Live Room Merger: A Real-Time Augmented Reality System for Merging Two Room Scenes

Chu-I Chao, Chien-Min Wang, Hsuan-Chi Kuo, Liang-Chi Tseng,
Shih-Kai Lin, Yu-Ju Tsai, Ching-Chi Lin, and Da-Fang Chang
Institute of Information Science, Academia Sinica, Taipei, Taiwan

ABSTRACT

A real-time augmented reality system is built to replace the background of the user's room (the 'observer room' or 'local room') with a 360-degree live video of another room (the 'remote room'). The user can see the merged room captured by an RGB-D camera mounted on the VR headset. A 360-degree image of the remote room is converted into a simple box-like room structure model in real time. The model is loaded into Unity and replaces the background of the observer room, with the result then displayed in a VR head-mount device.

CCS Concepts

• Computing methodologies---Computer graphics---Graphics systems and interfaces---Mixed / augmented reality
• Computing methodologies---Artificial intelligence---Computer vision---Computer vision problems---Reconstruction
• Computing methodologies---Artificial intelligence---Computer vision---Computer vision tasks---Scene understanding

Keywords

Augmented reality, real time, reconstruction, 3D modeling, panorama, indoor structure

Author Contact Information:
kmchao33@gmail.com, cmwang@iis.sinica.edu.tw,
hsuanchikuo@gmail.com, lctseng@cs.nctu.edu.tw,
r05922043@ntu.edu.tw, b02902052@ntu.edu.tw,
deathsimon@gmail.com, b02901126@ntu.edu.tw

VRIC '17, March 22–24, 2017, Laval, France

ACM ISBN 978-1-4503-4858-4/17/03…$15.00
http://dx.doi.org/10.1145/3110292.3110301

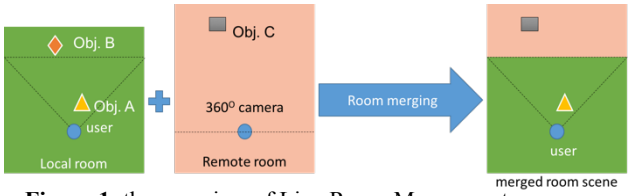

Figure 1: the overview of Live Room Merger system: Partial local room scene (given by the dotted lines) is replaced by the remote room scene.

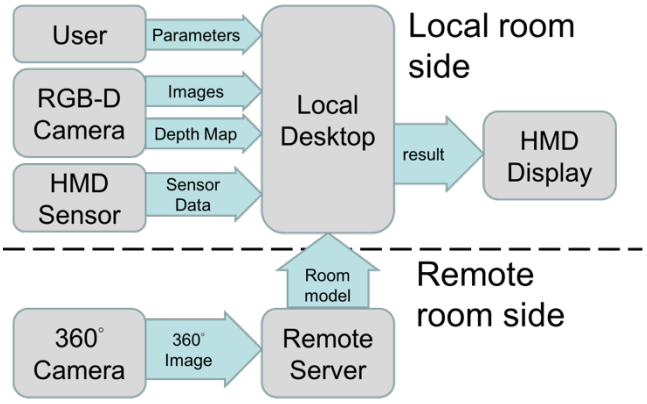

Figure 2: the data flow of Live Room Merger system.

1. INTRODUCTION

Augmented and virtual reality technologies (AR and VR) can be used to merge real and virtual worlds. Commercial VR devices, including Oculus Rift, HTC Vive and PS VR, synthesize the user's head or even body motion in VR applications. Intel's Project Alloy immerses users into virtual worlds. This paper seeks to merge an entire 360-degree indoor virtual scene with another, real world, indoor scene to create a more immersive distance learning or web conferencing experience by merging remote classrooms, or conference rooms.

A real time system is built to merge two indoor room scenes. As shown in Fig. 1, we reconstruct the remote room scene using a 360-degree camera and substitute it for the background of the local room scene. The user can then see the merged room through an RGB-D camera mounted on an HMD.

Panoramas have been widely used in multimedia applications to outline scenes. It has been a standard technique to capture and display different kinds of omni-directional information ([5], [6]),

leading to a large panoramic image utilization in a variety of circumstances. For example, lots of research has been done on understanding indoor scenes and reconstructing indoor floor plans from panoramas. Cabral and Furukawa [1] took multiple indoor panoramas as inputs and used structural cues from individual images for floor plan reconstruction. The PanoContext method [2] recovers the full room layout from a single panorama image, assuming a box-shaped room. Walls and floors are used as contextual information to recognize object categories and positions. Recent work by Yang and Zhang [3] recovered 3D room shapes from a single panorama using lines and superpixels in a constraint graph. Farin et al. [4] used a semi-automatic reconstruction process in which the user marks the room corners in the panoramic images to create a coarse reconstruction of the indoor environment. These studies all focus on complex floor plan and indoor environment reconstruction problems. The proposed system focuses on a real-time reconstruction method for simple room structures.

Our semi-automatic remote room structure is modeled on a spherical panorama, and aims to reconstruct the structure of a box- or cuboid-shaped room in real time. The user is required to specify the room's 8 corners. To enhance intuitiveness, cube mapping is used rather than directly converting a 360-degree spherical image into a cuboid-like room structure model. One constraint of the modeling method is that the input panorama should be taken with the camera directly facing any one of room walls.

Our system architecture consists of two main parts, as shown in Fig. 2. One is the semi-automatic remote room structure reconstruction from a spherical panorama in real time on the remote room side, and the other is the background replacement and display on the local room side, using the room structure model from the remote side, an RGB-D camera and an HMD.

To capture a live 360-degree video of the remote room, we use a Ricoh Theta S, a commercial 360-degree dual-fisheye camera which automatically stitches two fisheye images into a 360-degree spherical image in real time. The output 360-degree spherical live streaming video has a resolution of 1280x640 pixels at a frame rate of 15 fps. To generate the user's viewpoint on the local room side, we use a commercial RGB-D camera called a ZED Stereo Camera, mounted on an Oculus Rift DK2 headset. The ZED Stereo Camera's tracking and depth detection function also provides more degrees of freedom of head motion and retrieves the surrounding foreground. The desktop computer for the remote room structure reconstruction uses a 3.7Ghz Intel Xeon E5 CPU with a GTX 1080 graphics card and 64GB of memory, while the computer on the local room side uses a 3.5Ghz Intel Core i7 with a GTX 970 graphics card and 16GB of memory. Communication between these two computers is established via a 1Gbits wired network.

The software implementation on the local room side is based on Microsoft Windows 7 and Unity3D. The built-in 3D world and the Oculus Rift Plugin in Unity 3D facilitate system construction and scene display in the headset. The remainder of this paper is structured as follows. Section 2 introduces the proposed real-time box-like room structure modeling from a spherical panorama. Section 3 provides a detailed description of the local room side. Section 4 shows the experimental results and Section 5 concludes the paper and provides directions for future work.

[1]https://github.com/rlk/envtools

2. SEMI-AUTOMATIC BOX-LIKE ROOM STRUCTURE MODELING FROM A SPHERICAL PANORAMA

The box-like room structure modeling method from a spherical panorama references the position of the room's corners as given by the user via cube mapping. Depending on its simplicity, it can be implemented in real-time with GPU acceleration.

The process consists of two main stages: mapping a 360-degree image onto a cube, and then mapping the cube onto a cuboid with 8 vertices. The first stage is done with GPU and the second uses only CPU.

2.1 Mapping from A Spherical Panorama to A Cube Map

Instead of directly converting a 360-degree spherical image into a cuboid map, this stage helps us solve the problem more intuitively by converting the curvy edges of a box-like room to either horizontal or vertical lines. This is easily accomplished by rewriting the C++ code in an open source project on Github[1] using CUDA.

2.2 Mapping from A Cube Map to A Cuboid Map Given 8 Vertices of Cuboid

We divide the cube into eight equal mini-cubes, and then map each mini-cube to a mini-cuboid. We then merge the mini-cuboids to obtain a simple room model.

The first and the third steps are trivial, thus we focus on the mapping from a mini-cube to a mini-cuboid. This involves two

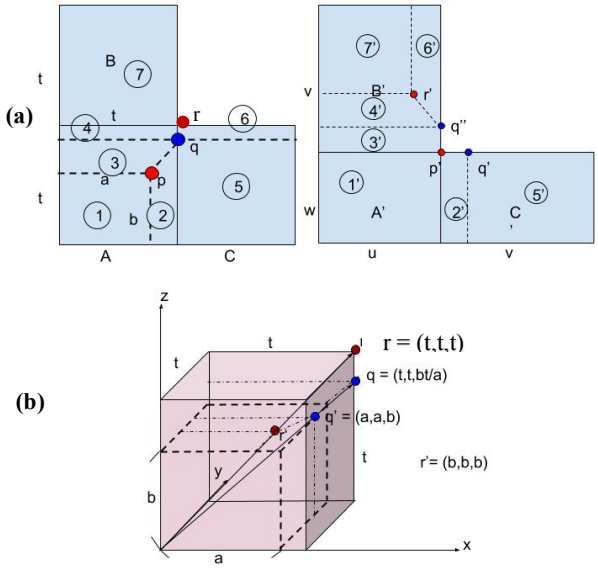

Figure 3: **(a)** 2D mapping between a mini-cube and a mini-cuboid. **(b)** 3D mapping between a mini-cube and a mini-cuboid.

steps: finding the ratio of the mini-cuboid's length, width, height and filling pixels into the mini-cuboid.

Problem Definition As shown in Fig. 3a, given three images *A*, *B*, *C* of a mini-cube, the length of the mini-cube *t* and a cuboid's vertex *p*, find the ratio of the length, width, height of the corresponding mini-cuboid and fill the pixels into mini-cuboid images *A'*, *B'*, *C'*.

Observation As described in Fig. 3a, let *u*, *v*, *w* respectively be the length, width and height of the mini-cuboid and *p = (a, b)* in plane *A*. In addition, we can suppose that *v>=u>=w* anytime by rotating the mini-cuboid to ensure that *p* is in the plane *A* and *a >= b*.

Observe that part *1* corresponds to image *A'*. Thus, we can assign *u=a, w=b*. In addition, *p* and *p'* are actually the same point in the xyz coordinate, so we get *v = t* and *u:v:w = a:t:b*. By obtaining the coordinate of q and q' in the xyz coordinate, we can correspond each part of the mini-cube with that of the mini-cuboid and further fill the pixels.

Proof As shown in Fig. 3b, let *O=(0, 0, 0), q=(t, t, k)* and *q'=(u, l, w)=(a, l, b)*. Notice that *Oq'q* forms a line in xyz coordinate, so we have *t/a = t/u = t/l = k/w = k/b => l = a, k = bt/a*. Furthermore, because *q'* and *q''* are actually the same point in the xyz coordinate, the length of *p'q''* is equal to the length of *p'q'*. On the other hand, *Or'r* also forms a line in the xyz coordinate. So we have *r'=(w, w, w)=(b, b, b)*. Now we obtain the correspondence between each part of the mini-cube and the mini-cuboid. This leaves the transformation. In mapping a 360 spherical image to a cube, we actually map the whole box-shaped room. Thus we perform the inverse operation. In mapping the whole box-shaped room to the cube, we perform a projective transformation from part *n'* in the mini-cuboid to part *n* in the mini-cube for *n=1~7*. The inverse operation of a projective transformation is also a projective transformation. Therefore, to fill pixels into the mini-cuboid, we perform a projective transformation from part *n* in the mini-cube to part *n'* in the mini-cuboid for *n=1~7* given the coordinate of *p, p', q, q', q'', r, r'*.

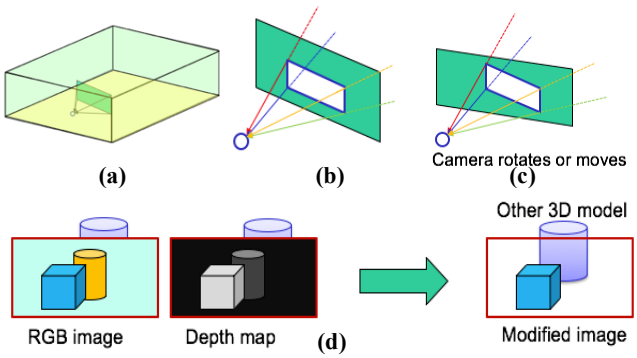

Figure 4: (a) shows the position setting of the remote room model and the view plane in Unity. **(b)** generates the area to be replaced in the view plane using 4 rays. **(c)** represents the condition when the user rotates or moves. **(d)** background replacement: make the background of the area to be replaced transparent and let the 3D model behind appears.

3. LOCAL ROOM SIDE DETAIL

In this part, we describe the use of a VR headset and an RGB-D camera to construct a merged room scene for display with GTX 970 graphics card acceleration.

3.1 Merged Room Scene Construction and Display

We use a C/C++ extension as an interface between Unity and other three components, including the ZED Stereo Camera, GPU and cuboid images sent from the remote side. C/C++ supports much more flexible memory operations than C# in Unity. ZED SDK is written in C++ so it cannot be used by Unity directly. Also, Unity (free license) doesn't support CUDA directly.

In the following, we describe each step for merged room construction and display.

Step 1: Reconstruct local and remote scene in Unity

For local and remote scene reconstruction, both require creating blank planes and then filling them with the scene image as the plane texture in Unity. The remote room model is reconstructed using six plane objects and cuboid images as the texture, and the local scene is represented by a camera object, as user's viewpoint, and a view plane which fills in RGB video frames taken by the ZED Stereo Camera.

The size and position of the box, as well as the initial position of the camera object and the view plane, are set by default in the world coordinate system.

Also note that the camera object and the view plane must both be in the box for background replacement, as shown in Fig. 4a.

Step 2: Set the area on the view plane and remove the background

The area to be replaced is a quadrangle defined by four rays from the view point through four points given by the user in the view plane, as described in Fig. 4b.

As shown in Fig. 4c, the area to be replaced will remain fixed in the world coordinate after the area is set, regardless of rotation or movement.

We now remove the background in the area. Here the background means objects at a user-determined distance from the ZED Stereo Camera. The removal is performed by using the GPU to making the area background transparent, thus rendering the remote scene behind visible through the transparent area. Figure 4d shows an example of the background replacement.

Step 3: Speed up the merged room scene display

Now the GPU is used to retrieve foreground images and to handle the display. However, Unity doesn't integrate CUDA, so a naïve, platform-independent approach for display is to copy the data from the GPU to the CPU in Unity, and then to write the data to the GPU again for head-mounted display. To avoid duplicating memory usage, we choose a platform-dependent approach. On Windows, Unity uses Direct3D 11 for rendering. CUDA provides advanced interoperability with Direct3D 11. First, we create the texture in Unity. We then retrieve the GPU texture pointer from Unity. Finally, we hardcopy the result of step 2 to the memory space to which the texture pointer points. This approach only

requires one GPU memory copy, which is much faster than the platform-independent approach.

Step 4: Track the head position and then update the depth threshold

For head tracking, both the trackers on Oculus Rift DK2 and ZED Stereo Camera are used because the Oculus Rift DK2 provides good head orientation tracking but the ZED Stereo Camera provides better position tracking. After obtaining the head's position, we update the depth threshold and apply it to the background removal process.

4. EXPERIMENT RESULTS

Figure 5: an example of the proposed room structure modeling from a spherical panorama via cube mapping.

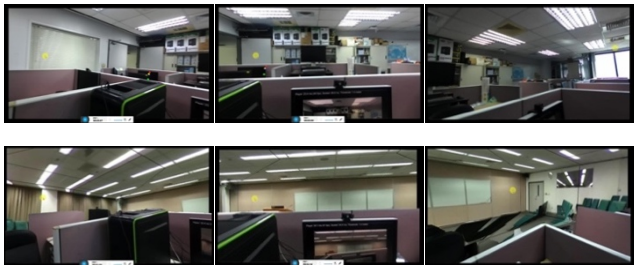

Figure 6: comparison with the original scenes above from similar viewpoints and the merged results below.

To evaluate the efficiency and the graphic result of our proposed room structure modeling method and Live Room Merger system, we merged our lab with a conference room. The reconstruction result of the conference room from a panorama via cube mapping is shown in Fig. 5. The merged results are shown in Fig. 6, compared with the original scenes from similar viewpoints.

We assess the efficiency on the remote room side in a workflow consisting of five stages. We first load frames of a live 360-degree spherical video from a computer connected to a Ricoh Theta S on the remote server, and then transform them into a cuboid-like room structure model via cube mapping. The result is compressed and sent to the local room desktop.

Compression is necessary to prevent the throughput from exceeding the bandwidth on an 1Gbits wired network. For fast compression and decompression, we use the LZ4 lossless compression algorithm from an open-source library provided on Github[2]. TCP is used for data transfer in both the first and last stages.

We accelerate the whole progress and achieve a frame rate up to 15 fps, the highest frame rates a Ricoh Theta S supports. The naïve approach runs the five stages sequentially and sends out the results at a rate of 6 fps. After the first try, we found that each stage has a similar execution time. Therefore, on the second approach we tried pipelining the five stages. Each stage hardcopies the results from the previous stage as input when it receives the signal. This approach obtains a frame rate of 9 fps.

The fastest approach is described here. The five stages are pipelined by maintaining a buffer pool of a size 5 and operating four blocking queues as the interfaces between adjacent stages. The result of each stage is stored in a buffer and the buffer pointer passes among stages using the blocking queue.

After data is received from the network and stored in a buffer, the buffer pointer is pushed into the queue at the first stage. The next three stages pop the pointer from the blocking queue of the previous stage, operate it, store the result in the same buffer and then push the pointer into the blocking queue of the current stage. At the last stage, we pop a pointer, send the data it points to through the network and push the same pointer into the queue of the first stage for memory reuse. Once a queue is full, no item can be pushed into the queue until next pop happens.

This approach needs no memory copies and allows each stage to simultaneously access and process one buffer due to the size of buffer pool.

The merged scene is rendered at an average rate of 50 fps on the local desktop.

5. CONCLUSION

A real-time augmented reality system replaces the background of a local room with a 360-degree live video of a remote room. A real-time semi-automatic box-like remote room structure modeling method is implemented based on a spherical panorama using pipelining and blocking queues. In addition, on the local room side, we describe the background replacement method and performance acceleration through the integration of compressed cuboid images, a ZED Stereo Camera, Oculus Rift, GPU and Unity. This system can help to provide a more immersive distance learning or web conferencing experience. Future work will seek to improve room modeling quality and speed up the whole progress to further improve the immersive experience.

REFERENCES

[1] R. Cabral and Y. Furukawa. Piecewise planar and compact floorplan reconstruction from images. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on. IEEE, 2014.

[2] Y. Zhang, S. Song, P. Tan, and J. Xiao. Panocontext: A whole-room 3d context model for panoramic scene understanding. In Computer Vision–ECCV 2014, pages 668–686. Springer, 2014.

[3] H. Yang and H. Zhang. Efficient 3D room shape recovery from a single panorama. In CVPR, 2016.

[4] D. Farin, W. Effelsberg, et al. Floor-plan reconstruction from panoramic images. In Proceedings of the 15th international conference on Multimedia, ACM, 823–826, 2007.

[5] M. Brown and D. G. Lowe. Recognising panoramas. In ICCV, vol. 3, 1218, 2003.

[6] M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. International Journal of Computer Vision 74, 1, 59–73, 2007.

[2]https://github.com/lz4/lz4