Sampling-based image and video matting without compositing equation

Johnson, Jubin

2017

Johnson, J. (2017). Sampling-based image and video matting without compositing equation. Doctoral thesis, Nanyang Technological University, Singapore.

<http://hdl.handle.net/10356/70341>

<https://doi.org/10.32657/10356/70341>

SAMPLING-BASED IMAGE AND VIDEO MATTING WITHOUT COMPOSITING EQUATION

JUBIN JOHNSON

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2017

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my supervisor Dr. Deepu Rajan without whom this research work would not have been possible. His insight and knowledge in this research area has been one of the main reasons to help me choose my research topic. His patience and willingness to dedicate time in providing valuable technical inputs to improve the quality of the research is greatly appreciated. His inputs in writing skills has been immensely helpful for presentation of this research work. Many thanks to my collaborator and friend Hisham Cholakkal for his valuable inputs and enlightening discussions throughout my research study. I would also like to dedicate a special thanks to Gayathri for her love and support. Thanks are also dedicated to the Multimedia Lab technicians for providing the research facilities and softwares useful for conducting this research. Lastly, I would like to thank my parents and friends for their constant support throughout my life.

Contents

Acknowledgements	i
List of Figures	vii
List of Tables	xiv
Abstract	xvii
1 Introduction	1
1.1 Motivation	5
1.2 Objectives	6
1.3 Contributions	7
1.4 Organization	8
2 Literature Review	9
2.1 Introduction	9
2.2 Color sampling based approaches	10
2.2.1 Ruzon and Tomasi’s method	11
2.2.2 Bayesian matting	11
2.2.3 Knockout system	12
2.2.4 Robust matting	12
2.2.5 Shared matting	13
2.2.6 Global sampling matting	15
2.2.7 Weighted color and texture matting	15
2.2.8 Comprehensive sampling	16
2.3 Alpha propagation based approaches	17
2.3.1 Poisson matting	17
2.3.2 Geodesic matting	17

2.3.3	Closed-Form matting	18
2.3.4	KNN matting	20
2.3.5	Learning-based matting	20
2.3.6	Iterative transductive matting	21
2.3.7	Local and non-local smooth prior (LNSP) matting	21
2.4	Video Matting	21
2.4.1	Geodesic video matting	24
2.4.2	Video SnapCut	24
2.4.3	Robust video matting using temporal neighbors	25
2.4.4	Video matting using multi-frame nonlocal Laplacian	26
2.4.5	Temporal matte filtering based video matting	26
2.4.6	Discontinuity-aware video cutout	27
2.4.7	Motion-aware KNN video matting	28
2.4.8	Temporally coherent and spatially accurate video matting	28
3	Sparse Codes as Alpha Matte	31
3.1	Introduction	31
3.2	Using sparse coding to generate alpha matte	33
3.2.1	Sample selection for universal set	34
3.2.2	Certainty of unknown pixels	36
3.2.3	Dictionary formation and sparse coding to α	38
3.2.4	Graph-based matte optimization	41
3.3	Experimental Results	44
3.3.1	Experiment 1 : Analysis of different components in the proposed method	45
3.3.2	Experiment 2 : Evaluation on benchmark dataset	49
3.3.3	Experiment 3 : Comparison with other feature coding methods	56
3.3.4	Experiment 4: Runtime performance	58
3.4	Conclusions	58

4	Multi-frame video matting with temporal trimap propagation	61
4.1	Introduction	61
4.2	Temporal trimap propagation using motion-assisted shape blending	63
4.2.1	System Overview	65
4.2.2	Shape blending between key-frames	65
4.2.3	Combining motion with shape interpolation	67
4.2.4	Gradient-based refinement and adaptive trimap generation	69
4.3	Multi-frame video matting	70
4.4	Experimental Results	73
4.4.1	Experiment 1: Evaluation of temporal trimap propagation step	75
4.4.2	Experiment 2: Evaluation of multi-frame graph model	78
4.4.3	Experiment 3: Evaluation of video matting algorithm	79
4.5	Conclusion	83
5	Reconstruction Errors as Alpha Matte	85
5.1	Introduction	85
5.2	Sparse reconstruction error as alpha matte	87
5.2.1	Sample selection and dictionary formation	87
5.2.2	From L1-regularized error to alpha	88
5.3	Patch-based temporal coherency frameworks	90
5.3.1	Non-local means using coherency sensitive hashing	91
5.3.2	Patch-based multi-frame graph model with motion	93
5.4	Experimental Results	96
5.4.1	Experiment 1 : Effect of patch-based temporal coherency step	97
5.4.2	Experiment 2: Evaluation of proposed video matting algorithm	97
5.4.3	Experiment 3: Evaluation of compositing applications of the ex- tracted mattes	101
5.4.4	Runtime performance	103
5.5	Conclusion	103

6	Conclusions	105
6.1	Summary of Contributions	105
6.2	Future Work	108
	Publications	111
	Bibliography	112

List of Figures

1.1	(a) Original Image, (b) Trimap, (c) Alpha matte, (d) Composited Image. Images are taken from [2].	3
1.2	Illustration of video matting. The first and second rows indicate five consecutive input frames of a video and its trimaps, respectively. The alpha mattes extracted by applying Comprehensive sampling [11] are shown in the third row. The composited video frames on a new background are shown in the bottom row.	5
2.1	Comparison of mattes generated by sampling-based methods. (a) Original Image, (b) trimap, alpha mattes of (c) Bayesian matting [36], (d) Robust matting [8], (e) Shared matting [9], (f) Global matting [10], (g) Weighted Color and Texture matting [12] and (h) Comprehensive sampling [11]. Errors are highlighted by the yellow arrows.	13
2.2	Illustration of sampling strategies of (a) Robust matting [8], (b) Shared matting [9], (c) Global matting [10] and (d) Comprehensive Sampling [11].	14
2.3	(a) Original Image, (b) Small and (e) large trimap, alpha mattes of (c,f) Poisson matting [13], (d,g) Geodesic matting [14].	18
2.4	Comparison of α -propagation methods. (a) Troll image, (b) trimap, alpha matte using (c) Closed-form matting [3], (d) KNN matting [16], (e) Learning-based matting [44], (f) LNSP matting [17].	19
2.5	Illustration of temporal inconsistency in video matting. First row shows the frames of a video sequence and per-frame matte extracted by comprehensive sampling [11] is shown in the second row.	22

2.6	Mattes obtained by [53] using temporal sampling to extend Robust matting to the 3-D domain. The yellow arrows indicate discontinuities due to low contrast in the sampling process.	25
2.7	Temporal inconsistencies in video matting. Four consecutive frames of the <i>Office</i> sequence are shown in the first row and the mattes in the zoomed regions obtained for Snap-cut [26], Bai <i>et al.</i> [28] and Shahrian <i>et al.</i> [30] are shown in the second, third and last row, respectively.	27
3.1	Visual comparison of alpha matte generated from multiple unpaired F and B samples using sparse coding with other state-of-the-art methods. (a) Original image with foreground (red) and background (blue) boundaries, (b) zoomed regions, (c) Bayesian matting [36], (d) Comprehensive sampling [11], (e) Weighted Color and Texture matting [12] (f) LNSP matting [17] and (g) proposed method.	32
3.2	Illustrative comparison of matte estimation between methods using paired samples and the proposed approach. (a) Pairwise approaches use pair/s of F and B samples to estimate α at a pixel. (b) The proposed approach eliminates the need to restrict matte estimation using (F, B) pairs. Instead, samples that produce least sparse reconstruction error at a pixel are selected.	33
3.3	Illustration of the proposed sampling strategy. (a) Input Image, (b) trimap, (c) image segmented into superpixels using [80] and (d) universal set of F and B samples.	35
3.4	Foreground probability for dictionary refinement. (a) Input image and its (b) foreground probability map, (c) zoomed window of the image and its (b) foreground probability map.	36
3.5	Defining low-certainty regions in complex areas. (a) Input image and (b) zoomed patches, corresponding (c) probability map, (d) thresholded mask for pixels in $[0.3, 0.7]$ and (e) low-certainty mask for each region.	38

3.6	Illustration of dictionary formation for sparse coding to generate the matte. Foreground and background regions are shown in blue and red respectively. (a) Input image, (b) trimap, (c) universal set of samples and (d) dictionary for a low-certainty pixel (green window) and high-certainty pixel (yellow window).	39
3.7	Initial alpha mattes generated by our proposed method.	40
3.8	Illustration of graph model on image for optimization of the matte. Each pixel is connected to its spatial neighbors (W_{lap}), feature-space neighbors (W_{knn}) and to the virtual foreground (W_F) and background (W_B) nodes.	42
3.9	Visual effect of alpha refinement using the graph model. (a) Input image, (b) initial estimate, (c) final alpha matte and (d) ground truth.	43
3.10	(a) Doll image and its trimaps (small, large and user-defined), (b) and (c) Images from test dataset.	45
3.11	Effect of each step in the proposed method towards visual quality of the matte. (a) Input image, matte extracted using (b) universal set as dictionary, (c) refined dictionary with probabilistic segmentation and (d) graph-based matte optimization.	46
3.12	Error plots showing the qualitative comparison between traditional sampling approaches (E_t) and the proposed sparse coding approach (E_s) on four images.	48
3.13	Qualitative comparison of proposed method against other state-of-the-art methods on the training dataset. (a) Image, zoomed regions showing the mattes of (b) Closed form [3], (c) Weighted color and texture [12], (d) Comprehensive sampling [11], (e) proposed method and (f) ground truth.	49
3.14	Qualitative comparison of proposed method on elephant, pineapple and plant images with top 5 methods at [25]. (a) Input image, (b) zoomed windows, (c) Comprehensive weighted color and texture [83], (d) Comprehensive sampling [11], (e) Iterative transductive matting [33], (f) LNSP matting [17], (g) Cluster-based matting and (h) proposed method. Zoomed in regions show the effectiveness of our method.	50

3.15	Additional results of proposed method in compositing applications. (a) Input image and trimap labeled in red and blue, (b) mattes extracted by our algorithm, (c) composited image and (d) zoomed region on composite image showing seamless extraction.	55
3.16	Visual comparison of feature coding methods for alpha matting. (a) Input image and zoomed window, (b) LLC , (c) LSC, (d) proposed method and (e) ground truth.	57
4.1	Illustration of temporal inconsistencies in video matting.	62
4.2	Proposed trimap propagation using foreground object contours. (a) Key-frame image with contour in yellow and its (b) trimap, (c) trimap generated by [58] seven frames ahead and its (d) alpha matte, (e) our propagated contour on the same frame, (f) our generated trimap used to extract the (g) alpha matte.	63
4.3	(a) Proposed framework for temporal trimap propagation, (b) Illustration of the generation of in-between contours (outlined region in (a)) using cat sequence.	64
4.4	Shape blending between key-frames marked in yellow. The shape interpolated curves are shown in blue.	66
4.5	Illustration of the proposed novel shape and motion combination strategy at a curve point s_i . The difference between $\theta^{Shp}(s_i)$ and $\theta^{Mot}(s_i)$ indicates the discord between shape-interpolation and motion flow.	67
4.6	Combining motion information with shape interpolation (blue) to obtain the final in-between curves (white).	68
4.7	Illustration of the adaptive trimap radius propagation. (a) Key-frame image with trimap, (b) unknown region with foreground, background and object contours, (c) zoomed region depicting radius values for contour points, (d) propagated trimap on $t + 10^{th}$ frame and its (e) extracted matte.	70
4.8	Illustration of sampling strategy for dictionary formation across multiple frames in the <i>Lemur</i> sequence. The unknown pixel in yellow considers foreground (red) and background (blue) samples from neighboring frames to avoid missing samples due to the dynamic nature of background.	71

4.9	Multi-frame graph model for temporally coherent video matting. (a) K -nearest neighbors of the pixel (yellow) in the second frame are encoded across multiple frames, (b) Each pixel (node) is connected to its feature space neighbors as well as the virtual foreground and background nodes in the graph. The blue, red, and white nodes indicate the definite background, definite foreground, and unknown pixels, respectively.	72
4.10	Video sequences from the dataset with ground truth mattes.	74
4.11	Qualitative comparison of proposed method for <i>Amira</i> , <i>Office</i> and <i>Goat</i> sequences. (a) Key-frame contour at frame t , (b) curve obtained using proposed in-between generation (yellow), (c) generated trimap using adaptive trimap parameterization, per-frame matte generated using (d) [27], (e) [28], (f) proposed method and (g) using hand-drawn trimap, all at frame $t + 5$. Red indicates the error (best viewed in color).	75
4.12	Quantitative comparison of the proposed trimap propagation framework with Bayesian matting [27] and temporally coherent matting [28] using (a) mean temporal coherence error (TCE) and (b) mean temporal jitter error (TJE) metrics for the extracted mattes.	76
4.13	Visual comparison of proposed graph model against temporal Laplacian refinement. (a) Three sequential frames of Unicorn and its zoomed regions, (b) temporal Laplacian result and (c) proposed method.	77
4.14	Temporal coherence: Quantitative comparison with [30], [26] and [28] on the <i>Arm</i> and <i>Dancer</i> sequences. Our mattes not only give smaller error, but also maintains a stable temporal coherence across the frames.	80
4.15	Qualitative comparison of the proposed method on <i>Goat</i> , <i>Office</i> and <i>Girl</i> sequences respectively. Arrows indicate regions of low quality mattes, showing the effectiveness of our method. Dataset courtesy of Adobe.	81
4.16	Estimated video mattes by the proposed method on four sequences.	82

5.1	Illustration of reconstruction error as alpha matte. (a) Input image with a window showing (b) zoomed region and its trimap, (c) color distribution of the pixels in RGB color space indicating foreground (F), background (B) and unknown regions, plot of reconstruction error in $[0, 1]$ using (d) F samples (e) B samples and (f) distribution of ground truth alpha values in the patch with 0 indicating black and 1 indicating white.	87
5.2	L1-regularized reconstruction error as alpha matte. (a) Input image and its trimap, plot of (b) ξ_F , (c) ξ_B and (d) estimated alpha map.	90
5.3	Illustration of the NLM framework that uses approximate K-nearest neighbor (AKNN) patches to maintain temporal coherence across the video. For the red patch, AKNN patches from immediate frames (blue) are used to find coherent patches from the next frames (green).	92
5.4	Illustration of the PatchGraph framework that uses motion information of the patch temporal coherence across the video. For the red patch, neighboring patches that share similar motion and appearance share similar weights in the weighted graph model.	94
5.5	Effect of patch-based temporal coherency frameworks compared to pixel-based estimation on the <i>Dancer</i> and <i>Dog</i> sequences.	96
5.6	Qualitative comparison of the video mattes on three consecutive frames of <i>Face</i> and <i>Girl2</i> sequences. (a) Input frames, estimated mattes of (b) SnapCut [26], (c) Bai <i>et al.</i> [28], (d) Shahrian <i>et al.</i> [30], (e) PixelGraph (chapter 4), (f) proposed PatchNLM and (g) proposed PatchGraph. Yellow arrows indicate discontinuities and errors in the matte. Best viewed electronically.	98
5.7	Qualitative comparison of the video mattes with ground truth on <i>Face</i> and <i>Woman</i> sequences. (a) Input frames and (b) zoomed region, mattes of (c) SnapCut [26], (d) Bai <i>et al.</i> [28], (e) Shahrian <i>et al.</i> [30], (f) PixelGraph, (g) proposed PatchNLM method and (h) ground truth.	99
5.8	Qualitative comparison of the proposed motion-infused PatchGraph with motion-aware Laplacian approach of Li <i>et al.</i> [65] on <i>Amira2</i> and <i>Kim</i> sequences.	100

5.9	Qualitative comparison of the video composites obtained from the Patch-Graph and PatchNLM frameworks on the <i>Chimp</i> and <i>Arm</i> sequences respectively.	102
-----	---	-----

List of Tables

2.1	Comparison of sampling-based approaches and propagation-based approaches in matting.	10
3.1	Evaluation of effect of each step in our method using sum of absolute difference (SAD) error.	47
3.2	Evaluation of matting methods by alpha matting website [25] over set of benchmark images with three trimaps with respect to SAD. Evaluations have been carried out in July 2015.	52
3.3	Evaluation of matting methods by alpha matting website [25] over set of benchmark images with three trimaps with respect to MSE. Evaluations have been carried out in July 2015.	53
3.4	Evaluation of matting methods by alpha matting website [25] over set of benchmark images with three trimaps with respect to gradient error. Evaluations have been carried out in July 2015.	54
3.5	Quantitative evaluation of the proposed sparse coding framework with other feature coding algorithms.	57
3.6	Comparison of running time of the proposed method with recent sampling-based approaches.	58
4.1	Ranks of video matting methods with respect to mean squared error (MSE) and temporal coherence error (TCE) and temporal jitter error (TJE) metrics.	78
4.2	Comparison of running time of the proposed method with recent sampling-based approaches	82

5.1	Comparison of mean temporal coherence error (TCE) measures of different video matting algorithms against the two frameworks of the proposed method.	100
5.2	Comparison of mean temporal jitter error (TJE) measures of different video matting algorithms against the two frameworks of the proposed method.	101
5.3	Comparison of running time of the proposed PatchNLM method with recent sampling-based approaches	103

Abstract

Image and video matting play a fundamental role in image and video editing applications. They are generally classified into α -propagation based approaches and color sampling based approaches. The correlation between neighboring pixels with respect to local image statistics is leveraged to interpolate the known alpha values to the unknown regions in α -propagation methods. In color sampling methods, foreground (F) and background (B) samples from known regions that represent the true colors of the unknown pixels are used to estimate alpha. Complex color distributions of foreground and background regions, highly textured edges, and unavailability of true F and B samples are some of the main challenges faced by current works. In addition to this, sampling methods have traditionally followed the compositing equation using (F, B) pairs for alpha estimation. When extended to videos, the unavailability of user-defined trimaps in each frame and the additional requirement of temporal coherency across the sequence makes the matte extraction process a highly challenging task. We aim to develop novel natural matting algorithms for both images and video that can alleviate the drawbacks faced by current methods in generating a good quality matte. We achieve the objectives through the following contributions.

First, a sampling-based image matting algorithm is proposed that utilizes sparse coding in the image domain to extract an alpha matte. Multiple F and B samples, as opposed to a single (F, B) pair is used to describe the color at a blended pixel. A carefully chosen dictionary made up of feature vectors from the F and B regions, refined through a foreground probability map ensures that the constrained sparse code coefficients can be approximated to the alpha value. Experimental evaluations on a public benchmark database show that our method achieves state-of-the-art results.

Second, a new video matting algorithm is proposed which uses a multi-frame graphical model to ensure temporal coherency in the extracted matte. The alpha value at

a pixel needs to be consistent and smooth across the video sequence for better temporal coherence. This is accomplished by simultaneously solving for the alpha mattes for multiple consecutive frames. An objective function is proposed that can be solved in closed-form as a sparse linear system. An adaptive temporal trimap propagation using motion-assisted shape blending is utilized to propagate the trimaps automatically between the key-frames. Experimental evaluations on an exclusive video matting dataset validates the effectiveness of the method.

Third, a new sampling-based video matting algorithm is proposed that reinterprets the matting problem from the perspective of sparse reconstruction error of F and B samples. Sampling methods generally select an (F, B) pair that produces the least reconstruction error. The significance of the error has been left unexamined. Two patch-based frameworks are used to ensure temporal coherency in the video mattes - a multi-frame non-local means framework using coherency sensitive hashing and a patch-based multi-frame graph model using motion. Qualitative and quantitative evaluations indicate the performance of the method in reducing temporal jitter and maintaining spatial accuracy in the video mattes.

Chapter 1

Introduction

Movie production and television industry has relied on digital visual effects to seamlessly create and manipulate images for introducing impossible action and characters in a real world setting. One of the most common operations in visual effects is to extract information from one scene and combine it with another, a process known as compositing. Cutting out the object (foreground) accurately from the scene is a prerequisite for seamless compositing. The problem is called matting, or pulling a matte, and essentially involves accurate estimation of foreground objects from an image or video sequence along with its partial or full pixel coverage over the background. This pixel coverage or blending factor is known as the alpha matte, or simply the matte. Once the matte has been obtained, it can be applied to various image editing and manipulation applications and special effects in film production.

Mathematically described by the *compositing equation* [1], the observed image color (I_z) of each pixel z in an image can be represented as the convex combination of a foreground (F_z) and background (B_z) color image pixel as

$$I_z = \alpha_z F_z + (1 - \alpha_z) B_z, \quad (1.1)$$

where α is the alpha matte and is a value within $[0, 1]$ with 0 indicating the pixel belonging to the background and 1 indicating it as belonging to the foreground. Any pixel with intermediate values of alpha are referred to as mixed pixels. It differs from the closely related area of binary segmentation where the α is constrained to either 0 or 1. The extension of the matting problem to video sequences - video matting helps in accurate

foreground object extraction from a video sequence. The objective of a matting algorithm is to find the foreground, background and alpha value at a given pixel location, given the observed color at that location. As can be seen from eq. (1.1), this is a highly ill-posed problem since it involves estimation of seven unknowns from just three known color component values and cannot be solved uniquely without additional constraints. Blue screen matting constrains the background to be blue (or green) and solves for the alpha. However, the focus of this thesis is on natural matting, where the image/video is taken in front of an arbitrary background. To overcome this, natural matting algorithms require additional information in the form of a tri-layer segmented image called the trimap. The trimap serves to roughly classify the image regions as definite foreground, definite background and unknown regions. The trimap is usually drawn by the user [2] or can be generated automatically, as in the case of video matting for each subsequent frame. Fig. 1.1(a) shows an input image and its trimap input is shown in Fig. 1.1(b). The goal of matting algorithms is to solve for alpha matte of the unknown regions (shown in gray) and if required, estimate the foreground and background colors in order for the foreground to be seamlessly extracted and composited onto another background.

Another input scheme simplifies the users' task by specifying simple scribbles [3,4] for the definite foreground and background regions. Certain assumptions on image statistics are then considered, to propagate and iteratively extend the definite regions in the image and ultimately extract the digital matte. It is useful to note that a few methods constrain the problem by using extra information when it is available, both in the image domain and the video domain. However, this extra information requires specially designed studio environment [5] or capturing devices [6] or multiple shots that are taken using different settings [7].

The alpha matting methods use the information given in the trimap or scribbles and make assumptions on image structure and statistics. Based on the assumptions, they are generally divided into three categories - color sampling based, alpha propagation based and a combination of both. In color sampling based approaches [8–12], a few representative samples are taken from the definite foreground and background regions from which a foreground and background sample pair is chosen by optimizing an objective

Figure 1.1: (a) Original Image, (b) Trimap, (c) Alpha matte, (d) Composited Image. Images are taken from [2].

function. This foreground (F) and background (B) sample pair is then used to estimate the alpha value (α) at the pixel with color I_z using eq. (1.1) which is given by

$$\alpha_z = \frac{(I_z - B)(F - B)}{\|(F - B)\|^2}, \quad (1.2)$$

where $\|\cdot\|^2$ is the Euclidean distance. α -propagation methods [3,13–16] leverage the local image statistics to propagate alpha from the known to the unknown regions. The third category includes methods [4,17,18] in which the matting problem is cast as an optimization problem, with color sampling forming the data term and alpha propagation forming the smoothness term; solving for the alpha matte becomes an energy minimization task.

Fig. 1.1 shows an example of alpha matting where the inputs are the image and its trimap (Fig. 1.1(a) and (b)). The alpha matte is shown in Fig. 1.1(c) which is used to extract the foreground and composite it onto a new background as in Fig. 1.1(d).

The mathematical implication of the video matting problem is the extension of the image matting to each frame of the video, i.e.,

$$I_z^t = \alpha_z^t F_z^t + (1 - \alpha_z^t) B_z^t \quad (1.3)$$

where F_z^t and B_z^t are the foreground and background colors at the pixel z in frame t . The extracted foreground sequence can be cut and paste onto another background scene or various special effects generated as in film production. The trimaps are usually specified for a few key-frames as it is impractical to make the user input the trimap for each frame. User-interfaces that help in easy editing like Soft Scissors [19] and volumetric painting [20] are useful for this task. These trimaps are then propagated to the intermediate frames to constrain the matting equation at each frame. The efficiency of video matting depends on temporal coherency, in addition to the spatial accuracy that is required in the case of image matting. Temporal coherency implies that the estimated matte of each frame should be temporally consistent throughout the video sequence. The human visual system is highly sensitive to fluctuations and jitter across the video. To ensure temporal coherency, the trimaps generated across the video sequence should also maintain their consistency. Low contrast and fast motion are factors that contribute to inaccurate matte in a frame, thereby leading to temporal jitter across the extracted video matte.

An example of video matting is shown in Fig. 1.2 where five consecutive frames of the *Kim* sequence are shown in the top row. The trimaps are estimated by propagating the trimaps from the first frame that the user marked using scribbles with white indicating foreground and black indicating background. The mattes are estimated by applying comprehensive sampling [11] on each frame. The extracted foreground is then composited onto a new background sequence as shown in the bottom row in Fig. 1.2.

Alpha matting helps in creating unique composites or facilitate other editing tasks like photo-enhancement [21], red-eye corrections [22] and non-photorealistic rendering [23]. Foreground object re-targeting [24] is another interesting application of image matting where the re-targeted image is forced to retain the foreground object by applying image matting methods to accurately estimate the foreground in the initial step. Video matting particularly finds itself an integral part of film production and special effects in movies. The ability to seamlessly transport the foreground objects (actors) to a new scene has been employed by the movie industry for many years now.

Figure 1.2: Illustration of video matting. The first and second rows indicate five consecutive input frames of a video and its trimaps, respectively. The alpha mattes extracted by applying Comprehensive sampling [11] are shown in the third row. The composited video frames on a new background are shown in the bottom row.

Image matting methods have significantly advanced over the years and state-of-the-art algorithms have been successfully incorporated into the commercial software packages. An online evaluation website [25] has enabled researchers to compare and analyze their algorithms with the current state-of-the-art. However, video matting systems are yet to be developed sufficiently to be applied on real productions and remains a significant area for further research potential.

1.1 Motivation

Color sampling based approaches are more common in literature due to the simplicity in their assumptions and superior performance. However, most of the works suffer from the drawbacks of complex color distributions between foreground and background, and highly textured regions. They rely on samples from the user-marked pixels and select the best foreground-background (F, B) pair as true approximations to the actual foreground

and background color of the unknown pixel. The compositing equation is leveraged to estimate the α from this pair. The quality of the estimated mattes are dependent on the selected samples and degrades when the true foreground and background colors are missing from the sample set. Existing methods [8–11] tried to address the missing samples problem by employing sampling strategies that select a comprehensive sample set of samples [11]. In this thesis, we address two questions related to matting. Given the set of samples, is it possible to better estimate the color of the mixed pixel with more than just an (F, B) pair as specified by the compositing equation? Also, is it possible to extract good matting algorithms without relying on the compositing equation?

In addition to the above challenges, video matting methods have an additional requirement of temporal coherence. Methods [26–28] that apply image matting algorithms to each video frame do not lead to temporally coherent results since they ignore the information present in the neighboring frames. The human visual system is highly sensitive to jitter and flickering across frames [29] and maintaining temporal coherence is crucial for a good video matting algorithm. At the same time, spatial accuracy of the matte should not be compromised. Striking a balance between the two has been the other motivation for our research related to video matting.

1.2 Objectives

The main objectives of this thesis are to develop new sampling-based algorithms for natural image and video matting that

- explore alternate approaches to bypass the compositing equation that has been traditionally followed by sampling-based methods,
- extract temporally coherent mattes on video sequences by incorporating information from multiple frames simultaneously,
- examine the influence of trimap propagation for temporal coherency,
- examine the role of reconstruction error at an unknown pixel.

The image matting method proposed in this research is compared with other state-of-the-art matting methods by the alpha matting website (www.alphamatting.com) on a benchmark dataset proposed by [2]. The images along with three sets of trimaps - small, large and user are available for download from the website. The three sets are provided to encode the variability in the trimap input. The ground truth is hidden from the public and the quality of the mattes are evaluated with respect to different error measures including Sum of Absolute Differences (SAD), Mean Squared Error (MSE) and gradient error. The video matting methods are compared using a dataset provided by Adobe [30] built exclusively for video matting containing sequences with low contrast, fast motion and hairy regions.

1.3 Contributions

The major contributions of this research are as follows:

- A non-parametric sampling-based image matting method is proposed that uses sparse coding to directly estimate the alpha matte. The use of multiple F and B samples as opposed to a single (F, B) pair leads to better reconstruction of the observed color at a pixel. By bypassing the compositing equation in estimating the matte, the sparse coded estimate better estimates the color of a mixed pixel, especially in regions of high texture and transparency. A graph-based optimization step is used to solve for the matte which ensures that spatial and feature neighbors share similar alpha values.
- A new video matting method is proposed that builds on the sparse coding framework to extract temporally coherent mattes. Temporal coherency is achieved through two stages - first, a temporal trimap propagation step is proposed that uses motion-assisted shape blending and adaptive trimap parameterization to propagate the trimap between user-defined key-frames. Second, a multi-frame graphical model is proposed that finds samples for the dictionary from across the frames followed by an efficient optimization function that solves for the mattes of multiple frames simultaneously.

- Continuing the theme of alternative approaches to the compositing equation, a new sampling-based video matting method is proposed that reinterprets the matting problem from the perspective of reconstruction error of an unknown pixel. The errors obtained during reconstruction using F and B samples are combined in a probabilistic framework to directly estimate the alpha. Two different patch-based strategies for maintaining temporal coherence in the video mattes are proposed. First, a multi-frame non-local means (NLM) framework that uses coherency sensitive hashing to select temporal approximate K-nearest neighboring (AKNN) patches is introduced. Second, a patch-based multi-frame graph model that reduces the errors introduced by the pixel-based formulation is compared for temporally consistent matte extraction.

1.4 Organization

The thesis is organized as follows. In chapter 2, a review of image and video matting is presented along with the drawbacks of current state-of-the-art methods. The challenges of pairwise methods are discussed and an image matting method that uses sparse coding to directly estimate the matte is proposed in chapter 3. The extension of the sparse coding based matting method to videos along with a framework for temporal trimap propagation is introduced next in chapter 4. In chapter 5, a second video matting algorithm is presented that uses reconstruction error as a measure of the matte. Two patch-based frameworks for temporally coherent matte extraction are discussed. Finally, the conclusions of thesis and some directions for future research in image and video matting are outlined in chapter 6.

Chapter 2

Literature Review

2.1 Introduction

The problem of accurate foreground extraction from images and video has been studied extensively over the past two decades. One of the major reasons for interest in the field was due to the emergence of special effects for television and film production. The mathematical formulation of the matting problem for foreground extraction was established by Porter and Duff [1]. The alpha channel was introduced as a blending factor to control the linear interpolation of foreground and background colors as described by the compositing equation. Alpha matting can be generally divided into two main categories- α -propagation based and color sampling based.

The α -propagation based methods [3, 4, 13–16, 31–34] assume correlation between the neighboring pixels under some image statistics and use their affinities to propagate the alpha values from the known regions to the unknown ones. The color sampling methods [9–12, 35–39] sample known foreground (F) and background (B) colors using specific sampling criteria and estimate the alpha matte using the best (F, B) pairs for the unknown pixels. Also, there exists methods, which combine both the previous terms into an energy optimization framework and solve for the alpha matte which minimizes the energy of the objective function. Video matting systems generally follow a two step framework - binary segmentation followed by matting at the foreground boundary region. Table 2.1 highlights the comparison between the two categories of approaches and their properties. In the following sections, we review significant image matting methods in both

Table 2.1: Comparison of sampling-based approaches and propagation-based approaches in matting.

	Sampling-based approaches	Propagation-based approaches
User input	Usually trimap-based	Usually scribble-based
Strengths	Intuitive and straightforward to use color information Works well in unknown smooth regions where samples are reliable. Different sample selection and refinement criteria have produced the best results so far	Smoothness is maintained based on the local affinity assumption More robust to complex color distributions Different features and structure of affinity matrix distinguish the methods
Weaknesses	Constrained by information in the trimap Appropriate samples are difficult to collect in textured regions	Can leverage only the local information unlike broader search regions in sampling approaches Textured regions block the propagation of alpha
Computation time	Non-parametric approaches are generally more time-consuming	Can be solved very efficiently as linear system of equations based on the affinity matrix
Memory usage	Less memory intensive	More memory intensive

categories followed by methods on video matting. A comprehensive survey on image and video matting is presented in [40, 41].

2.2 Color sampling based approaches

Color sampling approaches can be further sub-divided into parametric and non-parametric methods. The early approaches fall into the former category where parametric statistical models are fit to the known foreground and background samples. The alpha value is then estimated by considering the distance of the unknown pixels to these known color distributions. Early representative methods include Ruzon and Tomasi’s [35], Bayesian [36], Mishima’s Blue Screen method [37] and Corel’s Knockout algorithm [38]. They generate large fitting errors for textured regions, where it is insufficient to model the higher order statistics of color distribution. It tends to produce weak mattes when the trimap is coarse, leading to unreliable correlations between unknown pixels and F and B samples. Non-parametric methods [9–12, 39] on the other hand, collect a subset of known F and B samples and estimate the alpha matte from the best (F, B) pair found through an optimization process. Different sampling strategies and final pair selection criteria distinguish these methods. However, the quality of the estimated matte is highly dependent on the collected samples and the matte quality degrades if the true foreground and

background colors are not present. The main challenge of these methods is to obtain a comprehensive set of samples which describe the true F and B colors at that pixel using the compositing equation. A few important color sampling methods are reviewed here.

2.2.1 Ruzon and Tomasi's method

One of the earliest parametric matting methods was proposed by Ruzon and Tomasi [35] which uses non-oriented Gaussian distributions to model the foreground and background samples in a local neighborhood around the unknown region. The mixture of Gaussians from the foreground with their respective means and covariances are paired up with those from the background region and linearly interpolated to construct intermediate distributions for the unknown pixels. The alpha value corresponds to the intermediate distribution for which the observed color has maximum probability. The use of non-oriented Gaussians works well when the color variance of the local regions are small, but produces large fitting errors for textured regions. Also, the trimap is assumed to be drawn finely with a narrow band for the unknown region. An increase in the size of the band can affect the result as the local Gaussian distribution may fail.

2.2.2 Bayesian matting

Ruzon's approach was further improved in Bayesian matting [36] by using a continuously sliding window over a neighborhood, which marches inward from the foreground and background regions and utilizes nearby computed F , B and α values to construct oriented Gaussians. Alpha computation is formulated using a Bayesian framework and solved using maximum a posteriori (MAP) technique. It works well when the assumptions of a well-defined trimap and separable color distributions hold. However, as with Ruzon's method, it fails in textured regions or areas of complex color distribution as the Gaussian distribution fails to model the foreground and background accurately. Fig. 2.1(a,b) show an input image containing texture and complex color distribution along with its trimap. As can be seen from Fig. 2.1(c), Bayesian matting fails to extract a good matte since the texture in the book cover is misclassified as belonging to the foreground due to incorrect modeling of the Gaussian.

2.2.3 Knockout system

One of the early non-parametric natural matting methods was introduced by Corel in their Knockout system [38] where the F and B colors of the unknown pixels were approximated as the weighted sum of known F and B samples nearby. The weights are proportional to the distance of the known samples from the given pixel, greater distance leads to lower weights. The alpha value is estimated as a weighted sum of alpha values computed using these F and B values in each color channel. The simple assumption works well for images with well separated foreground and background, but fails when the true samples are missing from the sample set.

2.2.4 Robust matting

Wang *et al.* [8] proposed a non-parametric sampling method to select F and B samples that best represents the true foreground and background colors at an unknown pixel. Ten spatially closest foreground and background samples are chosen as shown in Fig. 2.2(a) which are then paired up to form an exhaustive set of (F, B) pairs for an unknown pixel. These (F, B) pairs are refined to select the 3 best pairs that satisfy the distance ratio, which is defined as the ratio between color fitness and the distance between the foreground and background colors of the (F, B) pair, i.e.,

$$R_d(F_i, B_j) = \frac{\|I_z - (\hat{\alpha}F_i + (1 - \alpha)B_j)\|}{\|F_i - B_j\|}. \quad (2.1)$$

As can be seen from eq. (2.1), distance ratio prefers (F, B) pairs widely separated in color space. The (F, B) pairs are also ranked on the basis of their color distance to the unknown pixel. The confidence of each (F, B) pair is computed based on the distance ratio and the rank of the sample pair among all the other pairs. Once the best (F, B) pair is selected, the alpha is computed using eq. (1.2). Robust matting performs better than the Knockout system as only good samples that linearly represent the observed color are used for matting. However, it fails to generate a good alpha matte for images where the foreground and background colors overlap considerably. The doll's hair and the curtain have similar color distribution causing the latter to be mistaken as part of the foreground in Fig. 2.1(d).

Figure 2.1: Comparison of mattes generated by sampling-based methods. (a) Original Image, (b) trimap, alpha mattes of (c) Bayesian matting [36], (d) Robust matting [8], (e) Shared matting [9], (f) Global matting [10], (g) Weighted Color and Texture matting [12] and (h) Comprehensive sampling [11]. Errors are highlighted by the yellow arrows.

2.2.5 Shared matting

Shared matting [9] utilizes an efficient sampling strategy to reduce the redundancy in selecting the (F, B) sample set for nearby pixels. The authors observe that nearby pixels share similar values for the (α, F, B) triplet, and therefore, divide the samples among themselves in the sample gathering stage. These results are shared and refined in the sample refinement stage. In the sample gathering stage, for each unknown pixel, a set of known F and B samples that are disjoint from its neighbors are collected. This is ensured by emanating rays from the unknown pixel at equal angles to the foreground and background regions and collecting the samples along the boundaries as shown in Fig. 2.2(b). The sample refinement stage selects the best samples with respect to spatial, photometric and probabilistic attributes of the image according to the function

$$(F_z^*, B_z^*) = \underset{(F_i, B_j) \in \Omega_z}{\operatorname{argmin}} (N_z(F_i, B_j) A_z(F_i, B_j) D_z(F_i) D_z(B_j)), \quad (2.2)$$

where Ω_z is the set of (F, B) pairs for the unknown pixel z and $N_z(F_i, B_j)$ measures the color fitness in a 3×3 neighborhood around z . $A_z(F_i, B_j)$ ensures that the computed alpha value correlates with the probability of belonging to the foreground. These values are then shared with the neighboring pixels in the sample refinement stage to better

Figure 2.2: Illustration of sampling strategies of (a) Robust matting [8], (b) Shared matting [9], (c) Global matting [10] and (d) Comprehensive Sampling [11].

estimate the alpha. The estimated matte is further refined using Laplacian refinement[6] which will be explained later (sec. 2.3.3). As the redundancy in sampling stage is reduced, this method allows for real-time implementation as well. However, like Robust matting, it suffers from missing the true samples as sampling is done only along the boundary. Moreover, when F and B regions have similar colors then the cost function cannot reliably find the best (F, B) pairs for unknown pixels that leads to erroneous matte. In Fig. 2.1(e), the blue color of the text does not feature as a sample from the foreground region causing the compositing equation to consider it as a foreground regardless.

2.2.6 Global sampling matting

Global sampling matting [10] further improved on the sampling strategy by collecting a global set of foreground and background samples from the boundaries of the known regions as shown in Fig. 2.2(c). The presence of a vast sample set is handled using an efficient random search based on PatchMatch [42] and a simple objective function. The objective function considers the chromatic distortion term that ensures a good fit for the sample pair with the compositing equation and a spatial cost term for favoring spatially close samples in order to find the best (F, B) pair as

$$E(F_z^*, B_z^*) = \underset{(F_i, B_j) \in \Omega}{\operatorname{argmin}} (wE_c(F_i, B_j) + E_s(F_i) + E_s(B_j)), \quad (2.3)$$

where Ω is the set of all (F, B) pairs collected from the boundaries of known regions. The E_c and E_s terms are similar to [8, 9] and encode the chromatic distortion and spatial priority cost in the pair. As with other sampling approaches, once the best (F, B) pair is found, alpha is estimated using eq. (1.2). Two different post-processing strategies - using guided filter [43] and Laplacian refinement of [3] are used to smooth the matte. Although it collects a large subset of samples, true samples may be missed if they are not present on the boundary of the trimap, making the results highly dependent on the quality of the trimap. As with other sampling methods, use of the best (F, B) pair does not work well in textured regions like the book cover and folder as shown in Fig. 2.1(f).

2.2.7 Weighted color and texture matting

One major drawback of color sampling based methods is the handling of (F, B) pairs with overlapping color distribution. Color alone cannot distinguish these regions and hence the estimated matte is not reliable. Weighted color and texture matting [12] tries to alleviate this problem by considering texture as a feature that complements color to improve matting. The sample selection process is carried out on two images - the original color image and the scaled three dimensional texture image obtained from Haar wavelet coefficients of the original image. The selection of the best (F, B) pair is carried out by maximizing an objective function

$$O = (C_\alpha)^{e_c} \times (T_\alpha)^{e_t}, \quad (2.4)$$

where C_α measures the color fitness of a particular (F, B) pair and T_α denotes the compatibility between the texture and color feature. The weights e_C and e_T are automatically selected based on the overlap of a given (F, B) pair- greater the overlap, greater is the influence by the texture feature in deciding the alpha matte as color will not be reliable in such regions. The best (F, B) pair is determined by brute force optimization, followed by a Laplacian post-processing step [3] to smooth the matte. The use of texture along with color generates better estimates in textured areas of complex color distribution, but fails to distinguish them in non-textured areas. The heater in the bottom left of the image is wrongly identified as foreground despite its dissimilarity with the foreground as shown in Fig. 2.1(g).

2.2.8 Comprehensive sampling

Most sampling based methods incorporate a sampling strategy that select (F, B) pairs that lie on the boundaries of the definite foreground and background regions. However, as shown in Fig. 2.2, these methods suffer from the *missing true samples* problem. The true foreground or background colors may not be present near the edges of the known regions leading to inconsistency in the matte. With this key idea in mind, comprehensive sampling [11] adopts a sampling scheme that collects a comprehensive set of samples from within the known F and B regions that encompass a wide range of color distributions in the image illustrated in Fig. 2.2(d). Sample selection follows a novel parametric two-level hierarchical clustering through Gaussian mixture models (GMM). The best (F, B) pair is selected by optimizing an objective function that accounts for the chromatic, spatial and color statistics in the image,

$$O_z(F_i, B_i) = K_z(F_i, B_i) \times S_z(F_i, B_i) \times C_z(F_i, B_i), \quad (2.5)$$

where K accounts for chromatic distortion, S enforces spatial constraints and C ensures that the (F, B) pair are widely separated in color space utilizing Cohen's d value for color distributions. Although a comprehensive set of samples have been collected, this fails to address the highly textured regions like the book's cover in Fig. 2.1(h) as a single (F, B) pair cannot represent the true observed color.

2.3 Alpha propagation based approaches

As the name suggests, these methods propagate the alpha from the known regions to the unknown regions using assumptions on the local image statistics and matte gradient field. The affinities between neighboring pixels in a small neighborhood (usually 3×3 window) are utilized to propagate the alpha value across the image. Here, we review current matting methods that use propagation.

2.3.1 Poisson matting

Poisson matting [13] assumes that the color variation between foreground and background is smooth, leading to the assumption that the matte field is smooth and is obtained by solving Poisson equations. In order to approximate the matte gradient, partial derivatives of the compositing equation are taken, followed by eliminating the gradient with respect to F and B (as they are assumed constant) to yield

$$\nabla \alpha = \frac{1}{(F - B)} \nabla I, \quad (2.6)$$

where ∇ is the gradient operator. The matte gradient is proportional to the image gradient and is available when $(F - B)$ is estimated. The simplest way to estimate F and B is to use nearest foreground and background colors and then Poisson equations are solved on the image lattice to recover the final matte. If F or B is not smooth enough in the unknown region, ∇F or ∇B will not be relatively small and eq. (2.6) becomes invalid. Thus, the user needs to locally manipulate the gradient matte to obtain high quality mattes. Fig. 2.3(a) shows the plant image and 2 sets of trimaps in Fig. 2.3(b,e). As can be seen, the F and B colors are not smooth in the unknown region presented by large trimap in Fig. 2.3(e), and hence, the estimated matte with Poisson matting is not accurate as shown in Fig. 2.3(f).

2.3.2 Geodesic matting

A scribble-based matting algorithm was proposed by Bai *et al.* [14] that uses weighted geodesic distance to propagate the alpha from user provided scribbles to unknown ones.

Figure 2.3: (a) Original Image, (b) Small and (e) large trimap, alpha mattes of (c,f) Poisson matting [13], (d,g) Geodesic matting [14].

The geodesic distance is defined as the smallest integral of weighted distances from the unknown pixel to the known regions. The alpha matte is then estimated as

$$\alpha_z = \frac{D_F(z)}{D_F(z) + D_B(z)}, \quad (2.7)$$

where $D_F(z)$ and $D_B(z)$ are geodesic distances of pixel z to foreground and background, respectively. Weighting functions are solved as first order geometric Hamilton-Jacobi equation in computationally optimal linear time. However it fails when the foreground and background colors overlap since the geodesic distances cannot discriminate between them. Fig. 2.3(d,g) illustrates this as seen by the leaf in the background being picked up as part of the foreground.

2.3.3 Closed-Form matting

A closed form solution to the matting problem was demonstrated by Levin *et al.* [3] by analytically deriving a cost function based on smoothness assumptions on local foreground and background colors in a natural image. Based on this key assumption, F and B are eliminated from the cost function resulting in a quadratic function in α , which can be solved using a sparse linear system of equations. The color line model assumes that each of F and B in a small window w (3×3 or 5×5) lies roughly on a line in the color

Figure 2.4: Comparison of α -propagation methods. (a) Troll image, (b) trimap, alpha matte using (c) Closed-form matting [3], (d) KNN matting [16], (e) Learning-based matting [44], (f) LNSP matting [17].

space. The alpha can then expressed as,

$$\alpha_i = \sum_c a^c I^c + b, \quad \forall i \in w, \quad (2.8)$$

where c is the color channel and a^c and b are constant within the window. The alpha is solved by minimizing a cost function resulting in a sparse linear system of equations:

$$(L + \lambda D)\alpha = \lambda b_s, \quad (2.9)$$

where L is a $N \times N$ matting Laplacian matrix (N is the number of pixels in the image), λ is a constant weighting parameter, D is a diagonal matrix whose diagonal elements are 1 for known pixels and 0 for unknown pixels and b_s is $N \times 1$ vector with alpha values for the known pixels and 0 for others.

The color line model assumption fails to uphold in highly textured regions due to the strong edges that block the propagation of alpha. Overlapping color distributions at the F and B regions present another challenge as the false color correlation among the neighboring F and B tends to wrongly propagate the alpha as shown in the bridge in Fig. 2.4(a)

being wrongly classified as part of the foreground in Fig. 2.4(c). Large kernels [32] for constructing the Laplacian improves on such regions to an extent, but leads to prohibitive memory and computational requirements. Iterative transductive matting [33], color clustering matting [34] and [45] modify the Laplacian matrix construction to better encode the correlation among the pixels.

2.3.4 KNN matting

The local color line model of [3] is replaced in KNN matting [16] by considering nonlocal principle to formulate the affinities among K nearest neighbors in a nonlocal neighborhood. The $N \times N$ affinity matrix is constructed by collecting nonlocal neighborhoods j of a pixel i and computing the similarity between the feature vectors using a kernel function defined as:

$$k(i, j) = 1 - \frac{\|X(i) - X(j)\|}{C}, \quad (2.10)$$

where $X(\cdot)$ is the feature vector and C is a normalizing constant. With the Laplacian matrix constructed, the optimal alpha can be estimated by solving a sparse linear system of equations using iterative conjugate gradient.

As shown in Fig. 2.4(d), KNN matting shows improved performance than closed-form matting in textured areas due to nonlocal neighborhood affinity. However, it is unable to produce good results in areas of overlapping F and B color distribution.

2.3.5 Learning-based matting

Learning-based matting [44] transforms matting into a semi-supervised learning problem. It assumes that the alpha value of each pixel is a linear combination of alpha values of neighboring pixels and estimates coefficients of the combination using a learning process. Parametric alpha-color models are learnt from foreground and background regions which are used to infer the matting coefficients of the unknown region. A local learning process trains dependencies between the neighboring pixels, relaxing the linear color model assumption of closed-form matting and propagates the information along the neighborhood. The global learning process selects F and B samples weighted by weighted ridge regression technique to train the global alpha-color models. Overlapping color distributions increases the false correlation among F and B samples leading to degraded matte in such regions as illustrated in Fig. 2.4(e).

2.3.6 Iterative transductive matting

Bei *et al.* [33] used transductive learning that integrates the unlabeled pixels along with the labeled ones in the learning process. An asymmetric Laplacian matrix is constructed where $L(i, j)$ and $L(j, i)$ are not calculated repeatedly thereby reducing the redundancy in the matrix calculation. The optimization function is written as

$$\min \alpha^T \mathbf{L} \alpha + \lambda_l (\alpha - \alpha^*)^T \mathbf{I}_l (\alpha - \alpha^*) + \lambda_u (\alpha - \alpha^*)^T \mathbf{I}_c (\alpha - \alpha^*), \quad (2.11)$$

where $\mathbf{I}_l$ and $\mathbf{I}_c$ are diagonal matrices with value 1 for labeled pixels and confidence value for unlabeled pixels respectively. λ_l and λ_u are the Lagrange constraint coefficients for labeled and unlabeled pixels respectively. The objective function is solved iteratively by adjusting the constraint coefficients gradually from small to large until a balanced alpha matte is obtained.

2.3.7 Local and non-local smooth prior (LNSP) matting

Chen *et al.* [17] proposed a propagation-based image matting algorithm by combining the nonlocal and local smoothing priors of previous affinity based approaches together with a color sampling based prior from Robust matting [8]. The three priors are formulated together in a weighted graph model. The resulting energy function contains three terms - color sampling as the data term, matting Laplacian as the local smoothness term and locally linear embedding (LLE) of K-nearest neighbors as the nonlocal smoothness term. The cost function is quadratic in α and can be minimized by solving a linear system of equations to generate the matte. The advantage of the method lies in its ability to utilize the advantage of both nonlocal and local neighbors in propagating the matte as most of the bridge is removed as shown in Fig. 2.4(f). However, the color sampling term influences the final matte and a poor estimate due to sampling false (F, B) pairs would drastically affect the matte.

2.4 Video Matting

In image matting, the information required to model the alpha matte is present in the image itself. However, the extension to videos becomes more challenging, since the trimap

Figure 2.5: Illustration of temporal inconsistency in video matting. First row shows the frames of a video sequence and per-frame matte extracted by comprehensive sampling [11] is shown in the second row.

or user constraints provided at the start of a video frame may not be conducive for the upcoming frames. Fast motion and low contrast of video sequences make it extremely difficult to extract temporally consistent mattes. In this section, we review existing literature in video matting to highlight the above challenges.

The quality of the estimated video matte depends mainly on two factors - spatial accuracy and temporal coherency [29]. Spatial accuracy indicates quality of the matte in a single frame, while temporal coherency requires consistency of the matte across the frames in the video. Temporal coherence is more important as the human visual system is more sensitive to temporal inconsistency in a video which results in jitter. The dynamic behavior of the F and B regions in video makes it highly challenging for the propagation of trimaps across the video whose quality is paramount for temporally coherent matte extraction. Even if spatial accuracy is maintained in each frame, the matte can exhibit temporal jitter if temporal coherency is not maintained. Fig. 2.5 illustrates this problem where the temporal inconsistencies are highlighted in the yellow windows. The third column shows the region between the shirt and the woman's hair having inconsistencies from the frames preceding and after it. During playback, the viewer sees this inconsistency as jitter.

Video matting methods [20, 27, 46–51] generally follow a two-step framework. User interaction is utilized to generate the trimap using binary segmentation in each frame.

Given the trimaps, matting algorithms are run in the second step to refine the foreground boundary. In the binary segmentation step, various methods like graph-cut optimization [20, 48], depth maps [52] and geodesic distances [14] have been used. The segmented boundaries are dilated by a few pixels to generate the trimap for the matting step. However, such a technique fails in images having large number of transparent pixels, like hairy regions or translucent objects. Since it is prohibitive to obtain user interaction for each frame, some methods [27, 28, 46, 53] propagate the trimaps obtained on these key-frames across the sequence. As mentioned before, trimap propagation is crucial for preserving the temporal coherency. A poorly propagated trimap may maintain the spatial accuracy of the matte in a particular frame, but does not ensure temporal coherency for the whole sequence. Different techniques for trimap propagation have been proposed such as optical flow [54–56], rotoscoping [57] and inferential statistics [58].

Once the trimap is obtained for each frame using either binary segmentation or trimap propagation, simply applying image matting algorithm frame-by-frame [27, 48, 59, 60] does not result in temporally coherent mattes by ignoring the content in the surrounding frames. Bayesian video matting [27] utilizes forward and backward optical flow for trimap propagation between key-frames followed by a Bayesian matting [36] step on the individual trimaps. The video cut-out system [48] considers the spatio-temporal volume to obtain binary segmentation by applying graph-cut on a 3-D graph constructed by connecting granular regions across frames. The granular regions are first obtained using pre-segmentation of the frames using watershed algorithm [61]. The foreground boundary is dilated to obtain narrow unknown regions followed by an modified Bayesian matting [62] on it to extract the final matte. Interactive video cut-out [20] also uses graph-cut for binary segmentation with a volumetric painting-based interface for marking foreground object across the video volume. This is followed by using GrabCut [63] for border matting.

Extracting a matte independently frame-by-frame results in poor temporal coherence as shown in Fig. 2.5. The per-frame estimated alpha optimizes the spatial accuracy individually, but needs to be optimized temporally as well. Some methods [26, 46, 53, 64, 65] try to solve this problem by extending the α -propagation approach temporally. The 3-D Laplacian matrix inherently introduces memory and computational constraints

when applied to videos. As discussed in the following sections, various ways to circumvent this have been developed. The temporal guided filter [66–68] has been applied to video sequences as a post-processing step to refine the video matte thereby enforcing temporal coherence. Below, we review a few significant methods in video matting.

2.4.1 Geodesic video matting

Bai *et al.* [14] utilize the weighted geodesic distance in the spatio-temporal space to segment foreground object, which is then refined to obtain the video matte. Scribbles marked in the first frame are propagated to the 3-D temporal volume using spatial and temporal gradients to guide the propagation. The color likelihoods are represented by kernel density estimation (KDE) for the calculation of the geodesic path. This fails to handle video sequences with overlapping color distributions where geodesic distances are unreliable, as in the case of images.

2.4.2 Video SnapCut

Video SnapCut [26] tries to mitigate the overlapping color distribution problem by using overlapping localized classifier windows for learning the foreground and background color models as well as prior shape models to guide the segmentation, before matting is carried out on the dilated boundaries of the segmented result. Each classifier in the window contains a local color model of F and B built from Gaussian mixture models (GMM), a shape model and color confidence. The local windows, centered on the foreground boundary are propagated forward to the next frame using optical flow. Given the binary mask, the trimap is generated by dilating and eroding the boundary and coherent matting [62] is applied to estimate the matte.

$$\alpha^t = \underset{\alpha^t}{\operatorname{argmin}} \sum_{x \in U} [\lambda_x^T (\alpha_x^t - \hat{\alpha}_x^{t-1})^2 + \lambda_x^C (\alpha_x^t - \tilde{\alpha}_x^t)^2] + \lambda^S \alpha^{tT} L \alpha^t, \quad (2.12)$$

where U represents the unknown regions. The first term enforces temporal coherence with the alpha prior computed from the previous frame and the second term is the color coherence term and the last term is the Laplacian smoothing term. λ_x^T, λ_x^C and λ^S are weighting constants. The final quality of the matte depends on the propagation of the classifier windows and errors, if any, are propagated to the future frames in areas of

Figure 2.6: Mattes obtained by [53] using temporal sampling to extend Robust matting to the 3-D domain. The yellow arrows indicate discontinuities due to low contrast in the sampling process.

complex color distributions and occlusions. Since optical flow is often erroneous, the warped alpha is often inaccurate to serve as the temporal prior for the current frame. This is evident from the *Office* sequence shown in Fig. 2.7 where the cluttered background produces errors that are propagated to the following frames.

2.4.3 Robust video matting using temporal neighbors

Robust matting [8] is extended to the video domain by Lee *et al.* [53] by sampling from the immediate neighboring frames. The previous and next frames are considered when estimating the matte of the current frame. A random walk based optimization that uses optical flow field is used to define the neighboring samples. The temporal sampling does not guarantee the samples to be reliable in videos of low contrast and produces jitter in the extracted mattes. Fig. 2.6 illustrates the inconsistencies in yellow arrows where the coat and the wall in the background have similar color distribution leading to samples being unreliable for matting.

2.4.4 Video matting using multi-frame nonlocal Laplacian

Choi *et al.* [64] extend the nonlocal Laplacian matting [31] to videos by considering a temporal block of frames together to ensure temporal coherence of the matte. A 3-D nonlocal neighborhood is constructed using kd-tree to constrain the extensive memory requirement and Approximate Nearest Neighbor (ANN) search is utilized to build the matting Laplacian matrix.

$$\alpha^T \mathbf{L}_F \alpha = \sum_{f=1}^F \sum_{i=1}^N \alpha(N_{3D}(i))^T \bar{G}_i^T \bar{G}_i \alpha(N_{3D}(i)), \quad (2.13)$$

where $\mathbf{L}_F$ is the multi-frame Laplacian constructed from F frames, each with N pixels. $N_{3D}(i)$ denotes the 3D nonlocal neighbors of a pixel i . The matte is estimated together for F frames by solving a linear system. The estimated alpha is further refined for flickering effects by using nonlocal mean regularization. The use of multiple frames in solving the matte is a crucial factor in better temporal coherence, however implementation is complicated with a specialized data structure. A matte regularization step is required to maintain temporal coherency after solving the mattes. The multi-frame video matting method proposed in chapter 4 is able to estimate the final mattes of multiple frames at once and does not require a post-processing step.

2.4.5 Temporal matte filtering based video matting

The importance of temporal coherence in a video matte is explicitly handled in [28] by incorporating a temporally coherent trimap propagation step and a novel temporal matte filter. An initial binary segmentation is obtained by using the video snap-cut system [26] followed by user interaction to specify trimaps on selected key-frames. The parameterized trimap in a key-frame is propagated to the next frame using optical flow and thin-plate interpolation to account for errors in optical flow computation. Robust matting [8] is used to generate the matte on the current frame. A post-processing step is applied using level set based interpolation between the mattes to temporally filter the estimated mattes for better temporal coherence. While this method reduces the temporal inconsistencies in the matte, it tends to oversmooth the resulting matte, thereby losing out on the fine details as shown in the third row of Fig. 2.7.

Figure 2.7: Temporal inconsistencies in video matting. Four consecutive frames of the *Office* sequence are shown in the first row and the mattes in the zoomed regions obtained for Snap-cut [26], Bai *et al.* [28] and Shahrian *et al.* [30] are shown in the second, third and last row, respectively.

2.4.6 Discontinuity-aware video cutout

Occlusion of foreground objects in a video sequence is a major challenge for any computer vision research related to videos. Foreground extraction suffers drastically due to the presence of occlusions in video. Multiple classifiers are employed to predict the probability map in [69], after which, a learning based step from real world examples is utilized to

combine the foreground probability map. This foreground probability map serves as the temporal prior for the coherent matting step similar to [26] to extract the matte. As color statistics are used for classification, this method inevitably suffers from complex foreground-background boundaries. Also, as the segmentation results are propagated by binary classifiers, it cannot be applied to semi-transparent objects or hairs.

2.4.7 Motion-aware KNN video matting

Li *et al.* [65] extended their KNN image matting algorithm [16] by using motion flow as an additional feature for formulating the 3-D KNN Laplacian. While most methods use optical flow in the trimap propagation stage, the authors argue that similar pixels should have similar motion in addition to appearance during matting. Combined with the color and spatial information, a motion-aware two-frame affinity matrix of size $2N \times 2N$ is built and solved in closed-form where N is the number of pixels in a frame. Low contrast and temporal resolution makes optical flow unreliable causing wrong propagation of temporal information, especially at the foreground object boundaries.

2.4.8 Temporally coherent and spatially accurate video matting

Shahrian *et al.* [30] proposed a sampling-based video matting algorithm that extends the sampling strategy of [11] to videos by including temporal samples from different color distributions over many video frames. An objective function is used to select the best (F, B) pair that takes into account the color, texture and temporal information and is defined as

$$E = (C_\alpha)^{e_C} \times (T_\alpha)^{e_T} \times (O_\alpha)^{e_O}. \quad (2.14)$$

Here, the first two terms measure the color fitness and texture penalty and the third term measures the compatibility of alpha in the current frame with the previously estimated alpha from preceding frames using warped optical flow. e_C , e_T and e_O are weights controlling the influence of each term towards the cost function. The estimated alpha is then refined using a Laplacian-based post-processing step. Although there exists an explicit temporal term in the objective function, alpha estimation is still performed independently at each pixel. This leads to inconsistencies in areas of high texture and

background clutter as optical flow is often erroneous in such cases. This is illustrated in the last row of Fig. 2.7 where discontinuities are present in the hair of the person.

Zou *et al.* [70] proposed a video matting method that builds on the sparse coding method proposed in chapters 3 and 4 by adding a low-rank constraint to the sparse coding problem. A dictionary learning step is also added to increase the discriminative ability of the foreground and background samples. However, the results seem to be improving mainly due to the effect of the Laplacian refinement step. A video matting evaluation benchmark [71] was introduced recently, however the methods being compared were image matting algorithms applied per frame. Previous work has made significant progress towards improving temporal coherency, but they still struggle in many instances. Most methods incorporate a 3-D extension to α -propagation, which can fail if the pixels in the neighborhood are too distant in space or time, especially in low-contrast F and B regions. Using temporal filters [66–68] can reduce the temporal jitter from the extracted matte, but tend to oversmooth the finer structures like hair in the video. This trade-off between spatial quality and temporal coherence should be the focus for future works in video matting.

Chapter 3

Sparse Codes as Alpha Matte

3.1 Introduction

Color sampling based approaches rely on known foreground and background colors as approximation to the true foreground and background colors of the unknown pixel in estimating the matte. The candidate samples from the foreground and background regions are paired together followed by selection of the best (F, B) pair through optimization of a cost function that takes the distance and chromatic distortion into account. The compositing equation is used to estimate the alpha value from the selected (F, B) pair and the quality of the matte depends on the selected pair. However, such approaches fail to pull an accurate matte when the color distribution of the foreground and background overlap, or when highly textured regions are present leading to unreliable sample pairs being selected. Also, α -propagation based methods fail to extract good quality matte on images with complex color distributions or textured images since the strong edges block the propagation of alpha. The problem is illustrated in Fig. 3.1 where the doll image contains two windows, depicting regions of complex distributions and texture and smooth hairy regions. Sampling-based methods like Bayesian matting [36], Comprehensive sampling [11] and Weighted Color and Texture matting [12] that use a single (F, B) pair are unable to reconstruct the pixel colors due to the texture in the book cover as shown in the top row of Fig. 3.1(c-e). The use of an explicit texture feature in [12] is still insufficient to model the textured pattern of the book. α -propagation based methods such as LNSP matting [17] oversmooth the matte in hairy regions as shown by the zoomed window in the bottom row in Fig. 3.1(f).

Figure 3.1: Visual comparison of alpha matte generated from multiple unpaired F and B samples using sparse coding with other state-of-the-art methods. (a) Original image with foreground (red) and background (blue) boundaries, (b) zoomed regions, (c) Bayesian matting [36], (d) Comprehensive sampling [11], (e) Weighted Color and Texture matting [12] (f) LNSP matting [17] and (g) proposed method.

In this chapter, we address the problem of selecting the best F and B samples from a given sample set that leads to the best alpha estimate at a pixel. The proposed method is based on sampling. However, there is a fundamental difference from other sampling-based approaches in the way alpha is estimated as illustrated in Fig. 3.2. For a given unknown pixel of color I in yellow, all sampling methods collect a subset of labeled F and B samples in red and blue respectively, from a local/global region depending on the sampling strategy. Current methods form an exhaustive set of (F, B) pairs by pairing each F sample with all the other B samples as illustrated in Fig. 3.2(a). The (F, B) pair that best 'represents' the true foreground and background is chosen from candidate pairs through optimization of a cost function. α is then determined by eq. (1.2). It is worth mentioning that Robust matting [8] selects 3 best pairs instead of just one and averages the alpha obtained for each pair. However, alpha is still estimated using (F, B) pairs. In the proposed method, estimation of α is not governed by eq. (1.2); instead, for a given unknown pixel, multiple F and B *unpaired* samples that best reconstruct it, are picked from a subset of labeled samples. The sum of the reconstruction coefficients directly gives α . This reinterpretation allows the matting framework to determine α based on more relevant F and B samples than with only one of each. As can be seen from Fig. 3.1(g) a good quality matte is extracted by using a better reconstruction of the observed color

Figure 3.2: Illustrative comparison of matte estimation between methods using paired samples and the proposed approach. (a) Pairwise approaches use pair/s of F and B samples to estimate α at a pixel. (b) The proposed approach eliminates the need to restrict matte estimation using (F, B) pairs. Instead, samples that produce least sparse reconstruction error at a pixel are selected.

at the pixel. The proposed framework is computationally faster as it removes the need of an expensive 2-D search for the best (F, B) pair as in [10–12].

3.2 Using sparse coding to generate alpha matte

The tool that enables the use of more than one F and B sample to directly estimate α as a sum of the samples' weights is sparse coding. The success of sparse coding in various computer vision tasks such as face recognition [72], super-resolution [73] and

image classification [74] can be attributed to the fact that images generally lie on low-dimensional subspaces or manifolds from which representative samples could be picked based on an appropriate basis [75]. In a general sense, the main aim of sparse coding is to estimate a good reconstruction of a signal based on linear combinations of ‘few’ good signals taken from a dictionary or codebook [75]. Mathematically it can be represented as an l_1 -norm regularized problem:

$$\beta_i = \operatorname{argmin} \|v_i - \mathbf{D}\beta_i\|_2^2 + \lambda \|\beta_i\|_1, \quad (3.1)$$

where $\mathbf{D}$ is the dictionary containing basis vectors of the feature space, v_i is the i^{th} feature (signal) to be coded and β_i is the sparse code for the given feature vector. λ is a weighting parameter that controls the sparsity in the generated code. Many algorithms to solve this problem efficiently have been extensively studied in literature [76–78].

The only other method that uses sparsity for matting is compressive matting [79]. There are two important differences between their method and ours. First, they use a fixed window from which the dictionary is formed. This severely restricts the possibility of obtaining a very incoherent dictionary since images are generally smooth over a small neighborhood. Second, they map the sparse codes to the α values using a ratio of the l_2 -norm of the sparse codes of F pixels to the sum of l_2 -norms of sparse codes of F and B regions. In our approach, by using special constraints on the sparse representation, the sum of the sparse codes for F directly give α .

As indicated earlier, the motivation of the proposed method is to determine α from a bunch of F and B samples in a sparse coding framework so that the codes directly provide the α values. The unknown pixels are categorized into high-certainty and low-certainty to allow for a comprehensive set of samples for dictionary formation. This is done through probabilistic segmentation of the image. Sparse coding at each pixel using the appropriate dictionary yields the α value. In this context, sample selection is presented first.

3.2.1 Sample selection for universal set

We use a pre-processing step to expand the known regions to unknown regions based on certain chromatic and spatial thresholds. An unknown pixel i is considered as foreground

Figure 3.3: Illustration of the proposed sampling strategy. (a) Input Image, (b) trimap, (c) image segmented into superpixels using [80] and (d) universal set of F and B samples.

if, for a pixel $j \in F$ [11]

$$(D(i, j) < E_{thr}) \wedge (\|I_i - I_j\| \leq (C_{thr} - D(i, j))), \quad (3.2)$$

where $D(i, j)$ is the Euclidean distance between pixel i and j in the spatial domain, I_i is the color at pixel i , and E_{thr} and C_{thr} are thresholds empirically set to 12 and 4, respectively. A similar formulation is applied to expand background regions.

Previous approaches usually sample known F and B samples from either the boundary pixels or randomly from within the known region as shown earlier in Fig. 2.2. The proposed sampling method is inspired by the comprehensive sampling approach of [11] in selecting samples from well within the known region boundaries to avoid missing true

Figure 3.4: Foreground probability for dictionary refinement. (a) Input image and its (b) foreground probability map, (c) zoomed window of the image and its (b) foreground probability map.

samples. The sample selection process is illustrated in Fig. 3.3. In order to reduce the sample space, we cluster labeled F and B pixels in a band of width 40 pixels along the boundary of the unknown region into superpixels using SLIC algorithm [80]. The size of each superpixel roughly varies depending on the image scene as shown in Fig. 3.3(c). The mean vector of each superpixel represents the F and B samples that make up the universal set. Fig. 3.3(d) shows the universal set for the input image in Fig. 3.3(a) where the foreground samples are represented as '+' and background as '.'. The feature vector used for coding is the 6-D vector $[R \ G \ B \ L^* \ a^* \ b^*]^T$ consisting of the concatenation of the two color spaces RGB and CIE-LAB, respectively. From the universal set, samples are chosen to form dictionaries based on the certainty of the pixel, as explained in section 3.2.3.

3.2.2 Certainty of unknown pixels

We wish to classify an unknown pixel as high-certainty or low-certainty depending on how well the colors of foreground and background regions are separated in the neighborhood of the unknown pixel. If an unknown pixel is labeled high-certainty, then it implies that the F and B colors in its neighborhood are well separated and consequently, the size of the dictionary can be small since we can then ensure that the dictionary will be formed from highly incoherent samples. Overlapping of F and B color distributions is one of the problems that recent sampling-based approaches like [11] try to address. Pixels with low-certainty come from areas that have potentially complex and overlapping color distributions requiring a larger dictionary so that the variability in color can be captured.

Since the feature used for coding is color and the complexity of a region for matting is dependent on the overlap of foreground and background colors, we use probabilistic

segmentation as a cue to label the unknown pixel as having low/high-certainty. We adopt a non-parametric sampling-based probability measure [81] to determine the probability of a given pixel i belonging to the foreground as

$$p(i) = \frac{p_f(i)}{p_f(i) + p_b(i)}, \quad (3.3)$$

where $p_f(i)$ is the foreground color affinity value given by

$$p_f(i) = \exp\left(-\frac{\sum_{k=1}^m \|c(i) - c(f_k)\|^2}{m \cdot \delta}\right). \quad (3.4)$$

Here $c(\cdot)$ is the RGB color value, m is the number of spatially close foreground samples (closeness is measured in terms of Euclidean distance from unknown pixel to centroid of a superpixel) and δ is a weighting constant. In our experiments, we set m and δ as 10 and 0.1 respectively. A higher value of $p_f(i)$ indicates that the pixel has higher affinity to F samples. A similar expression is applicable to $p_b(i)$.

Eq. (3.3) provides a probability value on whether a pixel belongs to the foreground. Fig. 3.4(c) is a zoomed region of Fig. 3.4(a) where there is an overlap in the foreground and background color distributions. Fig. 3.4(b) and (d) show the probability map for the image and zoomed window, respectively with higher intensity denoting higher probability for a pixel to belong to foreground. In areas of complex color distributions, the distance in color space of the unknown pixel to foreground and background pixels are similar. It has been observed that in such cases, the probability value as given by eq. (3.3) ranges from about 0.3 to 0.7. However, it is also observed that in regions with thin hairy structures whose color is significantly different from the background, the blending of colors causes the probability to also lie between 0.3 and 0.7. In order to distinguish such regions where the color distributions do not overlap, we consider a 7×7 neighborhood and classify the unknown pixel as *low-certainty* if the number of pixels with $p(i)$ in $[0.3, 0.7]$ is larger than 35 (fixed empirically). Fig. 3.5(a) shows an image marked with regions having foreground probability in $[0.3, 0.7]$. Fig. 3.5(b) shows the marked windows separately. The green windows are areas of complex color distributions. However, the yellow window at the top shows a hairy region with separable F and B colors. The zoomed regions of the probability map in Fig. 3.5(c) show that even in the hairy region, we can observe probabilities in $[0.3, 0.7]$ as shown in the thresholded mask in Fig. 3.5(d)

Figure 3.5: Defining low-certainty regions in complex areas. (a) Input image and (b) zoomed patches, corresponding (c) probability map, (d) thresholded mask for pixels in $[0.3, 0.7]$ and (e) low-certainty mask for each region.

. The additional neighborhood condition helps us to classify only the true complex color areas as the low-certainty mask, as shown in Fig. 3.5(e) in which 1 is assigned to every low-certainty pixel and 0 to high-certainty pixels.

3.2.3 Dictionary formation and sparse coding to α

The size of the dictionary for an unknown pixel depends on whether it is classified as *low-certainty* or *high-certainty*. If the pixel is of high-certainty, i.e., the color distributions of nearby F and B are well separated as determined by the probabilistic segmentation, then the dictionary is formed from 40 (chosen empirically) spatially closest F and B samples. Note that the samples here are the mean vectors of the spatially closest superpixels. As mentioned earlier, in such regions, the samples would be sufficiently incoherent for sparse coding.

The low-certainty pixels are potential areas of overlapping color distributions. In this case, the dictionary size is larger. Thus, for a given unknown pixel, one-third of the superpixels from the universal set that are closest in terms of Euclidean distance in the spatial domain constitute the dictionary. Fig. 3.6(a) and Fig. 3.6(b) show an input image and its corresponding trimap. The universal sample set of F and B regions in blue and red respectively, is shown in Fig. 3.6(c). For a given unknown pixel of low-certainty

Figure 3.6: Illustration of dictionary formation for sparse coding to generate the matte. Foreground and background regions are shown in blue and red respectively. (a) Input image, (b) trimap, (c) universal set of samples and (d) dictionary for a low-certainty pixel (green window) and high-certainty pixel (yellow window).

shown in green in Fig. 3.6(d), the dictionary is a larger subset of the universal sample set than that of a high-certainty pixel in yellow.

The 6-D color vector, which forms the feature vector for coding, is normalized to unit length. The dictionary is formed by concatenating the F and B samples horizontally as $\mathbf{D} = [F_1, F_2, \dots, F_n, B_1, B_2, \dots, B_n]$ which enables us to determine the location of each F and B atom during sparse coding. Given the dictionary $\mathbf{D}$ for an unknown pixel i , its sparse code is determined as

$$\beta = \operatorname{argmin} \|v_i - \mathbf{D}\beta_i\|_2^2 \quad \text{s.t.} \quad \|\beta_i\|_1 \leq 1; \beta_i \geq 0, \quad (3.5)$$

where v_i is the feature vector at i . The sparse codes β_i are generated using a modified version of the Lasso algorithm [78]. Given a dictionary D and a signal v , the algorithm

Figure 3.7: Initial alpha mattes generated by our proposed method.

returns coefficients β^j that are sparse, non-negative and less than or equal to 1. The procedure is outlined below:

- Start with all coefficients β^j equal to zero
- Find the predictor D^j most correlated with v .
- Increase the coefficient β^j in the direction of sign of its correlation with v . Take residuals and stop when another predictor D^k has as much correlation with residual

as D^j

- Increase (β^j, β^k) in their joint least squares direction. Set the non-negative coefficients to zero and recompute joint direction. If a non-zero coefficient hits 0, remove it from the prediction set.

The sparse coding procedure is presented with an appropriate set of F and B samples and the sparse coefficients sum up to less than or equal to 1. It has been observed that for most of the pixels, the sparse codes add up to 1. For the few exceptions, the sparse coefficients are normalized to sum up to 1. For example, in the trolls image shown in Fig. 3.2, out of the 124604 unknown pixels that underwent sparse coding, only 44 pixels had to be normalized as the corresponding codes did not sum up exactly to 1. In order to avoid negative values, the second constraint forces all coefficients to be non-negative. The sparse codes corresponding to atoms in the dictionary that belong to foreground regions are added to form the alpha for the unknown pixel. i.e.,

$$\hat{\alpha} = \sum_{p \in F} \beta^{(p)}. \quad (3.6)$$

Hence, the sparse codes directly provide the value of alpha. Fig. 3.7 shows the initial alpha mattes generated by our sparse coding method on five images from the dataset. The alpha mattes are visually similar to the ground truth as seen in the last 2 columns of the figure.

3.2.4 Graph-based matte optimization

As with other sampling-based approaches, the correlation between neighboring pixels are ignored while estimating the matte at each pixel. Post-processing is usually employed to smooth out the matte as in [11, 12]. We use the graph-based optimization followed by [8, 17] to obtain our final matte. A smoothness term consisting of the matting Laplacian [3] and the K -nearest neighbors [16] is combined with the initial sparse coded estimate $\hat{\alpha}$ as the data term in a graph model, and solved in closed-form as a sparse system of linear equations. Fig. 3.8 illustrates our graph model where red, blue and white nodes represent the pixels marked by the trimap as foreground, background and unknown respectively.

Figure 3.8: Illustration of graph model on image for optimization of the matte. Each pixel is connected to its spatial neighbors (W_{lap}), feature-space neighbors (W_{knn}) and to the virtual foreground (W_F) and background (W_B) nodes.

Two virtual nodes Ω_F and Ω_B , representing the foreground and background are connected to each pixel through the data weights $W_{i,F}$ and $W_{i,B}$ respectively.

We associate the initial estimate of the matte with a confidence value γ , which together indicates whether a given pixel belongs to the foreground or background. The data weights for pixel i are defined as

$$W_{i,F} = \gamma \hat{\alpha}_i, \quad W_{i,B} = \gamma(1 - \hat{\alpha}_i), \quad (3.7)$$

where a true foreground pixel should have higher value of $W_{i,F}$ and lower value of $W_{i,B}$, and vice versa. As γ encodes the confidence in the initial estimate, a larger weight is assigned to more confident estimates.

The confidence value at a pixel i is

$$\gamma(i) = \gamma_{sprec}(i) \cdot \gamma_{colrec}(i), \quad (3.8)$$

where $\gamma_{sprec}(i)$ measures the confidence in reconstructing the input feature vector based on the sparse coefficients and is formulated as

$$\gamma_{sprec}(i) = e^{-\|v_i - \hat{v}_i\|^2}. \quad (3.9)$$

Here, $\hat{v}_i = \mathbf{D}\beta_i$ and a better reconstruction results in a larger confidence. $\gamma_{colrec}(i)$ measures the chromatic distortion error given by

$$\gamma_{colrec}(i) = e^{-\|I_i - [F_i^{rgb} B_i^{rgb}] \hat{\alpha}_i\|^2}. \quad (3.10)$$

Figure 3.9: Visual effect of alpha refinement using the graph model. (a) Input image, (b) initial estimate, (c) final alpha matte and (d) ground truth.

To enforce smoothness constraint across the image, each pixel is connected to its spatial neighbors in a 3×3 window through the weights W_{lap} , and to its K -nearest neighbors in the feature space through the weights W_{knn} . For pixels i and j in a 3×3 window w_k , the weight W_{lap} is defined as [3]

$$W_{lap}(i, j) = \sum_{k|(i,j) \in w_k} (\delta_{ij} - \frac{1}{9}(1 + (I_i - \mu_k)(\Sigma_k + \frac{\epsilon}{9}I_3)^{-1}(I_j - \mu_k))), \quad (3.11)$$

where δ_{ij} is the Kronecker delta, ϵ is a small regularization constant (10^{-7}), μ_k and Σ_k are the mean and variance of colors in the window and I_3 is the 3×3 identity matrix. K -nearest neighbors at a pixel i in the $RGBxy$ feature space are used to compute the affinity as [16]

$$W_{knn}(i, j) = 1 - \frac{\|X_i - X_j\|}{\sigma}, \quad (3.12)$$

where j is a neighbor of i , and σ is a weighting constant to constrain the values of $W_{knn}(i, j)$ in $[0, 1]$. K is fixed as 12 in our experiments.

The energy function used for solving the matte is [17]

$$E = \lambda \sum_{i \in \mathcal{V}} (\alpha_i - h_i)^2 + \sum_{i=1}^N \left(\sum_{j \in \mathcal{N}_i} W_{ij} (\alpha_i - \alpha_j)^2 \right), \quad (3.13)$$

where N is the total number of nodes (pixels) in the graph model and $\mathcal{V}$ is the set of definite foreground and background pixels. The first term ensures that the final matte is

consistent with the user specified constraints while the second term ensures that neighboring pixels share similar alpha values. h_i enforces the user defined constraints – 0 if i is marked as definite background and 1 if marked as foreground. λ is fixed empirically to be 100. The set $\mathcal{N}_i$ represents the spatial and feature neighbors of a pixel i , along with the two virtual nodes. Hence, the weights W_{ij} combine both the smoothness weights W_{knn} , W_{lap} and the data weights $W_{i,F}$ and $W_{i,B}$ additively at each pixel, i.e., $W_{ij} = W_{knn}(i, j) + W_{lap}(i, j) + W_{i,F} + W_{i,B}$. The energy function is written in matrix form as

$$E = \lambda(\alpha - H)^T \Gamma (\alpha - H) + \alpha^T L^T L \alpha, \quad (3.14)$$

where

$$L_{ij} = \begin{cases} W_{ii}, & \text{if } i = j, \\ -W_{ij}, & \text{if } i \text{ and } j \text{ are neighbors,} \\ 0, & \text{otherwise.} \end{cases}$$

The weight $W_{ii} = \sum_{j \in \mathcal{N}_i} W_{ij}$, Γ is a $N \times N$ diagonal matrix with $\Gamma_{ii} = 1$ if $i \in \mathcal{V}$, else 0. H is a $N \times 1$ vector with user constrained values. Eq. (3.14) is a quadratic function in α and can be solved in closed-form as a sparse linear system. We use the bi-conjugate gradients stabilized solver in MATLAB (bicgstab).

$$\alpha = (L^T L + \lambda \Gamma)^{-1} H. \quad (3.15)$$

Fig. 3.9 demonstrates the effect of our graph-based optimization in refining the initial alpha estimate. The irregularities formed in estimating alpha independently at each pixel (Fig. 3.9(b)) is eliminated, leading to high quality mattes visually similar to the ground truth.

3.3 Experimental Results

The effectiveness of the proposed method is evaluated using the benchmark dataset [2] for image matting. The dataset consists of 35 images encompassing a wide range of transparency of pixels - from opaque to highly transparent. The first 27 images form the training set with publicly available ground truth for reference. The remaining 8 images form the test dataset whose ground truth is hidden from the public and used for benchmark evaluation and ranking [25]. The contribution of different components

Figure 3.10: (a) Doll image and its trimaps (small, large and user-defined), (b) and (c) Images from test dataset.

of the algorithm are shown in the first experiment. The effectiveness of the proposed method is evaluated quantitatively against current state-of-the-art on the test dataset. For each image, three trimaps namely small, large and user-defined are present as shown in Fig. 3.10. The performance is evaluated against the ground truth using three error measures [2]- sum of absolute difference (SAD), mean squared error (MSE) and gradient error. We conduct our experiments on both the training and test images by comparing the results quantitatively as well as qualitatively on three state of the art methods in image matting.

3.3.1 Experiment 1 : Analysis of different components in the proposed method

The proposed algorithm is evaluated quantitatively and qualitatively to show the influence of each step towards the final matte. Fig. 3.11(a) shows 4 images from the dataset and their mattes extracted using different aspects of the proposed algorithm. The second column shows the matte obtained using the universal set as the dictionary. Universal set refers to the dictionary formed from the band of superpixels along the boundary of the unknown region. As can be seen in all the images in Fig. 3.11(b), large inconsistencies are observed due to the presence of high color-correlated samples to the unknown pixel

Figure 3.11: Effect of each step in the proposed method towards visual quality of the matte. (a) Input image, matte extracted using (b) universal set as dictionary, (c) refined dictionary with probabilistic segmentation and (d) graph-based matte optimization.

at distances far from the immediate neighborhood. This is particularly severe in the elephant image in row 4 as the foreground and background regions have similar color distribution. Fig. 3.11(c) shows the improvement obtained in the mattes on using the refined dictionary taking the certainty of the pixel into consideration as explained in section 3.2.3. The graph-based optimization helps to remove the discontinuities that may occur due to the independent estimation of the matte via sparse coding. Row 1 in Fig. 3.11(d) shows the smoothed result as a result of the graph-based optimization on the initial estimate in Fig. 3.11(c).

Table 3.1 shows the quantitative contribution of each step of the proposed algorithm.

The performance is measured using SAD error over the test images, across three types of trimaps for each image - small, large and user. The high SAD error observed in the universal set agrees with the visual results in that a simple increase in the sample set for sparse coding does not result in better estimates. Initial estimate refers to the sparse coded estimate obtained on the refined dictionary, taking into account the certainty of the pixels. There is an improvement over the universal set because the spatial constraint controls the false correlated samples from being part of the sample set. Laplacian refinement is the post-processing step which maintains the smoothness of the matte [82]. This stage shows a marked improvement over the sparse coding stage because in formulating the sparse code optimization, we do not consider the neighboring consistency. Finally, the graph-based refinement is capable of enforcing non-local neighbors in the feature

Table 3.1: Evaluation of effect of each step in our method using sum of absolute difference (SAD) error.

Image	Trimap	Universal set	Initial Estimate	Laplacian refinement	Graph-based refinement
Troll	Small	21.3	15.2	13.7	12.6
	Large	35.8	28.0	25.8	20.5
	User	22.8	15.0	14.8	14.8
Doll	Small	13.7	8.6	6.4	5.7
	Large	16.3	9.9	8.2	7.3
	User	13.9	8.3	6.2	6.4
Donkey	Small	8.3	5.8	4.7	4.5
	Large	10.2	6.5	5.4	5.3
	User	8.9	5.4	4	3.7
Elephant	Small	5.8	2.1	1.8	1.4
	Large	6.6	3.9	3.1	3.3
	User	6.4	2.7	2.3	2.3
Plant	Small	13.9	7.5	5.9	6.3
	Large	17.3	9.2	8	7.9
	User	21.8	14.9	10.6	11.1
Pineapple	Small	12.1	5.4	4.5	4.2
	Large	16.9	9.6	8	8.3
	User	15.7	8.2	5.5	6.4
Plastic bag	Small	54.7	35.8	30.3	28.7
	Large	59.2	36.2	33.1	31.3
	User	48.5	37.6	29.2	27.1
Net	Small	42.7	31.2	27.7	23.6
	Large	52.6	33.8	27.2	25.1
	User	45.4	32.6	29	27.3

Figure 3.12: Error plots showing the qualitative comparison between traditional sampling approaches (E_t) and the proposed sparse coding approach (E_s) on four images.

space to be sharing similar alpha values, giving further improvement in the matte.

Conventional sampling-based methods aim to select an F and B sample that best *represents* the true F and B colors at a pixel. Even if the true F or B colors are not present in the set of candidate samples, alpha is computed using compositing equation by picking the best (F, B) pair from among the candidates. In other words, the best α at a pixel can be estimated from a (F, B) pair which is different from the ground truth F and B , simply because it is not present in the set of candidate samples. However, in the proposed approach, we argue that restricting the selection to a single F and B sample results in more errors. This is illustrated in Fig. 3.12 where the errors in reconstructing a color using paired and unpaired approaches are compared. 40 spatially nearest F and B samples are used as the candidate samples from which the conventional sampling method [8] chooses the best (F, B) pair that minimizes the chromatic distortion error $E_t = \|I_i - \hat{\alpha}_i F_i + (1 - \hat{\alpha})B_i\|$. Our sparse coding approach forms a dictionary of 80 atoms and selects a sparse representation that minimizes the reconstruction error $E_s = \|v_i - \mathbf{D}\beta_i\|$. As can be seen, the sparse coding framework is able to reconstruct the original color at the pixel with the lesser error. In fact, the error is more pronounced for [8] in fuzzy regions like hairy structures.

3.3.2 Experiment 2 : Evaluation on benchmark dataset

The effectiveness of the proposed method against current state-of-the-art is compared qualitatively and quantitatively in this experiment. Fig. 3.13 shows the visual comparison of the proposed matting method with other recent state-of-the-art methods on two images from the training dataset. Both the images are representative of thin hairy structures that are crucial for seamless extraction of foreground for a good matting algorithm. Closed-form matting [3] oversmooths the matte, leading to loss of fine detail in hairy regions (Fig. 3.13(b)), while weighted color and texture matting [12] in Fig. 3.13(c) uses texture feature in addition to color and is still unable to discriminate the finer structures from a complex background region. Comprehensive sampling [11] (Fig. 3.13(d)) utilizes a large sampling set, but results in smooth bands near the foreground-background boundary due to post-processing. The proposed method, shown in Fig. 3.13(e), is able to extract a high

Figure 3.13: Qualitative comparison of proposed method against other state-of-the-art methods on the training dataset. (a) Image, zoomed regions showing the mattes of (b) Closed form [3], (c) Weighted color and texture [12], (d) Comprehensive sampling [11], (e) proposed method and (f) ground truth.

Figure 3.14: Qualitative comparison of proposed method on elephant, pineapple and plant images with top 5 methods at [25]. (a) Input image, (b) zoomed windows, (c) Comprehensive weighted color and texture [83], (d) Comprehensive sampling [11], (e) Iterative transductive matting [33], (f) LNSP matting [17], (g) Cluster-based matting and (h) proposed method. Zoomed in regions show the effectiveness of our method.

quality matte by optimizing the sampling and smoothing terms which is visually closer to the ground truth given in Fig. 3.13(f).

The test dataset contains the 8 most difficult subset of images from the dataset. Qualitative comparisons with state-of-the-art methods at www.alphamattting.com¹ on the elephant, plant and pineapple images are shown in Fig. 3.14. The high overlap in F and B colors and complex texture make these images difficult for matting as shown by the zoomed patches in Fig. 3.14(b). The presence of another plant in the background causes the texture to misconstrue the foreground as background in [11, 83] as seen in Fig. 3.14(c,d) of the plant image. ITM matting [33] wrongly propagates the background into the leaves, the head of the pineapple, and the tail of the elephant in Fig. 3.14(e). Comprehensive sampling [11], which uses a larger sampling set, is able to differentiate the elephant tail but still classifies the leaves and pineapple as mixed pixels in the last two rows of Fig. 3.14(d). The smooth prior in LNSP matting [17] oversmooths the matte leading to a band between the foreground and background at the pineapple head in

¹Results as on July, 2015.

Fig. 3.14(f). Our method is able to reduce such artifacts to extract a visually superior matte in these areas as shown in Fig. 3.14(h).

Table 3.2: Evaluation of matting methods by alpha matting website [25] over set of benchmark images with three trinnaps with respect to SAD. Evaluations have been carried out in July 2015.

Method	Sum of Absolute Differences		Troll		Doll		Donkey		Elephant		Plant		Pineapple		Plastic bag		Net												
	overall rank	avg. small image user rank	avg. large image user rank	(Strongly Transparent) Input	(Strongly Transparent) Input	(Medium Transparent) Input	(Medium Transparent) Input	(Medium Transparent) Input	(Little Transparent) Input	(Little Transparent) Input	(Little Transparent) Input	(Little Transparent) Input	(Highly Transparent) Input	(Highly Transparent) Input	(Highly Transparent) Input	(Highly Transparent) Input													
Proposed method	7.6	8	7.6	7.3	12.6	20.5	14.8	5.7	7.3	6.4	4.5	5.3	3.7	1.4	3.3	2.3	6.3	10	7.9	11.6	4.2	8.3	6.4	28.7	31.3	27.2	23.6	25.1	27.3
LNSP Matting	7.7	5	6.6	11.3	12.2	22.5	19.5	5.7	8.1	8.8	4.6	5.9	3.6	1.5	3.5	3.1	6.2	8.1	10.7	4.1	7.1	6.4	21.5	20.8	16.3	22.5	24.4	27.3	
Iterative Transductive Matting	9.6	10.9	8.6	9.4	13.1	17.2	15.6	5.7	8.6	7.8	5.1	5.8	3.9	1.9	5.8	2.6	6.4	9.8	13.8	5.4	10.2	7.4	25.3	24.9	23.8	20.1	22.7	22.7	
Comprehensive sampling	9.7	8.3	9.3	11.5	11.2	18.5	14.8	6.5	11	9.5	8.9	4.5	4.1	1.7	3.1	2.3	5.4	9.8	13.4	5.5	11.5	7.4	23.9	22.4	22.8	23.8	28.1	28.1	
Comprehensive Weighted Color and Texture	10.1	10.5	10.1	9.6	14.6	16.3	15.7	6.8	10.8	7.9	4.3	5.3	4.1	1.7	3.5	2.2	5.4	9.9	12.8	4.3	7.4	6.2	28.3	28.1	25.4	24.3	30.2	28.7	
SVF Matting	10.5	12.6	9.8	9	18.7	30.7	19.1	6.8	7.7	7.6	4.7	5.4	3.4	1.9	4.7	2.9	5.8	8.7	10.5	4.3	8.4	5.6	21.2	22.1	17.1	25.6	26.1	30.6	
Sparse coded matting	10.6	13	11	7.9	13.7	25.8	14.8	6.4	8.2	6.2	4.7	5.4	4.8	1.8	3.1	2.3	5.9	8.3	10.6	4.5	8.3	5.5	30.3	33.1	27.2	27.4	27.2	29.1	
Weighted Color and Texture Matting	11.7	10	13	12.1	13.1	17.8	15.8	6.5	10	9.4	8.6	4.2	4.7	1.7	6.2	2.7	6.4	11	11.6	4.8	9.1	6.5	23.8	24.8	11	26.5	27.4	28.5	
CCM	11.8	14.1	11.1	10.3	13.8	20.8	16.9	6.4	8.9	8.2	4.7	5.9	3.6	2.5	4.3	3.2	7.1	9	10.6	4.9	8.1	5.7	25.6	27.5	17	25.4	27.5	28.1	
Shared Matting	12.8	12.1	14.9	11.5	10.8	20.5	15.6	7.8	11.6	8.1	4.2	5.3	4.2	2.1	5.8	2.9	5.9	9.2	11.4	5.9	8.8	6.8	34.9	34.9	34.3	23.9	24.4	25.7	
LNCLM matting	13	15.3	12.4	11.5	10.9	11.2	16.7	6.9	10	8.9	7.2	5.6	7.2	2.5	5.1	3.5	7.5	10	10.5	5.6	11.3	7.3	25.2	23.8	19.9	21.2	25	26.4	
Global Sampling Matting	14.4	11.9	16.4	15	10.9	22.7	15.4	6.3	9.5	9.2	4.7	6.4	4.3	2.2	5.6	3.7	6.9	14	12.9	6.3	10	12.9	25.8	27.5	16	22.5	24.4	23.7	
SRLO Matting	15.4	14.6	16.3	15.4	14.7	20	18.6	6.9	10.7	8.9	4.9	5.7	4.7	2.1	6.5	2.8	6.3	10.9	17.5	5.4	11.6	7.2	26.5	29.7	21	21.7	28.5	22.3	
Segmentation-based matting	15.5	16.4	14.4	15.8	12.8	23.5	16.6	6.6	10	8.3	7.3	4.8	6.1	2.1	3.9	3.1	6.7	13	14.7	6.9	8.8	8.2	31.6	35.6	29	24.5	26.7	26.7	
Improved color matting	16.2	16.5	15.4	16.6	14.9	21	24.5	6.7	14	9.5	8.5	4.6	6.1	2.6	5.4	3.4	7.5	20	19.3	6.7	10	8.4	26.1	26.7	15	23.6	26.7	26.7	
Local Spline Regression (LSR)	16.5	18.5	13.5	17.6	12.2	20.1	16.2	6.1	8.8	8.1	5.2	6.2	4.6	2.2	4.9	3.1	9.4	28	18.3	8.2	11.4	10.1	26.4	22.5	20.2	27.2	26.9	39.8	
Global Sampling Matting (filter version)	17.1	15.6	18.5	17.3	12.3	24.3	16.3	7.3	10.2	9.5	5.1	6.4	4.7	2.4	4.7	3.2	5.9	9.5	12.4	6.5	13.1	8.3	26.5	28.3	19	23.6	29.1	23.5	
KNN Matting	17.5	19.9	18	14.8	16.2	26	19.7	8.5	11.2	9.2	4.7	6.7	4.3	3.2	7.7	3.7	9.2	13.9	11.3	6.6	10.4	6.7	18.1	18.6	17.2	27.4	27.2	32.7	
Learning Based Matting	18.3	18.9	16.9	19.3	16.2	22.1	18.7	6.6	7.4	7.4	4.8	6.1	4.3	2.1	3.7	2.8	7.5	14.5	19.5	8.6	14.2	14.6	22.5	24.8	19.5	34.6	38.3	51.2	
LMSPIR	18.9	18.4	19.6	18.8	15.2	22	20	6.7	11.2	8.7	4.8	6.8	4.6	2.1	6.8	3.1	7.9	14.3	20.2	5.4	11.4	7.1	29.7	31.2	29.6	24.4	24.2	28.7	
Shared Matting (Real Time)	19.1	18.6	19.4	19.3	12.4	21.6	16.3	9.5	13.5	9.9	4.4	5.9	4.4	2.5	6.8	3.2	7.1	10.8	12.6	5.4	9.7	7.4	35.3	36.8	35.2	27.6	27.3	33.4	
Closed-Form Matting	19.6	18.6	17.4	22.8	12.7	21.9	17.2	5.9	8.8	8.6	4.7	6.1	4.3	2.2	6.6	4.1	7.3	12.1	19.3	8.3	14.9	13.4	34.2	32.4	27.4	26.5	26.2	26.7	
Sampling Based Image Matting Using NMF	20	14.6	21.3	24	15.7	19.7	17.5	7.7	14.8	13.3	4.1	6.3	5.2	1.5	4.4	7.7	7.1	17	16.1	5.2	11.8	11.3	28.1	29.3	27.3	34.7	34.7	45.7	
Cell-based matting Laplacian	22.9	25	21.5	22.1	15.1	22	21.7	7.6	14.3	9.2	5.7	6.9	5.2	3.1	6.7	4.2	8.6	14.4	14.9	8.5	13.5	11.1	27.7	26.2	27.3	32.4	32.6	37.5	
Large kernel Matting	23.3	25	22.6	22.3	17.2	27	21.8	7.2	15.8	16.8	5.3	6.6	4.6	2.9	8.2	4.2	8.6	12.1	14.7	8.4	13.4	11.2	33.7	31.8	25.1	32.1	32.9	38.4	
Improving Sampling Criterion	23.5	23	24.8	22.8	12.7	21.1	16	7.2	12.1	12.30	5.8	6.8	5.1	3.4	12.1	6.3	6.4	15.7	17.8	10.2	13.1	17.9	33.8	26.7	24	34.7	32.6	37.3	
Robust Matting	24.3	21.6	25	26.4	17.3	28	28.4	10.1	16.9	16.3	5.6	7.3	5.6	2.8	7.3	4.6	7.3	18	18.5	6.8	14.6	11.2	32.7	26.1	12	34.7	32.6	37.3	
SPS matting	26.1	25.9	29.1	23.4	13.4	16	23.4	12.5	16.5	12.4	6.3	9.4	6.5	4.5	12.7	5.4	9.5	16.4	18.6	7.9	18.1	13.6	26.1	30.9	22	26.2	33.5	29.5	
High-res matting	27.2	26.1	28.4	27.1	18.6	30	26.8	8.6	14.1	11.1	5.2	6.2	4.8	2.5	8.3	3.2	7.8	14.5	21.4	8.5	18.3	12.2	35.3	38.1	32	38.7	34.6	36.3	

Table 3.3: Evaluation of matting methods by alpha matting website [25] over set of benchmark images with three trimaps with respect to MSE. Evaluations have been carried out in July 2015.

Mean Squared Error	overall rank	Troll (Strongly Transparent) Input			Doll (Strongly Transparent) Input			Donkey (Medium Transparent) Input			Elephant (Medium Transparent) Input			Plant (Little Transparent) Input			Pineapple (Little Transparent) Input			Plastic bag (Highly Transparent) Input			Net (Highly Transparent) Input					
		avg. small rank	avg. large rank	avg. user rank	small	large	user	small	large	user	small	large	user	small	large	user	small	large	user	small	large	user	small	large	user			
Comprehensive Sampling SVR Matting	6.6	4.5	5.5	9.9	0.56	1.924	1.226	0.21	0.42	0.512	0.35	0.47	0.23	0.1	0.11	0.28	0.49	0.51	0.84	0.21	0.31	0.46	1.45	1.22	0.81	1.6	1.16	1.519
	8.5	11	8.1	6.5	0.511	1.211	0.812	0.38	0.512	0.59	0.33	0.46	0.21	0.113	0.22	0.12	0.518	0.69	0.73	0.36	0.43	0.32	1.713	1.814	1.514	12.16	12.8	1.39
	8.9	9.8	8.5	8.5	0.513	1.516	0.84	0.31	0.31	0.32	0.314	0.410	0.39	0.3	0.23	0.15	0.512	0.52	0.96	0.23	0.68	0.58	2.222	2.424	1.923	18	1.14	13.11
	9.5	8.9	9.3	10.3	0.44	1.212	0.85	0.311	0.616	0.622	0.316	0.33	0.314	0.17	0.24	0.13	0.21	0.56	0.98	0.311	0.916	0.510	1.610	1.56	1.510	111	12.11	13.10
	10.1	13	8.5	8.8	1.126	2.732	1.225	0.423	0.43	0.46	0.317	0.34	0.22	0.16	0.27	0.212	0.34	0.55	0.72	0.24	0.44	0.33	1.44	1.43	0.93	13.18	12.10	15.17
	10.5	10.5	10.9	10.1	0.720	0.83	0.813	0.418	0.720	0.511	0.34	0.32	0.26	0.15	0.28	0.11	0.32	0.714	1.14	0.22	0.42	0.31	2.223	2.120	1.719	110	17.18	14.16
	12.5	11.5	13.5	12.6	0.512	0.95	0.83	0.414	0.617	0.619	0.32	0.31	0.24	0.116	0.525	0.210	0.513	0.817	1.522	0.39	0.56	0.57	1.69	1.69	1.512	12.17	26.28	14.14
	12.5	14.3	13.3	10.1	0.618	2.731	0.811	0.310	0.47	0.31	0.321	0.416	0.317	0.112	0.26	0.14	0.45	0.53	0.95	0.25	0.55	0.34	2.324	2.626	2.124	13.19	13.12	14.15
	13.1	9.4	15.9	14	0.43	2.327	0.915	0.36	0.615	0.620	0.38	0.420	0.37	0.19	0.316	0.215	0.514	0.68	1.2	0.415	1.19	0.718	1.915	2.19	1.718	0.95	13	11.6
	13.6	14.8	12	14.1	0.615	0.96	0.810	0.34	0.718	0.618	0.425	0.49	0.313	0.117	0.527	0.218	0.620	0.612	1.421	0.419	0.514	0.718	1.916	1.68	1.513	0.82	0.92	0.92
Comprehensive Weighted Color and Texture	14	16.1	13	12.9	0.42	0.41	0.916	0.417	0.511	0.43	0.430	0.629	0.318	0.125	0.314	0.221	0.624	0.816	1.115	0.413	0.918	0.511	1.814	1.610	1.116	0.84	1.15	13.13
	14.4	13.6	16.3	13.3	0.55	1.616	0.914	0.525	0.926	0.514	0.36	0.45	0.316	0.118	0.420	0.210	0.47	0.611	0.97	0.412	0.69	0.512	2.929	2.828	2.728	17	13.13	15
	14.8	17	14.8	12.6	0.823	1.8	0.88	0.416	0.514	0.57	0.424	0.523	0.326	0.121	0.317	0.222	0.726	0.922	0.910	0.310	0.57	0.45	1.11	1.11	0.92	12.15	23.26	16.21
	15.3	11.9	17.1	17.1	0.719	0.94	0.86	0.420	0.824	0.828	0.21	0.733	0.38	0.2	0.25	0.533	0.48	0.713	0.61	0.38	0.713	0.717	1.712	2.17	1.921	16.25	24.27	2.23
	16.3	17.1	16.4	15.4	0.825	2.428	1.529	0.39	0.510	0.513	0.37	0.419	0.310	0.119	0.315	0.213	0.725	0.715	0.99	0.414	0.712	0.720	2.18	1.916	1.49	13.20	1.516	15.20
	16.4	17	15.4	16.9	0.514	2.225	1.120	0.312	0.45	0.44	0.313	0.418	0.311	0.18	0.29	0.26	0.621	0.54	1.219	0.418	0.710	0.823	2.727	3.31	3.630	14.23	19.21	16.22
	17.8	16.6	19.3	17.4	0.56	1.313	0.87	0.528	1.27	0.724	0.319	0.526	0.324	0.333	0.832	0.431	0.516	0.921	1.217	0.625	1.121	0.822	1.43	1.57	1.511	0.83	1.17	0.93
	17.9	17.9	18.4	17.5	0.722	1.7	1.17	0.421	0.825	0.723	0.423	0.412	0.325	0.122	0.628	0.217	0.410	0.818	1.523	0.417	1.20	0.615	2.18	2.322	1.716	19	14.15	14
	18.7	19.4	17.3	19.5	0.57	1.514	1.18	0.37	0.513	0.616	0.422	0.422	0.325	0.114	0.313	0.214	1.231	1.329	2.228	0.829	1.122	1.125	2.121	1.711	1.38	15.24	13.14	3.28
	19.4	20.1	18.6	19.4	0.826	1.617	1.327	0.419	0.48	0.510	0.315	0.417	0.315	0.115	0.211	0.29	0.519	1.225	2.28	0.931	1.730	2.32	1.68	1.712	1.4	22.28	27.29	42.32
Cell-based matting Laplacian	19.5	19.5	19.4	19.6	0.824	1.19	1.223	0.415	0.823	0.621	0.320	0.48	0.321	0.120	0.524	0.219	0.623	1.228	2.27	0.416	0.917	0.613	2.325	2.323	2.225	11.13	22.3	12.8
	19.5	17.6	18.9	22.1	0.510	1.823	1.122	0.32	0.46	0.615	0.310	0.415	0.312	0.110	0.312	0.211	1.230	1.420	2.331	0.828	1.629	1.630	3.30	2.727	1.922	13.21	12.9	5.34
	20	18.5	24	17.4	0.59	1.619	0.62	0.729	1.129	0.725	0.429	0.630	0.428	0.231	0.933	0.325	0.46	0.820	1.218	0.521	1.224	0.614	1.711	2.18	1.715	1.12	17.19	13.12
	20.3	18.6	20.9	21.4	0.616	2.428	1.121	0.524	0.721	0.727	0.427	0.525	0.429	0.226	0.419	0.327	0.33	0.610	1.13	0.522	1.225	0.821	2.17	2.221	2.428	12.14	15.17	1.17
	21.9	22	21.9	21.9	0.617	1.722	1.19	0.731	1.230	0.830	0.312	0.411	0.323	0.228	0.629	0.224	0.515	0.819	1.116	0.420	0.815	0.616	3.31	3.29	3.29	13.22	19.20	1.518
	22.6	24.6	21.6	21.5	1.230	1.620	1.730	0.313	0.95	0.45	0.428	0.524	0.327	0.123	0.528	0.223	0.928	1.124	1.420	0.624	1.123	1.24	2.728	2.425	1.717	21.27	2.22	2.828
	23.9	26.1	22.5	23	0.721	1.721	0.89	0.422	0.822	0.617	0.531	0.527	0.433	0.232	0.521	0.329	0.827	1.23	1.524	0.830	1.326	1.227	2.20	1.815	1.820	1.926	23.25	24.25
	24.2	21	24.8	26.9	1.129	2.833	1.731	0.720	1.531	0.931	0.39	0.421	0.320	0.124	0.522	0.326	0.517	1.226	1.925	0.623	1.528	1.228	1.57	1.813	2.627	24.29	23.24	2.927
	24.9	24.6	27.6	25.8	1.322	2.226	2.232	0.526	1.126	0.829	0.318	0.413	0.322	0.111	0.730	0.27	0.622	1.227	2.229	0.826	2.32	1.429	3.232	3.433	4.234	2.630	4.32	2.224

Table 3.4: Evaluation of matting methods by alpha matting website [25] over set of benchmark images with three trimaps with respect to gradient error. Evaluations have been carried out in July 2015.

Gradient error	overall rank	avg. small user		avg. large user		Troll (Strongly Transparent)		Doll (Strongly Transparent)		Donkey (Medium Transparent)		Elephant (Medium Transparent)		Pant (Little Transparent)		Pineapple (Little Transparent)		Plastic bag (Highly Transparent)		Net (Highly Transparent)								
		rank	rank	rank	rank	small input	large input	small input	large input	small input	large input	small input	large input	small input	large input	small input	large input	small input	large input	small input	large input							
Proposed method	7.7	6	6	11.1	0.21	0.34	0.23	0.22	0.23	0.422	0.46	0.44	0.45	0.21	0.51	0.44	1.914	1.96	2.816	1.13	1.86	1.89	1.410	1.520	1.222	0.41	0.54	0.58
LNSP Matting	8.3	6.6	7.1	11.3	0.23	0.36	0.21	0.21	0.21	0.314	0.43	0.511	0.43	0.32	0.53	0.619	1.811	1.97	2.715	1.24	1.64	2.15	1.416	1.38	0.910	0.613	0.817	0.613
Comprehensive sampling	8.6	8.9	7.9	9.1	0.25	0.23	0.26	0.212	0.314	0.421	0.45	0.43	0.44	0.36	0.54	0.43	1.12	1.71	2.32	1.513	2.413	1.912	1.38	1.39	0.96	0.722	0.716	0.719
Cluster-based Sampling Matting	10.3	10.5	10.1	10.1	0.21	0.22	0.22	0.216	0.315	0.316	0.420	0.512	0.420	0.31	0.712	0.58	1.810	2.10	2.713	1.25	2.111	1.88	1.520	1.413	0.84	0.59	0.66	0.59
CCM	10.3	12.6	9.5	8.8	0.26	0.38	0.25	0.28	0.36	0.36	0.415	0.613	0.413	0.416	0.66	0.56	2.119	2.315	2.58	1.37	1.75	1.64	1.35	1.35	0.83	0.723	0.818	0.723
SVR Matting	10.3	12.5	10.9	7.5	0.325	0.426	0.316	0.322	0.39	0.39	0.48	0.510	0.42	0.49	0.69	0.57	1.57	2.112	2.57	1.26	1.63	1.53	1.33	1.22	0.82	0.719	0.713	0.614
Sparse coded matting	10.6	12.1	9.4	10.3	0.324	0.421	0.324	0.317	0.310	0.313	0.418	0.56	0.415	0.37	0.52	0.41	1.46	1.74	2.34	1.12	1.62	1.32	1.519	1.627	1.120	0.54	0.53	0.43
Segmentation-based matting	11.1	14	9.5	9.9	0.27	0.35	0.27	0.26	0.22	0.32	0.411	0.516	0.48	0.47	0.65	0.58	2.120	1.73	2.59	1.919	2.212	2.520	2.34	1.729	1.223	0.55	0.52	0.41
Global Sampling Matting	11.4	11.5	12	10.6	0.22	0.312	0.24	0.24	0.34	0.35	0.48	0.514	0.47	0.415	0.816	0.620	2.121	2.114	2.712	2.21	2.822	2.621	1.412	1.36	0.99	0.58	0.69	0.57
Shared Matting	11.4	11.5	12.1	10.6	0.24	0.39	0.28	0.210	0.38	0.33	0.410	0.517	0.412	0.521	0.714	0.516	2.525	2.521	2.714	1.918	2.517	2.419	1.411	1.411	0.95	0.42	0.41	0.42
Improved color matting	11.7	13.1	12	10	0.28	0.37	0.29	0.210	0.38	0.33	0.410	0.517	0.412	0.521	0.714	0.516	2.525	2.521	2.714	1.918	2.517	2.419	1.411	1.411	0.95	0.42	0.41	0.42
Comprehensive Weighted Color and Texture	13.4	13.1	15.1	12	0.430	0.429	0.531	0.429	0.523	0.524	0.32	0.41	0.41	0.35	0.67	0.42	1.11	1.85	2.45	1	1.41	1.21	1.731	1.831	1.227	0.56	0.924	0.45
Global Sampling Matting (filter version)	15.9	14.8	16.5	17.4	0.28	0.427	0.317	0.325	0.419	0.423	0.528	0.520	0.527	0.527	0.816	0.728	1.34	1.72	2.21	2.20	2.720	2.217	1.34	1.24	1.224	0.43	0.714	0.44
Weighted Color and Texture Matting	16.3	15.6	15.8	17.6	0.214	0.313	0.210	0.324	0.416	0.326	0.47	0.42	0.48	0.522	1.125	0.727	1.915	2.417	3.325	1.512	2.9	1.911	1.415	1.414	1.118	0.616	0.926	0.616
LNCM matting	16.5	18.1	16.3	15	0.210	0.21	0.313	0.313	0.322	0.416	0.312	0.528	0.520	0.525	0.520	0.711	0.617	2.122	2.623	2.917	1.511	2.414	2.13	1.622	1.622	0.98	0.610	0.615
Sampling Based Image Matting Using NMF	17.7	10.9	18.5	23.6	0.316	0.420	0.428	0.213	0.416	0.419	0.31	0.529	0.418	0.34	0.920	1.626	1.916	2.416	2.510	1.615	2.415	2.824	1.37	1.37	1.320	0.615	0.925	0.724
SPS matting	17.8	17.4	19.4	16.5	0.312	0.318	0.312	0.430	0.730	0.526	0.522	0.522	0.526	0.529	1.430	0.724	1.35	2.9	3.26	2.123	2.721	2.116	1	1.22	0.81	0.617	0.823	0.722
Iterative Transductive Matting	17.8	19	16.5	17.8	0.319	0.422	0.314	0.214	0.626	0.528	0.520	0.57	0.421	0.38	1.124	0.515	2.16	2.113	3.21	1.817	2.516	2.316	1.528	1.410	1.14	0.716	0.612	0.511
SRLO Matting	18.6	19.4	20.1	16.3	0.328	0.424	0.326	0.328	0.628	0.520	0.524	0.56	0.410	0.417	1.226	0.510	1.23	1.98	2.811	1.49	2.519	1.86	1.625	1.933	1.16	0.721	0.715	0.721
Learning Based Matting	19	17.8	19.4	19.9	0.322	0.315	0.322	0.27	0.35	0.34	0.417	0.523	0.419	0.413	0.61	0.514	1.913	2.520	3.224	3.122	3.631	3.932	1.49	1.624	1.13	0.929	1.129	1.231
Local Spline Regression (LSR)	19.3	20.6	19.1	18	0.213	0.310	0.311	0.25	0.312	0.311	0.416	0.515	0.411	0.414	0.713	0.513	3.422	3.529	4.529	2.928	3.24	3.226	1.629	1.728	1.17	0.828	0.822	0.926
LNMPR	19.3	18.5	19.5	19.8	0.327	0.425	0.325	0.321	0.729	0.527	0.421	0.59	0.417	0.411	1.227	0.512	1.56	2.519	3.223	1.614	2.416	1.910	1.626	1.621	1.226	0.720	0.610	0.616
Closed-Form Matting	19.4	19.5	18.6	20.1	0.317	0.314	0.319	0.23	0.37	0.37	0.413	0.521	0.414	0.416	0.610	0.618	3.331	3.630	4.630	2.931	3.529	3.731	1.521	1.518	0.912	0.724	0.820	1.230
Improving Sampling Criterion	19.5	19.9	19.4	19.1	0.316	0.419	0.320	0.326	0.421	0.416	0.525	0.525	0.528	0.533	1.431	0.930	2.223	2.724	2.916	2.625	3.226	2.522	1.12	1.41	0.97	0.57	0.68	0.510
KNN Matting	20	22.5	19.3	18.4	0.323	0.316	0.315	0.318	0.311	0.315	0.533	0.628	0.633	0.524	1.21	0.521	2.828	2.825	3.20	1.510	1.98	1.65	1.516	1.515	0.911	0.826	1.230	0.927
Robust Matting	20.8	18.4	20.8	23.3	0.320	0.423	0.323	0.531	0.931	0.31	0.414	0.516	0.523	0.523	0.817	0.723	1.812	2.622	3.122	2.22	3.125	2.723	1.413	1.625	1.229	0.612	0.67	0.512
Large Kernel Matting	21.6	21.9	21.6	21.4	0.321	0.317	0.321	0.28	0.313	0.21	0.523	0.627	0.524	0.525	1.326	0.725	2.929	3.328	3.726	2.627	3.327	3.430	1.414	1.412	1.119	0.827	0.821	0.825
High-res matting	21.9	19.8	25.1	20.9	0.215	0.311	0.318	0.216	0.524	0.722	0.412	0.524	0.416	0.416	1.123	0.622	2.17	2.516	2.919	1.716	2.110	2.14	1.627	1.623	1.321	0.825	0.819	0.720
Shared Matting (Real Time)	23.5	24.1	22.5	23.8	0.329	0.531	0.430	0.623	0.932	0.420	0.412	0.524	0.422	0.626	1.123	0.622	2.17	2.516	2.919	1.716	2.110	2.14	1.627	1.623	1.321	0.825	0.819	0.720
Cell-based matting Laplacian	25.7	26.6	25.3	25.1	0.326	0.426	0.327	0.320	0.625	0.317	0.531	0.631	0.631	0.526	0.919	0.728	2.17	3.227	3.172	2.929	3.428	3.228	1.524	1.519	1.15	0.930	1.27	0.928
Transitive Weights	28.9	27.5	29.6	29.4	0.522	0.533	0.522	0.211	0.417	0.36	0.529	0.622	0.622	0.922	1.524	1.31	4.537	5.635	7.27	4.537	5.637	5.337	1.33	1.516	1.225	1.524	1.624	1.825
Random Walk Matting	29.8	30.5	29.3	29.5	0.431	0.530	0.429	0.319	0.422	0.310	0.527	0.626	0.530	0.730	1.22	0.829	4.836	6.437	6.736	4.436	5.36	4.736	1.730	1.626	1.322	1.735	1.925	1.724
Iterative BP Matting	31.5	30.1	31.9	32.5	0.533	0.532	0.633	0.733	1.234	1.234	0.634	0.833	0.634	0.731	1.533	1.132	2.726	3.631	4.932	2.930	3.630	3.227	1.623	1.830	2.936	1.131	1.332	1.232

Figure 3.15: Additional results of proposed method in compositing applications. (a) Input image and trimap labeled in red and blue, (b) mattes extracted by our algorithm, (c) composited image and (d) zoomed region on composite image showing seamless extraction.

Table 3.2 shows the relative ranking of all matting methods submitted at the benchmark website for alpha matting with respect to SAD error. The subscript next to each SAD value shows the rank for the particular image and trimap. The mattes for Doll, Elephant, Plant and Pineapple images are ranked the best with other images performing good as well. The proposed method has an average rank of 8, 7.6 and 7.3 for the small, large and user trimaps respectively, leading to an overall rank of 7.6 which is the best among all the methods reported at [25]. The performance with respect to MSE and gradient error are shown in Table 3.3 and Table 3.4 respectively. The proposed method is ranked 1st among all methods in gradient error with an overall rank of 7.7. This is attributed to the graph-based refinement that considers both feature-space and spatial neighbors in optimizing the matte.

The estimated mattes can be used to seamlessly extract the foreground and paste it onto a new background resulting in a composite image. Fig. 3.15 shows a few examples of images and their composites generated using the proposed method. Since sparse coding may result in more than one foreground and background sample being blended to obtain the color at an unknown pixel, the foreground colors for compositing are obtained by blending the multiple foreground RGB tuples with their corresponding sparse codes as the blending factor. i.e., $F^{rgb} = \alpha_1 F_1^{rgb} + \alpha_2 F_2^{rgb} \dots \alpha_m F_m^{rgb}$. The original image and their trimaps are indicated in the first column with red and blue indicating the foreground and background boundaries respectively. The mattes extracted using our sparse-coded algorithm is shown in Fig. 3.15(b). The foreground composited onto a different background is shown in Fig. 3.15 (c) and (d). The zoomed regions show the effectiveness of seamlessly compositing the foreground object into a different scene.

3.3.3 Experiment 3 : Comparison with other feature coding methods

Many coding schemes have been proposed earlier such as, hard-assignment coding (HC) [84], localized soft-assignment coding (LSC) [85], sparse coding (SC) [75] and locality-constrained coding (LLC) [86]. The proposed method that uses sparse coding is compared with other feature coding algorithms to demonstrate their applicability to the problem of alpha

Figure 3.16: Visual comparison of feature coding methods for alpha matting. (a) Input image and zoomed window, (b) LLC, (c) LSC, (d) proposed method and (e) ground truth.

Table 3.5: Quantitative evaluation of the proposed sparse coding framework with other feature coding algorithms.

Feature coding method	MSE		SAD	
	Small trimap	Large trimap	Small trimap	Large trimap
LLC	0.0241	0.0207	5.968	7.28
LSC	0.0176	0.0271	5.547	10.62
Proposed method	0.0118	0.0155	4.071	5.82

matting. In particular, LSC [85] and LLC [86] are chosen and compared both qualitatively and quantitatively on the images. The parameters are kept unchanged for this comparison. However, since LLC does not ensure that the returned feature codes are positive, the ratio of sum of squares of the l_2 -norm of the codes are used to arrive at its alpha value. Fig. 3.16 shows the visual quality of the mattes obtained using the 3 coding strategies. LLC biases the matte towards binary values while LSC produces more visually pleasing mattes. The use of specific constraints that steer the codes to alpha values in the proposed method produces the best visual results which agrees quantitatively as demonstrated by the error values presented in Table 3.5.

Table 3.6: Comparison of running time of the proposed method with recent sampling-based approaches.

Method	Average time taken (secs)								Avg. time per image (secs)
	Donkey	Doll	Elephant	Pineapple	Troll	Plant	Net	Plastic Bag	
Comprehensive sampling [11]	211.8	376.2	217.3	232.5	395.4	484.1	117.2	586.5	459.3
Comprehensive weighted color and texture [83]	156.8	205.6	167.8	195.8	326.8	253.9	475.2	296.6	259.8
Proposed method	79.9	146.8	102.2	121.9	339.8	237.9	639.2	307.7	246.9

3.3.4 Experiment 4: Runtime performance

Table 3.6 compares the running time of the proposed method with recent sampling-based approaches. MATLAB implementations were evaluated on a PC running Intel Xeon 3.2 GHz processor. Sparse coding is able to generate faster estimates from the same set of F and B samples as opposed to an exhaustive pair search followed by conventional sampling methods [11, 83] leading to better runtime performance. For the donkey image, the pre-processing and certainty estimation takes 32 seconds, the sparse coded matte estimation is completed in just 41 seconds followed by the graph-based optimization in 9 seconds.

3.4 Conclusions

In this chapter, we have presented an image matting algorithm that improves on current sampling-based approaches by moving away from the pairwise sample assumption followed by the compositing equation. In regions of high texture and overlapping color distribution, the best F and B sample does not represent the true F and B colors at the pixel. The use of multiple F and B samples to describe the observed color at a pixel, as opposed to (F, B) pairs leads to better matte estimation in textured regions as shown by the residual error plots. Sparse coding is chosen as the tool for reconstruction of the color features as it produces the best reconstruction while ensuring the number of bases chosen to a minimum. The use of specific constraints in solving the sparse coding problem leads to the codes directly providing the alpha values. A graph-based optimization step is proposed that ensures the smoothness in the final matte by combining the spatial and

feature-space neighbors in the optimization. The proposed method performs best among the current state-of-the-art in both SAD and gradient error measures when evaluated against all other matting methods at benchmark website.

In the next chapter, a video matting algorithm that builds up on the sparse coding principle for multi-frame temporally coherent matte extraction is presented.

Chapter 4

Multi-frame video matting with temporal trimap propagation

4.1 Introduction

Image matting dealt with extracting the foreground object from a single image requiring the user to define a trimap or mark scribbles on a single image. Sampling-based algorithms require a well-defined trimap as a scribble-based input cannot capture the foreground and background samples accurately for matting. The accuracy of a trimap largely determines the accuracy of a matte. When extended to videos, it is impractical to expect the user to mark a trimap on each and every frame of a long video sequence. Moreover, applying image matting algorithms to each frame with the availability of user-defined trimaps still does not guarantee *temporal coherency* in the estimated mattes of the video sequence. The additional challenge comes from motion of the foreground and background due to which the trimap in one frame may not be valid in a subsequent frame. The human visual system is highly sensitive to temporal jitter or inconsistencies in the video sequences. The phenomenon is illustrated in Fig. 4.1 where four consecutive frames of a sequence and its estimated mattes obtained by independently applying our image matting algorithm are shown. The yellow windows that are zoomed and shown in the third row highlight the temporal inconsistencies in the subsequent frames.

Sampling-based approaches in image matting collect samples marked as foreground and background from within the image. Video presents the opportunity to use samples from nearby frames to handle the problem of fast motion as demonstrated by [30]. However, solving for the alpha value at a pixel is still performed independently with respect

Figure 4.1: Illustration of temporal inconsistencies in video matting.

to a neighboring pixel in the video. One of the fundamental requirements of temporal coherency is consistency between similar pixels across the temporal domain.

In this chapter, a new video matting algorithm is presented that handles the challenges reported above in order to extract a temporally coherent video matte. First, a temporal trimap propagation method is introduced in section 4.2 that alleviates the task of specifying trimaps manually in every frame. An adaptive strategy for trimap propagation between key-frames is proposed that builds on 2-D shape blending and motion flow to ensure temporally coherent trimap generation. Second, a multi-frame video matting step is introduced in section 4.3 that simultaneously solves the mattes of multiple frames together. To achieve temporal coherence, the alpha at a pixel needs to be consistent not just with neighboring pixels in the same frame but also across similar pixels from nearby frames. To facilitate this, dictionaries are built using pixel samples from multiple frames for sparse coding. A weighted graph model is built across the block of frames and solved in closed-form to generate the video mattes. Quantitative and qualitative comparisons on multiple sequences from an exclusive video matting dataset are shown to demonstrate the effectiveness of the proposed video matting algorithm.

Figure 4.2: Proposed trimap propagation using foreground object contours. (a) Key-frame image with contour in yellow and its (b) trimap, (c) trimap generated by [58] seven frames ahead and its (d) alpha matte, (e) our propagated contour on the same frame, (f) our generated trimap used to extract the (g) alpha matte.

4.2 Temporal trimap propagation using motion-assisted shape blending

Temporal coherence for the video matte varies with even slight variations in the trimap generated. The task of manually drawing a trimap for each frame is tedious and impractical for long sequences.

Existing approaches use optical flow [27, 28] or learn statistical models of the foreground and background [58] to propagate the trimap to the *in-between* frames. Optical flow is often erroneous in videos with low contrast and fast motion, especially near the object boundaries which are crucial for matting. The errors introduced during motion flow progressively accumulate towards the later frames, requiring constant manual intervention. Learning color-based statistical models for foreground and background suffers in videos where there is significant overlap in the F and B colors. We propose a novel trimap propagation framework by first extracting boundary contours of a foreground object from key-frames several frames apart. Shape-interpolated contours are combined with the motion information to generate accurate edge-aware foreground contours for the intermediate frames. Algorithms for shape interpolation between curves, in gen-

(a)

(b)

Figure 4.3: (a) Proposed framework for temporal trimap propagation, (b) Illustration of the generation of in-between contours (outlined region in (a)) using cat sequence.

eral, ignore the underlying motion between the curves [87]. Trimap maps are generated from the contours by adaptive parameterization of the trimap thickness, which ensures that the estimated mattes are temporally coherent. The key-frame curve and its trimap in Fig. 4.2(a, b) is propagated to successive frames (Fig. 4.2(e, f)) after which a matting algorithm extracts the matte as shown in Fig. 4.2(g). [58] generates a poor trimap (Fig. 4.2(c)) that produces artifacts in its matte (Fig. 4.2(d)) due to overlapping color distributions of the cat and the wire.

4.2.1 System Overview

The framework of our proposed trimap propagation step is shown in Fig. 4.3(a). An input video sequence can have multiple key-frames where user interaction is required to specify the trimap. Once the key-frames and foreground contours have been selected, in-between contours are generated by combining shape blending and motion flow. For brevity, a sub-sequence with key-frames at either end of the sequence is shown in Fig. 4.3(b) to illustrate the process of in-between contour generation. Given a video sequence with foreground contours defined on key-frames t_a and t_b , the proposed strategy generates contours for M in-between frames $t : t_a < t < t_b$. Key-frame curves C_{t_a} and C_{t_b} , which correspond to foreground contours, can be generated by either manual interaction using a tool like Intelligent scissors [88], or by thresholding the matte obtained using their trimaps. A curve in frame t is denoted as a set of N discrete vertices $\{C_t(s_i) : i = 1, 2 \dots N\}$. Shape-interpolated curves between the key-frames are indicated in blue in Fig. 4.3(b), and are generated as explained in section 4.2.2. The motion information obtained using optical flow is then combined to the shape-interpolated curves of the two nearest frames from the key-frames, t_1 and t_M to obtain the in-between curves $C_{t_1}(s_i)$ and $C_{t_M}(s_i)$ as shown in second row of Fig. 4.3(b). Using these curves as the key-frames, the same procedure of combining shape interpolation and motion information is repeated for the $M - 2$ frames until all the in-between curves are generated. Once the in-between curves are generated, the trimaps are generated from them using the strategy detailed in section 4.2.4.

4.2.2 Shape blending between key-frames

Shape blending refers to the smooth transformation of one shape to another. Given 2 polygonal 2-D shapes with the equal number of vertices, shape blending or in-betweening is performed by a linear interpolation between the corresponding vertices. The vertex correspondence between the key-frames are established by following the approach of Sederberg and Greenwood [87] which is based on a physical model of a wire undergoing deformation. The key-frame curve C_{t_a} is modeled as a 'wire' that undergoes stretching and bending to reach its final shape in C_{t_b} . Correspondence is established by selecting the path that utilizes the least work in stretching and bending the curve C_{t_a} to C_{t_b} .

Figure 4.4: Shape blending between key-frames marked in yellow. The shape interpolated curves are shown in blue.

Stretching work is computed for each line segment between adjacent pair of vertices s_i and s_{i+1} as [87]

$$W_{stretch} = \frac{k_s |L_1 - L_0|^{e_s}}{(1 - c_s) \min(L_0, L_1) + c_s \max(L_0, L_1)}, \quad (4.1)$$

where L_0 and L_1 are the length of the segment before and after stretching respectively, k_s, e_s and c_s are constants fixed empirically. Eq. (4.1) favors stretches in the curve that do not significantly change the length of the line segment between adjacent vertices. Similarly, bending work is computed for each adjacent pair of line segments on the curve denoting the work spent in bending the angle between them as [87]

$$W_{bend} = k_b (\Delta\theta + m_b \Delta\theta^*)^{e_b}, \quad (4.2)$$

where $\Delta\theta$ measures the difference in angle between the key-frame line segments, $\Delta\theta^*$ measures deviation from monotonicity, and k_b, m_b and e_b are constants that encode the bending stiffness. For determining the correspondence, the constants k_s, e_s, c_s, k_b, m_b and e_b are fixed as 1, 2, 0.5, 0.2, 10 and 2 respectively as in [87]. It is to be noted that [87] requires the initial correspondence points to be known, whereas the proposed method automatically finds out the best vertex correspondence by employing a greedy search across all vertex pair combinations to pick the one with the least work. Once the key-frame correspondence has been established, the shape blended curves are generated by linear interpolation between the N vertices and denoted $C_{t_j}^{Shp}(s_i) : j = 1, 2 \dots M, i = 1, 2 \dots N$. Fig. 4.4 illustrates the shape blended curves in blue on the *Archaeology* sequence between two key-frame curves marked in yellow, 18 frames apart. As can be seen, shape-blending is able to accurately generate in-betweens in low-contrast regions of uniform motion, but fails to consider the edge information and sudden motion of the object and camera.

Figure 4.5: Illustration of the proposed novel shape and motion combination strategy at a curve point s_i . The difference between $\theta^{Shp}(s_i)$ and $\theta^{Mot}(s_i)$ indicates the discord between shape-interpolation and motion flow.

4.2.3 Combining motion with shape interpolation

Optical flow [54, 55] has been used in computer vision research to describe the motion between two frames as a flow field. It has become prevalent in trimap propagation by warping the user-marked trimap to the unmarked frames using the flow field. However, it is unreliable and erroneous in areas of low contrast or occlusions limiting its use to generate fewer in-betweens. The errors introduced at the start gets progressively compounded for the future frames. Shape blending cannot handle instances where the foreground object has non-uniform motion or when the camera is moving along with the object. As shown in Fig. 4.4, it works better in low contrast or highly textured regions which are likely areas for optical flow to fail. The proposed method utilizes this key idea that shape blending and optical flow are complementary to one another.

A novel strategy using angular distance to combine the motion information obtained using optical flow, with the shape interpolation framework is proposed and illustrated in Fig. 4.5. Using the optical flow field generated between frames t_{j-1} and t_j [54], a curve point s_i on frame t_{j-1} is warped to the frame t_j and denoted as the red point $C_{t_j}^{Mot}(s_i)$ in Fig. 4.5. The green point ($C_{t_j}^{Shp}(s_i)$) indicates the shape blended curve point in frame t_j obtained as explained above in section 4.2.2. The angle formed between the line segment

$C_{t_{j-1}}(s_i)C_{t_j}^{Shp}(s_i)$ and the horizontal positive X-direction is denoted $\theta^{Shp}(s_i)$. Similarly, $\theta^{Mot}(s_i)$ represents the same for the motion warped point on t_j . As seen from Fig 4.5, the difference between these angles is an indication of the discord in curve motion which is not captured using shape blending alone. This is attributed to camera movement as well as abrupt, non-uniform foreground motion. If the shape-blended and motion-warped point are near to each other (small angle difference), the mean position of the two is chosen to be the final curve point. However, if the angle criteria determines a discord between the two, further analysis is required as explained below.

As mentioned earlier, since optical flow is unreliable and erroneous in areas of low contrast or occlusions, directly applying this strategy is not beneficial. A normalized motion confidence map O_t is generated for a frame t as $e^{-|I_t - I_{t+1}^w|}$ where I_{t+1}^w is the motion-warped image from frame $t + 1$ to t . In areas of high motion confidence ($O_t > 0.9$), the motion-warped point is chosen to obtain the in-between curve. Low confidence indicates regions of low contrast, occlusions and cluttered background, making shape-interpolation more reliable than motion. Since the propagated curves are indicative of the foreground boundary edges, a gradient-based refinement fits the shape-interpolated curve to edges as detailed in section 4.2.4. The final in-between curve at frame t_j is denoted $C_{t_j}(s_i)$. Fig. 4.6 illustrates the final in-between curves generated in white on the foreground contours of the cat sequence. The uneven motion of the cat's head is lost during the shape interpolation step, but is captured by the motion information to describe the contours accurately.

Figure 4.6: Combining motion information with shape interpolation (blue) to obtain the final in-between curves (white).

Algorithm 1 Motion-assisted shape blending

```
1: for each vertex  $s_i$  in key-frames  $t_a, t_b$  do
2:   procedure SHAPE BLENDING( $(C_{t_a}, C_{t_b})$ )
3:      $work = Stretch\_work + bend\_work = 0$ 
4:     for each starting correspondence in  $t_a$  and  $t_b$  do
5:       procedure COMPUTE BEST WORK
6:          $best\_work = \min_j(total\_work_j)$ 
7:       end procedure
8:     end for
9:   end procedure
10:  procedure COMPUTE OPTICAL FLOW
11:     $O_t = \text{motion confidence}$ 
12:  end procedure
13:  if  $O_t > \text{threshold}$  then
14:    Combine motion with shape interpolation
15:  else Use shape interpolation alone
16:  end if
17:  Gradient-based refinement
18: end for
19: Return  $C_{t_j} : t_a < t_{1,2,...j} < t_b$ 
```

4.2.4 Gradient-based refinement and adaptive trimap generation

A local gradient-based refinement fits the curves to foreground boundaries in areas of low motion confidence by using edge features like gradient magnitude, gradient direction and laplacian zero-crossing. Laplacian zero-crossing represents areas of high gradient change, obtained by convolving the image with the laplacian kernel. At each sample point s_i on the shape-interpolated curve, a 10×10 neighborhood is considered and the pixel having the largest gradient magnitude and zero-crossing (indicative of foreground boundary) is chosen as the refined point. In case of multiple candidate points, the gradient direction constraint chooses the one with least change in gradient orientation from the initial orientation in the key-frame curves. The thickness of the trimap definition at the foreground boundary is crucial for determining the matte quality. For the trimap highlighted in yellow in Fig. 4.7(a), the hairy regions at the bottom require a wider unknown region to capture the mixed pixels whereas regions of sharp transition require a finely drawn trimap. A uniform dilation and erosion of the contours to generate the

Figure 4.7: Illustration of the adaptive trimap radius propagation. (a) Key-frame image with trimap, (b) unknown region with foreground, background and object contours, (c) zoomed region depicting radius values for contour points, (d) propagated trimap on $t + 10^{th}$ frame and its (e) extracted matte.

trimap would result in low accuracy if the thickness is insufficient in the hairy regions, or too wide for sharp boundaries. Moreover, trivially making the trimap too coarse would result in inaccurate samples being selected for sampling-based algorithms. Following [28], the trimap is parameterized by specifying a local outer radius $d_{s_i}^B$ and a inner radius $d_{s_i}^F$ for each contour point s_i on the key-frames as illustrated in Fig. 4.7(c). The foreground and background radii are propagated to the corresponding contour points of the in-between curves using linear interpolation and rasterized to produce the trimaps for all the in-between curves.

4.3 Multi-frame video matting

Once the trimaps have been generated for all the in-between frames, the proposed video matting algorithm takes the video frames and their respective trimaps as input and generates the video mattes for the sequence. The proposed algorithm extends the sparse coding framework introduced for images in chapter 3 to simultaneously extract temporally coherent mattes across multiple frames. To achieve temporal coherence, the alpha at a

Figure 4.8: Illustration of sampling strategy for dictionary formation across multiple frames in the *Lemur* sequence. The unknown pixel in yellow considers foreground (red) and background (blue) samples from neighboring frames to avoid missing samples due to the dynamic nature of background.

pixel needs to be consistent not just with neighboring pixels in the same frame, but also across similar pixels from nearby frames [28,30]. This is achieved by first constructing the dictionary comprising F and B samples from multiple frames followed by sparse coding to generate the matte. The initial estimate is further refined using an optimization step that builds a graph model across M consecutive frames and solves for the matte of M frames simultaneously.

For a given pixel i at frame t , the dictionary for sparse coding is formed by sampling across the multi-frame block. Fig. 4.8 illustrates the sampling strategy for the proposed multi-frame algorithm. The number of samples chosen from the current frame is the highest, with the number decreasing with increasing temporal distance as shown by the zoomed regions in the second row. By utilizing the samples from nearby frames, a comprehensive sample set is ensured for forming the dictionary.

The dictionary for a pixel in frame t of the proposed multi-frame sparse coding is formed by concatenating the F and B samples from across the M -frame block as

$$\mathbf{D}_M^t = [F_1^{t-2} F_2^{t-2} \dots F_n^{t-2} B_1^{t-2} B_2^{t-2} \dots B_n^{t-2}], \quad (4.3)$$

where n indicates the total number of samples and the superscript indicates the frame number. In order to assign higher priority to the intra-frame samples, weighted sparse

Figure 4.9: Multi-frame graph model for temporally coherent video matting. (a) K -nearest neighbors of the pixel (yellow) in the second frame are encoded across multiple frames, (b) Each pixel (node) is connected to its feature space neighbors as well as the virtual foreground and background nodes in the graph. The blue, red, and white nodes indicate the definite background, definite foreground, and unknown pixels, respectively.

coding is used to arrive at the initial alpha estimate. The weighting function is defined as $W_{sp}^t(i) = \frac{1}{\gamma^{|T-t|} \times e^{-\|i-j\|}}$ where j is the index of the definite foreground/background sample and T is the current frame under consideration. The sparse code for the pixel i in frame t is defined as [77, 78]

$$\begin{aligned} \beta_i^t &= \operatorname{argmin} \|v_i^t - \mathbf{D}_M^t \beta_i^t\| \\ s.t. \quad &\|diag(W_{sp}^t(i))\beta_i^t\|_1 \leq 1, \quad \beta_i^t \geq 0, \end{aligned} \quad (4.4)$$

where v_i^t is the 6-D feature vector $[R \ G \ B \ L \ a \ b]^T$. The first constraint denotes the weighted sparse coding criteria wherein higher penalty are given to inter-frame atoms of the dictionary through W_{sp} . The initial alpha estimate is obtained from the sparse codes as in eq. (3.6).

Fig. 4.9 illustrates the multi-frame graph model for videos. Similar to the image matting framework, the white nodes are the unknown pixels for which the alpha is to be estimated. However, each pixel is connected to its K -nearest neighbors in the feature space across frames for ensuring temporal smoothness as shown by the green lines in Fig. 4.9(b). It is to be noted that the video matting algorithm does not use spatial

Laplacian smoothness term in the graph formulation. As mentioned in section 2.4, the challenge in extending the matting Laplacian to the 3-D space arises from the oversmoothness caused by utilizing the spatio-temporal neighbors in defining the affinity. However, feature-space neighbors do not violate the assumption that similar features should share similar alpha values.

The confidence value for the initial estimate is identical to that used in eq. (3.7). The initial estimate and its confidence value forms the data term for the video model. To enforce temporal coherency across frames, each pixel is connected to its K -nearest neighbors in the feature space through the weights W_{knn} . The nearest neighbors are found using the FLANN algorithm [89]. Fig. 4.9(a) illustrates the feature neighbors across four frames for the pixel marked yellow. We maintain the simple feature used for the image algorithm and do not utilize motion flow which is time consuming. The same energy function as eq. (3.14) is used to determine the video matte across M frames, except that now the affinity matrix L is sparse and symmetric and of size $(M \cdot n + 2) \times (M \cdot n + 2)$, where n is the number of pixels in a frame. We fix M as 4 in our experiments. A higher number of frames can be used to form the multi-frame block, but the memory requirement is prohibitive due to the large size of the affinity matrix. The resulting cost function can be efficiently solved as a sparse linear system of equations using eq. (3.15) in the order of a few seconds for each frame. It is to be noted that the mattes of M frames are solved simultaneously using the proposed approach.

4.4 Experimental Results

The effectiveness of the proposed video matting algorithm is evaluated using a dataset provided by Adobe [30] built exclusively for video matting incorporating the challenges faced in video matting such as low contrast, fast motion and large areas of transparency. Five of the sequences have ground truth alpha available which was obtained by shooting the objects in front of a green screen, followed by extraction of alpha mattes and foreground colors. The extracted objects were then composited onto a new background to obtain the final sequences. Fig. 4.10 shows example frames from the sequences and their ground truth alpha mattes.

Figure 4.10: Video sequences from the dataset with ground truth mattes.

Unlike image matting, even in the presence of ground truth data, video matting methods do not have a consensus in measuring the perceived quality of video matting. MSE is useful in measuring the overall accuracy, but does not correlate to temporal coherency that is crucial for eliminating jitter and visually jarring artifacts in any video matting application. Being a subjective measure, it is not quite obvious as to what error metric can be used. Two measures for temporal coherency are used by existing methods to evaluate matte quality. The first is the ratio of temporal derivatives of the matte and image frames introduced by Lee *et al.* [53] and later used by recent methods [65, 67, 70]. It is a measure of the temporal jitter error (TJE) and is defined for the i^{th} pixel as

$$TJE = \frac{|\alpha_i(t+1) - \alpha_i(t)|}{|I_i(t+1) - I_i(t)|}, \quad (4.5)$$

where $\alpha_i(t)$ and $I_i(t)$ are the estimated alpha value and the RGB color in frame t . Shahrian *et al.* [30] measure temporal coherence error (TCE) as the mean squared error

Figure 4.11: Qualitative comparison of proposed method for *Amira*, *Office* and *Goat* sequences. (a) Key-frame contour at frame t , (b) curve obtained using proposed in-between generation (yellow), (c) generated trimap using adaptive trimap parameterization, per-frame matte generated using (d) [27], (e) [28], (f) proposed method and (g) using hand-drawn trimap, all at frame $t + 5$. Red indicates the error (best viewed in color).

between the alpha mattes and their respective ground truth for two sequential frames as

$$TCE(t) = MSE((\alpha_t^{GT} - \alpha_{t-1}^{GT}), (\alpha_t - \alpha_{t-1})), \quad (4.6)$$

where α_t^{GT} is the ground truth alpha at frame t . However, the calculation of TCE requires ground truth data to be available for evaluation. In order to obtain an unbiased evaluation criteria, the proposed method uses both the measures mentioned above for quantitative evaluations with respect to video matting.

4.4.1 Experiment 1: Evaluation of temporal trimap propagation step

In this experiment, the trimap propagation step of the proposed video matting method is compared qualitatively and quantitatively to demonstrate the effectiveness of the algorithm in generating temporally coherent trimaps. The dataset comprises sequences containing regions of low contrast, shadows and significant camera motion which are challenging for any trimap propagation algorithm. The publicly available optical flow algorithm [54] is used to determine the flow field. The number of key-frames are defined based on the video sequence, complex sequences with fast motion requiring key-frames

Figure 4.12: Quantitative comparison of the proposed trimap propagation framework with Bayesian matting [27] and temporally coherent matting [28] using (a) mean temporal coherence error (TCE) and (b) mean temporal jitter error (TJE) metrics for the extracted mattes.

defined closely. In order to compare with [58], which assumes that the sequences are shot with a fixed camera and requires a clean background plate to learn the background model, we implemented their method since their code is not publicly available. They produce poor results in videos with dynamic background as shown by the trimap and matte extracted in Fig. 4.2(c, d).

Fig. 4.11 shows the qualitative comparison of the mattes extracted by applying the same image matting algorithm [82] on the trimaps generated by different methods to the results of a careful hand-drawn trimap. The methods compared include our proposed trimap generation strategy, [27], and [28]. Red indicates the error in the mattes. Since the trimaps have been generated using different propagation criteria, the quality of the propagated trimaps alone should have an influence on deciding the effectiveness of the algorithm. However, as it is impractical to evaluate the quality of the trimap directly, the mattes estimated from the trimaps using a common matting algorithm is utilized instead. As can be seen from Fig. 4.11(d, e, f), the proposed method produces the least error deviation by combining both motion and shape interpolation. Applying optical flow alone fails due to the low contrast between background and foreground and fast motion in the *Amira*, *Goat* and *Office* sequences leading to poor results for [27]. The trimaps in Fig. 4.11(c) are of non-uniform thickness throughout the image area, resulting in a matte visually similar to that generated from a trimap drawn using hand.

Figure 4.13: Visual comparison of proposed graph model against temporal Laplacian refinement. (a) Three sequential frames of Unicorn and its zoomed regions, (b) temporal Laplacian result and (c) proposed method.

Table 4.1: Ranks of video matting methods with respect to mean squared error (MSE) and temporal coherence error (TCE) and temporal jitter error (TJE) metrics.

MSE						
Method	Face	Dancer	Arm	Smoke	Woman	Overall
Proposed	1.39	1.0	1.40	3.0	1.20	1.60
Shahrian <i>et al.</i> [30]	1.60	2.07	1.72	1.56	1.79	1.75
SnapCut [26]	3.02	3.02	3.03	1.77	3.29	2.82
Bai <i>et al.</i> [28]	3.97	3.89	3.85	3.66	3.70	3.81

TCE						
Method	Face	Dancer	Arm	Smoke	Woman	Overall
Proposed	2.26	1.44	1.25	3.06	1.03	1.81
Shahrian <i>et al.</i> [30]	2.52	1.60	1.95	1.16	1.98	1.84
SnapCut [26]	1.93	3.31	2.85	2.60	3.55	2.85
Bai <i>et al.</i> [28]	3.27	3.63	3.93	3.27	3.43	3.5

TJE						
Method	Face	Dancer	Arm	Smoke	Woman	Overall
Proposed	1.58	1.62	1.76	1.93	2.09	1.79
Shahrian <i>et al.</i> [30]	3.32	1.45	2.21	1.29	2.47	2.14
SnapCut [26]	2.53	3.73	2.52	3.91	3.30	3.20
Bai <i>et al.</i> [28]	2.57	3.19	3.50	2.85	2.13	2.84

The extracted mattes are also compared quantitatively against the ground truth using mean temporal coherence error in Fig 4.12(a). The same sequences are used to compare against the second metric - temporal jitter error although its evaluation does not require ground truth. The results are reported in Fig. 4.12(b). The proposed method produces the least error in TCE metric for the three sequences without needing explicit binary segmentation on each frame as in [28]. For the same sequences, the TJE error is the least except in one of the sequences showing the effectiveness of the proposed trimap propagation framework.

4.4.2 Experiment 2: Evaluation of multi-frame graph model

In this experiment, the multi-frame graph model in the proposed algorithm is evaluated and compared with the conventional temporal Laplacian refinement followed by existing methods [26, 46, 53]. The temporal Laplacian refinement is formulated using a data term that considers the matting Laplacian for the current frame and a smoothness term for

ensuring consistency with the previously estimated frame. It uses the matte from the previous frame that is warped using optical flow to the current frame to formulate the smoothness term. Optical flow is erroneous in low contrast can introduce errors even before alpha estimation. Three consecutive frames of the *Unicorn* sequence along with their zoomed in regions are shown in Fig. 4.13(a). The corresponding mattes extracted using the temporal Laplacian refinement and the proposed multi-frame graph model are shown in Fig. 4.13(b) and Fig. 4.13(c) respectively. The errors introduced in the previous frames are propagated to the current frame and leads to temporal inconsistencies as seen in the zoomed matte regions of Fig. 4.13(b). By using the temporal neighbors in the feature-space, along with the temporal dictionary for initial matte estimation, the proposed method is able to extract temporally consistent results across the entire block of frames.

4.4.3 Experiment 3: Evaluation of video matting algorithm

Quantitative evaluation

The dataset contains five videos with ground truth alpha available. We compare the proposed method against Bai *et al.* [28], SnapCut [26] and Shahrian *et al.* [30]. These methods were designed to maintain temporal coherence in the mattes. First, the accuracy of the extracted mattes is measured by comparing against the ground truth using mean squared error criterion as shown in Table 4.1. The proposed method not only outperforms all the methods overall, but also on each video except one. The poor performance on the Smoke sequence can be attributed to the inaccurate neighborhood estimation that causes noise in the final mattes. Table 4.1 also shows the performance of the proposed video matting framework on the mean temporal coherence error and temporal jitter error metrics over the five video sequences. The overall error is the least in the proposed method for both the measures demonstrating its effectiveness using an unbiased error comparison. Fig. 4.14 shows the temporal coherence error generated by the different methods plotted for each frame of the *Arm* and *Dancer* sequences. As can be seen, the proposed method produces the least fluctuations in temporal coherence while maintaining the lowest error possible.

Figure 4.14: Temporal coherence: Quantitative comparison with [30], [26] and [28] on the *Arm* and *Dancer* sequences. Our mattes not only give smaller error, but also maintains a stable temporal coherence across the frames.

Qualitative evaluation

The performance of the proposed method is evaluated qualitatively against the above three methods on three sequences from the dataset as shown in Fig. 4.15. The low contrast between the goat's beard and the surroundings causes [30] to lose out the beard. The results obtained by [26] and [28] oversmooth the matte resulting in spatial inaccuracy near the goat's mouth. The proposed method are able to achieve a good balance by picking the most relevant samples in building the dictionary. The cluttered background in the *Office* sequence causes the matte to be inconsistent across consecutive frames,

Figure 4.15: Qualitative comparison of the proposed method on *Goat*, *Office* and *Girl* sequences respectively. Arrows indicate regions of low quality mattes, showing the effectiveness of our method. Dataset courtesy of Adobe.

with background regions being misinterpreted as part of the foreground indicated by the different intensities in the sequential mattes. The proposed method is able to eliminate such inaccurate matching by using only the feature-space neighbors for the graph model and constraining the temporal pixels to be consistent. Also, as can be seen from the *Girl* sequence, the proposed approach is able to strike a good balance between temporal coherency and spatial accuracy by extracting mattes which are smoother in the temporal domain without losing out on the finer details like hair.

The performance of the proposed method is further evaluated on several sequences in Fig. 4.16. The *Amira* and *Chimp* sequences present the ability of the proposed method to deal with fine structures like hair. The *Dog* sequence is challenging due to the cluttered background where the snout of the dog is indistinguishable from the background.

Figure 4.16: Estimated video mattes by the proposed method on four sequences.

Table 4.2: Comparison of running time of the proposed method with recent sampling-based approaches

Video	Size	No. of Frames	Total Time (secs)	
			Shahrian <i>et al.</i> [30]	Proposed
Smoke	500×500	90	4491	3798
Arm	640×540	49	3260	1618
Dancer	480×360	40	5488	2803
Face	640×540	78	5378	4955
Archaeology	480×405	128	4980	2524
Woman	450×400	154	5541	2912

The estimated mattes show that the proposed method is able to extract accurate, yet temporally consistent results. The *Cat* sequence contains regions of shadow and dynamic background that lead to breaks in the foreground. The proposed method is able to avoid such discontinuities and preserves temporal coherence.

Runtime comparison Table 4.2 compares the running time of the proposed method with a recent sampling based video algorithm [30]. MATLAB implementations were evaluated on a PC running Intel Xeon 3.2 GHz processor. Sparse coding, as shown in chapter 3, is able to run faster than pairwise approaches. The linear system in multi-frame graph model is solved using the built-in stabilized bi-conjugate gradient function in MATLAB and runs in the order of a few seconds per frame.

4.5 Conclusion

In this chapter, we have presented a video matting framework that is designed specifically to address the need of temporal coherence in video matting. 2-D shape blending is combined with motion information to generate accurate contours across a wide range of complex sequences, followed by generation of temporally accurate trimaps with adaptive thickness. The second step continues the emphasis on temporal coherence by combining the sampling and matte optimization using a block of frames simultaneously. Solving the matte simultaneously for a group of frames is shown to reduce the temporal discontinuities when compared to a per-frame method. The proposed framework is evaluated objectively on two different temporal coherence evaluation metrics and outperforms recent methods on both the measures.

In the next chapter, a second video matting algorithm is introduced that continues the theme of alternative approaches to the compositing equation in estimating the matte.

Chapter 5

Reconstruction Errors as Alpha Matte

5.1 Introduction

The compositing equation describes the alpha value as the interpolation factor that blends the foreground and background together to obtain the composite color. As mentioned earlier, sampling methods find the best foreground and background sample pair that satisfies eq. (1.1) and use it to estimate the alpha value. The selection of the best (F, B) pair varies across different algorithms, albeit a common factor is to minimize the chromatic distortion error. The chromatic distortion error at a pixel i is a measure of the deviation from the compositing equation and is defined as

$$\varepsilon_i = \|I_i - (\hat{\alpha}_i F_i + (1 - \hat{\alpha}_i) B_i)\|. \quad (5.1)$$

Eq. (5.1) is indicative of the error produced in reconstructing the observed color at a pixel using the best (α, F, B) triplet from among the candidate samples. The significance of this residual error has been left unexamined in literature. In this chapter, continuing with the theme of alternative approaches to the compositing equation, a sampling-based algorithm is proposed that looks at matte estimation from the perspective of reconstruction error.

Fig. 5.1 illustrates the motivation behind using reconstruction error as a measure of the matte in a real image. A zoomed region of the input image in Fig. 5.1(a) and its trimap are shown in Fig. 5.1(b), representing a hairy region containing mixed pixels. The local smoothness assumption between the alpha values of neighboring pixels

is paramount to extracting a good matte. In a real image, the alpha would gradually transition between the definite foreground (F) and the definite background (B) with the true mixed pixel alphas having an intermediate value. As can be seen in Fig. 5.1(c), the RGB color distribution of pixels in the image patch varies smoothly between the foreground and background with the blending peaking at the middle of the unknown region. Similarly, the error obtained during reconstruction using F and B samples can be thought of as a probability measure that varies smoothly between the foreground and background regions, gradually rising from the definite regions and peaking at true mixed pixels. A small error obtained during reconstruction using F samples indicates higher probability for that pixel to belong to foreground as shown in Fig. 5.1(d) where the error rises smoothly as we proceed towards the background pixels. Conversely, an opposite trend is observed when B samples are used for reconstructing the pixel color as shown in Fig. 5.1(e). The actual distribution of alpha values in the patch is plotted in Fig. 5.1(f) which agrees with the behavior of the error plots shown beside it. This chapter introduces a matting algorithm that uses both the reconstruction errors from foreground and background to estimate a smoothly varying matte.

The previous chapter demonstrated the effectiveness in using multiple frames to ensure temporal consistency across neighboring frames. However, one of the drawbacks is the inaccurate neighbors that creep up at a pixel across the frames. A pixel-based neighborhood search can introduce noise into the matching process forcing incorrect pixels to share similar alpha values. For example, a pixel in the foreground may be inaccurately matched along with a neighbor in the background due to noise. In order to handle this, two patch-based multi-frame video matting algorithms for temporal coherency are introduced in this chapter. The first one incorporates a non-local means (NLM) framework that is prevalent in video denoising to optimize the error-based mattes. Neighboring patches in the multi-frame block are selected using coherency sensitive hashing [90]. The second framework builds on the graph-based optimization introduced in the previous chapter with two major additions - the inclusion of motion information of a patch in defining the graph weights, and using a patch as a whole to mine out temporal and spatial neighbors. Both the frameworks are shown to have a particular advantage in maintaining temporal coherence of the mattes using qualitative and quantitative comparisons on various sequences.

Figure 5.1: Illustration of reconstruction error as alpha matte. (a) Input image with a window showing (b) zoomed region and its trimap, (c) color distribution of the pixels in RGB color space indicating foreground (F), background (B) and unknown regions, plot of reconstruction error in $[0, 1]$ using (d) F samples (e) B samples and (f) distribution of ground truth alpha values in the patch with 0 indicating black and 1 indicating white.

5.2 Sparse reconstruction error as alpha matte

The use of a sparse combination of F and B samples produced better reconstruction than just a pair of (F, B) samples at a pixel as shown in chapter 3. Motivated by this, the method proposed here uses reconstruction error obtained using sparse coding as a measure of the matte. In this context, sample selection and dictionary formation for sparse coding are presented next.

5.2.1 Sample selection and dictionary formation

Conventional sampling methods [8–11] also employ strategies where a comprehensive set of samples can be obtained to account for the different color distributions present in the image. The objective there is not to represent the true F and B colors, but to pick those that better fit the compositing equation. However, as shown in Fig. 5.1, the objective of

using error reconstruction hinges on the assumption that the foreground and background are locally smooth, akin to propagation-based methods. The idea is therefore, to select a local subset of the known regions for the local smoothness assumption to hold. At each pixel, F and B dictionaries are formed by picking the spatially nearest pixels at a radius of 50 pixels from the definite foreground and background regions, respectively. Each sample is represented by the 8-D vector $[R \ G \ B \ L^* \ a^* \ b^* \ x \ y]^T$ formed by concatenating the RGB and CIELAB color-spaces, along with the X - Y coordinates. It is to be noted that the spatial coordinates are included in the feature to help enforce the smoothness constraint in the error estimate. As we move farther away from the trimap boundaries, the error would also increase smoothly. In order to reduce the sample space, the definite F and B regions are clustered into superpixels using SLIC segmentation [80] and the mean value of the pixel feature vectors in each superpixel denotes a sample. The samples for each frame are chosen from within the frame, unlike the sampling method of the previous chapter. As explained earlier, the objective of the algorithm is not to find the best samples for reconstruction, but to use a locally smooth subset of samples. In order to maintain temporal coherence across frames, the patch-based frameworks described later in sections 5.3.1 and 5.3.2 are used.

5.2.2 From L1-regularized error to alpha

The observed color at a pixel is reconstructed with the help of sparse codes built on a dictionary made up of F and B samples in chapter 3. As indicated in Fig 5.1(d), the reconstructed color using F samples alone will be prone to higher error in the background regions and vice versa. Hence, the proposed method requires separate F and B dictionaries in order to determine the reconstruction error with respect to each.

Given an unknown pixel i , let $\mathbf{D}_F^i$ and $\mathbf{D}_B^i$ denote the foreground and background dictionaries formed by locally sampling the definite regions as explained above. The sparse codes with respect to each dictionary are determined as

$$\beta_F = \operatorname{argmin} \left\| v_i - \mathbf{D}_F^i \beta_F^i \right\|_2^2 + \lambda \left\| \beta_F^i \right\|_1, \quad (5.2)$$

$$\beta_B = \operatorname{argmin} \left\| v_i - \mathbf{D}_B^i \beta_B^i \right\|_2^2 + \lambda \left\| \beta_B^i \right\|_1, \quad (5.3)$$

where v_i is the feature vector at unknown pixel i and λ for sparse coding is set to 0.1. The residual errors generated by reconstruction using F and B dictionaries are

$$\xi_F^i = \|v_i - \mathbf{D}_F^i \beta_F^i\|_2, \quad \xi_B^i = \|v_i - \mathbf{D}_B^i \beta_B^i\|_2. \quad (5.4)$$

ξ_F^i (ξ_B^i) is the error generated at the unknown pixel i when its feature is reconstructed using foreground (background) dictionary. A high value for ξ_F^i (ξ_B^i) indicates that the current pixel cannot be reconstructed well enough by the F (B) samples. Fig. 5.2 illustrates the reconstruction errors in four real images where they are plotted for visualization. ξ_F^i should ideally be 0 for foreground pixels and gradually increase as we move toward the background pixels. Similarly, ξ_B^i should ideally be 0 for background pixels and gradually increase as we approach foreground regions as seen in Fig. 5.2(c). A pixel with a true alpha value of 0.5, i.e. a truly mixed pixel should have comparable reconstruction errors in ξ_F^i and ξ_B^i .

The alpha value at a pixel can be interpreted as the probability of the pixel belonging to the foreground. An alpha value of 0.9 indicates 90% probability for the pixel belonging to the foreground. The remaining 10% represents the contribution towards background. Similarly, ξ_F^i and ξ_B^i can be interpreted as non-parametric probability measures of belonging to background and foreground, respectively. More specifically, ξ_B^i represents the probability of belonging to the foreground, given the known background information, i.e., $\xi_B^i = P(f(i)|\mathbf{D}_B)$. ξ_F^i represents the probability of belonging to the background, given the known foreground information - $\xi_F^i = P(b(i)|\mathbf{D}_F) = 1 - P(f(i)|\mathbf{D}_F)$. Based on the above observation, the alpha value is then estimated as

$$\hat{\alpha}_i = \frac{P(f(i)|\mathbf{D}_B)}{P(f(i)|\mathbf{D}_B) + P(b(i)|\mathbf{D}_F)} = \frac{\xi_B^i}{\xi_B^i + \xi_F^i}. \quad (5.5)$$

As can be seen from eq. (5.5), if a pixel truly belongs to the foreground, its foreground reconstruction error ξ_F^i will be smaller than the background reconstruction error ξ_B^i , thereby ensuring α is large. The estimated alpha maps are shown in Fig. 5.2(d) and indicates the effectiveness of this simple formulation using sparse reconstruction error.

Figure 5.2: L1-regularized reconstruction error as alpha matte. (a) Input image and its trimap, plot of (b) ξ_F , (c) ξ_B and (d) estimated alpha map.

5.3 Patch-based temporal coherency frameworks

Since the sampling strategy uses a local spatial subset of samples from within the frame, the alpha estimates obtained above lack temporal coherency by ignoring the information present in the nearby frames. The previous chapter introduced a multi-frame graph model that used feature neighbors at a pixel to maintain coherency. The disadvantage of a pixel-based framework for coherency lies in its inability to restrict the outliers noise that are incorrectly matched to the pixel. It is crucial for a good video matting algorithm to be able to find true neighbors that vary smoothly across the sequence. Two patch-based frameworks for ensuring temporal coherency in the final mattes are presented next which

provide better nearest neighbor matching.

5.3.1 Non-local means using coherency sensitive hashing

Denoising has been used to express the alpha at a pixel as the weighted sum of neighboring pixel alphas in many works [16, 31, 64]. The first framework proposes a patch-based non-local means (NLM) strategy that is derived from video denoising to maintain the temporal consistency across neighboring frames. NLM [91] was originally introduced to remove noise by averaging pixels in an image weighted by non-local patch similarities. The high search complexity in finding non-local neighboring patches restricts its use to a local neighborhood alone. Computing approximate K-nearest neighbors (AKNN) involves minimizing the L2 distance between the corresponding patch and its neighbors within the image or from another image. Two prominent methods of AKNN include kd-tree [92] and locality sensitive hashing [93], which are accurate but prone to large computation times. In order to combat the large computational overload, PatchMatch [42, 94] was introduced for AKNN and later to denoising in the work of Liu *et al.* [95]. PatchMatch relies on a efficient random search for best patches which are propagated to nearby patches and refined in each stage of its iteration. However, this trade-off in efficiency comes at a lower accuracy, requiring more iterations to obtain an accurate nearest neighbor field. Coherency sensitive hashing (CSH) [90] builds on the image coherence assumption of PatchMatch by replacing the random search by a hashing scheme where similar patches are hashed to the same bin. The above modifications lead to increased accuracy while reducing the computation time of PatchMatch. The proposed framework incorporates this CSH patch matching in a multi-frame block, where similar patches in the temporal neighborhood are used to propagate the matches to their neighbors.

The framework is illustrated in Fig. 5.3 where five sequential frames of the *Goat* video are shown. For a given image patch $P_T(i)$ (shown in red) centered at pixel i in frame T , approximate K-nearest neighbors (AKNN) in frames $T - 1$ and $T + 1$ (in blue) are initialized by creating L hash tables based on projection of the patches on Walsh-Hadamard kernels, followed by search for the best candidate patches. Coherency sensitive hashing relies on the assumption that the two images are coherent, i.e., for every pair of similar patches, their neighbors are also likely to be similar. In the next iteration, the AKNN

Figure 5.3: Illustration of the NLM framework that uses approximate K-nearest neighbor (AKNN) patches to maintain temporal coherence across the video. For the red patch, AKNN patches from immediate frames (blue) are used to find coherent patches from the next frames (green).

field for the initial candidates are searched for in their respective temporal neighbors $T - 2$ and $T + 2$. Under this setup, the temporal neighbors for $P_T(i)$ include a series of AKNN's $\{\mathcal{N}_{T-2}, \mathcal{N}_{T-1}, \mathcal{N}_T, \mathcal{N}_{T+1}, \mathcal{N}_{T+2}\}$. $\mathcal{N}_i = \{P(i_{tj})\}_{j=1}^K$ denotes the K neighboring patches in the t^{th} frame. NLM for denoising [95] considers the weighted contribution of the neighboring patches to remove the structured noise in an image patch. A similar strategy is argued for the proposed video matting algorithm whereby the neighboring alpha patches from across frames contribute towards smoothing the matte in the current frame. It is to be noted that the AKNN's are searched for in the image plane, whereas the NLM is to be applied on the corresponding alpha plane. The non-local means estimate of the alpha at a patch centered at pixel i in frame T is [95],

$$\alpha_T(i) = \frac{1}{\Omega} \sum_{t=T-2}^{T+2} \gamma^{|t-T|} \sum_{j=1}^K \hat{\alpha}(i_{tj}) \exp \left\{ -\frac{D_w(P(i), P(i_{tj}))}{2\sigma_t^2} \right\}, \quad (5.6)$$

where $\hat{\alpha}(i)$ is the initial alpha estimate of a patch centered at i and Ω is a normalization constant,

$$\Omega = \sum_{t=T-2}^{T+2} \gamma^{|t-T|} \sum_{j=1}^K \exp \left\{ -\frac{D_w(P(i), P(i_{tj}))}{2\sigma_t^2} \right\}. \quad (5.7)$$

$D_w(\cdot, \cdot)$ is a weighted sum of squared difference (SSD) over 2 patches denoted by

$$D_w(P(i_1), P(i_2)) = \sum_{u \in [-s, s]} (P(i_1 + u) - P(i_2 + u))^2 \exp \left\{ -\frac{\|u\|^2}{2\sigma_p^2} \right\}, \quad (5.8)$$

where s is the width of the patch and σ_p is set to $s/2$. γ is set to 0.9 to control the influence of temporal neighbors. The patches are of size 8×8 . Eq. (5.6) estimates the alpha matte for an entire patch centered at pixel i in frame t . For an alpha patch of width s , all patches whose centers are located within a radius of $\frac{s}{2}$ from i , contain the pixel at i . A simple Gaussian weighted averaging is performed to obtain the final NLM estimate at i as

$$\alpha_g(i) = \frac{\sum_{j \in \phi} \alpha(i) e^{-\frac{\|i - i_j\|^2}{2\sigma_p^2}}}{\sum_{j \in \phi} e^{-\frac{\|i - i_j\|^2}{2\sigma_p^2}}}, \quad (5.9)$$

where ϕ denotes the set of patches that contain the pixel i . The process is repeated for each frame of the video to obtain the final video mattes.

5.3.2 Patch-based multi-frame graph model with motion

The multi-frame weighted graph model introduced in section 4.3 was shown to be effective in maintaining temporal coherency and spatial accuracy simultaneously by combining the data weights and smoothness weights together. However, it still suffered from inconsistencies due to mismatches in defining pixel-based neighbors, especially in sequences with high transparency like *Dancer* and *Smoke*. The mattes were solved together in non-overlapping blocks to save computation time, which meant that the two frames at the edge of each multi-frame block were not able to share their neighbors, despite being near to each other. Li *et al.* [65] introduced motion as a feature for defining the affinity in the Laplacian matrix and showed improved results on moving objects. In this chapter, the proposed framework builds on these drawbacks and extends the graph model to use a patch-based neighborhood search that incorporates the motion of a patch into the graph model.

Fig. 5.4 illustrates our patch-based graph model on three sequential frames. The patch centered at pixel i shown in red is connected to its neighboring patches in the

Figure 5.4: Illustration of the PatchGraph framework that uses motion information of the patch temporal coherence across the video. For the red patch, neighboring patches that share similar motion and appearance share similar weights in the weighted graph model.

current frame, each denoted by a different color. The feature vector at i is defined as,

$$X_i = [P_t(i) \ (d_f, d_b) \ (x, y)]^T, \quad (5.10)$$

where $P_t(i)$ is the RGB patch centered at pixel i reshaped into a one-dimensional vector, (x, y) are the spatial coordinates at i . d_f and d_b denotes the mean forward and backward displacement of the patch centered at i using forward and backward optical flow fields [55]. X_i is a vector of size d_m . The patch size is fixed as 9×9 in order to balance the smoothness and the computational complexity of kd-tree neighbor search. Patches that share similar appearance and motion should also share similar alpha values. The motion warped patches from the current frame are indicated by the dotted rectangles in frame I_{T-1} and I_{T+1} . Therefore, a single patch in red in frame I_T is connected to all the other patches from within the frame and across the frame as shown in Fig. 5.4.

The data weights encode the confidence in the initial alpha estimate generated using reconstruction error. As matte estimation involves the contribution of a foreground and background dictionary separately, the data confidence term in eq. (3.8) ($\gamma(i) =$

$\gamma_{sprec}(i) \cdot \gamma_{colrec}(i)$) cannot be used directly. The confidence for error reconstruction at a pixel i is defined as

$$\gamma_i^e = e^{-\|v_i - [\mathbf{D}_F^i | \mathbf{D}_B^i] \beta^i\|}, \quad (5.11)$$

where β^i are the sparse codes obtained on concatenating the foreground and background dictionaries together in eq. (5.2). γ_i^e indicates the confidence in the samples for error reconstruction, a pixel having high reconstruction error using both the dictionaries together is indicative of the observed pixel color being different from the collected samples.

The smoothness weights among the neighboring patches are encoded as

$$W_{knn}((P(i), P(j))) = 1 - \frac{\|X_i - X_j\|}{\sigma}, \quad (5.12)$$

where σ is fixed as $\frac{d_m}{2}$. The resulting cost function is solved similar to eq. (3.15) to obtain the mattes. One limitation of the multi-frame graph model of the previous chapter is the use of non-overlapping multi-frame blocks. Consecutive frames on the boundary of each block are unable to share their neighbors in defining the model. In order to encode the information across the blocks, a temporal Laplacian refinement [26, 30] is performed to obtain the final mattes. i.e.,

$$\alpha^t = \underset{\alpha^t}{\operatorname{argmin}} (\alpha^{t^T} L_3 \alpha^t + \lambda_T \sum_{t_i=1}^2 (\alpha^t - \alpha^{t-t_i})^T D_{t_i} (\alpha^t - \alpha^{t-t_i})), \quad (5.13)$$

where the L_3 is 3-D Laplacian term using neighboring pixels from a window of size $3 \times 3 \times 3$. The neighboring pixels are taken from frames $t-1$ and $t-2$ that are warped to the current frame t using optical flow [55]. The second term in eq. (5.13) enforces the current frame to bias its alpha values to the estimated alpha values of the previous two frames where α^{t-t_i} is the estimated alpha at frame $t-t_i$ warped to t . λ_T is a constant that defines the importance of the temporal data term and is fixed as 0.1 in our experiments. D_{t_i} is a diagonal matrix that encodes the confidence in the optical flow estimation for the t_i^{th} frame whose diagonal entries are estimated as

$$O_{t_i}^C = e^{-\|I_i^t - I_i^{t-t_i}\|/\mu^{t_i}}, \quad (5.14)$$

where $I_i^{t-t_i}$ is the frame t_i warped to frame t and μ^{t_i} is a scaling constant defined as $\sum_i \|I_i^t - I_i^{t-t_i}\|$. Eq. (5.13) is solved as a linear system to obtain the final alpha matte at frame t .

Figure 5.5: Effect of patch-based temporal coherency frameworks compared to pixel-based estimation on the *Dancer* and *Dog* sequences.

5.4 Experimental Results

The effectiveness of the proposed method is evaluated on the video matting dataset used in the previous chapter. The first framework that uses non-local means is denoted *PatchNLM*, while the second framework is denoted *PatchGraph* in all subsequent evaluations. Trimaps are generated on each frame using the temporal trimap propagation framework described in chapter 4. In the first experiment, the effect of a patch-based approach in maintaining temporal coherency is evaluated on the video sequences. The proposed frameworks are compared qualitatively and quantitatively with recent video matting approaches namely, SnapCut [26], Bai *et al.* [28], Shahrian *et al.* [30] and the algorithm of chapter 4 (denoted *PixelGraph*) using both the temporal coherence metrics in the second experiment. Finally, a few composited sequences obtained using the proposed frameworks are shown to demonstrate its usefulness in video editing applications.

5.4.1 Experiment 1 : Effect of patch-based temporal coherency step

In this experiment, the effectiveness of a patch-based framework in maintaining temporal coherency is evaluated. Pixel-based neighbors are often noisy and may lead to inaccurate matching of neighbors which can produce inconsistencies in the final matte. Fig. 5.5 illustrates the problem in which two sequences from the dataset are shown with their pixel-based neighbors used in estimating the graph model in the second row. The *Dancer* sequence is characterized by a large unknown region of high transparency which is challenging for most sampling-based methods. The pixel-based estimate is unable to capture the smoothness in the unknown region as seen in the second row. Both the temporal frameworks proposed in this chapter, namely PatchNLM and PatchGraph are able to estimate temporally smooth mattes. However, the PatchNLM framework also has the detrimental quality of oversmoothing the hairy regions, leading to loss of detail as seen in the third row. In the *Dog* sequence, the cluttered background causes pixels from the background to be matched inaccurately to the foreground causing noise in the banded region between the two. The noise is visually jarring during playback even though the details are preserved. The patch-based frameworks discussed above ensures that the same patches are matched across the multi-frame block resulting in visually superior and temporally smooth video mattes.

5.4.2 Experiment 2: Evaluation of proposed video matting algorithm

The second experiment involves qualitative and quantitative comparison of the proposed frameworks against other video matting methods including the video matting algorithm presented in chapter 4. Fig. 5.6 shows the visual comparison of the proposed frameworks against recent video matting methods [26, 28, 30, 96] on three sequential frames of *Face* and *Girl2* sequences from the dataset. The yellow arrows indicate the regions of discontinuity or error between consecutive frames. The low contrast between the foreground and the background in *Girl2* is a challenging scenario for most matting algorithms. Laplacian-based smoothing used by SnapCut [26] and Shahrian *et al.* [30], and level-set interpolation used in Bai *et al.* [28] smoothen the mattes extensively resulting in loss of

Figure 5.6: Qualitative comparison of the video mattes on three consecutive frames of *Face* and *Girl2* sequences. (a) Input frames, estimated mattes of (b) SnapCut [26], (c) Bai *et al.* [28], (d) Shahrian *et al.* [30], (e) PixelGraph (chapter 4), (f) proposed PatchNLM and (g) proposed PatchGraph. Yellow arrows indicate discontinuities and errors in the matte. Best viewed electronically.

detail near the boundaries. Pixel neighbors tend to be unreliable for accurate temporal matching causing noise from the background to be matched as part of the foreground in PixelGraph. This is particularly evident in the *Face* sequence where the window grills are mistaken as the foreground in column (e) of Fig. 5.6. The use of patch-based neighbors in

Figure 5.7: Qualitative comparison of the video mattes with ground truth on *Face* and *Woman* sequences. (a) Input frames and (b) zoomed region, mattes of (c) SnapCut [26], (d) Bai *et al.* [28], (e) Shahrian *et al.* [30], (f) PixelGraph, (g) proposed PatchNLM method and (h) ground truth.

both the frameworks of the proposed method enables us to remove such artifacts in the final mattes. Fig. 5.7 compares the extracted mattes visually with the ground truth for the *Face* and *Woman* sequences. Both represent cluttered background with overlapping distributions which is challenging for most sampling-based algorithms. Our error based matte is accurately able to distinguish between the hair and the background when compared to the ground truth, showing its effectiveness in difficult regions. However, if the true F and B samples are not present in the dictionary, the reconstruction error cannot be trustworthy, leading to poor results. Fig. 5.8 compares the proposed PatchGraph framework that uses motion flow in defining neighboring patches to the motion-aware Laplacian method of Li *et al.* [65]. As seen from the results of the zoomed regions in yellow, the proposed method is able to maintain the fine hairy structures while ensuring temporally smooth mattes in subsequent frames.

The proposed frameworks are also compared quantitatively on the two temporal coherence metrics discussed in section 4.4, namely temporal coherence error (TCE) and temporal jitter error (TJE). Table 5.1 uses the TCE metric to compare the algorithms on the five sequences having ground truth. Table 5.2 compares the methods against recent works on the TJE metric. As TJE measures the difference in alpha values between consecutive frames, the ground truth is not required for its computation. Therefore, all the 15 sequences from the dataset are used in the comparison shown in table 5.2. As is evident from the results, the proposed PatchNLM framework is shown to perform better

Figure 5.8: Qualitative comparison of the proposed motion-infused PatchGraph with motion-aware Laplacian approach of Li *et al.* [65] on *Amira2* and *Kim* sequences.

Table 5.1: Comparison of mean temporal coherence error (TCE) measures of different video matting algorithms against the two frameworks of the proposed method.

Video	SnapCut [26]	Bai <i>et al.</i> [28]	Shahrian <i>et al.</i> [30]	PixelGraph	PatchGraph	PatchNLM
<i>Face</i>	3.10	4.36	3.34	4.30	2.56	3.31
<i>Dancer</i>	5.26	5.57	3.63	3.34	1.52	1.65
<i>Arm</i>	4.76	5.89	3.51	1.36	2.31	3.14
<i>Woman</i>	5.54	5.41	3.28	2.41	1.22	3.11
<i>Smoke</i>	4.23	4.33	1.16	4.67	3.32	3.33

on the temporal jitter error measure, whereas the PatchGraph framework performs better overall on the temporal coherence error measure. This disparity in the performance can be attributed to the inherent composition of both the error measures as explained below.

As mentioned in chapter 4, there is no consensus on objectively evaluating the perceptual quality of a video matting algorithm. Temporal coherency is essentially a subjective quality. Since temporal jitter is the most jarring visual effect that humans are highly sensitive to, it has to be minimized. However, this implies that a temporally smoother matte is justifiable even if there is a loss in detail near the mixed pixels (fine hairy structures, for example). Conversely, if the foreground object (actor) is being extracted for film production, apart from reducing jitter in the final composite, it is necessary to include the fine details from the source sequence as well. Eq. (4.5) defines the TJE metric as a measure

Table 5.2: Comparison of mean temporal jitter error (TJE) measures of different video matting algorithms against the two frameworks of the proposed method.

Video	SnapCut [26]	Bai <i>et al.</i> [28]	Shahrian <i>et al.</i> [30]	PixelGraph	PatchGraph	PatchNLM
<i>Face</i>	4.21	4.38	5.18	3.18	2.21	1.81
<i>Dancer</i>	5.70	5.0	2.16	3.35	3.29	1.48
<i>Arm</i>	4.45	5.36	3.5	3.06	3.02	1.58
<i>Woman</i>	4.63	3.26	3.74	3.22	2.96	3.16
<i>Smoke</i>	5.34	4.62	1.38	4.5	2.60	2.53
<i>Cat</i>	2.54	5.77	2.45	4.45	4.36	1.41
<i>Chimp</i>	4.36	5.54	3.45	2.45	3.45	1.81
<i>Girl</i>	5.53	5.31	2.98	2.75	2.55	1.84
<i>Whitegoat</i>	4.17	4.37	3.92	3.91	2.54	2.05
<i>Amira</i>	5.09	5.18	3.54	2.90	2.54	1.72
<i>Office</i>	4.88	4.35	3.64	2.94	2.11	3.05
<i>Girl2</i>	5.54	5.40	3.18	2.45	2.22	2.18
<i>Soccer</i>	4.66	4.35	2.51	3.75	2.96	2.74
<i>Unicorn</i>	4.0	4.08	3.92	3.53	2.71	2.74
<i>Dog</i>	5.16	4.40	3.84	3.79	1.73	2.05

of the temporal flicker and therefore, it favors algorithms of the former category where the onus is on extracting temporally smooth mattes. The true alpha values of the mattes have no bearing on the perceived quality, thereby favoring algorithms that are smoother in the temporal domain. The proposed PatchNLM framework is derived from video denoising, which justifies a smoother alpha matte and hence, its superior performance in reducing temporal jitter error. The TCE metric of eq. (4.6) uses the ground truth transition map to compare the perceived quality of mattes. It ensures lower error for algorithms that simultaneously maintain the spatial accuracy and temporal coherency of the mattes. The graph-based optimization discussed throughout the thesis ensures that the accuracy is maintained using the initial estimate as the data term, while the spatial and temporal neighbors in the feature domain ensure smoothness in the final mattes as evidenced by the better performance of the graph-based frameworks on this metric.

5.4.3 Experiment 3: Evaluation of compositing applications of the extracted mattes

We present qualitative results to evaluate the effectiveness of the extracted mattes for the application of compositing. Compositing is one of the fundamental applications of video matting, finding extensive use in movie industry. As the proposed method formulates

Figure 5.9: Qualitative comparison of the video composites obtained from the Patch-Graph and PatchNLM frameworks on the *Chimp* and *Arm* sequences respectively.

the matting problem from reconstruction errors, there does not exist foreground and background color samples as per the compositing equation. Hence, the foreground and background are solved using the method adopted by [3] using smoothness assumptions on the F and B colors. A cost function that minimizes the composition equation using derivatives of the foreground and background colors along with the matte derivatives are solved using a sparse set of linear equations to generate the F and B colors. The extracted F and α are substituted into the compositing equation (eq. (1.3)) with a new B to obtain the composites as shown in Fig. 5.9. The *Chimp* sequence is characterized by fine hairy structures which need to be extracted while preserving the details for seamless

Table 5.3: Comparison of running time of the proposed PatchNLM method with recent sampling-based approaches

Video	Size	No. of Frames	Total Time (secs)		
			Shahrian <i>et al.</i> [30]	Johnson <i>et al.</i> [96]	PatchNLM
Smoke	500×500	90	4491	3798	3628
Arm	640×540	49	3260	1618	2188
Dancer	480×360	40	5488	2803	2589
Face	640×540	78	5378	4955	4786
Archaeology	480×405	128	4980	2524	2961
Woman	450×400	154	5541	2912	3178

composition. The PatchGraph framework is able to accurately composite the sequence while maintaining the spatial accuracy of the matte as shown in row 3 of Fig. 5.9. The *Arm* sequence represents the challenging problem of motion blur on a textured background. Our PatchNLM framework is able to extract a smoothly varying video matte which is seamlessly composited on a rainy background in the last row.

5.4.4 Runtime performance

Table 5.3 compares the running time of the proposed PatchNLM method with recent sampling-based video matting approaches. MATLAB implementations were evaluated on a PC running Intel Xeon 3.2 GHz processor. PatchGraph uses backward and forward optical flow followed by solving a large linear system limiting its time performance. The proposed method performs comparable to the current approaches without compromising on the quality of the matte.

5.5 Conclusion

This chapter presents a sampling-based matting framework that treats the problem from the perspective of reconstruction error of a feature. The alpha values are locally smooth between the foreground and background regions gradually declining towards background similar to the reconstruction errors with respect to F and B samples. Foreground and background dictionaries, whose bases are used to reconstruct an unknown feature vector with L1-regularization are used to measure the error towards F and B respectively. Two multi-frame frameworks for video matting are presented, each based on exploiting temporal patch-based neighbors in defining the consistency across frames. The former uses

non-local means of approximate neighbors defined using coherency sensitive hashing, while the latter extends the multi-frame graph model by incorporating the motion of the patches in defining the weights of the graph. Experimental evaluations demonstrate that the proposed framework has superior performance over current matting methods using both the temporal coherence metrics.

Chapter 6

Conclusions

This thesis focuses on the development and analysis of new sampling-based algorithms for image and video matting. The methods proposed in this thesis have a unifying theme of using alternative interpretations of the matting problem moving away from the traditionally followed compositing equation. Such reinterpretations can further research in this area to develop matting algorithms that produce high quality matte estimates. An image matting algorithm that uses sparse coding in the image domain to better describe the color feature at the observed pixel is presented first. Two video matting methods are developed with the aim of reducing the temporal inconsistencies and jitter faced by current video matting methods. A novel temporal trimap propagation is proposed to propagate the trimap from user-marked key-frames to unmarked ones using motion flow and 2-D shape blending. A multi-frame graph model that is built using non-local feature neighbors and sparse coded estimates to obtain temporally coherent mattes is presented. A second video matting algorithm that uses sparse reconstruction error as a measure of the matte is presented that contains two frameworks for extracting temporally coherent mattes. Our main contributions are briefly summarized in the next section.

6.1 Summary of Contributions

The major conclusions from this thesis are:

- (i) Color sampling based approaches fail to extract a good matte when the selected (F, B) pair is unable to describe the observed color at the pixel even in the presence

of a comprehensive set of samples. This is particularly evident in regions of high texture and overlapping color distribution where the pairwise assumption of the compositing equation fails to hold as discussed in section 3.1. A new sampling-based image matting method is proposed that uses multiple F and B samples to better reconstruct the color at the pixel than with just an (F, B) pair. Sparse coding is the tool that is used to select a sparse subset of relevant F and B samples from the candidate set, although different feature coding strategies can be used as demonstrated in section 3.3.3. The candidate set is used to build an adaptive dictionary comprising color features from definite foreground and background regions using a probabilistic measure denoting the certainty of the pixel. Special constraints on sparse coding ensures that the sum of the codes from the foreground atoms directly provides the alpha value. The alpha values are shown to provide the least error in reconstructing the pixel color in Fig. 3.12. A graph based optimization step where the initial estimate forms the data term and the spatial Laplacian and feature neighbors forms the smoothness terms is proposed that can be solved efficiently in closed form. Experimental evaluations on a public benchmark dataset at the alpha matting website show that the proposed method ranks among the top against current state-of-the-art. As with other sampling-based methods, the alpha value extracted in areas of high transparency like plastic bag is biased to belong to the background, leading to poor results on such images. Propagation-based methods generally perform better in such scenarios.

- (ii) In video matting, in addition to spatial accuracy in each frame, the mattes are required to be temporally coherent since the human vision system is highly sensitive to jitter as described in chapter 4. Additionally, the trimaps are only available in certain key-frames and requires propagation of the trimap to the unlabeled frames. A new video matting algorithm is proposed that solves both the issues through incorporation of a temporal trimap propagation framework, followed by a multi-frame graph optimization that estimates the mattes of multiple frames simultaneously. 2-D shape blending is adaptively combined with motion flow to obtain accurate in-between contours which are then used to generate trimaps by parameterizing the adaptive thickness between the foreground boundary and unknown regions.

The effectiveness of the trimap propagation step in generating temporally coherent trimaps is demonstrated through quantitative and qualitative evaluations in section 4.4.1. Once the trimaps are generated, the sparse coding method in chapter 3 is extended through a multi-frame sampling strategy to collect the known samples from across the frames in generating the initial estimate. The mattes of four frames are solved together using a multi-frame graph model with the initial estimate as the data term and the temporal feature neighbors as smoothness terms that further enforces temporal coherency in the final mattes. Experimental evaluations on a dataset exclusive for video matting show that the proposed method performs among the top across all temporal coherency measures. Low contrast and temporal resolution makes optical flow unreliable leading to errors in trimap propagation that requires user correction in such scenarios.

- (iii) Sampling methods select the best (F, B) pair that minimizes the reconstruction error at the observed pixel as per the compositing equation. The applicability of this residual error as a probabilistic measure of alpha value is described in section 5.1. The residual error is shown to approximate a smoothly varying alpha field that transitions from a high value at the definite regions, peaking at the true unknown pixels. Once the initial alpha estimates are obtained using the error-based matting strategy, two patch-based frameworks for temporally coherent video matting are presented. The former uses non local means of approximate nearest neighbor patches that are selected using the highly efficient coherency sensitive hashing. A single patch is influenced by the neighbors taken across a five-frame block to generate temporally smooth mattes. The second framework extends the multi-frame graph model presented in chapter 4 to consider a patch as the node in the graph. Optical flow is used to define the motion flow between the patches to encode patches with similar appearance and motion with similar weights. The resulting cost function is solved in closed form to extract the final mattes. Experimental evaluations show the effectiveness of using a patch-based framework in reducing the noise incurred by pixel-based approaches. The two frameworks are shown to effectively produce temporally coherent mattes, each with a respective advantage over a particular temporal coherency metric. If the true F and B samples are not present in the locally

drawn sample set, the reconstruction error-based matte is not trustworthy, leading to poor initial estimates that cause low accuracy in the final mattes.

6.2 Future Work

In this section some of the potential areas of future research related to image and video matting are presented.

- (i) **Extracting better features:** Current matting methods work on single images to extract their mattes. The learning based approaches give matting methods the capability to learn and then improve their result based on what they learned. The learning/example based approaches can provide set of examples as training set for matting systems and then they can learn based on training set and improve their matting result and in order to reduce the user effort in specifying trimaps, they can suggest some potential foreground or background scribbles or even trimaps to user, based on what they learned from training set. Furthermore video matting systems can use the learning/example based approaches to improve process of trimap estimation for all frames even when foreground objects have fast motions in video sequence. The use of deep features that learn their representation from millions of training images for high-level image understanding has shown to improve the performance considerably in computer vision. Defining such set of features which works well for every image is hard and needs more research to be done in future.
- (ii) **Handling long video sequences:** Video matting algorithms must be responsive enough to handle the matting process of large number of frames while interactively keeping the user engaged in matting process. Therefore one of the key research is designing a responsive and interactive user interface for video matting
- (iii) **Removing the compositing equation for propagation approaches:** The α -propagation based matting methods use local color statistics in an image to propagate alpha from known region toward unknown region. The quality of estimated mattes degrades when foreground and background have similar colors due to false correlation between neighboring boundary pixels of different regions. Moreover

strong edges may block the propagation of alpha. Smoothness assumption is taken on the compositing equation to derive the affinity between the pixels. Therefore one potential research area for future work is designing a propagation-based matting method that moves away from the compositing equation in its formulation. As shown in the case of sampling-based methods, superior results can be obtained without relying on the matting equation.

- (iv) **Dealing with fast motions and illumination variance:** It is hard to estimate alpha mattes consistently across frames when foreground object has a fast motion or video captured in low contrast or low lighting conditions because F and B regions present similar color and thus it is hard to discriminate between them using the best reconstruction possible. Moreover, the reliability of optical flows are reduced in dealing with fast motions and low contrast videos therefore we need a more robust feature to describe the motion between the foreground and background in such scenarios.
- (v) **Defining an objective temporal coherency metric:** One of the main issues in video matting is maintaining temporal coherency. However, there is still no consensus on what a suitable performance metric is for videos. As demonstrated in chapter 5, based on the application for which the matte is being extracted, the criteria for measuring the perceptual quality of the algorithm can differ. A low contrast video shot on a hand-phone when used for foreground extraction can afford to lose some of the finer details at the cost of more smoothness in the mixed region. Ideally, a pixel at a position (x, y) in frame t , that undergoes motion to reach (x', y') in frame $t + 5$ should maintain an alpha value that varies smoothly between its path irrespective of the dynamic changes in the background. It would seem intuitive to include motion-flow warped mattes in designing the new metric. However, as optical flow itself is reliable in low contrast and fast motion areas, such a metric would inherently introduce the errors into the process. Despite having a well established benchmark for evaluating image matting, there is still no reliable video matting benchmark.

List of Publications

1. J. Johnson, D. Rajan and H. Cholakkal, “Sparse Codes as Alpha Matte”, *Proc. British Machine Vision Conference (BMVC)*, 2014.
2. J. Johnson, D. Rajan and H. Cholakkal, “Temporal Trimap Propagation using Motion-Assisted Shape Blending”, *IEEE Visual Communication and Image Processing*, 2015.
3. J. Johnson, E. S. Varnousfaderani, H. Cholakkal and D. Rajan, “Sparse Coding for Alpha Matting”, *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3032-3043, 2016.
4. J. Johnson, H. Cholakkal and D. Rajan, “L1-regularized Reconstruction Error as Alpha Matte”, *IEEE Signal Processing Letters*, vol. 24, no. 4, pp. 407-411, 2017.

Bibliography

- [1] T. Porter and T. Duff, “Compositing digital images,” in *ACM Siggraph Computer Graphics*, vol. 18, pp. 253–259, 1984.
- [2] C. Rhemann, C. Rother, J. Wang, M. Gelautz, P. Kohli, and P. Rott, “A perceptually motivated online benchmark for image matting,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1826–1833, 2009.
- [3] 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, no. 2, pp. 228–242, 2008.
- [4] J. Wang and M. F. Cohen, “An iterative optimization approach for unified image segmentation and matting,” in *IEEE International Conference on Computer Vision (ICCV)*, vol. 2, pp. 936–943, 2005.
- [5] J. Sun, Y. Li, S. B. Kang, and H.-Y. Shum, “Flash matting,” in *ACM Transactions on Graphics (TOG)*, vol. 25, pp. 772–778, 2006.
- [6] M. McGuire, W. Matusik, H. Pfister, J. F. Hughes, and F. Durand, “Defocus video matting,” *ACM Transactions on Graphics (TOG)*, vol. 24, no. 3, pp. 567–576, 2005.
- [7] N. Joshi, W. Matusik, and S. Avidan, “Natural video matting using camera arrays,” in *ACM Transactions on Graphics (TOG)*, vol. 25, pp. 779–786, 2006.
- [8] J. Wang and M. F. Cohen, “Optimized color sampling for robust matting,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2007.
- [9] E. S. Gastal and M. M. Oliveira, “Shared sampling for real-time alpha matting,” in *Computer Graphics Forum*, vol. 29, pp. 575–584, 2010.

- [10] K. He, C. Rhemann, C. Rother, X. Tang, and J. Sun, “A global sampling method for alpha matting,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2049–2056, 2011.
- [11] E. Shahrian, D. Rajan, B. Price, and S. Cohen, “Improving image matting using comprehensive sampling sets,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 636–643, 2013.
- [12] E. Shahrian and D. Rajan, “Weighted color and texture sample selection for image matting,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 718–725, 2012.
- [13] J. Sun, J. Jia, C.-K. Tang, and H.-Y. Shum, “Poisson matting,” in *ACM Transactions on Graphics (TOG)*, vol. 23, pp. 315–321, 2004.
- [14] X. Bai and G. Sapiro, “A geodesic framework for fast interactive image and video segmentation and matting,” in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1–8, 2007.
- [15] L. Grady, T. Schiwietz, S. Aharon, and R. Westermann, “Random walks for interactive alpha-matting,” in *Proceedings of Visualization, Imaging and Image Processing (VIIP)*, pp. 423–429, 2005.
- [16] Q. Chen, D. Li, and C.-K. Tang, “Knn matting,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 869–876, 2012.
- [17] X. Chen, D. Zou, S. Z. Zhou, Q. Zhao, and P. Tan, “Image matting with local and nonlocal smooth priors,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1902–1907, 2013.
- [18] M. Jin, B.-K. Kim, and W.-J. Song, “Adaptive propagation-based color-sampling for alpha matting,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 7, pp. 1101–1110, 2014.
- [19] J. Wang, M. Agrawala, and M. F. Cohen, “Soft scissors: an interactive tool for realtime high quality matting,” in *ACM Transactions on Graphics (TOG)*, vol. 26, p. 9, 2007.

- [20] J. Wang, P. Bhat, R. A. Colburn, M. Agrawala, and M. F. Cohen, “Interactive video cutout,” in *ACM Transactions on Graphics (TOG)*, vol. 24, pp. 585–594, 2005.
- [21] E. Eisemann and F. Durand, “Flash photography enhancement via intrinsic relighting,” *ACM transactions on graphics (TOG)*, vol. 23, no. 3, pp. 673–678, 2004.
- [22] G. Petschnigg, R. Szeliski, M. Agrawala, M. Cohen, H. Hoppe, and K. Toyama, “Digital photography with flash and no-flash image pairs,” in *ACM transactions on graphics (TOG)*, vol. 23, pp. 664–672, 2004.
- [23] R. Raskar, K.-H. Tan, R. Feris, J. Yu, and M. Turk, “Non-photorealistic camera: depth edge detection and stylized rendering using multi-flash imaging,” in *ACM Transactions on Graphics (TOG)*, vol. 23, pp. 679–688, 2004.
- [24] J. Wang and M. F. Cohen, “Simultaneous matting and compositing,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2007.
- [25] <http://alphamatting.com>.
- [26] X. Bai, J. Wang, D. Simons, and G. Sapiro, “Video snapcut: robust video object cutout using localized classifiers,” in *ACM Transactions on Graphics (TOG)*, vol. 28, p. 70, 2009.
- [27] Y.-Y. Chuang, A. Agarwala, B. Curless, D. H. Salesin, and R. Szeliski, “Video matting of complex scenes,” *ACM Transactions on Graphics (TOG)*, vol. 21, no. 3, pp. 243–248, 2002.
- [28] X. Bai, J. Wang, and D. Simons, “Towards temporally-coherent video matting,” in *Computer Vision/Computer Graphics Collaboration Techniques*, pp. 63–74, 2011.
- [29] P. Villegas and X. Marichal, “Perceptually-weighted evaluation criteria for segmentation masks in video sequences,” *IEEE Transactions on Image Processing*, vol. 13, no. 8, pp. 1092–1103, 2004.
- [30] E. Shahrian, B. Price, S. Cohen, and D. Rajan, “Temporally coherent and spatially accurate video matting,” *Computer Graphics Forum*, vol. 33, no. 2, pp. 381–390, 2014.

- [31] P. Lee and Y. Wu, “Nonlocal matting,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2193–2200, 2011.
- [32] K. He, J. Sun, and X. Tang, “Fast matting using large kernel matting laplacian matrices,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2165–2172, 2010.
- [33] B. He, G. Wang, C. Shi, X. Yin, B. Liu, and X. Lin, “Iterative transductive learning for alpha matting,” in *IEEE International Conference on Image Processing (ICIP)*, pp. 4282–4286, 2013.
- [34] Y. Shi, O. C. Au, J. Pang, K. Tang, W. Sun, H. Zhang, W. Zhu, and L. Jia, “Color clustering matting,” in *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 1–6, 2013.
- [35] M. A. Ruzon and C. Tomasi, “Alpha estimation in natural images,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 18–25, 2000.
- [36] Y.-Y. Chuang, B. Curless, D. H. Salesin, and R. Szeliski, “A bayesian approach to digital matting,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 264–271, 2001.
- [37] Y. Mishima, “Soft edge chroma-key generation based upon hexoctahedral color space,” 1994. US Patent 5,355,174.
- [38] A. Berman, P. Vlahos, and A. Dadourian, “Comprehensive method for removing from an image the background surrounding a selected subject,” 2000. US Patent 6,134,345.
- [39] L. Karacan, A. Erdem, and E. Erdem, “Image matting with kl-divergence based sparse sampling,” in *IEEE International Conference on Computer Vision*, pp. 424–432, 2015.
- [40] 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, 2007.

- [41] Q. Zhu, L. Shao, X. Li, and L. Wang, “Targeting accurate object extraction from an image: A comprehensive study of natural image matting,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 2, pp. 185–207, 2015.
- [42] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, “The generalized patchmatch correspondence algorithm,” in *Proc. of European Conference on Computer Vision (ECCV)*, pp. 29–43, 2010.
- [43] K. He, J. Sun, and X. Tang, “Guided image filtering,” in *Proc. of European Conference on Computer Vision (ECCV)*, pp. 1–14, 2010.
- [44] Y. Zheng and C. Kambhamettu, “Learning based digital matting,” in *IEEE International Conference on Computer Vision (ICCV)*, pp. 889–896, 2009.
- [45] F. Piti, “An alternative matting laplacian,” in *IEEE International Conference on Image Processing (ICIP)*, pp. 3623–3627, 2016.
- [46] Z. Tang, Z. Miao, and Y. Wan, “Temporally consistent video matting based on bilayer segmentation,” in *IEEE International Conference on Multimedia and Expo (ICME)*, pp. 370–375, 2010.
- [47] J. Moon, D. Kim, and R. Park, “Video matting based on background estimation,” *Proc. World Acad. Sci., Eng. Technol*, vol. 2, pp. 149–152, 2005.
- [48] Y. Li, J. Sun, and H.-Y. Shum, “Video object cut and paste,” in *ACM Transactions on Graphics (TOG)*, vol. 24, pp. 595–600, 2005.
- [49] M. Gong, L. Wang, R. Yang, and Y.-H. Yang, “Real-time video matting using multichannel poisson equations,” in *Proceedings of Graphics Interface*, pp. 89–96, 2010.
- [50] Z. Tang, Z. Miao, Y. Wan, and D. Zhang, “Video matting via opacity propagation,” *The Visual Computer*, vol. 28, no. 1, pp. 47–61, 2012.
- [51] M. Gong, Y. Qian, and L. Cheng, “Integrated foreground segmentation and boundary matting for live videos,” *IEEE Transactions on Image Processing*, vol. 24, no. 4, pp. 1356–1370, 2015.

- [52] O. Wang, J. Finger, Q. Yang, J. Davis, and R. Yang, “Automatic natural video matting with depth,” in *Pacific Conference on Computer Graphics and Applications*, pp. 469–472, 2007.
- [53] S.-Y. Lee, J.-C. Yoon, and I.-K. Lee, “Temporally coherent video matting,” *Graphical Models*, vol. 72, no. 3, pp. 25–33, 2010.
- [54] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, “High accuracy optical flow estimation based on a theory for warping,” in *Proc. of European Conference on Computer Vision (ECCV)*, pp. 25–36, 2004.
- [55] T. Brox and J. Malik, “Large displacement optical flow: descriptor matching in variational motion estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 3, pp. 500–513, 2011.
- [56] M. Sindeev, A. Konushin, and C. Rother, “Alpha-flow for video matting,” in *Proc. of Asian Conference on Computer Vision*, pp. 438–452, 2012.
- [57] A. Agarwala, A. Hertzmann, D. H. Salesin, and S. M. Seitz, “Keyframe-based tracking for rotoscoping and animation,” in *ACM Transactions on Graphics (ToG)*, vol. 23, pp. 584–591, 2004.
- [58] M. Sarim, A. Hilton, and J.-Y. Guillemaut, “Temporal trimap propagation for video matting using inferential statistics,” in *Proc. of IEEE International Conference on Image Processing*, pp. 1745–1748, 2011.
- [59] N. Apostoloff and A. Fitzgibbon, “Bayesian video matting using learnt image priors,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. I:407–414, 2004.
- [60] V.-Q. Pham, K. Takahashi, and T. Naemura, “Real-time video matting based on bilayer segmentation,” in *Proc. of Asian Conference on Computer Vision (ACCV)*, pp. 489–501, 2010.
- [61] L. Vincent and P. Soille, “Watersheds in digital spaces: an efficient algorithm based on immersion simulations,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6, pp. 583–598, 1991.

- [62] H.-Y. Shum, J. Sun, S. Yamazaki, Y. Li, and C.-K. Tang, “Pop-up light field: An interactive image-based modeling and rendering system,” *ACM Transactions on Graphics (TOG)*, vol. 23, no. 2, pp. 143–162, 2004.
- [63] C. Rother, V. Kolmogorov, and A. Blake, “Grabcut: Interactive foreground extraction using iterated graph cuts,” in *ACM transactions on graphics (TOG)*, vol. 23, pp. 309–314, 2004.
- [64] I. Choi, M. Lee, and Y.-W. Tai, “Video matting using multi-frame nonlocal matting laplacian,” in *Proc. of European Conference on Computer Vision (ECCV)*, pp. 540–553, 2012.
- [65] D. Li, Q. Chen, and C.-K. Tang, “Motion-aware knn laplacian for video matting,” in *IEEE International Conference on Computer Vision (ICCV)*, pp. 3599–3606, 2013.
- [66] A. Hosni, C. Rhemann, M. Bleyer, and M. Gelautz, “Temporally consistent disparity and optical flow via efficient spatio-temporal filtering,” in *Pacific-Rim Symposium on Image and Video Technology*, pp. 165–177, 2011.
- [67] N. Brosch, A. Hosni, C. Rhemann, and M. Gelautz, “Spatio-temporally coherent interactive video object segmentation via efficient filtering,” in *Pattern Recognition*, pp. 418–427, 2012.
- [68] C. Rhemann, A. Hosni, M. Bleyer, C. Rother, and M. Gelautz, “Fast cost-volume filtering for visual correspondence and beyond,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3017–3024, 2011.
- [69] F. Zhong, X. Qin, Q. Peng, and X. Meng, “Discontinuity-aware video object cutout,” *ACM Transactions on Graphics (TOG)*, vol. 31, no. 6, pp. 175:1–10, 2012.
- [70] D. Zou, X. Chen, G. Cao, and X. Wang, “Video matting via sparse and low-rank representation,” in *IEEE International Conference on Computer Vision*, pp. 1564–1572, 2015.
- [71] M. Erofeev, Y. Gitman, D. Vatolin, A. Fedorov, and J. Wang, “Perceptually motivated benchmark for video matting,” in *Proc. of British Machine Vision Conference (BMVC)*, 2015.

- [72] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, “Robust face recognition via sparse representation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 2, pp. 210–227, 2009.
- [73] J. Yang, J. Wright, T. S. Huang, and Y. Ma, “Image super-resolution via sparse representation,” *IEEE Transactions on Image Processing*, vol. 19, no. 11, pp. 2861–2873, 2010.
- [74] J. Yang, K. Yu, Y. Gong, and T. Huang, “Linear spatial pyramid matching using sparse coding for image classification,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1794–1801, 2009.
- [75] J. Wright, Y. Ma, J. Mairal, G. Sapiro, T. S. Huang, and S. Yan, “Sparse representation for computer vision and pattern recognition,” *Proceedings of the IEEE*, vol. 98, no. 6, pp. 1031–1044, 2010.
- [76] H. Lee, A. Battle, R. Raina, and A. Y. Ng, “Efficient sparse coding algorithms,” *Advances in neural information processing systems*, vol. 19, pp. 801–808, 2007.
- [77] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, *et al.*, “Least angle regression,” *The Annals of statistics*, vol. 32, no. 2, pp. 407–499, 2004.
- [78] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, “Online learning for matrix factorization and sparse coding,” *The Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.
- [79] S. M. Yoon and G. Yoon, “Alpha matting using compressive sensing,” *IET Electronics letters*, vol. 48, no. 3, pp. 153–155, 2012.
- [80] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, “Slic superpixels compared to state-of-the-art superpixel methods,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.
- [81] J. Ju, J. Wang, Y. Liu, H. Wang, and Q. Dai, “A progressive tri-level segmentation approach for topology-change-aware video matting,” in *Computer Graphics Forum*, vol. 32, pp. 245–253, 2013.

- [82] J. Johnson, D. Rajan, and H. Cholakal, “Sparse codes as alpha matte,” in *Proc. of British Machine Vision Conference (BMVC)*, 2014.
- [83] E. S. Varnousfaderani and D. Rajan, “Weighted color and texture sample selection for image matting,” *IEEE Transactions on Image Processing*, vol. 22, no. 11, pp. 4260–4270, 2013.
- [84] S. Lazebnik, C. Schmid, and J. Ponce, “Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 2169–2178, 2006.
- [85] L. Liu, L. Wang, and X. Liu, “In defense of soft-assignment coding,” in *IEEE International Conference on Computer Vision (ICCV)*, pp. 2486–2493, 2011.
- [86] J. Wang, J. Yang, K. Yu, F. Lv, T. Huang, and Y. Gong, “Locality-constrained linear coding for image classification,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3360–3367, 2010.
- [87] T. W. Sederberg and E. Greenwood, “A physically based approach to 2-d shape blending,” in *Proc. of ACM Computer Graphics and Interactive Techniques (SIGGRAPH)*, vol. 26, pp. 25–34, 1992.
- [88] E. N. Mortensen and W. A. Barrett, “Intelligent scissors for image composition,” in *Proc. of ACM Computer Graphics and Interactive Techniques (SIGGRAPH)*, pp. 191–198, 1995.
- [89] M. Muja and D. G. Lowe, “Fast approximate nearest neighbors with automatic algorithm configuration,” *Proc. of International Conference on Computer Vision Theory and Applications*, vol. 2, no. 331-340, p. 2, 2009.
- [90] S. Korman and S. Avidan, “Coherency sensitive hashing,” in *IEEE International Conference on Computer Vision (ICCV)*, pp. 1607–1614, 2011.
- [91] A. Buades, B. Coll, and J.-M. Morel, “A non-local algorithm for image denoising,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 60–65, 2005.

- [92] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu, “An optimal algorithm for approximate nearest neighbor searching fixed dimensions,” *Journal of the ACM*, vol. 45, no. 6, pp. 891–923, 1998.
- [93] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *Proc. of ACM Symposium on Theory of Computing*, pp. 604–613, 1998.
- [94] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, “Patchmatch: a randomized correspondence algorithm for structural image editing,” *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3, p. 24, 2009.
- [95] C. Liu and W. T. Freeman, “A high-quality video denoising algorithm based on reliable motion estimation,” in *Proc. of European Conference on Computer Vision (ECCV)*, pp. 706–719, 2010.
- [96] J. Johnson, E. S. Varnousfaderani, H. Cholakal, and D. Rajan, “Sparse coding for alpha matting,” *IEEE Transactions on Image Processing*, vol. 25, no. 7, pp. 3032–3043, 2016.