Improved Sampling-Based Alpha Matting in Images and Video

by

Chengcheng Hao

A thesis submitted to the University of Ottawa in partial fulfillment of the requirements for the degree of Master of Applied Science in Electrical and Computer Engineering

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Electrical Engineering and Computer Science
University of Ottawa

Ottawa, Ontario, Canada

September 2012

Abstract

Foreground extraction technology plays an important role in image and video processing tasks. It has been widely used in various industries. To better describe the overlap relationship between foreground and background, alpha channel is introduced. It reveals the opacity property of foreground objects. Thus, fully extracting a foreground object requires determining the alpha values for pixels, also known as extracting an alpha matte.

In this thesis, we propose an improved sampling-based alpha matting algorithm, which is capable of generating high quality matting results. By analyzing the weakness of previous approaches, we optimize the sampling process and consider the cost of each sample pair to avoid missing any good samples. The good performance is demonstrated even for complex images.

On the other hand, extracting foreground objects from video sequences is a more challenging task since it has higher demands on accuracy and efficiency. Previous approaches usually require a significant amount of user input and the results still suffer from inaccuracy. In this thesis, we successfully extend our algorithm to video sequences and let it run in an automatic fashion. Adaptive trimap, which is vital for matting, can be automatically generated and properly propagated in this system. Our method not only reduces the user interference but also guarantees the matting quality.

Contents

Abstract i

Contents ii

List of Figures iv

List of Acronyms vi

Dedication vii

Acknowledgement vii

1 Introduction 1

1.1 Image segmentation 1

1.2 Foreground extraction and image composition 3

1.3 Alpha matting 4

1.4 The contributions of the thesis 6

1.5 Thesis structure 7

2 Previous work on alpha matting 8

2.1 Hardware based (Matting with extra information) 9

2.2 Affinity-based methods 13
2.3 Sampling-based methods 17
2.3.1 Classic models of sampling-based methods 18
2.3.2 Sample selection approaches 22
2.4 Other methods 24

3 The improved sampling-based alpha matting for digital images 26
3.1 Failure examples in previous non-parametric sampling methods 27
3.2 Alpha matting using global-ranged smaples 29
3.2.1 Randomized search using PatchMatch and SampleMatch 29
3.2.2 Selecting criteria of optimal sample pairs 35
3.3 Experimental results 39
3.4 Summary 40

4 The sampling-based alpha matting for video 46
4.1 Background 46
4.2 Automatic adaptive trimap generation 49
4.2.1 Motivations 49
4.2.2 Clustering and binary mask 52
4.2.3 Adaptive trimap generation 54
4.3 Trimap propagation in temporal dimension 58
4.4 Experimental results 61
4.5 Summary 63

5 Conclusions and future work 66

References 68

List of Figures

1.1 Foreground extraction and image composition 3

1.2 The extracted object suffers from "color-spill" in hairy regions 4

1.3 Elements overlaying 5

1.4 Convex combination 6

2.1 Camera array and real-time system. 10

2.2 Foreground flash matting. 11

2.3 Multiparameter video camera and defocus matting. 12

2.4 Mishima's algorithm. 18

2.5 Ruzon-Tomasi's algorithm 19

2.6 Knockout algorithm. 20

2.7 Bayesian matting. 21

2.8 Examples of linear models. 23

2.9 "Geodesic samples" versus nearest sparse samples 24

2.10 Sampling method in Shared Matting 25

3.1 Two sampling methods comparison 28

3.2 Failed sampling examples 30

3.3 Another example that local samples are not reliable for estimation . . . 30

3.4 Global-ranged foreground and background sample set 31
3.5 Sort the foreground and background samples according to color intensity 32
3.6 Propagation of neighboring pixels' sample pairs 34
3.7 Compute the color cost 37
3.8 Encode criterion (ii) by considering the color weight 37
3.9 Matting results using local samples and global samples 39
3.10 Experimental results (still images) 40
3.11 Matting result comparison 1 41
3.12 Matting result comparison 2 42
3.13 Matting result comparison 3 43
3.14 Matting result comparison 4 44

4.1 Uniform trimap 50
4.2 Comparison of matting results using different trimaps 51
4.3 Generate binary mask by categorizing input image into two clusters . . 52
4.4 Foreground and background clusters and centroids 53
4.5 Define local sliding windows 55
4.6 Performance of sliding windows in complex situation 56
4.7 Indicators as a series of contour lines 57
4.8 Flow chart of adaptive trimap generation 57
4.9 Trimap propagation 62
4.10 Video matting comparison–Sequence 1 64
4.11 Video matting comparison–Sequence 2 65

List of Acronyms

MAP	Maximum A Posteriori
LPP	Locality Preserving Projections
TPS	Thin-Plate Splines
MRF	Markov Random Field
PDF	Probability Distribution Function
GMM	Guassian Mixture Model
UI	User Interface
MSE	Mean Squared Error
BP	Belief Propagation
CG	Conjugate Gradient
ML	Maximum Likelihood
CRF	Conditional Random Field
SAD	Sum of Absolute Differences
FC	Fuzzy Connectedness
FOV	Field of View
HDR	High Dynamic Range
HVS	Human Visual System

This thesis is dedicated to my parents

Acknowledgement

My sincerest gratitude goes first to my research supervisor Dr.Jiying Zhao, who offered his continuous support, advice and encouragement throughout my research work. He is always kind and encouraging, and taught me a lot during my research work.

My sincere thanks also go to my colleagues Zhenyi Luo and Wenyi Wang, for offering valuable advice in my research work.

I would like to express my gratitude to Dr.Yu Peng and Dr.Xuezhi Xiang, for their kindly help and support for preparing experimental facilities.

I am thankful to the support of my colleagues Yu Zhang and Han Wang. We studied together and shared ideas, which made me learn a lot.

I would like to take this opportunity to express my sincere appreciation to all those who, directly or indirectly, have lent me their helping hand these two years.

Lastly, and most importantly, from the deepest part of my heart, I express my sincerest Gratitude to my parents, for their perpetual love and spiritual and emotional support throughout my life.

Chapter 1

Introduction

1.1 Image segmentation

Image segmentation refers to partitioning a digital image into different segments. It is an essential technique in image processing and computer vision. This technique groups diverse elements into several categories based on their properties, and those properties can be gradience/color values, textures, or the significant and insignificant areas defined by users.

Image segmentation is becoming an interdisciplinary subject and plays a crucial role in various fields, such as:

- Medical Imaging [1], including measurement of tissue volumes, locating and diagnosis of tumors and other pathologies, etc;

- Face, iris and fingerprint recognition;

- Machine vision and traffic control systems;

- Agricultural imaging and pattern identification in satellite images.

However, there is no standard criterion or solution that can be generally applied

in various situations. Each of the existing algorithms is normally designed for specific problems. That is to say, to effectively solve problems in different areas, people need to combine this technique with related professional knowledge. So far, the existing algorithms can be categorized into five types [2] :

Edge-based methods

These methods are based on the assumption that there is often a sharp adjustment in gray scale or intensity near boundary regions. It segments object based on edge detection techniques and requires the object to have a closed region boundary.

Region-based methods

These methods often require additional inputs, such as seeds or seeded region. The regions grow iteratively by comparing the similarity between current pixel and other unallocated neighboring pixels, until all pixels are allocated to a certain region.

Thresholding methods

The key of these methods is to select the threshold values. It turns a gray-scale image into different regions according to their intensity levels.

Clustering methods

This technique assigns a set of objects into groups (known as *clusters*). Pixels are assigned into the same cluster if they have the same color and/or the same texture. Then the image is represented in terms of clusters of pixels.

Histogram-based methods

This technique computes a histogram from all of the pixels in the image, and the peaks and valleys in the histogram are used to locate the clusters in the image [3].

1.2 Foreground extraction and image composition

Foreground Extraction is a typical application of image segmentation technology. It normally works with *Image Composition* to separate the required foreground object from an image and then compose it with another background to generate a new scene. It is a vital technique in *Visual Effects* such as film making, advertising and television broadcasting. Figure 1.1 is an example of foreground extraction and image composition.

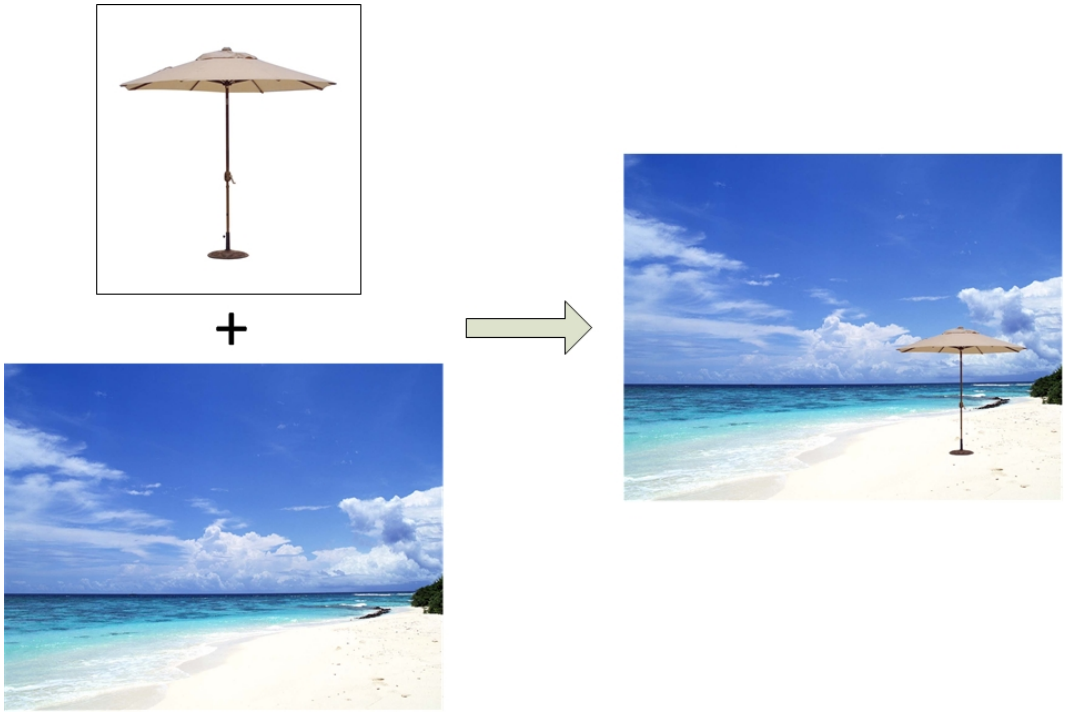


Figure 1.1: Foreground extraction and image composition.

However, the foreground objects are not always solid as in Figure 1.1. They may contain fuzzy edges, transparent or semi-transparent areas. And some of those parts normally exist at scales below the resolution of a camera, which means some details are even smaller than one pixel. In that case, the color and light of pixels may be affected by more than one element [4]. Consequently, the boundary information is not

enough to describe the properties of the foreground. Figure 1.2 shows the extracting result of a hairy (semi-transparent) object based on its boundary. The result suffers from "color-spill".

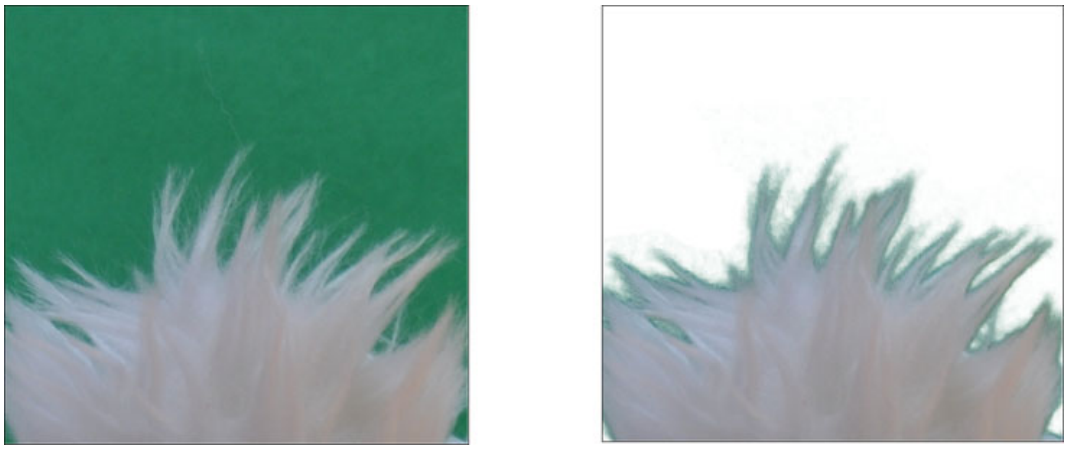


Figure 1.2: The extracted object suffers from "color-spill" in hairy regions.

1.3 Alpha matting

For those pixels with *mixed* colors, a measurement is needed to weigh the influence of different elements. Porter and Duff [5] introduced a concept of *Alpha Channel* in 1984 and mathematically modelled this problem by using a linear interpolation of foreground and background colors.

The alpha channel stores the opacity information with the values between 0 and 1. $\alpha = 0$ stands for a completely transparent foreground, in other words, those pixels do not have any coverage information and there is no color contribution from any geometry; $\alpha = 1$ means the pixel is opaque. For those semi-transparent areas, alpha values are between 0 and 1.

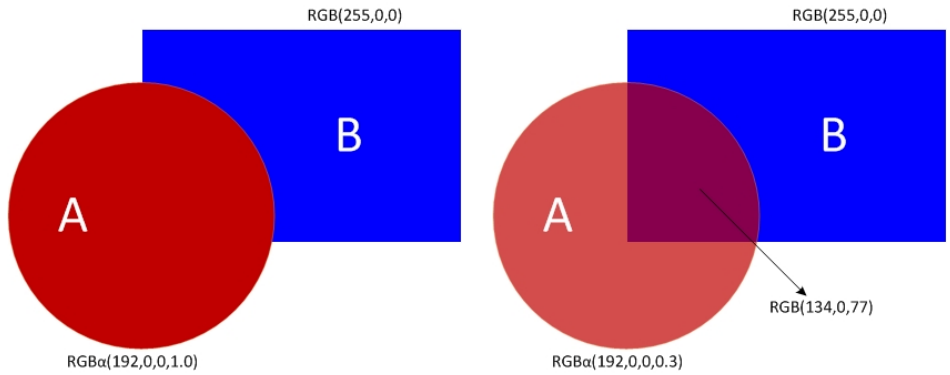
A monochrome raster image which interprets the alpha value of each pixel is known

as a *matte*. Therefore, fully separating the foreground from the background becomes a problem of determining pixel coverage and estimating the opacity (alpha values) for mixed pixels, also known as *extracting a matte*. Mathematically, the observed image $I_c(c = (x, y))$ is modelled as a linear combination of foreground image F and background image B by an alpha matte:

$$I_c = \alpha_c F + (1 - \alpha_c)B, \alpha \in [0, 1]. \tag{1.1}$$

If $0 < \alpha_c < 1$, we call pixel c *mixed*.

In Figure 1.3, A and B are two image elements, A appears in the foreground and B in the background. A can opaquely overlay B or with opacity of 0.3.



(a) Opaque A and B (b) Partially transparent

Figure 1.3: Elements overlaying.

Once those values are estimated, we can compose the foreground onto a new background, as shown in Figure 1.4.

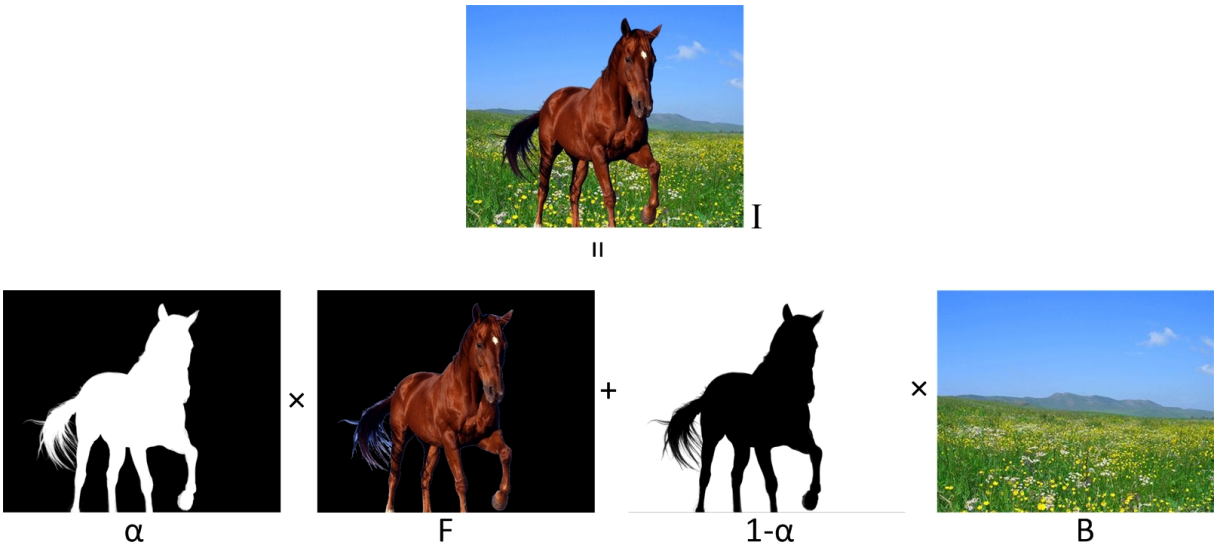


Figure 1.4: Convex combination. Image I is modeled as a convex combination of a foreground image F and background image B.

1.4 The contributions of the thesis

This thesis presents an improved sampling-based alpha matting algorithm for digital images and effectively extends it to video sequences. This approach provides high-quality foreground extraction results which can be seamlessly composed onto another background to generate a new scene.

The main contributions of the work are as follows: We compared in detail the existing sampling algorithms and analyzed their limitations, and we adopt a global-ranged sampling matting algorithm according to He et al.'s research work [6]. Based on this, we proposed a more advanced sample selection criteria which lead to more accurate matting results. Our approach overcomes the drawbacks in previous methods that the mixed pixels are sometimes erroneously estimated by using improper samples in some complex situations.

Moreover, we successfully extended the matting algorithm from still image to video sequences. We proposed a fully automatic process to generate adaptive and more accurate trimap to further improve the matting quality. Additionally, this automatic process highly enhances the efficiency since it does not need manually generating trimaps or scribbles.

We also illustrated in detail the implementation procedure of the modified application of Thin-Plate Spline technology in accurate trimap propagation, which is not specified in [7].

1.5 Thesis structure

This thesis is organized in five chapters. In Chapter 1, we introduce the basic idea of image segmentation, foreground extraction, and the concept of alpha channel and alpha matting. In Chapter 2, we review different kinds of alpha matting approaches and sampling methods. Chapter 3 analyses the failure reasons of previous sampling-based matting approaches and presents our improved global-ranged matting algorithm. In Chapter 4, we extend this algorithm to video. We analyze the challenges in video and propose our solution and implementation. In Chapter 5 we conclude the thesis and give some suggestions and ideas for future work.

Chapter 2

Previous work on alpha matting

The technology of alpha matting has been extensively studied and various matting algorithms and systems have been proposed. Generally, there are hardware based methods and software based methods. Hardware based methods usually rely on additional information provided by special equipment. Those additional information effectively enhance the efficiency, especially when dealing with video sequences. But the high cost makes them impossible to be widely used. By contrast, software based methods do not rely on special equipment. They directly deal with images or video. More specific, the software based methods can be categorized into two groups: affinity-based and sampling-based.

In this chapter, we will briefly introduce the hardware based methods and then focus on the software based methods, according to Wang's survey [8] and our study.

2.1 Hardware based (Matting with extra information)

Since the alpha matting technique is playing an increasingly significant role in various industries, many types of equipment are specially designed for solving the matting problem.

Matting with Camera Arrays

A system introduced in [9] uses a camera array to capture a collection of images, as shown in Figure 2.1. The central camera is defined as the reference camera and it computes the alpha mattes using information from the other cameras. The key idea behind this is the relative parallax between images, because the distance between the foreground and background allows foreground objects to be captured in front of different parts of the background. In this way, a synthetic aperture image which is focused on the foreground object is created. It reduces the variance of pixels re-projected from the foreground and increase the variance of pixels re-projected from the background. Then the final mattes are generated.

Based on the special equipment, this system has been demonstrated to be efficient, and the light computation enables it to pull the alpha mattes in almost real-time. Besides, it performs well on complex scenes such as hairy and transparent objects. However, limitations also exist. First, the accuracy is limited by aliasing in light fields. One the other hand, the variance of the background is considered as several orders of magnitude and is larger than that of the foreground, but it may not be true in some cases.

Flash Matting

Sun et al. [10] proposed a flash matting system which extracts mattes using flash/no-

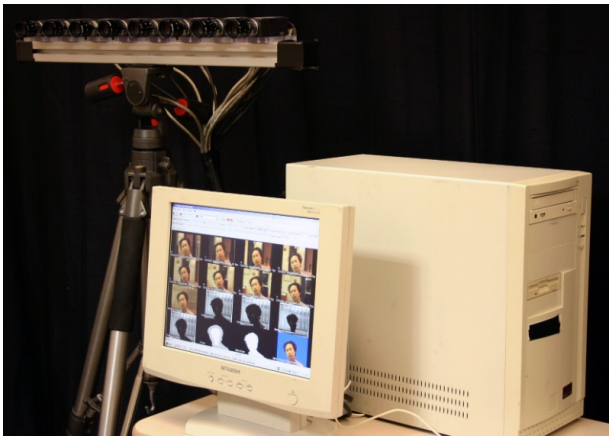


Figure 2.1: Camera array and real-time system. A linear array of 8 video cameras is used. Image is from [9].

flash image pairs. This system is based on the simple observation that only the foreground object has the most noticeable difference between the flash and no-flash images if the background is sufficiently distant. A joint Bayesian flash matting algorithm is also developed to recover the matte from flash/no-flash images when the background is complex or the foreground and the background have similar colors .

In Figure 2.2, (a) shows the non-flash image

$$I = \alpha F + (1 - \alpha)B, \tag{2.1}$$

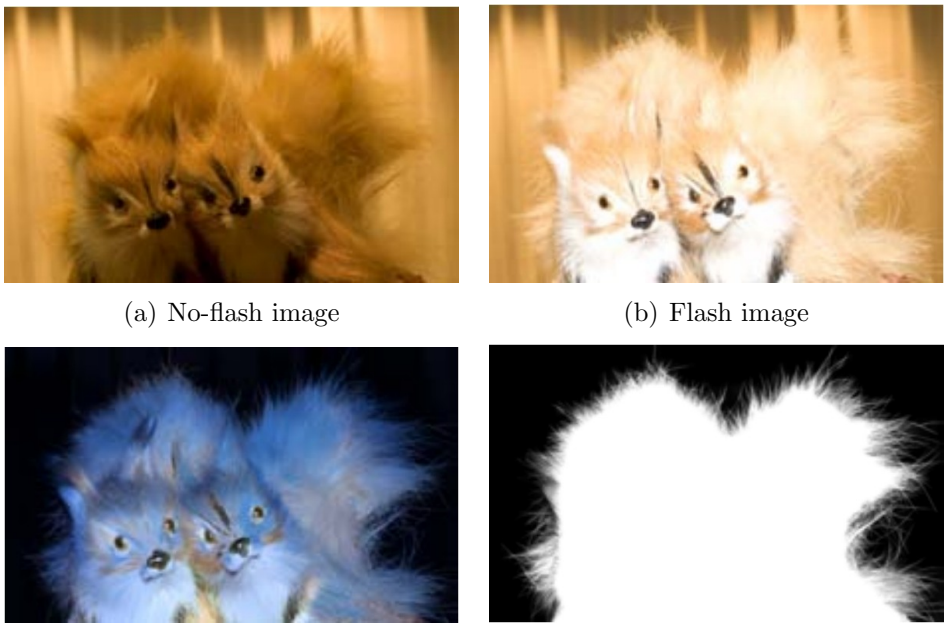
and (b) is the flash image

$$I^f = \alpha F^f + (1 - \alpha)B^f. \tag{2.2}$$

(c) is the flash-only image

$$I' = I^f - I. \tag{2.3}$$

This approach can generate good alpha mattes even when foregrounds and backgrounds are complex. However, an obvious limitation is the assumption that only the



(a) No-flash image (b) Flash image

(c) Flash only image (d) Extracted matte

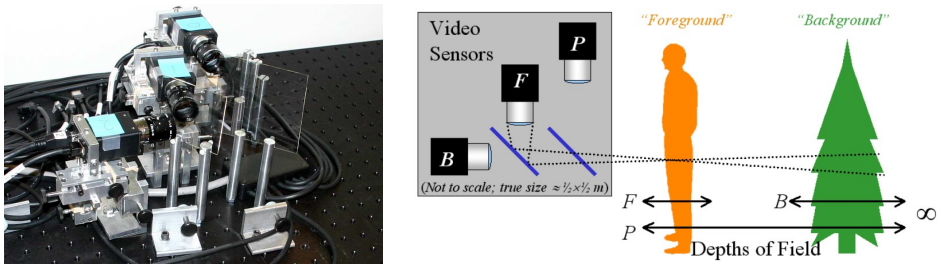
Figure 2.2: Foreground flash matting. Images are from [10].

foreground appearance is significantly changed by the flash, which is not always true in practice. Moreover, this approach also assumes that the image pair is pixel-aligned. Therefore, it is difficult to extract high-quality mattes when tiny objects have moved between two images in the time interval.

Defocus Matting

An approach using multi-sensor camera is developed in [11], named as *defocus matting*. They introduced an imaging device that records additional information during the capturing and the additional information comes from *defocus*. Three pixel-aligned video sequences are recorded in different focusing distance and depth of field. Figure 2.3 shows the multiparameter video camera and its application for defocus matting.

All sensors share a virtual optical center with varying parameters because of the beam splitters. The pinhole sensor has a small aperture which produces a large depth of field. The foreground and background sensors have larger apertures which create



(a) Multiparameter video camera (b) Illustration of the multiparameter video camera used for defocus matting

Figure 2.3: Multiparameter video camera and defocus matting. Images are from [11].

narrow depths of field. The foreground sensor focuses objects within around 0.5m of depth and defocuses objects far away. The background sensor focuses objects from around 5m to infinity and defocuses foreground objects. Given all three streams, for each frame, the matte is solved by a rather complex optimization process. To speed the process, they generate trimaps automatically using depth-from-defocus and choose the initial values for those unknown parts. Then the foreground and background colors and the final matte are calculated by solving the optimization problem with a gradient descent solver.

This work realized the application of a multiparameter camera to the video matting problem and allows each frame to be computed independently. Thus frames can be processed in parallel, which enhances the efficiency of the system. However, limitations also exist. As mentioned by the author, defocus matting is poorly conditioned when the images are underexposed or overexposed. On the other hand, the sensors behind two beam splitters only receive 25% of the incident light so they require stronger illumination than normal cameras.

2.2 Affinity-based methods

Affinity-based methods are software based. For the input images or frames, they assume foreground and background colors are locally smooth and treat the problem as interpolating the unknown alpha values from the known regions based on the predefined various affinities between neighboring pixels.

Poisson matting [12] operates directly on the gradient of the matte. This algorithm is based on the assumption that the intensity change in the foreground and background is smooth and thereby the gradient of the matte matches with the gradient of the image, then the matting problem can be formulated as solving Poisson equations. First, an approximate gradient field of matte is computed by taking the partial derivatives on both sides of the matting equation:

$$\nabla Iz = (Fz - Bz)\nabla\alpha_z + \alpha_z\nabla Fz + (1 - \alpha_z)\nabla Bz, \tag{2.4}$$

where $\nabla = (\frac{\partial}{\partial x}, \frac{\partial}{\partial y})$ is the gradient operator. Since Fz and Bz are assumed to be smooth, $\alpha_z\nabla Fz + (1 - \alpha_z)\nabla Bz$ is relatively small with respect to $(Fz - Bz)\nabla\alpha_z$. An approximate gradient field of matte is computed as:

$$\nabla\alpha_z \approx \frac{1}{Fz - Bz}\nabla Iz, \tag{2.5}$$

Then the final matte is computed by solving Poisson equations on the image lattice as:

$$\alpha^* = \arg\min_{\alpha} \int\int_{z\in\Omega} \|\nabla\alpha_z - \frac{1}{Fz - Bz}\nabla Iz\|^2 dz \tag{2.6}$$

with the Dirichlet boundary condition Eq. (2.7)(2.8) which is consistent with the pro-

vided trimap. Ω is the unknown region in the trimap, Ω_F and Ω_B are the foreground and background areas respectively.

$$\alpha|_{\partial\Omega} = \hat{\alpha}|_{\partial\Omega} \tag{2.7}$$

$$\hat{\alpha}_z|_{\partial\Omega} = \begin{cases} 1 & z \in \Omega_F \\ 0 & z \in \Omega_B \end{cases} \tag{2.8}$$

As we can see, based on this algorithm, an accurate $Fz - Bz$ value is essential to get the final alpha matte. However, it is hard to achieve for images with complex foregrounds and backgrounds. Failures in selecting samples could result in large errors in the final solution. Despite that a set of local filters and operations are defined to enable users to modify the final result, the manual adjustments in local areas are time-consuming for users.

A similar method which calculates the final alpha values by dealing with affinities is based on Random Walk [13]. The alpha value of each unknown pixel is set as the probability that a random walker starting from this point first reach a foreground pixel before striking a background pixel, when biased to avoid crossing the boundary of the foreground region. Mathematically, a weight is assigned between neighboring pixels to indicates the affinity among these nodes based on a typical function [14] as:

$$\omega_{ij} = exp(\frac{\|z_i - z_j\|^2}{\sigma^2}), \tag{2.9}$$

where z_i and z_j are vectors representing the RGB colors at pixel i and j, σ is a free parameter which can be automatically determined or manually set by the user. Then the Locality Preserving Projections(LPP) technique [15] is used to define a conjugate norm. The projections defined by LPP are given by the solution to the generalized

eigenvector problem of

$$ZLZ^T x = \lambda ZDZ^T x, \tag{2.10}$$

where D is a diagonal matrix, Z is a $3 \times N$ matrix with each z_i as a column and L is the graph Laplacian matrix given by

$$L_{ij} = \begin{cases} d_i & if \quad i = j \\ -\omega_{ij} & if \quad i \text{ and } j \text{ are adjacent nodes} \\ 0 & otherwise \end{cases} . \tag{2.11}$$

Denote the solution to the generalized eigenvector problem of Eq.(2.10) by Q, the final affinity is then defined as

$$\omega_{ij} = exp(\frac{(z_i - z_j)^T Q^T Q (z_i - z_j)}{\sigma^2}), \tag{2.12}$$

and lastly, the final alpha values are calculated according to the affinity by employing the random walk algorithm. The matrix L is decomposed into blocks corresponding to the known pixels $Pk = P_f \bigcup P_b$ and the unknown pixels P_u as

$$L = \begin{bmatrix} L_k & R \\ R^T & L_u \end{bmatrix} . \tag{2.13}$$

As illustrated in [16][17], the desired probabilities of unknown pixels belonging to a certain label are the solution to

$$L_u A_u = -R^T A_k, \tag{2.14}$$

where A_u is the vector of unknown alphas to be solved, and A_k is the vector which

encodes the boundary conditions of Eq.(2.8).

The closed-form matting [18] approach avoids the limitation of some previous methods that the final performance may be significantly lowered by inaccurate estimations. It derives a cost function from assumptions of local smoothness on foreground and background colors. The underlying assumption is that the foreground and background colors are smooth and can be fitted with linear models in small (typically 3×3 or 5×5) local windows, which leads to a quadratic cost function in α and can be minimized globally. Based on this assumption, alpha values in a local window ω can be represented as

$$\alpha_i = \sum_c a^c I_i^c + b, \forall i \in \omega, \tag{2.15}$$

where c stands for color channels, and a^c and b are constants in the window. The matting cost function is then defined as

$$J(\alpha, a, b) = \sum_{j \in I} \left(\sum_{i \in \omega_j} \left(\alpha_i - \sum_c a_j^c I_i^c - b_j \right)^2 + \epsilon \sum_c a_j^{c2} \right), \tag{2.16}$$

where a^c and b can be omitted from the cost function, yielding a quadratic cost in the α value:

$$J(\alpha) = \alpha^T L \alpha, \tag{2.17}$$

where L is an $N \times N$ matrix, whose (i, j)-th element is:

$$\sum_{k|(i,j) \in \omega_k} \left(\delta_{ij} - \frac{1}{|\omega_k|} \left(1 + (I_i - \mu_k)(\sum_k + \frac{\varepsilon}{|\omega_k|} I_3)^{-1} (I_j - \mu_k) \right) \right), \tag{2.18}$$

where μ_k is a mean vector of the colors in the window ω_k, $\sum_k$ is a covariance matrix, and I_3 is the 3×3 identity matrix. Then the optimal alpha matte is solved by

$$\alpha = \arg\min \alpha^T L \alpha. \tag{2.19}$$

This approach is capable of generating robust mattes for complex images, however, when the local smooth assumption is violated, the matting result is significantly influenced. And since no background and foreground colors are estimated explicitly, it may not be able to deal with fine details accurately.

In other algorithms such as [19][20][21][22][23], users make simple scribbles to specify the foreground and background regions. Then according to the Markov Random Field (MRF) theory or other probability calculating algorithms, other parts are assigned to the proper region by adjusting the weight and minimize the energy term. These methods are user-friendly but sometimes generate a coarse segmentation.

2.3 Sampling-based methods

For complex images with various local color distribution, the assumption of affinity-based methods that foreground and background colors are locally smooth often does not hold. Inaccurate results are sometimes generated.

Sampling-based methods directly estimate the alpha value at each mixed pixel, thus be more robust in complex images. Once the proper foreground and background samples are selected, the color information of the unknown pixels can be estimated and thereby the alpha values are solved. This method was first used in early *blue/green screen matting* (*chroma keying*) techniques, which is commonly used in the film industry. This technique photographs the subject against a constant-colored background so the background color is known as a priori knowledge and only the foreground colors and the opacity values need to be determined. We will talk more about it in our video matting

system in Chapter 4.

2.3.1 Classic models of sampling-based methods

Mishim [24] developed a chroma keying algorithm based on representative foreground and background samples (Figure 2.4). In this algorithm, there are two polyhedral approximations of a sphere centered at the average value $\bar{B}$ of the color of background samples in color space. And then the vertices of the background polyhedron are repositioned by moving them along several lines radiating from the center to make the polyhedron as small as possible but still contains all the background samples. Similarly, the vertices of the foreground polyhedron are adjusted to generate the largest possible polyhedron which does not contain any provided foreground samples. For any new composite color C, a ray is casted from $\bar{B}$ through C. The intersections with the foreground and background polyhedra are defined as F and B respectively. The α value is then defined by the fractional position of C along the line segment FB.

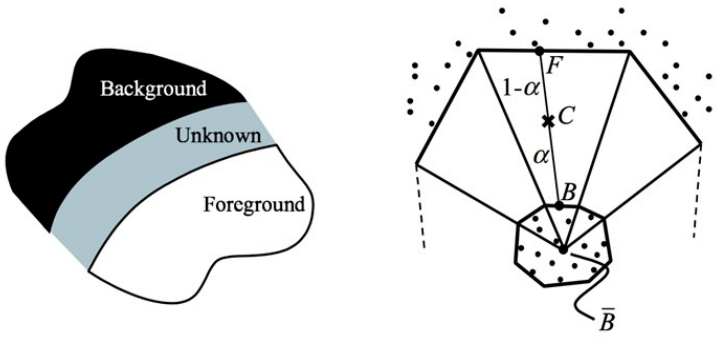


Figure 2.4: Mishima's algorithm. Diagrams are from [25].

Ruzon and Tomasi's approach [4] requires a clear separation of foreground, background and unknown region defined by users. This approach segments the unknown region into several sub-regions (Figure 2.5), and for each sub-region, a box is created

to encompass the sub-region. It also includes some of the nearby foreground and background regions. Then the foreground and background pixels included in the box are treated as samples from distributions of $P(F)$ and $P(B)$ respectively. The foreground pixels are categorized into coherent clusters, and each of them are modelled as isotropic Gaussians with mean F and diagonal covariance matrix $\sum_F$. Then the distribution of the whole foreground set is treated as a Gaussian mixture model (GMM). The background samples are operated with the same method and every foreground cluster is paired with background clusters. Finally, the observed color is considered as an element in an intermediate distribution $P(C)$, which is also defined as a sum of Gaussians. Each Gaussian has the mean value C and interpolated covariance $\sum_C$, The optimal alpha value is then generated when the intermediate distribution for the observed pixel has maximum probability. However, this method still has problems since Gaussian model may have large fitting errors in textured regions.

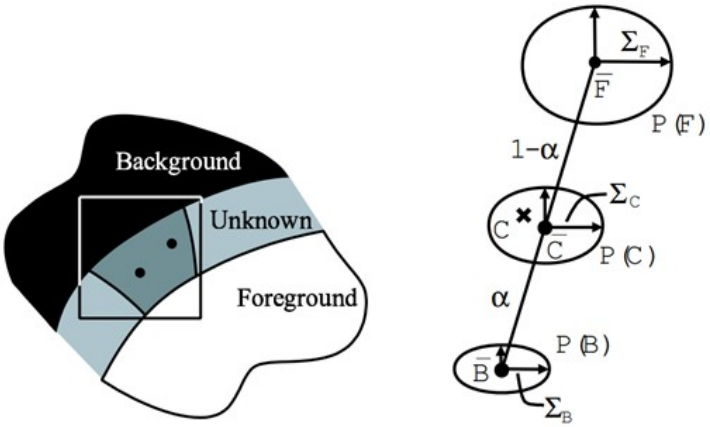


Figure 2.5: Ruzon-Tomasi's algorithm. Diagrams are from [25].

The Knockout system [26][27] first extrapolates the foreground and background colors into the unknown region and then estimates α valus of pixels in the unknown region

according to the given information. Particularly, for an mixed pixel C, its foreground color F_c is computed as a weighted sum of its nearby foreground colors. The weights are proportional to their spatial distances to C. The weight for the nearest pixel is set to be 1 while the weights for those pixels which are twice as distant as the nearest pixel are set to 0. Similarly, the background color B'_c is initially estimated based on nearby background samples, and refined later by considering the relative position of F, B and C in color space.

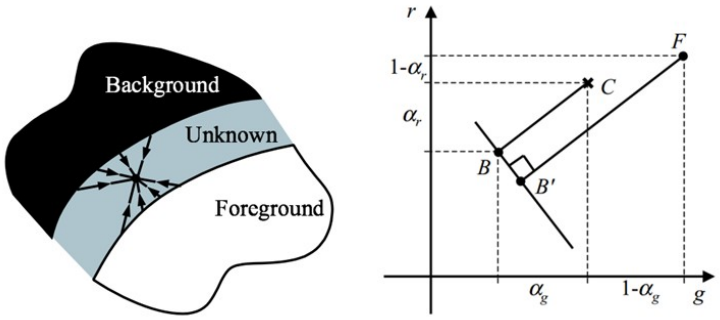


Figure 2.6: Knockout algorithm. Diagrams are from [25].

Figure2.6 shows a set of samples which contributes to the F and B calculation of a pixel in the known region. The alpha value is computed according to the estimated F and B by

$$\alpha = \frac{f(C) - f(B)}{f(F) - f(B)}. \tag{2.20}$$

For instance, in *green* channel the alpha value is estimated as $(g(C) - g(Bc))/(g(Fc) - g(Bc))$, where $g(.)$ is the color value in green channel. The final α is then calculated as a weighted sum of all these projection values in each color channel.

Chuang et al. [25] proposed a more advanced approach known as Bayesian matting

based on Ruzon and Tomasi's algorithm. This approach also models the foreground and background color distributions with varying sets of Gaussians. It creates a sliding window marching from foreground and background region to the unknown region to define neighborhood. In addition to the values from foreground and background regions, it utilizes nearby computed F, B and α values to construct the foreground and background Gaussian distributions, as illustrated in Figure 2.7. Furthermore, the matte parameters are computed by using the maximum a posteriori (MAP) technique in the proposed Bayesian framework. The MAP technique is closely related to maximum likelihood (ML). Then the optimal F, B and α values are calculated simultaneously. This technique is further extended to video sequences, as proposed in [28].

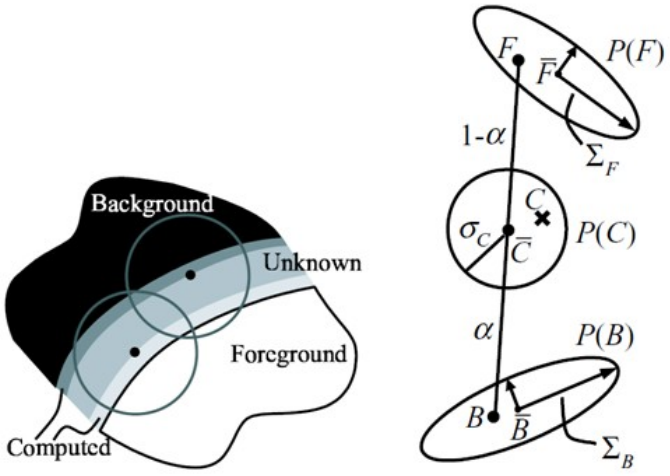


Figure 2.7: Bayesian matting. Diagrams are from [25].

Bayesian matting can generate accurate matting results when the given trimap is well-specified. However, when a coarse trimap is given, the correlations between unknown pixels and foreground/background samples are weak, which tends to generate noisy results. Furthermore, if the input image contains hightly-textured regions, then

using Gaussians is insufficient to model the high order statistics of color distribution.

2.3.2 Sample selection approaches

Some early sampling-based approaches normally collect a group of nearby foreground and background colors for alpha estimation. According to the linear model, they estimate alpha values by projecting unknown pixels onto the line between foreground and background cluster centers to fit the linear model, as shown in Figure 2.8(a).

However, they neglect the fact shown in Figure 2.8(b): compare with pixel C_2, pixel C_1 fits the linear model between F_1 and B much better than C_2, which means C_1 has high probability of being a true mixed pixel between the clusters F_1 and B. While for pixel C_2, there are two possible situations. First, it is very unlikely to be generated by a linear combination of the clusters F_1 B since it is far away from the interpolation line, and it is more likely to be an unmarked foreground or background pixel. Or, the other possible situation is, C_2 may not be an unmarked foreground or background pixel but a truly mixed pixel. However, there exists another cluster F_2 which makes C_2 fit the linear model much better, even it is farther to C_2 than F_1. Therefore, some previous approaches which just simply estimate an alpha value for C_2 based on its projection to line C_2' will lead to bad results.

Consequently, direct color sampling may be erroneous. Although Bayesian matting system utilizes Gaussian models for color fitting, it is still insufficient to determine proper samples which can best explain the mixed pixels.

Approaches like Bayesian matting and Belief Propagation(BP) [23] matting collect foreground and background samples in the marked known region which have the shortest spatial distance to the observed pixel. However, spatially nearest pixels are limited in a rather small range which can hardly cover all the possible colors of foreground and

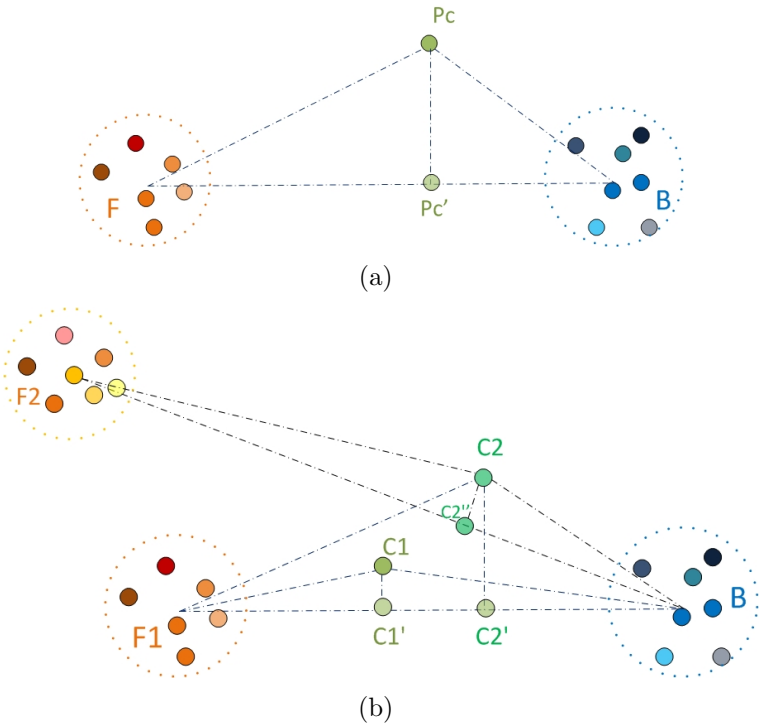


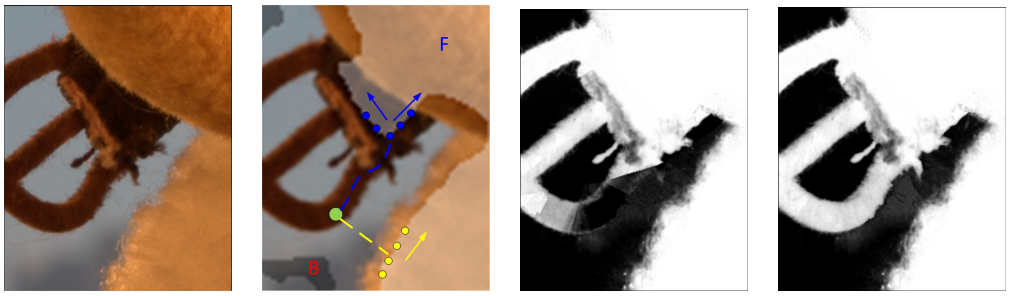
Figure 2.8: Examples of linear models.

background.

Robust matting [29] improved the sampling method by spreading the sampling along the boundaries of marked foreground and background regions. Therefore the sample set can better capture the variation of foreground and background colors.

However, this sampling method still has limitation in more complex object topologies. Improved color matting [30] pointed out this limitation and gave solutions by using "geodesic samples". Figure 2.9 shows examples given by [30]. In Figure 2.9(b), the green dot is an observed pixel in unknown region (denoted as C). Yellow sample set is collected from the spatially nearest pixels. However, as we can see, the pixels in this set are all brighter than C, which do not match the true foreground color of C. By contrast, blue set is collected from closest pixels in *geodesic distance*, whose colors are much closer to the observed pixel. The *geodesic distance* is defined as the shortest path

in a weighted graph (computed in the similar way in [31]) from an unknown pixel to the foreground boundary. Based on this, the blue path goes only through pixels that are likely to belong to the foreground object. Figure 2.9(c) is the result using spatial samples, and (d) gives better result using geodesic samples.



(a) (b) (c) (d)

Figure 2.9: "Geodesic samples" versus nearest sparse samples. Matting results are from [30].

Shared Matting [32] proposed another sampling method which collects samples by shooting rays from observed pixels to background and foreground, then the nearest foreground and background samples are collected along the lines, as illustrated in Figure 2.10. Each unknown pixel collects just a few samples but samples are shared with each other among neighboring pixels. Therefore, the final sample set of an unknown pixel is roughly considered as a combination of samples of its own and its neighbors.

However, limitations in the above methods are also obvious. Their performance tend to degrade when foreground and background patterns become complex or just have some special topological layout.

2.4 Other methods

Other methods such as proposed by Iverson [33] and Uya [34], use one-dimensional histogram to separate foreground and background. They only encompass a rectangle

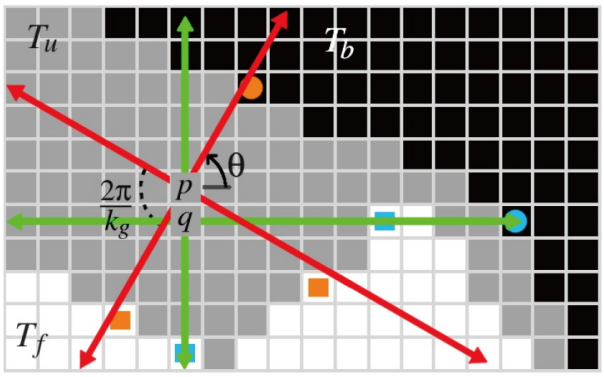


Figure 2.10: Sampling method in Shared Matting. Red arrows start at p define the paths for searching for foreground and background samples for p, and the nearest samples along this path are selected (marked in orange); Similarly, blue paths and samples are for pixel q. Diagram is from[32].

space to represent background color in color space, which normally fail in practice since the color distribution are usually more complicated. Miller [35] deals with the keying problem in a linear fashion. They use a circle to represent the background color area and a polygon to separate foreground and background regions. But this linear model is not robust enough when dealing with more textured or noise images. Yamamoto and Yonemitsu [36] further improved this approach by generating key signal from quadratic curve groups in the U-V chroma coordinates. Besides, there are many other color models ([37][38][39][40]) proposed to consider the foreground and background color distribution problem in three dimensional color space instead of two dimensional chromatic plane.

Chapter 3

The improved sampling-based alpha matting for digital images

As introduced in Chapter 2, although a large quantity of successful approaches for image matting have been proposed, none of those can be robust enough in diverse situations. Even with the help of special equipment, limitations still exist. However, not all the defects are unavoidable, so optimized approaches still can be generated to avoid the defects in other methods and minimize the drawbacks as much as possible.

In this chapter, we will propose an optimized sampling-based matting approach using trimap. This approach provides an optimized sampling method, based on the idea from [6]. It collects samples in a global range and uses SampleMatch technology to enhance the computational efficiency. We propose modified cost functions acting as selection criteria, and that provides more reliable foreground and background sample candidates for estimation. And high-quality matting results are then generated.

First we will analyze why previous sampling methods fail in experiments, and then propose our improved sampling-based alpha matting approach.

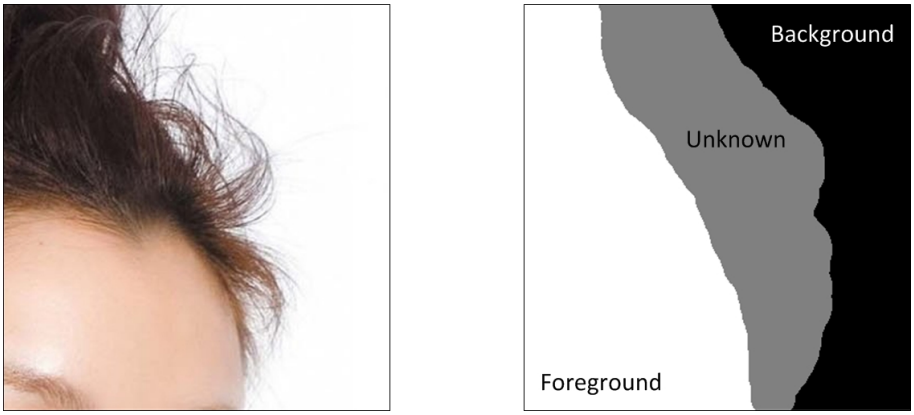
3.1 Failure examples in previous non-parametric sampling methods

According to the classic alpha compositing formular $C = \alpha F + (1 - \alpha)B$, once proper foreground samples F and background samples B are selected, the color information of the observed pixel C can be determined. Therefore, good selection of candidate samples will significantly influence the final result.

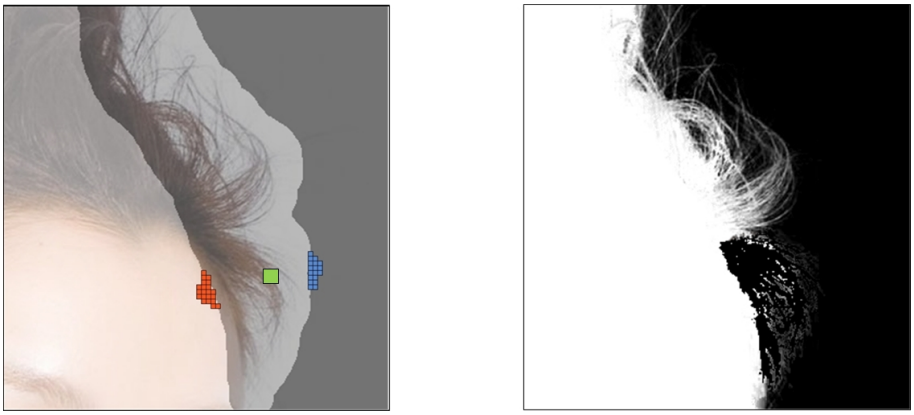
As we have introduced in Chapter 2, spatially nearest pixels are usually insufficient to provide true foreground and background color information. In contrast, the spreading sampling method better covers the foreground and background color distribution. Figure 3.1(b) shows sampling from nearest spatial neighbors and the generated alpha matte. Since the nearest samples do not contain any color information about the hair, the hair strands in the unknown region are failed to be estimated. Figure 3.1(c) shows the sample set collected along region boundaries. In this way, more foreground colors are captured and some of them provide the color information about the hair. The generated matte shows that this method performs better than the above.

When images become more complex, "geodesic samples" show their advantages by considering the weighted path from the unknown pixel to the foreground boundary, as has shown in Figure 2.9. Collecting samples by shooting rays from unknown pixels extends the limitation to even farther samples. But when foreground object contains complex topological layout, it tends to fail to reach good samples, as shown in Figure 3.2(f).

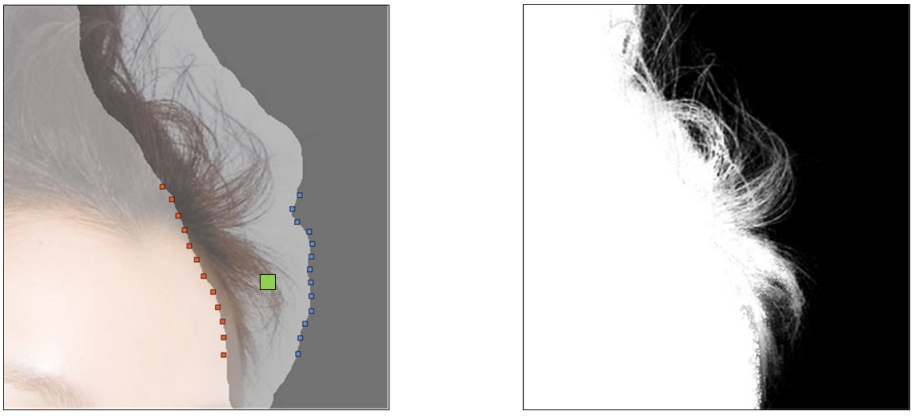
By comprehensively analyzing the sampling methods of Robust matting [29], Improved color matting [30] and Shared matting [32], their limitations can be summarized as:



(a) Original image and trimap



(b) Left: using nearest spatial neighbors. Right: the generated matte



(c) Left: using sparse samples. Right: the generated matte

Figure 3.1: Two sampling methods comparison. Red pixels represent foreground samples, blue ones represent background samples and the green point is the observed pixel in the unknown region.

For [29] and [30], although they do not point out, there is an implied assumption that the observed object is regarded as an extension from foreground in color space. In other words, the true foreground colors should be close to some of the collected samples. Therefore, when foreground and background regions contain significant discontinuities, especially when the color of the observed pixel only exist in the local unknown region, similar colors cannot be found in nearby foreground area, and direct color sampling may be erroneous. Although [30] gives more advanced sampling method than [29], still, the sample set is limited and often locally distributed. As for [32], since it uses lines across mixed pixel to pick samples, special shape will bring blind angles thus the true samples are still missed. Figure 3.2 gives an example that all the three sampling method failed.

Actually, in practice, it is not rare that significant discontinuities exit in the foreground and background regions. Figure 3.3 is another example that local foreground samples cannot provide reliable color information for the mixed pixels.

Therefore, only sampling in a global range can ensure that all the good samples are selected. However, this method faces a big challenge that the large sample sets bring a significant amount of computation. In the next section we will illustrate how to solve this problem according to the idea proposed in [6].

3.2 Alpha matting using global-ranged smaples

3.2.1 Randomized search using PatchMatch and SampleMatch

As we have mentioned in previous section, it is a challenging task to select suitable samples among a global-ranged set. Fortunately, according to the idea of PatchMatch

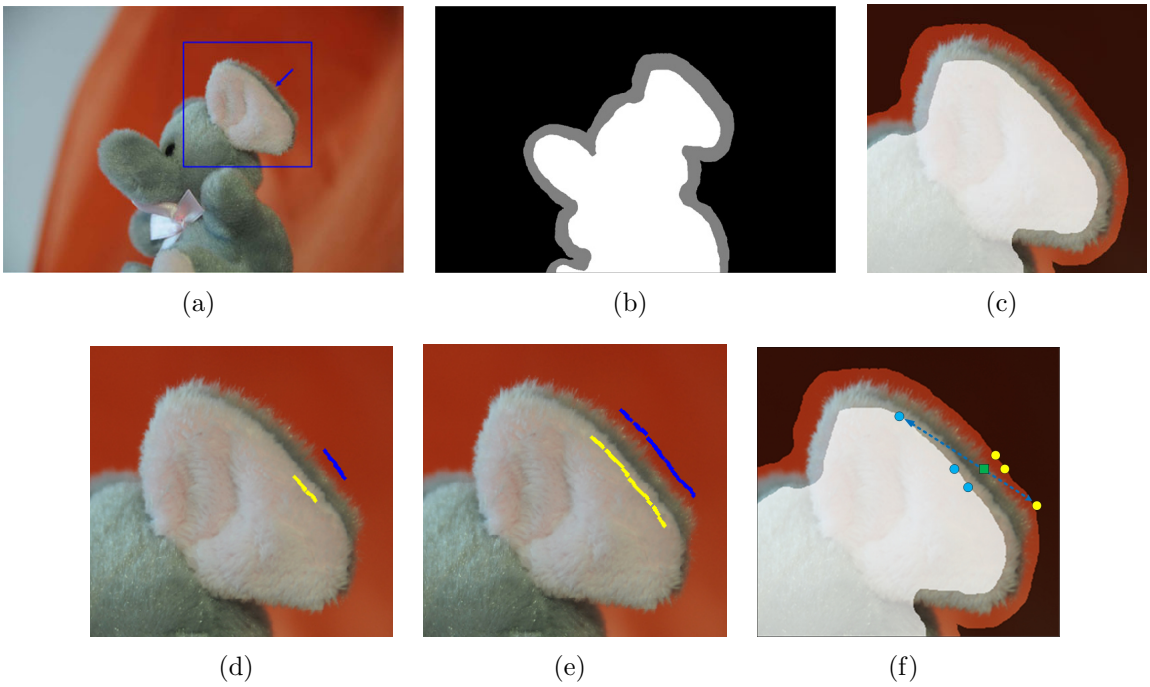


Figure 3.2: Failed sampling examples. (a) is the original image, (b) is the trimap and (c) is the cropped image with trimap. As we can see, in this area, the dark gray colors along the boundary of the elephant ear only exist in the unknown region. That is to say, for these unknown pixels we cannot find any clue from their nearby foreground colors. (d)-(f) shows that the selected samples of the three methods (Robust [29], Improved [30], Shared [32]) do not cover any color information of dark gray, so no true foreground sample can be used for further estimation.

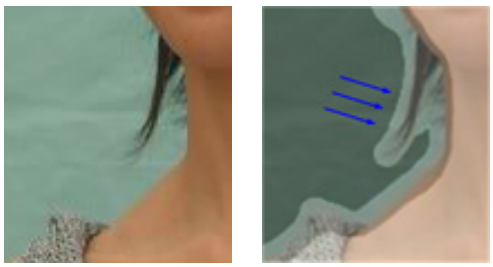


Figure 3.3: Another example that local samples are not reliable for estimation. According to different shooting angles, some hair strands are not connected with other parts. In this case, local foreground samples cannot provide true color information for mixed pixels.

[41] and SampleMatch [6], a randomized correspondence algorithm can be used to well address the challenging issue caused by huge sample sets.

As shown in Figure 3.4, all the foreground samples spread along the region boundary are collected and the sample set is denoted as **F**. Background sample set is collected in the same way and denoted as **B**.

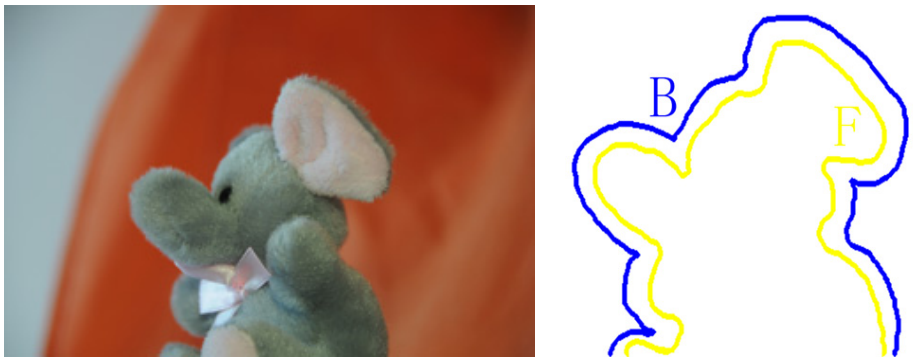
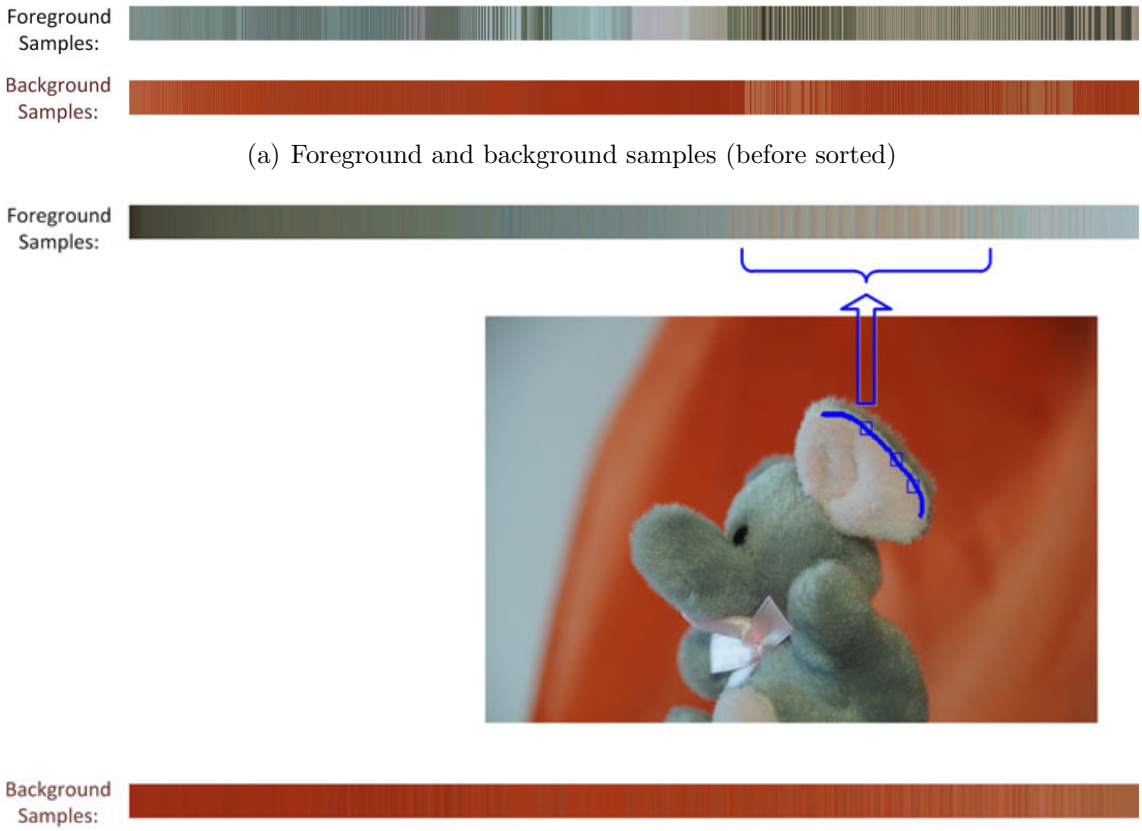


Figure 3.4: Global-ranged foreground and background sample set.

Then **F** and **B** are sorted respectively according to their color intensity values. The sorted sets are denoted as $\{B_i|i = 1, 2, 3, \cdots, N_B\}$ and $\{F_j|j = 1, 2, 3, \cdots, N_F\}$. Figure 3.5 shows that after sorted, pixels have similar color information are gathered together, which is essential for future search process.

A Foreground-Background search plane is then generated. The search algorithm runs iteratively in the plane. For each unknown pixel, the iteration proceeds in two steps: **Propagation** and **Random search**, to finally find out the suitable F-B pairs. Here, suitable F-B pairs are considered as foreground and background sample pairs which have the smallest cost. Bad samples are those which have very low color similarities to the unknown pixel and do not well fit the classic linear model, thus their cost will be high. How to compute the cost will be introduced in the next section.

For each unknown pixel I, if its current foreground and background sample pair



(a) Foreground and background samples (before sorted)

(b) Ordered foreground and background samples

Figure 3.5: Sort the foreground and background samples according to color intensity.

(F_I, B_I) has the smallest cost, we denote it as $\Psi_I(F_I, B_I)$. In practice, unlike [6] where they initialize Ψ_I by a random point in the search plane, we initialize Ψ_I by picking the nearest foreground and background samples since we still believe that in some cases, nearby samples can provide comparatively reliable estimation. Then Ψ_I is updated while the algorithm runs. Denote the two steps of **Propagation** and **Random Search** as P_k and S_k for pixel I_k, then for all the unknown pixels in the image, the algorithm proceeds in the order: P_1, S_1, P_2, S_2, P_3, S_3, $\cdots$, P_n, S_n.

Propagation.

We update Ψ_{I_k} by first considering the known optimal sample pairs Ψ_{I_t} of its neighboring pixels I_t (Figure 3.6), with the assumption that neighboring pixels tend to have similar foreground and (or) background colors. So if a pixel has found a good sample pair, this pair is likely to be suitable for its neighboring pixels as well.

Therefore, if I_t is a neighboring pixel of I_k and its sample pair also brings smaller cost for I_k, then

$$\Psi_{I_k} \Leftarrow \arg \min_{\Psi_{I_t}(F_{I_t}, B_{I_t})} \varepsilon(\Psi_{I_t}(F_{I_t}, B_{I_t})), \tag{3.1}$$

where '$\Leftarrow$' is an assignment operator, and ε is the cost of the current pair, which will be introduced in the next section.

Random Search.

After Propagation, the Ψ_{I_k} of the unknown pixel I_k being scanned is further updated by a sequence of searching trials. Since the background and foreground sample sets have been sorted, we use i, j to indicate the index of the current **F**-sample and **B**-sample of I_k. Then the searching trials are generated as:

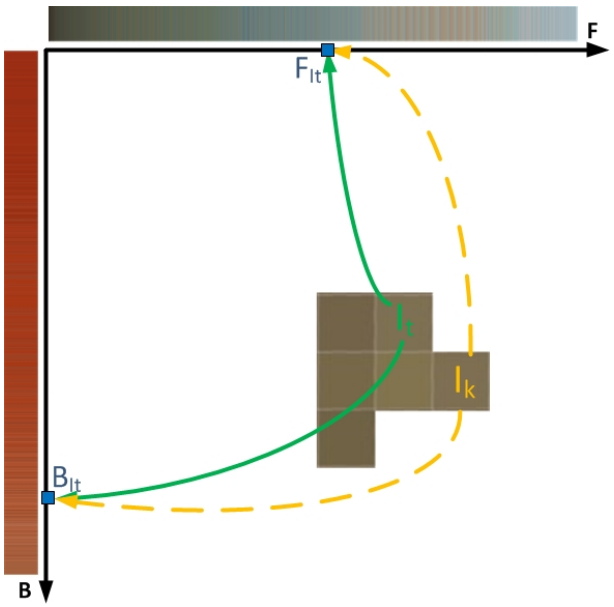


Figure 3.6: Propagation of neighboring pixels' sample pairs. For an unknown pixel I_k, if the sample pair of its neighboring pixel I_t (F_{I_t}, B_{I_t}) makes smaller cost than its own current sample pair, we consider (F_{I_t}, B_{I_t}) as the current optimal pair of I_k, i.e. Update $\Psi_{I_k}(F_{I_k}, B_{I_k})$ to $\Psi_{I_k}(F_{I_t}, B_{I_t})$.

$$(i_n, j_n) = (i, j) + \omega\lambda^n R_n, \tag{3.2}$$

where ω is the size of the search space, λ is a fixed ratio between search sizes, and λ^n is the $n-$th exponential of λ. R_n is a uniform random number in $[-1, 1] \times [-1, 1]$. In our applications, λ is set to be 0.5.

During this process, Ψ_{I_k} is updated if the new pair (F_{i_n}, B_{j_n}) brings a smaller cost. The random search step tests a sequence of candidate samples in the Foreground-Background search plane. The sequence goes in an ascending order and radius of the searching window $\omega\lambda^n$ decreases exponentially. The whole searching process stops until the search radius $\omega\lambda^n$ is below 1.

In [6], they consider the current sample pair as the final optimal one, since it already fall in the "optimal area". Because we have already ordered the sample sets, the good

pairs of a mixed pixel are grouped together in the search plane. Therefore, in a quite small range, most samples are appropriate for estimation. But to make sure we select the best samples, we continue to calculate the cost of the neighboring sample sets $\{F_n, B_n\}$ in the search plane and the neighboring sample sets $\{F'_n, B'_n\}$ in the image, to finally find out the good sample pairs.

In summary, since the searching trials are randomized, the number of iterations in a global range might be large in some cases. Therefore, the previous Propagation step gives a better starting point for the Random Search, which is helpful to reduce the required iterations. In the Random Search step, the search window is large at the beginning, and it determines the range of the local search area. As shown in Figure 3.5(b), the color of the inner ear is gathered in a certain range in the sorted **F**-sample set. When the search window is small, it guides the search inside the local area to find out the optimal sample pairs.

As we have mentioned, good sample pairs are considered as those which have the smallest cost. Next we are going to mathematically illustrate how to calculate the cost of sample pairs.

3.2.2 Selecting criteria of optimal sample pairs

According to the linear combination in Eq.(1.1), giving a sample pair (F_i, B_j) of a mixed pixel C, its alpha channel value can be estimated as

$$\hat{\alpha} = \frac{(C - B_j)(F_i - B_j)}{\|F_i - B_j\|^2}. \tag{3.3}$$

Good sample pairs should meet two criteria: (i). F_i and B_j should well fit the classic linear model as introduced in Section 2.3.2; (ii). They can provide reliable color

information for the unknown pixel, i.e. color similarities between samples and unknown pixels should be taken into account as well.

The authors in [6] define a color cost function

$$\varepsilon_c(F_i, B_j) = \|I - (\hat{\alpha}F_i + (1 - \hat{\alpha})B_j)\| \tag{3.4}$$

to describe how good a sample pair fits the linear combination. However, we follow the idea of [29] and define the color cost as

$$\varepsilon(F_i, B_j) = \frac{\|C - (\hat{\alpha}F_i + (1 - \hat{\alpha})B_j)\|}{\|F_i - B_j\|}. \tag{3.5}$$

Considering the situation shown in Figure 3.7, for the same d_c, the pair F_2B_2 performs a better linear fit than F_1B_1, so the color cost of F_2B_2 is smaller than F_1B_1. But according to He's method (Eq.(3.4)), the sample pairs F_1B_1 and F_2B_2 will generate the same cost. Therefore, the cost computed by Eq.(3.4) is not accurate enough to encode criterion (i). By contrast, according to Eq.(3.5), the computed cost of F_2B_2 is smaller than F_1B_1, which is more reliable.

Another situation that [6] does not consider is that some pixels in the unknown region actually belong to foreground or background. Thus pixels locate near foreground or background samples in color space are more likely to be foreground or background, and their alpha channels are either 0 or 1 (see Figure 3.8).

Therefore, for criterion (ii), we add two more weights $\omega(F_i)$ and $\omega(B_j)$ (similar as [29]) for each individual sample, to determine to what extent the observed pixel is likely to be fully foreground or background.

$$\omega(F_i) = 1 - exp(-\frac{\|F_i - C\|^2}{\mathcal{D}_F^2}) \tag{3.6}$$

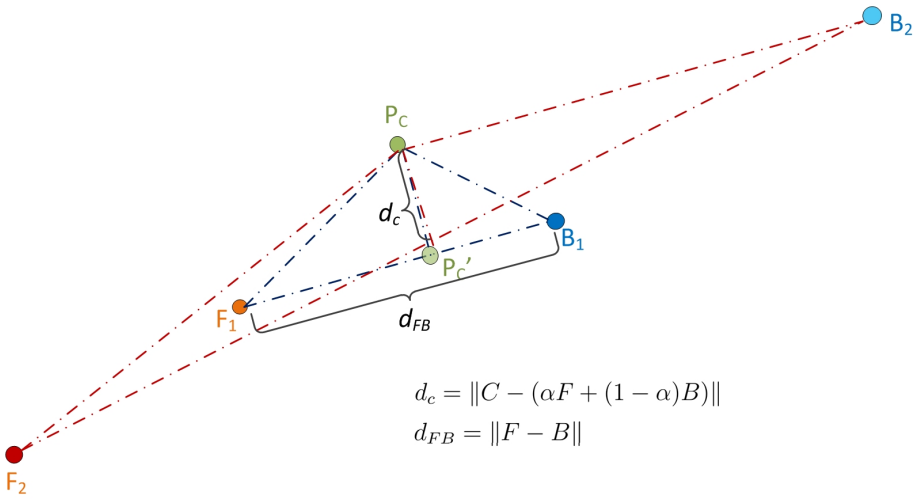


Figure 3.7: Compute the color cost. $d_c = \|C - (\hat{\alpha}F_i + (1-\hat{\alpha})B_j)\|$ and $d_{FB} = \|F_i - B_j\|$.

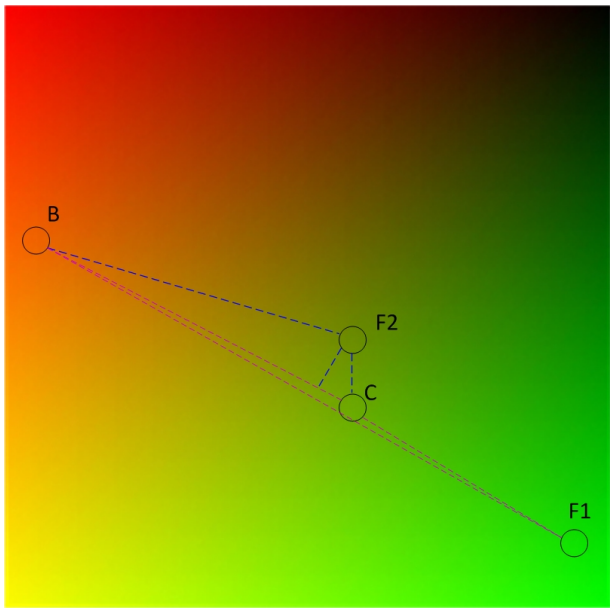


Figure 3.8: Encode criterion (ii) by considering the color weight. Although F_1B creates a good linear fit for pixel C, F_2 is actually a better foreground sample for C because it has nearly the same color as C.

$$\omega(B_j) = 1 - exp(-\frac{\|B_j - C\|^2}{\mathcal{D}_B^2}), \tag{3.7}$$

where $\mathcal{D}_F = \min_i \|F_i - C\|$ and $\mathcal{D}_B = \min_j \|B_j - C\|$ are the minimum distances between foreground/background sample and the current pixel in color space.

Then the total cost in color space is defined as

$$\mathcal{E}_c(F_i, B_j) = \varepsilon(F_i, B_j)\omega(F_i)\omega(B_j). \tag{3.8}$$

Besides, spatial distance also should be considered due to the large size of the sample set. For pairs which have the same performance, we only pick up the nearer ones in case that a false pair occasionally explains the color of the unknown pixel. But unlike [6], we define the spatial cost as

$$\mathcal{E}_s(F_i) = 1 - exp(-\lambda\|\chi_{F_i} - \chi_I\|/D_F) \tag{3.9}$$

$$\mathcal{E}_s(B_j) = 1 - exp(-\lambda\|\chi_{B_j} - \chi_I\|/D_B), \tag{3.10}$$

where χ_{F_i}, χ_{B_j} and χ_I are the spatial coordinates of the foreground/background sample and the unknown pixel, $D_F = min_i\|\chi_{F_i} - \chi_I\|$ and $D_B = min_j\|\chi_{B_j} - \chi_I\|$ are the nearest spacial distance between foreground/background sample and the current pixel. λ is a free parameter and set to be 0.1 in our system.

Combining all the above factors, we calculate the final cost of a sample pair (F_i, B_j) as

$$\mathcal{E}_{final}(F_i, B_j) = \gamma\mathcal{E}_c(F_i, B_j) + \beta(\mathcal{E}_s(F_i) + \mathcal{E}_s(B_j)), \tag{3.11}$$

where γ and β are free parameters and can be set to 100 and 1.

Figure 3.9 compares the results of using local samples and global-ranged samples.

Local samples failed to provide true foreground color information since the nearby foreground pixels are not coherent with unknown pixels in color space, while the global sampling method effectively finds out more proper candidate samples.

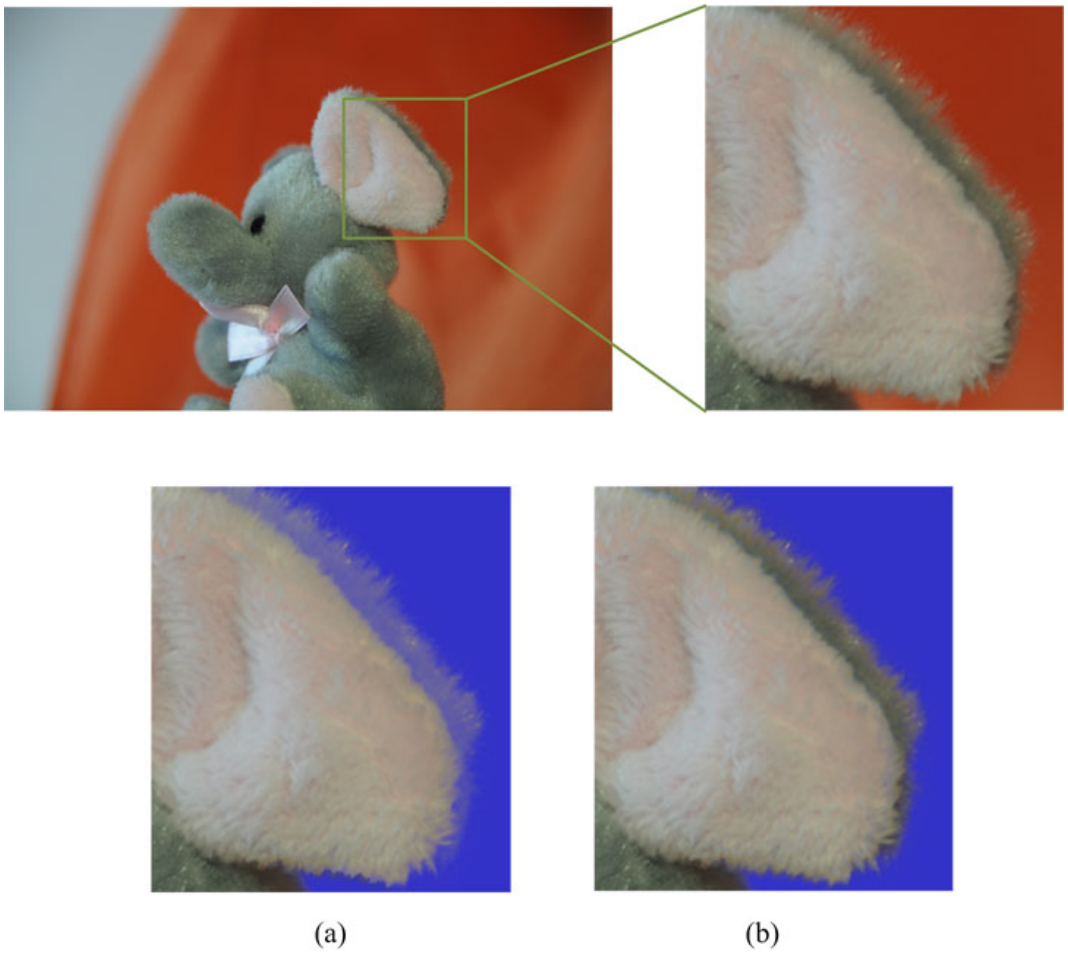


Figure 3.9: Matting results using local samples and global samples. (a) Using local samples. (b) Using global samples. In order to make it more obvious, we compose the results onto a blue background.

3.3 Experimental results

In this section we display more experimental results. In Figure 3.10, the first row are the original images from the evaluation database [42] and the second row are the extracted

foreground objects using our method.

The most difficult parts for alpha matting are: hairy boundaries, tiny single semi-transparent objects and areas where the color and texture of foreground and background are very close. Figures 3.11, 3.12, 3.13 and 3.14 compare our matting results with other algorithms in detail, especially those difficult parts.

In Figure 3.11, bad samples are normally selected by other methods due to the complex boundaries and the color/texture similarity between foreground and background. In Figure 3.12, single hair strand is fine and semi-transparent. They are easily affected by the complex background color and pattern. Those hair strands are usually missed and hard to be segmented out. In Figure 3.13 and Figure 3.14, some nearby samples provide bad estimation for the mixed pixels.

Figure 3.10: Experimental results (still images).

3.4 Summary

In this chapter, we proposed an improved alpha matting algorithm. We collect the foreground and background samples along region boundaries, and sort them according

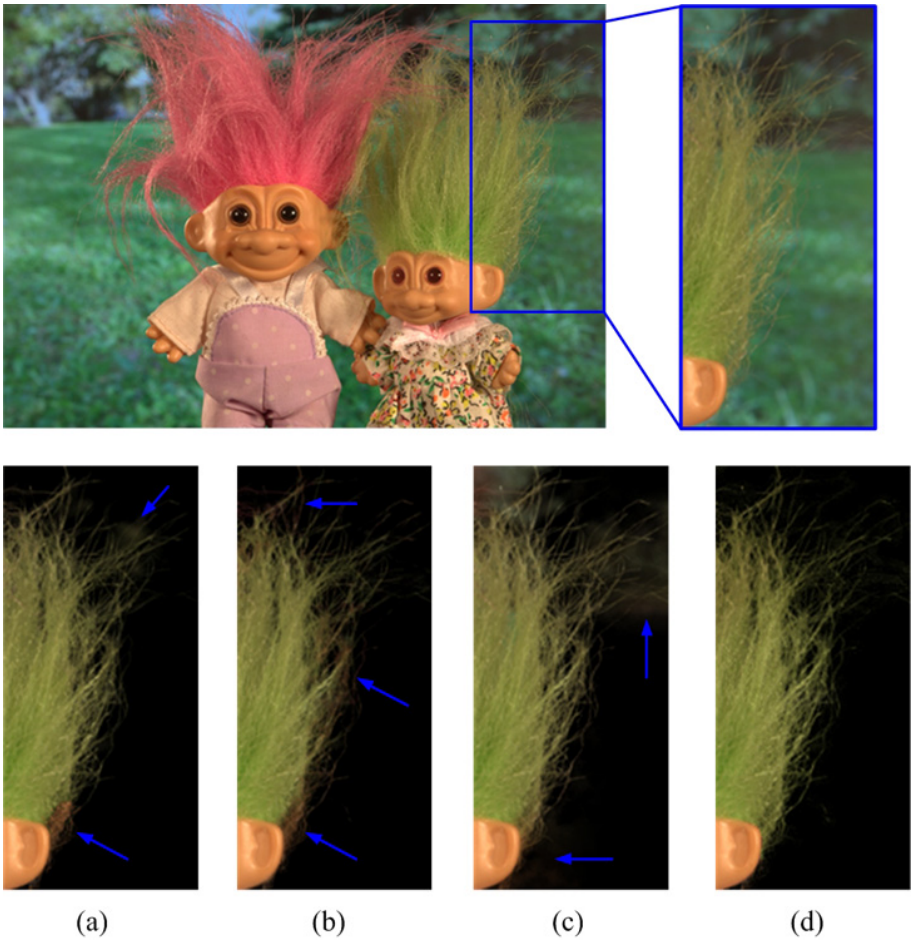


Figure 3.11: Matting result comparison 1. (a) Robust matting [29]; (b) Shared matting [32]; (c) Closed-form matting [18]; (d) Our method.

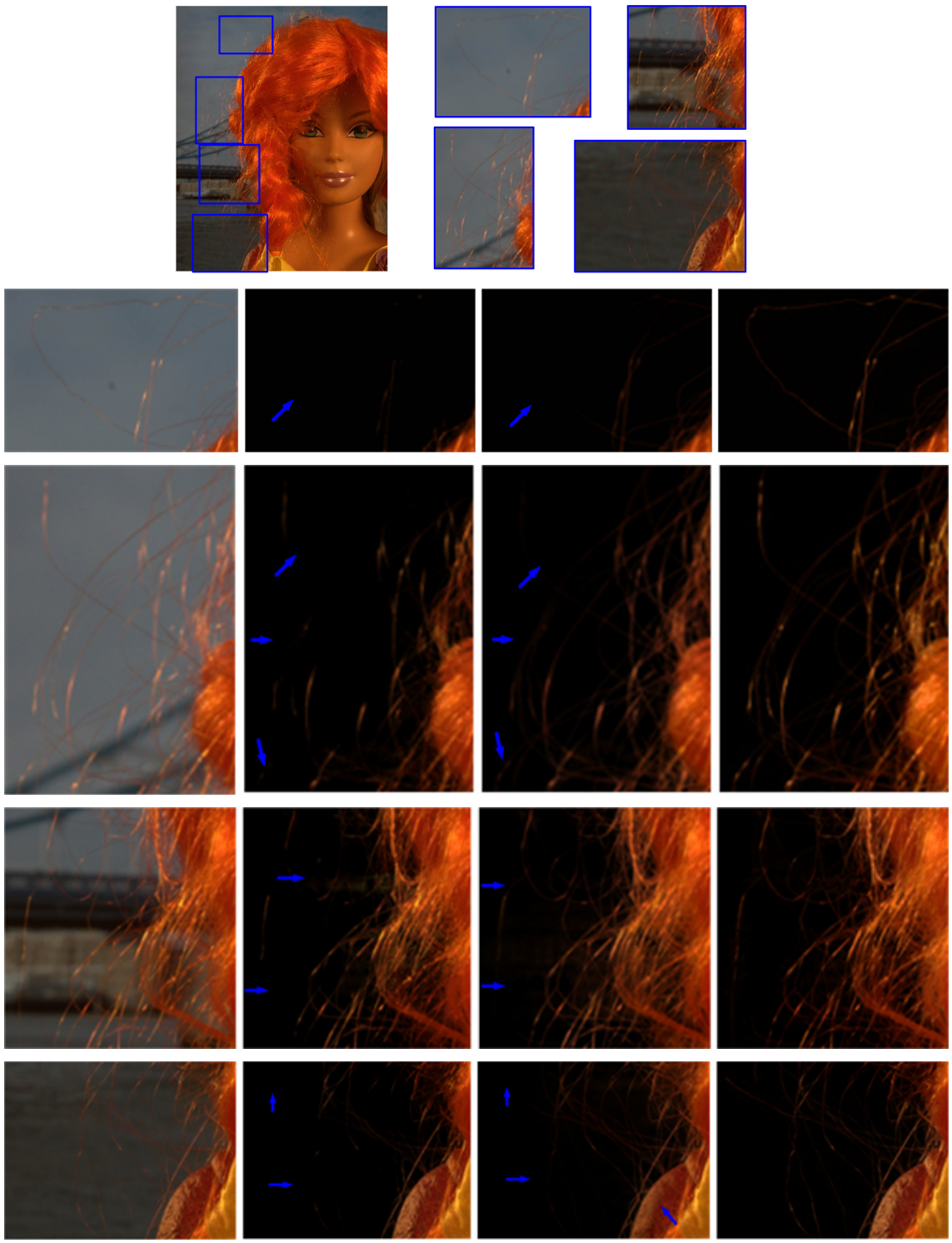


Figure 3.12: Matting result comparison 2. First column: Original cropped images; Second column: Robust matting [29]; Third column: Closed-form matting [18]; Fourth column: Our method.

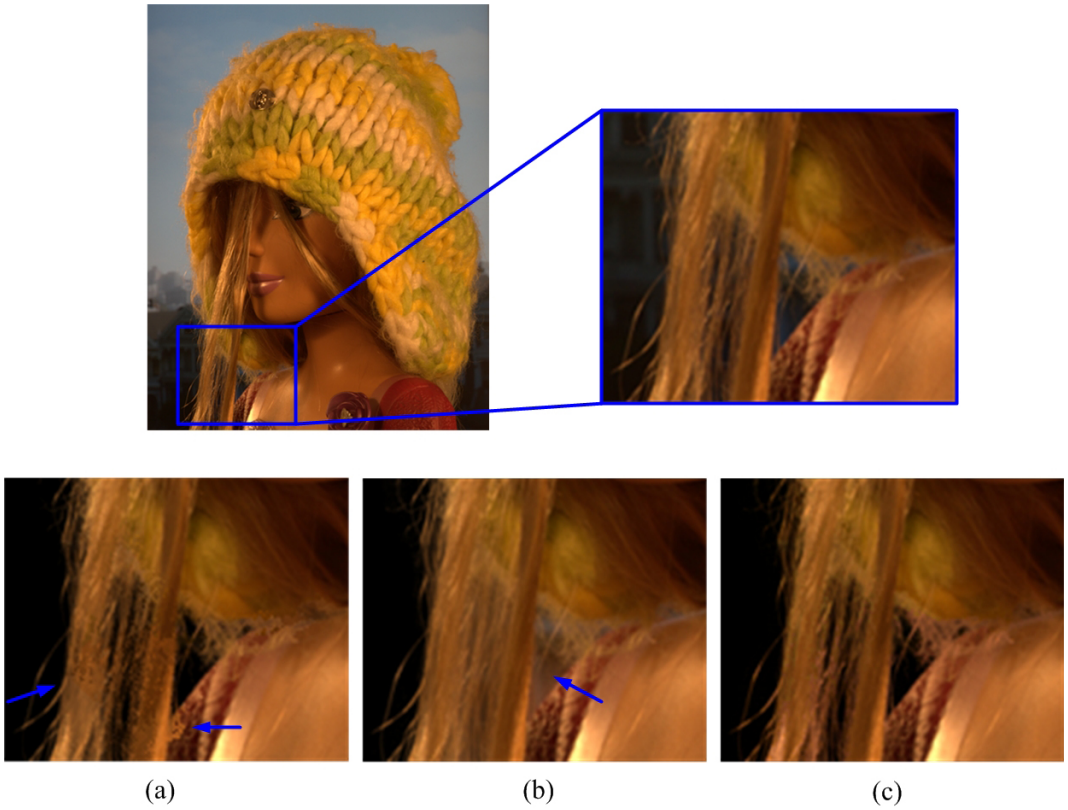


Figure 3.13: Matting result comparison 3. (a) Robust matting [29]; (b) Closed-form matting [18]; (c) Our method.

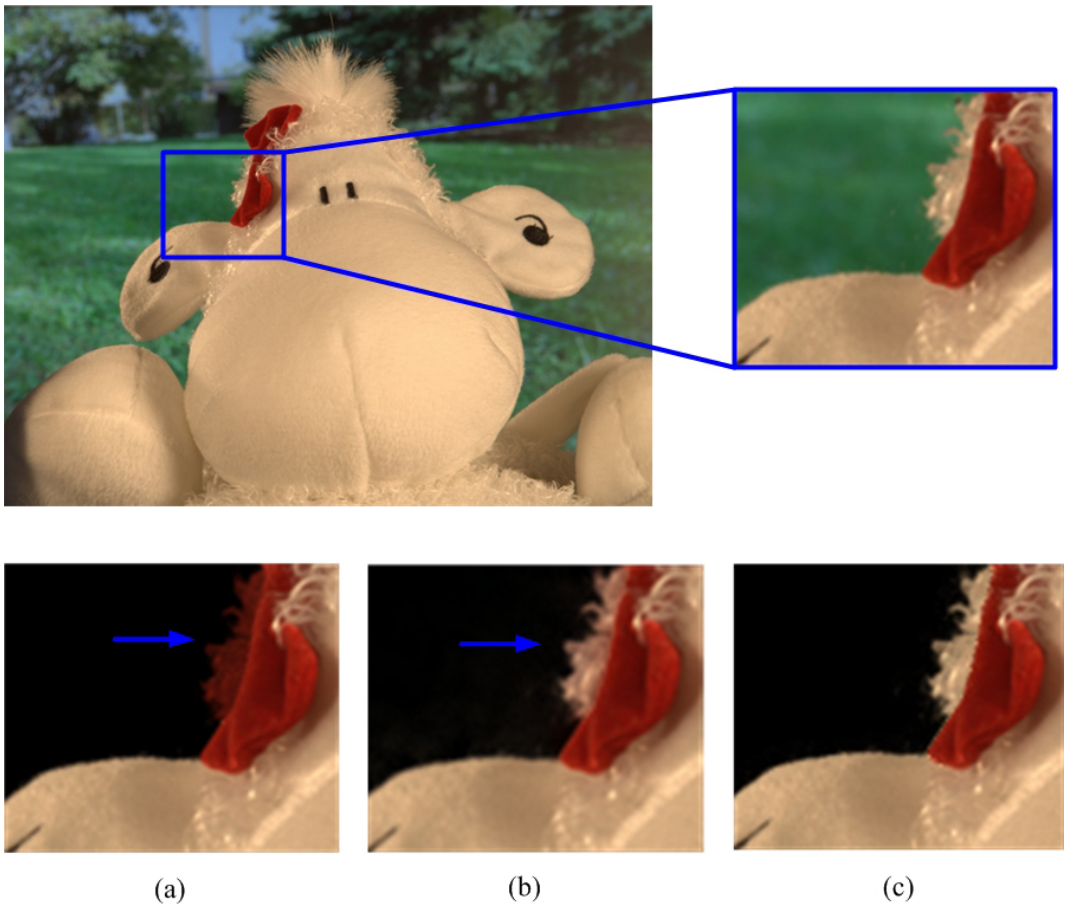


Figure 3.14: Matting result comparison 4. (a) Robust matting [29]; (b) Closed-form matting [18]; (c) Our method.

to color intensity values. Then a randomized sample matching algorithm is applied to efficiently find out the optimal sample pairs according to the specified cost function. We compared the performance with other matting algorithms which own good rank on the alpha matting evaluation website [42]. Our method generate matting results with higher quality.

Chapter 4

The sampling-based alpha matting for video

We have presented an optimized matting algorithm on still images. In this chapter we aim to address the more challenging task – video matting.

4.1 Background

Extracting foreground object from video streams is intrinsically a harder problem than from images. Tiny errors that may not be noticed in images will appear to be obvious in video sequences, especially when those errors happen in different regions in each frame. Therefore the matting is required to be as accurate as possible.

Wang [43] proposed a user guided system which operates on the video sequences as a whole. In practice, the user interface (UI) is hard to use. And since video matting is treated as a global optimization problem, any tiny modification made by the user in one frame may affect other unrelated regions in other frames, which is often uncontrollable

by users. Besides, since this system needs to go through all the frames at the same time, a significant amount of memory space is required.

In the video segmentation and matting system in [31], users guide the segmentation by specifying strokes on key frames and the scribbles are then propagated to other frames by calculating weighted distances in spatio-temporal space. But since weighting the geodesic value is dependent on the pixel value distributions, the performance degrades when these distributions significantly overlap.

Although SnapCut [44] further improved the previous methods using localized classifiers, still, it is an interactive system and its performance is determined by whether or not enough user assistance is provided. Besides, its performance on fuzzy boundaries are not very satisfactory.

Lee et al. [45] treat video sequences as a spatio-temporal cube and extend the 2D Robust matting algorithm [29] to 3D by regarding the time axis as a third spatial dimension. However, this method requires all the trimaps for every frame before processing, and users have to draw trimaps manually on each key frame, which is a huge workload and not practical. Moreover, in some cases, this method does not actually perform as well as expected, some foreground elements are sometimes cut out.

Some other methods ([46][47][48][49]) group pixels or segments into layers according to the affinity of local measurements [50]. But they sometimes fail to distinguish objects since that different objects may share similar color or motion information. Agarwala [51] interpolates partial foreground boundary splines by using Rotoscoping technology, but they cannot deal with fast topological change and normally require a large amount of user interaction.

In practice, the most common and popular application of video matting is in *special effects* in film making and videogames, weather forecast and news broadcasting , where

the objects are usually captured against single-color or gridded backgrounds. The color range of the background indicates the area to be removed in the image. This technique is also known as Chroma keying [52] since it is based on chroma information (as we have briefly introduced in Section 2.3).

It seems easier than matting in the natural background images since the background and foreground are distinct in color range. While in natural images, bad samples may be selected in the area where foreground and background colors are very close. Admittedly, when the foreground object is quite solid with simple and sharp boundaries and is of fully opaque, it can be extracted based on the estimation of the background color distribution. However, in quite a few cases, objects may have intricate boundaries, such as hair strands and fluffy toys, or semi-transparent parts of special material. The extracted object may suffer from "color-spill", or some tiny part along the boundaries might be cut out.

To accurately extract the foreground object and get a high quality result in such more complicated situation, the alpha channel is still required to indicate the opacity of each pixel. When we apply alpha matting to video, we need to consider about two main problems: spatial accuracy and complexity, including time complexity and operation complexity.

First, as we have mentioned, video matting requires accurate foreground pixel identification on each frame. In addition to a good matting algorithm itself, the trimap also has a significant influence on the matting result. Since we have proposed an improved matting algorithm which can generate good result, our next task is to optimize the trimap.

Secondly, unlike dealing with images, video matting has to consider more about efficiency. For images, processing time and complexity are not the most important

part. User intervention are very common, they can adjust several times to generate the final good result according to their needs. But when dealing with loads of video frames, too much user intervention is tedious and time-consuming.

Our target is to generate a fully automatic video matting system with its potential future application in real-time processing. However, all the systems mentioned above are user-guided. Users have to draw trimaps or scribble manually, and post-refinements are usually required in most cases. Thus, those methods can only be used for off-line processing.

In the following sections, we are going to describe our matting system in two aspects: automatic adaptive trimap generation and propagation.

4.2 Automatic adaptive trimap generation

4.2.1 Motivations

The previous image matting algorithms assume that the input image has already been labeled to different regions, i.e. the trimap is already given or drawn by users. But when applying to videos, it is not practical to let the user manually specify the trimap. We hope to find out a solution to generate trimaps by the system itself.

Li et al.[53], Wang et al.[43] and Bai et al.[44] proposed automatic methods to produce trimap according to the binary segmentation. Given a binary mask, they generate a uniform bandwidth along the mask boundaries and this uniform band represents the unknown region. To make sure all the elements along the boundaries are covered, the band should be wide enough. Figure 4.1 shows an example of when the object has fuzzy areas along the boundary, the unknown region appears to be too wide for other parts.

In practice, the trimap is a hard constraint for matting. It determines which pixels

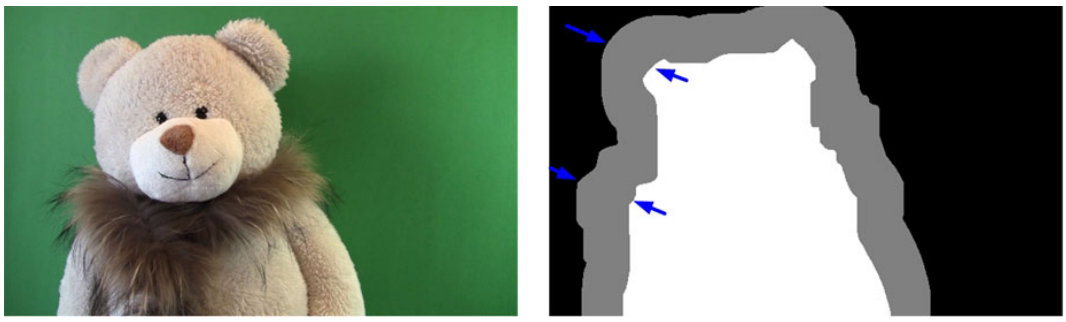


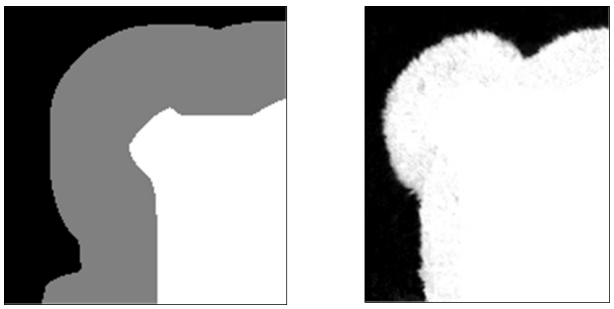
Figure 4.1: Uniform trimap.

need to be calculated and where the samples should be collected. The matting process can be very sensitive to changes of the trimap. For those areas which are comparatively solid, too wide unknown regions are redundant. These unnecessary parts not only increase the computation load, but also bring various artifacts, because many true foreground and background pixels are included and treated as unknown pixels. And their original color information can be changed during calculation. Figure 4.2 compares the matting result of using wide trimap and tighter trimap in a relatively solid area. We can see that wide trimap brings more artifacts.

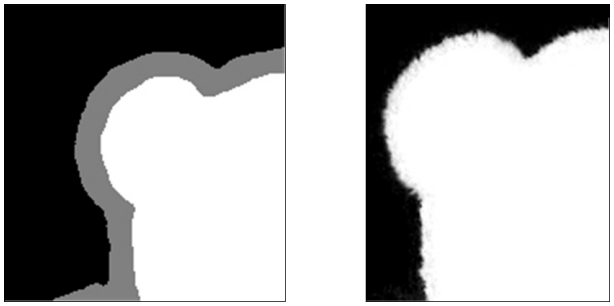
An accurate matting result can only be achieved by using an adaptive matting guide, i.e. an adaptive trimap, which can be wider in fuzzier area (such as long hair strands) and be tighter on relatively sharp boundary. Most of previous video matting systems are incapable of creating such trimaps.

Soft scissors [54] is an interactive system and they use scissor stroke to implicitly define a trimap. As users paint along the foreground boundary, the brush width and boundary conditions are automatically adjusted based on local statistics near the current brush stroke. Further manually adjustment is usually necessary in some situations. Obviously, this system relies too much on user interaction.

System in [7] uses adaptive trimap as well but also in an interactive way. The system



(a) Uniform trimap and the generated alpha matte



(b) Narrow trimap and the generated alpha matte

Figure 4.2: Comparison of matting results using different trimaps.

first generates a uniform triamp with the narrowest band, and then users modify the band width manually for more hairy areas.

Another approach proposed in [31] uses local sliding windows and the band width dynamically changes by complicated computation on Probability Distribution Function (PDF) and geodesics. However, according to the results, the computed width are not always reliable, since it is still wide in some solid area where it should be tighter.

System in [55] uses local sliding windows and the local fuzziness determines the width of the window. However, the computation of fuzziness is based on the already generated alpha matte, which means the local alpha profiles are also needed. Moreover, although the generated trimap is better than the uniform one, inappropriateness still exists.

In fact, the solid background offers distinct advantages that we can automatically

generated more accurate trimaps. We first cluster the foreground and background pixels and create a binary mask, and then use local indicators to generate the trimap with variable band width.

4.2.2 Clustering and binary mask

First, we use K-means clustering [56] algorithm to category the input image into two clusters. Since the background is single-color, we can get a rough segmentation of foreground and background. Then according to the label of each pixel, we generate the binary mask, which is with the same concept as in previous approaches. And morphological operation can be applied to get rid of some tiny disconnected patches. Figure 4.3 shows the generated binary mask of the input image.

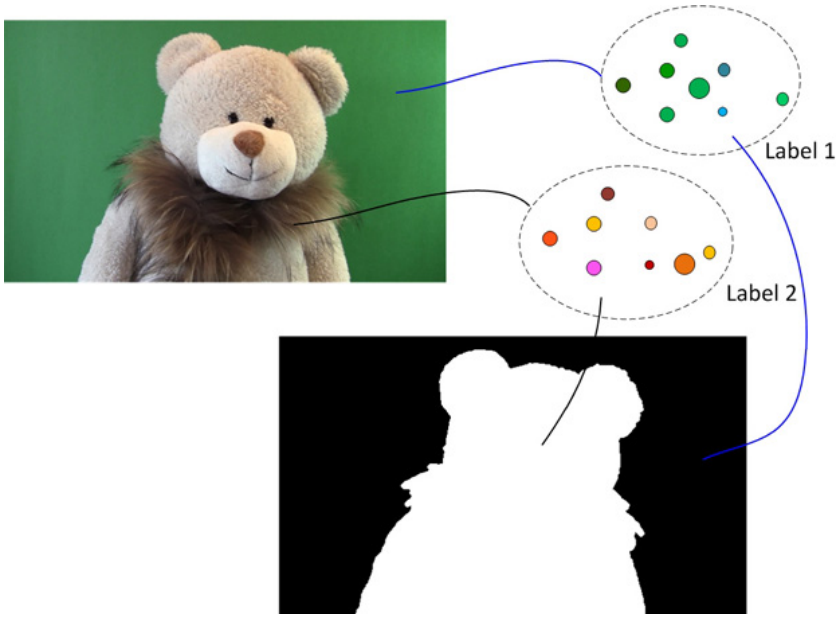


Figure 4.3: Generate binary mask by categorizing input image into two clusters.

Then another clustering process runs in foreground and background areas respectively. First, by dilating and eroding the binary mask, we separate the total foreground

and background areas for next clustering process. By doing this, we aim to avoid the possible influence from the mixed pixels along the boundary.

Secondly, we further subdivide the foreground into N_F clusters (the number of clusters is a preset parameter depends on the diversity of foreground colors, normally set 10-20 clusters if the foreground is not complex) and let pixels with similar colors are grouped together. Likewise, the background is subdivide into N_B clusters. Since the background is homochromous, fewer clusters are needed to get all pixels properly categorized. By doing this, we have $N_F + N_B$ centroids in the color space. Figure 4.4 is the clustered foreground and background pixels. Next, we will generate adaptive trimap according to those information.

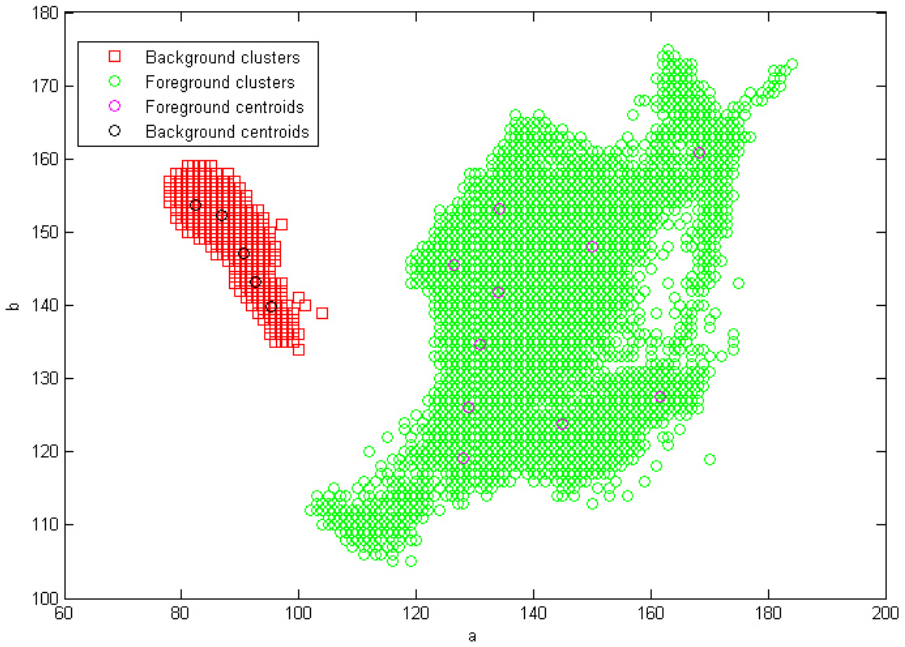


Figure 4.4: Foreground and background clusters and centroids. X-axis and Y-axis are a and b values in $L * a * b$ space respectively.

4.2.3 Adaptive trimap generation

Initially, we tried to use local sliding windows to adjust the band width of the unknown region, as proposed in previous approaches. But we experimentally found that there are drawbacks of using local windows.

To use local windows, there are three essential factors:

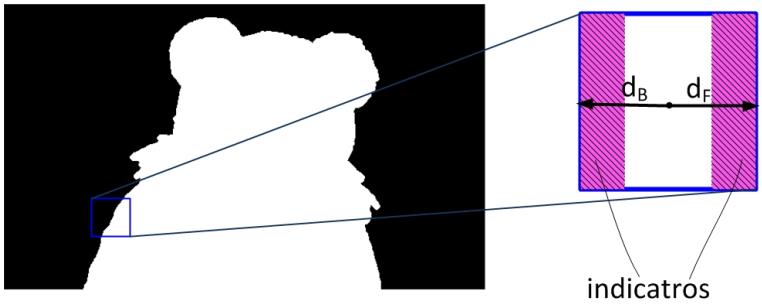
1. Define an *initial size*. It can be either large or small according to previous approaches:
   Define an initial large-sized window and let it keep shrinking until certain requirements are fulfilled;
   Define an initial small-sized window and let it keep growing until certain conditions are satisfied.

2. Define effective *detecting areas*.

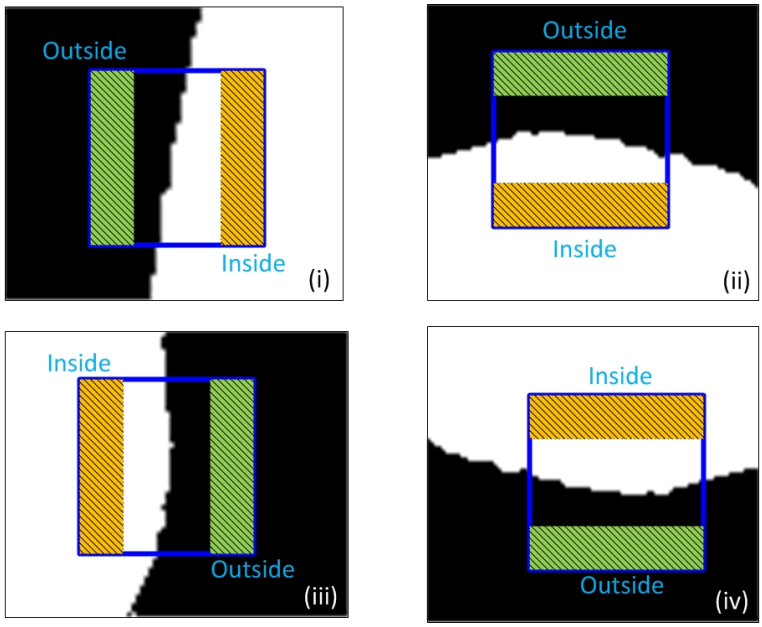
3. *Control item*, indicating the change of the local window.

We first define the initial window with small size. On both sides of the window, there are two effective *detecting areas*, detecting if there are any foreground/background pixels in this area. We name them as *indicator* (as illustrated in Figure 4.5(a)). Note that the outer part and inner part of the window are defined differently according to the general trend of the local boundary: vertically from bottom to top (i), vertically from top to bottom (iii), horizontally from left to right (ii) and horizontally from right to left (iv) (Figure 4.5(b)).

The *control item* defines the conditions when the window stops growing: we compare the minimum distance between the *indicator pixel* and the background centroids $\mathcal{D}_1$, and the minimum distance between the *indicator pixel* and foreground centroids $\mathcal{D}_2$.



(a)



(b)

Figure 4.5: Define local sliding windows.

1. For the *outer indicator*, d_B stops increasing until all pixels satisfy $\lambda\mathcal{D}_1 < \mathcal{D}_2$;

2. For the *inner indicator*, d_F stops increasing until all pixels satisfy $\lambda\mathcal{D}_2 < \mathcal{D}_1$. $\lambda \in [1, 6]$.

However, failures happen in the situation shown in Figure 4.6. At some complex corners, the window will keep growing to a big size, which is actually not necessary. And we guess this might be the similar reason that makes the results of previous approaches always contain inaccuracy.

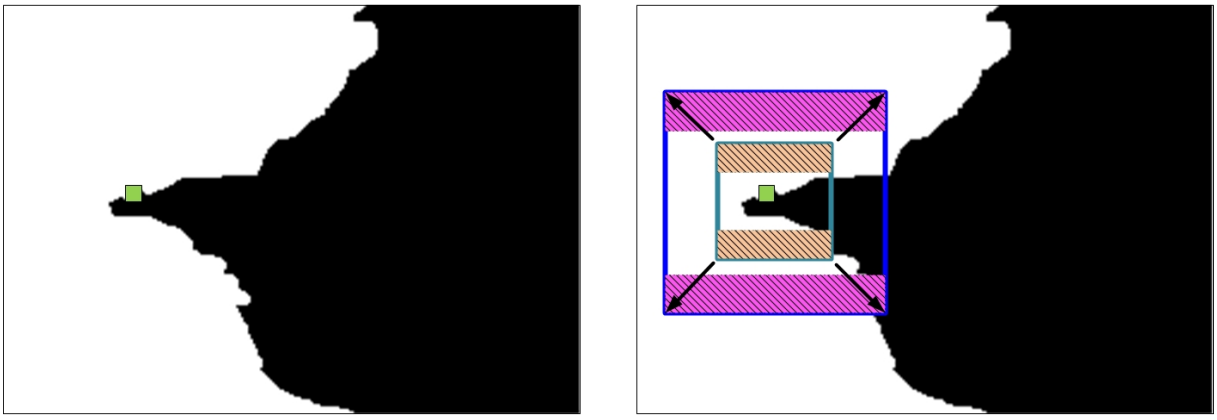


Figure 4.6: Performance of sliding windows in complex situation. In some complicated situations, the sliding window failed to calculate the proper band width.

We solved this problem by generating a series of contour lines. Given any point p_i on the binary contour, we draw a series of contour lines of the mask boundary, and these lines function as *indicators* (Figure 4.7). We check the pixels along the contour lines. The distance between current point p_i and contour lines increase from d_{p_i} to d'_{p_i} until all the pixels along the contour lines satisfy the conditions of the *control item*. Figure 4.8 displays the process of adaptive trimap generation.

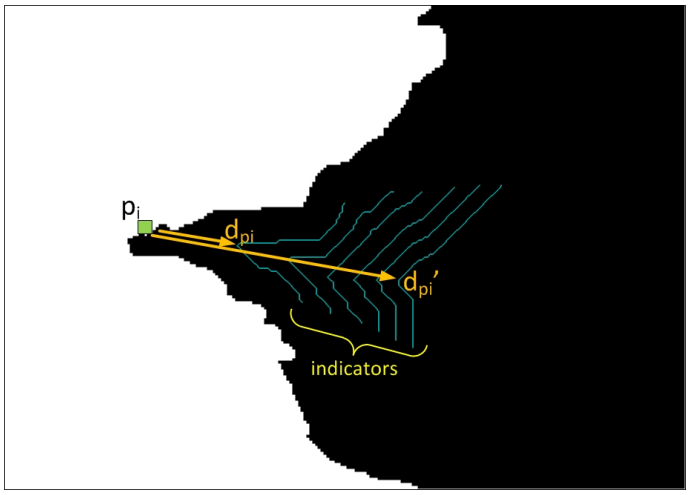


Figure 4.7: Indicators as a series of contour lines.

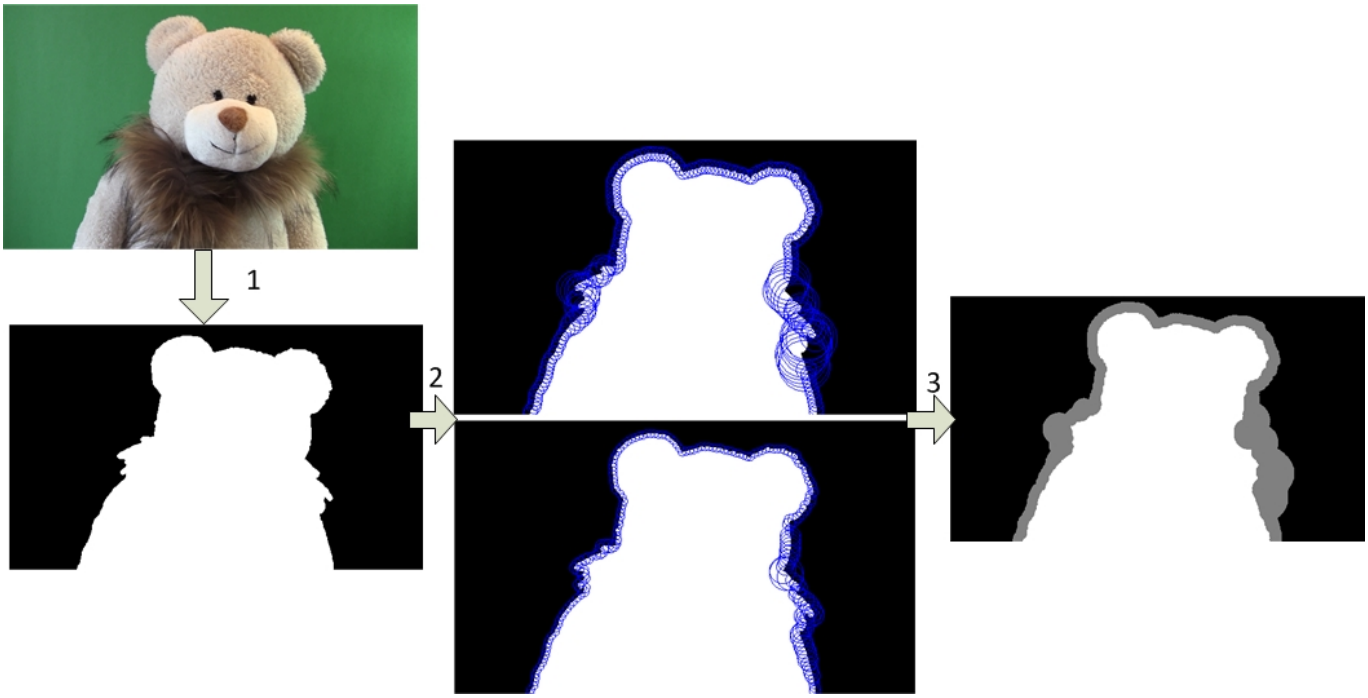


Figure 4.8: Flow chart of adaptive trimap generation. 1. Generate binary mask. 2. Calculate the outer and inner width. 3. Rasterize and generate trimap.

4.3 Trimap propagation in temporal dimension

After the accurate trimaps on key frames are generated, we hope the local band width can be temporally-coherent. To propagate the adaptive trimap frame-to-frame, we adopt the method proposed in [7]. In [7], they define local windows to cover all the local fractional alpha pixels and compute the outer and inner width. We do not define such kind of windows based on the following considerations:

1. Local windows cannot well address the complex topological problems, as we have demonstrated previously. Complex topological layout may lead to inaccurately calculated band width.

2. To ensure that the windows can cover all the fractional pixels, the generated alpha matte is required to be extremely accurate. Because if any tiny strand is failed to be calculated, they might be missed or become disconnected with other parts in the alpha matte. In that case, the local window will not be able to cover them and the computed window size will be incorrect.

Since we have already generated the data set that contains all the control points and their corresponding outer and inner width in previous step, we can use this information to propagate the parameterized trimaps. For current key frame t, we have the set $\Phi_t = \{p_i, r_i^F, r_i^B\}(i = 1, 2, \cdots, N)$ which contains the control points along the mask contour and their corresponding outer and inner radius. Then we use optical flow [57] to compute the relative displacement between the current frame and next frame, and find the new locations of the control points in next frame. We get $\Phi'_{t+1} = \{p'_i, r_i^F, r_i^B\}$.

Due to the possible errors of optical flow and topology changes between frames, the computed control points p'_i may not fall on the boundary of the binary segmentation mask of frame $t + 1$, thus the previous radius values are not appropriate for the new

frame. To assign the radius values to the points on the boundary, we employ the thin-plate interpolation method [58], as mentioned in [7].

Thin-Plate Spline (TPS) is an interpolation method providing a smooth interpolation between a set of given points. It aims to find an ideal (least bent) smooth surface that passes through all control points.

Given a set of interpolation values $Y = (y_1, y_2, \cdots, y_N)$ at the distinct points $X = (x_1, x_2, \cdots, x_N)$, the TPS fits a mapping function $f(\mathbf{x})$ between point sets $\{x_i\}$ and corresponding values $\{y_i\}$ by minimizing the following bending energy function:

$$E[f(x,y)] = \int\int_{R^2} [(\frac{\partial^2 f}{\partial x^2})^2 + 2(\frac{\partial^2 f}{\partial xy})^2 + (\frac{\partial^2 f}{\partial y^2})^2] dxdy. \tag{4.1}$$

The minimally bent surface is then given by:

$$f(x,y) = a_1 + a_2 x + a_3 y + \sum_{i=1}^{N} \lambda_i \Phi(\|(x,y) - P_i(x_i, y_i)\|), \tag{4.2}$$

where $\|.\|$ denotes the Euclidean norm, and $\Phi(r) = r^2 logr$ is the kernal function of thin-plate spline.

Applying to our situation, given a set of control points $\mathbf{C}$ and their corresponding radius $\mathbf{h}$, we define

$$h_i = a_1 + a_2 c_i^x + a_3 c_i^y + \sum_{j=1}^{N} \lambda_j \Phi(\|c_i - c_j\|), (i = 1, 2, ..., N). \tag{4.3}$$

Considering the orthogonality and side conditions, we define

$$\begin{pmatrix} K & P \\ P^T & O \end{pmatrix} \begin{pmatrix} \lambda \\ \mathbf{a} \end{pmatrix} = \begin{pmatrix} \mathbf{h} \\ \mathbf{o} \end{pmatrix} \tag{4.4}$$

where

$$K = \begin{pmatrix} \Phi_{11} & \Phi_{12} & \cdots & \Phi_{1N} \\ \Phi_{21} & \Phi_{22} & \cdots & \Phi_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \Phi_{N1} & \Phi_{N2} & \cdots & \Phi_{NN} \end{pmatrix} \tag{4.5}$$

and

$$P = \begin{pmatrix} 1 & c_1^x & c_1^y \\ 1 & c_2^x & c_2^y \\ \vdots & \vdots & \vdots \\ 1 & c_N^x & c_N^y \end{pmatrix} \tag{4.6}$$

Then we can solve the coefficients $\lambda = (\lambda_1, \lambda_2, ..., \lambda_N)$ and $\mathbf{a} = (a_1, a_2, a_3)$ by

$$\begin{pmatrix} \lambda \\ \mathbf{a} \end{pmatrix} = \begin{pmatrix} K & P \\ P^T & O \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{h} \\ \mathbf{o} \end{pmatrix} \tag{4.7}$$

Notice that different radius sets $\mathbf{h}_{outer}, \mathbf{h}_{inner}$ lead to different coefficients sets $(\lambda_{outer}, \mathbf{a}_{outer})$ and $(\lambda_{inner}, \mathbf{a}_{inner})$.

After the coefficients are confirmed, we get the final interpolation function for our trimap mapping:

$$f_B(x, y) = a_1^{outer} + a_2^{outer} x + a_3^{outer} y + \sum_{i=1}^{N} \lambda_i^{outer} \Phi(\|(x, y) - p_i'\|), \tag{4.8}$$

$$f_F(x, y) = a_1^{inner} + a_2^{inner} x + a_3^{inner} y + \sum_{i=1}^{N} \lambda_i^{inner} \Phi(\|(x, y) - p_i'\|), \tag{4.9}$$

p_i' is the new location of the control point computed according to optical flow. Then we uniformly select a new set of control points along the mask contour in frame $t + 1$

and denote it as $\mathbf{q} = \{q_1, q_2, \cdots, q_M\}$. Their outer trimap band width is computed as

$$d^B(q_j) = f_B(q_j), \tag{4.10}$$

and the inner width is

$$d^F(q_j) = f_F(q_j). \tag{4.11}$$

Now we already have the control points and their corresponding inner and outer radius on next frame. Then for each control point q on the mask boundary, the local band width $d = d_q^B + d_q^F$ is confirmed. Then we get the trimap for the new frame.

Figure 4.9 shows that given the current frame t and the already generated trimap, if we just propagate the trimap to frame $t + 1$ by optical flow, inaccurate trimap is generated due to the unavoidable optical flow errors and topology changes. In contrast, the application of TPS effectively updates the trimap for the new frame.

After the trimaps are parameterized and propagated from key frames to all other frames, alpha mattes are then computed. But we do not select the global background samples in this case since the background is constant. We just collect the nearby background samples to avoid the unnecessary computation.

4.4 Experimental results

In this section we display more examples in video sequences. The artifacts caused by coarse trimaps may not be noticeable in single image, but it will become more obvious in multiple continuous frames. Those artifacts can be eliminated by our matting algorithm using more accurate trimaps. Figures 4.10 and 4.11 compare the results using uniform trimaps and adaptive trimaps. We can see that when using uniform trimap,

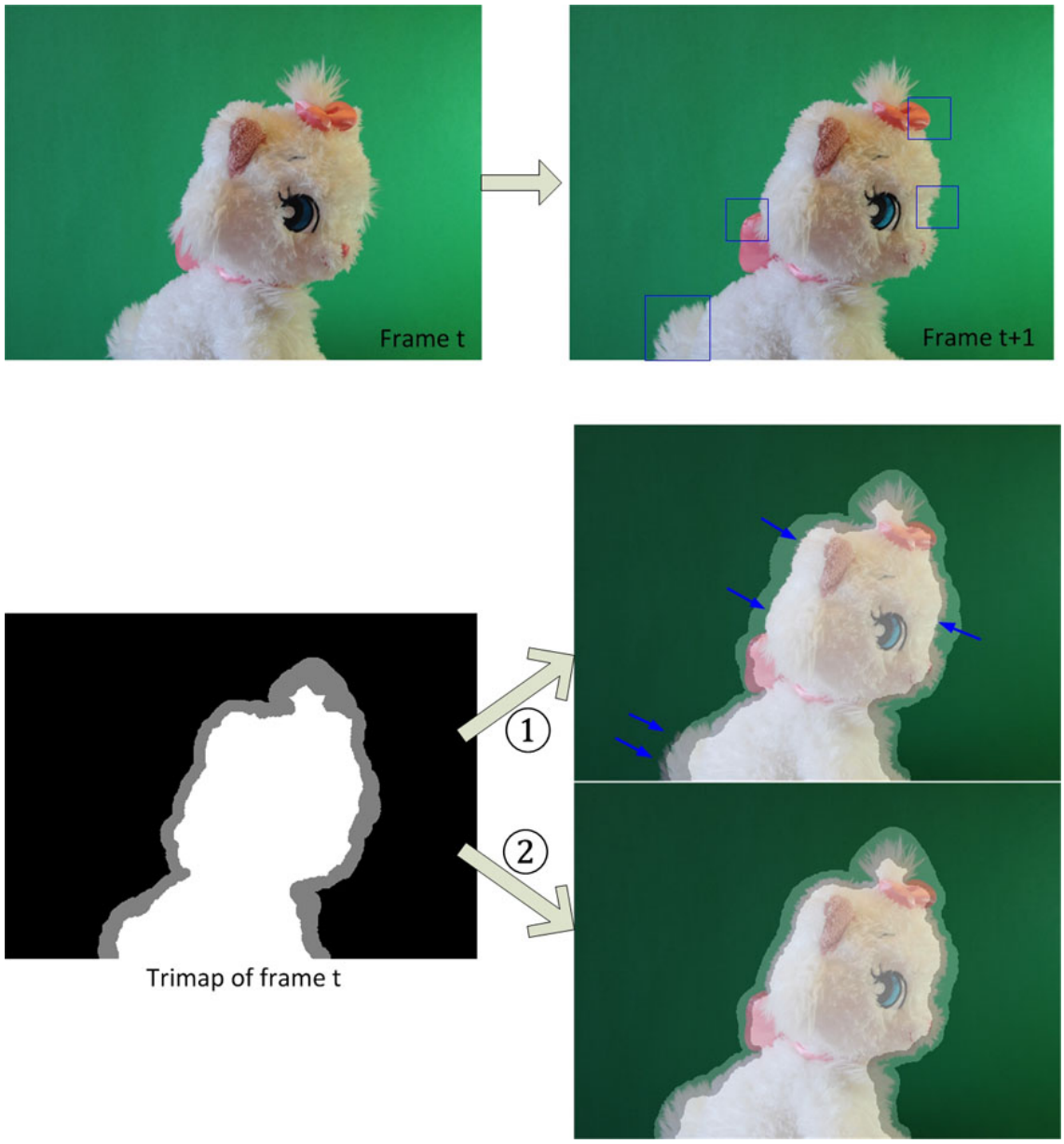


Figure 4.9: Trimap propagation from frame t to $t+1$. On the upper row, the blue windows indicate the topological changes from frame t to $t+1$. Arrow 1 indicates the propagation without TPS, blue arrows indicate the errors; Arrow 2 indicates the result adjusted by TPS.

some regions get too much color information from foreground and the mixed pixels are erroneously estimated. In contrast, the adaptive trimap provides proper guide for sample selection, to effectively exclude the negative influence from the irrelevant foreground samples.

4.5 Summary

In this chapter, we extended our matting algorithm to video sequence. We proposed trimap generation approach which produces adaptive trimap in an automatic fashion. It avoids the tedious work of manually generating trimaps and enhances the efficiency. It overcomes the drawbacks of local sliding windows and generates more accurate trimaps. And the accurate trimaps further improve the matting quality. The application of thin-plate spline makes the trimap propagation more accurate.

Figure 4.10: Video matting comparison–Sequence 1. (a) The original frames. (b) The results using uniform trimap. (c) The results using adaptive trimap.

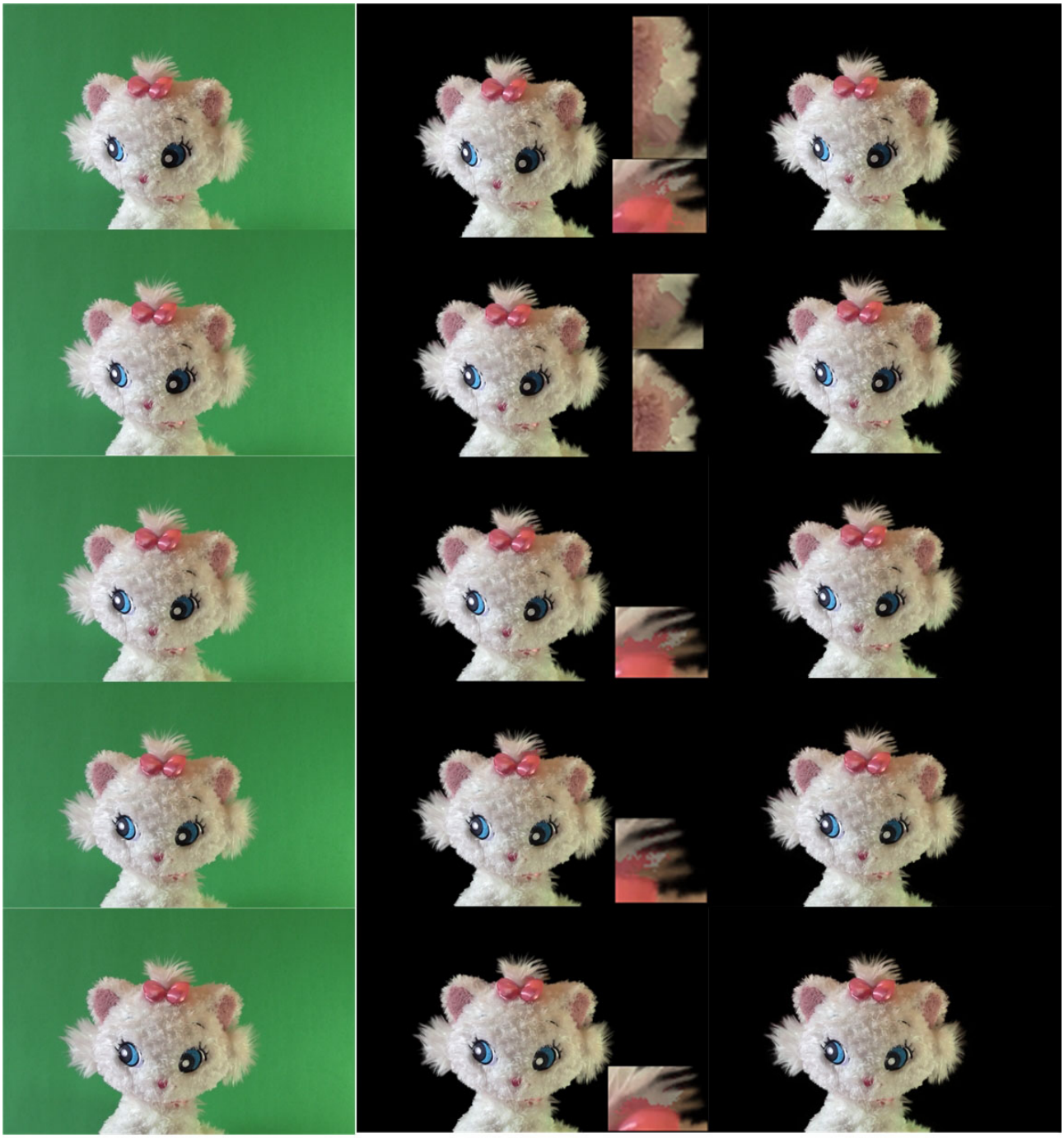


Figure 4.11: Video matting comparison–Sequence 2. (a) The original frames. (b) The results using uniform trimap. (c) The results using adaptive trimap.

Chapter 5

Conclusions and future work

In this thesis we surveyed previous foreground extraction technology both for images and video. By analyzing their limitations and drawbacks, we proposed an improved sampling-based alpha matting algorithm. It collects foreground and background samples in a global range and a randomized search algorithm is applied to select the good samples in a large sample set. The modified PatchMatch technique significantly enhances the efficiency of the searching process. Good samples are judged by certain criteria. We redefined the cost function which considers both color weight and distance weight.

When applying to video, more challenging tasks come out. Firstly, trimaps are vital for alpha matting but hand-draw trimaps or scribbles are not practical in video due to the enormous frames. Secondly, the quality of trimap has significant influences on the matting result, and the bad results will be more visible in video sequence than in images. We proposed a fully automatic approach to generate adaptive trimaps. The band width can be adjusted automatically according to the local situation, thereby generating more accurate trimaps. A modified thine-plate spline technique is also applied for the

accurate trimap propagation.

Future research directions include the proper design of temporal filter and real-time processing. When dealing with more complicated situations such as fast moving hairy regions, a temporal filter may need to be employed to avoid jitters. However, in order to have a good coherence with other frames, it may sacrifice the good quality of some already well-computed frames. Accordingly, the problem of how to design a good filter, and the balance between coherence and accuracy need to be seriously considered.

Besides, unlike some previous systems which require the video stream to be inputted as a whole, we deal with video sequence frame-by-frame and do not ask for the whole sequence while processing. Therefore, it can be further improved for real-time application.

References

[1] D. L. Pham, C. Xu, and J. L. Prince, "Current methods in medical image segmentation," *Annual Review of Biomedical Engineering*, vol. 2, pp. 315–337, 2000.

[2] Wikipedia, http://en.wikipedia.org/wiki/Segmentation_(image_processing), Last visit on 17 August 2012.

[3] L. G. Shapiro and G. C. Stockman, *Computer Vision*. New Jersey: Prentice-Hall, 2001.

[4] M. A. Ruzon and C. Tomasi, "Alpha estimation in natural images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, June 2000, pp. 18–25.

[5] T. Porter and T. Duff, "Compositing digital images," *Computer Graphics*, vol. 18, no. 3, pp. 253–259, July 1984.

[6] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun, "A global sampling method for alpha matting," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 20-25 June 2011, pp. 2049–2056.

[7] X. Bai, J. Wang, and D. Simons, "Towards temporally-coherent video matting," in *Computer Vision/Computer Graphics Collaboration Techniques*, 10-11 October 2011, pp. 63–74.

[8] J. Wang and M. F. Cohen, "Image and video matting: a survey," *Foundations and Trends in Computer Graphics and Vision*, vol. 3, no. 2, pp. 97–175, January 2007.

[9] N. Joshi, W. Matusik, and S. Avidan, "Natural video matting using camera arrays," in *Proceedings of ACM SIGGRAPH*, 2006, pp. 779–786.

[10] J. Sun, Y. Li, S. B. Kang, and H.-Y. Shum, "Flash matting," in *Proceedings of ACM SIGGRAPH*, vol. 25, no. 3, July 2006, pp. 772–778.

[11] M. McGuire, W. Matusik, H. Pfister, J. F. Hughes, and F. Durand, "Defocus video matting," *ACM Transactions on Graphics*, vol. 24, no. 3, pp. 567–576, 2005.

[12] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum, "Poisson matting," in *Proceedings of ACM SIGGRAPH*, 2004, pp. 315–321.

[13] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, "Random walks for interactive alpha-matting," in *Proceedings of Visualization, Imaging and Image Processing (VIIP)*, 2005, pp. 423–429.

[14] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut–Interactive foreground extraction using iterated Graph Cuts," in *Proceedings of ACM SIGGRAPH*, vol. 23, no. 3, 2004, pp. 309–314.

[15] X. He and P. Niyogi, "Locality preserving projections," in *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[16] L. Grady and G. Funka-Lea, "Multi-label image segmentation for medical applications based on Graph-Theoretic electrical potentials," in *ECCV Workshops CVAMIA and MMBIA*, 2004, pp. 230–245.

[17] L. Grady, "Random walks for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1768–1783, November 2006.

[18] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, pp. 228–242, February 2008.

[19] Y. Y. B. Marie and P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in N-D images," in *Proceedings of 8th IEEE International Conference on Computer Vision*, vol. 1, Vancouver, Canada, 7-14 July 2001, pp. 105–112.

[20] K.-H. Tan and N. Ahuja, "Selecting objects with freehand sketches," in *Proceedings of 8th IEEE International Conference on Computer Vision*, vol. 1, Vancouver, Canada, 7-14 July 2001, pp. 337–344.

[21] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum, "Lazy snapping," in *Proceedings of ACM SIGGRAPH*, 2004, pp. 303–308.

[22] K.-H. Tan and N. Ahuja, "A representation of image structure and its application to object selection using freehand sketches," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, 8-14 December 2001, pp. 677–683.

[23] J. Wang and M. F. Cohen, "An iterative optimization approach for unified image segmentation and matting," in *Proceedings of the 10th IEEE International Conference on Computer Vision*, vol. 2, Beijing, China, 17-21 October 2005, pp. 936–943.

[24] Y. Mishima, "Soft edge chroma-key generation based upon hexoctahedral color space," U.S. Patent 5 355 174, 1993.

[25] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski, "A Bayesian approach to digital matting," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, December 2001, pp. 264–271.

[26] A. Berman, P. Vlahos, and A. Dadourian, "Comprehensive method for removing from an image the background surrounding a selected object," U.S. Patent 6 134 345, 2000.

[27] A. Berman, A. Dadourian, and P. Vlahos, "Method for removing from an image the background surrounding a selected object," U.S. Patent 6 134 346, 2000.

[28] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski, "Video matting of complex scenes," in *Proceedings of ACM SIGGRAPH*, 2002, pp. 243 –248.

[29] J. Wang and M. F. Cohen, "Optimized color sampling for robust matting," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN,USA, 17-22 June 2007, pp. 1–8.

[30] C. Rhemann, C. Rother, and M. Gelautz, "Improving color modeling for alpha matting," in *Biologically Motivated Computer Vision*, 2008.

[31] X. Bai and G. Sapiro, "A geodesic framework for fast interactive image and video segmentation and matting," in *Proceedings of IEEE 11th International Conference on Computer Vision*, Rio de Janeiro, 14-21 October 2007, pp. 1–8.

[32] E. S. L. Gastal and M. M. Oliveira, "Shared sampling for real-time alpha matting," *Computer Graphics Forum*, vol. 29, no. 2, pp. 575–584, May 2010.

[33] V. Iverson, "Chroma-key color range determination," U.S. Patent 5 774 191, June 1998.

[34] M. Uya, "Chroma-key signal generator," U.S. Patent 5 838 310, November 1998.

[35] R. L. Miller, "Apparatus and method for compositing video images," U.S. Patent 5 812 214, September 1998.

[36] K. Yamamoto and J. Yonemitsu, "Key signal generating apparatus for digital chromakey system," U.S. Patent 4 533 937, August 1985.

[37] D. Pettigrew, "Compositing video image data," U.S. Patent 6 445 816, September 2002.

[38] A. Demay and M. L. Lan, "Device and method for determining a key for clipping a subject moving against a colored background," U.S. Patent 5 903 318, May 1999.

[39] R. Gehrmann, "Chromakey method and associated circuit arrangement," U.S. Patent 5 719 640, February 1998.

[40] A. Demay and M. L. Lan, "Device and method for processing a signal with a subject moving against a colored background," U.S. Patent 5 923 381, July 1999.

[41] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: a randomized correspondence algorithm for structural image editing," in *Proceedings of ACM SIGGRAPH*, no. 24, 2009.

[42] A. M. E. Website, http://www.alphamatting.com/index.html, Last visit on 20 August 2012.

[43] J. Wang, P. Bhat, A. Colburn, M. Agrawala, and M. Cohen, "Interactive video cutout," in *Proceedings of ACM SIGGRAPH*, 2005, pp. 585–594.

[44] X. Bai, J. Wang, D. Simons, and G. Sapiro, "Video SnapCut: robust video object cutout using localized classifiers," in *Proceedings of ACM SIGGRAPH*, no. 70, 2009.

[45] S.-Y. Lee, J.-C. Yoon, and I.-K. Lee, "Temporally coherent video matting," *Graphical Models*, vol. 72, no. 77, pp. 25–33, May 2010.

[46] N. Jojic and B. J. Frey, "Learning flexible sprites in video layers," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2001, pp. 199–206.

[47] J. Shi and J. Malik, "Motion segmentation and tracking using normalized cuts," in *Proceedings of IEEE International Conference on Computer Vision*, Bombay, 4-7 January 1998, pp. 1154–1160.

[48] J. Y. A. Wang and E. H. Adelson, "Representing moving images with layers," *IEEE Transactions on Image Processing*, vol. 3, no. 5, pp. 625–638, September 1994.

[49] J. Xiao and M. Shah, "Motion layer extraction in the presence of occlusion using graph cuts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 1644–1659, October 2005.

[50] M. Han, W. Xu, and Y. Gong, "Video object segmentation by motion-based sequential feature clustering," in *Proceedings of the 14th annual ACM international conference on Multimedia*, 2006, pp. 773–782.

[51] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz, "Keyframe-based tracking for rotoscoping and animation," in *Proceedings of ACM SIGGRAPH*, 2004, pp. 584–591.

[52] Wikipedia, http://en.wikipedia.org/wiki/Chromakey, Last visit on 25 August 2012.

[53] Y. Li, J. Sun, and H.-Y. Shum, "Video object cut and paste," in *Proceedings of ACM SIGGRAPH*, 2005, pp. 595–600.

[54] J. Wang, M. Agrawala, and M. F. Cohen, "Soft scissors: an interactive tool for realtime high quality matting," in *Proceedings of ACM SIGGRAPH*, no. 9, 2007.

[55] J.-H. Cho, T. Yamasaki, K. Aizawa, and K. H. Lee, "Depth video camera based temporal alpha matting for natural 3D scene generation," in *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, Antalya, 16-18 May 2011, pp. 1–4.

[56] D. MacKay, "An example inference task: clustering," in *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003, ch. 20, pp. 284–292.

[57] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *8th European Conference on Computer Vision*, vol. 4, Prague, Czech Republic, 11-14 May 2004, pp. 25–36.

[58] G. Wahba, *Spline Models for Observational Data*, ser. CBMS-NSF Regional Conference Series in Applied Mathematics. Society for Industrial and Applied Mathematics, 1990.