Omni-directional stereo for 360° 3D virual reality video

Prashanth Chandran
ETH Zurich
www.ethz.ch
chandranp@student.ethz.ch

Sasha Pagani
ETH Zurich
www.ethz.ch
paganis@student.ethz.ch

Julia Giger
ETH Zurich
www.ethz.ch
jgiger@student.ethz.ch

Abstract

The aim of our project is to produce omni-directional stereo images and videos, which are for example viewable in the Google Cardboard. The input for our pipeline are the images of a camera rig, which are stitched together for receiving omni-directional images. This process has to been carried out twice, once for the left and once for the right eye to get an omni-directional stereo image.

1. Introduction

The stereo view of our eyes is created by fusing the image from the left eye with the one from the right eye. This enables us to perceive a stereo impression of the world. Therefore, to produce stereo videos, we need to capture synchronized videos from two cameras set apart at interpupillary distance (IPD), which denotes the distance between the two eyes and is on average about 6.4 cm. However, the goal of this project is not only to produce stereo, but omni-directional, which means 360°, videos. The first simple idea coming in mind to capture such videos is to place two omni-directional cameras with a distance of IPD. One issue with such an approach is that the two cameras will see each other, which is undesirable. However, the more important problem is that objects lying on the line between the two camera centers will have no disparity (see figure 1). Disparity denotes the amount of shift in the image position between the left and the right eye. Therefore, this simple solution does not work and would not produce the desired omni-directional stereo videos.

The ideal solution would be to have a stereo image pair for every head orientation, for example one pair for each degree (see figure 2). However, this would be a huge amount of images. The main idea is to approximate this optimal view by only capturing the central ray of each camera instead of the full image at each head orientation (see figure 3). Figure 4 illustrates the extension of this approach to 360°.

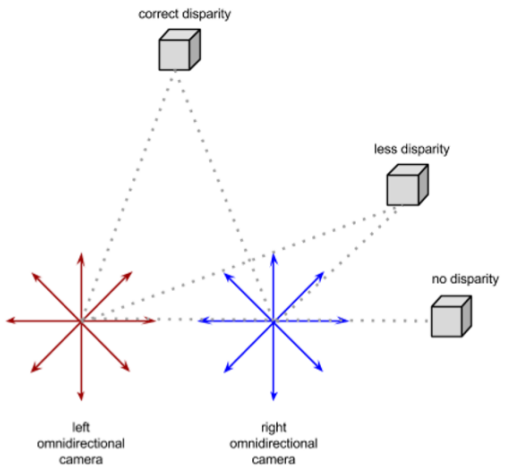

Figure 1. Illustration of two 360° cameras placed next to each other.

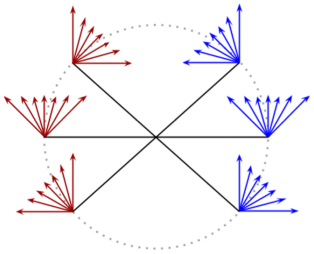

Figure 2. This image shows the most optimal situation, where a full image is captured for each head direction. The red rays illustrate the left eye and the blue ones the left eye.

2. Related work

The whole project is based on the paper ”Jump: Virtual Reality Video” by Anderson et al. Therefore, our main pipeline is the same as described in Anderson et al. However, we did not implement the sophisticated algorithm for the flow computation. We used a provided method of OpenCV for the flow estimation. Furthermore, we simpli-

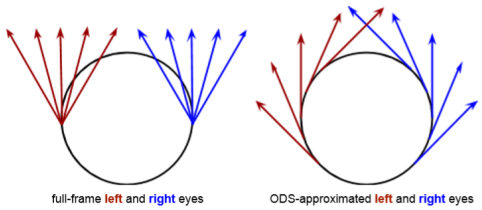

Figure 3. The left image shows the rays for capturing the full image at one head direction. The right image illustrates the ODS approximation by using only the central ray of each camera position.

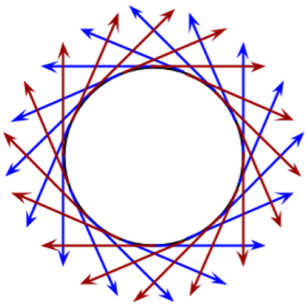

Figure 4. This image shows the ODS approximation for 360°. The red rays illustrates the left eye and the blue rays the right eye.

fied the exposure correction and the composition step. For the implementation details, we also used the article "Rendering Omni-directional Stereo Content" from Google.

3. Method

An overview of our pipeline can be found in figure 5. The inputs are 10 synchronized videos from a camera rig. The first step is the calibration of the different cameras of the rig, which includes the computation of the intrinsics and extrinsics of each camera. The next step is the optical flow estimation between the neighboring camera pairs. For having nice transitions in the final ODS stitch, an exposure correction between neighboring image pairs is applied. Based on the computed flow values, the view interpolation between the images is calculated and they are stitched together as a final step, which results in the desired ODS video.

3.1. Camera calibration

Additionally to the dataset, consisting of 10 videos, we received a calibration file from our supervisor, which contained the calibration data for each camera. This calibration data consists of the intrinsic and relative extrinsic of each camera. For the computation in the later stages of our pipeline, we also needed the absolute extrinsics of each camera, which were computed with the following formula: $E_i = ...$

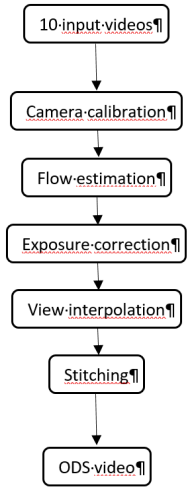

Figure 5. Overview of the pipeline.

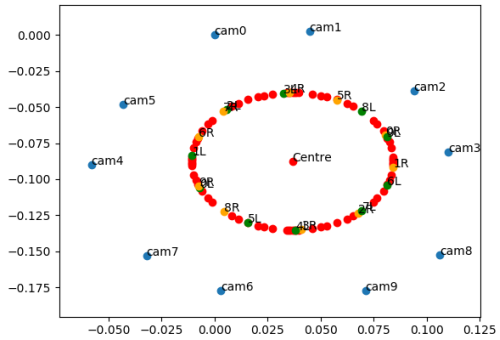

Figure 6. An illustration of our camera rig based on the calibration file.

3.2. Flow estimation

We did not implement such a sophisticated flow estimation algorithm as in Anderson et al. Instead, we used an existing per-pixel flow computation method from the OpenCV library.

3.3. Exposure correction

For the exposure correction, we interpolate linearly between the average image intensity of the neighboring image pairs.

3.4. View interpolation

We used the following interpolation from Anderson et al. for our view interpolation:

$$\theta_p = \frac{(\theta_b - \theta_1) * \theta_0 (\theta_0 - \theta_a) * \theta_1}{\theta_b - \theta_a + \theta_0 - \theta_1}$$

Where θ_0 and θ_1 are the headings of the two cameras in the ODS stitch. θ_a is the heading of a point in the first

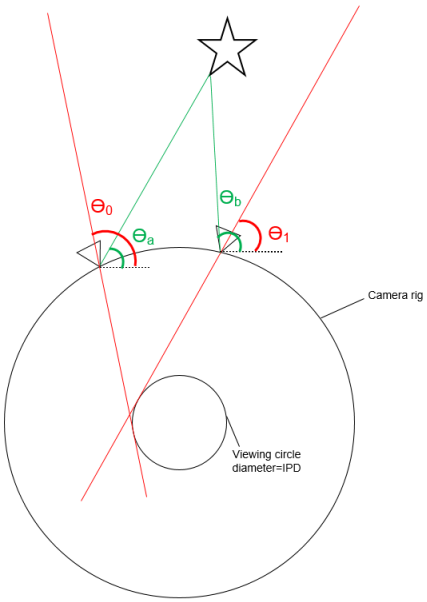

Figure 7. This is an illustration of the different angles used in the interpolation formula.

camera and θ_b is the heading of the same point in the second camera (see figure 7).

3.5. Stitching

4. Results

5. Discussion

6. Conclusion