SMART COMPOSITING: A REAL-TIME CONTENT-ADAPTIVE BLENDING METHOD FOR REMOTE VISUAL COLLABORATION

Wei Hong   April Slayden Mitchell   Mitchell Trott

HP Labs, Palo Alto, California 94034

ABSTRACT

This paper proposes a content-adaptive blending method, smart compositing, for displaying two overlapped video frames on the same screen while preserving the readability of both. A pixel-wise adaptive blending factor map is generated according to the edge and saturation information of the content of only the overlay frame. Using this blending factor map, regions of the overlay frame with edges or saturated color are assigned to be more opaque and the remaining regions are assigned to be more transparent. A halo is also created around the edges of the overlay content which enhances the edges and disambiguates them from the underlying frame. The proposed method is suitable for overlaying many different types of content (e.g. drawings, slides, texts, and pictures) and does not require any information (e.g., an opacity mask) from the application which generates the content. This method has low computational complexity and has been implemented in real-time.

Index Terms— Video compositing, content-adaptive, visual collaboration

1. INTRODUCTION

When collaborating via video conference, a local user needs to see both the video of the remote user and the shared content such as drawings, slides, images, source code, and web pages, etc. In conventional visual collaboration systems, the shared content and the remote user's video are often displayed separately or in a picture-in-picture fashion. In recent years, a few advanced visual collaboration systems [1, 2] are trying to overlap them to enable correct eye contact and gesture interaction. To preserve real-world ordering where shared content exists between the two users, the remote user should appear behind the shared content on the local screen. Thus, the video frame containing the shared content is referred as an overlay frame, and the video frame containing the remote user is referred as an underlying frame. When the two frames are overlaid on the same screen, a compositing mechanism is needed to make both of the frames naturally visible and legible to either user. If the application which generates the content could inform the compositing mechanism which parts of the content are relevant and thus should be opaque, i.e., providing an opacity mask, the compositing can be precisely done based on the given opacity mask. However, most existing applications do not provide such a mask, and it is not always possible to modify existing applications to produce needed opacity masks. Moreover, in some cases, the shared content may be a screen capture from another computer or a video from an external video input for which there is no way to obtain an opacity mask. Therefore, the compositing mechanism must analyze the shared content and determine the transparency of each element of the content without additional input. Since the shared content may be modified any time during a remote collaboration session by either user, the compositing mechanism must run in real-time to accommodate the content changes.

There are several existing methods that composite two video frames into a single frame. Alpha blending [3, 4] is the most well-known real-time technique for compositing two frames. It simply assumes that the overlay frame is uniformly transparent. However, there are several drawbacks of alpha blending. The readability of the content deteriorates because alpha blending creates ambiguity between the overlay frame and the underlying frame. It is hard for viewers to tell which part belongs to which frame. Also, the contrast and color saturation of both the overlay and underlying frames decreases. Another real-time compositing method is intensity-adaptive compositing [2] which sets the transparency to be proportional to the intensity of the overlay content. It works well for content with black backgrounds but fails for content with brighter backgrounds.

In recent years, several advanced methods have been introduced to achieve visually better compositing results. Multiblending [5] uses an emboss filter to find edges and blending the edges to the underlying frame by a linear light blending function, giving overlay content, especially palettes, a 3D effect thus making them more legible. This method also desaturates or remaps the color channels of the overlay palettes and blurs the underlying regions behind the palettes to disambiguate them. Multiblending works well for content which consists of only edges and texts, and does not contain important color information, e.g. palettes. For general content (e.g., images or slides), the 3D effect, desaturating, and color remapping destroy the overlay content. Blurring the underlying content is also not suitable for remote visual collaboration purpose because the gestures and eye contact on the underlying frame need to be shown clearly to each user and thus should not be blurred. The Free Space Transparency [6, 7] requires the application to know the types of content on the overlay and underlying frames. The content type, color, and spatial frequency are compared between the overlay and underlying content to determine a tile-based transparency map of the overlay frame and whether blur and desaturation are applied on obscured regions on the underlying frame. The Frame Space Transparency is not a blind method and thus cannot be easily generalized to a variety of content types. This tile-based method also creates a very coarse transparency map. Similar to Multiblending, blurring and desaturating the underlying frame are also not suitable for remote visual collaboration purpose. EdgeTop [8] locates edges of the remote user on the underlying frame by a Sobel edge detector and enhances these edges by rendering them in a single color. The background of the underlying frame is removed by a background removal algorithm. The single-color-edge presentation of the person is very suitable for sharing X-ray images or other monochrome images. But for general contents, the single-color edges look unnatural and distracting. In addition, the edge detection and background removal algorithms cannot be fully accurate, and thus can result in holes appearing on the remote user's body and enhancement of false edges. All of the

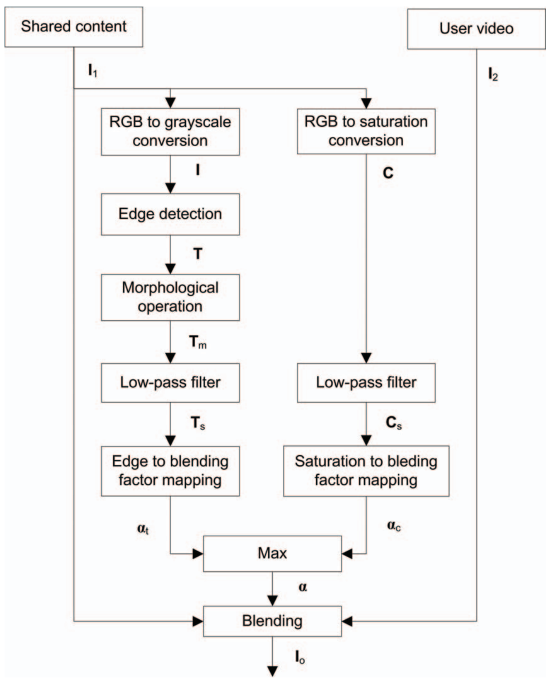


Fig. 1. Flow chart of the proposed method.

above advanced compositing methods have very high computational complexity and, therefore, are infeasible to be implemented in real-time.

In this paper, we propose smart compositing, a compositing method that works for any type of content (e.g. drawing, slides, texts, and pictures) without any prior knowledge (e.g. an opacity mask) of the content. Our method changes the transparency of every pixel on the overlay frame based on the edge and saturation information around that pixel. As a byproduct, a halo is created around edges of the overlay content which enhances the edges and disambiguate them from the underlying frame. The proposed method has low computational complexity and has been implemented in real-time.

2. PROPOSED METHOD

The block diagram of the proposed smart compositing is shown in Fig. 1. Ideally, both the overlay frame and the underlying frame can be analyzed to provide information for compositing the two frames. However, in order to minimize complexity, our method only analyzes the overlay frame which results in sufficient information for compositing the two frames. More specifically, a pixel-wise adaptive blending factor map is created according to the content of only the overlay frame. The purpose of the content-adaptive blending factor map is to designate unused regions of the overlay frame as more transparent and used regions as more opaque. We define used regions as the regions with edges or vibrant (saturated) color.

2.1. Edge information extraction

First, we extract edge information from the overlay frame. We denote the overlay frame as $\mathbf{I}_1(x, y)$ and the underlying frame as $\mathbf{I}_2(x, y)$. Without loss of generality, the input and output frames of our method are all in RGB format. The input overlay frame $\mathbf{I}_1(x, y)$ is converted from a RGB image to a monochrome intensity image $\mathbf{I}(x, y)$ for edge detection. An edge map $\mathbf{T}(x, y)$ of the image $\mathbf{I}(x, y)$ is created by an edge detection algorithm. Any popular edge detection algorithm such as Canny edge detector or Sobel operator can serve for this purpose. However, in our implementation, in order to reduce the complexity of the edge detection, a binary-value edge map is created by a simple edge detection algorithm described in Eq. 1.

For each pixel in the monochrome image $\mathbf{I}(x, y)$, the binary edge value is determined by comparing itself with its four surrounding pixels:

$$\mathbf{T}(x,y) = \begin{cases} 0 & \text{if } |\mathbf{I}(x,y) - \mathbf{I}(x-i, y-j)| \le \epsilon, \\ & \quad \forall i, j = -1, 1, \\ 1 & \text{otherwise,} \end{cases} \tag{1}$$

where ϵ is a predetermined threshold. For low computation complexity purpose, only four surrounding pixels are used. If computational resources are abundant, more surrounding pixels (diagonal pixels or pixels farther away from the center pixel) can be used to improve the quality of the edge map. If computational resources are scarce, only two surrounding pixels can be used instead of the four surrounding pixels to further reduce the computation complexity. The edge detection works well for edges but may have trouble for regions with sparse texture where holes may appear. Morphological operations, such as dilation operation or closing operation can be applied on the binary-value edge map $\mathbf{T}(x, y)$ to generate another binary-valued edge map $\mathbf{T}_m(x, y)$ without or with less holes inside sparse texture regions. Since most of the shared content (e.g. texts, slides, and drawings) is computer-generated, it lacks sparse texture regions and thus the morphological operation can be skipped if computational resources are scarce. Even for non-computer-generated shared content (e.g. images), holes can be avoided by choosing a small threshold ϵ which is below the noise level of the image.

A low-pass filter is applied on the binary-value edge map $\mathbf{T}(x, y)$ (or $\mathbf{T}_m(x, y)$ if a morphological operation has been applied) to create a continuous-value edge map $\mathbf{T}_s(x, y)$ with smooth transitions between flat and edge regions. The low-pass filter can be an averaging, Gaussian, or another type of filter. The size of the filter, which is chosen empirically, controls the width of the transition regions. A larger filter size creates smoother transition but is computationally more expensive.

The edge-adaptive blending factor map $\boldsymbol{\alpha}_t(x, y)$ is converted from the continuous-value edge map $\mathbf{T}_s(x, y)$ by a piece-wise linear mapping described as:

$$\boldsymbol{\alpha}_t(x,y) = \begin{cases} (\alpha_{t2} - \alpha_{t1})\, \mathbf{T}_s(x,y)/T_t + \alpha_{t1}, & \text{if } \mathbf{T}_s(x,y) < T_t, \\ \alpha_{t2} & \text{otherwise,} \end{cases} \tag{2}$$

where α_{t2} and α_{t1} are two user defined parameters for the maximum and minimum values of the edge-adaptive blending factor. $\mathbf{T}_t$ is also a user defined parameter that determines at which edge value the blending factor reaches its maximum. This mapping can be efficiently implemented by a lookup table. We choose this piece-wise linear function for lower complexity and easier parameter control, but other non-linear mapping functions would also work.

2.2. Saturation information extraction

In parallel with extracting the edge information, we extract saturation information from the overlay frame. The input overlay frame

$\mathbf{I}_1(x,y)$ is converted from an RGB image to single-channel saturation image, denoted as $\mathbf{C}(x,y)$. Similar to the low-pass filter applied to the binary-value edge map, a low-pass filter is also applied on the saturation image $\mathbf{C}(x,y)$ to create a smoothed saturation image $\mathbf{C}_s(x,y)$ with smooth transitions between saturated and non-saturated regions.

Similar to the edge-adaptive blending factor map in Eq. 2, the saturation-adaptive blending factor map $\boldsymbol{\alpha}_c(x,y)$ can be generated as:

$$\boldsymbol{\alpha}_c(x,y) = \begin{cases} (\alpha_{c2}-\alpha_{c1})\,\mathbf{C}_s(x,y)/T_c + \alpha_{c1}, & \text{if } \mathbf{C}_s(x,y) < T_c, \\ \alpha_{c2} & \text{otherwise,} \end{cases} \tag{3}$$

where α_{c2} and α_{c1} are two user defined parameters for the maximum and minimum values of the saturation-adaptive blending factor. T_c is also a user defined parameter that determines at which saturation level the blending factor reaches its maximum. This mapping can also be efficiently implemented by a lookup table.

2.3. Fusion of edge and saturation information

Since we want to make the regions with either edges or saturated color appear more opaque, we merge the edge-adaptive blending factor map $\boldsymbol{\alpha}_t(x,y)$ and the saturation-adaptive blending factor map $\boldsymbol{\alpha}_c(x,y)$ into the final blending factor map $\alpha(x,y)$ by choosing the larger value between them pixel-wise.

$$\boldsymbol{\alpha}_(x,y) = \max(\boldsymbol{\alpha}_t(x,y), \boldsymbol{\alpha}_c(x,y)). \tag{4}$$

The overlay frame $I_1(x,y)$ and the underlying frame $I_2(x,y)$ are blended by the blending factor map $\alpha_(x,y)$ to generate the output frame $I_o(x,y)$.

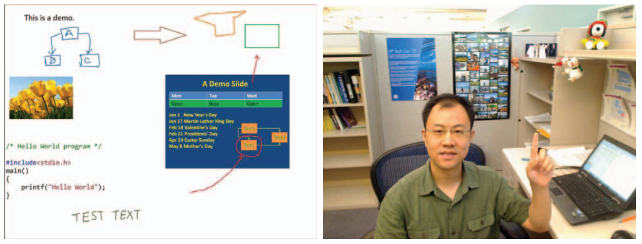
$$\mathbf{I}_o(x,y) = \boldsymbol{\alpha}(x,y)\mathbf{I}_1(x,y) + (\mathbf{1}-\boldsymbol{\alpha}(x,y))\mathbf{I}_2(x,y). \tag{5}$$

The proposed smart compositing can be used for any overlay content (e.g. drawing, slides, texts, and pictures) because it is a pure pixel-based approach. Since it does not require any application-specific information (e.g., an opacity mask), existing applications can be used with the proposed method without any modifications.

The proposed method also provides the flexibility to allow an application to change the transparency of certain regions by adding some random noise or dense pattern with small magnitude just above the threshold ϵ to the intended regions. This approach does not need the additional bandwidth required to carry an opacity mask and the content change is minimal due to the small magnitude of the noise or pattern.

3. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed method has been tested on many different types of content. Without loss of generality, we assume the pixel value is between 0 and 1. In our experiments, we chose ϵ to be 0 to keep all possible edges. We chose T_t to be 0.2, a_{t1} to be 0, a_{t2} to be 0.85, T_c to be 0.5, a_{c1} to be 0, and a_{c2} to be 0.8, based on our visual experiments to provide the best tradeoff between transparency and clarity of the shared content in order to produce the most aesthetically pleasing compositing result. We skipped the morphological operation and chose the low pass filter to be a cascade of two 5x5 averaging filters to improve performance. Since no processing is performed on the underlying frame and only simple filters and operations are applied on the overlay frame, the computational complexity of the proposed method is relative low making a real-time implementation possible. Most of the signal processing tasks are implemented using Intel Integrated Performance Primitives. In our performance test, compositing each frame takes 20.6ms on a 3.33GHz 6-core Xeon CPU for 1280x960 resolution, which is fast enough for a 30fps video. For many types of shared content (e.g., slides and pictures), since the shared content changes infrequently, the blending factor map $\boldsymbol{\alpha}(x,y)$ can be recalculated only when the shared content has changed in order to further reduce the computational complexity. A simple frame comparison can be used to detect the frame changes. The proposed method also requires low memory bandwidth because it only relies on information from the overlay frame to calculate the blending factor map as opposed to most of the existing advanced compositing methods which require information from both the overlay and underlying frames.



Mon	Tue	Wed
Open	Busy	Open

Jan 1 New Year's Day
Jan 17 Martin Luther King Day
Feb 14 Valentine's Day
Feb 21 Presidents' Day
Apr 24 Easter Sunday
May 8 Mother's Day

```
/* Hello World program */

#include<stdio.h>
main()
{
    printf("Hello World");
}
```

Fig. 2. Left: the testing overlay frame. Right: the testing underlying frame.

In order to test the effectiveness of the proposed smart compositing on different types of content, we created a testing overlay frame containing a slide, an image, a few drawings and texts as shown on the left of Fig. 2. The testing underlying frame shown on the right of Fig. 2 contains a person in a typical office environment. Fig. 3 shows the visual comparison of the results of several real-time compositing methods. Fig. 3 (a) shows the result of compositing with prior knowledge of the transparency (an opacity map) for all elements. In this experiment, all the text, drawings, and images are 85% opaque and all the solid color background boxes are 80% opaque which provides the best tradeoff between transparency and clarity of the shared content. This result can be treated as the ground truth of the expected compositing outcome as the overlay and underlying frame are both well exposed. But in certain areas, the background may have a similar color to that of the content over it making the content hard to read. For example, the blue rectangle at the top left and the green text at the bottom fade into the background color behind them in Fig. 3 (a). Fig. 3 (b) shows the result of alpha blending with an alpha of 0.5 to expose both the underlying and overlay frames as much as possible. The contrast of both the overlay and underlying frames deteriorates and everything looks washed out. Fig. 3 (c) shows the result of intensity-adaptive compositing [2]. This method fails for this shared content since the background of the content is not dark. The background has to be modified to be in black to achieve a good result. Fig. 3 (d) and (e) are the results of using only one leg of the proposed algorithm to obtain edge-adaptive compositing and saturation-adaptive compositing. In the edge-adaptive compositing, solid-color boxes become skeletons which lose the containment relationship with the content inside the boxes. In the saturation-adaptive compositing, content without color (e.g. black text at the top left corner in Fig. 3 (e)) almost completely disappears. Fig. 3 (f) shows

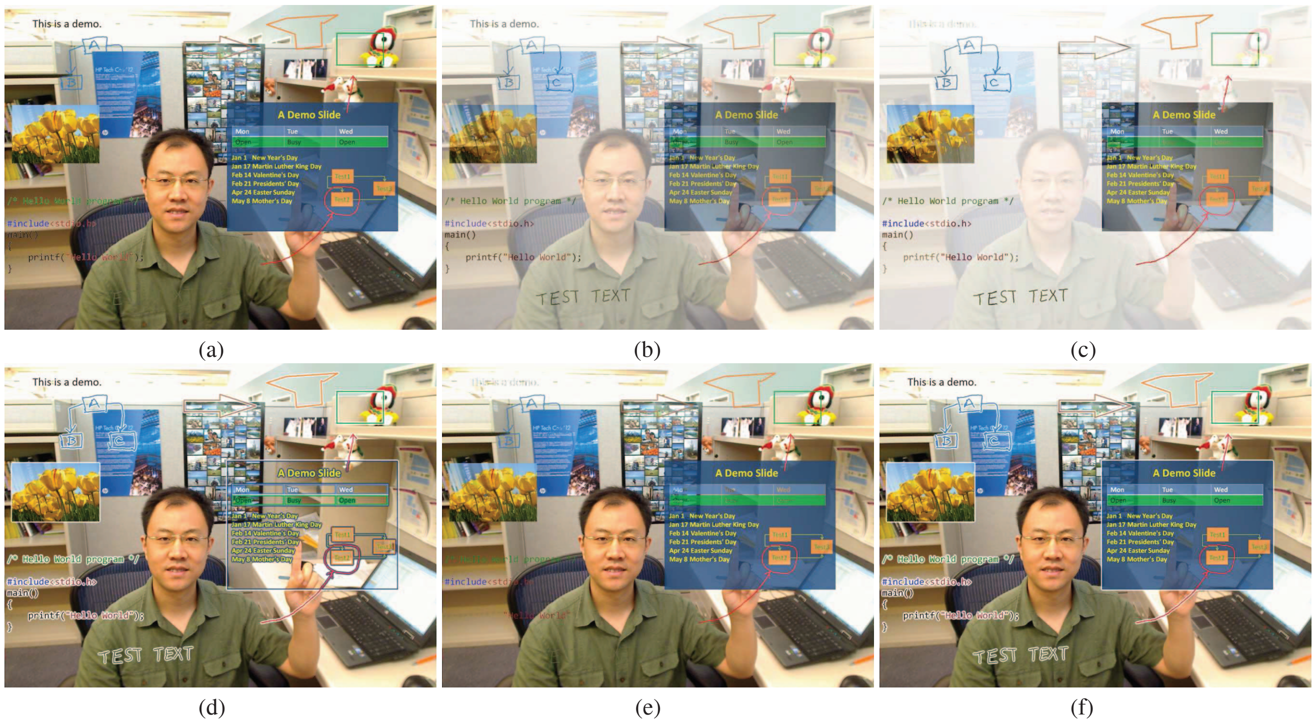


Fig. 3. The results of several real-time compositing methods. (a) Compositing with prior knowledge of the transparency of each element. (b) Alpha blending. (c) Intensity-adaptive compositing. (d) Edge-adaptive compositing. (e) Saturation-adaptive compositing. (f) Proposed smart compositing.

the result of the proposed smart compositing. The brightness, contrast, and color saturation of both the underlying user video and the overlay content are almost the same as those in Fig. 3 (a). The readability of the overlay frame is well preserved and even better than in Fig. 3 (a) because a halo is created around edges of the overlay content to enhance the edges and disambiguate them from the messy underlying frame. As you can see in Fig. 3 (f), the blue rectangle at the top left and the green text at the bottom stand out clearly from the background behind.

4. CONCLUSION

In this paper, we propose smart compositing, a real-time content-adaptive blending method, which allows both shared content and the remote user video to be legible simultaneously. Experimental results show that the proposed smart compositing can produce a preferred compositing result over the result with manually assigned opacity.

5. REFERENCES

[1] J. C. Tang and S. L. Minneman, "Videodraw: a video interface for collaborative drawing," *ACM Transactions on Information Systems (TOIS) - Special issue on computer–human interaction*, vol. 9, pp. 170–184, 1991.

[2] K.-H. Tan, D. Gelb, R. Samadani, I. Robinson, B. Culbertson, and J. Apostolopoulos, "Gaze awareness and interaction support in presentations," in *Proceedings of the international conference on Multimedia (ACM-MM '10)*, Firenze, Italy, 2010, ACM, pp. 643–646.

[3] D. Stotts, J. M. Smith, and K. Gyllstrom, "Facespace: endo- and exo-spatial hypermedia in the transparent video facetop," in *Proceedings of the Fifteenth ACM Conference on Hypertext & Hypermedia (HYPERTEXT '04)*, Eindhoven, The Netherlands, 2004, ACM, pp. 48–57.

[4] D. Stotts, J. M. Smith, and K. Gyllstrom, "Support for distributed pair programming in the transparent video facetop," *Lecture Notes in Computer Science*, vol. 3134/2004, pp. 150–192, 2004.

[5] P. Baudisch and C. Gutwin, "Multiblending: displaying overlapping windows simultaneously without the drawbacks of alpha blending," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '04)*, Vancouver, B.C., Canada, 2004, ACM, pp. 367–374.

[6] E. W. Ishak and S. K. Feiner, "Free-space transparency: exposing hidden content through unimportant screen space," in *Proceedings of 16th Annual ACM Symposium on User Interface Software and Technology (UIST '03)*, Vancouver, B. C., Canada, 2003, ACM, pp. 75–76.

[7] E. W. Ishak and S. K. Feiner, "Interacting with hidden content using content-aware free-space transparency," in *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology (UIST '04)*, Santa Fe, NM, USA, 2004, ACM, pp. 189–192.

[8] K. Gyllstrom, D. Miller, and D. Stotts, "Techniques for improving the visibility and "sharability" of semi-transparent video in shared workspaces," in *Proceedings of the 45th Annual Southeast Regional Conference (ACM-SE '45)*, Winston-Salem, North Carolina, USA, 2007, ACM, pp. 425–430.