ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

INTERACTIVE OBJECT EXTRACTION USING PROBABILISTIC GRAPHICAL MODELS

M.Sc. THESIS

Ali Volkan ATLI

Electronics and Communication Engineering Department

Telecommunication Engineering

SEPTEMBER 2014

ISTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

INTERACTIVE OBJECT EXTRACTION USING PROBABILISTIC GRAPHICAL MODELS

M.Sc. THESIS

Ali Volkan ATLI
(504101334)

Electronics and Communication Engineering Department

Telecommunication Engineering

Thesis Advisor: Assoc. Prof. Dr. İlker BAYRAM

SEPTEMBER 2014

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

Olasılıksal Grafik Modeller Kullanarak Etkileşimli Nesne Çıkarma

YÜKSEK LİSANS TEZİ

Ali Volkan ATLI
(504101334)

Elektronik ve Haberleşme Mühendisliği Bölümü

Telekomünikasyon Mühendisliği

Tez Danışmanı: Assoc. Prof. Dr. İlker BAYRAM

EYLÜL 2014

Ali Volkan ATLI, a M.Sc. student of ITU Institute of Science and Technology 504101334 successfully defended the thesis entitled **"INTERACTIVE OBJECT EXTRACTION USING PROBABILISTIC GRAPHICAL MODELS"**, which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. İlker BAYRAM** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Bilge GÜNSEL KALYONCU** Istanbul Technical University	.....
	Assist. Prof. Dr. Gökhan BİLGİN Yıldız Technical University	.....
		.....

Date of Submission : **16 December 2013**
Date of Defense : **20 January 2014**

There is always hope
- Aragorn

.

FOREWORD

I would like to express my sincere gratitude towards my advisor, Assoc. Prof. Dr. İlker BAYRAM who guided and supported me throughout the years. His office door was always open whenever I needed his advice.

A special thanks goes to my wife, Merve. This thesis would not have finished without her support.

September 2014 Ali Volkan ATLI

TABLE OF CONTENTS

Page

ABBREVIATIONS

G	**:** Graph
S	**:** Nodes in Graph
E	**:** Edges in Graph
N	**:** Neighborhood System
PGM	**:** Probabilistic Graphical Models
IOE	**:** Interactive Object Extraction
MRF	**:** Markov Random Field
GRF	**:** Gibbs Random Field
MAP	**:** Maximum A Posteriori
s	**:** Source
t	**:** Sink
O	**:** Object
B	**:** Background

LIST OF TABLES

Page

LIST OF FIGURES

Page

INTERACTIVE OBJECT EXTRACTION
USING PROBABILISTIC GRAPHICAL MODELS

SUMMARY

In this thesis, we present a comprehensive study of combining probability theory and graph theory (referred to as *probabilistic graphical models*) in interactive object extraction, with respect to both defining an objective function and finding the optimal solution. A two dimensional RGB image is divided into "object" and "background" regions using the graphical model. A graph is formed by connecting all pairs of neighboring image pixels to define an energy function in a context of *Maximum A Posteriori* based *Markov Random Field* estimation in terms of *likelihood* and *a priori* models. Certain pixels marked by a user indicate a *priori* identified as object or background *seeds* providing necessary clues about the image content. On the other hand, the *likelihood* is also calculated using these seeds.

Our aim is to find the globally optimal solution to cut the edges in the graph so that the object seeds are completely separated from the background seeds. The obtained solution gives the best balance of boundary and region properties satisfying the user seeds. The topology of our model is unrestricted and both object and background regions can consist of several isolated parts.

An object boundary can be anywhere but it has to separate the object seeds from the background seeds. These seeds can be loosely positioned inside the object and background regions. The framework is quite stable and normally produces the same results regardless of particular seed positioning within the same image object. In addition, the globally optimal solution can be very efficiently recomputed when the user adds or removes any seeds. This allows the user to get any desired object/background results quickly via very intuitive interactions.

We implemented an software application for Android-based smartphones in order to assess the effectiveness of the model in the real world. The application is a fast and generic implementation to perform interactive object extraction for color images. It was implemented in two parts; the user interaction (Graphical User Interface, GUI) depends on Android Software Development Kit, SKD (based on Java) and the algorithm's main implementation was written based on native C++ with standard template library because of portability. The algorithm part can be compiled in any platform without any porting effort. We tested it in Microsoft Windows 7, Ubuntu 12.04 LTS. In addition, some experimental results are presented in the related chapters. Results were obtained for different parameter values, overall time and image complexities.

Olasılıksal Grafik Modeller Kullanarak
Etkileşimli Nesne Çıkarma

ÖZET

Son yıllarda, olasılıksal grafik modeller, enerji fonksiyonlarının en küçük değerlerini bulmakta kullanılan en-iyileme yöntemleri arasında yerini almaya başlamıştır. Bu tezde, renkli imgelerde kullanıcı etkileşimiyle, olasılıksal grafik modellerin, ilgilenilen nesnenin elde edilebilmesi için, yani imgenin nesne ve arkaplan olmak üzere iki sınıflı gösteriminin elde edilmesi anlatılmaktadır. Kullanıcı etkileşimi, kullanıcının imgedeki bazı pikselleri nesne veya arkaplan olarak işaretlemesi sonucu nesneye veya arkaplana ait sınırlı ipuçlarının elde edilmesini açıklamak için kullanılmaktadır. Bu yöntemde ilgilenilen nesne(ler) birden fazla ve birbirlerinden ayrı olabilmektedir.

Kullanıcı etkileşimli nesne çıkarımı konusunu, otomatik nesne çıkarımı methodlarının hiçbir zaman tam olarak mükemmel olamayacağı temeline dayanarak ortaya çıkmıştır. Bu nedenle bu tezdeki amaç, Boykov ve Kolmogorov tarafından önerilen graf kesim yaklaşımı kullanılarak Android işletim sistemine sahip akıllı telefonlar üzerinde genel amaçlı bir nesne çıkarım uygulamasının gerçekteştirilmesidir. Bu uygulamada, kullanıcı, ilgilendiği nesnelere ve arkaplana ait az sayıda pixeli işaretlemektedir. Sonrasında uygulama, kullanıcı tarafından sağlanan nesne/arkaplana ait bu ipuçlarını kullanarak nesneleri çıkarmaktadır. Bu amaçla ilk olarak Markov Random Fields (MRF) temelli nesnenin bölgesel (regional) ve sınırlarına göre bir amaç fonksiyonu tanımlanmaktadır. Ardından, amaç fonksiyonuna ait en iyi çözüm Maximum a Posteriori (MAP) kestirim yöntemi ile bulunmaktadır. Eğer amaç fonksiyonu MRF kurallarınca doğru tanımlandıpı takdirde bulunan çözüm, tüm çözüm kümesi içerisindeki en iyi çözüm olacakdır.

İlk olarak P'yi 2 boyutlu imgemize ait pikseller, N'i de bu imgedeki tüm sıralanmamış piksel komşuluklarını $\{p,q\}$ olarak ve $X = \left(X_1, \cdots, X_p, \cdots, X_{|P|}\right)$ nesne'yi '1', arkaplanı '0' olarak tutan bir ikili imge olarak düşünülürse, ($X_p \in \{1, 0\}$), MRF-MAP temelinde amaç fonksiyonumuz;

$$E(X) = \lambda \cdot R(X) + B(X)$$

ve

$$R(X) = \sum_{p \in P} R_p(X_p)$$

$$B(X) = \sum_{\{p,\, q\} \in N} B_{\{p,\, q\}} \cdot \delta\{X_p,\, X_q\}$$

olarak tanımlanmaktadır. Formüldeki $\lambda \geq 0$ olmak üzere, bölgesel özellikler terimi olan $R(X)$ ile sınır özelliklerine ait terim $B(X)$ arasındaki görece (bağıl) önemliliği belirleyen parametredir. δ parametresi ise Kroneker delta olmak üzere ise;

$$\delta(X_p, X_q) = \begin{cases} 1 & \text{if } X_p \neq X_q \\ 0 & \text{diğer durumlarda} \end{cases}$$

Bu amaç fonksiyonuna göre, bölgesel özellik terimi $R(X_p)$, herhangi bir piksele nesne ($R_p(nesne)$)veya arkaplan ($R_p(arkaplan)$) olarak atanan sınıfa göre bireysel cezalandırma terimi olarak tanımlanmaktadır. ($R_p(\cdot)$) uygulamamızda p pikselinin nesne/background histogramına göre hesaplanmaktadır.

Sınır özellik terimi $B(X)$ ise X ikili nesne/arkaplan imgesinin sınır olarak hesaplanan yerleri ile ilgilidir. $B_{\{p,q\}} \geq 0$ katsayısı p ve q pikselleri arasındaki süreksizliği cezalandırması olarak tanımlanmış ve bu katsayı büyük olduğu durumlar p ve q piksel değerlerinin (intensity) birbirine benzer, sıfıra yakın olduğu durumda da bu piksel değerleri arasındaki farkın büyük olduğu sonucu çıkmaktadır.

Bu denklemlerin çözümü ise, en-iyileme teorisindeki maksimum akış minimum kesişim teoremlerinden (max-flow min-cut) olan Boykov ve Kolmogorov'a ait graf kesişim yöntemi ile gerçekletirilmektedir. Bu yöntemi diğer yöntemlerden ayıran en önemli özelliklerden birisi, matematiksel olarak izlenebilir bir yöntem değildir ve en kötü durum karmaşıklığı standart yöntemlerden daha kötüdür, fakat *çoğu durumda* en iyi hız ve performansa sahiptir.

Yöntem genişleme, akış-arttırma ve evlat-edinme olmak üzere üç aşamadan meydana gelmektedir. Genişleme aşaması, iki dinamik arama ağacının genişleşletilmesi ile başlar. Başlangıçta dinamik arama ağaçları sadece kullanıcı tarafından belirlenen nesne ve arkaplan düğümlerinden (node) oluşmaktadır. Sonrasında, her adımda, bu arama ağaçları çevrelerindeki herhangi bir arama ağacına ait olmayan graf düğümlerini kendilerine ekleyerek büyümeye çalışmaktadırlar. Bu aşama karşı ağaçtan bir düğüm ile karşılaşılana kadar devam etmektedir, eğer karşılaşma gerçekleşirse genişleme işlemini keserek akış-arttırma aşamasına geçilir.

Akış-arttırma, nesne ve arkaplan arasında bulunan yol üzerindeki en küçük graf kenarının bulunması ve bu değerin toplam akışa eklenmesi ile tamamlanmaktadır. Akış arttırma iki arama ağacının kesişme noktaları ile en küçük graf kenarının farklı olması durumunda iki ağaca da ait olmayan "yetim düğüm" olarak adlandırılan yeni düğümlerin ortaya çıkmasına neden olmaktadır. Evlat-edinme aşaması da bu durumda devreye girmektedir. Eğer akış-arttırma aşamasında yetim düğümler ortaya çıkmışsa, bu aşama da bu düğümlere yeni arama ağacı bulunur veya serbest bırakılır.

Akıllı telefon uygulaması, amaç fonksiyonunu grap temelli bir yaklaşım ile çözerek, en iyi çözüme ait ikili nesne/arkaplan imgesini elde etmektedir. Uygulama, kullanıcı ile etkileşimi sağlayan grafik arayüz katmanı ve ilgili algoritmaların olduğu katman olmak üzere, temel olarak iki yazılım katmanından oluşmaktadır. Arayüz katmanı, Android işletim sistemin yazılım geliştiricilere sağladığı Java tabanlı sistem fonksiyonları kullanılarak gerçeklenmiştir. Algoritma katmanı ise, uygulamanın ihtiyaç duyduğu hız ve performans nedeniyle, sadece C++ ve standard şablon kütüphaneleri kullanılarak geliştirilmiştir. Algoritma katmanı, gerekli C++ derleyicisinin olduğu herhangi bir işletim sisteminde, hiçbir değişikliğe gerek olmadan derlenebilmektedir. (Windows 7, Ubuntu 12.04 ve Android isletim sistemlerinde çalıştırılmıştır). Bu iki yazılım katmanı arasındaki bağlantı ise JNI (Java Native Interface) kullanılarak sağlanmıştır. JNI, C/C++ gibi temel programlama dillerinin Java ortamında kullanılmasını sağlayan

bir arayüzdür. Bu arayüz ile uygulamamız Java kodu ile yazılmış arayüzden doğrudan C++ fonksiyon çağrıları yapabilmektedir.

Algoritma katmanı temel olarak graf-oluşturma ve grafı çözme olmak üzere, iki dizi (threads) ve arayüz katmanından gelen belirsiz zamanlı çağrılardan oluşmaktadır. Belirsiz zamanlı çağrılar kullanıcının nesne/arkaplana ait pikselleri seçtiği zamanlardır ve algoritma katmanındaki graf oluşturma aşamasına katkısı yoktur. Dolayısıyla, graf oluşturma dizisi, kullanıcı nesne ve arkaplana ait pikselleri belirlerken grafın oluşturulmasından sorumludur, graf-çözme dizisi de grafın çözümlenmesi görevinden sorumludur. İki farklı dizi yapılmasındaki amaç, kullanıcı için gerekli zaman diliminde, kullanıcının sağladığı bilgilerden bağımsız olan hesaplamaların bitirilmesidir, böylece uygulamanın yaklaşık %40'lık zaman kazancı sağlanmasına olanak tanımaktadır. Bu amaçla tezde, bu konunun detayları anlatılmakta ve elde edilen sonuçlar yorumlanmaktadır.

1. INTRODUCTION

Partitioning an image or a video frame into mutually exclusive regions as *object* and *background*, the extraction of objects of interest, is one of the most challenging problems in computer vision and image processing. Even if object extraction algorithms have been, and are still being, widely studied, there is not any exact theory. Instead, a balanced and compromised solution is generated by emphasizing the desired properties of ideal regions, herewith each algorithm has its own set of features.

The object extraction methods can be mainly grouped into following categories; *automatic* and *interactive* (or semi-automatic). Nevertheless fully automatic methods seem never to be perfect, as compromise, interactive approaches have appealed more and more attention in the last decade and they are more mature recently, and several commercial products exist that feature advanced research solutions [1]. Unlike automatic methods, which cannot know the object of interest to the user, this methods allow a user to very quickly identify an arbitrary but particular object/background and return the object more accurately extracted. Therefore, they are today one of the most used computer vision technologies, not only for object extraction, but also for many areas.

Interactive object extraction (IOE) is different than the long-standing research topic of automatic object extraction. It needs user interaction which is commonly called *clues* or *seeds* to specify only a part of the object and/or the background. The extraction of the entire object using these clues makes the problem attractive and challenging in many respects and also reveals different problems such as what the user group is (e.g. novice or advanced users), what the user interface is and, how to measure the user involvement (e.g. total amount of interaction time, or number of user clues), etc.. Hence, there has been a lot of research topics for more than two decades, and there is also not any exact answer of what is the best interactive approach system today.

Briefly, IOE is the process of splitting an observed (possibly noisy) image (d) into mutually exclusive latent regions (x) as object and background with very little user interaction. It can be thought of as a binary labeling problem, where each pixel in the observed image is assigned to the one of the object or background regions.

One common way to find solutions for the problem is by reformulating the process as an *optimization problem*. To achieve this, an *objective function* (also called energy or cost function) is designed:

$$E \ : \ (x, \ d) \rightarrow \mathbb{R} \tag{1.1}$$

which assigns an energy $E(x, \ d)$ to each possible realizations $(x, \ d)$ of the latent regions in the solution space and the input image. Hence it guides the optimization for assessing how well (or bad) the candidate solution x fits for the input image and thus provides an intrinsic measure on the quality of the solution. Based on this framework, an optimal solution which yields the *minimum* (or maximum) energy is chosen as follows;

$$\mathbf{x}^{opt} = \underset{\mathbf{x}}{\operatorname{argmin}} \, E(x, \ d) \tag{1.2}$$

The role of the objective function $E(x, \ d)$ is regarded as the quantitative measure of the global quality of the solution in the solution space, called *modeling* and as a guide to the search for a minimal solution, called *inference* or *optimization* [2]. The overall architecture is formulated as energy minimization perspective due to various uncertainties in object and background. For notational simplicity, we will hereafter drop the symbol d from $E(x, \ d)$ and denote the objective function simply as $E(x)$ because the observed image d remains fixed.

The main difficulties in the modeling are due to the most object extraction problems in computer vision and machine learning are ill-posed and noisy. Besides, computing the perfect solution may be challenging in the optimization because the computation task is quite difficult when the objective function is non-convex in a solution space with thousands of dimensions. In such a case, it takes exponential time in theory, extremely slow in practice, when general purpose optimization techniques is used. [3]. Hence our primary concern has three main tasks: how to represent the problem, and how to define the objective function and how to find the optimal solution.

In this thesis, we have implemented an interactive object extraction application for Android smart-phones, based on probability and graph theory respect to both *modeling* and *optimization*, called *Probabilistic Graphical Models* (PGM). In particular, we focus on undirected graphs, also called *Markov Random Fields (MRF)s* in which the joint distribution can be specified as the product of potential functions over the edges of the graph in terms of boundary and region properties. An MRF is a set of random variables which have a Markov property described by an undirected graph. Besides, the *Maximum A Posteriori (MAP)* estimation is used as statistical decision on the MRF graph to formulate the objective function in terms of optimality. The objective function is the joint posterior probability of the MRF. Finally, our focus is on the binary labeling problem with very little user interaction, in which each pixel belongs to object or background in accordance with user clues.

The practical use of MRF models is after the improvement of [4] in an unpublished report and further developed in [5]. They stated the equivalence between MRFs and Gibbs distributions which points out the joint distribution of an MRF is a Gibbs distribution as simple form. This gives us a practical way of calculating prior probabilities which is useful for deriving Bayesian paradigm. The MAP-MRF framework were first introduced into vision in [6]. In the Bayesian perspective, the *a priori*, the *data likelihood* and the maximization of the posterior probability of the latent object variables correspond to the minimization of the objective function in Eq **1.2**. Briefly, we can find the maximum value of the posterior probability by minimizing the objective function (to be discussed in optimization part)

As a result, we construct a specialized graph for defining the *objective function* to divide the image into mutually exclusive regions as object and background. Thus, a user marks certain pixels, to specify these pixels have to be part of the object or the background. Thereby the user's choices provide us a few hints on what the user intends to segment [7]. We calculate the likelihood term using the seeds and the remaining pixels are added automatically to the graph satisfying the user's choices. The objective function is constructed regarding the boundary and region properties of object and background using *MAP-MRF* estimation. Thereby, the modeling is completed. Please note that, the clues provided by users are *hard constraints* and we definitely want these

clues to be satisfied. For example, if the user a pixel in the given image as part of the object, the pixel have to be part of object. The boundary and region properties are *soft constraints* which penalize the objective function dynamically to prevent the application from spending time in a space where the soft constraint are not satisfied.

In the optimization part, we use a *maximum-flow/minimum-cut* or *graph cuts* based optimization algorithm to find the maximum of the posterior distribution in the objective function provided from the graph. The first min-cut/max-flow algorithm were discovered in [8] to minimize certain type of objective functions in vision, and further developed by [9] (we refer to as *bf-maxflow* shortly throughout this thesis). The bk-maxflow belongs to the group of algorithms based on Ford-Fulkerson style "augmenting paths" [10]. The bk-maxflow has an interesting property: experimental comparison in vision problems shows the algorithm is significantly superior to the other, although the worst case complexity is worse than the complexities of the other standard algorithms theoretically,

1.1 Preliminaries

The remainder of the chapter introduces the basic necessary notions that will be used throughout this thesis. It aims to make it as self-complete as possible and it is partly excerpt from [2], [6], [4], [11], [5], [1], [12], [7], and [13], please refer to them for a larger and more in depth overview

1.1.1 Graphs

A graphical model is a probabilistic model for which a graph $\mathbf{G} = (\mathbf{S}, \mathbf{E})$ denotes the conditional dependence structure between random variables. The graph is composed of *nodes* (or sites) ($\mathbf{S}$), *edges* ($\mathbf{E}$), and *terminals*. Each node is associated with a random variable, each edge between a pair of nodes, $e_{ij} : \{i,\ j\} \in S$, encodes probabilistic interaction between the corresponding variables and, each terminal is a special node to represent user interaction, namely the *source (*$\mathbf{s}$*)* and the *sink (*$\mathbf{t}$*)*. The set of edges consists of two types of undirected edges: *n-link*s (neighborhood links) and *t-link*s (terminal links). The graph provides a compact representation for a family of joint probability distribution and user's choices. A sample graph is shown in Figure 1.1.

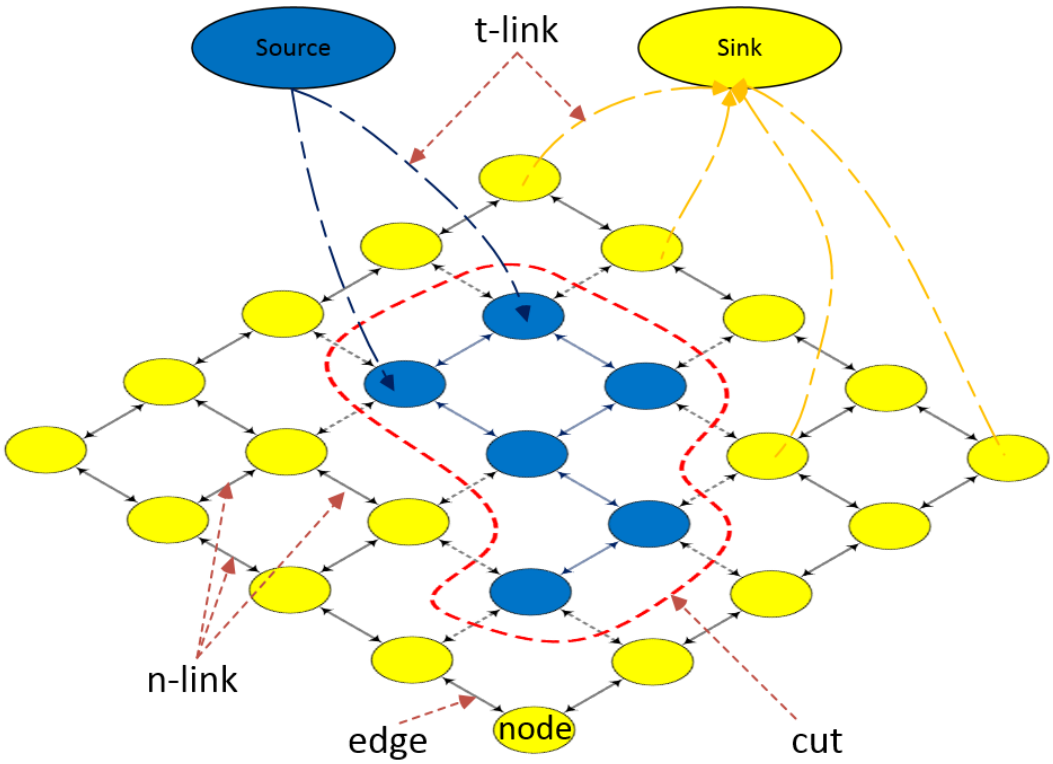


Figure 1.1: A sample graph with terminals. The set of edges consists of two types of undirected edges: n-links (neighborhood links) and t-links (terminal links).

1.1.2 Pixels and labels

We specify the object extraction problem in terms of a set of *pixels* for 2D image of size *mxn* and a set of *labels* taking two values as object or background, $\mathbf{L} = \{\mathbf{O}, \mathbf{B}\}$. Image pixels correspond to nodes of the graph and one-to-one mapping exists, so we use "node" throughout this thesis to refer to the corresponding image pixel if there is no ambiguity. The labels divide the image pixels into mutually exclusive regions which are different from the neighboring region and it has uniform and homogeneous properties. Let I be a regular lattice for the given image.

$$I = \{I(x,\ y)\} \in R^3 \mid (1 \leq x \leq m), (1 \leq y \leq n)\} \tag{1.3}$$

where the $(x,\ y)$ index pair corresponds to a pixel location at which the pixel has a color value by devoting eight bits to each of the (r)ed, (g)reen, and (b)lue components. A pixel $(x,\ y)$ is re-indexed by a single number i taking on values in $\{1,\ 2,\ \cdots,\ M\}$.

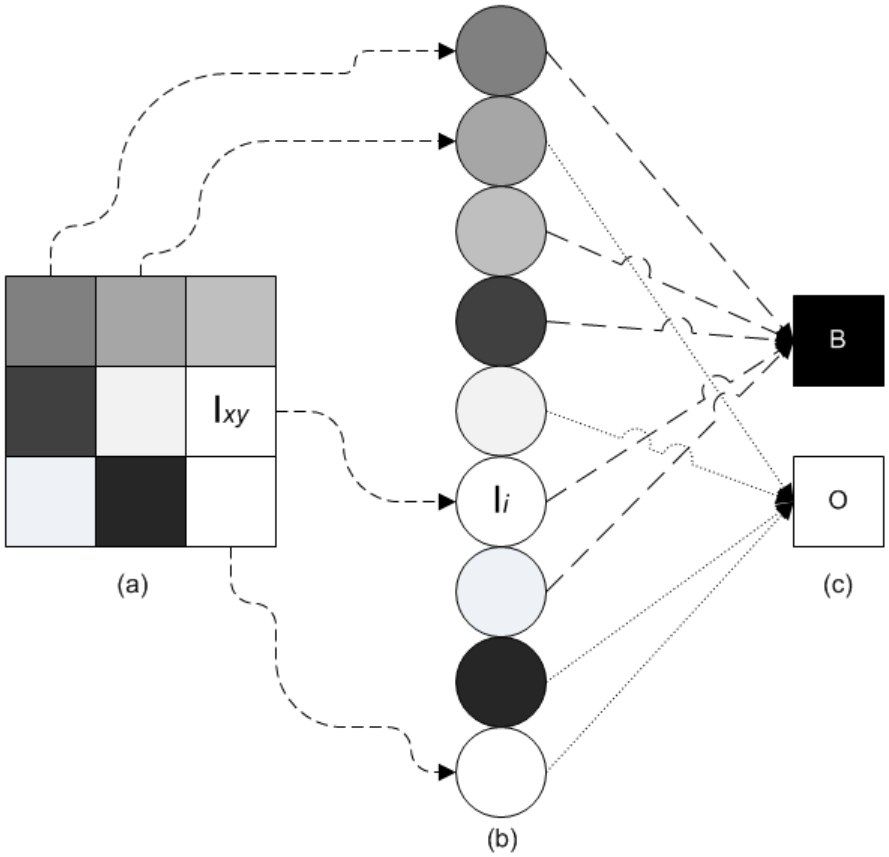


Figure 1.2: (a) Image, (b) Re-indexing for simplicity (c) Labels

This notation of single number location index is used in this thesis for images unless an elaboration is necessary to simplify formulation $(I(x,\ y) \rightarrow I(i))$.

The main goal is to assign a label from the label set to each pixel by using a labeling function f as follows;

$$f_i\ :\ I(i) \rightarrow \mathbf{L_k} \tag{1.4}$$

where the f_i is called as *labeling*, shown in figure 1.2.

For an object extraction problem of $m \times n$ pixels, there exist a total number of $2^{m \times n}$ possible candidate solutions. However, there are only a few good solutions among all solutions. For that reason, computing the globally optimal solution is an important topic in optimization and will be introduced in the following chapters.

1.1.3 Neighborhood system

In an image, each pixel is related to one another. Also one-to-one mapping exists between nodes and pixels, the nodes in S are related to one another via a neighborhood

system. Therefore if there is an edge connecting a node pair in the graph, the two nodes are called *neighbors*. The relationship for all neighbors is maintained by a *neighborhood system* **N** as follows;

$$\mathbf{N} : \{\mathbf{N}_i \mid \forall i \in S\} \tag{1.5}$$

where $\mathbf{N}_i$ denotes the set of all unordered nodes which are neighbors of the nodes $i \in S$. **N** has the following properties:

(i) A self-loop is not allowed, so a node cannot a neighbor pair to itself: $i \notin \mathbf{N}_i$

(ii) The neighboring relationship is mutual: $i \in \mathbf{N}_j \Leftrightarrow j \in \mathbf{N}_i$

According to these information, we can formulate the neighborhood system as follows;

$$\mathbf{N}_i = \{i \in S \mid dist\,(i,\ j)^2 \leq r,\ \ i \neq j\} \tag{1.6}$$

The $j \in N_i$ represents the neighbor of i and $dist\,(i,\ j)$ denotes the Euclidean distance between the nodes and r takes an integer value with respect to the order of neighborhood system. In this thesis, we use the 1^{st} order neighborhood system $(r = 1)$, (also called the 4-neighborhood system) meaning four nearest neighbor nodes; (see Fig. 1.3a), so the neighborhood system contains the following indexes; $\mathbf{N}_i = \{(x-1, y),\ (x+1, y),\ (x, y-1),\ (x, y+1)\}$. Note that nodes at or near the boundaries have fewer neighbors.

1.1.4 Cliques

A *clique* of a graph, c, is defined as a complete subgraph of the graph. It consists either of a single node $c = \{i\}$, or of a pair of neighboring nodes $c = \{i, j\}$, or of a triple of neighboring nodes $c = \{i, j, k\}$ and so on. The collections of single-site, pair-site and triple-site cliques will be denoted by C_1, C_2 and C_3 respectively, where

$$\begin{aligned} C_1 &= \{i \mid i \in S\} \\ C_2 &= \{\{i,\ j\} \mid j \in N_i,\ i \in S\} \\ C_3 &= \{\{i,\ j,\ k\} \mid i, j, k \in S \text{ are neighbors to one another}\} \end{aligned} \tag{1.7}$$

Note that the nodes in a clique are ordered, and $\{i,\ j\}$ is not the same clique as $\{j,\ i\}$, and so on. The collection of all cliques is as follows;

$$C = C_1 \cup C_2 \cup C_3 \cdots \tag{1.8}$$

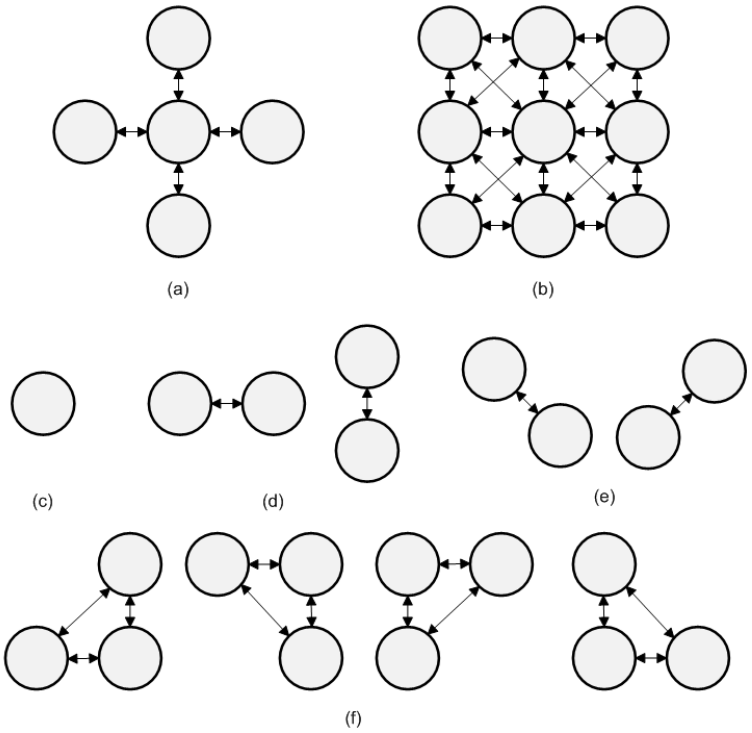


Figure 1.3: Neighbourhood and cliques on a lattice of regular sites (a) 1st order neighbourhood system, (b) 2nd order neighbourhood system and (c)-(f) cliques associated with 1st and 2nd order neighbourhood system.

where $\cdots$ denotes possible sets of larger cliques.

The type of a clique is determined by its size, shape and orientation. Fig 1.3 (c-e) show clique types for the 1st and 2nd order neighborhood systems. The single-node, horizontal and vertical pair-node cliques in (c) and (d) are all those for the 1st order neighborhood system. As the order of the neighborhood system increases, the number of cliques grow rapidly and so the involved computational expenses. That is the reason why we choose the 1st order neighborhood system in our implementation.

A *maximal clique* is a clique that cannot be extended by including one more adjacent vertex, meaning it is not a subset of a larger clique. Therefore the clique of largest possible size is referred to as a maximum clique. Since we used the 1st order neighborhood system, every node has four neighbors and the cliques shown in Fig 1.3 (c-e)

$$\mathbf{C} = \{i,\ j\} \mid j \in \mathbf{N}_i,\ (i,j) \in S \tag{1.9}$$

Note that a clique is not related to the terminal nodes, it is only related to the non-terminal node. Therefore a clique does not contain an t-link. For each maximal

clique in a graph, there is a corresponding special form potential function which takes as argument the state of clique in the graph and returns as value a non-negative real number (to be discussed later).

1.1.5 Graph cuts

Each edge e_{ij} including n-links and t-links is assigned a non-negative weight (cost) w_{ij} which is reflected by the edge's thickness in the figure 1.4 $\left(\forall e_{ij} \leftarrow w_{ij},\ \{i,\ j\} \in S\right)$. A cut is a subset of edges $C \subset \mathbf{S}$ such that the image is separated into two labels as object and background. The cost of a cut is defined as the sum of the weights of the extracted edges as follows;

$$| C | = \sum_{i \in O,\ j \in B} w_{ij} \tag{1.10}$$

The cut formalism is well suited for object extraction because a cut partitions the nodes in the graph. As illustrated the figure 1.4, this partitioning corresponds to the object extraction of an underlying image. A minimum cost cut extracts an object with a desirable balance of boundary and regional properties.

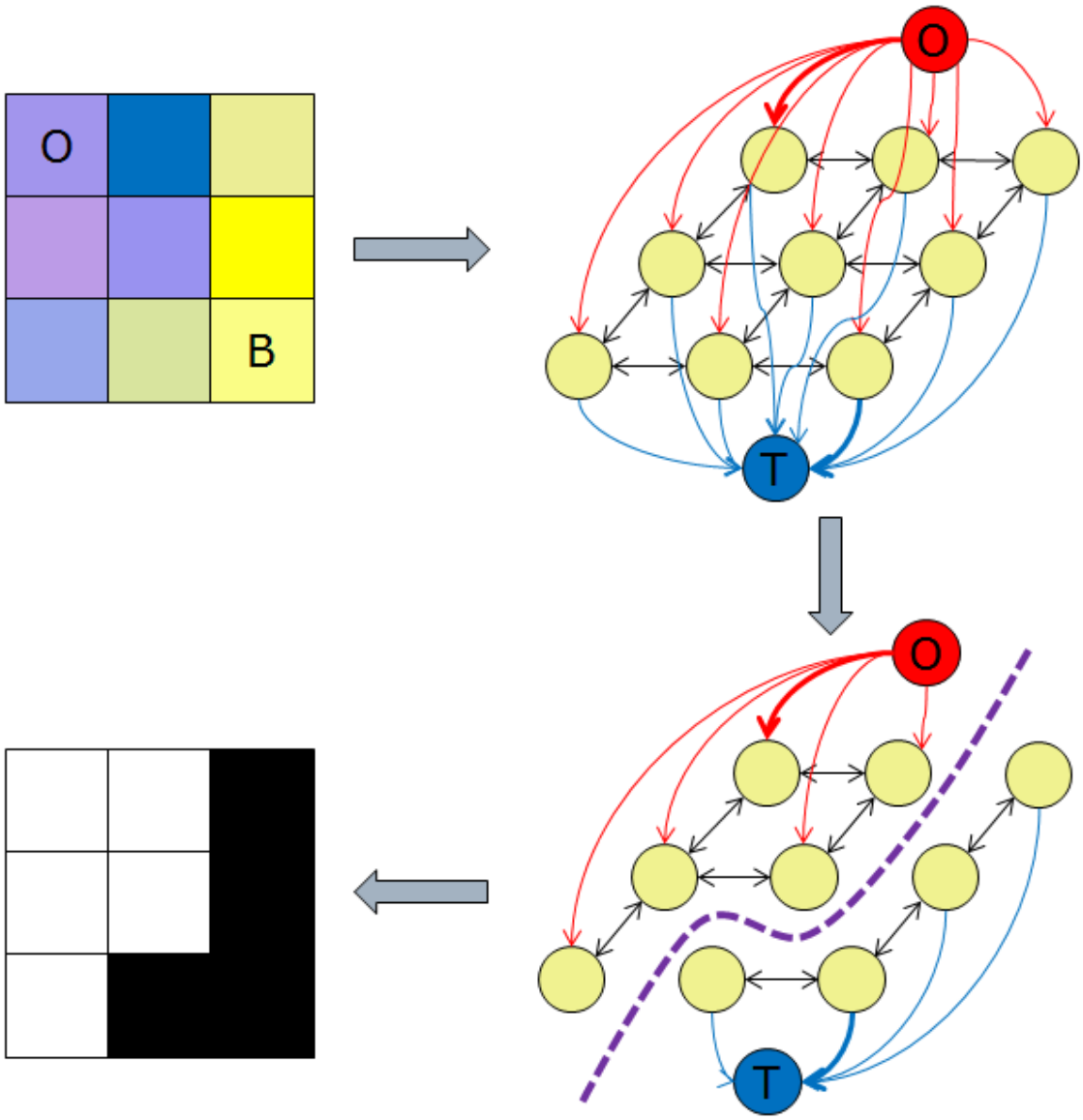


Figure 1.4: A simple 2D segmentation example of a 3x3 image. The source and sink terminals, marked by a user, are upper-left, lower-right, respectively. The cost of each edge is reflected by the edge's thickness. Initially, each node has two t-links for source and sink and each pair of neighboring nodes is connected by an n-link, but a cut separates two terminals and gives a binary labeling of the original image at the end.

2. MODELLING

2.1 Overview

As stated in the previous chapter, we use a probabilistic graphical model (shortly graphs) for *modeling* and *optimization*. In this chapter, we provide a brief but self-contained introduction to draw an analogy between images and graphs to present *modeling* stage, after that we mention Bayesian approach, based on an equivalence between MRF - *Gibbs distribution*. Last, we formulate how to use the equivalence in conjunction with *maximum a posteriori* (MAP) statistical decision and estimation criteria for leading in the next chapter of optimization.

MRFs were first introduced in [6], and they have the following useful properties [14]

- A convenient and consistent way for modeling uncertainty on images
- Images are dissected into an assembly of nodes corresponding to pixels
- Hidden variables (edges) associated with the nodes are introduced into a model designed to explain the values (colors) of all pixels
- A joint probabilistic model is built over the pixel values and the edges
- The direct statistical dependencies between edges are expressed by explicitly grouping these edges

Basically, an MRF is associated with an undirected graph of a joint probability distribution. The MRF - Gibbs equivalence provides an explicit formula for the joint probability distribution in terms of an *objective function*. The motivation behind constructing a graph is to efficiently represent the joint probability distributions over object and background segments. Each node of the graph is associated with a random variable and each neighboring pair of nodes is connected with an edge

representing a statistical dependency between neighboring pixel intensities. The statistical dependency encodes probabilistic interaction.

The practical use of MRFs is ascribed to the theorem stating the equivalence between MRFs and Gibbs distributions as established by [15] and further developed by [5]. This is because the joint distribution is required in most applications but deriving the joint distribution from conditional distributions is very difficult for MRFs [2]. The theorem sets out the joint distribution of an MRF is a Gibbs distribution which provides a simple way of specifying the joint distribution.

In this chapter, we formulate IOE problem in terms of energy minimization based on graphs. Our objective function, acquired from the MRF - Gibbs distribution theorem, serves as *soft constraint*s on regional and boundary properties in the image and the user clues on what object is intended to extract provide us *hard constraint*s. In optimization, the object and background regions are directly computed by minimizing the objective function, even if there is no prior model of what the boundary looks like or where it is located. Briefly, these regions are extracted automatically by computing a global optimum among all candidates satisfying the soft and hard constraints.

2.2 Basics of Markov Random Fields

An MRF is a probability distribution defined over a discrete set of labels L and a graph $\mathbf{G} = (\mathbf{S}, \mathbf{E})$ consisting of M nodes. For each node ($i \in \mathbf{S}$) contained in the graph, let $X = \{X_1, \cdots, X_M\}$ be a family of random variables defined on the nodes $\mathbf{S}$, in which each random variables X_i takes a value x_i in L. The family X is called a random field. We use the notation $X_i = x_i$ to denote the event that X_i takes the value x_i and the notation $(F_1 = f_1, \cdots, X_M = x_M)$ to denote the joint event. For simplicity, a joint event is abbreviated as $X = x$ where $x = \{x_1, \cdots, x_M\}$ is a configuration (or realization) of X.

As a note, we use "probability distribution" to refer to "probability mass function" and $p(x)$ to denote the probability distribution on a random variable X. Since the one-to-one mapping between a node and the associated random variable, we use "node" to refer to the corresponding random variable in cases where there is no ambiguity.

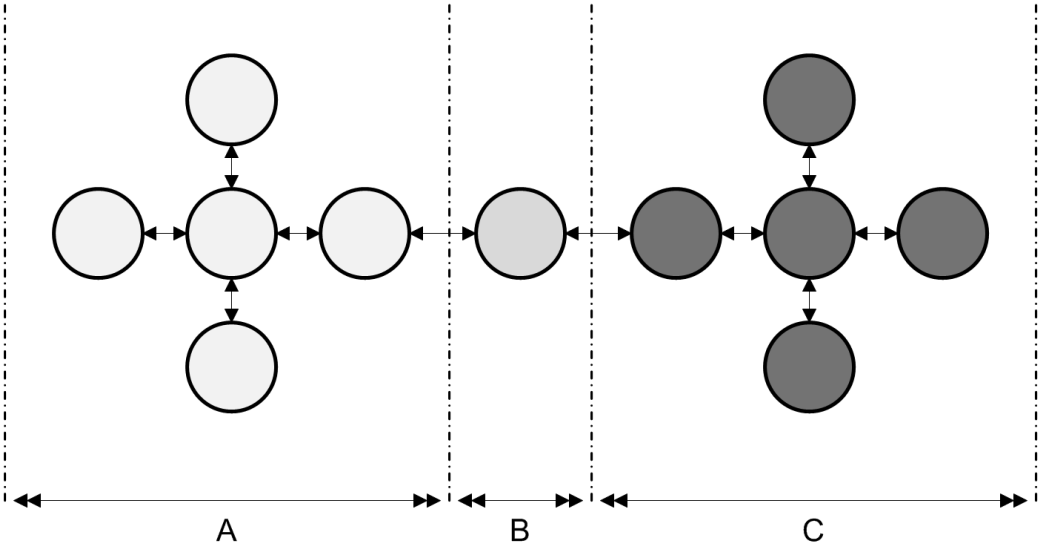


Figure 2.1: The node B separates the sets A and C due to the path from A and C necessarily passes through B. Hence, in any MRF with this graph, random variables x_A, x_C will be conditionally independent the given variable x_B

Thus an MRF defines a probability distribution by assigning to each vector x. However, not all distributions are valid in context of MRF. A valid distribution should hold the following local independence assumptions (referred to as local Markov property) which impose that each node is independent of its non-descendant nodes given all its parents.

Definition 1 *The random variable $x = \{x_i\}$ is said to be a MRF with respect to the graph G, if whenever each separation relationship "A separated from C by B" in G corresponds to a conditional independence $A \perp C \mid B$, then the random variables x_A, x_C are conditionally independent given the variable x_B;*

$$p(x_A, x_B \mid x_C) = p(x_A \mid x_C) \cdot p(x_A \mid x_B) \tag{2.1}$$

In the above definition says that an MRF is a probability distribution where the probability of a particular variable x_i depends only on a small number of its neighbors, (see 2.1)

$$p(x_i \mid x) = p\left(x_i \mid x_j\right), \quad j \in N_i \tag{2.2}$$

In the lights of this information, an MRF relies upon the following two conditions:

$$\begin{array}{ll} p(x) > 0, \ \ \forall x \in X & \text{(Positivity)} \\ p(x_i \mid x_{\mathbf{S}-\{i\}}) = p(x_i \mid x_{N_i}) & \text{(Markovianity)} \end{array} \tag{2.3}$$

where N_i represents all neighboring nodes of i in G, while $\mathbf{S} - \{i\}$ denotes all nodes of the graph G except for i.

The Markovianity indicates that each node is independent from all other nodes given its neighbors which inherit from the local independence assumptions. It depicts the local characteristics of X in which only neighboring nodes have direct interactions with each other. In fact, any nodes that are nearby, not just adjacent, are likely to have the same label, but also all pixels have correlations, explicitly linking labels. Therefore, if the positivity condition is satisfied, Markovianity tell us each random variable X_i depends on just its own neighbors, so it is characterized by its local property. In other words, it tells the joint distribution of X is determined entirely by the local conditional distributions. Given its neighborhood, a variable is independent on the rest.

2.3 Gibbs Random Fields

The Markovianity tells us that the conditional probabilities $p(x_i \mid x)$ can be used to generate a prior model $p(x)$. The joint distribution of X is determined entirely by the local conditional distributions $p(x_i \mid x_{N_i})$. But it is not clear how to actually construct the "global" joint distribution ($p(x)$) from these local functions. Fortunately, there exists a simple way of specifying a probability distribution for which the conditional probabilities are Markovian. As shown by [6], a Gibbs distribution of the form can be used as follow;

Definition 1 *The random variable $x = \{x_i\}$ is said to be a Gibbs random field with underlying graph G, if and only if its configuration obey a Gibbs distribution. A Gibbs distribution has the following form*

$$p(x) = \frac{1}{Z} \times exp(-E(x)) \tag{2.4}$$

where Z is a just normalization constant so that the sum of all probabilities is equal to unity.

$$Z = \sum_{x \in \chi^M} exp(-E(x)) \tag{2.5}$$

where χ^M is the all possible configuration space. The energy function $E(x)$ is written as a sum of local clique energies

$$E(x) = \sum_{c \in C} V_c(x) \tag{2.6}$$

is a sum of clique potentials $V_c(x)$ over all possible cliques C. The value of $V_c(x)$ is a real function that depends only on a few neighboring nodes. Thus, to build up our conditional probabilities, we use a linear summation of simple energy terms. These local energies are used as a set of soft constraints that penalize unlikely configurations of our prior model. Obviously, when the sum of all clique potentials for a certain configuration x is small, the probability distribution $p(x)$ will be high.

The key result of MRF is the theorem by [15] which states that *X is an MRF on G with respect to N if and only if X is a GRF on G with respect to N*. Therefore, due to we use 1^{st} order MRF in this thesis, we can restrict our attention to MRF's with pairs of neighboring pixels (which is an important special case) as follows; . When only cliques of size up to two are considered, the energy function can be written as;

$$p(x) \propto exp\left(\sum_{i \in V} \sum_{j \in N_i} V_{ij}(x_i, x_j)\right) \tag{2.7}$$

In conclusion, an MRF is characterized by its local property (the Markovianity) whereas a GRF is characterized by its global property (the Gibbs distribution). The Hammersley-Clifford theorem states that the joint probability distribution of any MRF can be written as a Gibbs distribution. That is to say, Hammersley-Clifford establishes the equality of the MRF and Gibbs models. This solves the problem of how to specify the joint distribution of an MRF in terms of local functions: it can now be specified by defining the potential on every maximal clique.

2.4 Maximum A Posteriori Estimation

A Bayesian model is a statistical description of an estimation problem that consist of two separate components: *prior* and *likelihood* distributions. These probabilistic distributions is combined to obtain a *posterior* distribution as a probabilistic description of the current estimate of x given the image data d. We use Bayes' rule

to compute the posterior;

$$p(x \mid d) = \frac{p(d \mid x)\ p(x)}{p(d)} \tag{2.8}$$

where $p(x)$ and $p(x \mid d)$ are the prior distribution and the likelihood distribution of x for d fixed (also called the conditional distribution of the observed image data), respectively, and $p(d)$ is the density of d which is a constant when d is given;

$$p(d) = \sum_{x} p(d \mid x) \tag{2.9}$$

Assuming a generative model for image data g given a state of configuration x as

$$g(x) = H(x) + n \tag{2.10}$$

where H is a linear or non-linear operator and n is a zero mean Gaussian i.i.d. noise. Since the noise at each pixel is independent, we have;

$$p(d \mid x) = \prod_{i \in S} p(d_i \mid x_i) \tag{2.11}$$

Without loss of generality, $p(d \mid x)$ can be obtained from the generative model in **Eq. 2.11** as

$$p(d \mid x) = exp\left(-\sum_{i \in V} V_i(x_i)\right) \tag{2.12}$$

Now we can derive the posterior distribution using Bayes' rule. From **Eq. 2.12** and **2.7**; we have

$$p(x \mid d) = \frac{p(d \mid x)\ p(x)}{p(d)} = \frac{1}{Z} \times exp\left(-\sum_{i \in V} V_i(x_i) - \sum_{i \in V} \sum_{j \in N_i} V_{ij}(x_i, x_j)\right) \tag{2.13}$$

Therefore, we finally get the optimal solution;

$$\mathbf{x}^{opt} = \underset{\mathbf{x} \in \chi}{\operatorname{argmax}}\ exp(-F(x)) \tag{2.14}$$

where

$$F(x) = \sum_{i \in S} V_i(x_i) + \sum_{i \in S} \sum_{j \in N_i} V_{ij}(x_i, x_j) \tag{2.15}$$

In order to compute the MAP estimate, we only need to minimize $F(x)$

2.5 Graph Cut Framework

The graph cut framework was first outlined by [7]. The object extraction problem is reformulated as a discrete binary labeling problem. Given an input image d, let us identify the set of nodes S with the set of image pixels. $d(i)$ represent the intensity of pixel or node i. In addition, the neighborhood system N contain all unordered pairs $\{i,\ j\}$ of neighboring nodes and there are only two labels – one for the background and one for the foreground, $L = \{O, B\}$. This setting corresponds to the object extraction problem where each of the image pixels is to be assigned exactly one of the labels depending on whether we classify it as a background or a object.

In object extraction problem, the **Eq. 2.15** type energy has proven to be sufficiently robust and is the most common in image processing. In order to provide better understanding, we re-formulate it as follows;

$$F(x) = \underbrace{\lambda \cdot \sum_{i \in \mathbf{S}} V_i(x_i)}_{\text{data term}} + \underbrace{\sum_{\{i,\, j\} \in \mathbf{N}} V_{ij}\left(x_i,\, x_j\right)}_{\text{prior term}} \tag{2.16}$$

where the data term expresses the dependence of the solution on the input data, the prior term serves as a regularization (or stabilization) evaluating constraints given by our a priori knowledge on the expected solutions to the problem and the coefficient λ controls the trade-off between the data and prior terms. Here, mapping $x = \{x_1, \cdots, x_M\}$ is the unknown labeling assigning each node $i \in S$ label $x_i \in L$. The data term $V_i(x) : L \to R$ encode how much the assignment of label x_i to node i disagrees with the observed image data at that node, it is a kind of *label penalty*. The prior term $V_{ij}(x) : L\ \ x\, L \to R$ is used for encoding the *interaction potential*, i.e., the contextual constraints that are imposed on the labels of neighboring nodes. Based on the total cost $F(x)$, the goal is to find a labeling with the minimum total cost.

Boykov and Jolly [7] proposed to solve this problem in the graph cut framework using the objective function introduced in **Eq. 2.16** The data energy terms were chosen as negative log-likelihoods of the expected intensity distribution in the labels to impose individual penalties for assigning a pixel as either *object* or *background*. The term is

given by

$$V_i(x_i) = -log\left(p\left(d(i) \mid \mathbf{H}_k\right)\right) \quad k \in \{\mathbf{O}, \mathbf{B}\} \tag{2.17}$$

where $p(\cdot)$ evaluates the conditional probability of intensity $d(i)$ and $\mathbf{H}_k$ is the RGB intensity value distribution for the label k. The probability of a pixel belonging to a particular label is proportional of the likelihood which is calculated from the color histogram of the pixel associated to the related label. It can be seen that the preferred label for each pixel is that with the higher probability because it leads to a smaller energy penalty. Further, the prior term $V_{ij}(x_i, x_j)$ between a pair $\{i, j\}$ of neighbor nodes is defined as a cost term to represent penalties for discontinuity for the pair. It penalizes the variation of the states between the two nodes, and takes the following form of a Generalized Potts model;

$$V_{ij}(x_i,\, x_j) = B_{ij}\, \left(1 - \delta(x_i, x_j)\right) \tag{2.18}$$

where $\delta(\cdot)$ is Kronecker delta which is defined as follows;

$$\delta(x_i,\, x_j) = \begin{cases} 1 & \text{if } x_i \neq x_j \\ 0 & \text{otherwise} \end{cases} \tag{2.19}$$

and B_{ij} is a penalty that is assigned to inconsistently labeled neighboring nodes i and j. In particular:

$$B_{ij} \approx exp\left(-\frac{\left(d\left(i\right) - d\left(j\right)\right)^2}{2\sigma^2}\right) \cdot \frac{1}{distance\left(i,\, j\right)} \tag{2.20}$$

where σ is a parameter, and based on this function, if a discontinuity between neighboring pixels' intensities is bigger than σ, $V_{ij}\left(x_i, x_j\right)$ penalizes a few for overall energy function. However, if neighboring pixels' intensities are similar, $V_{ij}\left(x_i, x_j\right)$ penalizes a lot for overall energy function. In particular, B_{ij} is 1 for zero contrast and declines to 0 as the contrast approaches infinity. It is also weighted by the distance of the two sites.

The energy of a solution is always a linear combination of the data term V_i and the prior term V_{ij} so both factors have to be taken into considerations. Based on the data term, the labels will tend to agree with the expected intensity distributions. At the same time, the discontinuities in the labels are preferably placed at pixels that are separated

Table 2.1: Assigning the weights of edges

edge	weights	for
$e_{\{ij\}},\ \{i,j\} \in n\text{-}edge$	$V_{ij}\left(x_i, x_j\right)$	$\{i,j\} \in N$
$e_{\{is\}},\ \{i,S\} \in t\text{-}edge$	$\lambda \cdot V_i(\text{"}bkg\text{"})$	$i \in S, i \notin O \bigcup B$
	∞	$i \in O$
	0	$i \in B$
$e_{\{it\}},\ \{i,T\} \in t\text{-}edge$	$\lambda \cdot V_i(\text{"}obj\text{"})$	$i \in S, i \notin O \bigcup B$
	0	$i \in O$
	∞	$i \in B$

by high contrast because assigning different labels to neighboring pixels with similar intensity is highly penalized in the prior term. Due to the fact that high contrast is typically encountered along edges, the object boundary (i.e., the boundary between the labels 0 and 1) will also tend to follow the edges in the image. The σ parameter controls how high the contrast should be in order to be perceived as a significant edge.

The next step is to determine the *n-edge* and *t-edge*' costs. Each pixel node has two *t-edge*s connecting to the each terminal and several *n-edge* connecting to the neighborings nodes. (Note: if a pixel is at a corner of the given image, it has two n-edges, if it is at border of the image, it has three n-edges and otherwise it has four n-edges). The table 2.1 gives us how to assign these edges;

Visual intuition on the graph construction for a 2D image is illustrated in Fig. 2.2. Finding a minimum st-cut in the graph yields a globally optimal object extraction. The energy of this labeling is equal to the cost of the cut.

The energy penalties acting in V_i and V_{ij} are soft constraints. However, the great benefit of the graph cut framework is that it allows natural and elegant integration of hard constraints as well. By deliberately choosing $e_{si} = \infty$ we can force the minimum cut to severe the other t-link at pixel i because otherwise it would have infinite cost and it would not be minimum. This means that pixel i will always be assigned label 0 irrespective of the rest of the energy function. Analogously, this can be done for the label 1, i.e., a priori classification of selected pixels can be easily hard-wired into the graph. This mechanism greatly facilitates user-friendly interactive segmentation. As proposed by Boykov [8] the user can select foreground and background seeds

$\{i,j\}$	$B_{\{i,j\}}$	$\{i,j\} \in N$
$\{i,S\}$	$\lambda \cdot R_i(\text{"bkg"})$	$i \in S,\ i \notin O \cup B$
	∞	$i \in O$
	0	$i \in B$
$\{i,T\}$	$\lambda \cdot R_i(\text{"obj"})$	$i \in S,\ i \notin O \cup B$
	0	$i \in O$
	∞	$i \in B$

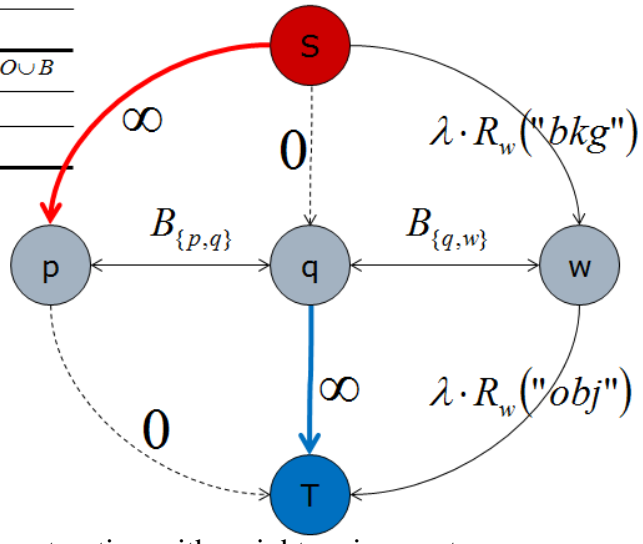


Figure 2.2: Sample graph construction with weight assignments

by painting some regions in the image using red and blue brushes and optimal segmentation is found satisfying these hard constraints.

2.5.1 Min-cuts as optimal object extraction

We consider the problem of extracting an object in an image by partitioning the graph into two disjoint set of nodes O and B. We aim to minimize the cut to extract the object in terms of properties built into the edge's weights. Let $\widehat{C}$ defines a minimum cut which is the sum of the boundary edges $(i,\ j)$ where $i \in s$ and $j \in t$ and $\widehat{X}$ defines the interest of object. In this section, we state that the $\widehat{C}$ provides the optimal object among all realizations with satisfying hard constraints. In order to prove that, we need one technical lemma. Let us assume that **F** denotes a sef of all *feasible* cuts C on the graph such that

- C severs only one t-link for each node
- $\{i,\ j\} \in C$ if p and q are linked to different terminals
- if $i \in O$ then $\{i,\ t\} \in C$
- if $i \in B$ then $\{i,\ s\} \in C$

Lemma : The minimum cut on **G** is feasible

Proof : $\widehat{C}$ cannot sever both t-link, it only severs one t-link for all nodes since it is a cut for separating the terminals. Similarly, it severs an n-link $\{i,\ j\}$ if i and j are connected to the different terminals because each cut must separate the terminals. If $\{i,\ j\}$ are connected to the same terminal, then $\widehat{C}$ should not sever unnecessary the n-link not to destroys the minimality. Last two properties are clear because ∞ is larger than the sum of all n-links costs for any given node. For example, if $i \in O$ then $\widehat{C}$ should not sever $\{i,\ s\}$ (costs ∞) because we could have constructed a smaller cost by severing all n-links from i (less than ∞) as well as the other t-link $\{i,\ t\}$ (zero cost).

A unique interest of object realization $X(C)$ can be defined for any feasible cut $C \in F$ as follow;

$$X_i(C) = \begin{cases} \text{"obj"} & \text{if } \{i,\ t\} \in C \\ \text{"bkg"} & \text{if } \{i,\ s\} \in C \end{cases} \tag{2.21}$$

Since any feasible cut are t-linked to one of two t-links at each pixel, the definition is compatible. In conclusion, the lemma shows us a minimum cut $\widehat{C}$ is feasible and we can define an object as $\widehat{X} = X(\widehat{C})$. To complete explaining of how $\widehat{X}$ is the optimal solution, we need the next theorem.

Theorem : The object $\widehat{X} = X(\widehat{C})$ defined by the minimum cut $\widehat{C}$ minimizes the cost function among all realizations satisfying hard constraints.

Proof : A cost of any feasible cut by using the table of edge weights as follows;

$$\mid C \mid = \lambda \cdot \sum_{i \notin O \cup B} V_i(X_i(C)) + \sum_{\{i,\ j\} \in \mathbf{N}} V_{ij}\left(x_i,\ x_j\right) \cdot \delta\left(X_i(C),\ X_j(C)\right) \tag{2.22}$$

The only difference $E(X)$ and $\mid C \mid$ is that $E(X)$ contains all nodes for regional term but $\mid C \mid$ contains only difference of all nodes and the union of O and B. Also, the intersection of O and B is empty;

$$i \notin O \cup B = i \in \mathbf{V} - (i \in O + i \in B) \tag{2.23}$$

Therefore we can reformulate the equation as follows;

$$\mid C \mid = E(X) - \lambda \cdot \left(\sum_{i \in O} V_i(\text{"}obj\text{"}) + \sum_{i \in B} V_i(\text{"}bkg\text{"}) \right) \tag{2.24}$$

Since $\mid C \mid = E(X) - const(C)$, we can prove why energy optimization equivalent to min-cut as follows;

$$\begin{aligned} E(\widehat{X}) &= \mid \widehat{C} \mid + \text{const} \\ &= \min_{C \in F} \mid \widehat{C} \mid + \text{const} \\ &= \min_{C \in F} E(X(C)) \end{aligned} \tag{2.25}$$

Our overall goal is to find a *globally minimum cut* in all possible cuts with satisfying the hard constraints. The globally minimum cut will give us a partitioned graph and segmented image.

Data: Image Pixels: $i \in d$
Result: Only *n-edges*: $e_{ij} \notin$ *t-edges*
begin
  for $i \in d$ **do**
    for $j \leftarrow N_i$ **do**
      $e_{ij} \leftarrow w_{ij} = \exp\left(-\frac{(d(i)-d(j))^2}{2\cdot\sigma^2} \cdot \frac{1}{dist(i,\, j)}\right)$
    end
  end
end

Algorithm 1: Assign $V_{ij}\left(x_i, x_j\right)$ to each edge

Data: User Seeds: $objList, bkgList \in S$
Result: *t-edges*: $\varepsilon \notin$ *n-edges*
begin
  $H_O \leftarrow -log(histogram(objList))$;
  $H_B \leftarrow -log(histogram(bkgList))$;
  for $i \in d$ **do**
    if $i \in objList$ **then**
      $e_{is} \leftarrow \infty$
    end
    else if $i \in bkgList$ **then**
      $e_{it} \leftarrow \infty$
    end
    else $i \notin objList$ and $i \notin bkgList$
      $e_{is} \leftarrow -\lambda H_O(i)$
      $e_{it} \leftarrow -\lambda H_B(i)$
    end
  end
end

Algorithm 2: Assign $V_{ij}(x_i, s)$ and $V_{ij}(x_i, t)$ to each node

3. OPTIMIZATION

In optimization theory, the maximum flow / mininim cut theorem states that the maximum amount of flow passing from the source to the sink is equal to the minimum capacity [16] in a flow network. Thus, the minimum cut problem can be solved by finding the maximum flow from the source to the sink. The idea is to deal with the graph G as a flow-network with the edges E representing pipes and send as much flow as possible from source to sink. The pipe capacities are given by w_{ij} determining the thickness or diameter of the pipes, i.e., the maximum amount of flow that can pass through. Formally, *flow* f is a mapping $f : E \rightarrow R^{+}$ assigning each edge $e_{ij} \in E$ a non-negative value w_{ij}. The flow is considered *feasible* if it satisfies the following three properties;

- Capacity Constraint

  $0 \leq f_{ij} \leq w_{ij} \qquad \forall e_{ij} \in E : (i,\ j) \in S$

- Anti-Symmetry

  $f_{ij} = -f_{ji} \qquad \forall e_{ij} \in E : (i,\ j) \in S$

- Flow Conservation

  $\sum_{j \in N_i} f_{ij} = 0 \qquad i \in S \setminus \{s,\ t\}$

According to the capacity constraint, the flow along an edge cannot exceed its capacity. Anti-symmetry condition implies that for each non-terminal node the total amount of incoming and outgoing flow must be the same between nodes pair. Flow conservation condition implies that the total flow to a non-terminal node is zero, except for the source, which *produces* flow, and the sink, which *consumes* flow.

The value of a flow $| f |$ is defined as the amount of flow leaving the source or equivalently the amount of flow arriving at the sink:

$$| f | = \sum_{\{i,\, s\} \in S} f_{si} - \sum_{\{i,\, s\} \in S} f_{is} = \sum_{\{i,\, t\} \in S} f_{it} - \sum_{\{i,\, t\} \in S} f_{ti} \qquad \textbf{(3.1)}$$

3.1 Determining Maximum Flow

The goal of *the maximum flow problem* is to find a feasible flow of the maximum value. There are mainly two strategies for establishing the maximum flow on a graph; (i) Goldberg-Tarjan style "push-relabel" and (ii) Ford-Fulkerson style "augmenting paths". Before we get started we should review the notion of a *residual graph* and *residual capacity*. Given the original graph G and a flow f, the residual capacity r_{ij} of an edge $e_{ij} \in E$ is the maximum additional flow that can be sent from i to j using both edges e_{ij} and e_{ji}. This is to say that the residual capacity is equal to the difference between capacity and current flow, formally $r_{ij} = w_{ij} - f_{ij}$. Edges which have a flow equal to their maximum capacity are said to be *saturated*. If an edge is saturated, its residual capacity is zero. A residual graph is a directed graph with the same topology as G and edge weights reflecting the actual residual capacity with respect to f.

The augmenting path strategy for determining the maximum flow was devised by Ford and Fulkerson. It is a general framework in which the following steps are performed;

1. Start with no flow along any edge and with a residual graph which is exactly the same as the graph G.

2. Find an augmenting path from from source to sink along unsaturated edges in the residual graph.

3. Push the flow along the path by the amount equal to the minimum residual capacity encountered on the path. The amount is the largest amount that will not cause one of the capacities to be exceeded. Then the amount is subtracted from the residual capacity of each edge on the path and added to the residual capacity of opposite edges. At least one edge on the path becomes saturated.

4. Go back to step 2 and repeat. When no such path exists, the flow will be maximised.

An example of the Ford-Fulkerson process is depicted in Fig 3.1. Note that in some cases there is more than one possible pattern for the flows in the edges which give the overall maximum flow. In conclusion, the Ford-Fulkerson algorithm always finds a maximum flow.

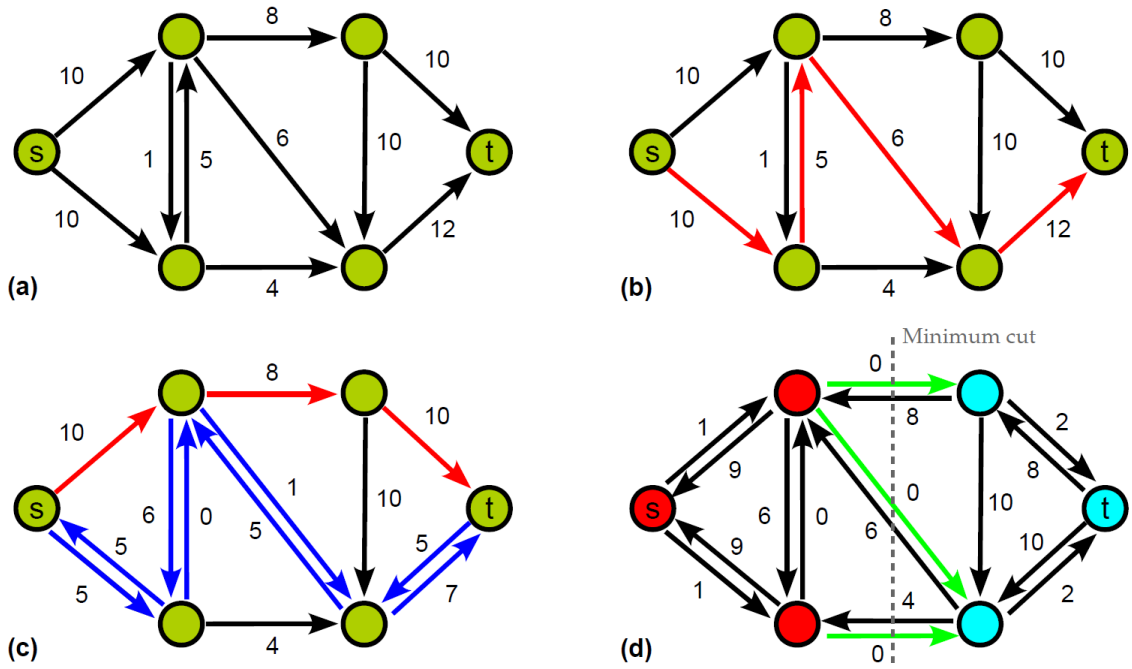


Figure 3.1: Maximum flow computation using the augmenting path strategy. (a) The input graph with edge weights w_{ij}. (b) The initial residual graph with residual capacities r_{ij} being the same as w_{ij} in the input graph (zero flow). An augmenting path is shown in red. (c) Residual graph after pushing 5 flow units across the path. Edges with updated residual capacity are shown in blue. A new augmenting path is shown in red. (d) Residual graph at the end of the computation. The terminals source and sink correspond to red and cyan nodes, respectively. Saturated edges comprising the minimum cut edges are shown in green, from [17]

Push-Relabel methods follow a different strategy than that the augmenting path strategy. The purpose of them is to push as much flow as possible from the source to the sink. Hence they do not maintain a valid flow in each iteration. Instead, they work with pre-flow which is a flow that satisfies capacity constraints but depart from the flow conservation properties by allowing a node to store "flow excess". If a node has a positive flow excess, called active node. The algorithm also associates a label for each node by giving a low bound on distance of the node from the sink. In addition, an residual edge is said to be admissible if it is the middle of two labels. Basically the algorithms have the simple operations; push flow and relabel an overflowing node. An active node attempts to push its excess flow towards nodes through admissible edges and the node's label is increased by 1. For more detailed information about them, please refer the original papers [18], [16].

3.2 Determining Minimum Cut

The famous Ford-Fulkerson theorem states that the maximum total value of a flow equals the minimum capacity of a cut C in every networks. Corresponding to the maximum flow, we can find a cut of capacity $C = \mid f \mid$. Since no flow can have a total value greater than C, and no cut can have a capacity less than f, these numbers are simultaneously the maximum and the minimum referred to in the theorem. In particular, the source terminal s consists of nodes reachable in the final residual graph via unsaturated edges from the source. Thus the sink terminal t simply contains the rest of the graph nodes. In the example from Fig. 3.1, these terminals correspond to the red and cyan nodes, respectively. The set of edges going from source to sink and comprising the minimum cut is highlighted in green. Notice that edges constituting the cut set are always saturated at the end of the computation, i.e., their residual capacity is zero. The cost of the cut is the sum of the initial weights of the green edges which is 18. This is the same as the flow value that can be determined from the residual graph by comparing the residual and initial capacities of edges leaving the source or entering the sink. Finally, neither the solution of the minimum cut problem nor the maximum flow problem has to be unique, several minimum cuts may be possible in the graph.

3.3 The Experimental Algorithm of Boykov and Kolmogorov

As we have already mentioned, a minimum cut is traditionally computed by using maximum flow algorithms. In this section, we present the maximum flow algorithm which is a variety of the augmenting-path algorithm developed by Boykov and Kolmogorov in [9]. We will refer to it simple as *bk-maxflow* hereafter.

Standard augmenting path algorithms find the *shortest* paths from source to sink nodes and augment them by subtracting the bottleneck capacity found on that path from the residual capacities of each edge and then add the bottleneck capacity to the total flow. Additionally the it is also added to the residual capacity of the reverse edges. If no more paths in the residual-edge tree are found, the algorithm terminates. In order to achieve that they depend on uninformed search that aims to expand and examine all nodes of a graph or combination of sequences by systematically searching through every solution. Therefore, it exhaustively searches the entire graph from source to

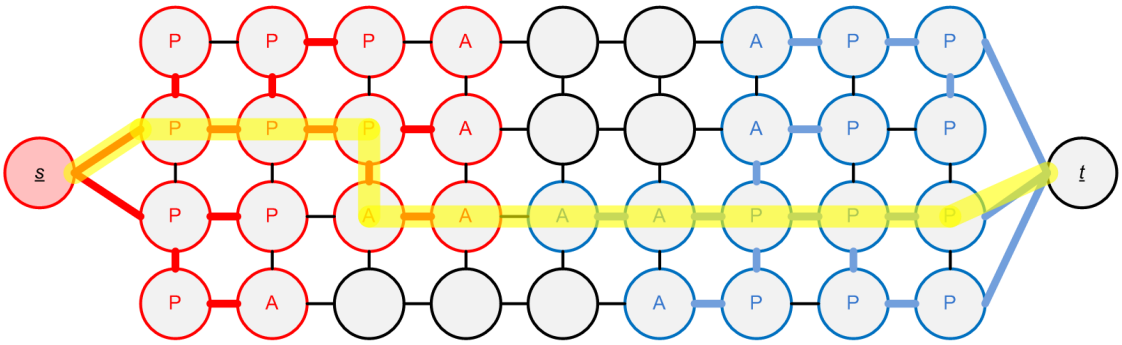


Figure 3.2: Boykov-Kolmogorov maximum flow algorithm scheme with the source and sink trees and active (A), passive (P) and free (F) nodes. An augmenting path (bold yellow) is found when the two dynamic search trees touch

sink without considering the goal until it finds it. Hence, they are very expensive and time-consuming operation practically. However, instead of finding a new shortest path from source to sink in each iteration, the bk-maxflow algorithm keeps the already found paths (even if it is not the shortest path) in order to fasten the standard augmenting path algorithms as follows:

First, bk-maxflow builds up two *dynamic search trees* for detecting augmenting paths, one rooted from the source and the other rooted into the sink. Second, it reuses trees to avoid starting from scratch again. Last, the augmenting path found are not necessarily shortest paths; thus the time complexity of the standard augmenting path is not valid and bk-maxflow has no polynomial-time bound. In fact, the lack of a polynomial time bound is disappointing because the maximum flow problem has been extensively studied from the theoretical point of view and is one of the better understood combinatorial optimization problems. However, bk-maxflow is superior in practice to general-purpose methods on many vision instances. The algorithm is an interesting development from a practical point of view.

3.3.1 Algorithm overview

The main idea of the bk-maxflow is illustrated in Fig. 3.2. It is based on the augmenting path principle and its popularity and efficiency stems from the way augmenting paths are searched. It grows two dynamic search trees rooted from source s and rooted into sink t in the residual graph. Each node can belong to exactly one of source and sink trees or *free* at anytime. If nodes are on the border of a search tree, the tree can grow

from them, we call them as *active* nodes while a passive node is inside the tree. The passive nodes cannot grow their trees because they have no chance to interact with a free node or the other tree, they are completely blocked by the their own tree. However, an active node can come in contact with an free or another node that belongs to the other tree. A node which is not a member of neither of the two trees is called *free*. If a free node is acquired by a search tree, it becomes an active node of it.

The algorithm repeatedly picks active nodes from the search trees and tries to grow the tree by acquiring their free neighbors. An augmenting path is found when the two trees touch each other. This stage is called *growth*. In the *augmentation* stage, the maximum flow of the path is sent along this path. This step may break the trees into forests as edges become saturated. Subsequently, *adoption* stage is executed where search trees are restored by finding new parents for the orphan nodes and the whole process continues with the tree growth stage again.

The algorithm maintains a flow f and the dynamic trees of residual graph G_f, s rooted from source nodes s_{user} that are marked by user as object, and t rooted into sink nodes t_{user} that are marked by user as background. Initially dynamic search trees only contains user clues which are all active node at the beginning. The algorithm keeps the list of all active nodes, A and orphans, O for every iteration. At each iteration, a node can be an $s\text{-}node$, $t\text{-}node$, $s\text{-}orphan$, $t\text{-}orphan$ or free. In addition each node maintains a parent connection which is *null* for free and orphan nodes.

Initially, s and t contain user seeds, all nodes in these trees are active, and all parent connections are $null$. Therefore, all nodes in S are $s\text{-}node$, in T are $t\text{-}node$, the remaining nodes are $free$. The algorithm expands the trees to grow in each iteration. The goal of an iteration is to grow the trees by acquiring new children from a set of free nodes. The newly acquired children nodes are active. Each iteration executes growth phase until an augmenting path is found. Then the augmentation phase augments the path, followed by adoption steps to process orphan nodes. The general structure of the algorithm is as follows;

The outer loop of the algorithm consists of three phase: *growth*, *augmentation*, and *adoption* described respectively in the following subsections.

Data: $S = \{s_{user}\}$, $T = \{t_{user}\}$, $A = \{s_{user}, t_{user}\}$, $O = \{\emptyset\}$, $B = \{\emptyset\}$, $path = null$
Result: Dynamic Object Extraction Image
begin
  while *true* **do**
    grow on both search trees simultaneously to find a new *path*
    if *path* = *null* **then**
      Display dynamic extraction result
      Terminate the algorithm!
    end
    augment on the *path*
    adopt orphans
  end
end

Algorithm 3: General Structure

growth phase

In this phase, the algorithm tries to expand the trees by scanning their active nodes and adding newly-discovered free nodes that are connected through a non-saturated edge to the tree from which they have been discovered. The newly-added nodes become active nodes of the corresponding search trees and the current active node becomes the parent of the newly-added node. Each active node becomes passive after all neighboring nodes are scanned. The growth phase continues with the next active node. If no active nodes remain, the algorithm terminates. If a node discovers another node from the other search tree through an unsaturated edge, then the augmentation phase starts. In this case a path from source to sink is found, as shown in Fig 3.3.

The growth phase picks an active node i and scans neighboring nodes $j \in N_i$ by examining free nodes with non-saturated residual edges $w_{ij} > 0$. If i and j are in the same tree, it does nothing. However, if j is free, it is set to the associated tree $TREE(j) \leftarrow TREE(i)$, and its parent is set to $PARENT(j) \leftarrow i$. The $TREE(\cdot)$ indicates the affiliation of each node i so that

$$TREE(i) = \begin{cases} s_{node} & \text{if } i \in s \\ t_{node} & \text{if } i \in t \\ free & \text{otherwise} \end{cases} \tag{3.2}$$

If all edges out of i are scanned, i becomes passive. If a scan of i discover a node from the other tree, the algorithm is interrupted to perform an augmentation phase.

Data: $S = \{s_{user}\}$, $T = \{t_{user}\}$, $A = \{s_{user}, t_{user}\}$, $O = \{\emptyset\}$, $path = null$
Result: path
begin
  while $A \neq \emptyset$ **do**
    pick an active node $i \in A$
    for $j \in N_i$ **do**
      if $e_{ij} \leq 0$ **then**
        if $TREE(j) = \emptyset$ **then**
          $TREE(j) \leftarrow TREE(i)$
          $PARENT(j) \leftarrow i$
          $A \leftarrow A \cup \{i\}$
        end
        else if $TREE(j) \neq \emptyset$ **then**
          if $TREE(j) \neq TREE(i)$ **then**
            return **path**
          end
        end
      end
    end
    $A \leftarrow A - \{i\}$
  end
  return **null**
end

Algorithm 4: Growth Stage

augmentation phase

This phase gets the path found by the growth phase and augments the flow on it by its bottleneck residual capacity. At first, it finds the bottleneck capacity of the found path, and then it updates the residual-capacity of the edges from this path by subtracting the bottleneck capacity from the residual graph. Furthermore the residual capacity

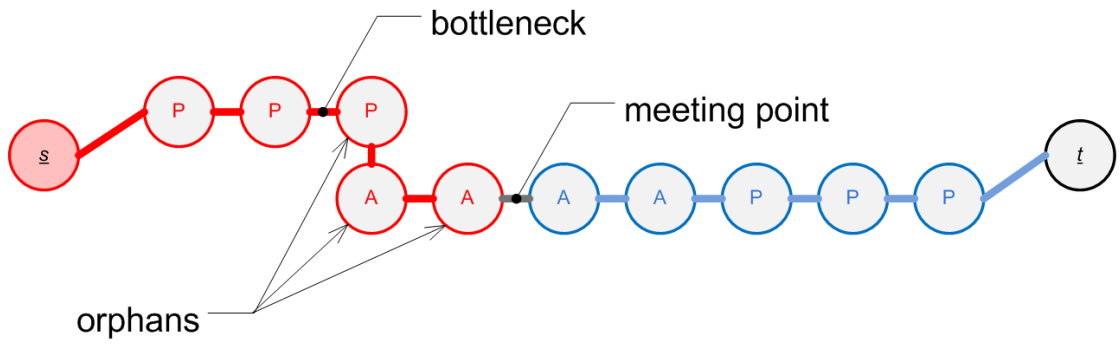


Figure 3.3: Boykov-Kolmogorov maximum flow algorithm scheme with the source and sink trees and active (A), passive (P) and free (F) nodes. An augmenting path (bold yellow) is found when the two dynamic search trees touch

of the reverse edges are updated by adding the bottleneck capacity. This phase can destroy the built up search trees, as it creates *at least* one saturated edge. That is to say that the search trees collapse to forests, because a condition for the search trees is, that each vertex in them has a valid (=non-saturated) connection to a terminal. Thus some tree edge(s) on the path become saturated and their endpoints farthest from the corresponding root become *orphans* since the edges linking them to their parent are not valid anymore, as shown in Fig 3.3.

If an edge e_{ij} becomes saturated and both i and j are in S, then j becomes an *s-orphan*. If both i and j are in T, i becomes a *t-orphan*. If i is in S and j is in T, then a saturation of e_{ij} does not create orphans. Orphans are placed on a list and processed in the adoption phase

The augmentation stage starts with the path from source to sink, and updates the orphan list and the residual graph by pushing the bottleneck capacity through the path.

Data: path
Result: orphans
begin
  $\Delta = 0$ (for bottleneck capacity)
  for $e_{ij} \in Path$ **do**
    if $\Delta < e_{ij}$ **then**
      $\Delta \leftarrow e_{ij}$
    end
  end
  update the residual graph by pushing flow Δ through the path
  for $e_{ij} \in Path$ **do**
    if $e_{ij} = 0$ *(if saturated)* **then**
      if $TREE(j) = s$ & $TREE(i) = s$ **then**
        $PARENT(j) \leftarrow \emptyset$
        $O \leftarrow O \cup \{j\}$
      end
      else if $TREE(j) = TREE(i) = T$ **then**
        $PARENT(i) \leftarrow \emptyset$
        $O \leftarrow O \cup \{i\}$
      end
    end
  end
end

Algorithm 5: Augmentation Stage

adoption phase

In this phase, the search trees are reconstructed. As described before, the augment-phase can create orphans, so the algorithm attempts to find a new valid parent for each orphan from the endpoint farthest of the corresponding root to the bottleneck. The orphans are ordered according to their distance to the terminals (smallest first). It is important that the new parent must be in the same tree with the orphan and also be connected through a non-saturated edge. If the algorithm finds a new parent for an orphan, it is marked as the orphan's new parent. However, if no new valid parent is found, this node becomes a free node, and it's children become orphans as well. The adoption phase processes orphans until there are none left.

```
Data: Orphans
Result: Restored Search Trees
begin
    while O ≠ ∅ do
        pick and remove an orphan node i from O
        for j ∈ N_i do
            if TREE(j) ≠ TREE(i) then
                if w_ij ≤ 0 then
                    // some nodes in a search tree may originate from orphans
                    // therefore, we must check the "origin" of j
                    if j's origin ≠ O then
                        // i remains in its search tree
                        // the active status of i remains unchanged
                        PARENT(i) ← j
                        break;
                    end
                end
                A ← A ∪ {j}
                if PARENT(j) = i then
                    O ← O ∪ {j}
                    PARENT(j) ← ∅
                end
            end
            TREE(i) ← ∅
            A ← A − {i}
        end
    end
end
```

Algorithm 6: Adoption Stage

A timestamp is stored for each node which shows in which iteration of the algorithm the distance to the corresponding terminal was calculated. This distance is used and gets calculated in the adoption-phase. In order to find a valid new parent for an orphan, the possible parent is checked for a connection to the terminal to which tree it belongs. If there is such a connection, the path is tagged with the current time-stamp, and the distance value. If another orphan has to find a parent and it comes across a node with a current timestamp, this information is used.

The distance is also used in the grow-phase. If a node comes across another node of the same tree while searching for new nodes, the other's distance is compared to its distance. If it is smaller, that other node becomes the new parent of the current. This can decrease the length of the search paths, and so amount of adoptions.

The bk-maxflow algorithm is mainly implemented as described by [9]. and can be found in C++ Boost Graph libraries.

4. IMPLEMENTATION AND RESULTS

In the previous chapters, we presented the PGM in terms of MAP-MRF modeling and bk-flow optimization. Now we show that the IOE implementation can be efficiently used on Android Operating System (OS) based mobile devices. We present results in the single image with timing and extracted objects.

The Android application was implemented in two parts; the user interaction (Graphical User Interface, GUI) depends on Android Software Development Kit, SDK, (based on Java) and the algorithm's main implementation (based on native C++ with Standard Template Library, STL). The main algorithm was implemented using native C++ motivated by the Android Native Development Kit, NDK, and Java Native Interface, JNI, because of performance limitations of Java. We use JNI and NDK somewhat interchangeably but they could also be used to define very specific and different things. JNI could be used to define all Java applications to use native code, whereas NDK would be more related to the Android system in specific. NDK defines a way for managed code (written in the Java programming language) to interact with native code (written in C/C++). JNI is the interface between the Java code and the native code. It works both ways, that is you can use JNI to call native code from your Java programs and to call Java code from your native code. The native code normally resides within a library (.so file) and is typically written in C/C++. JNI defines two key data structures, *JavaVM* and *JNIEnv*. Both of these are essentially pointers to pointers to function tables. (In the C++ version, they're classes with a pointer to a function table and a member function for each JNI function that indirects through the table) The JavaVM provides the *invocation interface* functions to create and destroy a single JavaVM. The JNIEnv provides most of the JNI functions. Native functions all receive a JNIEnv as the first argument used for functions stack. For this reason, we cannot share a JNIEnv between functions (for further details please refer to the android developer website at http://developer.android.com/index.html)

Our main implementation were written in pure C++, it can be compiled in any device without any porting effort. Therefore, the main implementation can be used in any platform, we tested it in Microsoft Windows 7 using Microsoft Visual Studio 2013 Express and Ubuntu 12.04 LTS using g++ without any change for different platforms. Only GUI part must be changed, please note that, we used OpenCV (it is free for academic and commercial use because of BSD license) to supports Windows, Linux for handling image informations, but our main focus is on Android based smart-phones. The max-flow/min-cut optimization algorithm developed in [9] is used to solve the minimum cut problem. It can also be found in Boost Graph Library as boykov_kolmogorov_max_flow() function which calculates the maximum flow of a network. The calculated maximum flow will be the return value of the function, please refer to the Boost C++ Graph Library for more information. In fact, there were two reasons why JNI is used in our application; first we need to get rid of performance bottlenecks and second we need to use the native legacy max-flow code without porting effort.

Briefly, the application contains four parts; n-link edge assignment, marking seeds, t-link-edge assignment and computing graph cuts. The application is made up of two threads; the *n-link edge assignment* thread and the *main* thread to handle segmentation. It also has a asynchronous marking seeds event in which the user can mark the seeds whenever (s)he wants. We designed n-link edge assignment as a separate thread because it is independent from the remaining part of the application. Besides, we have enough time to calculate n-links while the user is marking the seeds. However, if the user finishes the marking seeds before the n-link assignment is not finished, the main thread waits until the n-link assignment is completed. So there is a synchronization between the two thread.

Results were obtained on a Samsung Note II N7100 smart-phone with Quad-core 1.6 GHz Cortex-A9 processor and 2 GB DDR RAM running Android OS, v4.1.1 (Jelly Bean). We used different parameter values and scene of different complexity during experimentation. The execution time of the application was also measured. In the implementation, the n-link assignments are done in a different thread during the user

marks object/background seeds to reduce overall time. And if new seeds are added then the t-links are updated at the beginning of executing max-flow/min-cut algorithm.

4.1 Results

In this section we consider a number of examples that illustrate how the application can be used to extract objects in images. Results were obtained for different parameter values, overall time and image complexity. The images size is always the same 1024x768. Times taken for the n-links assignment was approximately 3 seconds for all images because n-link assignment only depends on the number of pixel and the order of the neighborhood. Besides, the t-link assignment does not affect the total time because is done during n-link assignment. The max-flow/min-cut algorithm in total took about 6 seconds.

The screen-shots of the smart-phone application can be seen in 4.1. Our interface allows a user to enter seeds with the smart phone's S-Pen device. Throughout the thesis we indicate object seeds by red color and background seeds by green or blue colors to increase visibility of the seeds. In general, the user can extract any object, but at first, if the results are not satisfactory in some part of the image, the user can add new object and/or background seeds providing additional clues where the object and/or background was wrong, until all problems are corrected. The exact choice of seed positioning is not relevant. Normally, different object seeds positions within a region of "similar" intensity inside one object region cannot change the optimal solution. It can also be shown that additional object seeds inside the object region cannot change the result. Both properties are equally true for background seeds.

The object seeds provide the necessary clues on what parts of the image are interesting to the user. There are no strict requirements on where the object seeds have to be placed as long as they are inside the object(s) of interest. Such flexibility is justified, in part, by the ability of the algorithm to efficiently incorporate the seeds added later on, when the initial extraction results are not satisfactory. The object seeds can be placed sparingly and they do not necessarily have to be connected inside each isolated object. Potentially, the algorithm can create as many separate object segments as there are connected components of object seeds. Nonetheless, the isolated seeds (or components

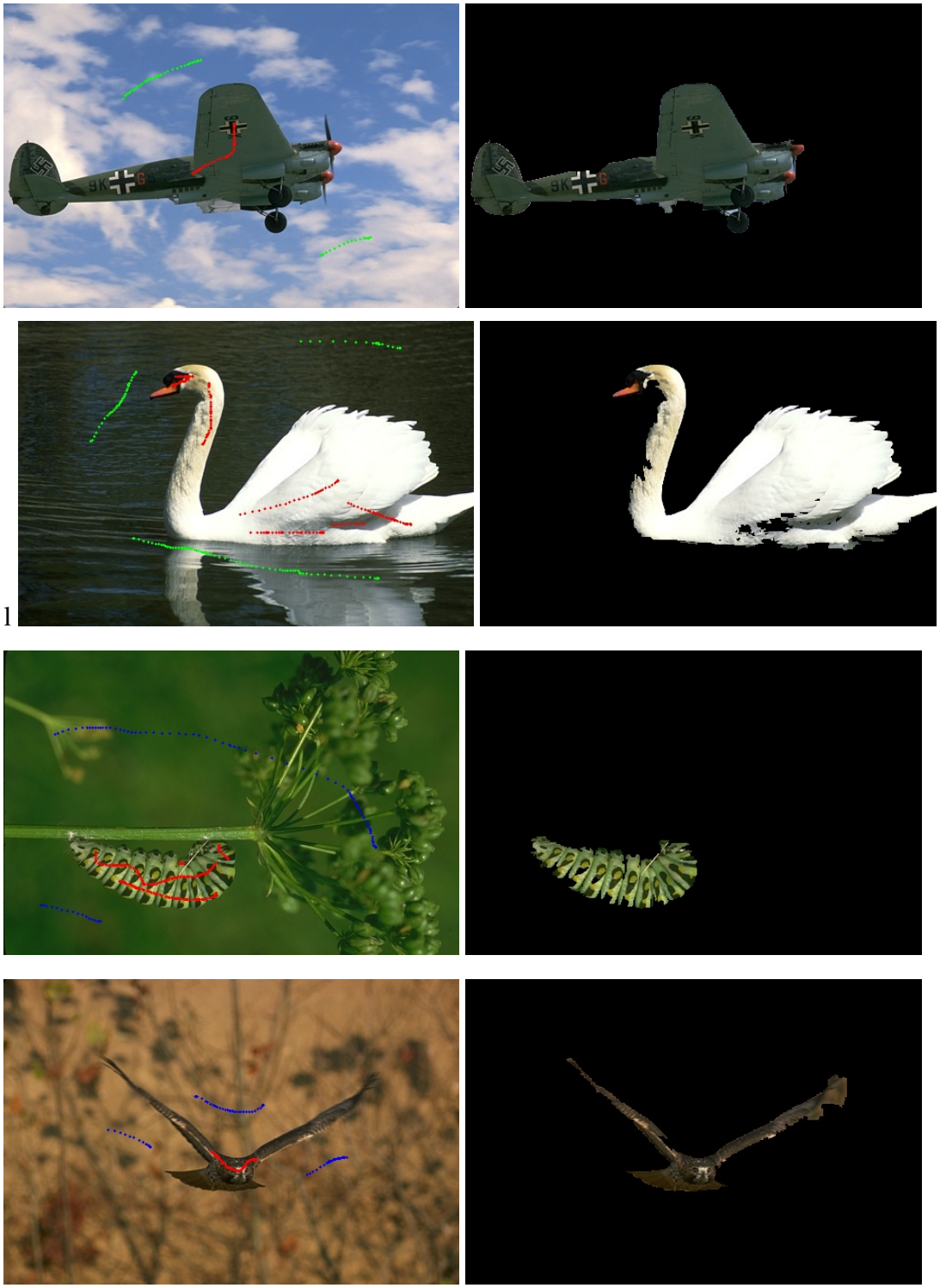


Figure 4.1: The screen-shots of the smart-phone application for $\lambda = 2$ and $\sigma = 7$

of seeds) located not too far from each other inside the same object are likely to be segmented out together [19].

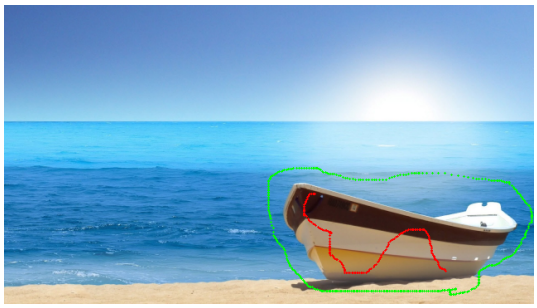
The background seeds should provide the clues on what is not an object of interest. In many situations the pixels on the image boundary are a good choice for background seeds. If the objects of interest are bright, then background seeds can be spread out in the dark parts of the image. Note that background seeds are very useful when two similar objects touch in some area of the image and one of them is of interest while the other is not. In this case there is a chance that the two objects may be merged into a single segment. This can be avoided by placing a background seed in the undesired object which would force the extraction to separate the two objects at their merge point. Also, when the extraction merges two objects of interest that should be separated, the user can add a background seed in between to effectively separate the two objects.

If we write the equation **2.16** as complete explicit formula

$$F(x) = \lambda \cdot \sum_{i \in \mathbf{S}} -log\left(p\left(d(i) \mid \mathbf{H}_k\right)\right) + \sum_{\{i,\, j\} \in \mathbf{N}} exp\left(-\frac{\left(d(i) - d(j)\right)^2}{2\sigma^2}\right) \cdot \frac{1}{dist\,(i,\ j)} \left(1 - \delta(x_i, x_j)\right) \tag{4.1}$$

where we see that two parameters exist we can change, λ and σ. Fig 4.2 and Fig 4.3 show the effect of different values of λ. When λ is high, it means that we are giving more importance to the observed data. Therefore, high value of λ makes it similar to thresholding based segmentation. This makes segmentation difficult if the object and background have similar color. Fig 4.2 and Fig 4.3 also show the effect of different values of σ. The cost function penalizes a lot for discontinuities between pixels of similar intensities when $\mid d(i) - d(j) \mid < \sigma$. However, if pixels are very different, $\mid d(i) - d(j) \mid > \sigma$, then the penalty is small. Intuitively, this function corresponds to the distribution of noise among neighboring pixels of an image.

(a) Original Image



(b) Original Image with user seeds

(c) $\lambda = 10$ and $\sigma = 0.1$

(d) $\lambda = 0$ and $\sigma = 2$

(e) $\lambda = 10$ and $\sigma = 2$

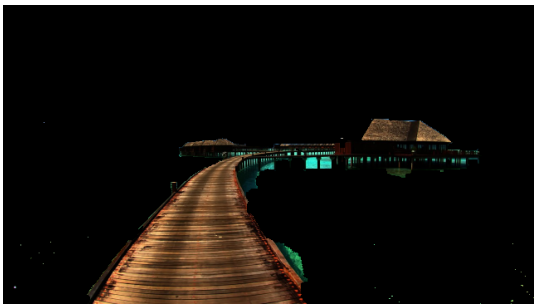
(f) $\lambda = 40$ and $\sigma = 2$

(g) $\lambda = 10$ and $\sigma = 100$

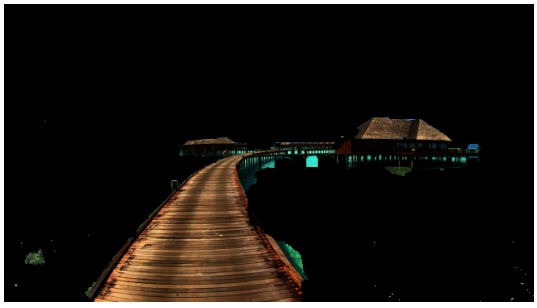
Figure 4.2: Results1: the screen-shots of the smart-phone application for different parameter values

(a) Original Image

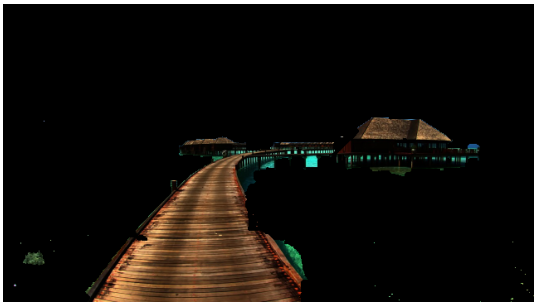
(b) Original Image with user seeds



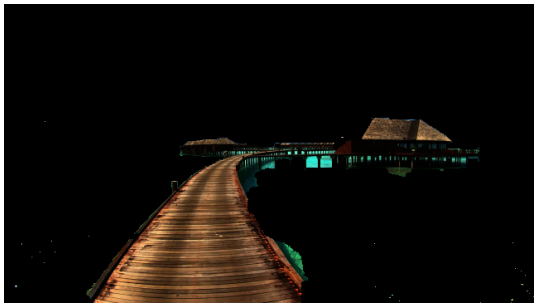
(c) $\lambda = 10$ and $\sigma = 0.1$



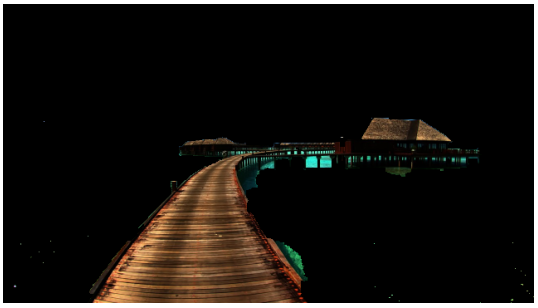
(d) $\lambda = 0$ and $\sigma = 2$



(e) $\lambda = 10$ and $\sigma = 2$



(f) $\lambda = 40$ and $\sigma = 2$



(g) $\lambda = 10$ and $\sigma = 100$

Figure 4.3: Results2: the screen-shots of the smart-phone application for different parameter values

5. CONCLUSION

In this thesis, we implemented an application based graph-cut and MRF-MAP framework for extracting objects from images. This was done in order to provide insight into the power of the framework, so that it could be applied to more diverse problems in future. In many ways the graph-cut optimization and max-flow algorithms to object extraction can be seen as a unifying framework that combines many good features of the previous methods like snakes, active contours, and level sets while providing efficient and robust global optimization applicable to N-D problems [20]. The results showed that the framework provides a fast and easy way for extracting object interactively for color images.

It should also be noted that among many object extraction methods, the level set methods and the graph minimal cut approaches have emerged as two powerful paradigms to extract the objects of images. Both methods are based on fundamentally different representations of images. Level sets are formulated as infinite-dimensional optimization on a spatially continuous image domain. The Graph Cuts are defined as minimal cuts of a discrete graph representing the pixels of the image. Therefore, graph cuts framework uses implicit representation of object boundaries which makes them a discrete counterpart of level-sets. In adddition, the graph cuts framework is very flexible with initialization. The method does not have to have an initial contour/surface, but it may take advantage of some shape prior or of some initial approximate guess, if available. Also the algorithm can be accelerated for videos where previously segmented image frame is similar to the new frame.

REFERENCES

[1] **Rother, C.**, **Kolmogorov, V.**, **Boykov, Y. and Blake, A.**, 2011, Interactive Foreground Extraction using graph cut, Microsoft Technical Report: MSR-TR-2011-46.

[2] **Li, S.**, 2000. Modeling image analysis problems using Markov random fields, Handbook of Statistics, vol. 20, pp. 1–43, Wiley, New York.

[3] **Kolmogorov, V. and Zabih, R.**, 2004. What Energy Functions Can Be Minimized via Graph Cuts?, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26(2)**, 147–159.

[4] **Hammersley, J.M. and Clifford, P.**, 1971, Markov fields on finite graphs and lattices, `http://www.statslab.cam.ac.uk/~grg/books/hammfest/hamm-cliff.pdf`, unpublished report.

[5] **Besag, J.**, 1974. Spatial Interaction and the Statistical Analysis of Lattice Systems, *Journal of the Royal Statistical Society. Series B (Methodological), Vol. 36, No. 2.*, 192–236.

[6] **Geman, S. and Geman, D.**, 1984. Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *Pattern Analysis and Machine Intelligence, IEEE Transactions on (Volume:PAMI-6 , Issue: 6)*, 721 – 741.

[7] **Boykov, Y.Y. and Jolly, M.P.**, 2011. Interactive Graph Cuts for Optimal Boundary and Region Segmentation of Objects in N-D Images, *Proceedings of "Internation Conference on Computer Vision", Vancouver, Canada*, vol.I, p.105.

[8] **Greig, D.M.**, **Porteous, B.T. and Seheult, A.H.**, 1989. Exact Maximum A Posteriori Estimation for Binary Images, *Journal of the Royal Statistical Society. Series B (Methodological)*, **51(2)**, 271–279.

[9] **Boykov, Y. and Kolmogorov, V.**, 2004. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision, *IEEE PAMI*, **26(9)**, 1124–1137.

[10] **Ford, L.R. and Fulkerson, D.R.**, 2010. Flows in Networks, *Princeton University Press*.

[11] **Yuri Boykov, O.V. and Zabih, R.**, 1998. Markov Random Fields with Efficient Approximations, *IEEE Conference on Computer Vision and Pattern Recognition.*, 648–655.

[12] **Wang, C. and Paragios, N.**, 2012. Markov Random Fields in Vision Perception: A Survey.

[13] **Komodakis, N.**, 2006. Optimization Algorithms for Discrete Markov Random Fields, with Applications to Computer Vision, *University of Crete, Doctor of Philosophy Thesis*.

[14] **A., B., P., K. and C., R.**, 2011. Markov Random Fields for Vision and Image Processing, *MITpress*.

[15] **Hammersley, J.M. and Clifford, P.**, 1971. Markov fields on finite graphs and lattices.

[16] **Ford, L.R. and Fulkerson, D.R.**, 1956. Maximal flow through a network, *Canadian Journal of Mathematics 8: 399*.

[17] **Danek, O.**, 2012. Graph Cut Based Image Segmentation in Fluorescence Microscopy, *Faculty of Informatics Masaryk University, PH.D. Thesis*.

[18] **Goldberg, A. and Tarjan, R.**, 1988. A new approach to the maximum flow problem, *Journal of the ACM*, 921–940.

[19] **Boykov, Y. and pierre Jolly, M.**, 2000. Interactive organ segmentation using graph cuts, 276–286.

[20] **Boykov, Y. and Funka-Lea, G.**, 2006. Graph Cuts and Efficient N-D Image Segmentation, *International Journal of Computer Vision*, **70(2)**, 109–131.

CURRICULUM VITAE

Name Surname: Ali Volkan ATLI

e-Mail: avatli@yahoo.com

Professional Experience:

Senior Software Developer at Siemens - Infrastructure & Cities / Smart Grid R&D
Embedded Software Developer at Netas - Defense and Strategic Solutions R&D

List of Publications:

A. V. Atli, E. Özturk, *"Non-invasive Respiratory Motion Tracking using M-Sequence UWB Radar"*, International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), pp: 1–4, ISSN: 2157– 0221, Oct. 2011, Budapest

A. V. Atli, İ. Bayram, *"Distance Regularized Level Sets with Signed Pressure Force Image Segmentation with Global Active Contours* accepted for SIU 2012

E. Öztürk, A. V. Atli, M. S. Mercan, A. Çalışkan, I. Y. Erer, *"Clutter Reduction Methods in Ultra Wideband Radar and Time/Performance Evaluation on ARM Processors"* accepted for SIU 2012

A. V. Atli, O. Urhan, S. Ertürk, M. Sönmez, *"A Computer Vision Based Fast Approach to Cutting Tool Condition Monitoring"*, Proceedings of the Institution of Mechanical Engineers, Part B, Journal of Engineering Manufacture, Vol. 220, No. 9, pp. 1409-1415, Sep. 2006

A. V. Atli, M.K. Gullu, S. Ertürk, A. Karson, N. Ates, *"Tracking the Rotation of Lab Animals via Computer Vision"*, Signal Processing and Communications Applications, SIU 2007 IEEE 15th 11-13 June 2007

A. V. Atli, O. Urhan, S. Ertürk, M. Sönmez, *"Kesici Takım Aşınmasının Görüntü İşleme ile Belirlenmesi"*, 13. IEEE Sinyal İşleme ve İletişim Uygulamaları Kurultayı (SİU 2005) Bildirileri Kitabı, pp. 84-86, 16-18 Mayıs 2005, Kayseri, TÜRKİYE, 2005