OMNIDIRECTIONAL FREE VIEWPOINT VIDEO USING PANORAMIC LIGHT FIELDS

Steven Maesen, Patrik Goorts, Philippe Bekaert

Hasselt University - tUL - iMinds
Expertise Centre for Digital Media
Wetenschapspark 2
3590 Diepenbeek, Belgium

ABSTRACT

In this paper, we describe a system to create an omnidirectional free viewpoint experience using only a small number of input cameras. The input cameras are placed on a circle and we create a large number of novel virtual viewpoints on that circle. Next, we choose a position within that circle and compute the omnidirectional image that is visible from that position by considering the collection of virtual images as a light field. The corresponding pixels in the virtual images are selected by tracing rays from the desired viewing position. Changing your position inside the circle, results in an adapted view-dependent rendering. This creates a free viewpoint 3D VR experience. We demonstrate our method using the game engine Unity combined with the Oculus Rift.

Index Terms **—** Omnidirectional, Free Viewpoint Interpolation, Unity, Oculus, VR, Light Field, View-dependent Rendering

1. INTRODUCTION

In this paper, we will combine the active topics of omnidirectional rendering and view-dependent rendering. We created a system where it is possible to move and look around in a scene where the image you see adapts to your viewpoint. To accomplish this, a small number of input cameras, placed on a circle, is used. We chose for a small number to keep the system practical. Next, we extend the amount of cameras to render the images of virtual cameras placed on the same circle. This way, we effectively create a circular light field.

A light field is a function that describes the flow of light in a point, from a direction. This means that a light field is a 5D function per color. When the complete function is defined, it is possible to create a viewpoint from the scene from any point looking in any direction. However, the complete light field is seldom known. For example, when the geometry and texturing of a scene is known, a complete light field can be defined. Nevertheless, creating a correct and dense geometry of a real scene is still an open problem.

Another approach is sampling the light field using cameras and then defining a dense light field from this information. A camera is essentially a light field sampler, because the incident light in the camera center in a number of directions is known. However, the postion is limited to the camera center and no other positions in the light field are known. Therefore, we extend the light field to multiple locations.

In this paper, we will place a small number of cameras in a circle. This defines a sparse light field sample. Then, we render the images of a large number of virtual cameras on that circle C. All these virtual cameras define a dense, circular light field. We can then use this light field to create a new viewpoint inside the circle of virtual cameras. We will focus on cylindrical panoramic images. By tracing a view ray from the desired camera center M_v, we will intersect the center of a virtual camera and, as a consequence, it's image plane. The color recorded on the image plane (that is part of the light field) is the same as the color of the ray arriving in the desired point M_v (assuming the space in the circle is empty). By tracing rays in all directions, we can create a virtual cylindrical panorama image.

To demonstrate the effectiveness of our approach, we built a prototype using the game engine Unity [1] combined with the Oculus Rift [2], an immersive head mounted display. To provide high resolution at high framerates, we demonstrate both prerendering a dense set of these cylindrical panoramic images by varying M_V, and rendering the needed panoramic viewpoint in realtime using fast RAM memory.

Typically, this problem is addressed by using omnidirectional cameras. Kralla et al. [3] use multiple positions of an omnidirectional camera to generate disparity maps, but no free viewpoint application is proposed. Birklbauer et al. [4] use focal stacks for reconstructing a light field. Kawauchi et al. [5] use an array of omnidirectional cameras on a robot to generate a dense light field. By sampling this light field, free viewpoint navigation is possible.

Creating circular (i.e. omnidirectional or 360°) light fields can alternatively be done with specialized hardware. Examples are the Neo system of Jaunt [6], the Immerge system of Lytro [7], or the system of Ergunay et al. [8]. Chiba et al. [9] describe a system using a moving DSLR. They use a dense capturing method and do not generate intermediate views; only the real captured images are used for resampling. As a consequence, no moving scenes can be captured. Debevec et al. [10] use a similar method.

Our system does not require an omnidirectional system or a cumbersome and sometimes expensive setup. We only require multiple conventional rectilinear or equidistant (fisheye) cameras; the creation of the light field is moved to processing. Furthermore, we can increase the free viewpoint area by moving the cameras; the area is not limited by the hardware design.

2. VIEW INTERPOLATION

The first step is creating the dense light field. To generate all the virtual viewpoints on C, we use our previously described 2-staged plane sweep method [11]. First, we create a depth map for every real camera using a consensus-based plane sweep. We divide the space before the camera image in planes, parallel to the image plane. Next, we project all available input images to each plane. The projected images will overlap. Each pixel on the plane corre-

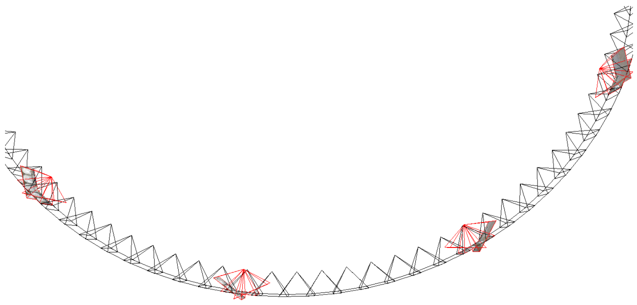
Figure 1. Part of the generated circular light field. The black camera outlays are generated using free viewpoint rendering. The input images are represented using red camera outlays.

sponds to a pixel on the image plane on the camera for which we are calculating the depth map. We can calculate an error metric per pixel by checking the difference between the projected, overlapping colors. By repeating this for every considered depth layer, we can obtain a minimum cost per pixel on the image plane. We save the depth corresponding with the lowest cost, thus obtaining a depth map. Next, we clean up the depth map with a bilateral (edge-aware smoothing) filter, where the edges in the color image of the considered camera are used.

Using these depth maps, we can create novel viewpoints. We repeat the previous plane sweep approach, but this time for a virtual, novel camera position. This time, we consider also the previously created depth values. When projecting the color values, we also project the depth values and compare them to the distance of that pixel to the corresponding camera. If the distance is too large, the cost is set to infinity. Using these depth maps allows us to handle possible occlusions and improve our cost function. This will increase the quality compared to other methods [11].

We will render a large number of virtual camera positions on the circle C, as can be seen in Figure 1. The more cameras are available, the better the final result (up to a certain point). This will create a dense circular or panoramic light field. In our approach, we will use a uniform distribution of virtual cameras, but this is not a requirement.

3. RENDERING OMNIDIRECTIONAL IMAGES

In the previous step, we created a dense circular light field, which we can resample to create novel omnidirectional cylidrical images from a viewpoint inside the circle C. We define the desired position of the virtual panoramic image as M_v with coordinates (x_v, y_v). The panoramic image visible from this point can be defined by a cylinder V with center M_v. We opted for a cylinder and not a sphere to allow fast view-dependent rendering.

The concept of the rendering is shown in Figure 2. We define a ray R_V starting from the center of the desired cylinder. Every ray is defined by an angle a_v. We can now calculate the intersection (x_i, y_i) between the circle of the rendered input viewpoints C and the ray R_V for the angle a_v. This intersection determines which virtual input camera to use. We normalize the circle to have a radius of 1 to reduce the complexity of the calculations, without loss of generality.

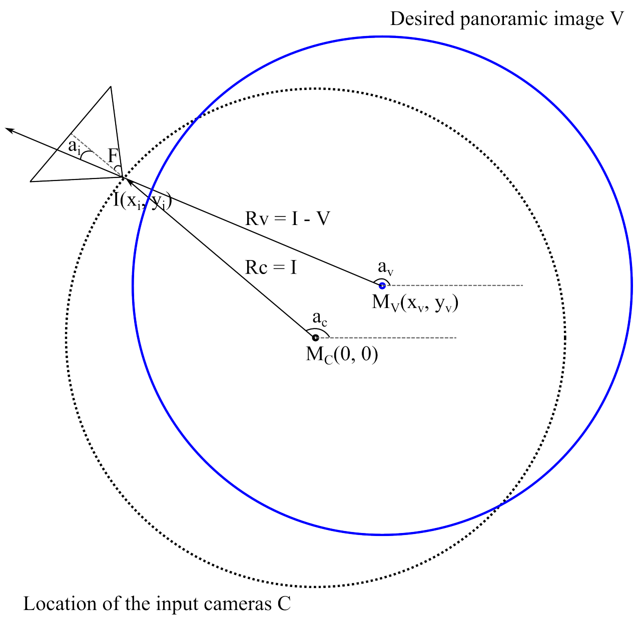

Figure 2. Overview of the rendering of the panoramic image. The virtual viewpoints are created on circle C with center M_C. In this image, we will render cylinder V with center M_V. We trace rays R_v from M_V to C, calculate the intersection to determine the camera I on C and calculate the angle a_i to determine which column of pixels to select from the image of I. By varying a_V, we can render V completely, based on the cameras rendered on C.

$$x'_v = x_v + cos(a_v) \tag{1}$$
$$y'_v = y_v + sin(a_v) \tag{2}$$
$$d_r = \sqrt{(x_v - x'_v)^2 + (y_v - y'_v)^2} \tag{3}$$
$$D = x_v y'_v - x'_v y_v \tag{4}$$
$$\Delta = \sqrt{r^2 d_r^2 - D^2} \tag{5}$$
$$x_1 = \frac{(y'_v - y_v)D + (x'_v - x_v)\Delta}{d_r^2} \tag{6}$$
$$x_2 = \frac{(y'_v - y_v)D - (x'_v - x_v)\Delta}{d_r^2} \tag{7}$$
$$y_1 = \frac{-(x'_v - x_v)D + |y'_v - y_v|\,\Delta}{d_r^2} \tag{8}$$
$$y_2 = \frac{-(x'_v - x_v)D - |y'_v - y_v|\,\Delta}{d_r^2} \tag{9}$$

This will yield 4 possible points (x_i, y_i). The correct point is determined by using the sine and cosine of the angle. Using the coordinates (x_i, y_i), a_c can easily be extracted. The same goes for $a_i = a_v - a_c$.

Using a_c and a_i, we select the corresponding image from our set of prerendered virtual cameras on the circle C, and the pixel column.

$$index = \frac{a_c \times \#(I)}{2\pi} \tag{10}$$
$$column = \frac{(a_i + F)W}{2F} \tag{11}$$

where W is the width of the input image and $\#(I)$ the number of images on C. We select the column of pixels in the selected rendered input image and place it in the final panoramic image corresponding with M_V in the column corresponding with a_v. This way, a full panoramic image is created.

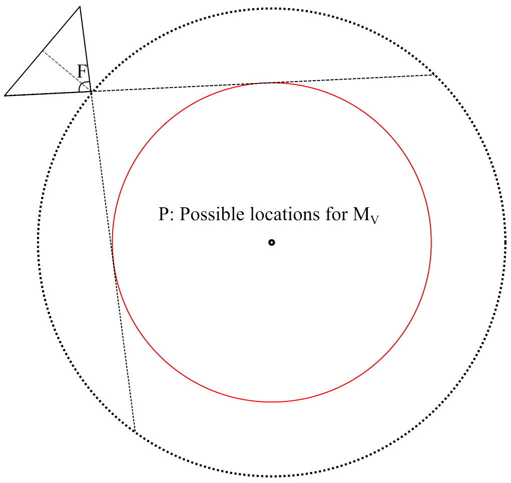

Figure 3. The position of the center of the cylinder is defined by the field-of-view F of the virtual cameras, shown by the red (inner) circle P. Within this circle, view-dependent rendering is possible.

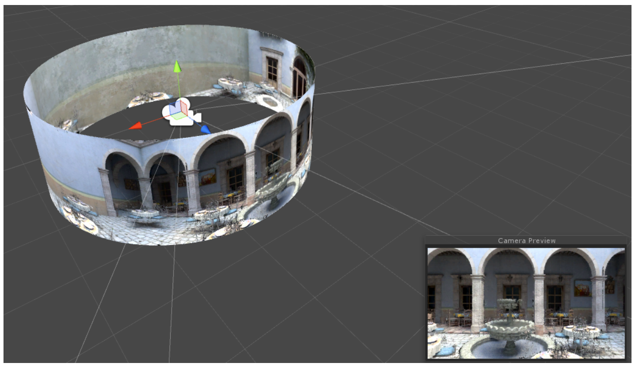

Figure 4. The scene in unity. The scene consists of a cylinder and a camera in the middle. The rotation of the camera is determined by the Oculus tracker and the image on the cylinder is adapted to the position of the Oculus.

The set of viewpoints we can effectively render is dependent on the field-of-view F of the virtual cameras, and are located in the circle P with radius $sin(F/2)$. If the angle a_i is larger than $F/2$, no color information is available and the rendered panoramic image V is incomplete. This is shown in Figure 3. The range of possible virtual panoramic viewpoints must be restricted. Alternatively, the field-of-view of the virtual cameras on C must be increased, if possible. This is defined by the input setup; the number and field-of-view of the real cameras define the maximum field-of-view of the virtual cameras.

4. FREE VIEWPOINT NAVIGATION

To create a free viewpoint VR experience, we display the images in a virtual reality device, more specifically the Oculus Rift [2]. We created a virtual cylindrical projection screen in the Unity Game Engine [1] and track the position of the head to determine the correct viewpoint to display. A virtual scene is required to alleviate the large field-of-view of the Oculus Rift, such that the space is immersive and the borders of the cylinder are not visible. Being able to look around and move your head, enhances the VR experience and creates a 3D effect.

As dense light fields have large storage needs while at the same time requiring fast random access, 2 different approaches were considered using commodity hardware. First, current day solid-state drives have large storage capacities and high data transfer speeds that suit our needs. However true random access is still not as fast as needed to render high resolution panoramic images in real-time. Instead a large number of panoramic images at different positions inside P are prerendered. Accessing the correct nearby viewpoint is fast enough to provide a good experience. However, storage is not sufficient for our application. On $500GiB$, only 16500 viewpoints can be stored, which is not sufficient for a smooth experience.

Alternatively, we used RAM to represent the circular light field. While more costly and limited in storage capacity, RAM allows for true random access needed to render a novel viewpoint in real-time. The memory throughput requirements for our dataset are $1080 \times 9216 \times 3 = 30MiB$ per frame, and a storage requirement of $7GiB$. Our tests show that a view-dependent image could be generated using our Unity plugin at a rate of 120 Hz, or $3.6GiB/s$, with low latency. This gives a very smooth and responsive VR experience while allowing the rendering of omni-directional stereo images for an enhanced 3D effect.

5. RESULTS

In our prototype we used 10 rendered input views from a scene captured with fisheye lenses with a field of view of 160°, such

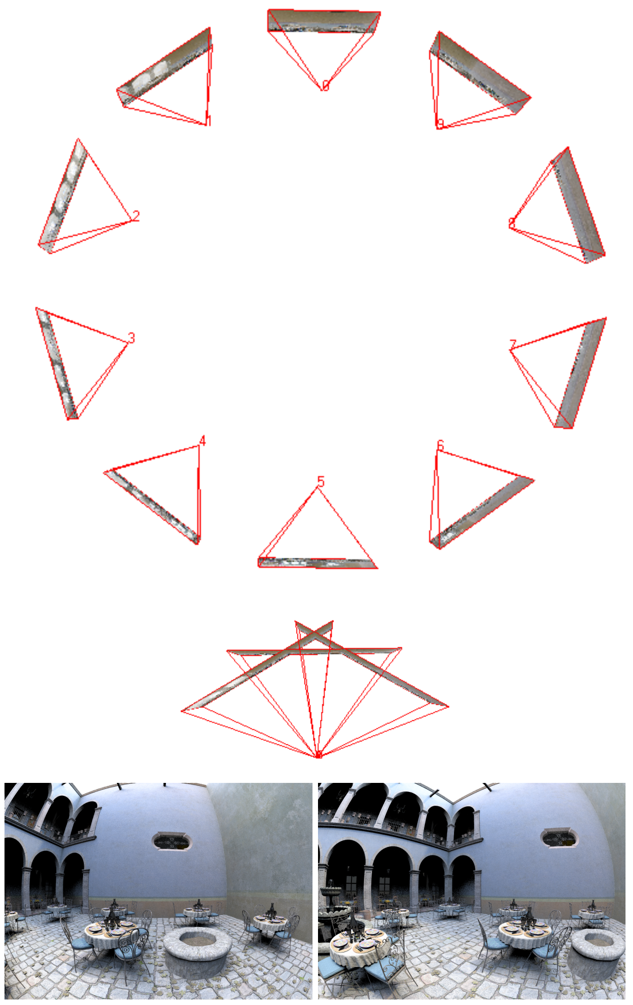

Figure 5. Camera setup and some of the input images. This setup is used to create the final result in our prototype. Top: Circular setup of cameras with fisheye lenses. Middle: conversion to 3 rectilinear images per camera. Bottom: Input fisheye images.

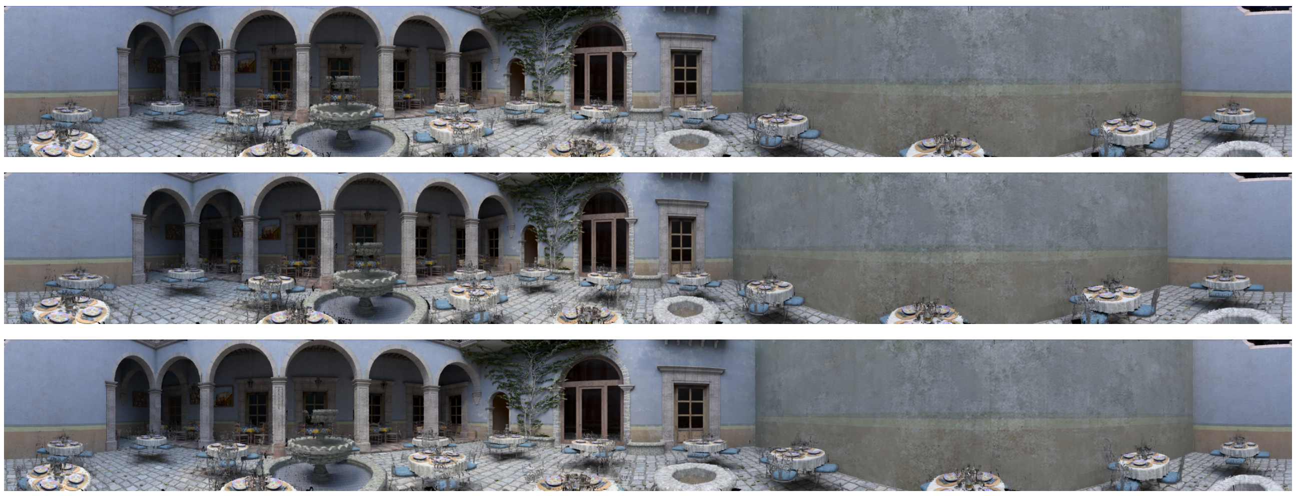
Figure 6. Results from 3 different viewpoints. The parallax effect is clearly visible.

that each point in space is covered by at least 3 camera images. The radius of the circle is 2 meters. This is shown in Figure 5. We opted for rendered input data to better evaluate the results. Each fisheye camera is transformed to 3 rectilinear images (Figure 5, middle) [12], each pointing in a different direction. This way, view interpolation of Section 2 can be applied.

These set of input images were used to generate 9216 virtual images with a horizontal field-of-view of 75°. This constructed circular light field is stored in an efficient way to allow the generation of view dependent omnidirectional images. These generated images have a resolution of 9216 covering the full 360 degrees, while keeping the height of both the input and output image constant at 1080. This allows us present crisp high resolution images to the user wearing a head mounted display at a required high frame rate. Due to the method for creating the panoramic images, rendering is fast, but only horizontal movement of the viewer is possible.

Some rendered viewpoints are shown in Figure 6. As can be seen, the results look good and clearly show the resulting parallax.

6. CONCLUSION

We presented a system for view-dependent rendering of omnidirectional images. Our input consists of a sparse images set from conventional cameras. By using view interpolation, we construct a dense circular light field, which we use to create the view-dependent omnidirectional images inside the circle. We demonstrated the speed and immersiveness of our system using the Unity Game Engine combined with the Oculus Rift.

Acknowledgement

This research was funded by the EU FP7 project DreamSpace. http://www.dreamspaceproject.eu/

7. REFERENCES

[1] Unity, "Unity game engine," 2016, https://unity3d.com/.

[2] Oculus, "Oculus rift vr engine," https://www.oculus.com/en-us/rift/, 2016.

[3] Bernd Krolla, Maximilian Diebold, Bastian Goldlucke, and Didier Stricker, "Spherical light fields," in *Proceedings of the British Machine Vision Conference*, 2014.

[4] Clemens Birklbauer and Oliver Bimber, "Panorama light-field imaging," in *ACM SIGGRAPH 2012 Posters*, New York, NY, USA, 2012, SIGGRAPH '12, pp. 61:1–61:1, ACM.

[5] Kensaku Kawauchi and Jun Rekimoto, "Quantized reality: Automatic fine-grained spherical images acquisition for space re-construction," in *Proceedings of the 13th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, New York, NY, USA, 2014, VRCAI '14, pp. 235–238, ACM.

[6] Jaunt, "First look at neo," 2016, https://www.jauntvr.com/neo-first-look/.

[7] Lytro, "Built for the next generation of immersive storytelling," 2016, https://www.lytro.com/immerge.

[8] Selman Ergünay, Vladan Popovic, Kerem Seyid, and Yusuf Leblebici, "A novel hybrid architecture for real-time omnidirectional image reconstruction," in *Proceedings of the 9th International Conference on Distributed Smart Cameras*, New York, NY, USA, 2015, ICDSC '15, pp. 152–157, ACM.

[9] Naoki Chiba and Terence T. Huang, "Capturing spherical light fields of a real scene," in *ACM SIGGRAPH 2004 Posters*, New York, NY, USA, 2004, SIGGRAPH '04, pp. 49–, ACM.

[10] Paul Debevec, Greg Downing, Mark Bolas, Hsuen-Yueh Peng, and Jules Urbach, "Spherical light field environment capture for virtual reality using a motorized pan/tilt head and offset camera," in *ACM SIGGRAPH 2015 Posters*, New York, NY, USA, 2015, SIGGRAPH '15, pp. 30:1–30:1, ACM.

[11] Lode Jorissen, Patrik Goorts, Sammy Rogmans, Gauthier Lafruit, and Philippe Bekaert, "Multi-camera epipolar plane image feature detection for robust view synthesis," in *3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON), 2015*. IEEE, 2015, pp. 1–4.

[12] F Bettonvil, "Fisheye lenses," *WGN, Journal of the International Meteor Organization*, vol. 33, pp. 9–14, 2005.