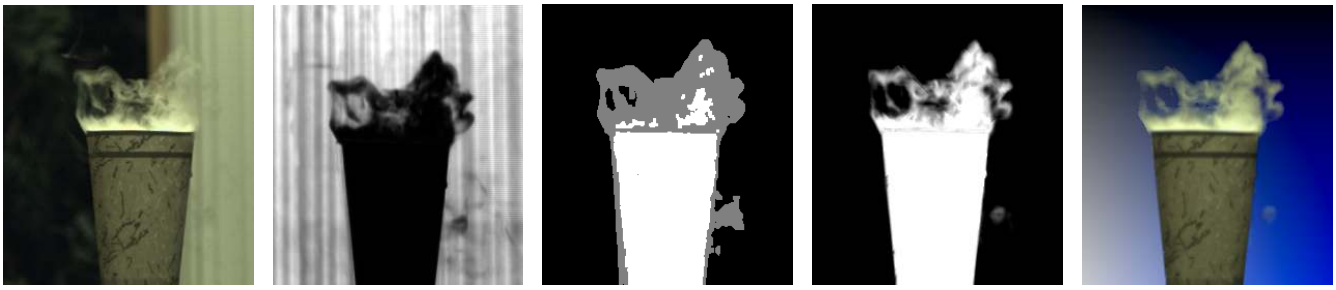
Natural Video Matting using Camera Arrays

Neel Joshi *
University of California, San Diego

Wojciech Matusik
MERL

Shai Avidan
MERL



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

Figure 1: Natural video matting. (a) We use a camera array to capture a collection of images of a scene. Here we show a single camera's image. (b) We synthetically refocus the data and compute the variance of the refocused images (darker means lower variance). (c) From the "variance image" we automatically compute a trimap. (d) We propagate the variances into the unknown region of the trimap and use these measurements to solve for the alpha matte. (e) We then compute the alpha multiplied foreground and composite it with a new background.

Abstract

We present an algorithm and a system for high-quality natural video matting using a camera array. The system uses high frequencies present in natural scenes to compute mattes by creating a synthetic aperture image that is focused on the foreground object, which reduces the variance of pixels reprojected from the foreground while increasing the variance of pixels reprojected from the background. We modify the standard matting equation to work directly with variance measurements and show how these statistics can be used to construct a trimap that is later upgraded to an alpha matte. The entire process is completely automatic, including an automatic method for focusing the synthetic aperture image on the foreground object and an automatic method to compute the trimap and the alpha matte. The proposed algorithm is very efficient and has a per-pixel running time that is linear in the number of cameras. Our current system runs at several frames per second, and we believe that it is the first system capable of computing high-quality alpha mattes at near real-time rates without the use of active illumination or special backgrounds.

Keywords: Alpha Matting and Compositing, Light Fields, Image-Based Rendering

1 Introduction

Efficient and high-quality compositing is an important task in the special effects industry. Typically, movie scenes are composited from two different layers (foreground and background), where each of these layers can be computer-generated or may be from real footage filmed at different locations. To use the foreground content of one sequence as the foreground layer in a composite video, the foreground elements must be separated from the background in the source video. This process, known as matting, separates a foreground element from the background by estimating a color F and an opacity α for each foreground pixel. With a single image, this problem is highly underconstrained and can be posed as an equation in seven unknowns (α, RGB foreground F, and RGB background B) and three measurements (the RGB video frame I) at each pixel:

$$I = \alpha F + (1-\alpha)B. \quad (1)$$

The most widely used matting method, blue-screen matting, constrains the problem by filming actors in front of a known (blue or green) background. While this simplifies the problem, the method has significant limitations in that it can only be used in a movie studio or a similarly controlled environment.

Ideally, one would like to compute an alpha matte from a regular video stream taken in a natural, uncontrolled environment – a process known as natural video matting. Recently, there has been significant progress in this area; however, many current methods require user input, have potentially lengthy run-times, and have difficultly with highly textured scenes.

In this paper, we present an algorithm that uses a camera array to compute mattes for natural scenes with textured backgrounds. Relative parallax in the array images, due to separation between the foreground and background, allows us to capture foreground objects in front of different parts of the background. Given a sufficiently textured background, we capture the foreground object in front of several background colors, which constrains the matting problem. We project the color values from each camera to the depth of the foreground object and use mean and variance statistics computed from these values to automatically compute a trimap and subsequently α and F. We perform this projection using synthetic refocusing [Isaksen et al. 2000] and sweep across depths near the foreground object to optimize the focus per pixel; as a result, we can generate mattes for non-planar objects. We compute alpha mattes using a novel matting equation that works with pixel variance, instead of working directly with pixel values. The result is a fast and automatic algorithm that avoids the difficult problems of computing the background depth and reconstructing the 3D scene and, as a result, works with arbitrarily complex background scenes. An additional benefit of our algorithm is that the per-pixel running time is proportional to the number of cameras. This makes the algorithm

*email: njoshi@cs.ucsd.edu

extremely efficient and amenable to real-time performance, and it is easily implemented on either a CPU or GPU.

The main contributions of our paper are the following: extending the matting equation to deal with the variance of pixel measurements, using a camera array for alpha matting, presenting an automatic method for computing a trimap and alpha matte, and constructing a real-time system for natural video matting. We show results for a large range of objects (people, hair, trees, fluids, glass, and smoke) captured against a variety of backgrounds (office environments and trees) using our camera array.

In the next section, we will discuss some of the previous work in this area. In section 3, we will present our alpha matting algorithm. In section 4, we will present our alpha matting system. Lastly, we present results for static and dynamic scenes in section 5, followed by a discussion of our method and our conclusions.

2 Previous Work

The problem of alpha matting has been researched for almost half a century. The first matting algorithms and systems can be attributed to Vlahos [1958; 1971; 1978]. Blue-screen matting was mathematically formalized by Smith and Blinn [1996], who also showed that imaging a foreground against two different backgrounds gives a robust solution for both the alpha and the foreground color. Their method has been extended to work with more complex light transport effects (e.g., refraction) by Zongker *et al.* [1999] and Chuang *et al.* [2000]. However, these methods require active illumination and multiple images.

Recently, there has been significant work in natural image matting. Bayesian Matting [Chuang et al. 2001], which is based on the method of Ruzon and Tomasi [2000], uses a user specified trimap and a probabilistic model of foreground and background colors to compute a matte. This method has been extended to video [Chuang et al. 2002]; however, trimaps still need to be specified manually for keyframes. In a further extension, Zitnick *et al.* [2004] use a multi-camera system to reconstruct 3D scene geometry and use Bayesian Matting to compute alpha mattes at depth discontinuities. While their depth computation is automatic, their system is not real-time – it requires off-line processing to compute both the depth and alpha mattes. Our system, in contrast, is not dependent on scene reconstruction and therefore can handle structurally complex scenes, and it runs at 3 to 5 frames per second. Furthermore, as researchers have noted [Li et al. 2005], Bayesian Matting fails for highly textured areas where local spatial coherence or smoothness assumptions are violated. By contrast, our method makes no assumptions about the foreground, while requiring the background to have at least some texture, and performs very well even when both the foreground and background are highly textured.

Poisson Matting [Sun et al. 2004] addresses some of the issues of color based methods by posing matting as solving Poisson equations of the matte gradient field. This resembles our work in spirit, in that it does not work directly on the image color but on a derived measurement, but it differs from our method in that it works on still images, requires some user input, and takes several minutes to process a single frame.

McGuire *et al.* [2005] compute alpha mattes for natural scenes using three video streams that share a common center of projection but vary in depth of field and focal plane. While their method is automatic, its running time is many minutes per frame. In addition, the foreground object must be in the foreground focus range during shooting, whereas our method can be used to compute mattes for any depth plane after shooting, using synthetic refocusing.

A related area of work is that of object "cut and paste", where mattes are computed after performing binary segmentation with minimal user input [Rother et al. 2004; Li et al. 2004]. These methods have also been extended to video [Wang et al. 2005; Li et al. 2005]. These methods tend to be limited to "border matting", where alpha is computed along the object border after hard segmentation; it's unclear if these methods can compute mattes for semi-transparent objects such as fluids or smoke. Also, as they require user input, the total amount of processing time can still be significant. Kolmogorov *et al.* [2005], on the other hand, describe an automatic real-time system that uses graph cuts on a stereo video to segment the foreground for a video conferencing scenario. However, this work also only computes alpha on object borders.

Our work is most similar to that of Wexler *et al.* [2002], which also uses multiple images to compute an alpha matte. Where our work differs is that Wexler *et al.* pose the problem in a Bayesian framework and consider several different priors, including bounded reconstruction, α distribution, and spatial consistency. These priors are derived from assumptions about alpha matte structure and thus limits the types of objects that can be successfully matted. Wexler *et al.* also assume that the background is mostly planar and that it is known or can be estimated. In contrast, our method does not make assumptions about alpha matte structure or the physical background structure, and we do not explicitly estimate the entire background color layer. Furthermore, Wexler *et al.* do not discuss real-time aspects of their system, while our system runs at near real-time rates.

In recent years, camera arrays have been used for a wide variety of applications in computer graphics and computer vision. For an extensive list of citations and discussion of previous work we encourage the reader to read Wilburn *et al.* [2005].

3 Alpha Matting Algorithm

Our algorithm resembles past approaches in that it computes a trimap that is then upgraded to an alpha matte. However, there are several relevant differences between our work and previous approaches. First, we compute the trimap automatically and do not assume that a user supplies it interactively. Second, our algorithm uses higher order statistics (i.e., variances) of image measurements that we then propagate into the unknown region of the trimap. Previous methods propagate measurements into the unknown region; however, they propagate pixel values directly which makes limiting assumptions about the scene content. Propagating variances is much less limiting, as we will show in the next section. Specifically, our algorithm proceeds as follows:

1. Automatically find the dominant foreground depth plane and then perform a local depth search to account for non-planarity
2. Compute the mean and variance of color values from each camera projected to their corresponding depths
3. Automatically compute a trimap based on variance
4. Propagate the variance from the background and foreground regions to the unknown region
5. Propagate the mean from the background to the unknown region
6. Compute α and αF

The per-pixel running time of the algorithm (including the computation of the variances and the trimap) is linear in the number of cameras in the array. We will first describe the last three steps of our algorithm, as they form the core of our contribution. Discussion of the first three steps of the algorithm is deferred to section 4.

3.1 Motivation

Given n images of a scene, we consider the following matting equation of a given scene point p:

$$I_i(p) = \alpha(p)F_i(p) + (1-\alpha(p))B_i(p), \tag{2}$$

where $I_i(p)$ corresponds to the intensity of point p recorded in image i. $F_i(p)$ and $B_i(p)$ are the foreground and background values that, as a function of the transparency $\alpha(p)$, are mixed to give $I_i(p)$. We will drop the notation p wherever possible to improve readability. Specifying a different F_i for every image means that we allow

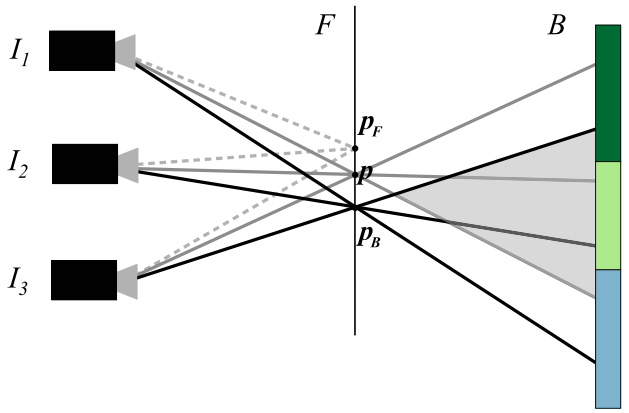


Figure 2: Propagating image statistics. The three cameras on the left $\mathbf{I_1}$,$\mathbf{I_2}$, and $\mathbf{I_3}$ observe a foreground object F and a background object B. The point p has a trimap label of *unknown*, and p_F and p_B are the points nearest to p labeled, respectively, *foreground* and *background*. To solve for α and αF, we need the measurements $\mathbf{var}(\mathbf{I}(p))$, $\mathbf{var}(\mathbf{F}(p))$, $\mathbf{var}(\mathbf{B}(p))$, $\mathbf{mean}(\mathbf{I}(p))$, and $\mathbf{mean}(\mathbf{B}(p))$; however, we can only measure $\mathbf{var}(\mathbf{I}(p))$ and $\mathbf{mean}(\mathbf{I}(p))$. We estimate $\mathbf{var}(\mathbf{F}(p))$, $\mathbf{var}(\mathbf{B}(p))$, and $\mathbf{mean}(\mathbf{B}(p))$ from the measurements at p_F and p_B. We note that the rays going through the points p and p_B hit overlapping regions (indicated by the shaded triangle) on the background and, as a result, the mean and variance of the two points should indeed be similar. In practice, this overlap is quite large. We only assume that the variance for the points p and p_F, *not* their mean, is the same, which is equivalent to saying that they can have different albedo but their view-dependent statistics (e.g., specularity level) are the same.

for view-dependent effects, such as specularity. However, we assume that the transparency of the point is view-independent and hence α is fixed across all images.

We consider $\{I_i(p)\}_{i=1}^n$, $\{F_i(p)\}_{i=1}^n$, and $\{B_i(p)\}_{i=1}^n$ as sampling the random variables $\mathbf{I}, \mathbf{F}$, and $\mathbf{B}$, respectively, and rewrite the matting equation using these variables:

$$\mathbf{I} = \alpha\mathbf{F} + (1-\alpha)\mathbf{B}, \tag{3}$$

We wish to solve for α and αF using these random variables and do this by using second-order moments of $\mathbf{I}, \mathbf{F}$ and $\mathbf{B}$ (i.e., variances) to solve for α and first-order moments (i.e., means) of $\mathbf{I}$ and $\mathbf{B}$ to solve for αF. Note that we do *not* use the mean of $\mathbf{F}$.

Recall that the fourth and fifth steps of our algorithm propagate image measurements from the foreground and background labeled pixels to the unknown pixels. While one could propagate the mean pixel values of the foreground object and solve for an alpha matte using mean statistics alone, this assumes that foreground objects have low spatial frequency albedo, which is a very limiting assumption, whereas propagating the variances allows objects with both low and high spatial frequency albedo. This is an important point and is one of the key components of our algorithm.

Specifically, let p be the scene point under consideration and denote p_F and p_B as the closest points that are labeled as foreground and background, respectively, in the trimap. We make the following approximations:

$$\begin{aligned} \mathbf{var}(\mathbf{F}(p)) &\approx \mathbf{var}(\mathbf{F}(p_F)) \\ \mathbf{var}(\mathbf{B}(p)) &\approx \mathbf{var}(\mathbf{B}(p_B)) \\ \mathbf{mean}(\mathbf{B}(p)) &\approx \mathbf{mean}(\mathbf{B}(p_B)) \end{aligned} \tag{4}$$

These approximations make the two following assumptions. The first and second-order statistics (e.g., mean and variance) of the closest background point p_B are the same as the statistics of the corresponding background colors that scene point p is viewed against. This is a plausible assumption because the rays going from the camera centers through the points p and p_B will hit similar parts of the background, as illustrated in Figure 2. In practice, as the background is significantly far from the foreground object and the distance between p and p_B is small, the ray bundles going through these two points overlap significantly. The second-order statistics of the closest foreground point p_F are the same as the second-order statistics of the scene point p. This is equivalent to stating that view-independent properties (e.g., albedo) of the scene point and its closest foreground point can be completely different but their view-dependent statistics (e.g., specularity level) are the same.

3.2 Mathematical Derivation

We now derive a new variance-based matting equation to solve for α and then F. We first take the variance of equation 3:

$$\mathbf{var}(\mathbf{I}) = \mathbf{var}[\alpha\mathbf{F} + (1-\alpha)\mathbf{B}]. \tag{5}$$

If we assume that $\mathbf{B}$ and $\mathbf{F}$ are statistically independent then:

$$\begin{aligned} \mathbf{var}(\mathbf{I}) &= \mathbf{var}[\alpha\mathbf{F} + (1-\alpha)\mathbf{B}] \\ &= \langle[(\alpha\mathbf{F} + (1-\alpha)\mathbf{B}) - \langle\alpha\mathbf{F} + (1-\alpha)\mathbf{B}\rangle]^2\rangle \\ &= \langle[\alpha(\mathbf{F} - \langle\mathbf{F}\rangle) + (1-\alpha)(\mathbf{B} - \langle\mathbf{B}\rangle)]^2\rangle \\ &= \alpha^2\langle(\mathbf{F} - \langle\mathbf{F}\rangle)^2\rangle + (1-\alpha)^2\langle(\mathbf{B} - \langle\mathbf{B}\rangle)^2\rangle \\ &= \alpha^2\mathbf{var}(\mathbf{F}) + (1-\alpha)^2\mathbf{var}(\mathbf{B}) \end{aligned} \tag{6}$$

where $\langle\mathbf{X}\rangle$ denotes the mean value of $\mathbf{X}$. The assumption that $\mathbf{B}$ and $\mathbf{F}$ are statistically independent is manifested in going from the third to the fourth line of equation 6 where the expected value of term $\alpha(1-\alpha)(\mathbf{F} - \langle\mathbf{F}\rangle)(\mathbf{B} - \langle\mathbf{B}\rangle)$ is assumed to be equal to zero. In order to compute α, we need to solve a quadratic equation in α:

$$[\mathbf{var}(\mathbf{F}) + \mathbf{var}(\mathbf{B})]\alpha^2 - 2\mathbf{var}(\mathbf{B})\alpha + [\mathbf{var}(\mathbf{B}) - \mathbf{var}(\mathbf{I})] = 0. \tag{7}$$

The solutions to this quadratic equation are:

$$\alpha = \frac{\mathbf{var}(\mathbf{B}) \pm \sqrt{\triangle}}{\mathbf{var}(\mathbf{F}) + \mathbf{var}(\mathbf{B})}, \tag{8}$$

where

$$\triangle = \mathbf{var}(\mathbf{I})[\mathbf{var}(\mathbf{F}) + \mathbf{var}(\mathbf{B})] - \mathbf{var}(\mathbf{F})\mathbf{var}(\mathbf{B}). \tag{9}$$

Equation 8 provides two algebraic solutions; however, when $\mathbf{var}(\mathbf{B}) \gg \mathbf{var}(\mathbf{F})$, as is the case in practice, then one of the solutions is often greater than 1 making it inconsistent with the constraint that $\alpha \in [0,1]$. In case both solutions are valid and consistent, we take their average. Algebraic analysis reveals all possible cases and can be summarized as follows:

$$\alpha = \begin{cases} 0 & \mathbf{var}(\mathbf{I}) > \max(\mathbf{var}(\mathbf{B}), \mathbf{var}(\mathbf{F})); \\ \frac{\mathbf{var}(\mathbf{B}) + \sqrt{\triangle}}{\mathbf{var}(\mathbf{B}) + \mathbf{var}(\mathbf{F})} & \mathbf{var}(\mathbf{B}) < \mathbf{var}(\mathbf{I}) \le \mathbf{var}(\mathbf{F}); \\ \frac{\mathbf{var}(\mathbf{B}) - \sqrt{\triangle}}{\mathbf{var}(\mathbf{B}) + \mathbf{var}(\mathbf{F})} & \mathbf{var}(\mathbf{F}) < \mathbf{var}(\mathbf{I}) \le \mathbf{var}(\mathbf{B}); \\ \frac{\mathbf{var}(\mathbf{B})}{\mathbf{var}(\mathbf{B}) + \mathbf{var}(\mathbf{F})} & \frac{\mathbf{var}(\mathbf{B})\mathbf{var}(\mathbf{F})}{\mathbf{var}(\mathbf{B}) + \mathbf{var}(\mathbf{F})} \le \mathbf{var}(\mathbf{I}) \le \min(\mathbf{var}(\mathbf{F}), \mathbf{var}(\mathbf{B})); \\ 1 & \mathbf{var}(\mathbf{I}) < \frac{\mathbf{var}(\mathbf{B})\mathbf{var}(\mathbf{F})}{\mathbf{var}(\mathbf{B}) + \mathbf{var}(\mathbf{F})}. \end{cases} \tag{10}$$

The first and last lines of equation 10 represent the cases in which no valid solutions exist, and we clip α appropriately. Lines 2 and 3 represent the case in which one of the solutions is outside the range $[0,1]$, and we can safely choose the other solution. In practice, we only encounter the case shown in line three. Line 4 is when both solutions are valid, and we set α to be their average.

Note that two discontinuities occur. One occurs when the value of $\mathbf{var}(\mathbf{I})$ switches the solution from lines 2 or 3 to line 4, and the second is from line 4 to 5. These are due to switching from an exact solution to an approximate solution. For both solutions the

discontinuity is no more than $\frac{\mathbf{var}(\mathbf{F})}{\mathbf{var}(\mathbf{B})+\mathrm{var}(\mathbf{F})} = \frac{1}{1+c}$, where c is defined as the ratio between background and foreground variances, i.e., $\mathbf{var}(\mathbf{B}) = c\mathbf{var}(\mathbf{F})$. In the worst case $c = 1$ (the background and foreground variances are the same), and the error is 0.5 (i.e., we set α to 0.5 where, in fact, it should have been either 0 or 1). In practice we found that c is on the order of 100, which makes for a roughly 0.01 error in alpha estimation.

Note that if we assume that our scene is diffuse ($\mathbf{var}(\mathbf{F}) = 0$), then equation 8 has no ambiguity:

$$\alpha = 1 - \sqrt{\frac{\mathbf{var}(\mathbf{I})}{\mathbf{var}(\mathbf{B})}}. \tag{11}$$

We compute αF using the following equation:

$$\alpha F = \langle \mathbf{I} \rangle - (1-\alpha)\langle \mathbf{B} \rangle, \tag{12}$$

where $\langle \mathbf{I} \rangle$ is the mean of the corresponding pixel value in all images, α is recovered from equation 10, and $\langle \mathbf{B} \rangle$ is the mean of the background pixel values. To visually preserve view dependent effects, such as highlights, without blurring them in the computed αF, we compute $\langle \mathbf{I} \rangle$ and $\langle \mathbf{B} \rangle$ using a weighted mean with the highest weight placed on the central reference camera with the weight dropping for cameras that are farther away. To summarize, we estimate mean and variance statistics using equation 4 and compute α and αF using equations 10 and 12, and our result is most accurate when $\mathbf{var}(\mathbf{B}) \gg \mathbf{var}(\mathbf{F})$.[1]

4 Alpha Matting System

Our system works with an array of synchronized video cameras that capture the scene of interest. The central camera is defined as the reference camera and our goal is to compute an alpha matte on the reference camera using information from the rest of the cameras. The algorithm requires the mean and variance statistics, as well as the trimap, to be computed at the foreground depth. We will now describe how we compute these values.

4.1 Automatically Selecting the Foreground Depth

To compute the mean and variance statistics, it is necessary to identify corresponding pixels in each camera image for each foreground scene point p. We do this by synthetically refocusing [Isaksen et al. 2000] the array images to the foreground object's depth plane. To automatically pick this plane, we have developed an "autofocus" algorithm. Our method is similar in spirit to autofocus in a consumer camera, where a camera varies its focus and picks the setting that gives the strongest gradients in pre-defined regions (often appearing as red-rectangles in the viewfinder). We instead sweep a synthetic focal plane through the scene, sum the variance at each plane within each of 9 10x10 pixel focus regions, and pick the depth that gives the minimum variance in one of these regions. Low variance at a depth plane implies that a number of features are aligned and thus an object is present; we use 9 focus regions distributed about the image since the foreground object could appear in different parts of the image. We limit the depth range during the sweep such that we do not autofocus on the background. Currently, our autofocus method is "on-demand". We typically run it once at the beginning of a sequence and use this depth for the entire sequence; however, if the foreground object moves significantly off this depth, we run the method again.

We also provide a "manual focus" interface that allows the user to override the automatic focusing when necessary. The user can pick the foreground depth by interactively sliding the synthetic plane of focus through the scene; the synthetic aperture image is displayed in real-time, so that the user can choose the plane where the foreground object is best focused. We have found that this method is a relatively simple and intuitive way to select the foreground depth.

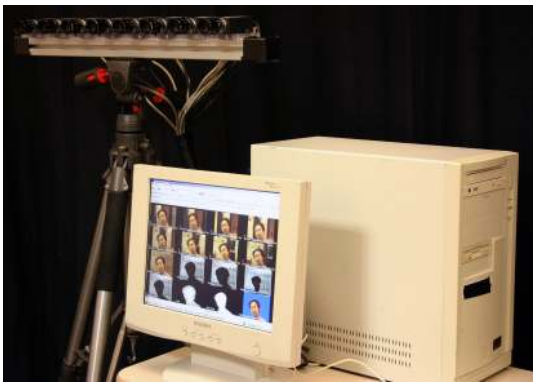


Figure 3: Camera array and real-time system. We use a linear array of 8 video cameras. We can compute alpha mattes at several frames per second at quarter-VGA resolution and generate VGA results offline at about a second per frame.

Once the array images are aligned to a depth plane, features on this plane have minimal variance, while features off this plane, such as the background have high variance, which is the desired effect. However, if the foreground is non-planar, textured areas off the plane will also exhibit higher variance, which could cause erroneous α values. Fortunately, it is relatively simple to handle this non-planarity by adjusting the per-pixel depth. We perform a local search by sweeping a plane over a small depth range near the foreground reference plane and store the minimum variance value and corresponding mean per-pixel over that range. [2] This allows us to automatically adjust the focus slightly on a per-pixel basis.

4.2 Automatic Trimap Computation

As described in section 3.1, since we only observe $\mathbf{I}$, we have to approximate the variance of $\mathbf{F}$ and $\mathbf{B}$ using nearby scene points. For each point labeled as unknown in the trimap, we use the variance of its nearest foreground and background points to estimate $\mathbf{var}(\mathbf{F})$ and $\mathbf{var}(\mathbf{B})$. We then compute alpha using equation 10.

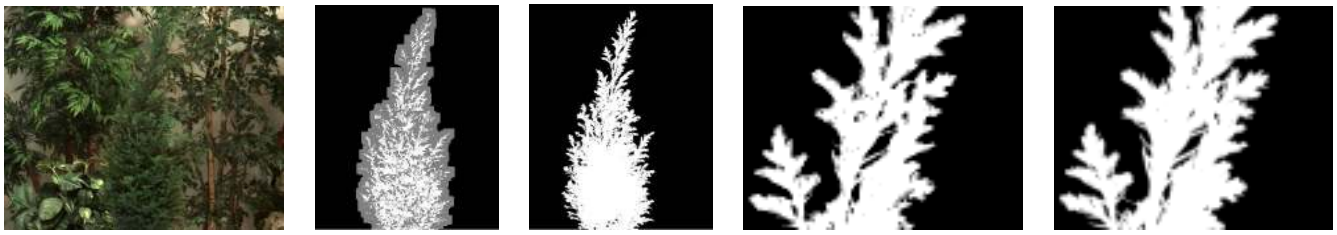
We construct the trimap by filtering $\mathbf{var}(\mathbf{I})$ with a 9x9 pixel median filter to smooth out small fluctuations due to noise. We then use a relatively standard approach of erosion and dilation combined with double-thresholding to create a trimap with conservatively wide unknown regions. Variances less than 100 are considered foreground, greater than 5000 are background, and the rest are unknown. We erode the foreground and background regions to remove small disconnected areas and dilate the unknown pixels inwards and outwards by 5 pixels.

4.3 Implementation

We use a linear array of 8 Basler cameras shown in Figure 3. The resolution of each camera is 640x480 pixels (Bayer pattern). The cameras have external hardware triggers and can provide synchronized video capture up to 30 frames per second. All cameras are connected to one 3GHz PC over Firewire. We geometrically calibrate our camera array (both extrinsics and intrinsics) using standard computer vision methods. We assume that the centers of projection of our cameras lie on a line. Furthermore, we compute homographies that rectify all camera planes. We perform basic color calibration by placing a Macbeth color checker in the scene so it is viewable by all cameras and then compute a color transform for

[1] As a point of clarification, αF is an RGB value, while α is a single-channel value; therefore, the mean values are RGB values, while the variance values we use are the lengths of the RGB variance vectors.

[2] For our experiments the depth range is about a foot around the plane. While the range only has to be narrow enough to avoid including the background, a relatively narrow range is desirable for faster computation.



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

Figure 4: Results for a tree in front of a background of trees. A horizontally translating camera captures 30 images of a static scene. Our algorithm automatically pulls out the matte. (a) Central image. (b) Trimap. (c) Alpha matte. (d) Zoom in on the alpha matte. (e) Zoom in on the ground truth.

each camera to match its image of the color checker to that of the central reference camera. We also perform vignetting calibration and have found this to significantly improve the quality of our variance estimates and, by extension, our trimaps and alpha mattes.

In our implementation, we distinguish between two scenarios. The first is an online system that can process quarter-VGA images at several frames per second, and the second is an offline system that can produce high-quality VGA results at about one frame per second. The distinction between the two methods is in the search stage. Specifically, in the offline system we perform our depth search for every pixel, whereas in the online system we compute the variance for one plane only, filter it to remove high variances due to non-planarity of the foreground, and then perform our depth search only for the pixels in the unknown region of the trimap.

5 Results

In this section, we present results using our algorithm. First, we show results for static scenes. For the static results, we acquired a sequence of images of each object with a horizontally translating camera. Each object is captured by 30 to 40 images. In Figure 4, we show results for a tree filmed in front of several other trees. Pulling a matte for this scene is very challenging as the foreground and background structure is complex. Furthermore the colors in both layers are similar. A matting method that assumes low-frequency backgrounds would have difficulty with such a scene as would any 3D reconstruction method. Our method, however, recovers a high-quality alpha matte. In Figure 4, we compare our result to a ground truth result obtained using triangulation matting [Smith and Blinn 1996]. Note that our result recovers details such as single pine needles, which are also seen in the ground truth matte.

In Figure 5, we show results for an object that has high-frequency content. The fur on the doll is various shades of pink and white. For this object, our use of variance statistics is very important as the colors for neighboring foreground pixels can vary significantly.

Our method can pull mattes for foreground objects filmed over a wide variety of backgrounds. In Figure 5, we show results for a Santa doll filmed in an office corridor. The background structure in this scene is significantly different than that of the previous results; the background depth range is large and it also includes a glass door with strong specular reflections, yet we can successfully pull a matte for the intricate hair structure on Santa's beard.

In our final static result, we explore how the number of images affects the alpha matte quality. To do this, we captured a sequence of 120 images of a stuffed gorilla and applied our algorithm to varying subsets of images ranging from 8 up to 120 images. Figure 6 shows the resulting alpha mattes for different numbers of images. As we drop the number of images used, the alpha matte quality is retained. In general, the number of cameras needed to get good results is a function of scene content and camera spacing. While there is no magic number of cameras needed for our method, we have obtained good results with as few as eight cameras.

We now show single-frame snapshots for results computed for dynamic scenes. The reader is encouraged to view the video clips of these scenes and several additional scenes in our supplementary materials. Figure 7 shows two examples of pulling a matte for intricate hair structure at two frames per second. The first case is a snapshot from a video sequence of an actor talking and moving his head. Of interest is the fact that there is a person moving in the background. The actor is correctly pulled from the scene, as can be seen in the composite. The second case shows results for an actor filmed with a moving camera array. This case is difficult because the statistics of the background constantly change as the camera is panning. Figure 8 shows a zoom-in on a VGA result.

Our algorithm can handle not only hair and trees, but can pull mattes for transparent objects, such as fluids and smoke, without any special modifications to the algorithm. In Figure 9, we show a VGA result for coffee being poured into a glass pot. This scene is very challenging due to highlights on the coffee pot, the transparency of the glass, and bubbles from the liquid being poured. In Figure 1, we show a VGA result for a video sequence of smoke. Another frame from this sequence is shown in Figure 10. Note that both results accurately capture a single ring of smoke. The objects in these scenes are difficult to model, yet the alpha mattes are surprisingly good.

We envision that in a movie studio a director would wish to pull an alpha matte for a high-quality movie camera that was not directly integrated into our system. For such a setup, we would use the cameras from our system to generate an alpha matte for the viewpoint of the movie camera. For this application, we need to generate an alpha matte for a virtual viewpoint. As our camera positions are known, and our alpha matte is computed for a single foreground depth, computing alpha for a virtual view is trivial using simple image-based rendering techniques. In Figure 10, we show a validation of this by computing a matte, using only the data from the seven outer cameras, for the viewpoint of the central camera. This result is virtually identical to the result that uses all eight cameras.

6 Discussion and Future Work

While we have achieved good results with our method, our system does suffer from several limitations. First, we assume that alpha is fixed and not view-dependent. While true in practice for many objects, some materials exhibit view-dependent alpha due to self-occlusion. Self-occlusion causes a high variance for pixels in the synthetically refocused image. This results in an incorrect alpha value. Using a narrow baseline for our cameras limits these errors. For our scenes, where the background was a couple meters from the foreground, we found that a half-meter baseline worked well. Additionally, using a per-camera weighting term designed to preserve view-dependent effects [Buehler et al. 2001] could reduce these errors. By weighting cameras closer to the reference view more heavily, we can limit the effects of self-occlusion.

Second, we are limited by aliasing in our light fields. In practice, we have found errors due to aliasing to be significant only for our measurements for pixels on the background. Aliasing causes

the variance measurements to be incorrect due to periodic textures being aligned when we synthetically refocus our data. This causes a background pixel to have non-zero alpha. There are several design improvements that could alleviate these problems. The first is to use more cameras. We believe our algorithm would work well with a large, dense camera array such as that shown by Wilburn *et al.* [2005]. Even with a small number of cameras, different camera distributions could reduce aliasing artifacts. For example, cameras could be concentrated more towards the center reference camera. If one were to use a 2D grid, the system would benefit from background color variation that occurs both horizontally and vertically. Furthermore, in man-made settings, as background structures are primarily horizontal and vertical, using a diagonal cross arrangement could be useful, as it would maximally spread these features in the synthetically refocused images.

Third, we assume that the variance of the background is several orders of magnitude larger than that of the foreground, when this is not the case, our method will fail. Fortunately, this is true for many scenes, and even very specular surfaces have $\mathbf{var}(\mathbf{F})$ a few orders of magnitude lower than $\mathbf{var}(\mathbf{B})$.[3] Furthermore, $\mathbf{var}(\mathbf{B})$ can be high even for scenes that do not have high-frequency backgrounds. Even a background with a very low spatial frequency, such as two different and constant colors, side by side, can be enough to generate a high variance, provided that rays from the camera array sample both colors; in the limit, a foreground point needs to be imaged in front of two different background colors [Smith and Blinn 1996]. Nevertheless, when $\mathbf{var}(\mathbf{B})$ is low, the input reduces to a single camera input with a known background value. In this case, one can use existing algorithms such as blue-screen matting [Smith and Blinn 1996] or Bayesian Matting [Chuang et al. 2001]. A natural extension, therefore, is to combine our algorithm with existing methods such that image content will dictate the appropriate alpha matting algorithm.

One can generalize equation 6 to higher order statistics; this a potentially useful extension worth investigation. More generally, one can consider the *distributions* and not just means and variances for pulling the matte. While 8 cameras may be enough to estimate the mean and variance of a distribution, due to the aliasing issues discussed above, it is not enough to explicitly model a distribution. However, using a camera array of, say, 100 cameras, would make it possible to use more sophisticated distribution models.

7 Conclusions

We have presented a fast, automatic system for high-quality natural video matting using a camera array. Our matting algorithm works well on difficult scenes. It is efficient, orders of magnitude faster than previous natural video matting systems, and is amenable to real-time implementation as it is linear in the number of cameras.

8 Acknowledgements

We would like to thank the anonymous reviewers, Frédo Durand, Matthias Zwicker, Bennett Wilburn, Dan Morris, and Merrie Morris for their helpful comments on earlier revisions of this paper. We would also like to thank John Barnwell for his help with building the camera array structure. Lastly, we are very grateful to our patient actors – we appreciate their willingness to model dramatic hairstyles for our videos. This work was completed while the first author was an intern at MERL; he was additionally funded by NSF grant DGE-0333451.

References

BUEHLER, C., BOSSE, M., MCMILLAN, L., GORTLER, S., AND COHEN, M. 2001. Unstructured lumigraph rendering. In *Proceedings of ACM SIGGRAPH 2001*, ACM Press/ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 425–432.

CHUANG, Y.-Y., ZONGKER, D. E., HINDORFF, J., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2000. Environment matting extensions: towards higher accuracy and real-time capture. In *Proceedings of ACM SIGGRAPH 2000*, ACM Press/ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 121–130.

CHUANG, Y.-Y., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2001. A bayesian approach to digital matting. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR 2001)*, IEEE Computer Society, vol. 2, 264–271.

CHUANG, Y.-Y., AGARWALA, A., CURLESS, B., SALESIN, D. H., AND SZELISKI, R. 2002. Video matting of complex scenes. *ACM Transactions on Graphics 21*, 3, 243–248.

ISAKSEN, A., MCMILLAN, L., AND GORTLER, S. J. 2000. Dynamically reparameterized light fields. In *Proceedings of ACM SIGGRAPH 2000*, ACM Press/ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 297–306.

KOLOMOGROV, V., CRIMINISI, A., BLAKE, A., CROSS, G., AND ROTHER, C. 2005. Bi-layer segmentation of binocular stereo video. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR 2005)*, vol. 2, 407 – 414.

LI, Y., SUN, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Lazy snapping. *ACM Transactions on Graphics 23*, 3, 303–308.

LI, Y., SUN, J., AND SHUM, H.-Y. 2005. Video object cut and paste. *ACM Transactions on Graphics 24*, 3, 595–600.

MCGUIRE, M., MATUSIK, W., PFISTER, H., DURAND, F., AND HUGHES, J. 2005. Defocus Video Matting. *ACM Transactions on Graphics 24*, 3, 567–576.

ROTHER, C., KOLMOGOROV, V., AND BLAKE, A. 2004. Grabcut: interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics 23*, 3, 309–314.

RUZON, M. A., AND TOMASI, C. 2000. Alpha estimation in natural images. In *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR 2000)*, vol. 1, 18–25.

SMITH, A. R., AND BLINN, J. F. 1996. Blue screen matting. In *Proceedings of ACM SIGGRAPH 1996*, ACM Press/ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 259–268.

SUN, J., JIA, J., TANG, C.-K., AND SHUM, H.-Y. 2004. Poisson matting. *ACM Transactions on Graphics 23*, 3 (August), 315–321.

VLAHOS, P., 1958. Composite photography utilizing sodium vapor illumination (u.s. patent 3,095,304), May.

VLAHOS, P., 1971. Electronic composite photography (u.s. patent 3,595,987, July.

VLAHOS, P., 1978. Comprehensive electronic compositing system (u.s. patent 4,100,569), July.

WANG, J., BHAT, P., COLBURN, A., AGRAWALA, M., AND COHEN, M. 2005. Interactive video cutout. *ACM Transactions on Graphics 24*, 3, 585–594.

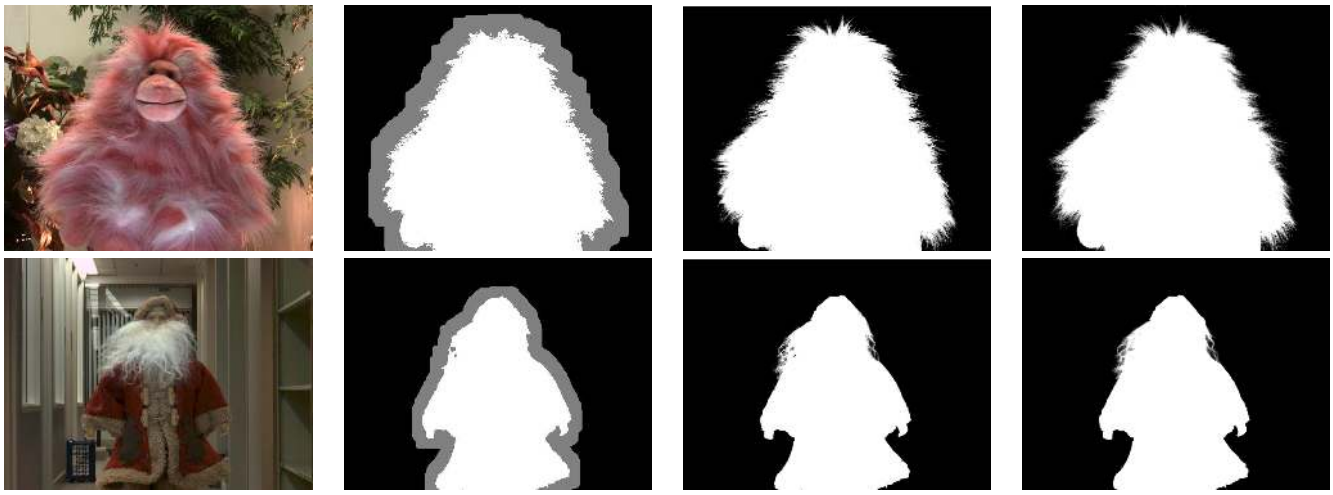
WEXLER, Y., FITZGIBBON, A., AND ZISSERMAN., A. 2002. Bayesian estimation of layers from multiple images. In *Proceedings of 7th European Conference on Computer Vision (ECCV)*, vol. III, 487 – 501.

WILBURN, B., JOSHI, N., VAISH, V., TALVALA, E.-V., ANTUNEZ, E., BARTH, A., ADAMS, A., HOROWITZ, M., AND LEVOY, M. 2005. High performance imaging using large camera arrays. *ACM Transactions on Graphics 24*, 3, 765–776.

ZITNICK, C. L., KANG, S. B., UYTTENDAELE, M., WINDER, S., AND SZELISKI, R. 2004. High-quality video view interpolation using a layered representation. *ACM Transactions on Graphics 23*, 3, 600–608.

ZONGKER, D. E., WERNER, D. M., CURLESS, B., AND SALESIN, D. H. 1999. Environment matting and compositing. In *Proceedings of ACM SIGGRAPH 1999*, ACM Press/ACM SIGGRAPH, Computer Graphics Proceedings, Annual Conference Series, 205–214.

[3] In our scenes $\mathbf{var}(\mathbf{F})$ is on the order of a few hundred or less, while $\mathbf{var}(\mathbf{B})$ is upwards of several thousand. These units are on the order of 8-bit RGB levels squared, i.e. 0 to 255^2

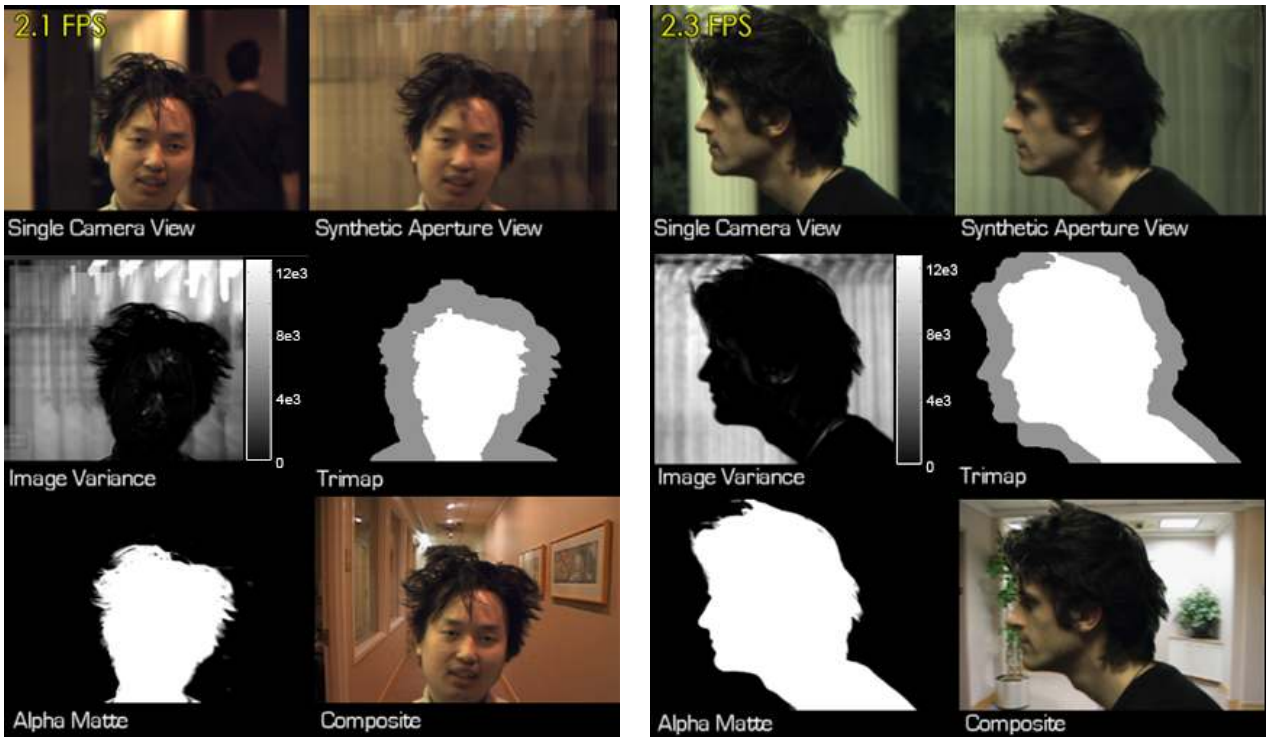


(a) (b) (c) (d)

Figure 5: Results on static scenes. A horizontally translating camera captures 40 images of a static scene. Our algorithm automatically pulls out the matte. (a) Central image. (b) Trimap. (c) Alpha matte. (d) Ground truth.

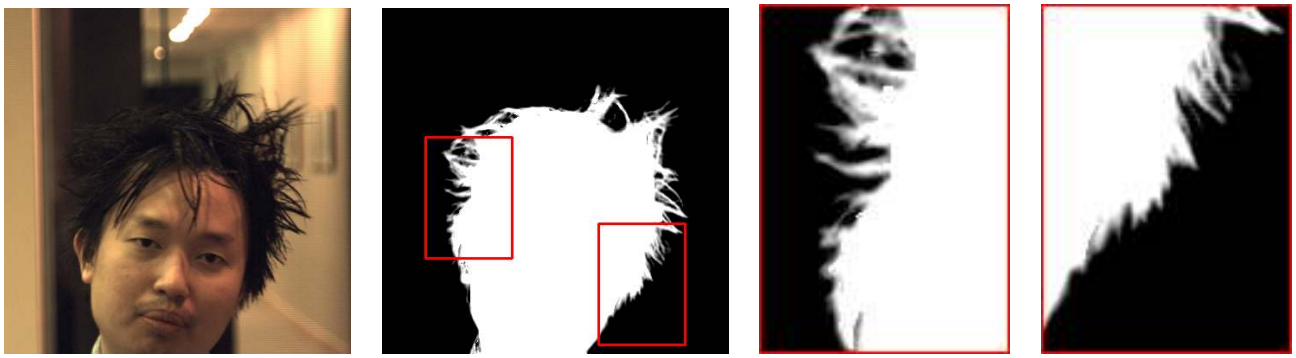
(a) (b) (c) (d)

Figure 6: Varying the number of images used. (a) Central image. (b) Alpha matte using 120 images, (c) 60 images, and (d) 8 images.



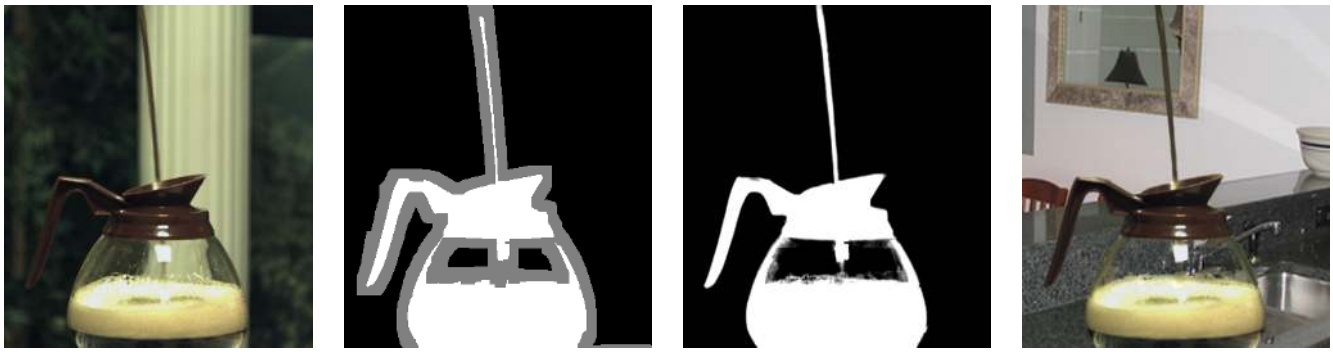
(a) (b)

Figure 7: Alpha matting people and hair. (a) A snapshot from a video sequence of a moving head. Observe that there is a moving person in the background that is correctly segmented out in the composite. (b) A snapshot from a video sequence obtained by a moving camera array.



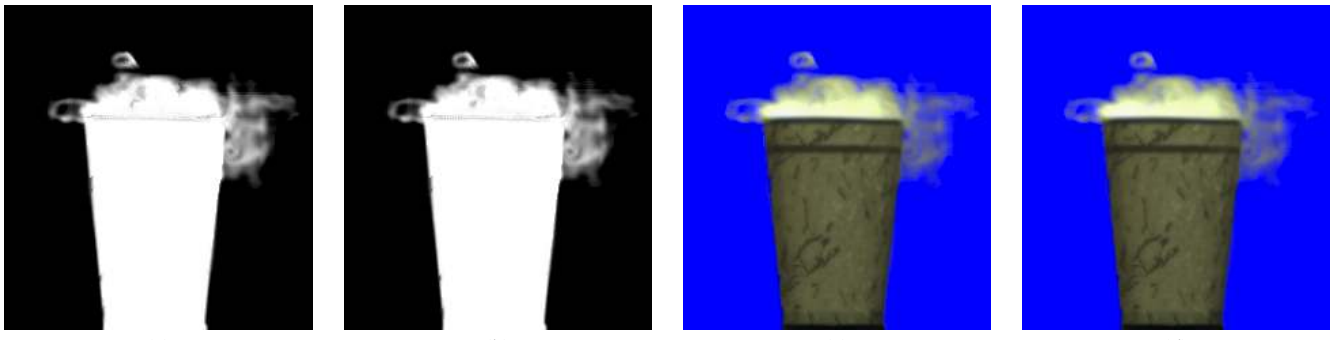
(a) (b) (c) (d)

Figure 8: VGA alpha matte. (a) Image from the central camera. (b) Alpha matte. Red rectangles mark the zoom-ins shown in (c) and (d).



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

Figure 9: Alpha matting semi-transparent objects. A VGA snapshot from a video sequence of coffee pouring into a glass pot. (a) Central image. (b) Trimap. (c) Alpha matte. (d) Composite.



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

Figure 10: Computing alpha mattes for a virtual camera view. (a) Alpha Matte computed using the 7 outer cameras in our array for a virtual view co-located with the center camera. (b) Alpha Matte using all eight cameras. (c) Composite using the alpha multiplied foreground computed for the virtual view. (d) Composite using the alpha multiplied foreground computed with data from all eight cameras.