Tile-Image Merging and Delivering for Virtual Camera Services on Tiled-Display for Real-Time Remote Collaboration*

Giseok CHOE†, Nonmember and Jongho NANG†a), Member

SUMMARY The tiled-display system has been used as a Computer Supported Cooperative Work (CSCW) environment, in which multiple local (and/or remote) participants cooperate using some shared applications whose outputs are displayed on a large-scale and high-resolution tiled-display, which is controlled by a cluster of PC’s, one PC per display. In order to make the collaboration effective, each remote participant should be aware of all CSCW activities on the tiled display system in real-time. This paper presents a capturing and delivering mechanism of all activities on tiled-display system to remote participants in real-time. In the proposed mechanism, the screen images of all PC’s are periodically captured and delivered to the Merging Server that maintains separate buffers to store the captured images from the PCs. The mechanism selects one tile image from each buffer, merges the images to make a screen shot of the whole tiled-display, clips a Region of Interest (ROI), compresses and streams it to remote participants in real-time. A technical challenge in the proposed mechanism is how to select a set of tile images, one from each buffer, for merging so that the tile images displayed at the same time on the tiled-display can be properly merged together. This paper presents three selection algorithms; a sequential selection algorithm, a capturing time based algorithm, and a capturing time and visual consistency based algorithm. It also proposes a mechanism of providing several virtual cameras on tiled-display system to remote participants by concurrently clipping several different ROI’s from the same merged tiled-display images, and delivering them after compressing with video encoders requested by the remote participants. By interactively changing and resizing his/her own ROI, a remote participant can check the activities on the tiled-display effectively. Experiments on a 3×2 tiled-display system show that the proposed merging algorithm can build a tiled-display image stream synchronously, and the ROI-based clipping and delivering mechanism can provide individual views on the tiled-display system to multiple remote participants in real-time.

key words: tiled-display system, remote collaboration, real-time video capture and streaming, CSCW, remote virtual camera

1. Introduction

A tiled-display system consists of multiple display devices linked in a grid configuration to provide a very large display with very high resolution, where each display device is controlled by an independent computer such as a PC. In order to provide a consistent image that is spread over several displays and refreshed periodically, the rendering and displaying task should be temporally synchronized. Many researchers sought to develop such parallel rendering and synchronization mechanisms for tiled-display system. Notable examples are Softgenlock[2], TerraVision[16], Scalable Adaptive Graphic Environment (SAGE)[19], WireGL[11], and Garuda[20]. Since these systems can render and display several (virtual) objects simultaneously on a tiled-display and these objects are generated by different applications operated by local or remote participants, they are widely used as a collaboration tool in virtual space, for example, a multi-party collaborative environment based on Access Grid[9].

In order to extend these systems as an effective remote collaboration tool, CSCW activities on a tiled-display, such as rendering, moving, and resizing of the (virtual) objects, should be delivered to remote participants accurately. One way of delivering these activities is to capture the rendering events on a tiled-display generated by CSCW applications, and deliver them to remote participants to be displayed on their own displays by the same applications. However, since this solution requires that the remote participants have exactly the same hardware (i.e., a tiled-display hardware) and software platforms that are usually very expensive, it is not a practical approach. Another way is to capture the whole screen of a tiled-display with a video camera and broadcast it to remote participants in a compressed form. This solution is a simple approach and is widely used in remote collaborations[6],[10] (or remote lectures[15],[18]), in which a white board’s contents or projected presentation images are automatically captured by a video camera and broadcast to remote participants. However, it cannot be applied to tiled-display systems because detailed information on a tiled-display cannot be delivered accurately by a video camera. For example, where the resolution of tiled-display consists of 3×3 displays each having 1,280×1,024 resolution, the total resolution is 3,840×3,072 (11,796,480 pixels). Here, the detailed information on the tiled-display cannot be delivered to remote participants even when a full HD video recorder (1,920×1,080 = 2,073,600 pixels) is used for recording and broadcasting. Of course, the detailed information may be delivered to remote participants if the zoom-in function of the video recorder is used. However, this approach still fails to meet the requirements because each remote participant usually wants to see different parts of a tiled-display but only one video recorder is usually used for capturing the collaboration process. How to effectively deliver the CSCW activities on a tiled-display to remote participants in real-time would be the key technology to extend...
the tiled-display system as a remote collaboration tool.

This paper presents an effective mechanism for delivering the CSCW activities on a tiled-display to remote participants regardless of the applications used to realize the collaboration and virtual object rendering. A widely used approach for recording the activities of an application running on a PC is to capture the video frame buffers (and/or audio buffers) periodically and make them as a video stream. This approach is used in famous screen recording tools such as CamStudio [3] and ACA Screen Recorder [1]. The mechanism proposed in this paper extends this approach so that the tiled-display screen images are periodically captured and delivered to remote participants as a compressed video stream. In the proposed mechanism, each PC controlling a tile display captures its video frame buffer (i.e., makes a tile image) and sends it to a Merging Server, and the Merging Server maintains separate buffers to store a sequence of the tile images from the PCs. It selects one tile image from each buffer, and builds a tiled-display screen image by assembling the images based on their coordinates on the tiled-display system. This process is periodically executed to make a sequence of tiled-display screen images, whereupon the images are compressed and delivered to remote participants after clipping out the ROI.

An interesting technical challenge in this approach is how to build a screen shot (or image) of the tiled-display consisting of tile images that are displayed at the same time on the tiled-display. It is not a trivial problem because the capturing process is controlled by independent PCs, which usually have different CPU loads, so that the capturing time cannot be the same although a periodic capturing based on a global timer is used. This paper presents three selection algorithms that select one tile image from each buffer, where the selection criteria are different by algorithms. The selection criteria are based on the sequence of tile images, based on the capturing time of tile images, and finally based on the visual consistency as well as the capturing time of tile images. This paper also proposes a mechanism that provides a virtual camera to each remote participant. Since multiple ROI’s can be clipped concurrently from the same merged tile-display image and each ROI can be dynamically moved and resized by a remote participant independently, the remote participants can inspect the areas of tile-display they wish to see in detail. This mechanism enables the remote participant to carefully check the CSCW activities on the tiled-display system. Experimental results on a 3 × 2 tiled-display system show that the proposed merging algorithm can build a video stream of merged images synchronously, and provide multiple virtual cameras to remote participants in real-time. The proposed system will be helpful to extend a tiled-display system as an effective remote collaboration tool.

2. Related Works

Tiled-display systems have been used to render complicated graphical objects or high resolution images on a large and high resolution display. Lately, they are extended as effective collaboration tools for remote participants. The display hardware can be CAVE (CAVE Automatic Virtual Environment), multiple-monitor desktops, tiled LCD panels, projector arrays, stereoscopic displays, or volumetric displays [14]. Most software-related studies on tiled-display systems have focused on how to distribute the rendering tasks to multiple computers (usually PCs), and how to synchronously display the rendering results on a tiled-display. Existing parallel rendering models on tiled-display may be classified as either client-server models, in which a user interacts with a single instance of the application that runs on a client node, where the node generates the geometry and distributes it to rendering servers, or master-slave models, in which the application executes on every node and a master node handles all user interactions and synchronizes the state changes among all other nodes [14].

The synchronization mechanisms widely used in these models are to send (or broadcast) a special “SYNC” packet to other nodes via a TCP out-of-band data channel [19] or a control signal using a parallel port [2]. Another synchronization mechanism used for the parallel decoding/displaying of the compressed video on tiled-display system is to use the hardware clock of the decoding/displaying node [4], where the decoding/displaying nodes only reference their hardware clocks for displaying the video frame after synchronization of the start times of all nodes. Since the displaying video frame is periodic and the errors in hardware clocks are negligible, display of video frames on a tiled-display can be temporally synchronized as shown in [4]. All such systems focus on how to display the virtual objects or video frames synchronously on a tiled-display. However, the algorithm proposed in this paper concerns how to capture and merge the rendering results on a tiled-display synchronously so that the rendering results displayed on tiled-display at the same time are delivered to remote participants accurately.

Recently, Visualcasting [12] was proposed to extend the tiled-display for remote collaboration in the tiled-display environments. In this research, rendered images of visual applications are independently multicast and displayed over remote tiled-displays to enable sharing of their visual outputs among remote participants. However, since the sizes and positions of the remote windows displaying multicast visual outputs over remote tiled-displays are different from one another, remote participants cannot have the same view on their tiled-displays. This condition violates the famous principle called WYSIWIS (What You See Is What I See) [17] for effective collaboration. Furthermore, a maximum of \( k \times n \) communication sessions need to be maintained when \( n \) remote participants collaborate using \( k \) visual applications because all rendered images of visual applications are multicast to all remote participants independently. This implies that Visualcasting can only be used in an environment where an ultra-high speed network is provided even if the visual outputs are compressed for multicasting. In addition, the applicability of Visualcasting...
is limited to collaboration environments where all remote participants are equipped with a tiled-display system. On the other hand, since, in our approach, the tiled-display is regarded as a shared white board displaying all rendered images of running visual applications and it is delivered to the remote participants, all participants can have the same view on their tiled-displays. In addition, our approach requires only \( n \) communication sessions for remote collaboration regardless of the number of visual applications, because all rendered images of visual applications are synchronously displayed together over the tiled-display and its contents are captured and delivered to remote participants. Furthermore, the applicability of our approach is broader than Visualcasting because there is no limitation on the remote displays for collaboration.

3. Design of a Remote Collaboration Tool for Tiled-Display System

For a tiled-display system to be extended as an effective collaboration tool, CSCW activities, such as rendering, moving and resizing events on virtual objects, need to be shared accurately among remote participants. The sharing mechanism proposed in this paper is to provide a virtual camera on the tile-display (a shared workspace), showing the visual contents of tile-display, which are the final results of CSCW activities, to remote participants. By changing the position and size of the virtual camera view, each remote participant can look around the visual contents of the tiled-display with his/her own view. This helps the participants understand the CSCW activities over the tiled-display. The virtual camera can be implemented by capturing and merging the visual content of each tile image and delivering the (part of) merged tile-image to remote participants. Figure 1 shows an example of the proposed system in which each PC periodically captures and sends its video frame buffer† to the Merging Server, and the Merging Server assembles, clips, compresses, and streams the contents of the tile-display to two remote participants. The Merging Server can also store locally the sequence of whole screen images of tiled-display as a compressed video stream for a collaboration report, as shown in Fig. 1. Figure 2 shows an internal architecture of the proposed system in more detail. After the tile images are selected and assembled by “Screen Merging” engine in Fig. 2, a full-screen image of the tiled-display can be reconstructed at the Merging Server. The full-screen image should be buffered before being fed into the adaptation/encoding engine because the speeds of the merging process and adaptation/encoding processes can be different and vary over time. “Buffers for Merged Tiles and Mixed Audio” in Fig. 2 are used for alleviating these differences and variations. The following sections describe the component processes in Fig. 2. First, Sect. 3.1 describes how to synchronously capture the video frame at each PC in a temporally global manner. Since the system loads of PCs are usually different, the capturing process cannot be temporally synchronized exactly. Section 3.2 describes how to alleviate this skew among captured tile images when merging them to build a screen shot of the tiled-display. Finally, Sect. 3.3 describes how to provide a virtual camera on tiled-display to remote participants by clipping the ROI.

3.1 Periodical Capturing of Tile Images

In order to capture the screen images displayed in the tiles, each PC (or capturing slave) controlling the individual tile display reads its video frame, resizes it to reduce the communication overheads if required, and finally sends the resized frame to a Merging Server with its global time stamp that indicates when the tile image was captured. This task is executed periodically to generate a sequence of tile images. Figure 3 shows the capturing procedure of a PC controlling individual tile display.

When a session is initialized, the timers in capturing slaves and the Merging Server are first synchronized using Cristian’s algorithm[5], which is primarily used for clock synchronization in low latency intranets. Then, the capturing slaves start the loop of capturing, resizing, sending, and waiting tasks. Since two timers are incremented by their respective hardware clocks after they are synchronized and the errors in the hardware clocks are negligible, the capturing processes of all slaves can be theoretically synchronized without repeated synchronization messages. Therefore, it is expected that the global time stamps of captured tile images, which are captured at the same period, are the same. However, it is not true in real environments. Let us explain the reasons and describe our proposed solution below.

At the capturing slaves, the capturing task is executed every \( 1/f \) second where \( f \) is the pre-defined capturing rate, and usually implemented as a callback function. However, this callback function cannot be executed at every \( 1/f \) second exactly if other real-time processes, such as video renderers, are concurrently running on the slaves. Figure 4 shows the actual elapsed times of capturing/resizing/YUV_encoding/sending tasks on slaves under WindowsXP. The elapsed time of these tasks is about 24 ms when it is running alone, and about 30 ms~40 ms when other three video decoder/renderers are running concurrently. It means that the capturing process cannot be temporally synchronized exactly with other PC’s although a global timer-based synchronization mechanism is used because the CPU loads of the PC’s controlling the tiles are usually different. The Merging Server should provide a mechanism to alleviate this skew among the captured tile images when building a screen image of a tiled-display. Let us describe this mechanism in the following section.

3.2 Merging Captured Tile Images

Let \( n \) be the number of tiles (or capturing slaves) in the tiled-

†In this paper, we only focus on capturing and delivering of the screen images of tiled-display because the audio activities such as speech and sound effects can also be captured from the buffers in the sound card and delivered in the same way.
Fig. 1  An example of tiled-display system as a remote collaboration tool.

Fig. 2  System architecture of the remote collaboration tool for tiled-display system.

Fig. 3  Internal algorithm of capturing slave.

procedure CapturingSlave(\( S_k \)) {
    Synchronize the Global Timer using Cristian's algorithm [9];
    // Periodically capture, resize, and send the captured tile image with its time stamp
    while (not END_OF_SESSION()) {
        Capture the tile image by reading the video frame buffer; //
        Resize the captured tile image if required;
        Send the captured tile image to Merging Server with its time stamp;
        Wait until the start of next period;
    }
}
The Merging Server maintains $n$ buffers, each of which keeps a sequence of captured tile images that are captured and delivered from the capturing slaves. Figure 5 presents an internal algorithm of the Merging Server. After synchronizing the global time stamps with all slaves, the Merging Server starts two tasks (receiving the tile im-

```plaintext
procedure MergingServer() {  
    Synchronize the Global Timer using Cristian’s algorithm [9];  
    parbegin 
    parfor all slaves, $S_i (1 \leq i \leq n)$, participating the session do 
        while (not END_OF_SESSION()) 
            Receive the captured tile image from slave $S_i$;  
            Save the received tile image to buffer $B_i$; 
        end_while 
    end_parfor 
    // Select the tile images, merge them, clip ROI, compress, and deliver 
    do 
        Sleep($\alpha$); // $\alpha$ is the initial buffering time 
        while (not END_OF_SESSION()) 
            parbegine 
                Select a set of tile images (one tile image from each buffer);  
                Merge the selected tile images to build a screen-shot of tiled-display;  
                Save the merged tile image to the buffer;  
                Wait until the start of next period; 
            end_do 
            do 
                Get a merged-tile image from buffer; 
                parfor all remote participants, $R_i (1 \leq i \leq k)$ do // $k$ is the number of participants 
                    Clip the ROI for $R_i$ from the merged tile image, and compress/deliver; 
                end_parfor 
                Wait until the start of next period; 
            end_do 
        end_parbegine 
    end_while 
    end_do 
} 
```

Fig. 4 Experimental capturing time on slaves under WindowsXP.

Fig. 5 Internal algorithm of merging server.
ages and merging them) concurrently. In the receiving task, it receives the captured tile images from all slaves concurrently and saves them to their respective buffers. When starting the merging task, it waits for a little while (α is the parameter to control this latency) to initially buffer the captured tiled images. After the initial buffering, it selects a set of n tile images for the j-th merging period from n buffers (one tile image from each buffer), and assembles them to build a screen-shot of the tiled-display. After clipping the ROI’s of all participants from the merged tile-image, it compresses and delivers them to remote participants concurrently. Then, it waits until the end of the j-th period. It iterates these selecting, assembling, clipping, compressing, and delivering tasks until the end of a remote collaboration session. The technical challenge in this algorithm is how to select a set of tile images from these buffers so as to minimize the skews and jitters in a sequence of merged tiled-display images. Let us describe the proposed selection algorithms in more detail.

A simple selection algorithm would be to sequentially select one tile image from each buffer one by one, and merge them to build a merged tile-image as shown in Fig. 6.

Theoretically, the time stamps of all tile images in $A_j$ selected by the algorithm in Fig. 6 should be the same, because the capturing tasks at all slaves are controlled by the synchronized global time clock. However, in reality, these time stamps are different because the actual capturing times are influenced by the workloads of the capturing slaves as shown in Fig. 4. Consequently, the intervals between the time stamps of two successive captured tile images are usually irregular, so that the simple merging algorithm shown in Fig. 6 cannot generate a sequence of visually consistent merged tiled-display images, especially when an object spreading over several displays is changing its appearance.

A more intelligent algorithm is to select the tile images from each buffer whose time stamps are closest to its target time stamp, as shown in Fig. 7. It can generate a sequence of visually consistent merged tiled-display images because it selects the tile images based on their actual capturing times rather than their sequence in the buffers.

Unfortunately, the selection algorithm based on the actual capturing time shown in Fig. 7 still has the possibility of generating a visually inconsistent tiled-display image if a visual object spreading over the tiled-display changes its appearance dynamically when the slave captures its video frame buffer. This problem can be resolved by checking the visual contents as well as the capturing time stamps of tile images when selecting tile images from the buffers. That is, a set of tile images are selected so that their visual contents are similar to each other and their capturing time stamps are close to the target time stamp. An algorithm to select a set of such tile images is presented in Fig. 8, in which

\[
\text{procedure SelectTileImages}_3(target\_time\_stamp) \{
\text{for all buffers, } B_j[(l \leq i \leq n)] \text{ do}
\text{Select the tile image whose time stamp is closest to the target\_time\_stamp;}
\text{end for}
\text{Return the selected tile images;}
\}
\]

Fig. 8 A selection algorithm based on actual capturing time and visual contents.
the differences between the edge pixel values of neighboring tiles are additionally used to find a visually consistent set of tile images. The algorithm first selects a set of tile images based on the actual capturing times of tile images as the algorithm in Fig. 7, then secondly the algorithm checks the visual consistency of the tile images in this set. The visual consistency of the tile images is checked by comparing the values of two pixels that are adjacent in the merged tiled-display image but belonging to different tile images. Note that AVGEdgePixelDiff($I_i, A$) in Fig. 8 is a function that returns the average of all differences between boundary pixels in the tile image $I_i$ and their corresponding (or adjacent) boundary pixels of other tile images in $A$.

3.3 Providing Individual Virtual Camera on Tiled-Display to Remote Participants

The merged tiled-display images are buffered in the Merging Server, and they are delivered to remote participants. This CSCW scenario is explained in Fig. 9. Let us now describe this scenario in more detail.

A remote participant specifies an ROI that indicates the area that he/she wants to see in the tiled-display. The Merging Server clips and resizes the ROI from the merged tiled-display image, and iterates this task for all merged tiled-display images to generate a stream of ROI’s. This stream is delivered to remote participants after compression using a specified video encoder. Since multiple streams of ROI’s can be clipped from the stream of merged tiled-display images, multiple remote participants can be serviced concurrently. By changing the size and position of the ROI, a remote participant can view the area of tiled-display image that he/she wants to observe. This implies that the proposed system can provide an individual virtual camera to each of the participants. Figure 10 shows an example of what a remote participant can see at his/her display by changing his/her ROI.

3.4 Theoretical Scalability Analysis

The computing pipeline proposed in this paper can be abstracted as shown in Fig. 11. Here, all captured tile images are delivered to the Merging Server via an internal network, and the merged/clipped/compressed video stream is delivered to remote participants via an out-going network.

Since all pixels in the tiled-display should be first delivered to the Merging Server periodically, the scale of the tiled-display that can be supported in the proposed system is limited by the bandwidth of the internal network connecting the tiled-display and the Merging Server. The required bandwidth of the internal network, $B_{in}$, can be computed as follows:

$$B_{in} = R_T \times (3 \text{ bytes/pixel}) \times (8 \text{ bits/byte}) \times (f \text{ frames/sec}),$$

where $R_T$ is the resolution of the tiled-display of size $n \times m$ ($R_T = (n \times m) \times R_{tile}$, and $R_{tile}$ is the resolution of each tile ($R_{tile} = a \times b$). For example, if an Ethernet with 10 Gbps bandwidth and 70% utilization ratio is used as the internal network, a tiled-display system with 16 tiles can be supported in the proposed system when the resolution of each tile is $1,280 \times 1,024$ and the capturing rate is $15$ frames/sec ($1,280 \times 1,024 \times 3 \times 8 \times 15 \approx 16$). If the network bandwidth is scaled-up to 20 Gbps, a tiled-display system with 32 tiles can be supported in the proposed system because it is linearly pro-
Fig. 10  An example of zoom-in and zoom-out functions of virtual camera.

Let us assume that the resolutions of the tile and remote clients’ displays are the same, i.e., \( R_T = R_C = a \times b \). It is a reasonable assumption because the remote participant usually uses a desktop PC. If the remote participant selects the region of tiled-display that is bigger than \( R_C \), it is first clipped from the merged tiled-image and down-sampled to \( R_C \). Then, it is compressed with a video encoder such as H.264 whose compression ratio is greater than 100. In this case, the remote client views the tiled-display in a zoom-out fashion as shown in Fig. 10 (a). From this assumption and analysis, we can estimate the bandwidth of the out-going network, \( B_{out} \), as follows:

\[
B_{out} = \frac{R_C \times (3 \text{ bytes/pixel}) \times (8 \text{ bits/byte}) \times (f \text{ frames/sec})}{\text{Compression_Ratio_of_Encoder}}
\]

If we assume that, \( R_C = 1,280 \times 1,024 \), \( \text{Compression_Ratio_of_Encoder} = 100 \), and \( f = 15 \text{ Hz} \), the required network bandwidth for each remote client is about 4.5 Mbps \((\frac{1,280 \times 1,024 \times 3 \times 8 \times 15}{100} \approx 4.5 \text{ M})\). For example, if an Ethernet with 1 Gbps bandwidth and 70% utilization ratio is used as the out-going network, the maximum number of remote participants who can join a session is about 159 \((\frac{1,280 \times 1,024 \times 0.7}{4.5 \times 1.024} \approx 159.2)\) in the proposed system. It means that the bandwidth of the out-going network, \( B_{out} \), is not a problem to increasing the number of remote participants joining a session. Actually, the number of remote participants who can join a collaboration session is limited by the CPU power of the Merging Server because the Server compresses the clipped images of size \( R_C \) in real-time. If the number of remote participants is \( p \), \( p \) video streams need to be concurrently compressed at the Merging Server. From the experiments on a Merging Server (2 CPUs: Intel® Xeon® E5462 @2.80 GHz \times 2, total 8 Cores), we found out that a maximum of 7 remote clients can be supported in the pro-
posed system when encoding video streams \(R_C = 1,280 \times 1,024, f = 15\) Hz with an H.264 S/W video encoder \([7]\). It means that number of remote clients joining a collaboration session is limited by the CPU power of the Merging Sever, and it is limited to 7 in our experimental system. This problem can be relieved easily by introducing multiple Merging Servers in the proposed system. In this extension, each PC sends captured tile images to multiple Merging Servers concurrently using the broadcasting capability of Ethernet, whereupon each Merging Server independently assembles the tile images and delivers the clipped tiled-display images to remote participants after compression. From this analysis, we conclude that the number of remote participants joining a collaboration session is limited by the CPU power of Merging Server, and it can be linearly scaled up by introducing multiple Merging Servers.

4. Experiments and Analyses

We have implemented the proposed remote collaboration tool for tiled-display system on a 7 \(\times\) 4 and 3 \(\times\) 2 tiled-display systems. The internal network connecting the tiled-display system and the Merging server is 1 Gbps Ethernet. Figure 12(a) shows a snapshot of a 7 \(\times\) 4 tiled-display system concurrently displaying two video streams using the real-time video player proposed in \([4]\), and Fig. 12(b) shows the screen shot of the client that specifies a part of this tiled-display image as an ROI\(^{1}\). Let us first present some performance metrics and the experimental results on a 3 \(\times\) 2 tiled-display system.

4.1 Evaluation Criteria

The quality of a sequence of merged tiled-display images can be characterized by the delay that denotes how much time is required to get the merged tile-image at the \(j\)-th period, the skew that denotes the variance of the time stamps of the selected tile images for \(I_M\), the jitter that denotes the variance of the differences of the time stamps between successive merged tiled-display images, and finally the visual consistency that denotes the average edge pixel differences of tile images in \(I_M\). These parameters are formally defined as follows, where \(I_M\) consists of the set of tile images \((I_1^M, I_2^M, \ldots, I_n^M)\), \(T S_{\text{avg}}\) is the average time stamp of the tile images constituting \(I_M\), \(I_M(k)\) is the value of \(k\)-th edge pixel of tile image \(I_j^k\) \((1 \leq k \leq E(I_j^k))\) where \(E(I_j^k)\) is the number of edge pixels in \(I_j^k\), and \(Adj(I_j^k)\) is the value of the pixel that is adjacent to \(I_j^k\) (in tiled-display coordinate system but belonging to other tile image);

- the delay in \(I_M\) (Delay\(^{\text{i}}\)): \(\frac{\sum_{i=1}^{n} TS_j^i}{n} - TS_{\text{avg}}\), where

\[
TS_{\text{avg}} = \frac{\sum_{i=1}^{n} TS_j^i}{n}
\]

- the skew of \(I_M\) (Skew\(^{\text{i}}\)): \(\frac{\sum_{i=1}^{n} |TS_j^i - TS_{\text{avg}}|}{n}\)

- the jitter in \([I_j^{i-1}, I_j^M, I_j^{i+1}]\) (Jitter\(^{\text{j}}\)): \(\left(\frac{TS_j^i - TS_j^{i-1}}{TS_{\text{avg}} - TS_j^i}\right)\)

- the visual consistency of \(I_M\) (VC\(^{\text{c}}\)): \(\frac{\sum_{i=1}^{n} \sum_{k=1}^{E(I_j^k)} |I_j^k(k) - Adj(I_j^k)|}{E(I_j^k)}\)

4.2 Experimental Results

We have experimented with the three tile selection algorithms presented in Sect.3.2 on a 3 \(\times\) 2 tiled-display system, and measured their performances in terms of the metrics presented in the previous section. In order to show the visual consistency of the merged tiled-display image, we use the real-time video player \([4]\) on the tiled-display, as a collaboration tool, whose outputs are displayed on the whole tiled-display screen.

Two parameters presented in this paper, \(\alpha\) for the initial buffering latency and \(\varepsilon\) for the threshold for checking the visual inconsistency between neighboring tiles, are highly dependent on the network conditions and the pixel distribution of rendered images produced by visual applications, respectively. For example, if the network load is relatively low or the network has a relatively high bandwidth so that the captured tile-images arrive at the Merging Server regularly, a relatively small initial buffering latency is required. If these conditions are not satisfied, the buffering latency should be large enough to alleviate the jitter in the arrival times of captured tile images. It is a well-known problem in the video streaming service, which is usually resolved using the initial buffering technique. It means that we cannot determine the initial buffering latency computationally, but experimentally. In our experiments, we found that the initial buffering latency of 100 msec was sufficient to alleviate the jitters in the arrival times of captured tile images. Similarly, the threshold for checking visual inconsistency is highly dependent on the pixel distribution characteristics of rendered images and the activities of virtual objects produced by visual applications. For example, if there are many edges in rendered images or the appearance of graphical object is radically changed in rendered images, a relatively high threshold value should be used to determine the visual consistency between tile images. Otherwise, a relatively low threshold should be used to determine the visual inconsistency. It means that this threshold value is highly dependent on the rendered images produced by visual applications and thus difficult to determine statically. It is similar to determining a threshold in the shot change detection \([13]\) for automatically finding the shot boundaries in a video stream that consists of a sequence of diverse shots. There have been some studies \([8], [13]\) conducted to detect the shot boundaries by dynamically adjusting the threshold based on inves-
Fig. 12: Screen shots of video player on 7 × 4 tiled-display and remote participant’s display.

Fig. 13: Experimental time stamps of tile images selected at the same period.

The time stamps of tile images selected at j-th period

The pixel differences in previous frames are determined by checking the visual inconsistency between tile images. In our experiments, the threshold is dynamically changed by aggregating the average of the edge pixel differences in previous temporal/spatial tile images. If the edge pixel difference is 3 times the average value, the algorithm reports that a tile image inconsistency occurs.

Figure 13 shows a snapshot of actual time stamps of the tile images that have merged at each period by the three tile image selection algorithms, and Fig. 14 shows their respective performances related to the temporal metrics such as the delay, skew, and jitter when the frequencies of the capturing and merging tasks are 15 frames/second, i.e., \( f = 15 \text{ Hz} \). As shown through this experiment, the delay of the simple sequential algorithm is worse than in other algorithms because it never considers the time stamps of tile images when selecting them from the buffers. Also note that the selection algorithm based on actual capturing time stamps shows a superior performance with respect to the temporal performance metrics because it selects the tile images that are closer to the target time stamp at each period. However, since the delays/jitters/skews of all of the three selection algorithms are still less than the period of the video stream consisting of merged tiled-display images.
Fig. 14  Delay, jitter, and skew of three selection algorithms.

(a) By Simple Sequential Algorithm

(b) By Capturing Time-based Algorithm

(c) By Capturing Time and Visual Consistency-based Algorithm

Fig. 15  An example of sequence of merged tiled-display images.
Since a tiled-display system can concurrently display multiple types of visual objects such as images, videos, and virtual graphical objects that are generated by independent remote and/or local applications, it has been used as an effective tool for network-based multi-party collaboration. In order to deliver CSCW activities on a tiled-display to remote participants, this paper proposes an application-independent approach that periodically captures the tile images, merges them in a synchronous manner, and finally delivers them as a compressed video stream after clipping an ROI. It is an extension of the widely used remote collaboration tools on PCs such as CamStudio, which periodically captures the video frame buffers of a PC and delivers them to remote participants as a compressed form. Since the capturing times of tile images can be different from each other because each PC controlling the tile display performs different tasks while capturing their frame buffers, a set of tile images should be carefully selected in order to generate a video stream that has good temporal as well as visual qualities. This paper proposes three tile selection algorithms such as a simple sequential selection, a capture time-based selection, and finally a capture time and visual consistency-based selection. Experimental results on a $3 \times 2$ tiled-display system show that the selection algorithm based on the capturing time and visual consistency can produce a video stream that is temporally natural and visually consistent. We also propose a mechanism that provides an independent virtual camera to each remote participant so that the participant can see the part of a tiled-display accurately while changing the ROI and zoom-level as desired. Since the proposed delivering mechanism is independent of the rendering and CSCW applications used on a tiled-display system, it can be applied to all tiled-display systems to extend them as an effective network-based multi-party collaboration tool.

5. Concluding Remarks

References

[1] ACA Systems, ACA Capture Pro, http://www.acasystems.com, 2008.
[2] J. Allard, V. Gouranton, G. Lamarque, E. Melin, and B. Raffin, “Softgenlock: Active stereo and GenLock for PC cluster,” Proc. Workshop on Virtual Environments 2003, pp.255–260, 2003.
[3] Camstudio: Free Streaming Video Software, http://camstudio.org, 2008.
[4] G. Choe, J. Yu, J. Choi, and J. Nang, “Design and implementation of a real-time video player on tiled-display system,” Proc. IEEE 7th International Conference on Computer and Information Technology, pp.621–626, 2007.
[5] F. Cristian, “Probabilistic clock synchronization,” Distributed Computing, vol.3, no.3, pp.146–158, 1989.
[6] E. Bbara, N. Kukimoto, J. Leigh, and K. Koyamada, “Tele-immersive collaboration using high-resolution video in tiled-displays environment,” Proc. 21st International Conference on Advanced Information Networking and Application Workshops, vol.2, pp.953–958, 2007.

[1] http://ffmpeg.org/ffmpeg-doc.html
[2] U. Garg, R. Kasturi, and S.H. Strayer, “Performance characterization of video-shot-change detection methods,” IEEE Trans. Circuits Syst. Video Technol., vol.10, no.1, pp.1–13, 2000.
[3] S. Han, J. Kim, K. Choi, and J. Kim, “Integrating multiple HD video services over tiled-display for advanced multi-party collaboration,” Proc. SPIE, vol.6391, pp.63910W.1–10, 2006.
[4] L. He and Z. Zhang, “Real-time whiteboard capture and processing using a video camera for remote collaboration,” IEEE Trans. Multimed., vol.9, no.1, pp.198–206, 2007.
[5] G. Humphreys, M. Eldridge, I. Buck, G. Stoll, M. Everett, and P. Hanrahan, “WireGL: A scalable graphics system for clusters,” Proc. SIGGRAPH 2001, pp.129–140, 2001.
[6] B. Jeong, Visualcasting: Scalable Real-time Image Distribution in Ultra-High Resolution Display Environments, PhD Dissertation, University of Illinois at Chicago, 2009.
[7] W. Kim and J. Kim, “An adaptive shot change detection algorithm and its implementation on portable multimedia player,” IEEE Trans. Consum. Electron., vol.55, no.2, pp.628–635, 2009.
[8] T. Ni, G. Schmidt, O. Staadt, M. Livingston, R. Ball, and R. May, “A survey of large high-resolution display technologies, techniques, and applications,” Proc. IEEE Virtual Reality Conference, pp.223–236, 2006.
[9] L. Rowe and V. Casalaina, “Capturing conference presentation,” IEEE Multimedia, vol.12, no.4, pp.76–84, 2006.
[10] R. Singh, B. Jeong, L. Renambot, A. Johnson, and J. Leigh, “TeraVision: A distributed, scalable, high resolution graphics streaming system,” Proc. 2004 IEEE International Conference on Cluster Computing, pp.391–400, 2004.
[11] M. Štefk, D.G. Bobrow, G. Foster, S. Lanning, and D. Tatar, “WYSIWIS: Early experiences with multiuser interface,” ACM Trans. Information System, vol.5, no.2, pp.147–167, 1987.
[12] C. Zhang, Y. Rui, J. Crawford, and L. He, “An automated end-to-end lecture capture and broadcasting system,” ACM Trans. Multimedia Computing, Communications, and Applications, vol.4, no.1, pp.6.1–6.23, 2008.
[13] B. Jeong, L. Renambot, R. Jagodic, R. Singh, J. Aguilera, A. Johnson, and J. Leigh, “High-performance dynamic graphics streaming for scalable adaptive graphics environment,” Proc. 2006 ACM/IEEE conference on Supercomputing, pp.24, 2006.
[14] Nirmishes, P. Harish, and P.J. Narayanan, “Garuda: A scalable tiled display wall using commodity PCs,” IEEE Trans. Vis. Comput. Graphics, vol.13, no.5, pp.864–877, 2007.
Giseok Choe  received his B.S. and M.S. degrees in computer science and engineering from Sogang University, Seoul, Korea, in 2006 and 2008 respectively. He is currently a PhD Student at Sogang University. His research interests include multimedia contents management and retrieval, and video processing.

Jongho Nang  received his Ph.D. and M.S. degrees in computer science from Korea Advanced Institute of Science and Technology (KAIST), Daejon, Korea, in 1992 and 1988, respectively, and his B.S. degree in computer science from Sogang University, Seoul, Korea, in 1986. He has been a professor of Computer Science and Engineering Department, Sogang University, Seoul, Korea, since 1993. His research interests include multimedia system, parallel processing, and internet technology.