixture model is however fixed. We apply an unsupervised algorithm for estimating the number of Gaussian components to be used for the models. The results obtained show that segmentation accuracy is increased by estimating the Gaussian components required, prior to applying the Grabcut algorithm.

***Keywords*:** Interactive image segmentation, Gaussian mixture models, Minimum description length, Expectation maximization, Mincut/maxflow algorithm

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Image segmentation is a fundamental step in many areas of computer vision including object recognition, video surveillance, face recognition, fingerprint recognition,iris recognition, medical analysis etc. It provides additional information about the contents of an image by identifying edges and regions of similar color and texture. Although a first step in high level computer vision tasks, there are many challenges to an ideal image segmentation. Segmentation subdivides an object into its constituent regions or objects. The level of detail to which the subdivision is carried on depends on the problem being solved. That is the segmentation should stop when regionns or objects of interest have been detected. For example, if an image consists of a tree, the segmentation algorithm may either stop after detecting the entire tree or further divide the tree into trunk and leaves.

Formally, image segmentation may be defined as follows [21]

Definition 1.0.1. Let R represent entire spatial region occupied by the image. We may view image segmentation as a process that partitions R into n subregions, $R_1, R_2, R_3 ... R_n$ such that

1. $\cup_{i=1}^{n} R_i = R$
2. R_i is a connected set, $i = 1, 2...n$
3. $R_i \cap R_j = \varnothing$ for all i, j and $i \neq j$
4. $Q\left(R_i\right) =$ TRUE for $i = 1, 2, 3...n$
5. $Q\left(R_i \cup R_j\right) =$ FALSE for any adjacent regions R_i and R_j

Here $Q\left(R_k\right)$ is the logical predicate defined over the points in the set R_k. Two regions are said to be adjacent if their union forms a connected set.

Condition 1 requires that each pixelshould necessarily be in some region. Condition 2 requires that all the points in any given region should be connected in some predefined sense. Condition 3 keeps the regions disjoint. Condition 4 gives the properties that are to be satisfied by the pixels in a segmented region. Finally, condition 5 requires that two adjacent regions are different according to the predicate Q. Segementation techniques are generally classified into

1. Discontinuity based segmentation
2. Similarity based segmentation

1.1 Discontinuity based segmentation

In discontinuity-based approach, abrupt changes in the intensity level of images are used to partition the image. Derivatives can be used to find local changes in the intensity levels. First and second derivatives are suited for the purpose. Derivatives of a digital function are calculated in terms of differences. For a one-dimensional function $f(x)$, the first and second order derivatives are obtained by

Spatial filters are used to calculate first and second order derivatices of the entire image. Fig. 1.1 shows a general 3×3 spatial filter mask.

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Figure 1.1: 3×3 spatial filter mask

We compute the sum of products of the mask coefficients with the intensity values to calculate the derivatives. The response R of the mask at the center point is given by

$$R = \sum_{k=1}^{9} w_k \cdot z_k \tag{1.1}$$

where z_k is the intensity of the pixel whose spatial location corresponds to the location of the k^{th} coefficient of the mask.

The discontinuity-based segmentation is achieved through: (1) Point detection, (2) Line detection, and (3) Edge detection.

1	1	1
1	-8	1
1	1	1

Figure 1.2: Point detection mask

-1	-1	-1
2	2	2
-1	-1	-1

(a) Horizontal

2	-1	-1
-1	2	-1
-1	-1	2

(b) 45°

-1	2	-1
-1	2	-1
-1	2	-1

(c) Vertical

-1	-1	2
-1	2	-1
2	-1	-1

(d) -45°

Figure 1.3: Line detection masks

1.1.1 Point detection

Point detection is based on second derivative. Fig 1.2 shows the Laplacian mask for point detection. A point is detected at location (x, y) on which the mask is centered if the absolute values of the response of the mask at that point exceeds a specified threshold. Such points are given label 1 in the output while the other points are given a label of 0. This gives us a binary image. If g is the output image and T is the non-negative threshold, we have

$$g(x,y) = \begin{cases} 1, & \text{if } R(x,y) \geq T \\ 0, & \text{otherwise.} \end{cases} \tag{1.2}$$

where R is given by equation 1.1

1.1.2 Line detection

For line detection too, second order derivatives are used. Fig 1.3 shows the masks for lines in different directions. We filter the image independently with four masks. Let R_1, R_2, R_3 and R_4 denote the responses to the four masks. If at a given point in the image $|R_k| > |R_j|$ for all $j \neq k$, then that point is said to be more likely be associated with a line in the direction of mask k.

1.1.3 Edge detection

To find the strength and direction of the edge at location (x, y) of image f, the gradient ∇f is used. ∇f is defined as

$$\nabla f = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

−1	−1	−1
0	0	0
1	1	1

(a) Horizontal

−1	0	1
−1	0	1
−1	0	1

(b) Vertical

Figure 1.4: Prewitt masks

−1	−2	−1
0	0	0
1	2	1

(a) Horizontal

−1	0	1
−2	0	2
−1	0	1

(b) Vertical

Figure 1.5: Sobel masks

This vector points in the direction of greatest rate of change at location (x, y). The magnitude of vector ∇f is given by $M(x, y)$ where

$$M(x, y) = \sqrt{g_x^2 + g_y^2}.$$

Direction of the gradient vector is given by

$$\alpha(x, y) = \tan^{-1} \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

The angle is measured with respect to the X-axis. The direction of edge at any arbitrary point (x, y) is orthogonal to the direction $\alpha(x, y)$ of the gradient vector at that point.

A simple approximation to the partial derivatives using masks of size 3 $\times$ 3 is given by Prewitt. The Prewitt masks are shown in Fig 1.4. A slight variation of the Prewitt masks are the Sobel masks shown in Fig 1.5. The advantage of using Sobel masks is that they have better noise-suppression characteristics as compared to Prewitt masks.

A widely used algorithm for edge detection is the one proposed by Canny [13]. The algorithm involves five steps [1]

- Smoothing: The image is smoothed with the Gaussian filter. Due to this noise will not be detected as edge.
- Finding gradients: Gradients at each pixel in the smoothed image are determined by applying Sobel masks shown in Fig 1.5.
- Non-maximum suppression: The blurred edges in the image of the gradient magnitude are converted to sharp edges. This is done by preserving all local maxima in the gradient image,and deleting everything else. The steps for each pixel in the gradient image are:

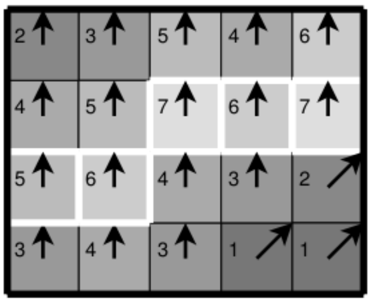


Figure 1.6: Illustration of non-maximum suppression . The edge strengths are indicated by numbers, while the gradient directions are shown as arrows. The resulting edge pixels are marked with white borders. Figure taken from [1].

1. Round the gradient direction α to nearest 45°. This corresponds to the use of 8-connected neighbourhood.
2. The edge strength of the current pixel is compared with the edge strength of the pixel in the positive and negative gradient direction.
3. If the current pixel has maximum strength, we preserve the value of the edge strength. Otherwise we remove the edge.

Fig 1.6 illustrates the non maximum suppression.

- Double thresholding: We mark the potential edge points by thresholding. The Canny edge detection algorithm uses double thresholding. Those edge pixels that are stronger than the high threshold are marked as strong; edge pixels weaker than the low threshold are suppressed and edge pixels between the two thresholds are marked as weak.
- Edge tracking: Strong edges are included in the final edges set. We include the weak edges only if they are connected to the strong edges.

1.2 Similarity based segmentation

Similarity based methods put pixels that are similar i.e those who satisfy some predefined predicate Q, in the same region. They can be classified into (1) Region growing ,(2) Region splitting and merging.

1.2.1 Region growing

For region growing we have a predefined criteria for growth. We group pixels into larger gropus if they satisfy the predefined criteria. We start with a set of seed points. To these seed points we append those pixels to the seed points that are similar to the seed points.

Let $f(x, y)$ denote an input array. $S(x, y)$ denotes a seed array containing 1 at the location of seed points and 0 elsewhere and Q denote the predicate to be applied at each location (x, y). Following steps are involved in a basic region-growing algorithm: [21]

1. Search for all connected compoments in $S(x, y)$ and erode each connected component to one pixel. All such pixels are labelled as 1 while other pixels are labelled as 0.

2. Form an image f_Q such that at a pair of coordinates (x, y), let $f_Q(x, y) = 1$ if the input image satisfies the given predicate, otherwise 0.

3. Let g be an image formed by appending to each seed point in S all the 1-valued pixels in f_Q that are 8-connected to the seed point

4. Label each connected component in g with a different region label. This is the segmented image obtained by region growing.

1.2.2 Region splitting and merging

Let R represent the entire image region and Q be the predicate. To segment R we successively divide R into smaller regions R_i so that, for any region R_i, $Q(R_i) =$ TRUE. The splitting technique has a convenient representation in the form of quad trees, that is, trees in which each node has exactly four descendants. We start with the entire region. If $Q(R) =$ FALSE, we divide the region into quadrants and so on. If no further splitting is possible the merging stage begins. In this stage those adjacent regions whose combined pixels satisfy the constraint Q are merged i.e R_j and R_k are merged iff $Q(R_j \cup R_k) =$ TRUE.

Chapter 2

Literature Review

2.1 Intelligent scissors

Intelligient scissors is one of the earlier approaches used for interactive image segmentation. Segmentation using intelligient scissors or the live-wire tool [26] requires the user to enter seed points through mouse around the object to be segmented. The intelligent scissors selects the boundary as an optimal path between the current mouse position and previously entered seed point.

Let p and q be two neighboring pixels in the image. $l(p, q)$ is the cost on the link directed from p to q. We compute the cost function as weighted sum of image features: Laplacian Zero-Crossing c_z, Gradient Direction c_d, Gradient Magnitude c_g, Inside-Pixel Value c_i and Outside-Pixel Value c_o, Edge Pixel Value c_p.

$$l(p, q) = w_z \cdot c_z(q) + w_g \cdot c_g(q) + w_d \cdot c_d(p, q) + w_p \cdot c_p(q) + w_i \cdot c_i(q) + w_o \cdot c_o(q) \tag{2.1}$$

Laplacian zero crossing c_z provides for edge localization around the object. It creates a binary cost feature. If the pixel is on the zero crossing then Laplacian component cost from all links to that pixel is zero. Also, from a pair of neighbouring pixels which have opposite signs for their Laplacians, the pixel which is closer to zero is treated as having zero-crossing at that pixel. The gradient magnitude feature c_g distinguishes betweneen strong and weak edges. A smoothness constraints is added to the boundary by gradient direction c_d. It assigns a high cost for sharp changes in gradient direction at neighboring pixels, while the cost is low if the direction of the gradient at the two pixels is similar to each other.

Continuous training is performed while the boundary detection is performed i.e. the algorithm learns the characteristics of already detected boundary and uses them to make decision about current boundary. This allows the algoritihm to select edges which are similar to the already sampled edges rather than just selecting the strong edges. Training features are updated interactively as the object boundary is being defined. The training considers pixels fron the most recently defined object boundary, which allows the algorithm to adapt to gradual changes in edge charcacteristics. The image features c_p, c_i and c_o are used for training. Edge pixel values c_p are simply the scaled source image pixel values directly beneath the portion of the object boundary used for training. The inside pixel value c_i for pixel p is sampled a distance k from p in the gradient direction and the outside pixel value is sampled at an equal distance in the opposite direction. Following values are generally set for the weight coefficients $w_z = 0.3$, $w_g = 0.3$, $w_d = 0.1$, $w_p = 0.1$, $w_i = 0.1$, and $w_o = 0.1$.

We assign weights to the edges and compute an optimal path from each pixel. This creates an optimal spanning tree. A varaint of Dijkstra's algorithm is used for the purpose. The limitation of this approach is that multiple minimal paths may exists between the current cursor position and the previous seed point which increases the amount of user interaction required to get a satisfactory result.

2.2 Graph-cut for image segmentation

Y.Boykov and M-P Jolly in [9] and [12] proposed an interactive technique for segmentation of N-dimensional image. The user specifies a set of object and background pixels which form the hard constraints on the segmentation problem i.e. a segmentation is valid only if it correctly classifies the seed points as per user input. The soft constraints are specified such that both the region and boundary properties of the image are considered. These soft constraints define a cost function. The goal is to find the global minimum of the cost function that satisfies the hard constraints. To achieve this we define the graph structure for the image in a manner that minimum graph cut corresponds to the optimal solution.

Let P be set of pixels. Let N be the neighbourhood system. N consists of all unordered pairs $\{p, q\}$ of neighboring elements in P. Let $(A_1, A_2, A_3 ... A_{|P|})$ be a binary vector. Each A_p can either be "obj" or "bkg" , specifying that the pixel p belongs to object or background respectively. The cost function is then defined as

$$E(A) = \lambda R(A) + B(A) \tag{2.2}$$

where

$$R(A) = \sum_{p \epsilon P} R_p(A_p) \tag{2.3}$$

$$B(A) = \sum_{(p,q) \epsilon N} B_{\{p,q\}} \cdot \delta(A_p, A_q) \tag{2.4}$$

and

$$\delta\left(A_p, A_q\right) = \begin{cases} 1, & \text{if} A_p \neq A_q. \\ 0, & \text{otherwise.} \end{cases} \tag{2.5}$$

$R\left(A\right)$ is the regional term and $B\left(A\right)$ is the boundary term in the cost function $E\left(A\right)$. λ represents the relative importance of the regional term and the boundary term. $R_p\left(A_p\right)$ is the penalty of assigning the pixel p to A_p where A_p can either be "obj" or "bkg" as mentioned before. Boundary term is the summation of the the edge weights between those pixels p, q such that p and q belong to different classes.

In order to calculate the regional term the object and background seeds are used. Let O and B denote the set of object and background pixels respectively. Two histograms are created, one each for object and background, from the seeds entered by the user. These histograms are used to calculate the object and background intensity distributions $Pr(I/O)$ and $Pr(I/B)$. The regional penalities are then set to the negative log-likelihoods of the probabilities.

$$R_p\left(\text{“}obj\text{”}\right) = -\ln \Pr(I_p/O) \tag{2.6}$$

$$R_p\left(\text{“}bkg\text{”}\right) = -\ln \Pr(I_p/B) \tag{2.7}$$

$B_{p,q}$ represents the penalty for discontinuity between neighboring pixels. $B_{p,q}$ must be large when the pixels are similar to each other and close to zero when the pixels are dissimilar. The penalty also decreses with increase in distance between the pixels. $B_{p,q}$ is thus given by,

$$B_{p,q} \propto exp\left(\frac{-\left(I_p - I_q\right)^2}{2\sigma^2}\right) \cdot \frac{1}{dist\left(p, q\right)} \tag{2.8}$$

This equation sets a high value for $B_{p,q}$ if $|I_p - I_q| < \sigma$ while the value is small when $|I_p - I_q| > \sigma$, where σ is the expected value of the intensity difference between neighboring pixels over the entire image.

To segment the image, graph $G\left(V, E\right)$ is created. The set V of vertices includes two types of nodes. Every pixel p belonging to P is a node in graph.In addition two more nodes, object terminal S and background terminal T are created. Therefore

$$V = P \cup \left(S \cup T\right)$$

Graph consists of edges of two types known as t-links and n-links. The edges between neighboring pixels of the image are known as n-links. The edges between each pixel and the two terminals S and T are known as t-links. Denoting the n-links by the neighbourhood set N and the t-links for pixel p by $\{p, S\}, \{p, T\}$, the set E of edges is

$$E = N \underset{p\epsilon P}{\cup} \{p, S\} \cup \{p, T\}$$

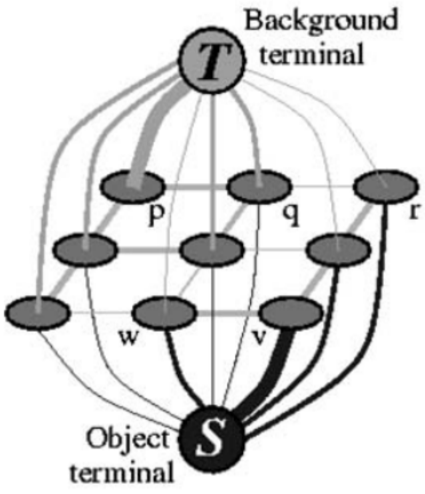


Figure 2.1: Graph structure of the image. Figure taken from [12]

. Figure 2.1 shows such a graph.

Weights are assigned to the edges in E according to the Table 2.1

Edge	Weight	Pixel description
$\{p,q\}$	$B_{p,q}$	$\{p,q\} \in N$
$\{p,S\}$	$\lambda.R_p\,(\text{“}bkg\text{”})$	$p \in P, p \notin O \cup B$
	K	$p \in O$
	0	$p \in B$
$\{p,T\}$	$\lambda.R_p\,(\text{“}obj\text{”})$	$p \in P, p \notin O \cup B$
	0	$p \in O$
	K	$p \in B$

where

$$K = 1 + \max_{p \in P} \sum_{q:\{p,q\} \in N} B_{p,q}$$

Table 2.1: Assignment of weights to the edges of the graph

The final step involves applying the maxflow/mincut algorithm. This algorithm is described in section 2.5.3

2.3 Grabcut algorithm

Rother et al. [28] extended the image segmentation technique described in [12] for color images. Their technique reduced the user interaction to drawing a rectangle around the object to be segmented. Further additions include iterative estimation and incomplete labelling, using the Gaussian Mixture Models(GMMs) for color data modelling.

The cost function, regional term and the boundary term are the same as in equation 2.2, 2.3 and 2.4 respectively. Calculations for coefficients $B_{p,q}$ and $R_p\left(A_p\right)$ are updated as described below.

The modified equation for the boundary coeffiecients $B_{p,q}$ considering color pixels is,

$$B_{p,q} = \gamma \cdot exp\left(\frac{-\left(z_p - z_q\right)^2}{2\sigma^2}\right) \cdot \frac{1}{dist\left(p,q\right)} \tag{2.9}$$

where z_p and z_q are the colors of pixels p and q respectively. The constant γ was set to 50, which was found to be a versatile setting for a wide variety of images [6].

The regional coefficients $R_p\left(A_p\right)$ are estimated using GMMs for object and background regions. A trimap is considered for the pixels . The value of the trimap for each pixel can either be TrimapObject, TrimapBackground or TrimapUnknown. User creates an initial trimap by drawing a rectangle around the object. Pixels inside the rectangle are marked as TrimapUnknown. Pixels outside of rectangle are marked as TrimapBackground. TrimapObject is initially an empty set. The TrimapUnknown pixels are then used to learn the initial object GMM while the TripmapBackground pixels are used to learn background GMM. The number of components in each GMM is set to 5.

Each pixel in the TrimapUnknown is assigned to the most likely Gaussian component in the object GMM. Similarly, each pixel in the TrimapBackground is assigned to the most likely Gaussian component in background GMM.

The GMMs are discarded and new GMMs are learned from the pixel assignments to Gaussian components done in the previous step. These new GMMs are used to calculate the regional coefficients $R_p\left(A_p\right)$. $R_p\left(\text{``}obj\text{''}\right)$ and $R_p\left(\text{``}bkg\text{''}\right)$ are the likelihoods that the pixel p belongs to the background and object GMM respectively.

$$
\begin{aligned}
R_p\left(A_p\right) = -\ln \sum_{i=1}^{K} \Bigg[&\pi\left(A_p, i\right) \cdot \frac{1}{det\Sigma\left(A_p, i\right)} \\
&\times \exp\left(\frac{1}{2}\left[z_p - \mu\left(A_p, i\right)\right]^T \Sigma^{-1}\left[z_p - \mu\left(A_p, i\right)\right]\right)\Bigg]
\end{aligned}
\tag{2.10}
$$

where $\mu\left(A_p, i\right)$, $\Sigma\left(A_p, i\right)$ and $\pi\left(A_p, i\right)$ are the mean, covariance matrix and weight of component i of object GMM if $A_p =$ "*obj*" and background GMM if $A_p =$ "*bkg*"

After calculating the boundary and regional coefficients, weights are assigned to the edges of the graph as per Table 2.1. TrimapObject and TripmapBackground represent the set O and B in the table. This is followed by the mincut/maxflow algorithm which separates the object and the background pixels.

After the initial segmentation result is displayed, the user can mark certain pixels as object pixels or background pixels, which get added to the set TrimapObject and TrimapBackground respectively, and reestimate the segmentation.

2.4 Maximum likelihood estimation

The maximum likelihood estimation is a method of estimating the parameters of a model. It selects the model parameters that maximize the agreement of the selected model with the observed data.

2.4.1 Expectation maximization algorithm

The Expectation Maximization(EM) algorithm [16] [5] allows to find the maximim likelihood solutions to problems with latent variables. Let X denote the set of all observed data, while Z represent the set of latent variables. The set of model parameters is denoted by θ. Thus the log likelihood function is given by

$$
\ln p\left(X|\theta\right) = \ln \left\{ \sum_{Z} p\left(X, Z|\theta\right) \right\} \tag{2.11}
$$

The set $\{X, Z\}$ is the complete data set while the observed set $\{X\}$ is the incomplete data set. The likelihood function for the complete data set is given by $p\left(X, Z|\,\theta\right)$. The EM algorithm makes the assumption that maximization of the complete data log likelihood is straightforward. We have the posterior distribution of the latent variables $p\left(Z|X, \theta\right)$. Since we do not have the complete data log-likelihood we consider its expected value under the posterior distribution of the latent variable. This is the E-step of the EM algorithm. In the M step we maximize this expectation. If the current estimate for the parameters is θ^{old} then a pair of E and M step gives rise to the revised estimate θ^{new}. θ_0 is the initial setting for the parameter.

In the E step, we use the current parameter values θ^{old} to find the posterior distribution of the latent variables given by $p\left(Z|X,\theta^{old}\right)$. We then use this posterior distribution to find the expectation of the complete-data log likelihood evaluated for some general parameter value θ. This expectation, denoted $Q\left(\theta,\theta^{old}\right)$, is given by

$$Q\left(\theta,\theta^{old}\right) = p\left(Z|X,\theta^{old}\right)\ln p\left(X,Z|\theta\right) \tag{2.12}$$

In the M step, we determine the revised parameter estimate θ^{new} by maximizing this function.

$$\theta^{new} = \underset{\theta}{\operatorname{argmax}}\ Q\left(\theta,\theta^{old}\right) \tag{2.13}$$

After the M step we check for convergence of either the log likelihood or the parameter values. If the convergence is not achieved we assign $\theta^{new} \leftarrow \theta^{old}$ and repeat the E and M steps.

2.4.2 EM for Gaussian Mixture Models

We now describe the EM equations obtained when it is applied to GMMs. For a D-dimensional vector x,the multivariate Gaussian distribution takes the form

$$\mathcal{N}\left(x|\mu,\Sigma\right) = \frac{1}{\left(2\pi\right)^{D/2}}\frac{1}{\left|\Sigma\right|^{1/2}}\exp\left\{-\frac{1}{2}\left(x-\mu\right)^{T}\Sigma^{-1}\left(x-\mu\right)\right\} \tag{2.14}$$

where μ is a D-dimensional mean vector, Σ is a $D\times D$ covariance matrix, and $|\Sigma|$ denotes the determinant of Σ. The K-component GMM can then be written as linear superposition of the K Gaussians in the form

$$p\left(x\right) = \sum_{k=1}^{K}\pi_k\mathcal{N}\left(x|\mu_k,\Sigma_k\right) \tag{2.15}$$

Let z be a K-dimensional binary variable such that at any time only one of $z_k = 1\,(0 \le k < K)$. Thus we have $z_k \in \{0,1\}$ and $\sum_{k=1}^{K} z_k = 1$. By using

Baye's theorem, we can calculate the conditional probability of z given x.We denote this $\gamma(z_k)$.

$$\begin{aligned}\gamma(z_k) \equiv p(z_k = 1|x) &= \frac{p(z_k = 1)\, p(x|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)\, p(x|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(X|\mu_k, \Sigma_k)}{\sum_{j=1}^{n} \pi_j \mathcal{N}(X|\mu_j, \Sigma_j)}\end{aligned} \tag{2.16}$$

$\gamma(z_k)$ is generally known as the responsibilty that component k takes for observation x.

Let N be the number of observed data samples denoted by $\{x_1, x_2, x_3 ... x_N\}$. The data set is represented by a $N \times D$ matrix X where n^{th} row is x_n^T. Z is the $N \times K$ latent variable set with rows z_n^T. Assuming that the data points are drawn independently, the log likelihood function is given by,

$$\ln p(X|\pi, \mu, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k) \right\} \tag{2.17}$$

Setting the derivative in equation 2.17 with respect to mean vector μ_k equal to zero, we get

$$0 = -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)}{\sum_{j=1}^{n} \pi_j \mathcal{N}(x_n|\mu_j, \Sigma_j)} \tag{2.18}$$

Multiplying by Σ_k^{-1}, assuming that it is non-singular, we get

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\, x_n \tag{2.19}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}) \tag{2.20}$$

Similarly, setting the derivative the derivative in equation 2.17 with respect to covariance matrix Σ_k equal to zero, we get

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})\, (x_n - \mu_k)\, (x_n - \mu_k)^T \tag{2.21}$$

While maximizing equation 2.17 with respect to π_k, we have to consider the constraint on π_k that it must sum to one $\sum_{k=1}^{K} \pi_k = 1$. Using the Lagrange's method we incorporate this constraint and get resulting equation as

$$\ln p\left(X|\pi, \mu, \Sigma\right) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) \tag{2.22}$$

Setting the derivative of above equation with respect to π_k equal to zero, we get

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}\left(x_n|\mu_k, \Sigma_k\right)}{\sum_{j=1}^{n} \pi_j \mathcal{N}\left(x_n|\mu_j, \Sigma_j\right)} + \lambda \tag{2.23}$$

Multiplying both sides by π_k and sum over k, we find that $N_k = -N$. Using this we get,

$$\pi_k = \frac{N_K}{N} \tag{2.24}$$

Thus the EM equations for Gaussian mixture models can be summarized as follows:

1. Initialize the parameters (π, μ, Σ) of the GMM and evaluate the initial value of the log likelihood.

2. E-step: Evaluate the responsibilities using the current parameter values

$$\gamma\left(z_{nk}\right) = \frac{\pi_k \mathcal{N}\left(x_n|\mu_k, \Sigma_k\right)}{\sum_{j=1}^{n} \pi_j \mathcal{N}\left(x_n|\mu_j, \Sigma_j\right)}$$

3. M-step: Re-estimate the parameters using the current responsibities

$$\mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma\left(z_{nk}\right) x_n$$

$$\Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma\left(z_{nk}\right) \left(x_n - \mu_k\right) \left(x_n - \mu_k\right)^T$$

$$\pi_k^{new} = \frac{N_K}{N}$$

where

$$N_k = \sum_{n=1}^{N} \gamma\left(z_{nk}\right)$$

4. Evaluate the log likelihood

$$\ln p\left(X|\pi, \mu, \Sigma\right) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}\left(x_n|\mu_k, \Sigma_k\right) \right\}$$

   If the change in the likelihood is greater than threshold, return to step 2.

2.5 Minimum cut algorithms

2.5.1 Flow Network [15]

Flow networks can be used to express a variety of problems. Formally, it can be defined as follows :

Definition 2.5.1. A ***flow network*** $G = (V, E)$ can be defined as a directed graph in which each edge $(u, v) \in E$ is assigned non-negative capacity $c\left(u, v\right) \geq 0$. We distinguish two vertices in a flow network, source S and target T. The number of in-coming edges for S and the number of out-going edges for T are zero. Each vertex $v \in V$ lies on some path from S to T. Such networks are also known as s-t networks.

Definition 2.5.2. We can define a ***flow*** in flow network $G\left(V, E\right)$ by a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties

- Capacity constraint: For all $u, v \in V$, we require $0 \leq f\left(u, v\right) \leq c\left(u, v\right)$
- Flow constraint: For all $u \in V - \{s, t\}$, we require

$$\sum_{v \in V} f\left(v, u\right) = \sum_{v \in V} f\left(u, v\right)$$

If $(u, v) \notin V, f\left(u, v\right) = 0$

The maximum flow problem aims to find the flow that has maximum value between two vertices S and T.

One of the important theorems related to flow networks is the maxflow/mincut theorem.

Theorem 2.5.1. ***Max-flow min-cut theorem*** *Let $N = (V, E, s, t)$ be an st-network with vertex set V and edge set E, and with distinguished vertices S and T. Then for any capacity function $c : E \to \mathbb{R}^{\geq 0}$ on the edges of N, the maximum value of an st-flow is equal to the minimum value of an st-cut.*

The max-flow min-cut theorem allows us to find the minimum cut seprating the vertices S and T in a st-network by finding the maximum flow between vertices S and T.

2.5.2 Augmenting path algorithms

Different approaches have been used to solve the maxflow problem. In this section we describe the augmenting path algorithm for getting the maximum flow.

If G is the flow network and f represents the flow then the edges in the residual network G_f represent the amount of additional flow that the edge can admit. The weight of an edge (u, v) in G_f is known as residual capacity $c_f(u, v)$ along that edge.

Definition 2.5.3. A path p from source S to target T in the residual graph G_f is known as ***augmenting path***.

The first augmenting path algorithm developed is the Ford-Fulkerson algorithm [20] published in 1956. In each iteration the algorithm finds an augmenting path and pushes the maximum possible flow through the path. The algorithm continues as long as there is an augmenting path from source S to sink T. The steps in the algorithm are:

1. Initialize flow $f(u, v) \leftarrow 0$ for all edges (u, v)
2. While there is a path p from S to T in G_f, such that $c_f(u, v) > 0$ for all edges $(u, v) \in p$
    (a) Find $c_f(p) = min\{c_f(u, v) : (u, v) \in p\}$
    (b) For each edge $(u, v) \in p$
        i. $f(u, v) \leftarrow f(u, v) + c_f(p)$
        ii. $f(v, u) \leftarrow f(v, u) - c_f(p)$

To find an augmenting path time required is $\mathcal{O}(E)$. The flow increases by an amount of atleast 1 in each iteration. Thus the runtime of Ford-Fulkerson is bounded by $\mathcal{O}(Ef)$ where E is the number of edges and f is the maximum flow in the graph, provided the capacities are integer values.

The Dinic variation of the Ford-Fulkerson algorithm extends the algorithm by specifying that the search for augmenting paths be done with a breadth first search, finding the shortest augmenting path at every step. With this method the minimum cut can be found in time $\mathcal{O}(V^2E)$ where V is the number of nodes.

2.5.3 Boykov Kolmogorov mincut/maxflow algorithm

As described above, the augmenting-path algorithm performs a breadth-first search to find an augmenting path. In computer vision problems this search involves scanning majority of the image pixels. This operation is very expensive and performing it multiple times makes the general augmenting-path algorithm slow for computer vision applications. To overcome this problem the BK maxflow algorithm [10] builds two non-overlapping search trees, one rooted at source T_{source} and other rooted at target T_{target}. In T_{source} all edges from each parent to its children are non-saturated i.e. have non-zero residual capacity, while in T_{target} edges from children to their parents are non-saturated. The nodes that are not in either of these trees are called free nodes. There are two types of nodes in the trees, active A and passive P. The nodes along the outer boundary of the tree are known as active nodes, while the internal nodes are known as passive nodes. The tree can grow along the active nodes by acquiring new free nodes along the non-saturated paths. This is not possible with passive nodes as they are surrounded by other nodes of the same tree. As the tree grows along the non-saturated paths extending from the active nodes, the tree may come in contact with a node from other tree. This gives the augmenting path. The algorithm involves three stages as described below:

1. Growth stage: The growth stage expands the search trees. Active nodes acquire new children by exploring adjacent non-saturated edges from the set of free nodes. These nodes that are acquired are now considered as the active members of the corresponding search tree. After all the neighbors of the given active node are explored the node becomes passive. Whenever the active node encounters a neighboring node that belong to the other tree, the growth stage stops. This means that a path has detected from the source to the sink.

2. Augmentation stage: In this stage, we push the maximum possible flow through the augmenting path. Due to this some of the nodes along the augmenting path will become saturated as their residual capacities become zero. These edges are then removed from the tree. This creates "orphan" nodes. At the end of augmenting stage, each tree gets divided into atleast two trees. The roots of these trees are either orphans or S or T.

3. Adoption stage: The single tree structure is restores in this stage. This is done either by finding a new parent for an orphan node or declaring it as a free node. For each orphan node all the neighbors are scanned that belong to the same tree as the orphan originally belonged to. The neighbour is made the parent of the orphan, if we find a non-saturated edge between orphan and its neighbour. If we do not find any such edge, we declare the orphan as a free node and all its children are declared as orphans. The stage terminates when there are no orphan nodes in the graph.

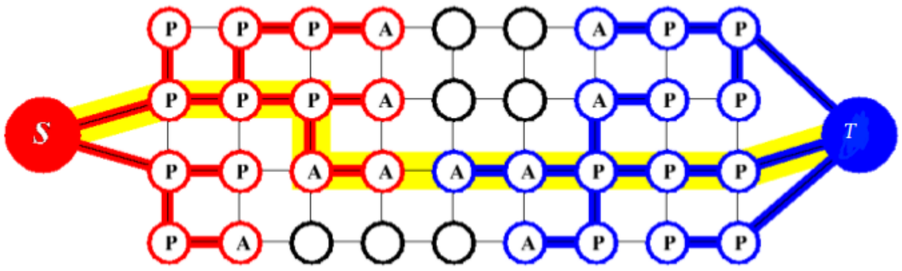


Figure 2.2: Search trees T_{source} (red nodes) and T_{target} (blue nodes) at the end of the growth stage. The augmenting path is shown by yellow line. A and P represent active and passive nodes respectively. Free nodes appear in black. Figure taken from [10]

The maximum number of augmentations possible for BK algorithm is $|C|$, i.e the cost of the minimum cut. Due to this the worst case complexity is given by $\mathcal{O}\left(V^2E\,|C|\right)$. Many standard augmenting path algorithms have less time complexity than BK algorithm. However, empirically it was shown that on problems in image processing, BK algorithm performs better as compared to others.

2.6 MDL based estimation for GMMs

Bouman et.al. [7] developed an algorithm for modelling Gaussian mixtures based on the Minimum Description Length(MDL) criteria proposed by Rissanen. The algorithm estimates number of Gaussian components required to best fit the data along with the component parameters. We use this algorithm with a slightly modified MDL criteria to account for mixtures known as the Mixture MDL criteria [19].

The EM algorithm described in section 2.4 gives the maximum likelihood (ML) estimation of observed data involving latent varibales. In order to use EM for mixture models, the number of components of the mixture should be known before hand. However for applications like image segmentation the number of components are not known. It is observed that the likelihood obtatined by the EM algorithm increases with the increase in number of components. This is because increase in the number of components results in better fitting of the data by the mixture model. However, choosing too many components, leads to the problem of over-fitting of data. The resulting mixture model is not very useful for classification purpose.

In order to overcome this problem, a penalty term is added to the likelihood which penalises higher order models. The penalty term acts as a trade-off between the likelihood of the data and the model complexity. The algorithm adds the penalty term based on the MDL principle suggested by Rissanen. Rissanen used the encoding length as a criteria for estimating the model accuracy. According to the MDL principle, the best model is the one that requires minimum number of bits to encode the data and parameters of the model.

A MDL estimate depends on the particular coding technique used. However, Rissanen developed an approximate estimate based on some assumptions which gives th MDL criteria as

$$MDL\,(K,\theta) = -\ln(X|K,\theta) + \frac{1}{2}L\ln(NM) \tag{2.25}$$

where θ is the parameter set, K is the number of components, X is the data set containing N M-dimensional elements, L is the number of bits required to specify parameter θ. For Gaussian mixtures the value of L is given by

$$L = (K-1) + K\left(M + \frac{M\,(M+1)}{2}\right) \tag{2.26}$$

In most cases where MDL is used, all data points have equal importance in estimating each component of the parameter vector. This is not the case in mixtures, where each data point has its own weight in estimating different parameters. Applying this fact, we arrive at the following MDL criteria,

$$MDL\,(K,\theta) = -\ln(X|K,\theta) + \frac{1}{2}L\ln(NM) + \frac{1}{2}L\sum_{k=1}^{K}\ln\pi_k \tag{2.27}$$

Substituting for the log-likelihood from equation 2.17, the objective is to minimize the MDL criteria given by

$$MDL\,(K,\theta) = -\sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}\left(x_n|\mu_k,\Sigma_k\right)\right\} + \frac{1}{2}L\left(\ln(NM) + \sum_{k=1}^{K}\ln\pi_k\right) \tag{2.28}$$

Minimization of this expression is not straight forward as it involves latent variables. The Expectation maximization(EM) algorithm is thus used for this purpose. We begin with maximum number of clusters K. The value of K is then sequentially decremented. The EM algorithm is applied at each value of K. This gives a local minimum of the MDL criteria for a particular K. The value of K and the corresponding parameters that gave the smallest value of the MDL criteria is then selected.

The initial component parameters are chosen to be

$$\pi_k = \frac{1}{K_0}, \text{ where } K_0 \text{ is the initial number of components} \tag{2.29}$$

$$\mu_k = x_n, \text{ where } n = \frac{(k-1)(N-1)}{(K_0-1)} + 1 \tag{2.30}$$

$$\Sigma_k = \frac{1}{N}\sum_{n=1}^{N} x_n x_n^t \tag{2.31}$$

In order to select the components to be merged we find the change in MDL criteria that occurs due to merging of the components. The upper bound on the change in the MDL criteria is given by,

$$d(l,m) = \frac{N\pi_l}{2}\log\left(\frac{|\Sigma_{(l,m)}|}{|\Sigma_l|}\right) + \frac{N\pi_m}{2}\log\left(\frac{|\Sigma_{(l,m)}|}{|\Sigma_m|}\right) \tag{2.32}$$

We search over the set of all pairs (l,m) and find the component pair which minimizes $d(l,m)$. This pair of components is then merged to reduce the number of components.

$$(l,m) = \arg\min_{(l,m)} d(l,m) \tag{2.33}$$

In order to merge two components, we constrain the parameters of two components to be equal. Thus components l and m, may be effectively merged in a single subclass by constraining their mean and covariance parameters to be equal.

$$\mu_l = \mu_m = \mu_{(l,m)} \tag{2.34}$$

$$\Sigma_l = \Sigma_m = \Sigma_{(l,m)} \tag{2.35}$$

Here $\mu_{(l,m)}$ and $\Sigma_{(l,m)}$ denote the parameters of the new subclass. Let $\theta_{(l,m)}$ denote the new parameter set formed by combining two classes. We can obtain the values of parameters $\mu_{(l,m)}$ and $\Sigma_{(l,m)}$ by considering equation 2.28 as a function of $\theta_{(l,m)}$ and maximizing it under the constraints given by equations 2.34 and 2.35. The new mean and covariance matrix obtained given by,

$$\mu(l,m) = \frac{\pi_l\mu_l + \pi_m\mu_m}{\pi_l + \pi_m} \tag{2.36}$$

$$\Sigma(l,m) = \frac{\pi_l\left(\Sigma_l + \left(\mu_l - \mu_{(l,m)}\right)\left(\mu_l - \mu_{(l,m)}\right)^t\right) + \pi_m\left(\Sigma_m + \left(\mu_m - \mu_{(l,m)}\right)\left(\mu_m - \mu_{(l,m)}\right)^t\right)}{\pi_l + \pi_m} \tag{2.37}$$

The steps og the algorithm can be summarized as follows:

1. Initialize GMM with maximum number of Gaussians, K_{max}
2. Initialize θ^1 with equations 2.29, 2.30 and 2.31
3. Apply the iterative EM algorithm to minimize the MDL criteria for current K
4. Store the parameter set θ along with the MDL value obtained.
5. If the number of Gaussians is greater than 1, resuce the number of Gaussians by applying equation 2.33 and go to step 3.
6. Select the value K and corresponding parameters θ which give minimum value of MDL over all the values of K.

2.7 Perceptual color difference

The International Commision of Illumination(CIE) has devised a distance metric called ΔE that represents the perceptual differnce between two colors in the L*a*b* colorspace. In 1976, the first color difference formula for two colors (L_1^*, a_1^*, b_1^*) and (L_2^*, a_2^*, b_2^*) was defined. It is given by ΔE_{ab}^*,

$$\Delta E_{ab}^* = \sqrt{(L_2^* - L_1^*) + (a_2^* - a_1^*) + (b_2^* - b_1^*)} \tag{2.38}$$

The 1976 definition was extended in 1994 to address perceptual non-uniformities. The color difference ΔE_{94}^* is given by,

$$\Delta E_{94}^* = \sqrt{\left(\frac{\Delta L^*}{k_L S_L}\right) + \left(\frac{\Delta C_{ab}^*}{k_C S_C}\right) + \left(\frac{\Delta H_{ab}^*}{k_H S_H}\right)} \tag{2.39}$$

where k_C and k_H are usually taken as unity and k_L is application dependent. $k_L = 1$ for graphic arts and 2 for textiles.

$$\Delta L^* = L_1^* - L_2^*$$

$$C_1^* = \sqrt{a_1^* + b_1^*}$$
$$C_2^* = \sqrt{a_2^* + b_2^*}$$
$$\Delta C_{ab}^* = C_1^* - C_2^*$$
$$\Delta a^* = a_1^* - a_2^*$$
$$\Delta b^* = b_1^* - b_2^*$$
$$\Delta H_{ab}^* = \sqrt{\Delta a^{*2} + \Delta b^{*2} - \Delta C_{ab}^{*2}}$$

$$S_L = 1$$
$$S_C = 1 + K_1 C_1^*$$
$$S_H = 1 + K_2 C_1^*$$
$$K_1 = 0.045 \text{ and } K_2 = 0.014$$

Chapter 3

GMM-MDL and Grabcut

The original Grabcut algorithm uses a fixed number of components for both the object and background GMM. However it is observed that the segmentation results vary considerably for some images with change in the number of components used to model Gaussian mixtures. Fig 3.1 shows an example image and change in segmentation output as the number of components changes.

Initially, the number of components for the object GMM and background GMM are estimated separately using the MDL principle as described in 2.6. The maximum number of components for the GMM is set to 12. Fig 3.2 shows the number of components estimated for 6 input images.

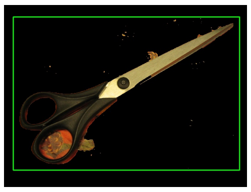
Analysing the parameters of the estimated components, it is seen that some of the estimated components have considerably low weights. Table 3.1 shows the weights of individual components for object and background GMM for image in Fig 3.2d.

$k =$	1	2	3	4	5	6	7	8	9	10
$GMM_{Obj}\ \pi_k$	0.07	0.15	0.09	0.23	0.10	0.27	0.06			
$GMM_{Bkg}\ \pi_k$	0.19	0.10	0.14	0.04	0.07	0.02	0.02	0.17	0.15	0.06

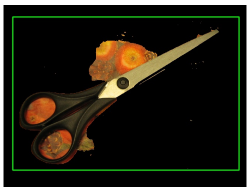
Table 3.1: Weights of individual components of GMM estimated for Fig 3.2d

The components with low weights do not have significant contribution while assigning weights to the edges of the graph in Grabcut algorithm. Increase in the number of components however increases the computation cost. To reduce the computation cost, a lower threshold value π_{TH} is set on the weight of a component in GMM. For any component k_1,if its weight π_{k_1} is found to be less than π_{TH}, k_1 is merged with the component k_2 such that the mean color of k_2 is perceptually most similar to mean color of k_1. To find the perceptual similarity between two mean colors, they are first converted

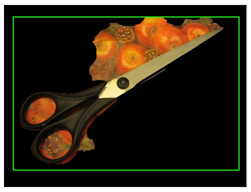
(a) Input image



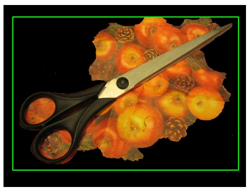
(b) K=1



(c) K=3



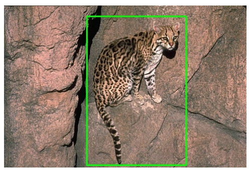
(d) K=5

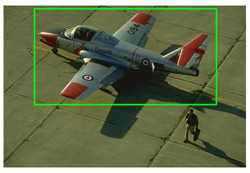


(e) K=8

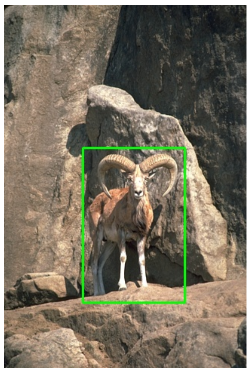
Figure 3.1: Illustration of change in segmentation results with change in number of components. Number of components in object GMM = Number of components in background GMM = K

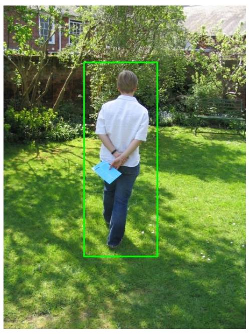
to L*a*b* color space. The difference in the two colors is then calculated by using equation 2.39 The resulting GMM component parameters are then used by the Grabcut algorithm to perform segmentation. Fig 3.3 summarizes the method used.


(a) K_{Obj}=9,K_{Bkg}=9


(b) K_{Obj}=13,K_{Bkg}=13

(c) K_{Obj}=11,K_{Bkg}=13


(d) K_{Obj}=7,K_{Bkg}=10


(e) K_{Obj}=9,K_{Bkg}=14

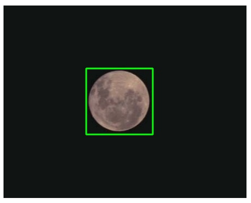

(f) K_{Obj}=6,K_{Bkg}=2

Figure 3.2: K_{Obj} and K_{Bkg} represent number of components estimated for object and background GMM respectively. K_{Obj} is estimated from the pixels inside rectangle while K_{Bkg} is estimated from pixels outside the rectangle.

User Initialization

- User draws a rectangle around the object to be segmented.

Estimate number of Gaussian components for object and background Gaussian Mixture Models

1. Using pixels inside the rectangle for object GMM and pixels outside the rectangle for background GMM, estimte the number of components for each GMM separately using MDL principle.
2. For all components k, having weight π_k less than π_{TH}, merge component k with component whose mean vector color is perceptually most similar to mean vector color of component k.

Grabcut segmentation

1. Assign each pixel p to component k which has maximum Gaussian probability for p. Pixels inside the rectangle are assigned to components of object GMM while pixels out the rectangle are assigned to components background GMM.
2. Learn GMM parameters fron the pixel assignments in the step 1
3. Constuct the graph
4. Estimate segmentation using BK maxflow/mincut algorithm
5. Repeat from step 1, until convergence

Figure 3.3: GMM-MDL and Grabcut

Chapter 4

Results and Conclusion

Images from the Berkley image segmentation database [2] and the bounding boxes provided by the authors of Grabcut algorithm [3] were used for evaluation of the segmentation algorithm. F-measure is used as a metric for comparision of results.

Fig 4.1 shows the output obtained for four input images. The F-measure comparision for 44 images is plotted in Fig 4.2. There is in increase in F-measure of 28 out of 44 (about 64%) images.

Thus we see that the performance of the Grabcut algorithm can be improved by applying MDL based estimation to Gaussian mixture models. For certain images however, the number of Gaussian components is not correctly estimated. This leads to decrease in segmentation quality. Different methods for GMM estimation can be applied and tested to make the method applicable to wider variety of images.

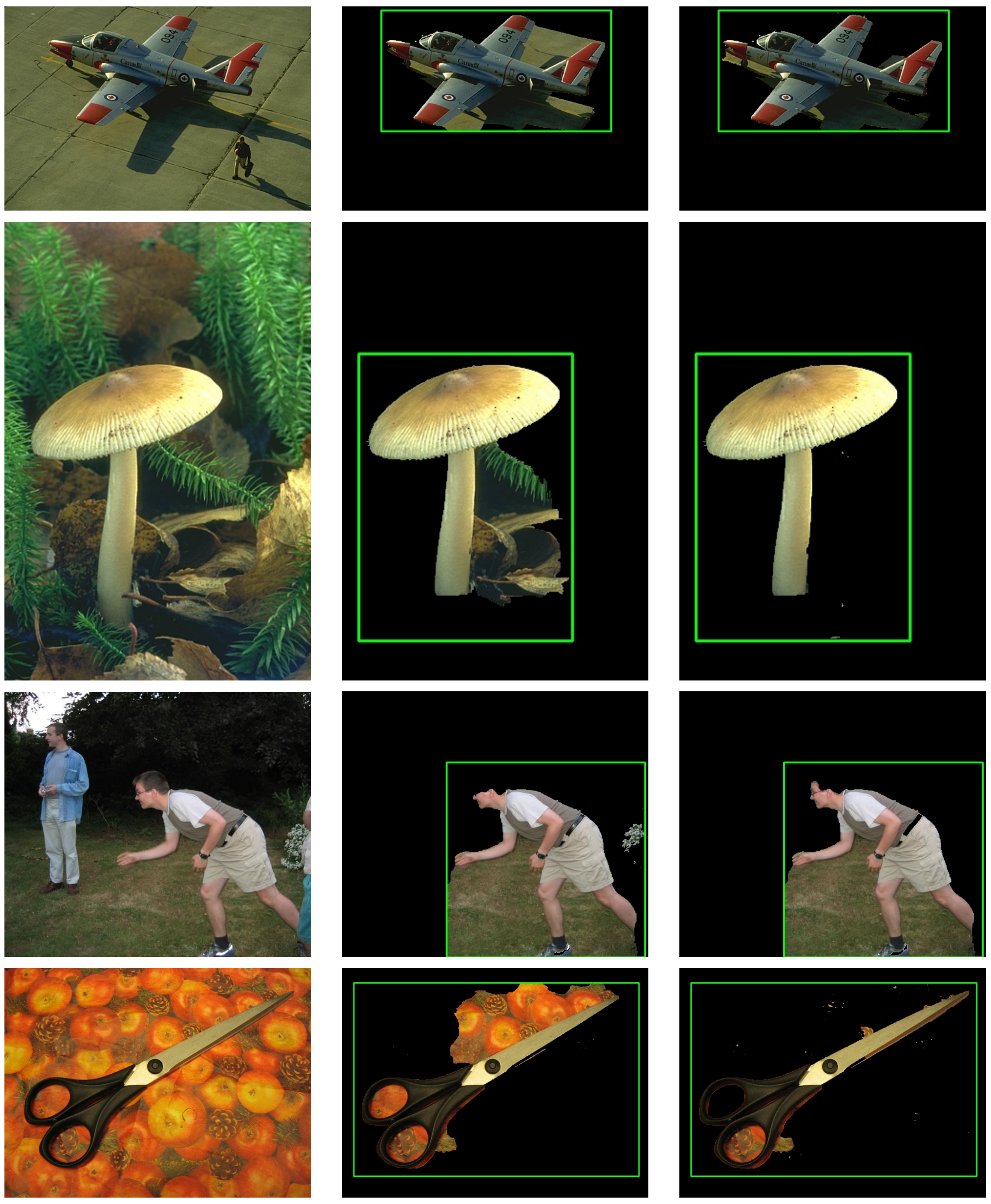


Figure 4.1: Input images are shown in column 1. Column 2 shows the output of the Grabcut algorithm while column 3 shows the output when MDL based estimation is applied to GMMs used in Grabcut algorithm.

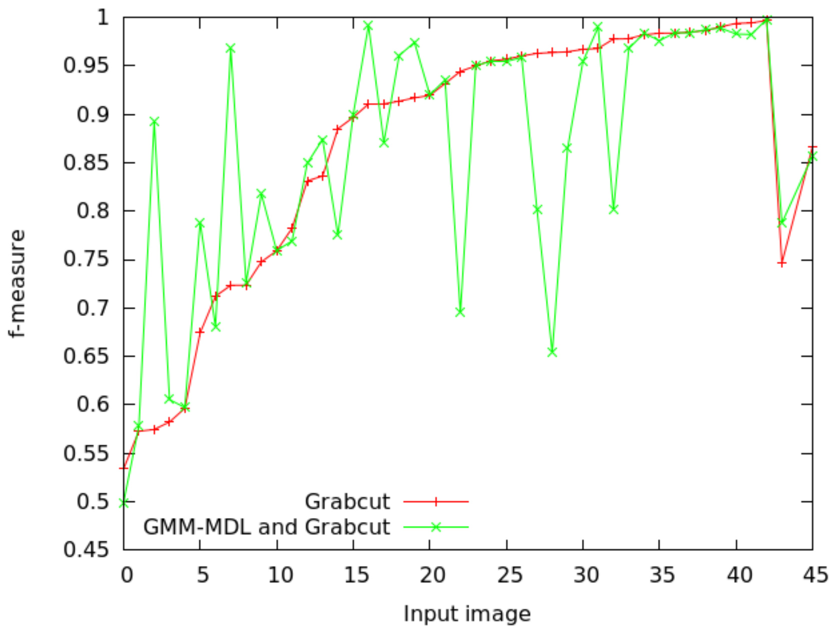


Figure 4.2: Comparision of f-measure of segmentation results

Bibliography

[1] `www.cse.iitd.ernet.in/~pkalra/csl783/canny.pdf`.

[2] `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/`.

[3] `http://research.microsoft.com/en-us/um/cambridge/projects/visionimagevideoediting/segmentation/grabcut.htm`.

[4] Andrew Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *Information Theory, IEEE Transactions on*, 44(6):2743–2760, 1998.

[5] Christopher M Bishop et al. *Pattern recognition and machine learning*, volume 1. springer New York, 2006.

[6] Andrew Blake, Carsten Rother, Matthew Brown, Patrick Perez, and Philip Torr. Interactive image segmentation using an adaptive gmmrf model. In *Computer Vision-ECCV 2004*, pages 428–441. Springer, 2004.

[7] Charles A Bouman, Michael Shapiro, GW Cook, C Brian Atkins, and Hui Cheng. Cluster: An unsupervised algorithm for modeling gaussian mixtures, 1997.

[8] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient nd image segmentation. *International Journal of Computer Vision*, 70(2):109–131, 2006.

[9] Yuri Boykov and Marie-Pierre Jolly. Interactive organ segmentation using graph cuts. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2000*, pages 276–286. Springer, 2000.

[10] Yuri Boykov and Vladimir 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, 2004.

[11] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(11):1222–1239, 2001.

[12] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 105–112. IEEE, 2001.

[13] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (6):679–698, 1986.

[14] Daniel Chen, Brenden Chen, George Mamic, Clinton Fookes, and Sridha Sridharan. Improved grabcut segmentation via gmm optimisation. In *Digital Image Computing: Techniques and Applications (DICTA), 2008*, pages 39–45. IEEE, 2008.

[15] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, Clifford Stein, et al. *Introduction to algorithms*, volume 2. MIT press Cambridge, 2001.

[16] Arthur P Dempster, Nan M Laird, Donald B Rubin, et al. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society*, 39(1):1–38, 1977.

[17] Alexandre X Falcão, Jayaram K Udupa, Supun Samarasekera, Shoba Sharma, Bruce Elliot Hirsch, and Roberto de A Lotufo. User-steered image segmentation paradigms: Live wire and live lane. *Graphical models and image processing*, 60(4):233–260, 1998.

[18] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[19] Málrio AT Figueiredo, José MN Leitão, and Anil K Jain. On fitting mixture models. In *Energy minimization methods in computer vision and pattern recognition*, pages 54–69. Springer, 1999.

[20] Lester R Ford and Delbert R Fulkerson. Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404, 1956.

[21] Rafael C Gonzalez, Richard Eugene Woods, and Steven L Eddins. *Digital image processing.* Pearson Education India, 2004.

[22] Mark H Hansen and Bin Yu. Model selection and the principle of minimum description length. *Journal of the American Statistical Association*, 96(454):746–774, 2001.

[23] Christian Hennig. Methods for merging gaussian mixture components. *Advances in data analysis and classification*, 4(1):3–34, 2010.

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

[25] 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.

[26] Eric N Mortensen and William A Barrett. Interactive segmentation with intelligent scissors. *Graphical models and image processing*, 60(5):349–384, 1998.

[27] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[28] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions on Graphics (TOG)*, volume 23, pages 309–314. ACM, 2004.

[29] Justin F Talbot and Xiaoqian Xu. Implementing grabcut. *Brigham Young University*, 2006.