Rendering of Stereoscopic 360° Views from Spherical Image Pairs

Dash Bodington, Jayant Thatte, Matthew Hu
{*dashb, jayantt, matthu*} @*stanford.edu*

Department of Electrical Engineering, Stanford University

June 5, 2015

Abstract

In this work, a technique for rendering stereoscopic 360° images from a pair of spherical images, for virtual reality applications, is presented. The spherical images are captured using a Ricoh Theta[1] camera from two viewpoints separated by a known, vertical displacement. Using vertical, rather than horizontal, camera displacement allows the computation of depth information in all viewing directions, except *zenith* and *nadir*, which is crucial for rendering an omnidirectional stereo view. Stereo correspondence information from a three-dimensional similarity accumulator is combined with the RGB information in the captured images to produce dense, noise-free disparity maps that preserve edge information. From cleaned disparity maps, depth information is calculated from disparity and pixels are remapped to a new viewpoint in which artifacts are filled before display on a Samsung Gear VR [2]. Using our method high quality stereo images can be rendered with a reasonably good perceived depth.

1 Introduction

Depth-based stereoscopic image rendering and 3D reconstruction has been an important area of research in multimedia, broadcasting and in computer vision. The area has received a lot of attention from the broadcast research community for its applications in 3D television (TV) [1], [2]. Whereas the classical approach to 3D TV requires the transmission of two video streams, the depth-based approach can achieve the result using only one video stream and one stream of depth maps, thus rendering 3D content using a smaller bandwidth [3], [4]. Similarly, depth-based 3D perception is an important pre-requisite for applications in computer vision and virtual and augmented reality [5], [6], [7].

The remainder of this report is structured as follows. Section 2 gives an overview of the equipment used for the project and its setup. Section 3 talks about the algorithms used for pixel-level, raw disparity computation and the various post-processing techniques used to generate the final disparity map. Section 4 gives a brief geometric description of stereo view generation algorithm and then moves on to talk about the technique used for inpainting the newly exposed areas due to *disocclusion*. Section 5 displays images at various stages in the processing pipeline and comments on the quality of the produced results. Section 6 outlines possible improvements to the work done in this project and the directions in which the work discussed in this report can be extended in future.

[1]`https://theta360.com/uk/`

[2]`http://www.samsung.com/global/microsite/gearvr/`

2 Equipment Setup and Preprocessing

2.1 Equipment Setup

Pairs of images used for disparity calculation and stereo view manipulation were captured with a Ricoh Theta camera. The Ricoh Theta has two opposite-facing 185° fish-eye lenses and sensors. This captures a 360° by 180° image of a scene when the two images are stitched together by the Theta software into an *equirectangular* image at a resolution of 1792 by 3584 pixels.

Capturing each scene requires recording two of these images from viewpoints displaced vertically by a known distance. The displacement must be sufficient to cause objects at selected depths to move one pixel or more, but large shifts can be problematic because objects are distorted as they move, and smaller pixel shifts can be calculated faster. Once processing is performed in MATLAB [3], the final stereo views are loaded onto the Samsung Gear VR, a virtual reality headset which houses a Samsung Galaxy S6 as a display. The device keeps track of the direction its wearer is facing and displays the correct image to create an immersive stereoscopic 360° experience.

2.2 Preprocessing

The image captured using the setup described above need to be aligned to account for possible horizontal translation and rotation of one viewpoint with respect to the other. Since

[3]`http://www.mathworks.com/products/matlab/`

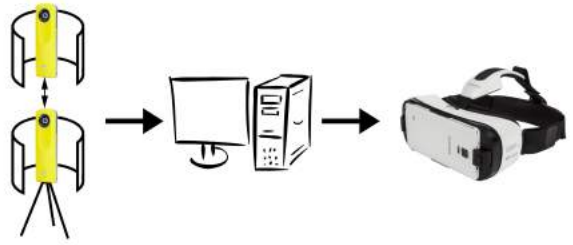

Figure 1: Image pairs are captured with the Ricoh Theta on a tripod, processed in Matlab, and displayed on a Samsung Gear VR Headset.

the camera is mounted on a tripod, which is leveled using a leveling bubble on the tripod head, any undesired rotation and horizontal translation is going to be very small in magnitude (compared to the size of the image).

The images could therefore be aligned by holding one image constant and cyclically shifting the other image, one pixel column at a time, until maximum correlation was achieved between the two images. It was found that the horizontal shift between images in around 10-20 pixels (when the horizontal image dimension is 3584 pixels), which corresponds to a rotation of roughly 1-2°.

3 Disparity Computation

3.1 Initial Disparity Estimation

The first step in our process is to generate a disparity map from our two vertically shifted 360° images. The disparity map corresponds to the pixel-level shifts between the two images captured, which will then be used with our measured camera shift to calculate the depth map. We initially tried implementing a variety of techniques offered by different papers [8], [9]. However, the method that worked best with our setup was a slightly modified version of the similarity accumulator technique [10]. For this technique, the paper assumes that the disparity is only along a single direction (vertical in our case), and builds a three-dimensional similarity accumulator based on image shifts in that direction. The three-dimensional accumulator is essentially a stack of images, where each "layer" in the stack is the original image with a shifted second image subtracted from it.

First, we establish a range of disparities that are possible $d = [0, d_m]$. For each value of $d = d_i$, we shift the upper image vertically by d_i and store the absolute difference between the shifted image with the lower image. Each difference image gets stored along the third dimension, resulting in an accumulator that is $1792x3584xdm + 1$.

The paper then proposes that the initial disparity D can then be calculated with the accumulator a with:

$$D(u, v) = \text{argmin}_{d \in [d_{\min}, d_{\max}]} a(u, v, d) \tag{1}$$

That is, essentially, for each pixel in the difference image, find the smallest value along the third-dimension of the accumulator and that corresponding image shift will be the disparity value at that pixel.

We discovered that while this technique was must faster than the typical block-matching algorithm, it is also much noisier due to its pixel-by-pixel search. Even with the modified mean filter proposed by the paper to address this problem, we decided to use a 6x6 window search through the accumulator instead of a 1x1. For each pixel, we look at the 6x6 window surrounding that pixel and find the disparity that gives the minimum euclidean norm of that window. Finally, the paper suggests heavy median filtering and morphological closing in order to smooth out some other noise. Some of the results from this initial disparity estimation are shown in 2.

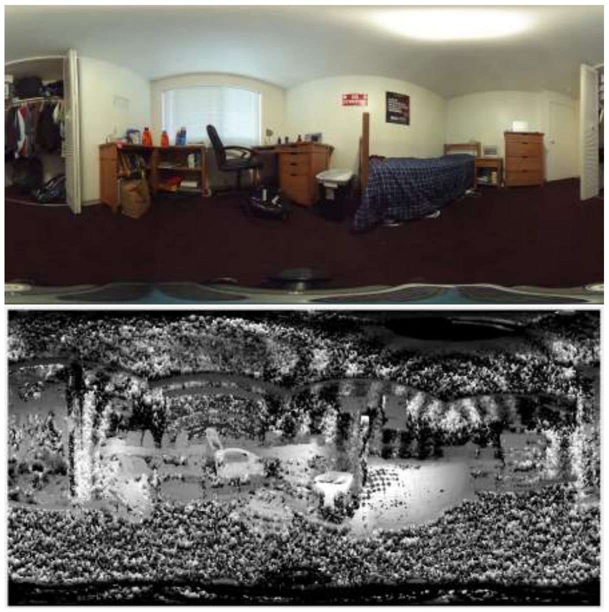

Figure 2: The top image is the lower image taken by the ricoh theta. The bottom image is the initial disparity estimation. Brighter pixels correspond to greater disparity and therefore closer image

3.2 Post-Processing

As we can see, the algorithm described above provides a good first-pass disparity estimation. However, there is still a lot of processing that needs to be done in order to improve our disparity map. First of all, there is a lot of noise along the floor and walls of the images. Generally, it is fairly difficult for algorithms to calculate disparity for regions with low texture, and as a result, we get a lot of noise. Looking at the accumulator we had generated, we discovered that these pixels corresponding to these areas of low texture had very low variance across the disparity axis in the accumulator. We therefore made a search through each pixel in the accumulator and zeroed out all disparities corresponding to low variance across the third-dimension of the accumulator. This removed a lot of the noise in the image shown 3.

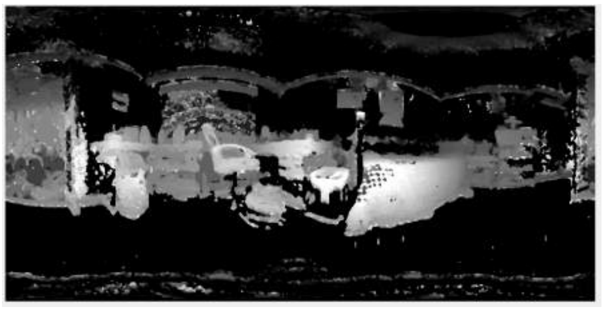
Figure 3: The disparity map with the noisy regions zeroed out. The disparity map now appears a lot cleaner

Median filtering and morphological closing earlier removed some of the smaller holes in the image, but there are still large undefined regions due to the noise removal. We performed region labeling for all of the undefined regions of the disparity map and classified them according to their size. For smaller regions, we simply filled them in by averaging the pixel values along their borders. This was accomplished by performing morphological edge detection on each of the undefined regions to get an outer edge, and then averaging over the disparity values corresponding to that border. This method works because most of the small holes are generally from the un-textured regions of a single object. Therefore by averaging over the border disparity values, we generally get objects with consistent disparity values.

For larger regions, we followed the method outlined in [10] by looking along the scan lines and only using disparity values directly above and below the undefined region to fill in the holes. To implement this, we essentially broke each region into separate columns and filled in each group of columns with the median of the bordering disparity values. Finally, we used a large horizontal blurring filter in order to smooth out the sharp edges and filled in the large holes with the result. The resulting images are shown 4.

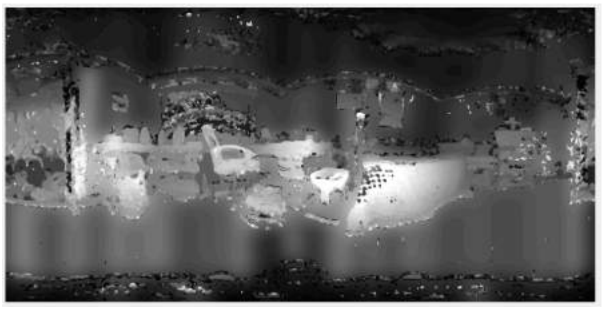
Figure 4: The resulting disparity map with the undefined regions filled in.

Overall, our disparity maps give a good estimate of the image shift. In many of the images shown above, we can see a gradient of the pixels gradually getting darker as the objects get further away. We can also distinguish separate objects fairly well and see disparity discontinuities between close and far objects. The main problem with this disparity map is that it is still somewhat noisy. Also, edges are not clearly defined due to our closing filter to fill in holes. As a result, we needed further processing in order to improve this map.

3.3 Disparity Averaging based on Image Segmentation

The disparity map generated thus far, still has a considerable pixel-level noise rendering it unusable for any meaningful stereo application. Most simple low pass filtering operations fail to preserve sharp edges and discontinuities in the disparity. Hence, segmentation-based averaging was used to smooth the generated disparity map.

In this novel technique, the image is first segmented using RGB data in the image using a modified version of Density-Based Spatial Clustering of Applications with Noise (DBSCAN). The advantage of this technique is that it can form arbitrarily shaped, connected segments based on the color information. An image is split into several thousand such segments and disparity values are averaged over each segment. Since the segmentation is based on the image color, edge information is preserved very well. Similarly, having a large number of segments ensures that each object in the image is comprised of several segments, which helps in preserving disparity gradients across objects.

At the end of this step, the disparity map is noise-free and still preserves the edge information very well. This is the final version of the disparity map, which will be used further, for rendering stereo views. Below are sample results from this stage. It can be seen that even though all the pixel-level noise is removed, the disparity gradient across the bed is still preserved and sharp discontinuities in disparity are maintained.

4 Stereo View Rendering

Before new views can be rendered, simple processing must be done to the disparity to convert it to distance, and smooth it slightly. Knowing camera displacement between the captured images allows the conversion of disparity values to accurate distances from the camera. Once distance values are calculated, the RGBD equirectangular image contains information for every pixel on both color and spatial position in spherical coordinates.

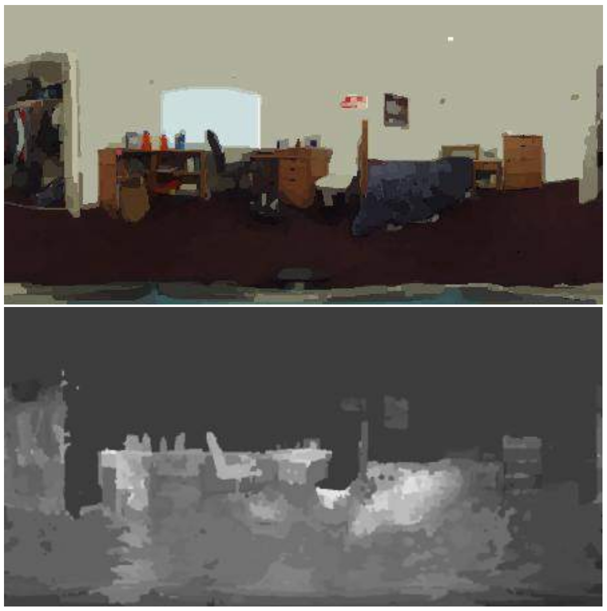

Figure 5: The top image is the room segmented by color. The bottom image is the resulting disparity map after averaging the segmented sections.

4.1 Viewpoint Geometries

In order to produce a stereoscopic scene, image transformations based on depth information must be performed. These transformations can be of two types, realistic transformations, and pseudo transformations. The goal of both is to produce pairs of images which appear 3D, but a realistic transformation is incompatible with a display device such as an Oculus or Samsung Gear VR, which only accepts two equirectangular images per 360 scene.

In a realistic environment, the parallax shift of objects is a function of angular position, horizontally. There is no parallax effect, for example, at 90 degrees from the direction a person is facing. If an image like this was put into an Oculus device, it could not be immersive because a 90 degree head turn would face the user toward a region of no parallax, while he would have maximum parallax in peripheral vision. This project implemented this realistic 3D view generation, but a different geometry was used to generate display views for the Oculus device.

Because the VR headset only displays one 360 image for each eye, the movement of every object, regardless of horizontal position, is calculated as if the object was in the center of the visual field. This approximation still produces quality 3D views because object shifts always appear correct in the center of the subjects visual field, and as visual acuity falls off away from the center, it is harder to tell that offsets are not realistically rendered.

4.2 Inpainting

Because producing stereoscopic views is a depth-based remapping of all pixel values, it is likely that some areas in the generated view will be empty. These image gaps need to be filled to make the image acceptable. There are many ways to fill these holes in the image, but because the goal is to make the image look realistic and to preserve sharp edges on shifted foreground objects, this project used a depth-sensitive content mirroring operation to add content to holes. This method is relatively simple compared to some content generation inpainting techniques, but works sufficiently well to avoid noticeable image artifacts in standard (non-translated) stereoscopic rendering [11]. Unlike most inpainting tasks, however, we have knowledge of depth, and want to preserve discontinuity on one side of the hole where the boundary of an object is.

To fill holes, the image is scanned line by line. When the beginning of a hole is encountered, its length on the line is calculated, and the depths of the pixels bounding the hole are compared. To preserve the discontinuity at the close end of the hole and blend with the further end, information from the deeper side is mirrored into the hole. This process is performed line-by-line for the entire image, and is then performed column-by-column in a similar manner so that holes are filled with an average of mirrored information from their deeper vertical and horizontal edges.

This process works well when holes are small, however, creating large holes can be problematic because it becomes obvious that information has been copied. Stereoscopic views usually contain holes which are not noticeable when filled, but translated views are more distorted and contain more holes which are harder to discretely fill.

4.3 Translated Views

Though generating stereo images involves translated viewpoints, this project also produced stereoscopic views whose central position was different from the camera position. Rendering a translated view, as if the stereo views were generated from a translated camera, is slightly more difficult than a simple stereoscpoic view. Before performing stereoscopic view generation, all pixel positions are calculated in cartesian coordinates so that they can be modified according to the new imaginary camera position, and then are transformed back to spherical coordinates where the pseudo view generation is simpler, and remapping is more straightforward. When generating true (realistic) stereoscopic views, all pixel location changes are done in cartesian coordinates because the entire remapping can be done in one step.

5 Results

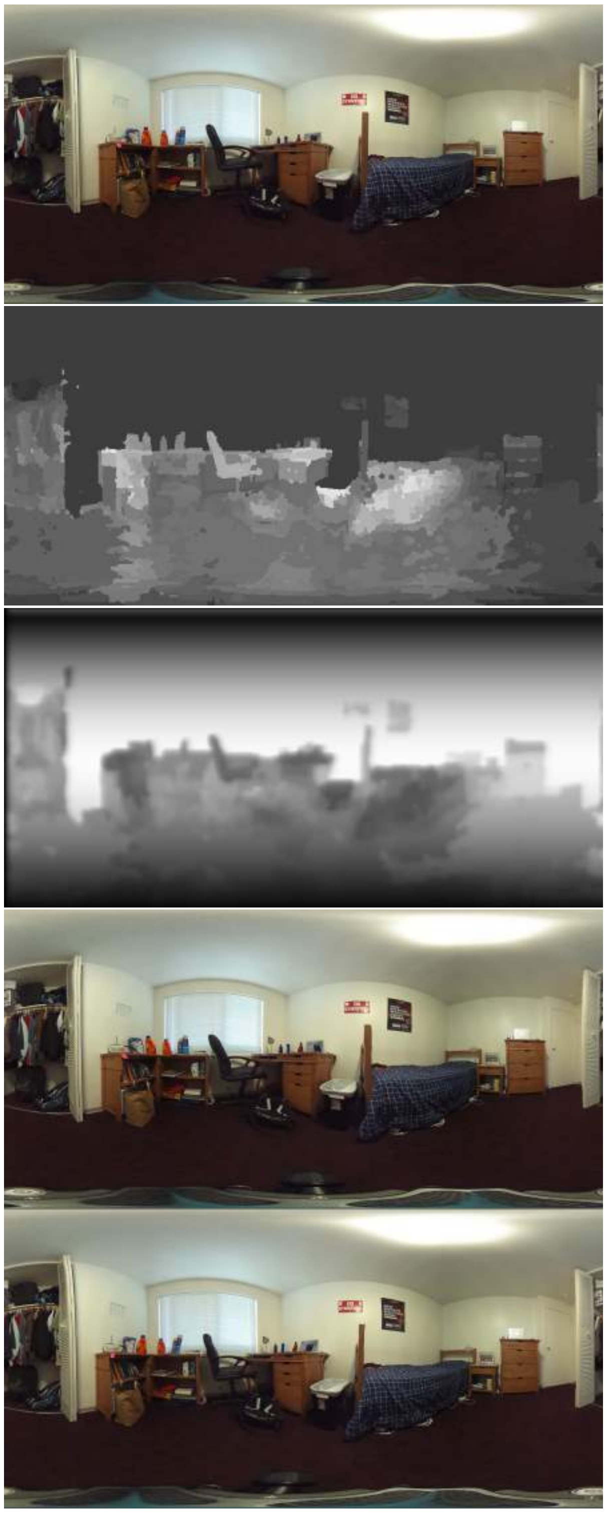

Figure 6: Images from top to bottom: original lower image, segmented disparity map, depth map, left view, and right view.

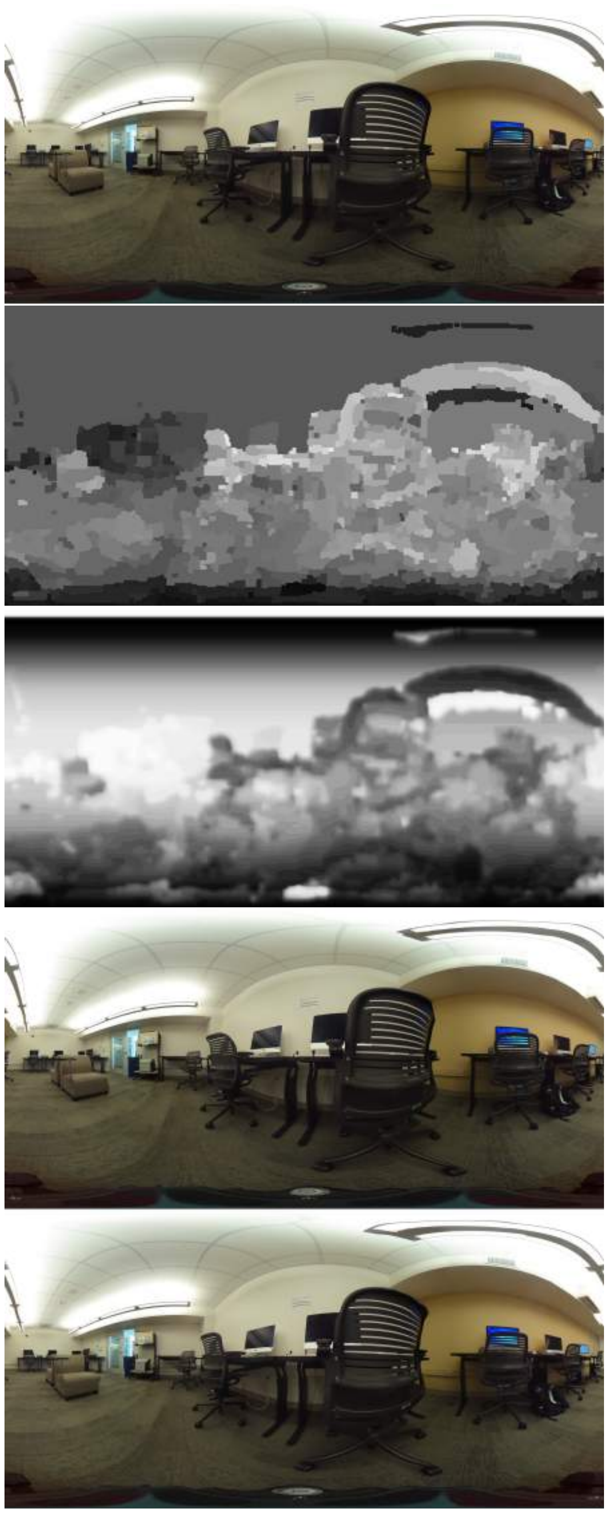

Figure 7: Images from top to bottom: original lower image, segmented disparity map, depth map, left view, and right view.

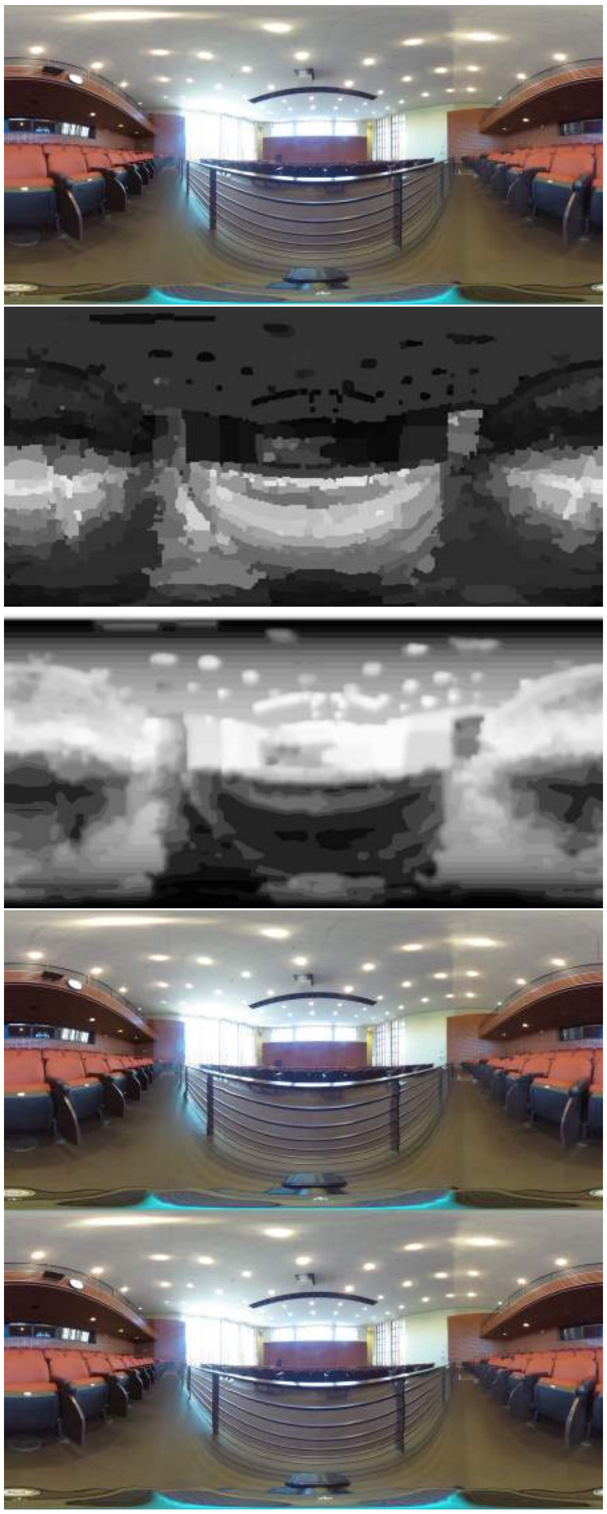

Figure 8: Images from top to bottom: original lower image, segmented disparity map, depth map, left view, and right view.

The stereo images rendered using our method are of good quality and contain minimal inpainting and disparity artifacts. The images also produce a reasonably good perceived depth. The image produced do have some geometric distortions. A possible cause for this could be the high geometric distortion introduced by the fish-eye lenses in Ricoh Theta, which has not been compensated for, in this project. It is expected that running camera calibration and lens *undistortion* algorithms prior to using our method will produce better results.

Another challenge is that since our camera displacement is vertical, it is hard to estimate depth information of vertical objects such as pillars, that have no texture along the vertical direction. Similarly, by the virtue of the equipment setup, there cannot be any disparity information along the baseline i.e. towards zenith and *nadir*. This is, however, generally not an issue, since most human observers looking at an image tend to focus on image regions within a band around the equator, and two-image 360° VR views are not capable of displaying depth at angles near the poles.

6 Future Work

While this project successfully demonstrated a capture-to-display stereoscopic 360° system, there is a great deal of improvement which can still be done. The current system uses one camera which is moved between photos, but attaching two Ricoh Theta cameras and triggering them together would allow for live scene capture with compact hardware, a more elegant solution than current concepts which rely on large assemblies of many cameras.

Processing is likely the area in which the greatest amount of improvement could be made. Disparity calculations may be improved by taking into account camera distortion during matching, better filtering could reduce artifacts in disparity with less distortion, and improved inpainting methods would make larger translations from the camera position possible. Processing speed is also an area in which significant improvement is necessary.

Another possible extension is to add the effect of live modification of view with translation by pre-computing a stack of images and pulling the right image into the display based on the VR translation output, or posiibly computing on-the-fly, should such a processing speed be achieved.

Improvements in the processing would open up a large number of possibilities in entertainment, such as stereoscopic 360° video in which the viewer is able to look around and make small head movements.

References

[1] L. Zhang and W. J. Tam, “Stereoscopic image generation based on depth images for 3d tv,” *Broadcasting, IEEE Transactions on*, vol. 51, no. 2, pp. 191–199, June 2005.

[2] G. Thomas and O. Grau, “3d image sequence acquisition for tv & film production,” in *1st International*

Symposium on 3D Data Processing Visualization and Transmission, 2002, pp. 320–326.

[3] J. Flack, P. Harman, and S. Fox, "Low bandwidth stereoscopic image encoding and transmission," in *Proceedings of SPIE-IS&T Electronic Imaging*, 2003, pp. 206–214.

[4] C. Fehn, "Depth-image-based rendering (dibr), compression, and transmission for a new approach on 3d-tv," pp. 93–104, 2004. [Online]. Available: http://dx.doi.org/10.1117/12.524762

[5] H. Kim and A. Hilton, "3d scene reconstruction from multiple spherical stereo pairs," *International Journal of Computer Vision*, vol. 104, no. 1, pp. 94–116, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11263-013-0616-1

[6] H. Kim, M. Sarim, T. Takai, J.-Y. Guillemaut, and A. Hilton, "Dynamic 3d scene reconstruction in outdoor environments," in *3DPVT*, 2010, <p>Copyright 2010 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE.</p>. [Online]. Available: http://epubs.surrey.ac.uk/110210/

[7] M. Schonbein and A. Geiger, "Omnidirectional 3d reconstruction in augmented manhattan worlds," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, Sept 2014, pp. 716–723.

[8] K.-H. Bae, D.-S. Yi, S. C. Kim, and E.-S. Kim, "A bidirectional stereo matching algorithm based on adaptive matching window," in *Applications of Digital Image Processing XXVIII*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, A. G. Tescher, Ed., vol. 5909, Aug. 2005, pp. 676–683.

[9] H. Kim and K. Sohn, "Hierarchical depth estimation for image synthesis in mixed reality," pp. 544–553, 2003. [Online]. Available: http://dx.doi.org/10.1117/12.473879

[10] J. Schmidt, H. Niemann, and S. Vogt, "Dense disparity maps in real-time with an application to augmented reality," in *Applications of Computer Vision, 2002. (WACV 2002). Proceedings. Sixth IEEE Workshop on*, 2002, pp. 225–230.

[11] A. Criminisi, P. Perez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *Image Processing, IEEE Transactions on*, vol. 13, no. 9, pp. 1200–1212, Sept 2004.

A Team Roles

Dash Bodington: Focused primarily on post-disparity calculations and processing. This includes disparity filtering, disparity to distance calculation, viewpoint geometries and pixel remapping, and inpainting. He also participated in team meetings and material preparation.

Jayant Thatte: Focused on stages in the pipeline up to and including the generation of final disparity maps. This included initial aligning of images and computing raw disparity maps as well as post-processing. Tried out a couple of different methods for computing raw disparity — Window-based disparity estimation and bidirectional disparity mapping [8], [9], which were later depreciated in favor of accumulator-based disparity calculation. Also, implemented image segmentation based on RGB data and segmentation-based disparity averaging to produce the final disparity maps.

Matthew Hu: Focused primarily on disparity map calculations. Generated initial disparity maps, worked on hole filling, and tested various techniques to improve maps. Also worked with the rest of the team on setup, image capture, and poster/report preparation.