Robust Real-Time Bi-Layer Video Segmentation Using Infrared Video

Qiong Wu, Pierre Boulanger and Walter F. Bischof
Department of Computing Science
University of Alberta
{qiong,pierreb,wfb}@cs.ualberta.ca

Abstract

In this paper, we propose a novel method for the automatic segmentation of a foreground layer from a natural scene in real time by fusing infrared, color and edge information. This method improves on previous video foreground/background segmentation algorithms by making the system totally independent of changes in ambient lighting. A powerful data acquisition unit was developed using an optical technique that automatically gives synchronized and registered color video and infrared (IR) video at 850 nm. Using the fused information produced by the IR video one can then automatically initialize a pentamap, which is processed by the graph cuts algorithm. We show that the pentamap can simplify the graph construction process, improve the efficiency of the graph cut algorithm, and allow to use a more reliable color Gaussian Mixture Model (GMM) than the usual trimap method. At the end of the processing pipeline, a simple border-blurring algorithm is used to simulate matting effects on the foreground-background boundary. Experimental results are presented.

1. Introduction

This paper addresses the problem of the automatic and efficient segmentation of a foreground layer from a natural scene that is robust to changes in ambient lighting. Our aim is to segment foreground objects from a natural background in real-time without user interaction. A prime application of this system is in telepresence where there is a need to remove the background and replace it with a new one. The result of our segmentation process is a binary label map where each pixel is classified as belonging to either the foreground or the background. In the matting process, which joins the foreground with the new background, we do not really compute alpha values but instead use border-blurring, which results in a much smoother transition from foreground to background. In Section 2, we review the various methods proposed in the literature to solve this segmentation problem. In Section 3, we give a general overview of the proposed method. In Section 4, we describe the graph cut segmentation algorithm. In Section 5, we describe the complete segmentation pipeline. In Section 6, we describe the border matting method based on local blurring. In Section 7, we present more experimental results, and in Section 8, we present our conclusions.

2. Previous Approaches

Foreground-background segmentation has been an active research area for many years. Several methods have been proposed in the past, including background subtraction, foreground segmentation of stereo video, depth-camera assisted foreground segmentation and various interactive image segmentation methods.

Background subtraction is often applied in cases where one wants to separate the foreground from a static background. In these simplified cases, a simple adaptive thresholding technique can be used where a background template is used as a reference. More sophisticated algorithms use time-adaptive, per-pixel mixtures of Gaussians color models [7] capable of dealing with dynamic backgrounds. However, these algorithms often fail because of camera noise and non-constant illumination. They also do not perform well when the foreground pixels do not have sufficient contrast with the background area. In general, most of these algorithms are not robust to ambient illumination.

Foreground segmentation based on stereo video uses depth information reconstructed from stereo video to assist foreground segmentation [6, 9]. Dense depth computation from stereo video is not only computationally intensive but is also not accurate and not robust because it is frequently impossible to compute depth in low-textured or homogeneous

regions. In addition, the accurate computation of stereo occlusion is essential for achieving good segmentation results.

Depth-camera assisted foreground segmentation uses a regular web camera in combination with a time-of-flight range sensor to acquire depth data to perform foreground-background segmentation [12, 13]. Depth cameras have many desirable properties, including robustness against illumination changes and easy processing. These systems have, however, problems with scaling, resampling and synchronization. There are some depth video cameras available today, e.g. the ZCam [8], that are capable of producing perfectly synchronized RGB and depth signals. Unfortunately these sensors are very expensive and difficult to integrate with normal video technology.

Interactive image segmentation, e.g. level sets, active contours and graph cuts based methods, are based only on color and/or edge information. Many of these algorithms are very good and in some cases optimal. Unfortunately, they all require manual user input, and they are not really applicable to our problem, unless one can find an automatic way of pre-segmenting the images. In some ways, our proposed system falls into this category as it uses the IR image to pre-segment the color image, with graph cuts algorithm being used to further improve this pre-segmentation.

3. Proposed Method

Traditional segmentation methods based on color and edge information, e.g. Grabcut [11], can only achieve desirable result with user interaction. In order to do automatic foreground-background segmentation, one needs to have more information than what is given in the original color video. This inspired us to seek a solution that involves combining infrared (IR) information with color and edge information. Previous approaches that combine color video with depth information, such as stereo video and 3D cameras, all have major drawbacks, as described in the last section. This motivated us to use an IR camera, which is less expensive than 3D camera and much faster and robust than dense depth computation from stereo video. The way we combine the IR camera and the color camera also saves us from resampling and geometric cameras alignment.

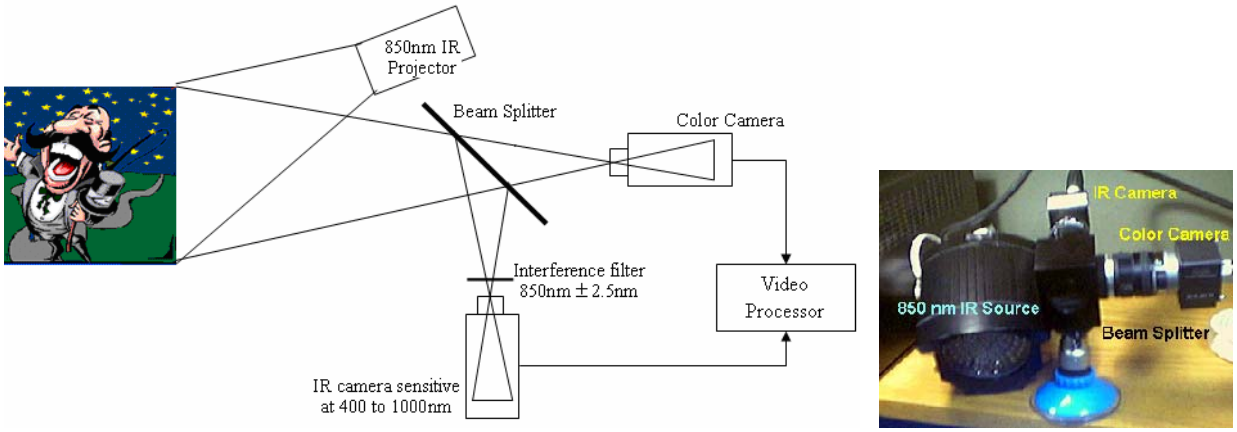


Figure 1: Data acquisition unit.

(a) Input frame: IR image and color image (b) Background substitution in three different frames

Figure 2: An example of input images and segmentation results of the proposed system: (a) One frame of an automatically synchronized mirrored video pair: IR video and color video, (b) Segmentation results using the proposed algorithm for three different frames.

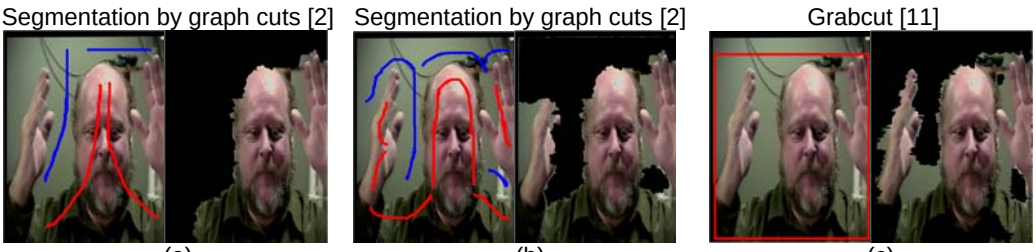


Figure 3: Comparison with other segmentation methods: (a) and (b) show how the segmentation results of the trimap graph cut algorithm are sensitive to the initial trimap specified by the user. The red brush stokes denote foreground seeds and the blue brush stokes denote background seeds. (c) Segmentation result for Grabcut: the user input is shown on the left and the segmentation result is shown on the right.

As shown in Figure 1, the foreground is illuminated by an invisible IR source at 850 nm that is captured by an infrared camera tuned to 850 nm, with a narrow-band (850nm ± 25 nm) filter being used to reject all light except the one produced by the IR illuminator. The advantage of this system is that it gives full control of the illumination process, which is independent of ambient lighting in the visible spectrum. In the proposed configuration, the IR and the color images are synchronized in time and space, using a genlock mechanism for temporal synchronization and an optical beam splitter for spatial registration. With this system, there is no need to align the images using complex calibration algorithms since they are guaranteed to be coplanar and coaxial. One advantage of the IR images provided by the system is that one can segment the foreground from the background using simple thresholding techniques. Figure 2(a) shows an image pair produced by the acquisition system. As one can see, the IR image is a mirror version of the color image. This is due to the beam splitter, and it can be easily corrected with a simple image transposition.

For each frame, the IR image is used to pre-segment the color image using a simple thresholding technique, and some noises are removed by the morphological operation (e.g. the light spot in Figure 2(a) left IR image). This pre-segmentation is used to initialize a pentamap (see below), which is then used by graph cuts algorithm to find the complete foreground region. Finally, a border-blurring algorithm is applied in a narrow strip around the foreground boundary to remove boundary artifacts and to simulate alpha-matting. Figure 2(b) shows the segmentation results.

4. Image Segmentation Using Graph Cut

Boykov *et al.* [2-4] proposed the graph cuts based segmentation algorithm to perform various segmentation tasks. In this algorithm, a graph is constructed according to an energy function derived from the original image. The image segmentation is obtained by minimizing the energy function corresponding to the min-cut of the constructed graph. In this section, we introduce the algorithm and some notations used in this paper.

4.1. Trimap Estimation

In foreground-background segmentation, the input to the graph cut algorithm is a trimap T, which contains regions labeled foreground T_F, background T_B and unknown T_U. Usually the trimap is coarsely initialized by the user, and the image is segmented largely according to the foreground-background color models derived from the trimap. For example, the segmentation approach of [2], which we call segmentation by graph cuts, requires the user to select foreground and background seeds. Grey-value histograms for foreground and background are calculated from these seeds. The remaining pixels in the unknown region T_U are labeled according to the probability of their intensity belonging to the different histograms. Two examples of this approach are shown in Figure 3(a) and 3(b). Another major segmentation approach based on graph cuts is the Grabcut [11] algorithm. In this algorithm, the user initializes the trimap by drawing a rectangle around the desired object. Color GMMs for the background and the foreground regions are initialized according to the trimap. GMMs are then evolved iteratively to minimize the energy function. An example of the input and output images is shown in Figure 3(c). The color model in both approaches is very sensitive to the initial trimap provided by the user. In many cases, the color models derived from the trimaps are not reliable. This can be seen in Figures 3(a) and 3(b), which have slightly different trimaps, but very different segmentation results. Using Grabcut [11] algorithm,

the user needs to further edit the segmented image to add or remove pixels after the initial segmentation.

4.2. Segmentation by Energy Minimization

To segment an image by graph cut, we consider an image as an array $\mathbf{Z}= (\mathbf{z}_1,\mathbf{z}_2\ldots\mathbf{z}_{|P|})$, where $\mathbf{z}_n$ is the RGB color vector for the nth pixel and |P| is the number of pixels. The segmentation of the image is defined by a binary vector $f=(f_1,f_2,\ldots,f_{|P|})$, $f_n \in \{0,1\}$, where 1 is the foreground label and 0 is the background label. A good segmentation should correspond to the minimum of a "Gibbs" energy function in the form of:

$$E(f) = D(f) + V(f) \tag{1}$$

D(f) and V(f) are defined as:

$$D(f) = \gamma \sum_{p \in P} D_p(f_p) \tag{2}$$

$$V(f) = \sum_{\{p,q\} \in N} [f_p \neq f_q] V_{p,q}(f_p, f_q) \tag{3}$$

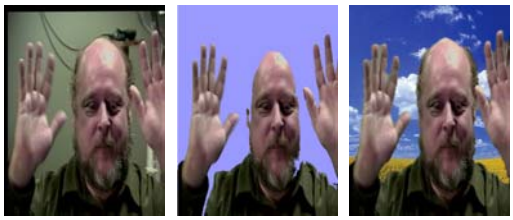
where γ specifies the relative importance of D(f), [.] is a delta function that gives 1 for $f_p \neq f_q$ and 0 otherwise, and N is the set of all pairs of neighboring pixels. The data term D(f) gives a penalty for assigning different labels to each pixel, and the smoothness term V(f) corresponds to the penalty of the edge information.

An undirected graph G=(V,E) is constructed according to this energy function. Each node in the graph corresponds to a pixel in the original image. There are two additional terminal nodes, one for the object, called OBJ, and the other for the background, called BKG. The weight of edges connecting nodes and terminals are given by the data term, and the weight of edges connecting neighborhood nodes are given by the smoothness term. The segmentation of the image is found by solving the min-cut on the graph G, which should correspond to the minimum value of the energy function.

5. Proposed Segmentation Method

We propose a method composed of three major steps: 1) data acquisition and calibration; 2) initialization of the pentamap according to the input data; and 3) pentamap optimization using the graph cuts algorithm.

5.1. Data Acquisition and Calibration



(a) (b) (c)

Figure 4: (a) The original color image, (b) foreground found by thresholding the IR image, and (c) foreground segmentation result.

As mentioned before, the design of our data acquisition unit provides major advantages over other hardware-assisted segmentation methods. Our camera design is very similar to the one used by McGuire *et al.* [10], except that they combine two regular web cameras while we combine a color and an IR camera, and they need a special studio setup in order to do foreground segmentation.

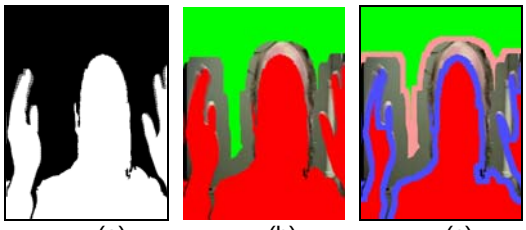
Our design has three main advantages. First, the glass beam-splitter partitions incident light between two perpendicular output paths, creating mirror images that are perfectly synchronized in time and space. Second, the IR camera has the same resolution as the color camera, so the two mirror images can be directly correlated without having to perform scaling or re-sampling. Third, unlike a 3D camera with an IR source attached to the camera (e.g. the Swiss ranger SR-2 3D camera), the IR source is separated from the IR camera in our design. One advantage of this design is that the user can put the IR source closer to any object that should be segmented. Hence the foreground is defined as the object closer to the IR source rather than the one closer to the camera. Since not all light emitted by the IR source is reflected back to the camera, a part of the foreground may not be captured by the IR camera, as shown in Figure 4. One can find the missing foreground parts by applying the graph cuts algorithm.

Figure 5: MacBeth calibration pattern.

Before any further processing, we must perform color calibration and optical centre estimation. To do so, we use the MacBeth calibration pattern shown in

Figure 5. After selecting four corresponding point pairs in the IR and color images, the offset caused by the small differences in the position of the camera-lens centers can be estimated by computing the translation among these point pairs. This step can be done automatically by recognizing corresponding points. In addition, we also calibrate the color of the camera using the mean-square technique described in [1], where the reference is supplied by 24 independently calibrated painted squares of the Mac Beth color chart.

5.2. Pentamap estimation



(a) (b) (c)

Figure 6: (a) Foreground IR MASK; (b) Trimap with red=T_F, green=T_B, and the remaining area = T_U; (c) Pentamap with red=T'_{CF}, green=T'_{CB}, blue=T'_{LF}, pink=T'_{LB}, and the remaining area=T'_U.

Before introducing the pentamap, let us first take a look at some features of the IR image and how it can be used to initialize a trimap. In an IR image, the brighter part indicates an object closer to the IR source (belonging to foreground), and the dark part is either far from the IR source (belonging to background) or in the light-shadow area (belonging to foreground). We call the brighter area foreground MASK. It can be estimated by thresholding the IR image, MASK=$\{p \in P | z_p \geq T\}$. Since we are sure that MASK belongs to the foreground, we can say T_F=MASK. A missing foreground part is within a certain distance from the MASK (because the missing foreground parts are connected to MASK), hence we can predict that any area outside of this distance belongs to the background. Assuming this distance is τ (in number of pixels), we can represent the predicted background by applying the morphological operation T_B=P-MASK.dilation(τ). The remaining area is unknown, so T_U=P- T_F- T_B= MASK.dilation(τ)-MASK.

For the traditional graph cuts algorithm, the trimap is used to build color GMMs for foreground and background, which are derived from T_F and T_B respectively. Here, we propose the idea of a pentamap, which can derive more reliable color GMMs, leading to more accurate segmentation results.

A pentamap T' partitions the image into five regions, certain foreground (CF), certain background (CB), local foreground (LF), local background (LB) and unknown (U). We can represent this as

T': P $\rightarrow$ $\{T'_{CF}, T'_{CB}, T'_{LF}, T'_{LB}, T'_U\}$, with
T'_{CF} =$\{p | p \in$ MASK.erosion(s)$\}$
T'_{LF} =$\{p | p \in$ MASK- $T_{CF}\}$
T'_{CB} =$\{p | p \in \sim$(MASK.dilation(τ +s))$\}$
T'_{LB} =$\{p | p \in$ MASK.dilation(τ +s)- MASK.dilation(τ)$\}$
T'_U =$\{p | p \in T'$- T'_{CF}-T'_{CB}- T'_{LF}- $T'_{LB}\}$.

Given these definitions, one can transform a pentamap into a trimap as follows:

$T'_{CF} + T'_{LF} = T_F$
$T'_{CB} + T'_{LB} = T_B$
$T'_U = T_U$

Figure 6 shows an example of trimap and pentamap. In the pentamap model, T'_{LF} (T'_{CB}) is a narrow strip of width s that is separated from T_F (T_B). In our approach, color GMM for foreground will be derived from T'_{LF} rather than T_F (color GMM for background will be derived from T'_{LB} rather than T_B), given that it is reasonable to assume that the color in the unknown region is consistent with the color in its neighborhood regions rather than the whole map. That is, the color in T'_U should be consistent to the color of T'_{LF} or T'_{LB} rather than the whole region of T_F and T_B. In the experimental section, we show that our pentamap performs better than the trimap.

A pentamap can be automatically initialized from the IR image. The threshold T can be fixed since the intensity of the IR image does not change as ambient light changes. The threshold was set to T=0.004 in our experiment. The value of τ is determined by the configuration of the IR camera and the distance between the foreground object and the IR source. If the user does not change the camera configuration during the video capture, the value of τ increases with the distance of the object to the IR source. For example, if the distance is d, τ=f(d). Since we define the foreground object as the object within distance D_{max} to the IR source, $\tau=\tau_{max}$=f(D_{max}). The value of s (in the definition of T'_{CB} and T'_{LB} above) does not change the segmentation result much if $s \in$ [15, 25].

5.3. Graph Cut

One advantage of the pentamap is that it simplifies the graph complexity and thus improves the efficiency of the graph cut algorithm.

An example of the graph construction (using only T-links) is shown in Figure 7. For each image to be segmented, an undirected graph G=(V,E) is defined with a set of nodes V and a set of undirected edges E that connect these nodes. A node is created for:

- T'_{CF}
- T'_{CB}
- Each pixel in T'_{LF}
- Each pixel in T'_{LB}
- Each pixel in T'_{U}

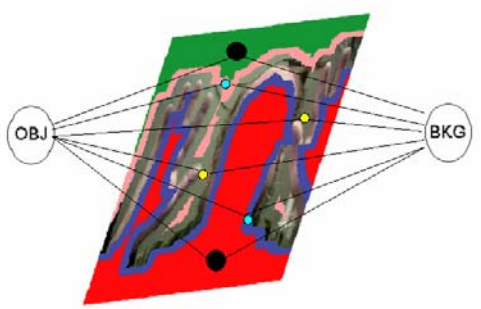


Figure 7: Graph construction with T-links.

Edges are added between nodes pair in the following cases:

- $\{V_i, OBJ/BKG\}$, $V_i \in V$. Such edges are called T-links.
- $\{V_i, V_j\}$, $V_i \in T'_{LB}/T'_{LF}/T'_{U}$, $V_j \in T'_{U}$, $\{i,j\} \in N$ and $i \neq j$. Such edges are called N-links.

The weight of T-links corresponds to the penalty of assigning a node to the corresponding terminal, which is given by the data term (2). The weight of N-links corresponds to the contrast/edge information, which is given by smoothness term (3).

As in [11], we use the GMM as color space model. The GMM for the foreground is derived from T'_{LF}, and the GMM for the background is derived from T'_{LB}. Each GMM is a full-covariance Gaussian mixture with M components (M=10), which can be interpreted as the number of color clusters. We represent the GMM for the foreground as $K_F=\{K_{F1}, K_{F2}\ldots K_{FM}\}$, and similarly for the background $K_B=\{K_{B1},K_{B2}\ldots K_{BM}\}$. For each pixel p in the unknown area T'_{U}, the probability that it belongs to the foreground and background, respectively, is defined in (4) and (5):

$$\forall p \in T'_U, \Pr(p \mid f_p = 1) = \max_{i=1\ldots M} \Pr(Z_p \mid K_{Fi}) \quad (4)$$

$$\forall p \in T'_U, \Pr(p \mid f_p = 0) = \max_{i=1\ldots M} \Pr(Z_p \mid K_{Bi}) \quad (5)$$

The edge weights are defined in Table 1, where $K=\infty$ (one can use a very large value for it in the implementation). D(f) and V(f) in (1) now become

Edge	Weight	For
{Vi, OBJ}	K	$Vi \in T'_{CF}, T'_{LF}$
	0	$Vi \in T'_{CB}, T'_{LB}$
	$-\ln \Pr(p \mid f_p = 0) * \gamma$	$Vi \in T'_{U}$
{Vi, BKG}	K	$Vi \in T'_{CB}, T'_{LB}$
	0	$Vi \in T'_{CF}, T'_{LF}$
	$-\ln \Pr(p \mid f_p = 1) * \gamma$	$Vi \in T'_{U}$
(Vi,Vj)	$\exp(-\lVert z_i - z_j \rVert^2 / \beta)$ β is the expectation of $2\lVert z_i - z_j \rVert^2$	$\{i,j\} \in N$, $i \neq j$, $V_i \in T'_{LB}/T'_{LF}/T'_{U}$, $V_j \in T'_{U}$

Table 1: Edge weight table

$$D(f) = \gamma \sum_{p \in P} -\ln \Pr(p \mid f_p) \quad (6)$$

$$V(f) = \sum_{\{p,q\} \in N} [f_p \neq f_q] \exp(-\lVert z_p - z_q \rVert^2 / \beta) \quad (7)$$

Compared to previous segmentation techniques based on graph cuts, our graph construction is much simpler in terms of number of nodes and edges. Rather than creating a node for every pixel in the image, all pixels in T'_{CF} (and T'_{CB}) are represented by a single node. This prevents a cut from being made across the T'_{CF} (and T'_{CB}) area. In addition, we add neighborhood edges under very strict conditions: Since a cut can only happen in the unknown area, a contrast term (Vi,Vj) is computed only in T'_{U} or between T'_{U} and T'_{LF}/T'_{LB} as the prediction of the object boundary. The worst-case runtime complexity for solving a min-cut problem is $O(mn^2)$, where n is the number of nodes and m is number of edges in the graph [3]. In our approach, the number of nodes for the same image can, on average, be reduced to n/5 and the number of edges can be reduced to m/2, so runtime can be reduced to $1/50*O(mn^2)$ on average.

6. Border Matting Using Blurring

In the proposed method, border blurring is applied to the object border in order to blend the foreground with the new background. We achieve the equivalent of alpha-matting without calculating alpha values. Figure 8 shows results before and after border blurring.

Alpha-matting actually calculates a weighted average of foreground color and background color for each pixel, and a Gaussian blurring filter pre-calculates a weighted average of neighborhood colors for each pixel. These two definitions are very similar, especially when a pixel is at the border between foreground and background.

Border blurring begins with the "hard" segmentation produced by the graph cuts algorithm, denoted here as F. A blurred boundary contour can be defined by morphological operations: Blur_mask={F-F.dilation(s1).erosion(s2).dilation(s3)}. (In our experiments, we used s1=4, s2=12, and s3=6).

Figure 8: Boundary after segmentation (left), and after border blurring (right).

Figure 9: Comparison of border blurring with Bayesian matting. The images on the left are generated by Bayesian Matting and the images on the right are generated by border blurring.

In this way, one can get a smooth boundary strip Blur_mask that contains all pixels of boundary artifacts.

A Gaussian filter is then applied to the Blur_mask, so that there is a smooth transition between foreground and background, eliminating obvious artifacts at the boundaries. Conventional matting techniques, such as Bayesian matting (see [5]), compute α values based on the color of neighborhood pixels. This computation is very slow and does not perform well for objects with relatively smooth boundaries. We implemented both, Bayesian matting and border blurring in Matlab. For an image of size 365*480, Bayesian matting took more than 40 minutes and border blurring took less than 0.1 seconds. The results for both methods are shown in Figure 9. The results presented here look very similar to the border matting results of Grabcut [11]. Our method is, however, much simpler and more efficient.

7. Experimental Results

Figure 10 shows further experimental results of background substitution before border blurring is applied. We show results for trimaps and pentamaps. It took on average only 0.1 seconds for processing a 365*480 image on a 2GHz Pentium desktop machine with 1G RAM.

Because of time constraints, we did not do experiments with changing backgrounds. It is, however, not hard to see that, as long as the moving background is outside the effective distance of the IR source, the foreground MASK contains only the interested foreground object, making our method valid even in the presence of changing backgrounds.

8. Conclusion

This paper presents a new algorithm for bi-layer segmentation of natural video in real time using a combination of IR and color images. The proposed design can automatically digitize synchronized video sequences without the need for further temporal or geometric processing. One of the benefits of this hardware design is that the pentamap can be initialized robustly with information acquired by the IR camera, which is independent of ambient lighting. There are, however, two shortcomings with our hardware design. First, our hardware can automatically recognize the foreground object only if it is within the effective distance of the IR source, and this distance acts like a plane dividing foreground and background. Therefore, the user may need to move the IR source around and find the best position by observing whether the IR image yields a good foreground MASK. Second, if an

object appears closer than the foreground it will also be captured.

The approach proposed here is a not only a new solution to real-time bi-layer segmentation, but also to motion tracking and many other segmentation problems that are based on graph cut algorithms and sensor fusion.

9. References

[1] P. Boulanger, "From High Precision Color 3D Scanning of Cultural Artifacts to its Secure Delivery over the WEB: A Continuum of Technologies", *Workshop on Recording, Modeling and Visualization of Cultural Heritage*, May 2005, CD proceedings.

[2] Y. Boykov, and M.-P. Jolly, "Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images", *Proc. IEEE Int. Conf. on computer vision*, 2001, CD-ROM.

[3] Y. Boykov, and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004, pp. 1124-1137.

[4] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts", *International Conference on Computer Vision*, 1999, pp. 377-384.

[5] Y.-Y. Chuang, B. Curless, D. Salesin, and R. Szeliski, "A Bayesian approach to digital matting", *In Proc. IEEE Conf. Computer Vision and Pattern Recog.*, 2001.

[6] C. Eveland, K. Konolige, and R.C. Bolles, "Background modeling for segmentation of video-rate stereo sequences", *Proc. IEEE Computer Vision and Pattern Recognition (CVPR),* Santa Barbara, CA, USA, Jun 1998, pp. 266-271.

[7] N. Friedman, S. Russell, "Image Segmentation in Video Sequences: a Probabilistic Approach", *Proc. 13th Conf. on Uncertainty in Artificial Intelligence*, Aug 1997, pp. 175-181.

[8] G. Iddan and G. Yahav, "3D Imaging in the studio (and elsewhere)", *Proc. SPIE*, 2001, pp. 48-55.

[9] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross, and C. Rother, "Bi-layer Segmentation of Binocular video", *Proc. CVPR,* San Diego, CA, US, 2005, pp. 407 – 414.

[10] M. McGuire, W. Matusik, and W. Yerazunis, "Practical, Real-time Studio Matting using Dual Imagers", *Eurographics Symposium on Rendering (EGSR),* Jun 2006, pp. 235-244.

[11] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts", *SIGGRAPH*, 2004, pp. 309-314.

[12] N. Santrac, G. Friedland, R. Rojas, "High resolution segmentation with a time-of-flight 3D-camera using the example of a lecture scene", *Fachbereich mathematik und informatik*, Sep 2006.

[13] O. Wang, J. Finger, Q. Yang, J. Davis, and R. Yang, "Automatic Natural Video Matting with Depth", *Pacific Conference on Computer Graphics and Applications (Pacific Graphics)*, 2007.

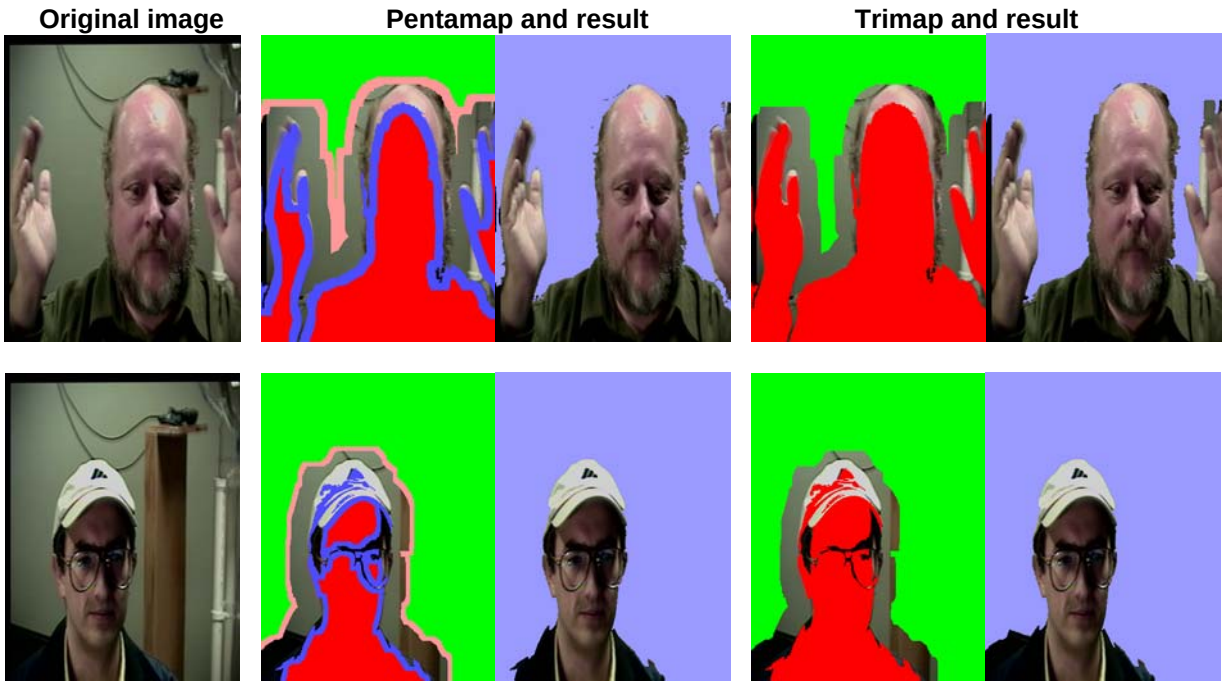


Figure 10: Comparison of pentamap and trimap results. In the upped panel, s=25, τ=55 and γ=2.2. In the lower panel, s=15, τ=55 and γ=2.2.