A low-cost, end-to-end virtual reality concept From capture to viewing on mobile devices

Ben Webster (bmwebster@wisc.edu): *Software, Image Processing, Writing*
Chase Roossin (roossin@wisc.edu): *Mobile, Web, Backend, Writing*
Felix Tsao (ftsao@wisc.edu): *Hardware, Software, Image Processing, Writing*

***Abstract*—Our project presents a low-cost, end-to-end virtual reality solution which handles capture of virtual reality video, processing/stitching VR video and lastly, end-user viewing on mobile phones. In this report we discuss the methods and design considerations, largely driven by minimizing equipment costs and development time, all in consideration of increasing accessibility of VR content generation to consumers, artists, hobbyists and low-budget productions. We also explore the issue of organizing virtual reality experiences towards a particular application. Currently, VR content is presented in fairly disorganized collections on YouTube, Facebook and other outlets. We also discuss our prototype mobile app, *Exhibit*, that hones VR technology to a particular application for enhancing the real-estate experience.**

Keywords*—*virtual reality, computer vision, multi-camera systems, camera calibration, light-field, homography, feature matching, projective geometry, ionic framework, 3D-printing, computer-aided design, LaTeX.

I. Introduction/History

If we define virtual reality as an attempt at creating an illusion that a viewer is present somewhere they are not, the earliest attempt at this definition of virtual reality was in the nineteenth century with 360 degree mural panoramic paintings (Figure 1). Another attempt at virtual reality was stereoscopic viewers, also developed in the 19th century (Figure 2).

Figure 1 (left) Battle of Borodino 1812
Figure 2 (right) Becker's Stereo View 1860

In the 20th century, Edward Link created the Link trainer, the first example of a virtual reality flight simulator [1][2]. This was a major development in the virtual reality movement that allowed pilots to train safely. Twenty years later, in the mid 1950s, cinematographer Morton Heilig created the sensorama. The sensorama was an extremely high-tech theatre for the time, that incorporated stereo speakers (new for the time), stereoscopic displays, fans, smell generators, and motorized chairs, to immerse the viewer in the virtual world of the film. In the 1960s, the Philco Corporation created the first proper precursor to current virtual reality that we know today. They built a product called Headsight, that had a video screen for each eye, and head tracking ability built in. This originally was not meant for virtual reality, because the term had not been coined yet, but was cutting edge for the time when computers still filled several rooms of a building. In the 1990s, countless companies hopped on board with the idea of virtual reality. Many of these entities were video game companies, but practically all of the products released were marked as failures. In 1993, Sega announced that they would be releasing a VR headset, but never got past the prototype phase, and failed to release it. In 1995, Nintendo managed to release a virtual reality product called Virtual Boy, but the device was not a success on the market, and stands as a major failure for Nintendo [1][2].

II. Motivation / Related Work

Today's VR development

With the explosive development of computer and mobile technology, high-resolution graphics displays and various sensors have become widely available in the United States and other parts of the world. Companies like Google have extended the mobile platform into an accessible virtual reality headset through the introduction of inexpensive Google Cardboard viewers and a 360 video player on YouTube. The rise of virtual reality today, seems to have potential to fulfill the empty promises made in the 1990s, when claims were made ahead of their time about virtual reality [3].

In the coming years, up to 2020, the field of VR is projected to grow from a few billion dollar market to a thirty billion dollar market [4]. When Mark Zuckerberg first wore Oculus Rift, he said that he knew he was "seeing the next great technology platform thats going to redefine the way we connect with each other in the future" [5]. An example of Facebook's push for VR development can be observed through their two billion dollar acquisition of Oculus Rift, deployment of 360 video in user newsfeeds and announcement of their new open-source high video data throughput network switch [6][7].

As advancements and widespread adoption of mobile phones have enabled easily accessible VR headsets and viewers,

Thanks to Garage Physics for camera hardware funds and 3D printing
Also thanks to other contributors for mobile development from CS407:
Eric Smith (ejsmith@wisc.edu)
Samareh Shahmohammadi (shahmohammad@wisc.edu)
Vanessa Cao (cao54@wisc.edu)

optical content generation remains to be a formidable challenge at the industry-level, let alone for the consumer and hobbyist levels. In the past two years, various companies have been investing in virtual reality camera technology and virtual reality production software tools. Google Jump, was one of the first commercially available industry solutions for select customers in November 2015 for 15,000 USD (Figure 3). Following soon after, announcement of Nokia Ozo at 60,000 USD (Figure 4) and Lytro Immerge at 200,000+ USD (figure 5) are planned to deploy to the professional market Q1 of 2016. It is safe to say that these price ranges are far beyond what many consumers and hobbyists can afford. Additionally, there is a significant challenge in high-throughput networking to support the massive data rates captured by VR cameras, but we will not address them in this report [7]. Lytro Immerge is announced to be shipped with an accompanying processing server and users for Google Jump currently queue their projects on Google's servers for processing.

VR camera developments released/announced in Q4 2015

Figure 3: Jump

Figure 4: Ozo

Figure 5: Immerge

Lack of targeted application

Even though there has been explosive growth in VR technologies, most capture solutions reside in professional and corporate markets and current consumer solutions are relatively poor and expensive. Ricoh Theta S currently retails for 350 USD, but is on backorder. Video samples and specifications about the Theta S state that the entire 4π steradian field of view is mapped to a 1080p resolution stream [8]. This results in a low effective resolution to a viewer when watching through a VR headset. Additionally, more expensive consumer solutions like Bublcam, 799 USD, [9] have been suffering from poor initial prototype reviews, and have led to a receding release date. The first Bublcam group shipment went out December 15th, 2015 and reviews are just trickling in.

We discuss our design of an easy to distribute VR camera setup which can be recreated with common, off-the-shelf items. Our design also creates higher-quality content for less cost than current consumer solutions. Additionally, expensive panoramic video stitching software and multi-camera mounts also present a hurdle for budding VR enthusiasts and consumers. Assuming high accessibility of multi-camera systems, the panoramic video stitching software presents another hundreds of dollars in cost [10][11].

The last issue is that once VR video has been created with expensive hardware and software tools, content sharing outlets are fairly untargeted. Currently, VR content resides in large collections on YouTube and Facebook with little organization. There are few applications to view VR content with a specific purpose like NYTVR and VRSE, but are still quite general. We discuss our mobile application prototype *Exhibit*, which directs VR to a real-estate application. Home buyers, renters and investors can virtually visit physically distant properties from the comfort of a single location. These virtual experiences can be done in rapid succession which eliminates travel costs and buying time. Property sellers and realtors can easily add listings for static VR photos, but will require a VR video solution as mentioned earlier, if they want to provide motion and guided tours.

III. Problem Statement

Current hurdles in optical VR content generation include expensive hardware and software tools. Another issue is lack of VR viewing outlets with particular applications. We discuss how we address these two issues with an arbitrary, easy to distribute multi-camera design, simplified video stitching process and targeted real-estate mobile application called *Exhibit.*

IV. Methods / Techniques

In this section we discuss our solution and design choices, which are motivated by driving down costs for VR video generation and viewing. Our goal is to take an arbitrary number of wide-angle video cameras, oriented in a flat circular arrangement and create a stitched panoramic video file, that is intended to be mapped onto a 3D viewing sphere. We stitch together multiple video tracks on a desktop/laptop machine and serve it as a conventional video file to mobile phones for viewing. The mobile phones render the panoramic video file on a viewing sphere and displays a subset of the viewing sphere based on the gyroscope orientation of the phone. Since the phone is fixed onto a user's face, it provides head/neck orientation information.

Figure 6: User viewing VR content through a Google Cardboard viewer

Tools and Libraries

Any assortment of the following hardware can be used and more cameras are ideal but ultimately drives up cost and computation. We found that 6 cameras is barely enough to cover 360° about the z-axis using SJ4000 cameras.

Table 1: Hardware and Electronics

General Item(s)	Cost
6× SJ4000 Cameras 1080p @ 30fps 170° vertical FOV	$510
3D printed mount *See included STL software files*	Free for students, otherwise ≤ $100 from local 3D printing hub
6× Samsung EVO Class 10 SDHC Cards	$45
Standard Photo Tripod	–
Standard ISA Desktop/Laptop Multicore CPU, GPU ideal CSL Enterprise i7-4970/GTX750	–

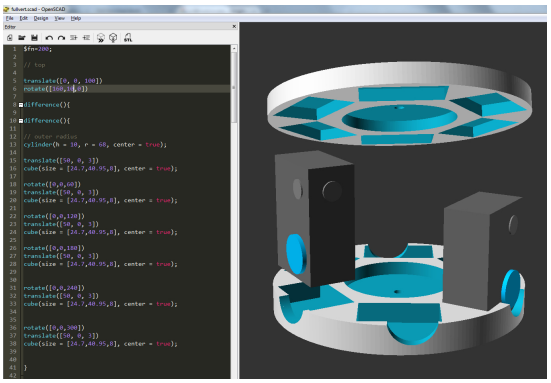
Figures 7,8: SJCAM Multi-Camera System and 3D-printed mount

Included SCAD files have adjustable dimension parameters to rapidly permute new camera mounts to fit arbitrary numbers of GoPro, NoPro and other small, rectangular cameras. Our budget allowed for six cameras.

Table 2: Software and Libraries

Name	Cost
OpenSCAD 2015.3	Free, GNU Public License
Adobe After Effects Adobe Media Encoder CC 2015, Windows 7 x64	Free for students, on most campus computers otherwise free 1 month trial, $30/mo.
Ionic Mobile Framework	Free, MIT License
Three.js WebGL Library	Free, MIT License
OpenCV 3.0.0	Free, New BSD License

Figure 9: Easy-to-adapt 3D-printable mount for other camera types



Total Costs

Total student cost for 6 camera system: $555.00
Conservative non-student cost for same system: $685.00

Hardware constraints / guided software workflow

From the arbitrary 3D camera mount design, we introduce two constraints that simplify the stitching process. The assumptions introduced are that the relative positions of the video tracks in relation to one another are static and that they are symmetrically oriented in a flat horizontal plane where each lens is pointed normal to the equator of the projection (Mercator, equirectangular, spherocylindrical). With these two assumptions and constraints, we can leverage the simplicity of 2D translation for registering relative video tracks and radial symmetric properties to apply the same warping to all video tracks.

The following initial stitching method was a human guided stitching process and ultimately ended up implementing our final demos. Our intended automation of this initial process using C++/OpenCV was not producing the results shown in the next section, which are desired. We discuss our automation challenges later after the main intuition of our stitching pipeline. Unlike homework 4, which uses a global projection technique, our initial method uses a local parametric warp to approximate sections of an equirectangular projection map. Registration of features and blending is similar to homework 4 to stitch together separate tracks.

Prototype estimator for Mercator Projection with Bézier warp

The initial stitching prototype results are done by human interface with a video compositing program, Adobe After Effects, to help build intuition on construction of a Mercator-like panorama. We perform each step in a rigid and mechanical way considering how these steps would be expressed using standard programming techniques.

As the cameras are a common radius away from the optical center and rotated about the z-axis, we make the assumption that our video tracks are translated by a simple x offset and allow for a small error offset in y, when working in a 2D video composition. This technique is similar to registration by translation when using a cylindrical projection for creating a panorama. Additionally, since the cameras are effectively identical and oriented in a radially symmetric manner, we make the assumption that the warping from projection for each video track is the same.

To approximate the warp by projection for a single video track, we use a Bézier or parametric warping filter by specifying a set of control points in After Effects to produce an "hourglass" looking result shown in Figure 10. We intuit the control values to accomodate matching of most features at the frame edges. We didn't think of acquiring calibration images for the cameras, but those would help with choosing the correct Bézier control points. For video tracks that have a larger vertical field of view, this warping pattern extends into a "tornado" or "wormhole", where the pixels farther from the horizontal and vertical mid-lines of the original frame are

warped near the left/right corners of the global compositing frame.

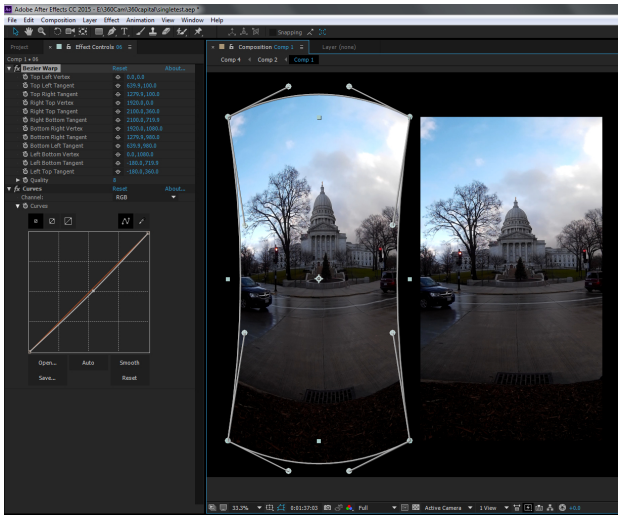


Figure 10: Bézier warped frame and its original in After Effects

We apply the same warping pattern to the remaining video tracks and register them by x, y translation or more colloquially, by "clicking and dragging" until unique objects line up in both frames. We apply color and exposure correction using a RBG curves tool and feather the edges with a mask to blend the seams. We pad the left and right of the six video tracks with a video track modulo to track number for perspective wrap-around about the z-axis.

Figure 11: All six video tracks with same warping and addition pad tracks on both sides, registered and blended

When mapped onto a viewing sphere, our tracks appear as planes that intersect at an angle dependent on the number of cameras:

Figure 12: Warped 2D video tracks appear as planes on viewing sphere

Lastly, we crop the video to remove the jagged effect of the intersecting planes to produce a cylindrical-like screen and scale/crop the width of the composition at the same position of the edge padding tracks to produce a continuous perspective wrap-around about the z-axis:

Figure 13: Final screenshot of output video

We serve this video VR viewing outlets on mobile devices, YouTube 360 and Facebook 360.

Video Results

YouTube 360 (View in Google Chrome Web Browser):
https://www.youtube.com/watch?v=lM7lKqry0ZM

We sort of stumbled across parametric Bézier warping by intuitive trial and error. We initially experimented with cylindrical, spherical and Macator projections using various warpers from OpenCV 3.0.0 but found that there was a large error in matching of objects between frames. We were able to intuit that a 'pinch' along the horizontal midline of frames would reduce the error of feature registration between frames from observation and some understanding of projective maps [12]. One of the warping tools in After Effects we saw fit was the Bézier warp and looking back, we also think that it also does some lens abberation correction as well for our results [13]. Addtionally, in a fairly recent mathematics paper, Bézier projections have been seen to provide highly accurate generation of 3D computer-aided design meshes on a local level [14].

Exhibit: Mobile Application

After generating a stitched equirectangular video file, it can be uploaded to VR viewing outlets like YouTube 360, Facebook 360 and applications like *Exhibit*. Using the Three.js WebGL, library we create a simple scene consisting of a spherical shell centered at (0,0,0), a virtual stereo camera also at (0,0,0) and an ambient light source. We use the video file as the texture for the spherical shell and negate the x dimension of the shell so that the video is not "flipped" as the camera is inside the shell. The virtual stereo camera only provides a pseudo-stereo as we didn't generate separate stereoscopic video tracks for each eye. We include our 3D scene as `app.js`.

Figure 14: Promotion for Exhibit, showcases split VR viewer

V. Issues and Future Improvements

Artifacts

Our video result shown in the previous link has artifacts that are noticeable when moving objects transition between seams. Additionally, there are some visible mis-registration of features in the frames. This can be allieved by taking calibration footage to find better Bézier control points and introducing more cameras. However, the second option drives up cost and computation resources.

Automation Issues

With our 6 camera solution, the manufacturer specifications suggested a 170° vertical field-of-view and about 90° horizontal for each camera. While there was overlap between adjacent video tracks, it was thinner than expected and relative size can be seen in Figure 11. When we attempted to begin automating the process using OpenCV 3.0.0, we had issues with find enough features to properly return a translation using both sample stitching pipelines. Running the OpenCV stitchers on images with larger common areas worked fine. We also tried adjusting work and seam megapixel search areas but it did not help. Future six camera configurations with SJ4000 should sacrifice vertical field of view and utilize a horizontal seating for the cameras to provide a larger overlapping between video tracks. We have CAD'ed a mount with horizontal seating but haven't been able to print it due to time and printer maintenance.

Additionally, we found no Bèzier surface module in OpenCV 3.0.0. A future project for implementing a Bèzier warp for the library could be useful for the method we discussed as well as other applications. We include our experimentation C++ code in videostitcher.cpp

Exposure Locking

Addtionally, inconsistent exposure poses an inconvenience which can be relieved if one manages to find out which command code on the SJ4000 is for exposure lock. The current SJ4000 menu does not present an exposure locking feature. When connected to an SJ4000 at `192.168.1.254` with a computer over WiFi, we know that commands can be issued through a URL command of the form: `192.168.1.254/?custom=1&cmd={}&par={}`

where `par` is some parity flag (1, 2, 3...), i.e. when we request,

```
192.168.1.254/?custom=1&cmd=3014
```

we get an XML list of some subset of camera commands:

```
<Cmd>2002</Cmd> Resolution
<Cmd>2003</Cmd> Cyclic Record
<Cmd>2004</Cmd> WDR
<Cmd>2006</Cmd> Motion Detection
<Cmd>2007</Cmd> Audio
<Cmd>2008</Cmd> Date Stamp
<Cmd>2010</Cmd> Live View Size
<Cmd>1004</Cmd> Capture Mode
<Cmd>1002</Cmd> Image Size
<Cmd>1005</Cmd> Quality
<Cmd>1006</Cmd> Sharpness
<Cmd>1007</Cmd> White Balance
<Cmd>1008</Cmd> Color
<Cmd>1009</Cmd> ISO
<Cmd>2005</Cmd> Exposure
<Cmd>1011</Cmd> Anti Shake
<Cmd>3025</Cmd> Frequency
<Cmd>3026</Cmd> Rotate
<Cmd>3011</Cmd> Reset to Defaults
<Cmd>3010</Cmd> Format
<Cmd>3007</Cmd> Auto Power Off
<Cmd>3003</Cmd> WiFiName
<Cmd>3004</Cmd> Password
<Cmd>3008</Cmd> DV Language
```

The camera also serves its feed at:
`rtsp://192.168.1.254/sjcam.mov`

VI. Conclusion

In the past, many dreamed of Virtual Reality with high hopes but fell short due to technical issues. Significant advancements in mobile technology over the past decades have enabled accessibility of powerful technologies to many people. We hope that this time, virtual reality will develop into a beneficial technology for all and enhance communication of our ideas and visions with one another.

ACKNOWLEDGMENT

Thanks again to Garage Physics for funding for camera hardware and 3D-printing services.

Also, those who helped with the mobile application in CS407: Eric Smith, Samareh Shahmohammadi and Vanessa Cao.

Lastly, thanks to Professor Dyer for a great semester!

CODE AND LIBRARIES

OpenCV 3.0.0, C++, 100 lines
Three.js, Javascript, 200 lines for 3D scene only
OpenSCAD, CAD markup, 100 lines

TASKS

Ben Webster: Contributed to writing introduction and helped with other parts of paper. Also, aided in discussion and development of stitching pipeline for arbitrary camera mount. Created powerpoint presentation and helped code mobile viewing application.

Chase Roossin: Contributed to development of VR viewing application, website, backend and writing.

Felix Tsao: Contributed to development of VR camera mount, stitching pipeline, mobile VR viewer and writing.

REFERENCES

[1] "History Of Virtual Reality." History. N.p., 2015. Web. 16 Dec. 2015.

[2] Riegler, Alexander, Markus F. Peschl, Karl Edlinger, Gnther Fleck, and Walter Feigl. "Virtual Reality." Abstracts of. Frankfurt Am Main: Peter Lang, 2001. Web. 16 Dec. 2015.

[3] http://www.nbcnews.com/tech/innovation/promise-virtual-reality-starting-look-very-real-n482631

[4] Merel, Tim. "Augmented And Virtual Reality To Hit $150 Billion, Disrupting Mobile By 2020." TechCrunch. Tech Crunch, 6 Apr. 2015. Web. 19 Dec. 2015.

[5] Sena, Pete. "What The Growth Of Virtual Reality Will Mean For Brands." TechCrunch. Tech Crunch, 20 Oct. 2015. Web. 16 Dec. 2015.

[6] Maher, Saba. (2015, September 23) *Introducing 360 Video on Facebook* [Online]. Facebook Newsroom.
Available: http://newsroom.fb.com/news/2015/09/introducing-360-video-on-facebook/

[7] Meritt, Rick. (2015, November 19) *Facebook Shifts Switch to 100G* [Online]. EETimes.
Available: http://www.eetimes.com/document.asp?doc_id=1328303

[8] https://theta360.com/en/

[9] http://www.bublcam.com/

[10] http://store.kolor.com/360-degree-video/software/video-stitching-software/autopano-video-2-x.html

[11] http://www.videostitch.com/

[12] http://www.kolor.com/wikien/action/view/Understanding_Projecting_Modes

[13] https://helpx.adobe.com/after-effects/using/distort-effects.html#bezier_warp_effect

[14] http://arxiv.org/abs/1404.7155

[15] OpenCV http://docs.opencv.org/master/