Video Matting of Complex Scenes

Yung-Yu Chuang[1] *Aseem Agarwala*[1] *Brian Curless*[1] *David H. Salesin*[1,2] *Richard Szeliski*[2]

[1]University of Washington [2]Microsoft Research

Abstract

This paper describes a new framework for *video matting*, the process of pulling a high-quality alpha matte and foreground from a video sequence. The framework builds upon techniques in natural image matting, optical flow computation, and background estimation. User interaction is comprised of garbage matte specification if background estimation is needed, and hand-drawn keyframe segmentations into "foreground," "background," and "unknown". The segmentations, called *trimaps*, are interpolated across the video volume using forward and backward optical flow. Competing flow estimates are combined based on information about where flow is likely to be accurate. A Bayesian matting technique uses the flowed trimaps to yield high-quality mattes of moving foreground elements with complex boundaries filmed by a moving camera. A novel technique for smoke matte extraction is also demonstrated.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Bitmap and framebuffer operations; I.4.6 [Image Processing and Computer Vision]: Segmentation—Pixel classifi cation

Keywords: Alpha channel, blue-screen matting, image-based rendering, layer extraction, matting and compositing, video processing.

1 Introduction

Video matting is a critical operation in commercial television and film production, giving a director the power to insert new elements seamlessly into a scene or to transport an actor into a completely new location. In the *matting* or *matte extraction* process, a foreground element of arbitrary shape is extracted, or *pulled*, from a background image. The *matte* extracted by this process describes the opacity of the foreground element at every pixel. Combined with the foreground color, the matte allows an artist to modify the background or to *composite* (i.e., transfer) the foreground onto a new background. *Still-image matting*, a somewhat user-intensive process, is useful for photo editing operations, and several techniques have been demonstrated for this application. In this paper, we address the more challenging problem of *video matting*: pulling a matte from a video sequence of a foreground element against a natural background. The challenge is to achieve the quality currently attainable with recently developed still-image matting techniques without requiring the user to edit each frame and without introducing temporal artifacts.

The most common methods for pulling video mattes are blue screen matting, difference matting, and rotoscoping. In *blue screen matting*, also known as *chroma keying*, foreground elements are filmed in front of a solid color background. Vlahos (as summarized by Smith and Blinn [1996]) pioneered this technique and developed a number of heuristics with tunable parameters for extracting the matte from each frame. This method can be fairly effective but is restricted to tightly controlled studio environments.

For natural backgrounds, an alternative approach is to begin with a *clean plate*, i.e., a shot of the scene without the actors present, and to then subtract the clean plate from the frames containing the actors. Mapping the difference values at each pixel to opacities yields a *difference matte* [Kelly 2000]. The mapping is user-defined and fails where there are similarities in color between foreground and background, requiring additional user interaction to correct the matte.

The last commonly used production matting technique is *rotoscoping*. In this process, the user draws an editable (e.g., B-spline) curve around the foreground element at each frame or at selected *keyframes*, often with the help of an image snapping tool that adheres to high gradient areas [Gleicher 1995; Mortensen and Barrett 1995; Blake and Isard 1998]. When placed at keyframes, the curves can then be interpolated or tracked over time so as to adhere to foreground contours [Mitsunaga et al. 1995; Blake and Isard 1998]. Tracked results tend to require less editing than interpolated results, but they often still require frame-by-frame hand adjustment in order to pull a high-quality matte. In addition, post-processing is needed to convert the contour into opacity profiles using either ad hoc feathering or smoothness assumptions about the background and foreground.

One of the most significant limitations of rotoscoping is its inability to work in regions where the matte is complex and not easily tracked over time with curves, e.g., around wisps of hair. By contrast, recent advances in still image matting, especially *Bayesian matting* [Chuang et al. 2001], have proven quite successful in pulling mattes in such difficult areas. Bayesian matting begins with a user-supplied *trimap*, i.e., a segmentation of the scene into three regions: "definitely foreground," "definitely background," and "unknown" (Figure 3). By collecting nearby foreground and background statistics, the opacity, as well as foreground and background colors, can be estimated at each pixel in the unknown region. The difficulty in using this technique for video matting, however, is the need for the user to create a trimap at each frame.

In this paper we describe a video matting approach that builds upon the Bayesian matting method and leads to a new kind of rotoscoping tool. The approach employs computer vision algorithms to incorporate as much information from other frames as possible. In particular, we leverage optical flow techniques to flow trimaps between hand-drawn trimap keyframes, thus reducing user involvement. Further, when the background can be estimated with mosaicking techniques, we improve both the trimap and the matte by borrowing the background colors from the nearest frame in which those colors are known to be uncontaminated by the foreground.

Our algorithm for flowing the trimaps based on optical flow is novel. However, the primary contribution of this paper is finding and adapting a good set of existing algorithms and devising an overall framework for applying them together. Using our framework and a modest amount of user interaction, we demonstrate detailed matte extractions for actors with complex silhouettes filmed against natural backgrounds by both stationary and moving cameras. In addition, we demonstrate a simple extension that allows matte extraction of foreground smoke.

The remainder of the paper is organized as follows. First, we discuss the existing techniques we build upon (Section 2) and then describe our video matting algorithm (Section 3). Next, we show results of applying our matting algorithm to video footage both with and without background estimation (Section 4). Finally, we summarize the work and describe a number of future research directions (Section 5).

2 Background

In this section, we describe the three main components of prior work that we use in building our video matting algorithm: Bayesian matting, optical flow, and background estimation. In each case, we discuss related work and the particular method we have selected.

2.1 Bayesian matting

Recently, several techniques have emerged to solve the *natural image matting* problem, i.e., extracting a matte from a still image of a foreground object in front of a natural background. More formally, given the compositing equation at a pixel [Porter and Duff 1984],

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

compute the foreground color F, background color B, and opacity (a.k.a., *alpha*) α that yields the observed color C. Clearly, this problem is underconstrained (3 knowns and 7 unknowns). To simplify matters, the latest techniques require the user to specify a trimap. The matte is then extracted only in the unknown region using the nearby background and foreground colors to constrain the problem.

Corel's Knockout package appears to do this by computing weighted sums of foreground and background pixels that abut the unknown region, followed by a simple projection step to estimate alpha, as described in patents (e.g., [Berman et al. 2000]). Ruzon and Tomasi [2000] introduced a statistical method that partions the unknown band into regions and builds multi-modal foreground and background distributions within these regions. These distributions are first interpolated to estimate alphas, and then the mean foreground and background colors are perturbed until they satisfy the compositing equation.

More recently, Hillman *et al.* [2001] and Chuang *et al.* [2001] improved on Ruzon and Tomasi's technique. Hillman *et al.* use principal component analysis (PCA) to estimate the optimal alpha, foreground and background simultaneously, while Chuang *et al.* use a Bayesian approach. In this paper, we adopt the technique of Chuang *et al.* which appears to yield the best mattes.

Given an observation C for a pixel, the algorithm tries to find the most likely values for α, F and B. Using Bayes's rule, we can express the problem as the maximization over a sum of log-likelihoods:

$$\begin{aligned} &\arg\max_{F,B,\alpha} L(F, B, \alpha|C) \\ = &\arg\max_{F,B,\alpha} L(C|F, B, \alpha) + L(F) + L(B) + L(\alpha) \end{aligned} \tag{2}$$

where $L(\cdot)$ is the log-likelihood function, i.e., the log of probability $L(\cdot) = \log[P(\cdot)]$, and the $L(C)$ term is dropped, because it is a constant with respect to the optimization parameters.

The algorithm proceeds by growing, contour by contour, into the unknown region, heading inward both from the foreground the background borders. At each unknown pixel, a circular region encompasses a set of trimap foreground and background pixels, as well as any foreground and background values previously computed nearby in the unknown region. The foreground samples are then separated into clusters, and weighted mean and covariance matrices are used to derive Gaussian distributions that describe $P(F)$. The same is done for the background colors. Given these distributions, the Bayesian matting approach solves for the maximum-likelihood foreground, background, and alpha at the unknown pixel.

Hillman *et al.* [2001] have additionally applied their alpha matting approach to moving image sequences. They match a low-resolution version of the current image to the previous image and classify each pixel as foreground or background if the corresponding pixels are mostly of the same foreground/background class. Undecided pixels are classified by searching for corresponding edges in the previous frame and using their foreground and background color statistics. Unfortunately, it is hard to gauge the quality of their approach since only three static frames from a simple sequence are presented.

2.2 Optical flow

Optical flow algorithms can be used to estimate the inter-frame motion at each pixel in a video sequence. Given two neighboring frames, C_i and C_{i+1}, we can think of each pixel $\mathbf{x}$ in C_{i+1} as coming from a shifted location $\mathbf{x} + \mathbf{u}$ in C_i:

$$C_{i+1}(\mathbf{x}) = C_i(\mathbf{x} + \mathbf{u}) \tag{3}$$

where $\mathbf{u}$ describes the "velocity" of the pixel and is itself a spatially varying function over the image. We refer to $\mathbf{u}(\mathbf{x})$ as the *flow field* between two frames.

Over the years, researchers have developed a number of techniques for estimating the flow field, many of which are compared and summarized by Barron *et al.* [1994]. Two assumptions common to many of these techniques are that the color of a source and destination pixel should be similar, and that the flow field should exhibit some amount of spatial coherence. Thus, the problem becomes one of optimizing a data term (color similarity) plus a regularization term (smooth flow).

One of the better-performing optical flow techniques is due to Black and Anandan [1996]. In addition to estimating regularized flow, their technique employs robust statistics to avoid large errors caused by outliers and to allow for discontinuities in the flow field. Their method handles large motions using a multi-scale approach to flow estimation. In our video matting process, we have incorporated Black's optical flow algorithm, for which an implementation is available on the author's Web page.

In order to use optical flow to its fullest advantage, we make two important observations. The first observation arises when we consider *disocclusions* in an image sequence, i.e., when a feature not present in one frame appears in the next frame. Optical flow breaks down here because it cannot find source pixels that "explain" the new feature. However, if we view this same event when running through the frames in the reverse direction, the event becomes an *occlusion*, and optical flow does not have a problem. This insight has also been noted and used in the image coding literature [Sun et al. 2000]. We'll refer to it as "Observation 1."

The second observation is that optical flow tends to perform poorly at the boundaries between foreground and background layers that have distinct motions, yet are blended together. As noted above, Black's algorithm does allow for discontinuities in the flow field. However, we have found that this allowance does not always work, particularly for complex silhouettes. This second insight will be called "Observation 2."

Later in the paper (Section 3.2), we use these observations to decide how to flow information through the spatiotemporal video volume.

2.3 Background estimation

Clean plate techniques are straightforward when the camera is locked down or when it is attached to a motion control rig that permits reproducing camera motion both with and without the actors. In some cases, however, a partial clean plate can be assembled from nearby video frames, if the background motion can be reliably estimated from frame to frame. Such *image mosaicking* techniques

have recently been used in MPEG-4 video coding to compactly transmit a static portion of the scene viewed from a panning camera [Lee et al. 1997]. Once a conservative foreground mask (sometimes called a *garbage matte*) has been specified, through either manual [Lee et al. 1997] or automated [Wang and Adelson 1994] means, the remaining background fragments can be assembled using perspective image mappings to form a composite mosaic [Szeliski and Shum 1997], which can then be reprojected into each original frame to form a dynamic clean plate. While these previous approaches have been successfully applied in image coding and surveillance, they have not been used to obtain pixel-accurate alpha mattes.

In this paper, we require user input to establish a garbage matte, and we adopt the mosaicking method of Szeliski and Shum [1997] to compute frame-to-frame registration. This method works under the assumption that the background undergoes only planar-perspective transformation. This assumption is equivalent to requiring that the background be planar or, less restrictively, that the camera's optical center translate a negligible amount relative to the distance to the background between frames being registered.

3 Video matting

Our approach to video matting combines these earlier techniques with a modest amount of user interaction. Figure 1 illustrates the flow of user interaction, data, and computation, which we summarize here.

Where possible, a background plate B can appreciably improve the quality of the mattes obtained. To aid this process, the user draws a set of garbage mattes G that conservatively eliminate the foreground and enable background estimation. Next, the user draws trimaps at selected keyframes. These keyframe trimaps K can be fairly crude, as shown in Figure 3, and thus can be drawn quickly. In particular, the user draws a thick boundary that encompasses the regions where alphas need to be estimated and also partitions the image into foreground and background. To distinguish the two, the user selects which partition or partitions are to be flood-filled as foreground, and the remainder is flood-filled as background. The choice of keyframes is something the user adapts to with experience; for example, keyframes are helpful in areas where the topology of the foreground layer changes.

Once the initial set of trimaps is specified, the labelings are passed through the volume using optical flow, resulting in trimaps T at every frame. The flow of information considers where optical flow is likely to succeed and where it might fail. In order to narrow the bands of uncertainty in the trimaps as they flow through the volume, they are converted to alpha mattes α by the Bayesian matting process at each step of the process and then converted back into trimaps. If background information is available, the flow process can be improved using better alpha mattes (and thus trimaps) and by using a form of difference matting that improves the trimaps in regions where flow fails.

Finally, if the matte is not satisfactory, the user can select a frame and edit the trimap with a simple painting tool. In practice, we provide an image of the alpha matte to edit, as this tends to expose the structure of the image more than the trimap, but we permit the user to paint alphas of only 0 or 1. The edited alpha matte is then converted to a trimap and becomes a new keyframe. We then re-run the trimap interpolation method to make maximum use of the new information.

The output of the system is the estimated foreground F, estimated background $\hat{B}$, and alpha α for every frame. Note that, even when the background is available, the estimated background color may be slightly different in order to satisfy the maximum-likelihood criterion in Bayesian matting.

In the remainder of this section, we discuss in greater detail the fundamental computational blocks: background estimation, trimap interpolation, and alpha estimation. We conclude by describing a simple extension for extracting a smoky foreground layer from a known background.

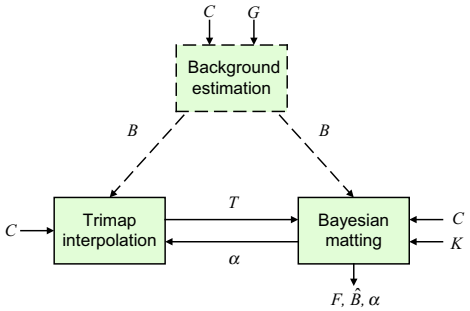

Figure 1 *Video matting flow chart. The primary computational blocks of our process are the background estimation, trimap interpolation, and Bayesian matting components. Each block receives the original image sequence C as input. For background estimation, the user also provides a garbage matte G to remove the foreground; then, a background B is estimated and used in trimap flow and matting. To obtain a video matte, the user provides keyframe trimaps K, which are then converted to alpha mattes α and passed to the trimap interpolator. The interpolated trimaps T are refined through alpha matting at each step. Once the trimaps are complete, the Bayesian matting algorithm generates an estimated foreground F, background $\hat{B}$, and α for all frames.*

3.1 Background estimation

Before beginning background estimation, a garbage matte is necessary to mask out, in a conservative manner, all pixels that could possibly contain foreground contributions. While automatic methods have been published for estimating a garbage matte, and indeed we have tried using the keyframe trimaps to assist in this process, they do not always work. Making errors in the background plate can seriously degrade the quality of the alpha matte. Thus, we follow the approach commonly used in the film industry: we require the user to provide the garbage mattes. For our system, the user must draw a rectangle at a handful of keyframes. These rectangles are automatically interpolated over all the frames, and can then be adjusted immediately by the user to obtain conservative garbage mattes. For a sequence of 100 frames, it took about 5 minutes to specify garbage mattes.

Given the garbage mattes, we follow the registration procedure described in Section 2.3. However, instead of constructing a single clean plate, we fill in the missing parts of each frame with the color from the temporally nearest frame that contains a background color for that pixel. In practice, we find that copying pixels from nearby keyframes has the advantage of reducing errors due to small amounts of parallax or gradual temporal variations that may arise with illumination changes, motion blur, and defocus.

3.2 Trimap interpolation

To pull a complete video matte, we require a trimap for each frame. Constructing the trimaps manually is tedious and time-consuming; on the other hand, a fully automatic approach is unlikely to succeed in giving high-fidelity mattes. Thus, we have developed a semi-automatic system that calculates trimaps for every frame using hand-drawn trimaps for selected keyframes.

To take advantage of spatiotemporal coherence within the video volume, we employ optical flow. The flow field acts as a guide for passing trimap labelings through the volume between keyframes. In principle, we could simply start from the first keyframe and flow its trimap forward in time. Observation 1, however, tells us that disocclusions will cause errors in flow that can frequently be resolved by viewing flow in the opposite direction. The solution is clear: run flow in both directions —forward from one keyframe and back-

ward from the next —and combine the observations according to a measure of per-pixel accuracy for each prospective flow.

In the following sections we describe how we measure the accuracy of optical flow and then present our algorithm for running and combining bi-directional flow.

3.2.1 Accuracy of optical flow

In this section, we devise a method for determining the accuracy of optical flow based on the observations in Section 2.2. This accuracy test is built on precomputed error maps, i.e., the per-pixel prediction errors from one frame to another. To create a forward error map E_i^f for a flow from frame $i-1$ to i, we first compute the image predicted by flow, $C_i(\mathbf{x}-\mathbf{u})$, bilinearly resampled onto the pixel grid. At each pixel of this warped image, we can compute the difference between the predicted color and the observed color as the L_2 distance between the pixels in RGB color space, $E_i^f = ||C_i(\mathbf{x}) - C_i(\mathbf{x}-\mathbf{u})||$. A similarly computed backward error map E_i^b measures the accuracy of flow from frame $i+1$ to i.

After computing error maps for each frame in both directions of optical flow, we combine these to create two sets of *accumulated error maps*. Since we are propagating trimap values from the nearest keyframes, it is not enough to know the error for single frames of optical flow. Instead, we need the accumulated error in optical flow in both directions from the nearest keyframes. If frame i is a keyframe, the forward accumulated error map A_{i+1}^f is simply set to E_{i+1}^f. To compute A_{i+2}^f, we first warp the values of A_{i+1}^f forward in time using the calculated flow. Then, we set A_{i+2}^f equal to this warped accumulated error map plus E_{i+2}^f. This set of calculations is performed from each keyframe forward until the following keyframe is reached. The backward accumulated error maps A_i^b are computed similarly. Thus, at a frame j, the accumulated error maps A_j^f and A_j^b give us a measure at each pixel of the accuracy of flow estimation from the previous and following keyframes.

3.2.2 Combining forward and backward flow

Once optical flow and the error maps have been calculated, we flow the trimaps forward in time from the hand-drawn keyframes. That is, a trimap is formed at frame $i+1$ by warping the keyframe trimap i using the calculated forward flow vectors. We add an additional *validity bit* for each pixel of the flowed trimaps that indicates whether the calculated trimap value is trusted; this validity bit $V_i(x,y)$ is set to 0 if $E_i^f(x,y)$ is greater than a certain threshold (experimentally set to 30). This bit indicates whether a flowed trimap value is to be trusted. When calculating trimap $i+2$ from trimap $i+1$, the validity bits are also warped forward and combined conjunctively. That is, if forward flow vectors indicate that pixel (x,y) in frame $i+1$ flows to (x',y') in frame $i+2$, the validity bit at (x',y') in trimap $i+2$ is set to $V_{i+1}(x,y) \wedge (E_{i+2}^f(x',y') < 30)$.

After calculating each warped trimap, we improve it by performing Bayesian alpha estimation. Untrusted pixels whose validity bits are 0 are labelled as "unknown": we are not confident of the labelling of these pixels and do not want them corrupting the color distributions. The result of the alpha estimation is thresholded (within 10 of "definitely foreground" or "definitely background") and used to improve the trimap. If alpha estimation confirms the labelling of an untrusted pixel, the validity bit is set to 1. If a pixel labelled as "unknown" is identified as "foreground" or "background", the label is changed accordingly.

In the second pass, we start from each keyframe and flow backwards in time. Now, however, we must combine the trimap prediction in the forward direction with the trimap prediction in the backward direction. This is made simple by the accumulated error maps that we calculated earlier. First, the backward trimap prediction for frame i is calculated using a technique symmetric to the forward trimap calculation. Then, at each pixel, we use the lesser of A_i^f and A_i^b to select a trimap value. In practice, we add an additional penalty term to this comparison; if the forward trimap label is "unknown" we add a penalty of 50 to A_i^f, and likewise for the backward trimap. Our results were improved by this penalty term because of Observation 2; "unknown" pixels are near depth discontinuities and should be trusted less.

If a background plate is available, it is useful to incorporate this information when flow is invalid from both directions. In the event that the forward and backward validity bits are both 0, we can resort to a simple form of difference matting. In particular, we compute the RGB L_2 distance between the observed color and the background color and apply user defined thresholds to map the distance to a trimap value.

Finally, during the second pass each computed trimap is passed through alpha estimation, and the results are used to improve the trimap as described earlier.

3.3 Bayesian matting with background

When the background is not available, we run the matting algorithm on the video frames and trimaps exactly as described in Section 2.1. When available, however, the estimated background offers three distinct advantages. First, the distribution of the background is tighter and more accurate. (However, due to sensor noise, the background color is still not a point in color space, but is modeled by a Gaussian distribution with a small standard deviation centered at the estimated color.) Figure 3 shows that the extracted alpha matte is much improved with the help of the clean plate. Second, we no longer need to compute background statistics by sampling neighborhoods and computing means and covariances, thus speeding up the matting process. Finally, the neighborhood windows no longer have to be large enough to span the unknown region and include pixels in the known background region. These last two factors yield a factor of 10 speedup in the matting process.

3.4 Smoke matting

We have also developed a simple extension for extracting mattes of flowing, participating media, such as smoke, given a known background. Figure 4 illustrates the "smoke matting" process for a smoking actor. First, applying the video matting technique described in this section to just the actor, we pull his matte and remove him from the scene. We then compute the difference between the matted-out image and the background image, and all pixels that are different by more than a threshold (5% in our example) are selected for estimating foreground statistics. By treating the color of the smoke as a constant value that is simply composited with a varying alpha over the background, we need only discover that foreground color in order to estimate the matte. For each selected pixel, if it has been mixed with smoke, then we expect its color to lie somewhere along a line in RGB color space between the pixel's known background color and the foreground smoke color. By taking all of the selected pixels, we can construct a set of these lines that, barring degenerate configurations, will roughly intersect at the foreground smoke color. Thus, we compute an initial estimate of the foreground color as the least-squares nearest intersection of all the lines. We then project the approximate intersection point onto each of the original lines and form a foreground distribution used by Bayesian matting. The smoke matte is calculated at every pixel in the image without the actor, and then the actor and smoke mattes are combined.

4 Results

We have applied our new video-matting algorithm to a number of video sequences. The final results are best viewed in video form, but we present stills for several instructive examples in this section.

sequence	frames	initial keys	edited keys	frames/sec
Amira (Fig 2)	91	10	2	15
Kim (Fig 3)	101	11	4	15
Smoke (Fig 4)	176	10	1	15
Baseball (video)	161	12	3	30
Jurassic (video)	96	11	1	30

Table 1 *Details for the five test sequences.*

Figure 2 demonstrates the importance of using bi-directional optical flow for computing interpolated trimaps. For this example, no background estimation is performed. In flowing from keyframe 27 to 28, a disocclusion of the foreground occurs, causing errors in the optical flow, which lead to "unknown" labels in these regions. Flowing from 29 to 28 solves this problem, but introduces some disocclusion in the background. Combining the two flows using the forward/backward sweep described in Section 3.2 gives a more accurate trimap and a better matte and composite.

Figure 3 illustrates the utility of background estimation. After the user provides the garbage matte, a background sequence is constructed. The keyframe trimap, combined with the input image, can be used to create a matte using the Bayesian method even without the background, but the matte contains a number of errors, as shown in the composite over blue. When including the background, but using the method of Ruzon and Tomasi, the matte also exhibits artifacts. In fact, when playing this result back as a video, temporal flashing artifacts arise, which are likely due to the static neighborhoods assembled independently for each frame. Finally, using the Bayesian method combined with the background yields a matte that, while not perfect, is free of many of the artifacts of the other two methods, and composites well over novel backgrounds.

Figure 4 shows a result of smoke matting and illustrates a composite over an edited background. In this case, we acquired the original background by locking down the camera and filming in the absence of the actor. Note that while the resulting matte is not perfect around the silhouette of the actor, for the purposes of background editing in a particular region, the matte is good enough. If more edits were required, the user could focus them only in the regions that would be composited over the background edits.

Both time of execution and the amount of user interaction depend heavily on the nature and resolution of the sequence. Overall, our system can be divided into an unsupervised pre-processing phase and an online phase. For the 640x480 Kim sequence (Figure 4), preprocessing took about 80 seconds per frame; the calculation of optical flow was by far the largest component of this time. For the online phase, drawing a keyframe trimap for this sequence took about 2 minutes per trimap. Interpolating these trimaps took 12 seconds per frame, with alpha estimation as the bottleneck. The interpolation process is unsupervised, and the user can draw more keyframes in parallel. As mentioned in Section 3, the user can then hand-edit any errors to form additional keyframes; this is quick and requires only a couple minutes for the entire sequence.

All of our algorithms scale linearly with pixels of resolution, but other factors such as the area of unknown regions in the trimaps also affect running time. Choosing the frequency of keyframes also depends on the sequence. Complicated geometry such as wispy hair generally requires a keyframe every 10 frames. Sequences of simpler geometry require keyframes every 20–30 frames. Care should be taken to add keyframes when objects enter or leave the field of view. The number of keyframes required for each sequence, along with the number of additional hand-edited keyframes after the first pass of the algorithm are given in Table 1. Two sequences listed are found only on the accompanying video.

5 Conclusion

In this paper, we have presented a new process for video matting that is capable of pulling mattes of foregrounds with complex silhouettes filmed in motion over natural backgrounds. Our primary contribution is a framework that pulls together pieces of existing research and combines their strengths while working around their weaknesses. The result is a new kind of rotoscoping approach that flows trimap image segmentations over time and enables the extraction of detailed mattes around complex foreground silhouettes. In the process, we have introduced a novel method for combining bi-directional optical flow to interpolate trimaps. Further, we have introduced a simple procedure for extracting mattes of participating media filmed against a known background.

In the future, we hope to develop an optical flow algorithm that incorporates the notion of blended, complex foreground and background layers, thus improving flow estimates and allowing us to accumulate foreground and background color distributions *temporally* as well as spatially. In addition, when the user edits selected trimaps, it should be possible to generate new trimaps and mattes more quickly, perhaps interactively, by taking advantage of the locality of these edits. Ultimately, we would like to develop a complete tool with an integrated and powerful user interface that could be tested by and improved with the help of rotoscoping artists.

Acknowledgments

The authors would like to thank Dan Goldman and actors Amira Fahoum, Kim Bishop and Adam Skrine. Thanks to Universal Studios and Amblin Entertainment for permission to use their material. This work was supported by NSF grants CCR-987365 and DMS-9803226 and by industrial gifts from Intel, Microsoft and Pixar.

References

BARRON, J. L., FLEET, D. J., AND BEAUCHEMIN, S. S. 1994. Performance of optical fbw techniques. *International Journal of Computer Vision 12*, 1, 43–77.

BERMAN, A., DADOURIAN, A., AND VLAHOS, P., 2000. Method for removing from an image the background surrounding a selected object. U.S. Patent 6,134,346.

BLACK, M. J., AND ANANDAN, P. 1996. The robust estimation of multiple motions: Parametric and piecewise-smooth fbw fi elds. *Computer Vision and Image Understanding 63*, 1, 75–104.

BLAKE, A., AND ISARD, M. 1998. *Active Contours.* Springer Verlag, London.

CHUANG, Y.-Y., CURLESS, B., SALESIN, D., AND SZELISKI, R. 2001. A Bayesian approach to digital matting. In *Proceedings of Computer Vision and Pattern Recognition (CVPR 2001)*, vol. II, 264 – 271.

GLEICHER, M. 1995. Image snapping. In *Proceedings of ACM SIGGRAPH 95*, 183–190.

HILLMAN, P., HANNAH, J., AND RENSHAW, D. 2001. Alpha channel estimation in high resolution images and image sequences. In *Proceedings of Computer Vision and Pattern Recognition (CVPR 2001)*, vol. I, 1063–1068.

KELLY, D. 2000. *Digital Composition.* The Coriolis Group.

LEE, M.-C., ET AL. 1997. A layered video object coding system using sprite and affi ne motion model. *IEEE Transactions on Circuits and Systems for Video Technology 7*, 1, 130–145.

MITSUNAGA, T., YOKOYAMA, T., AND TOTSUKA, T. 1995. Autokey: Human assisted key extraction. In *Proceedings of ACM SIGGRAPH 95*, 265–272.

MORTENSEN, E. N., AND BARRETT, W. A. 1995. Intelligent scissors for image composition. In *Proceedings of ACM SIGGRAPH 95*, 191–198.

PORTER, T., AND DUFF, T. 1984. Compositing digital images. In *Computer Graphics (Proceedings of ACM SIGGRAPH 84)*, vol. 18, 253–259.

RUZON, M. A., AND TOMASI, C. 2000. Alpha estimation in natural images. In *Proceedings of Computer Vision and Pattern Recognition (CVPR 2000)*, 18–25.

SMITH, A. R., AND BLINN, J. F. 1996. Blue screen matting. In *Proceedings of ACM SIGGRAPH 96*, 259–268.

SUN, S., HAYNOR, D., AND KIM, Y. 2000. Motion estimation based on optical fbw with adaptive gradients. In *Proceedings of International Conference on Image Processing (ICIP 2000)*, vol. I, 852–855.

SZELISKI, R., AND SHUM, H.-Y. 1997. Creating full view panoramic mosaics and environment maps. In *Proceedings of ACM SIGGRAPH 97*, 251–258.

WANG, J. Y. A., AND ADELSON, E. H. 1994. Representing moving images with layers. *IEEE Transactions on Image Processing 3*, 5, 625–638.

Figure 2 *Combining bi-directional flow. For frames 27, 28, and 29 (shown above) and the current trimaps for frames 27 and 29 (neither shown), we estimate the trimap for frame 28. Flowing the trimap in the forward direction ($27 \rightarrow 28$) yields a trimap with extra uncertainty due to disocclusion as the actor's face turns to the right. The trimap predicted by flowing backward ($28 \leftarrow 29$) has less uncertainty in those regions, but suffers from disocclusions where the background is newly exposed. By combining the information in both these trimaps, we can compute trimap 28 automatically, which is better than either one alone. The right column shows a composite into a new scene (top) using the pulled matte (bottom) based on the combined trimap.*

Figure 3 *Background estimation in Bayesian matting. On the far left are the keyframe trimap and estimated background for a single frame. Using the original image (a) and the trimap without the background, Bayesian matting pulls a matte with errors, as shown in a composite over blue (b). Including the background but using instead the method of Ruzon and Tomasi gives an improved result (c), but flaws in the matte are still visible. Applying the Bayesian matting with background results in a higher-quality matte and composite (d).*

Figure 4 *Smoke matting. The input image (a) is part of a sequence for which keyframe trimaps and trimap flow have been computed, and for which a background plate is available. Using the alpha matte as a garbage matte, the foreground actor is removed (b). After applying the participating-media matting algorithm described in Section 3.4, we obtain a matte for the smoke, which is combined with the actor's matte to yield a complete matte (c). We can then composite the foreground over an edited version of the background as shown here (d).*