LIGHT WEIGHT BACKGROUND BLURRING FOR VIDEO CONFERENCING APPLICATIONS

Cha Zhang, Yong Rui and Li-wei He

Microsoft Research
One Microsoft Way, Redmond, WA 98052

ABSTRACT

Background blurring is an effective way to both preserving privacy and keeping communication effective during video conferencing. This paper proposes a light weight real-time algorithm to perform background blurring using a fast background modeling algorithm combined with a face detector/tracker. A soft decision is made at each pixel whether it belongs to the foreground or the background based on multiple vision features. The classification results are mapped to a per-pixel blurring radius image to blur the background. The algorithm produces satisfactory results under a wide range of conditions, and occupies less than 30% of the CPU cycles on a 3GHz Pentium 4 machine without further optimization.

Index Terms— Video signal processing, Teleconferencing

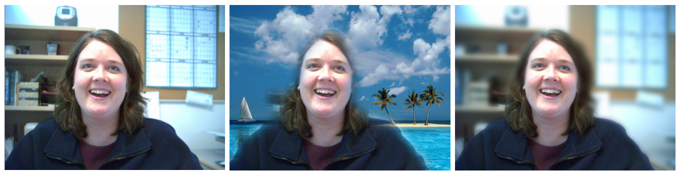

Fig. 1. Original image (left), background replacement (middle) and background blurring (right). Background replacement requires higher segmentation accuracy, while the blurring needs less. Both are good approaches to protect privacy and keep the effectiveness of the conversation.

1. INTRODUCTION

Video conferencing has become more and more popular thanks to the emergence of high speed Internet and reduced price of high quality web cameras. Wide-spread instant messaging software such as MSN Messenger, Yahoo! Messenger and AOL Instant Messenger all support voice/video chatting, where people can view each other while talking. There are however privacy concerns raised about video conferencing. For instance, some people do not want to show their living rooms or bedrooms to other people, hence they are not enthusiastic about the new video communication media.

There are a number of approaches to overcoming the privacy issue. For example, the Logitech Video Effects [1] can replace a talking face with a 3D animated avatar. It is fun to play with such effects, though the expressiveness of the avatars is often limited and cannot deliver information as rich as that conveyed by true faces. An alternative solution is to separate the foreground (fg) and background (bg) objects in the video, so that the background can be replaced by a different image or video. This would be ideal for preserving privacy while maintaining the effectiveness of the conversation, except that automatic video fg/bg segmentation is a very challenging task. In addition, human eyes are very sensitive to segmentation errors during background replacement, which demands the segmentation algorithm to be extremely accurate. Existing approaches are either too slow to be processed in real time [2, 3], or assuming a known background [4, 5], or requiring a stereo camera pair [6]. Few of them have achieved the efficiency, accuracy and convenience needed in real-world applications.

In this paper, we explore the idea of blurring the background in video conferencing instead of replacing the background completely. This has a number of advantages. First, as one would imagine, after background blurring, the foreground objects will stay focused while the background objects will be blurred, hence it protects the privacy of the person while keeping the conversation effective. Second, people are much more forgiving to the errors made in background blurring than background replacement, as shown in Figure 1. This allows us to develop very efficient fg/bg classification algorithms without concerning too much about the classification errors. Finally, background blurring has the similar effect as a wide aperture video camera which always focuses on the foreground objects. It can make the foreground objects pop up as the background is blurred, creating an extra dimension to the video.

We developed a fast pixel-level background modeling algorithm to perform fg/bg classification, assuming the camera and the background are static. Pixels that remain the same colors for a long period of time are considered as background pixels. With such a background model, we make a soft decision at each pixel how likely it belongs to the foreground. The likelihoods are then mapped to a per-pixel blurring radius image, which is used to blur the video frame after smoothing.

One problem with the above algorithm is that the foreground objects can get blurred if they stay still for a long while. An important observation we made is that faces are by far the most important objects that should always stay focused in video conferencing even though they are still. We employ a face detector/tracker on top of the above background modeling approach. The detected face regions are used to modulate the blurring radius image, making sure that the faces are not blurred. In addition, face pixels will not be pushed into the background model during the process.

Despite the simplicity of the proposed algorithm, we achieve satisfactory results on real-world sequences. The algorithm does not require information about the background before hand since it can learn the background adaptatively. This capability also makes it very robust to sudden lighting variations, camera shaking, etc. More importantly, the algorithm is very light weight. Without much optimization, it runs for 320×240 videos at 15 fps on a desktop computer with Pentium 4 3GHz processor comfortably (20-30% CPU usage) without blocking other applications.

The rest of the paper is organized as follows. The background modeling process is discussed in Section 2. The face detector and tracker are described in Section 3. The blurring process is presented in Section 4. Experimental results and conclusions are given in Section 5 and Section 6, respectively.

2. BACKGROUND MODELING

Pixel level background modeling has been extensively studied in literature. To name a few, Wren et al. [4] used Gaussian distribution to model the color variation of the background pixels, mostly for indoor environments. Later it was extended to Gaussian mixture models [7, 8], non-parametric kernel density estimators [9, 10] and three state HMM [11] to adapt to outdoor, dynamic backgrounds. A separate region-level or even object-level model can be added to further improve the background modeling quality in dynamic scenes [12, 13]. Since most video conferencing applications are indoor, we find a single Gaussian distribution is sufficient to model the color variations of background pixels. Another reason to use single Gaussian distribution is its simplicity and efficiency in implementation. On the other hand, a more sophisticated background modeling algorithm could certainly improve the system performance, if the additional computational cost is affordable.

Let an input video frame be I_t at time t. Let $\mathbf{c}_t(\mathbf{x})$ be the color of the pixel at location $\mathbf{x}$ in I_t. To make the background model robust to lighting variations, we convert the video frame from the RGB color space to the YUV color space, and use only the UV components for the task. Thus $\mathbf{c}_t(\mathbf{x})$ is a two dimensional vector. At each pixel, we maintain a background mean and a background variance, denoted as $\mathbf{m}_t(\mathbf{x})$ and $\mathbf{\Sigma}_t(\mathbf{x})$, respectively. Initially, all pixels are assumed to be foreground pixels. And we let $\mathbf{m}_0(\mathbf{x}) = [-1, -1]^T$, and $\mathbf{\Sigma}_0(\mathbf{x}) = \epsilon\mathbf{I}$, where $\mathbf{I}$ is a 2D identity matrix and ϵ is a tiny number (e.g., 10^{-3}).

Given a new input frame, the likelihood of a pixel belonging to the background can be computed using the standard Gaussian kernel:

$$p_t = \exp\{-\frac{1}{2}(\mathbf{c}_t - \mathbf{m}_{t-1})^T(\mathbf{\Sigma}_{t-1})^{-1}(\mathbf{c}_t - \mathbf{m}_{t-1})\}. \quad (1)$$

Here the pixel location variable $\mathbf{x}$ is ignored for conciseness. If this probability is above a certain threshold, e.g., $p_t > 0.2$, the new pixel will be used to update the background model as:

$$\begin{aligned} \mathbf{\Sigma}_t &= \alpha(\mathbf{c}_t - \mathbf{m}_{t-1})(\mathbf{c}_t - \mathbf{m}_{t-1})^T + (1-\alpha)\mathbf{\Sigma}_{t-1} \quad (2) \\ \mathbf{m}_t &= \alpha\mathbf{c}_t + (1-\alpha)\mathbf{m}_{t-1}, \quad (3) \end{aligned}$$

where α is a decay factor indicating the updating rate. The above updating mechanism can handle slow background variations very well.

Another functionality we need in background modeling is the ability to push a pixel into the background if its color does not change for a long period of time. To enable this, we also compute a running mean $\mu_t(\mathbf{x})$ and a running variance $\mathbf{\Omega}_t(\mathbf{x})$ for each pixel. Whenever a new frame comes in, these running means and variances are updated similarly as:

$$\begin{aligned} \mathbf{\Omega}_t &= \alpha(\mathbf{c}_t - \mu_{t-1})(\mathbf{c}_t - \mu_{t-1})^T + (1-\alpha)\mathbf{\Omega}_{t-1} \quad (4) \\ \mu_t &= \alpha\mathbf{c}_t + (1-\alpha)\mu_{t-1}. \quad (5) \end{aligned}$$

Initially we let $\mu_0(\mathbf{x}) = \mathbf{c}_0(\mathbf{x})$ and $\mathbf{\Omega}_0(\mathbf{x}) = \rho\mathbf{I}$, where ρ is a big number (e.g., 20). If a pixel's color remains constant for a long period, the trace of the covariance matrix $\mathbf{\Omega}_t$ will decrease. If the trace is smaller than a certain threshold, the pixel will be push into the background, i.e., we set $\mathbf{m}_t(\mathbf{x}) = \mu_t(\mathbf{x})$ and $\mathbf{\Sigma}_t(\mathbf{x}) = \mathbf{\Omega}_t(\mathbf{x})$. Note a similar mechanism was adopted in [14] for surveillance applications.

3. FACE DETECTION AND TRACKING

The background modeling algorithm in the last section works reasonably well if the foreground person is constantly moving. Unfortunately, many people do not move around all the time during a video conferencing session. When the person stays still for a while, the above algorithm will gradually merge the foreground pixels into the background, generating a blurry foreground person. While there exist more sophisticated algorithms for background modeling [13], they suffer from the same problem inherently.

We observe that in video conferencing applications, the face is by far the most important foreground object that should always be in focus. Therefore we adopt a face detector and a face tracker to identify the face region to help remedy the above mentioned problem. The face detector is based on the work in [15], which is a three-step detector consisting of a linear pre-filter, a boosting chain and a number of post filtering algorithms such as support vector machine and color filters. The detector has a high detection rate for frontal faces with low false alarms thanks to the post-filters, however its detection rate on profile faces is relatively low and it is too expensive to run the detector on every video frame. Our solution is to combine the detector with a color-based non-rigid object tracker called pixel classification and integration (PCI) [16]. PCI has a few advantages over the popular mean-shift (MS) algorithm. It ensures a global optimal solution rather than a local optimal solution in MS. It is also computationally more efficient than MS, with better scale adaption and appearance modeling.

The key then is how to combine the face detector and the PCI tracker into a single framework that has good performance yet being efficient. We use the face detector as both a detector and a verifier for the process. If no face is detected in the scene, the detector will be fired once every second. Otherwise, it is used to verify the sub-image cropped by the tracked face twice a second. If a face is not verified for a number of tries, the detector is launched again for a whole image detection. The detected/tracked faces are expanded slightly up and down to cover the hair and the neck of the person. We then generate a face background likelihood map as:

$$f_t(\mathbf{x}) = \begin{cases} 0 & \text{if the pixel belongs to a face (expanded)} \\ 1 & \text{otherwise.} \end{cases} \quad (6)$$

That is, pixels belonging to a face have probability 0 as background, and 1 otherwise. In addition, if a pixel belongs to a face region, it will not be pushed into the background model no matter how small the trace of $\mathbf{\Omega}_t(\mathbf{x})$ is.

4. BACKGROUND BLURRING

The two background likelihood maps are combined into one for background blurring. Let:

$$q_t(\mathbf{x}) = \min(p_t(\mathbf{x}), f_t(\mathbf{x})). \quad (7)$$

We map this combined likelihood image into a blurring radius image as:

$$r_t(\mathbf{x}) = \begin{cases} \frac{q_t(\mathbf{x})}{\delta} r_{\max} & \text{if } q_t(\mathbf{x}) < \delta \\ r_{\max} & \text{otherwise} \end{cases} \quad (8)$$

where $r_{\max}$ is the maximum blurring radius set by the user, δ is a small thresholding probability. If $q_t(\mathbf{x})$ is greater than δ, the pixel

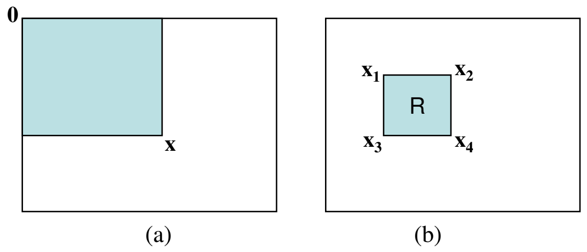

Fig. 2. The integral image. (a) Calculating the integral image (Equation 9). (b) The sum of colors in region R is computed as $\sum_{\mathbf{x}\in R} c_t(\mathbf{x}) = C_t(\mathbf{x}_4) + C_t(\mathbf{x}_1) - C_t(\mathbf{x}_2) - C_t(\mathbf{x}_3)$.

will be fully blurred. We find $\delta = 0.01$ works well. The blurring radius image is then used to blur the original video input. That is, for each pixel, we take the corresponding blurring radius and blur the input image by averaging pixels within the radius. Various kernels can be used such as Gaussian or rectangular kernels.

One challenge we face during the blurring process is that it can be very expensive, because each pixel can have a different blurring radius. When the maximum radius $r_{\max}$ is large, the adaptive blurring procedure can be very slow. Fortunately, for certain blurring kernels such as the rectangular kernel, this procedure can be greatly sped up with the help of integral images [17]. As shown in Figure 2, the integral image is calculated as:

$$C_t(\mathbf{x}) = \sum_{\mathbf{z}\in R(\mathbf{0},\mathbf{x})} c_t(\mathbf{z}) \tag{9}$$

where $R(\mathbf{0}, \mathbf{x})$ is the rectangular region formed by the origin and $\mathbf{x}$, as shown in the shaded region in Figure 2(a). The computational cost for calculating the integral image is low – two additions for each pixel in the image [17].

After the integral image has been calculated, the sum of colors in an arbitrary rectangular region can be computed with 3 additions, as shown in Figure 2(b). The blurred pixel is thus the sum of pixels within the radius divided by the size of the rectangular region, which can be computed efficiently for arbitrary size of neighbors.

Two example frames illustrating the background blurring process are shown in Figure 3. They are taken from the same sequence but at different time instances. Figure 3(i) is at frame #488, where the person is moving around a lot. The background modeling works very well in this case, producing almost identical results as the proposed approach. Figure 3(ii) is at frame #1100, where the person has been still for a long while. It can be seen in (ii-b) that the background modeling algorithm will classify most of the pixels as background, which blurs both the foreground and the background, as shown in (ii-c). This is unwanted and prevented by the proposed approach, as shown in (ii-d) to (ii-f). Note the original likelihood map is very noisy due to sensor noises. We blur the radius image with a 21×21 box blurring filter, which results in a much smoother radius image as shown in (ii-e). The end result (ii-f) still looks very good.

5. EXPERIMENTAL RESULTS

The above proposed algorithm has been tested under a wide range of conditions, e.g., moving vs. still foreground objects, lighting changes, frontal vs. non-frontal faces, single vs. multiple persons, etc. A number of example sequences are shown in Figure 4.

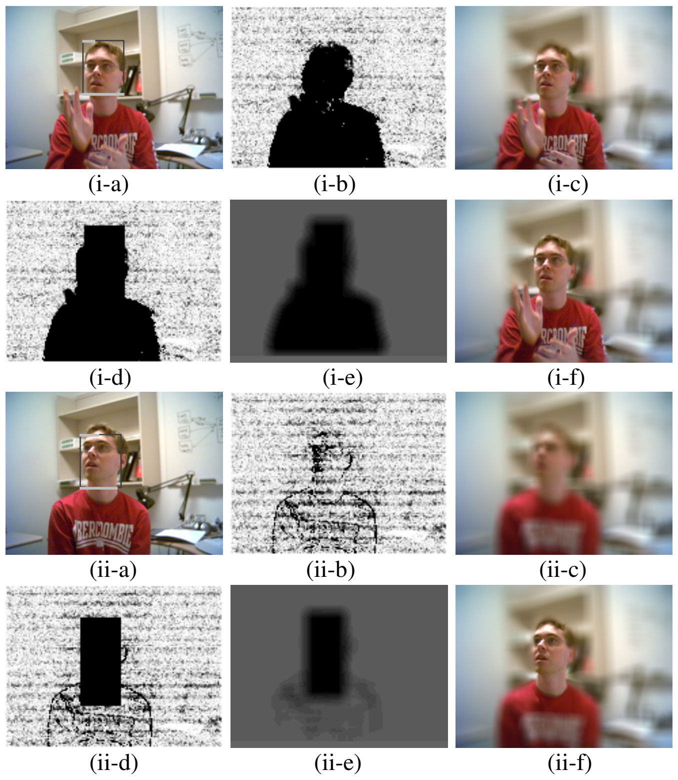

Fig. 3. Two examples of the background blurring process. (i) frame #488, where the person is moving around. (ii) frame #1100, where the person has been still for a while. (a) Original frame and the face detection/tracking results. (b) Background likelihood if face detection/tracking is not enabled. (c) Blurring result using the likelihood map in (b). (d) Background likelihood computed as Equation 7, where the face detector/tracker has been considered. (e) The actual blurring radius map computed as Equation 8 and smoothed. (f) Blurring result generated from (e).

In Figure 4, sequence (a) and (b) have relatively cluttered background, but the foreground persons do not move much. It is noticeable that the body of the foreground persons are blurred after a long period of time, however, the faces stay focused thanks to the integrated face detector/tracker. This creates a dreamy effect, which can emphasize the important role of face expressions during a conversation and deemphasize the motionless body and background.

Sequence (c) is an extended sequence that contains lighting variations, non-frontal faces and large motions. Frame #601 has a non-frontal face. Around frame #758, the person turns off one of the lights for a while. The algorithm quickly adapts to such lighting changes and the background stays blurred during the process. From frame #896 to #1090 the person moves dramatically, yet the face tracker locks on the person's head very well and the background blurring during this period is very successful.

Sequence (d) is an example to show that our algorithm can handle multiple persons in the scene with ease. The only change is in Equation 6, where multiple faces will be masked out instead of one. Sequence (e) is another challenging example with cluttered background and large motion. Again the algorithm produces reasonable results.

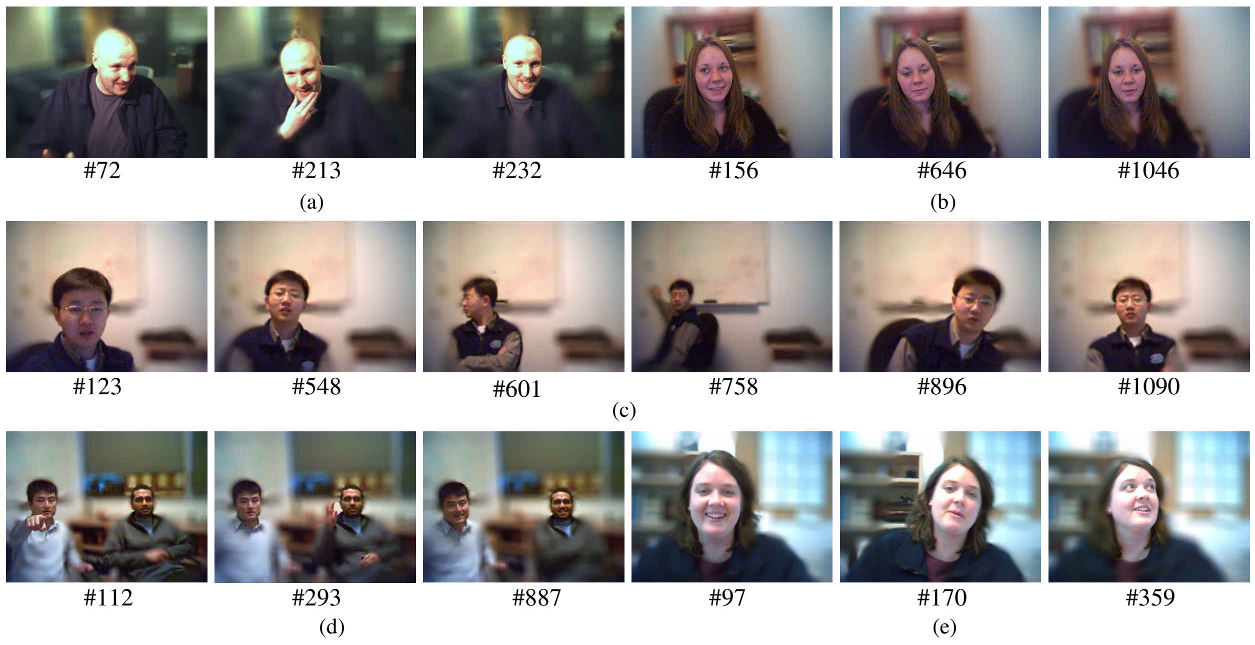

Fig. 4. Results for background blurring in various sequences. Sequences are labeled (a)-(e).

The proposed algorithm can fail under certain conditions, such as sequence (e) frame #170 in Figure 4. In this frame the person suddenly moves to the right and unoccluded a large portion of the background. This region is not blurred because the algorithm has never seen it before. This can be remedied if there is an extended period before the blurring process during which the camera can learn a static background model. On the other hand, we choose not to assume a static background model because it is inconvenient for the user to get such a model, and such a model can easily fail if the camera is shaken or the background objects are moved.

6. CONCLUSIONS

This paper proposed a light weight real-time background blurring framework for video conferencing applications. By combining a fast background modeling algorithm and face detection/tracking, we create an efficient background blurring algorithm which works on a wide range of conditions. Note the algorithm does not assume an extended period to learn the background. Rather, the background model is built during the process, which makes the system very practical to use.

7. REFERENCES

[1] Logitech Inc. http://www.logitech.com/.

[2] E. H. Adelson and J. Y. A. Wang, "Representing Moving Images with Layers,", *IEEE Trans. on Image Processing*, Vol. 3, No. 5, pp.625-638, 1994.

[3] J. J. Xiao and M. Shah, "Motion Layer Extraction in the Presence of Occlusion using Graph Cut,", *CVPR 2004*.

[4] C. Wren, A. Azarbayejani, T. Darrel and A. Pentland, "Pfinder: Real Time Tracking of the Human Body,", *IEEE Trans. on PAMI*, Vol. 19, No. 7, pp.780-785, 1997.

[5] H. Luo and A. Eleftheriadis, "Model-Based Segmentation and Tracking of Head-and-Shoulder Video Objects for Real Time Multimedia Services," *IEEE Trans. on Multimedia*, Vol. 5, No. 3, pp.379–389, Sep. 2003.

[6] V. Kolmogorov, A. Criminisi, A. Blake, G. Cross and C. Rother, "Bi-layer Segmentation of Binocular Stereo Video,", *CVPR 2005*.

[7] C. Stauffer and W. E. L. Grimson, "Adaptive Background Mixture Models for Real-Time Tracking," *CVPR 1999*.

[8] N. Friedman and S. Russell, "Image Segmentation in Video Sequences: A Probabilistic Approach,", *Proc. of the 13th Conf. on Uncertainty in Artificial Intelligence 1997*.

[9] A. Elgammal, D. Harwood, and L. S. Davis, "Non-Parametric Model for Background Subtraction," *ECCV 2000*.

[10] A. Mittal and N. Paragios, "Motion-Based Background Subtraction Using Adaptive Kernel Density Estimation," *CVPR 2004*.

[11] J. Rittscher, J. Kato, S. Joga, and A. Blake, "A Probabilistic Background Model for Tracking," *ECCV 2000*.

[12] M. Harville, "A Framework for High-Level Feedback to Adaptive Per-Pixel, Mixture-of-Gaussian Background Models," *ECCV 2002*.

[13] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers, "Wallflower: Principles and Practice of Background Maintenance," *ICCV 1999*.

[14] J. Connell, A.W. Senior, A. Hampapur, Y.-L. Tian, L. Brown and S. Pankanti, "Detection and Tracking in the IBM PeopleVision System", *ICME 2004*.

[15] R. Xiao, M.-J. Li and H.-J. Zhang, "Robust Multipose Face Detection in Images," *IEEE Trans. on CSVT*, Vol. 14, No. 1, pp.31-41, Jan. 2004.

[16] C. Zhang and Y. Rui, "Robust Visual Tracking via Pixel Classification and Integration,", *ICPR 2006*.

[17] P. Viola and M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *CVPR 2001*.