Bayesian Video Matting Using Motion Based Segmentation

Colm Elliott Dante De Nigris

December 17, 2008

Abstract

Video matting attempts to extract foreground images from an image sequence, as well as the alpha-mattes that describe their transparency. This papers presents an automated *video matting* framework using motion-based segmentation and bayesian matting. We build upon existing techniques for layer segmentation based on optical flow to generate a coarse trimap, by identifying image pixels as belonging to one of background, foreground or unknown. Furthermore, we estimate occluded background for each frame by warping background pixels from neighboring frames. Bayesian methods for video matting are used for identifying detailed alpha-matte and foreground based on a gaussian mixture model distribution for foreground pixels.

1 Introduction

Video matting is used extensively in the film industry to transplant scenes from an original background to an alternate one. To achieve high quality results, accurate alpha mattes are required in order to blend foreground to a new background in a realistic fashion. In general, this process is done manually, where foreground objects are traced out by hand, superimposed on a new background and then *feathered* to make the transitions more realistic. This keying or rotoscoping procedure can be very time consuming as it must be done at every frame. Here we use the work done in [1] and [5] as a basis for automating these segmentation and feathering procedures. By extracting foreground objects as well as the alpha matte from each original frame, we can directly produce realistic transitions from an extracted foreground to a new background, bypassing the requirement for artifical feathering. This approach shows promise, at least for relatively simple scenes where optical flow can be computed accurately.

2 Problem Formulation

Our formulation of the video matting problem consists of two main stages: a background estimation stage that generates a coarse trimap and an estimate of the background (including background occluded by foreground objects) for each frame, and a matting stage, which provides a detailed alpha-matte and foreground.

2.1 Background Estimation

The goal of the background estimation stage is twofold : to generate a trimap for each frame in an image sequence by labelling image pixels as one of foreground, background or unknown, and secondly to reconstruct an estimate of occluded background in each frame using information from surrounding frames. Our background estimation technique is based heavily on the approach of Wang and Adelson [5], which uses optical flow and affine motion models to identify portions of the image that are undergoing similar global motions. Regions in an image with common affine motion are grouped together to form a *layer*. In this way an image can be segmented into multiple layers, one of which will be considered as background. The affine motion used corresponding to the background layer in the segmentation stage is then used to warp background from one frame to another frame, providing an estimate of background pixels that are occluded in one frame but not in another.

2.2 Bayesian Matting

The *natural image matting* problem, i.e. identifying an alpha-matte and a foreground in front of a natural background, is generally explained in terms of the compositing equation, introduced by Porter and Duff [4]

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

In other words, the original image, C, is considered to be a *composite* of a foreground image, F and a background image, B. The ratio of foreground image to background image in a particular pixel is regulated by alpha, α, whose value is between zero and one.

Generally, the matting problem is identified as having seven unknowns (i.e. $\alpha, F_r.F_g, F_b, B_r, B_g, B_b$) and three constraints. However, assuming that we already have the value of the background (obtained from the background estimation), the compositing equation is left with four unknowns (i.e. $\alpha, F_r.F_g, F_b$) and 3 constraints.

The techniques used for solving the compositing equation show differences mainly on the *a priori* assumptions made about the foreground and alpha. We will focus on a bayesian framework that assumes a gaussian mixture model for the foreground pixels and a beta-distribution for alpha.

3 Framework

3.1 Background Estimation

The first stage in background estimation is the segmentation of each frame into *layers*, using motion analysis to identify regions in the image that can be described by common affine motions. One of these layers will be considered the background layer, while the rest will be considered foreground, giving us an image *trimap*. The second stage involves background reconstruction, where we use information from surrounding frames to estimate occluded background in each frame.

3.1.1 Layer Segmentation

It is assumed that regions undergoing a common affine motion belong to the same rigid object or layer. Affine motion is defined as follows:

$$A_x(x,y) = a_{x0} + a_{xx}x + a_{xy}y \quad (2)$$

$$A_y(x,y) = a_{y0} + a_{yx}x + a_{yy}y \quad (3)$$

where x and y are the pixel coordinates. $A_x(x,y)$ and $A_y(x,y)$ are thus the reconstructed velocity estimates at pixel x, y based on the affine motion parameters. We will define $a_x = [a_{x0}\ a_{xx}\ a_{xy}]$ and similarily $a_y = [a_{y0}\ a_{yx}\ a_{yy}]$

To determine the affine motion that best describes a common motion for a region P_i, we solve for the least squares solution,

$$[a_{x_i}\ a_{y_i}] = [\sum_{P_i} \phi\phi^T]^{-1} \sum_{P_i} (\phi[V_x(x,y)\ V_y(x,y)]) \quad (4)$$

where $\phi^T = [1\ x\ y]$, the summation is applied across all pixel coordinates in the region P_i and $V_x(x,y)$ and $V_y(x,y)$ describe the optical flow vector of the pixel at x, y. It is assumed that the optical flow can be accurately calculated.

By subdividing the image into non-overlapping NxN regions and solving equation (4) for each of these regions, we can come up with a set of local affine motions. A k-means clustering approach is then used to group common affine motions and determine which regions in the image can be closely described by the same affine motion. Each distinct motion corresponds to a layer, and in this way we can segment the image into a background layer and one or more foreground layers.

3.1.2 Background Reconstruction

After segmenting each frame based on motion, we will have a defined subset of pixels and an affine motion corresponding to the background layer for each frame. Applying the appropriate affine motion to a given frame gives us a warped version of the

background that fits onto to the background in the ensuing frame. Applying subsequent affine motions allows us to warp the background from one frame to any other frame. In order to find the pixel coordinates x_{i+n}, y_{i+n} in frame $i+n$ corresponding to coordinates x_i, y_i in frame i, we apply the following transformation:

$$x_{i+n} = x_i + \sum_{j=0}^{n-1} [1 \; x_{i+j} \; y_{i+j}] a_{x_{i+j}} \tag{5}$$

$$y_{i+n} = y_i + \sum_{j=0}^{n-1} [1 \; x_{i+j} \; y_{i+j}] a_{y_{i+j}} \tag{6}$$

where a_{x_i} and a_{y_i} are the affine motion paramaters associated with the background layer in frame i. This warping technique allows us to estimate portions of the background that are occluded by foreground objects in one frame but not in another.

3.2 Bayesian Video Matting

[1] Once a reasonable estimate of the background and the foreground is obtained from the image segmentation, it is possible to develop a bayesian framework that seeks the optimal foreground and alpha-matte values subject to regularization constraints. In other words, the maximum *a posteriori* (MAP) value of the foreground image, F, and the alpha-matte, α, given the original composite image, C, and the estimate background image, B, will be the values assigned to unknown pixels in the coarse trimap.

$$\{F, \alpha\} = \arg_{\mathrm{F},\alpha} \max P(F, \alpha | C, B) \tag{7}$$

Bayes' rule allows us to express the *posterior* probability in terms of prior and conditional probabilities.

$$P(F, \alpha | C, B) = \frac{P(C, B | F, \alpha) P(F) P(\alpha)}{P(C) P(B)} \tag{8}$$

Furthermore, by taking the negative log of the posterior and eliminating the terms that do not depend on F and α (i.e. $P(C)$ and $P(B)$), the problem can be expressed as an energy minimization.

$$\{F, \alpha\} = \arg_{\mathrm{F},\alpha} \min \{L(C, B | F, \alpha) + L(F) + L(\alpha)\} \tag{9}$$

where each of the three terms is explained in detail in the following sections.

[1] The bayesian framework is based on [1]

3.2.1 Reconstruction Likelihood

The term $L(C, B|F, \alpha)$ is referred to as the reconstruction likelihood, which mainly expresses the likelihood of observing a certain composite image and background given the true foreground and alpha-matte value. In other words, it models the noise on the observation. We will follow the general assumption that the composite image is affected by additive gaussian noise with variance σ^2. Assuming that the gaussian distribution is independent for every pixel and every component in the RGB space, we can easily compute the reconstruction likelihood as the sum of per-pixel reconstruction errors.

$$L(C, B|F, \alpha) = \sum_{\mathbf{x}} \frac{1}{2\sigma^2} ||C(\mathbf{x}) - \alpha(\mathbf{x})F(\mathbf{x}) - (1 - \alpha(\mathbf{x}))B(\mathbf{x})||^2 \tag{10}$$

where $\mathbf{x}$ represents a pixel position (x, y, t) (or (x, y)).

3.2.2 Foreground Likelihood

The foreground likelihood, $L(F)$, is based on the assumption that foreground pixels in a square neighborhood follow a Gaussian mixture model (GMM) in the RGB color space similar to the one found in [2]. Hence, to compute such likelihood we first need to estimate the GMM distribution for each pixel with the foreground pixels in its neighborhood, leading to a mean vector, μ_k, and a covariance matrix, σ_k, for each identified gaussian k. The full reconstruction likelihood of the foreground is given by:

$$L(F) = -\sum_{\mathbf{x}} \log(\sum_{k}^{N_k} G(F(\mathbf{x}); \Sigma_k, \mu_k) \tag{11}$$

where each per-cluster Gaussian distribution is

$$G(X; \Sigma, \mu) = \frac{1}{\sqrt{(2\pi)^3|\Sigma|}} \exp(-(X - \mu)^T \Sigma^{-1} (X - \mu)) \tag{12}$$

with μ representing the mean of the gaussian and Σ the covariance of the gaussian.

In other words, to compute the foreground likelihood for a particular pixel $\mathbf{x}$, one proceeds as follows. First, we identify the foreground pixels found in a square neighborhood centered on $\mathbf{x}$. Then, we partition the pixels into clusters based on their RGB values. Each pixel is assigned a weight, w_i, that gives importance to pixels closer to $\mathbf{x}$ and with a higher alpha-matte. Specifically, we use a gaussian kernel, g, to model the spatial coherence assumption. The final weight for each pixel

is the product of the gaussian kernel at the pixel position and the squared value of the alpha-matte, $w_i = g_i \alpha_i^2$.

Finally, the weighted mean and covariance of each cluster C_k is computed with its associated pixels.

$$\mu_k = \frac{1}{W} \sum_{i \in C_k} w_i F_i \tag{13}$$

$$\Sigma_k = \frac{1}{W} \sum_{i \in C_k} w_i (F_i - \mu_k)(F_i - \mu_k)^T \tag{14}$$

where $W = \sum_{i \in C_k} w_i$.

3.2.3 Alpha Prior

The bayesian approach allows us to incorporate additional priors biasing the results towards values that follow such prior distribution. Intuitively, it is easy to show that the alpha-matte has a non-uniform distribution. For example, notice that the alpha-matte will generally tend toward zero or one values, since the only regions that require alpha-matte values between zero and one are found in the transitions between foreground and background regions, and they represent a small portion of the full image. In other words, one can assume that the alpha-matte prior distribution should be U-shaped with peaks in zero and one. Specifically, we will make use of the Beta Distribution which has the following density:

$$p(\alpha) = \frac{\alpha^{\eta-1}(1-\alpha)^{\tau-1}}{\beta(\eta, \tau)} \tag{15}$$

The negative log likelihood of the beta distribution is expressed as

$$L(\alpha) = (1-\eta)\log(\alpha) + (1-\tau)\log(1-\alpha) + K_\beta \tag{16}$$

where K_b is a constant and can therefore be ignored in the minimization framework. Several authors have shown that typical values for the beta distribution parameters are $\eta = \tau = 0.25$.

3.2.4 Energy Minimization

As stated above, the MAP search becomes an energy minimization when taking the negative log of the posterior. The complete expression of the energy per pixel $\mathbf{x}$ that should be minimized is shown below.

$$
\begin{aligned}
E_{\mathbf{x}} &= \sum_{\mathbf{x}} \frac{1}{2\sigma^2} ||C(\mathbf{x}) - \alpha(\mathbf{x})F(\mathbf{x}) - (1-\alpha(\mathbf{x}))B(\mathbf{x})||^2 \\
&\quad -\lambda_1 \sum_{\mathbf{x}} \log(\sum_{k}^{N_k} G(F(\mathbf{x}); \Sigma_k, \mu_k) \\
&\quad \lambda_2 \left[(1-\eta)\log(\alpha) + (1-\tau)\log(1-\alpha)\right]
\end{aligned}
$$

The analytic expression for the gradient and the hessian are straightforward to compute. Therefore, we can use any readily available optimizer to minimize the cost function. In this work, we make use of a constrained nonlinear optimizer based on the trust-region variant of the Newton Method, which is readily available in MatLab.

4 Algorithm

The complete algorithm is easily divided into two separate modules. The first module takes as input the full video sequence and computes the coarse trimap and estimated background image for each image in the sequence. The second module takes as input the original video sequence, the coarse trimap and the estimated background image and computes the unknown alpha-matte and foreground pixel values.

4.1 Layer Segmentation

The algorithm for layer segmentation is based on [5]. The image is first subdivided into non-overlapping NxN pixel regions. In general, a value of $N = 5$ was used. A set of 6 affine motion parameters, a_i, is calculated for each region, P_i, using the least squares solution shown in (4).

Regions that do not have a coherent motion across most pixels in the region are identified by calculating the residual error, σ_i, and comparing to a residual error threshold:

$$\sigma_i^2 = \frac{1}{N^2} \sum_{P_i} (V(x,y) - A_i(x,y))^2 \tag{17}$$

where the summation is performed across all pixels in P_i, $V(x,y)$ is the actual optical flow at x, y and $A_i(x,y)$ is the reconstructed flow based on the least squares solution of affine motion paremeters, a_i, for region P_i:

$$A_{x_i}(x,y) = a_{x0_i} + a_{xx_i}x + a_{xy_i}y \tag{18}$$

$$A_{y_i}(x,y) = a_{y0_i} + a_{yx_i}x + a_{yy_i}y \tag{19}$$

Regions with a residual error above the threshold are assumed to cross layer boundaries, and as such cannot be well approximated by a single motion model. These boundary regions are not used in the first stage of clustering.

The affine motion coefficients, a_i, for each of the NxN regions that meet the residual error threshold serve as input vectors (each of length 6) to a k-means clustering algorithm. For the initial frame, the initial k cluster centroids are chosen randomly from among the a_i. For subsequent frames, the k initial centroids can be initialized to the final k affine motion parameters from the previous frame. In [5], k did not need to be specified, but for simplicity we used a fixed k chosen manually depending on the image sequence.

The original algorithm used an affine distance metric for measuring similarity between two affine motions. We have instead used a modified distance metric based on the squared difference between the reconstructed motion vector at the midpoint of the region when using the least-squares affine solution for the region, a_i, and when using the affine motion parameters of the kth centroid, a_k:

$$D_v(a_i, a_k) = (A_i(\bar{x}_i, \bar{y}_i) - A_k(\bar{x}_i, \bar{y}_i))^2 \tag{20}$$

where $\bar{x}_i$ and $\bar{y}_i$ are the coordinates at the midpoint of region P_i and A_i is the velocity at the midpoint of the i^{th} NxN region as reconstructed by a_i. This approach gave better results on the sequences that we considered.

Once each of the a_i has been assigned to one of the k clusters, the cluster centroids are recomputed by taking the least squares estimates of affine motion parameters based on the A_i calculated at the midpoints of regions assigned to each cluster:

$$[a_{x_k}\ a_{y_k}] = [\sum_{C_k} \phi\phi^T]^{-1} \sum_{C_k} (\phi[A_{x_i}(\bar{x}_i, \bar{y}_i)\ A_{y_i}(\bar{x}_i, \bar{y}_i)]) \tag{21}$$

where the summation is applied across each of the NxN regions in cluster C_k.

This method of computing centroids also differs from [5], where a direct mean of the a_k in a given cluster was used. Although slightly more computationally expensive, the least squares method gave better segmentation for the sequences tested.

Assignment to clusters and recomputation of cluster centroids continues until such time that the algorithm converges, namely when no P_i are reassigned from one iteration to the next.

Once we have determined the k affine motions based on the clustering of affine motions of NxN regions, we fine-tune the clustering on a pixel by pixel basis. Our distance metric for assigning a pixel at x, y to one of the a_k is:

$$E_{v_k}(x, y) = (V(x, y) - A_k(x, y))^2 \tag{22}$$

where $V(x, y)$ is the actual optical flow for pixel x, y. This pixel by pixel clustering also allows us to incorporate pixels from regions on layer boundaries that did not meet the residual error threshold and therefore were not used in the original NxN region clustering stage. The pixel by pixel clustering iterates in the same manner as the regional clustering stage, except that actual optical flow at each pixel is used for computing least squares estimates for centroids, as in (4).

The final output of the clustering procedure is k layers, each described by a set of affine motion parameters, a_k, as well as a set of pixels assigned to that layer. Pixels whose motion can not be described by one the k affine motions with sufficient accuracy are not assigned to any of the layers and remain as *unknown*. We used a threshold for E_{v_k} of 1 in order for a pixel to be assigned to a cluster.

As the results of the k-means clustering algorithm are sensitive to the choice of initial centroids, we repeat the clustering procedure 3 times (with different random initial centroids) for the first frame, and keep only the results that gives us the minimum error at convergence, where error is measured as the sum of minimum distances, D_v, for all regions P_i. Subsequent frames use the a_k computed in the previous frame as the initial centroids for the clustering stage for that frame. As these are generally very good initial estimates, we need only run the clustering algorithm once.

4.2 Background Reconstruction

Not many details are provided in [5] on the background reconstruction, other than that for unknown background pixels in a given frame, the median value of the corresponding pixels in all warped frames is used for reconstruction:

$$BG_i(x, y) = median\{W_{i,1}(x, y), W_{i,2}(x, y), ..., W_{i,n}(x, y)\} \tag{23}$$

where BG_i is the background for frame i and $W_{i,m}$ is the m^{th} frame warped to match the i^{th} frame. Only frames where $W_{i,m}(x, y)$ corresponds to background in frame m will be used in the median operation.

Applying an affine motion to a frame is a *forward* operation, in the sense that it is easy to determine what pixel coordinates in frame $i + 1$ correspond to coordinates x, y in frame i:

$$x_{i+1} = x + [1\ x\ y]a_{x_i} \tag{24}$$

$$y_{i+1} = y + [1\ x\ y]a_{y_i} \tag{25}$$

The corresponding coordinates in frame $i + n$ can be found in a similar manner, applying succesive affine transformations, as in (6). In general, x_{i+n} and y_{i+n} will be

non-integer and as such an interpolation is required (we used linear interpolation) to get the corresponding RGB pixel value.

The forward nature of the application of the affine motion makes warping an earlier frame to a later frame a more complicated operation. In order to determine what pixel coordinates in frame $i-1$ correspond to pixels x, y in the current frame, we must solve for x_{i-1} and y_{i-1} from the following set of equations:

$$x = x_{i-1} + [1\ x_{i-1}\ y_{i-1}]a_{x_{i-1}} \tag{26}$$

$$y = y_{i-1} + [1\ x_{i-1}\ y_{i-1}]a_{y_{i-1}} \tag{27}$$

For the general case of warping frame $i-n$ to frame i, instead of trying to solve for x_{i-n} and y_{i-n}, we chose to warp the entirety of frame $i-n$ to frame i using forward affine transformations , as in (25). We then map the resultant non-integer (and non-uniform) pixel coordinates by interpolating on to the original uniform pixel grid.

4.3 MAP Estimate

As mentioned before, the first module provides the coarse trimap and the estimated warped background for each frame. The second module will then solve the MAP estimate of each pixel found in the unknown region specified in the trimap. The MAP estimate is readily obtained by minimizing the corresponding cost function (i.e. the negative log likelihood).

To actually compute the value of the cost function it is necessary to estimate the GMM distribution of the foreground pixels, the details of how to obtain such an estimation are found in the following section. The arguments that minimize the cost function are obtained with a trust-region Newton optimizer .

5 Implementation

5.1 Background Estimation

5.1.1 Layer Segmentation

In [5], the authors suggest that one does not need to specify the number of clusters, k, to the clustering algorithm. We decided to manually specify k and only use sequences where the number of layers did not change over the course of the sequence.

The combination of distance metrics and method for computing centroids that were used in the k-means clustering algorithm gave better performance than those used in [5]. While using these metrics always resulted in convergence for the sequences

tested, it is suspected that these new metrics results in a clustering algorithm that does not guarantee convergence.

The layer segmentation algorithm partitions the frame into k layers. However, for the purposes of creating a trimap, we are interested only in a background layer and a foreground layer. We assume that the layer with the most pixels is background, and we group all other layers as one foreground layer.

In order to guard against any errors at layer boundaries and to ensure that our trimap could be fully *trusted*, we added a buffer of 2 pixels labelled as unknown on foreground to background transitions. Since we have a full estimate of the background underneath these *unknown* pixels, the matting process should in theory be able to deduce the correct alpha-matte for these buffers.

5.1.2 Background Reconstruction

One issue with warping of frames was interpolation *leakage*. Non-background pixels in the neighbourhood of background pixels were incorporated during the interpolation process. In order to safeguard against this, an additional buffer of unknown pixels was used around background pixels for frames that were to be warped.

In some cases we were not able to fully reconstruct the background for a given frame. This caused problems when missing background occured in pixels that were not labelled as foreground (i.e. labelled as unknown) in the trimap. In these cases, the nearest known background pixel was used as the background estimate for that pixel.

5.2 Bayesian Matting

The square neighborhood was chosen to be of size $N = 19\text{x}19$, the standard deviation of the gaussian kernel was chosen to be 8, and all λs were set to 1. In addition, it was of interest to provide good initial estimates of α and F, as to reduce the number of iterations the optimizer required to reach the minima. The initial estimate of α was chosen to be the either the mean value of α in a 4x4 neighborhood or the ratio of foreground pixels to background pixels in the 19x19 square neighborhood. On the other hand, the initial estimate of the foreground was chosen to be the weighted mean of the pixels with non zero αs in the square neighborhood.

The function that provides the likelihood value of specific F and α needs to estimate the gaussian mixture model of the foreground pixels found in the neighborhood before computing the likelihood and its respective gradient and hessian. The foreground pixels are identified as all the pixels in the most recent foreground estimation that have a respective alpha value higher than zero and are therefore no longer part of the unknown region or a pixel that was part of region identified as a certain foreground in the initial coarse trimap. The identified foreground pixels are

then partitioned into three clusters with a k-means algorithm. Finally, the sample weighted mean and covariance of each cluster is computed and the likelihood value with its respective gradient and hessian can be computed.

6 Results

To simplify our matting problem, synthetic image sequences from the optical flow benchmark project [3] were used. These sequences had the added advantage of having ground truth optical flow, which allowed us to eliminate any error caused by our optical flow estimations. Maxime's Matlab script was used to extract the ground truth optical flow from the binary benchmark data. While it may be unrealistic to expect perfect or near-perfect optical flow estimates in a real world scenario, the assumption that optical flow could be accurately calculated allowed us to tackle a more manageable problem. We had every intention of using optical flow that we ourselves computed as well as using real world video examples, but long computation times, a lengthy debugging process and time constraints limited our experiments with full matting of entire sequences to synthetic sequences with optical flow given.

Most of our experimentation used the simple_medium_medium sequence, which is the same as the medium sequence used in assignment 3 (except we used a colour version). This sequence had fairly simple foreground motion, with a car and satellite moving independently. The sequence also had a fairly strong rotational camera motion so that the background was not stationary.

The layer segmentation was remarkably accurate, except right at the boundaries between foreground objects and background. We used $k = 3$ and the algorithm could separate the car, satellite and background. Because of the buffering used at transitions between foreground and background, the tail of the satellite was always labelled as unknown, which made accurate matting of this portion difficult. A movie showing the clustering process and segmentation of the first 5 frames of a sequence can be found on our results website[2].

The background reconstruction process also worked very well when using ground truth optical flow. In the simple_medium_medium sequence, the satellite moves almost with the camera, meaning that some portions of the background are never non-occluded, such that a full background estimate can not be reconstructed. The background reconstruction process for frame 1 of the sequence is shown on our results website. The left image shows the frame 1 background being updated with new information from subsequent frames that are being warped to frame 1, while the right image shows the backgrounds from those subsequent frames in turn being warped to frame 1. Around the edges of the remaining unknown background pixels,

[2] http://www.cim.mcgill.ca/~dante/videomatting.html

we can see some *leakage* of foreground pixels from other frames, but in general the results are good.

The overall background estimation process for the sequence is shown on our results website. Here we can see the trimap and reconstructed background for every frame in the sequence, which serve as inputs to the bayesian matting phase.

Figure 1 shows an example of the results obtained when solving for the unknowns in the coarse trimap with the framework proposed in 3.2. The coarse trimap shows the initial segmentation, where the black region corresponds to background, the white region corresponds to foreground, and the gray region corresponds to the unknowns. In other words, the energy minimizer only resolves the F and α values of the gray regions.

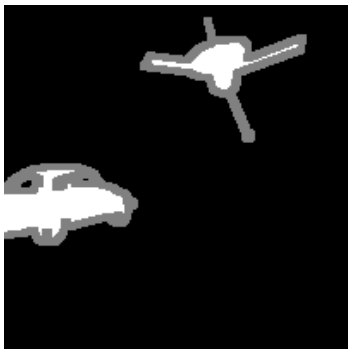
It is of interest to note that even though the satellite's antennas were not in the regions identified as foreground in the coarse trimap segmentation, it was still possible to recover such features with the use of the bayesian framework. We can also see that the algorithm was able to obtain all the features of the car without including background pixel colors.

On the other hand, the algorithm had some difficulties resolving some of the satellite features. In particular, we can see that the alpha-matte value of the solar panels tend to have values inferior to one, while our intuition would tell us that their alpha value is close to one. We can try and explain such complications by the nature of both the foreground and background pixels. For example, we can observe that the solar panel foreground pixels and their respective background pixels share a number of light grey and blue colors. Hence, we can expect that the reconstruction error of pixels with similar foreground and background colors will be small. Furthermore, since there is considerable variance in the neighborhood (the solar panels exhibit a bit of texture), we can also expect the foreground energy to be relatively small. Indeed, intuition tells us that the panels don't have any kind of transparency since we have a physical model in our mind when we look at the image. However, if we ignored the physical model in our mind, we could easily imagine the possibility of transparency in the panels.

The phenomenon explained in the previous paragraph is easily spotted as *speckle* in the alpha-matte image. If we assume that the alpha-matte values expected should be smooth and spatially continuous, we could easily incorporate additional smoothing techniques or other regularizing constraints that minimize the *speckle*.

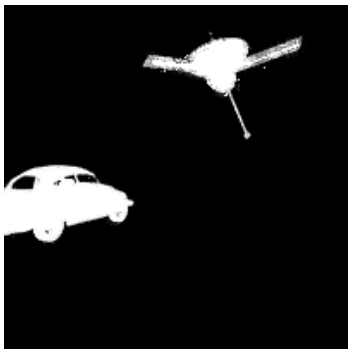
A video of the alpha-matted foreground superimposed on a new background for the image sequence is shown on our results website. Also shown is a movie showing the different stages of the entire matting process.

(a) Original Image



(b) Coarse trimap

(c) Image with New Background



(d) Alpha-matte

Figure 1: *Results for a frame of sample video sequence. 1(a) shows the image in the original frame, 1(b) shows the coarse trimap obtained from motion-based layer segmenation, 1(c) shows the αF imposed over a different background, 1(d) shows the alpha-matte obtained from the bayesian framework*

7 Discussion

We can see that given true optical flow, one can very accurately segment an image sequence into layers and reconstruct occluded background for each frame, at least for relatively simple motions and with a small number of layers. Even under these ideal conditions, there are several areas where the algorithm could be improved, namely:

1. Estimation of background in portions that are occluded (or semi-occluded) in all frames
2. Better characterization of the buffer required to ensure an accurate trimap
3. Better characterization of the buffer required to ensure warped background is truly background
4. More efficient algorithms for reconstructing background, especially when warping an earlier frame to a later frame

Solving (2) and (3) would go a long way to solving (1) as they would make the trimap more precise and reduce the occurence of actual background pixels being labelled as unknown. For remaining background pixels that cannot be reconstructed via warping, a more advanced image reconstruction using neighbouring pixels could be used. For (4), more effort is required to determine a way to interpolate only the subset of pixels that are needed for warping as opposed to interpolating an entire frame, as was done here for warping an earlier frame to a later frame.

As our experimentation was limited, we can not draw many conclusions about how the segmentation (and overall matting) would perform if used on real video, but we can assume that several factors would inhibit the performance of our segmentation approach, some of which are:

1. Innacurate optical flow estimates
2. Motions that cannot be modelled as affine
3. Foreground objects that are sometimes moving and sometimes stationary
4. Sequences where k changes frequently
5. Background with high contrast and detail

One could develop heuristics to deal with (3) and more robust clustering and heuristics to deal with (4). Complicated background would limit the efficacy of the

background warping in cases where the affine motion was not exactly correct, especially if source and target frames were far from each other. In cases where (2) holds, a more complicated motion model would need to be used. If using estimated optical flow, some heuristics and smoothing could compensate for errors in optical flow calculation, and this uncertainty would have to be incorporated as *coarser* trimap, making the matting process more difficult. Manual specification of k could also help in identifying trusted optical flow estimates.

An obvious extension to the background estimation process would be to make a multi-layer trimap, where different foreground objects could be considered seperately.

We have seen that the overall matting obtained with the proposed framework is reasonably good, when the trimap and background detection can be done accurately. In particular, the alpha-mattes obtained portrayed a detailed description of the foreground object even though it showed a few artifacts shown as speckle. Smoothing techniques in the spatial domain are an interesting possibility to study as to guarantee a continuous alpha-matte. Additional priors can also be considered to obtain better quality alpha-mattes. For example, Wexler et al [6] proposed a spatiotemporal consistency prior which smoothes alpha along, but not across, edges in space and time, and incorporated it in the bayesian framework.

References

[1] N Apostoloff and A Fitzgibbon. Bayesian video matting using learnt image priors. *Computer Vision and Pattern Recognition*, 2004.

[2] Y Chuang, B Curless, D Salesin, and R Szeliski. A bayesian approach to digital matting. *IEEE Computer Society Conference on Computer Vision and ...*, Jan 2001.

[3] B McCane, K Novins, D Crannitch, and B Galvin. On benchmarking optical flow. *Computer Vision and Image Understanding*, 2001.

[4] Thomas Porter and Tom Duff. Compositing digital images. *SIGGRAPH Comput. Graph.*, 18(3):253–259, 1984.

[5] J Wang and E Adelson. Representing moving images with layers. *Image Processing*, Jan 1994.

[6] Y Wexler, A Fitzgibbon, and A Zisserman. Bayesian estimation of layers from multiple images. *Lecture Notes in Computer Science*, Jan 2002.