CS231A Project Report: 3D Capture and Reconstruction Techniques in Project Starline by Google

Sheng Zhang
SUID: 06488070
sheng2@stanford.edu

Abstract

Google's Project Starline processing was implemented and tested in this project. As a necessary step, the camera calibration algorithm by Microsoft's Zhengyou Zhang was implemented from scratch. RGBD captures were then processed by Starline's processing algorithm interpreted from the paper. The results show that the camera calibration step likely has too much extrinsic calibration error, leading to 3D reconstruction error. The interpreted starline algorithm performs well, with potential improvement identified.

1. Introduction

In the final project, the algorithms used in Google's Starline project, a 3D face capture and reconstruction system [2], is implemented.

This project's use case is appealing as it aims at improving the user experiences of long distance conference call or video chat. This is especially becoming attractive amid a global pandemic that has lasted for more than two years now, where people rely more and more on virtual meetings. On the other hand, this project nicely includes several topics introduced in the class, including but not limited to: camera calibration, depths estimation, stereo view geometry and metrology and etc. It would a good opportunity to practice the theory taught in the class.

The problem in essence is a multi-view 3D reconstruction problem. First two RGBD cameras are set up to record videos of a person from left and right angle. In order to calculate the 3D world coordinate of the face, camera calibration is required to get focal length of the camera. In addition, the relative positioning of the cameras, i.e. the extrinsics, is also required in order to "stitch" the left and right views together correctly.

Next a novel view needs to be synthesized from the left and right RGBD frames. A brute force way of rendering is to go through all the pixels in the input views from left and right side, calculate the world coordinate and then project to the output view. Google's approach is to go from output views, and use ray casting to search for correct input view pixels.

2. Related Work

The most related work of this project would be the Project Starline by Google [2], which is what this project is based upon. Starline is a 3D telepresense system including capture, transmission, render, and autosteropic display.

There are other similar attempts at 3D video conferencing systems. Maimone [3] used five kinect units to capture 3D scene. Then each Kinect view was rasterized and triangulated into a depth map. The image rendering for each pixel was done using a normal-based weighting of the views, by checking the nearest surface.

Kuster et al. [1] realize symmetric telepresence. The 3D capture was done using a depth sensor, and the results are transmitted combining both color and depth. The deficiency in this approach was that only one depth sensor was used, leading to occlusion artifacts.

Zhang et al. [4] reconstruct multiple depths images using multiple IR projectors and cameras. A sparse 3D cloud was then created and transmitted with color streams. The difference from Google's apporach was that Starline does not render a point cloud, but rather fuses the color frame at the receiving end after transmitting color and depth frames separately.

3. Technical Approach

Detailed breakdown of the work is as follows:

- Camera calibration
- Capturing RGBD data
- Post processing including first compression and decompression
- Reconstruction to novel views

The following subsections describe my plan to implement this project.

3.1. Capture

Data source of this project will be captured by two iPhones pointing at the subject from different angles. The RGBD video files between the two cameras are synchronized offline.

To capture a reference image of the novel view for the purpose of quality assessment, A third RGB camera (reference camera) is used to capture from the novel view point. This camera is to be included in the calibration process to compute the extrinsics, such that the novel view can be reconstructed towards this reference camera direction.

3.2. Camera Calibration

3.2.1 Homography Estimation and Optimization

The first step of camera calibration is to estimate homographies for each calibration image. A ChArUco pattern is used in this project, and the pattern coordinate is easily calculated based on ChArUco checker size and ID size. OpenCV built-in function is used to recognize ChArUco ID and the corners.

Once the pattern coordinates $P_{ij} = [X_{ij}, Y_{ij}, 1]$ and their corresponding image coordinates $[u_{ij}, v_{ij}]$ are found, where $i = 0...M$ images and $j = 0...N$ points in the image. The Direct Linear Transform (DLT) method is used to compute homography. More specifically the following homogeneous system is setup and solved by singular value decomposition.

$$\begin{bmatrix} \mathbf{P_{00}} & \mathbf{0} & -u_{00}\mathbf{P_{00}} \\ \mathbf{0} & \mathbf{P_{00}} & -v_{00}\mathbf{P_{00}} \\ \cdots & & \\ \mathbf{P_{MN}} & \mathbf{0} & -u_{MN}\mathbf{P_{MN}} \\ \mathbf{0} & \mathbf{P_{MN}} & -v_{MN}\mathbf{P_{MN}} \end{bmatrix} \begin{bmatrix} h_{00} \\ h_{01} \\ h_{02} \\ h_{10} \\ h_{11} \\ h_{12} \\ h_{20} \\ h_{21} \\ h_{22} \end{bmatrix} = 0 \tag{1}$$

3.2.2 Camera Intrinsic and Extrinsic Extraction

Next, the camera intrinsic and extrinsic matrices are extracted from the homographies. The equations used in the implementation can be found in the original Zhang's paper [5]. Equation details are skipped for now in midterm report.

3.2.3 Lens Distortion Extraction

The camera radial lens distortion is modeled as follows

$$D(r, \mathbf{k}) = k_0 r + k_1 r \tag{2}$$

where r is radius in the normalized camera image plane. The distortion parameters are solved using least square method. Details are skipped for now in midterm report. From the results of camera calibration, the distortion is relatively low. Therefore it will be skipped in reconstruction process.

3.2.4 Parameter Refinement

Finally, all the parameters including intrinsic, extrinsic and distortion parameters are refiled using nonlinear optimization Levenberg-Marquart method. The Jacobian is calculated numerically.

3.3. Reconstruction and Rendering

3.3.1 Point Cloud Reconstruction

In this step, the world frame coordinates for each pixels in RGBD frame is calculated. Since depth is already known, the X and Y coordinates are simply calculated based on depth and camera intrinsics, as follows.

$$X = \frac{d(u - u_c)}{f_x} \tag{3}$$

$$Y = \frac{d(v - v_c)}{f_y} \tag{4}$$

3.3.2 Novel View Depth Map Synthesizing

Second step of the reconstruction is to reconstruct a depth map for the output novel view. This is a key implementation that is new to this paper. An efficient reconstruction method is proposed in the Starline paper. The fundamental idea is to search the depth of nearest surface from ray casted from each pixels in the novel view.

First an initial depth map is constructed. The initial depth estimate is not fully described in the paper. In the project the following procedure is used. First for every N pixels in the left and right depth map, the coordinate is transformed to novel view camera reference frame by utilizing camera extrinsics. Then the points are projected to the novel view image plane. Since not every pixels are processed, the resulting novel view depth map is at most $N \times N$ downsampled. The map is then upsampled to full resolution by utilizing simple image scaling process. Next a Gaussian filter is used to smooth out discontinuities. Finally the values of full resolution initial depth map are reduced by 100mm as a conservative initial guess of the novel view depth.

Then, for each pixel in novel view, we first take a conservative initial guess of the depth, and transform the point to left and right camera's reference frame (e.g. P_l, with its depth noted as $P_l.z$), and then to each camera's image plane (e.g. u_l, v_l for left camera) to look up the depth of the points that would have been projected to the same image coordinates (e.g. D_{l,u_l,v_l}). The signed distance (e.g. s_l)

is then calculated by subtracting guessed depth from actual depth. When the guessed points are behind the surface of the volume the signed distance is negative, while when the guessed points are in front of the surface of the volume the signed distance is positive. The weighted sum s of left and right side signed distances is then calculated and added to the guessed depth in novel view for the next iteration, until the weighted sum changes sign.

The weight of pixel i in camera $j, j \in$ left, right is defined as $w_j(i) = \min(.001/\sigma_i, 1)$ with $\sigma_i = \sqrt{\frac{1}{N_i}\sum_{k\in N_i}(d_i - d_k)^2}$. The essence of the weighting is that if depth has more variance (more noise) in a local area (e.g. 7x7 region), the pixel is given less weight.

Algorithm 2 describes details of my interpretation of the technique.

Algorithm 1 Starline Novel View Depth Map Synthesizing

```
1:  function GENERATENOVELDEPTH($D_l, D_r$)
2:      $D_i \leftarrow$ GenInitDepth($D_l, D_r$)
3:      $D_n \leftarrow 0$                        ▷ Initialize output depth map
4:      for $u, v \in D_n$ do
5:          $s \leftarrow$ INF
6:          $D_{n,u,v} \leftarrow D_{i,u,v}$
7:          while $s \geq 0$ do
8:              $P_n \leftarrow$ GenWorldCoord($u, v, D_n[u,v], K$)
9:              $P_l \leftarrow$ Transform($P_n, E_l$)
10:             $P_r \leftarrow$ Transform($P_n, E_r$)
11:             $u_l, v_l \leftarrow$ Project($P_l, K$)
12:             $u_r, v_r \leftarrow$ Project($P_r, K$)
13:             $s_l \leftarrow D_l[u_l, v_l] - P_l.z$
14:             $s_r \leftarrow D_r[u_r, v_r] - P_r.z$
15:             $s = w_l s_l + w_r s_r$
16:             $D_n[u,v] \leftarrow D_n[u,v] + 0.8s$
17:         end while
18:     end for
19:     return $D_n$
20: end function
21: ________________________________________
22: function GENINITDEPTH($D_l, D_r$)
23:     $i, j \leftarrow 0$
24:     for $u, v \in D_l, D_r$ do
25:         $P_l \leftarrow$ GenWorldCoord($u, v, D_l[u,v], K$)
26:         $P_r \leftarrow$ GenWorldCoord($u, v, D_r[u,v], K$)
27:         $u_l, v_l \leftarrow$ Project($P_l, K$)
28:         $u_r, v_r \leftarrow$ Project($P_r, K$)
29:         $D_{init}[i,j] \leftarrow$ mean($D_l[u_l, v_l], D_r[u_r, v_r]$)
30:     end for
31:     $D_i \leftarrow$ ImageUpSample($D_init, N$) $\times$ 0.5
32:     return $D_i$
33: end function
```

3.3.3 Rendering

On the rendering front, it is important to reconstruct a smooth surface for the face. The the image-based geometry fusion technique [2] described in the Starline paper also utilizes the weight calculated in the previous section. The pixel color of novel view frame is calculated by a weighted sum of input pixel colors from left and right view. As a result of the weighting, the pixel values from frames that have locally noisy depth capture is less weighted.

Algorithm 2 Starline Novel View Color Rendering

```
1:  function RENDERCOLOR(u, v, $D_n, RGB_l, RGB_r$)
2:      $C_{out} = None$             ▷ Initialize output color frame
3:      for $u, v \in C_{out}$ do
4:          $d \leftarrow D_n[u,v]$
5:          $P_w \leftarrow$ GenWorldCoord($u, v, d, K$)
6:          $P_l \leftarrow$ Transform($P_w, E_l$)
7:          $P_r \leftarrow$ Transform($P_w, E_r$)
8:          $u_l, v_l \leftarrow$ Project($P_l, K$)
9:          $u_r, v_r \leftarrow$ Project($P_r, K$)
10:         $Color_l \leftarrow RGB_l[u_l, v_l]$
11:         $Color_r \leftarrow RGB_r[u_r, v_r]$
12:         $C_{out}[u,v] = w_l Color_l + w_r Color_r$
13:     end for
14:     return $C_{out}$
15: end function
```

4. Results

4.1. Capture

The RGBD cameras are setup to ensure the subject's face can be captured in full. Figure 2 shows the RGBD camera output, including color image and depth image, respectively. Note that for the RGBD camera used, the raw RGB frame resolution does not match raw depth map resolution. Image reszing is done to match the two. It can be seen that the color and depth frames register to each other well.

4.2. Camera Calibration

The camera calibration algorithm described above has been implemented. During the verification of the code, it is found that a robust capture setup is essential to extract accurate extrinsics. Originally frames from a video files are used and frames from each camera corresponding to a global time stamp are used.

The intrinsic matrix extracted is as follows. It can be seen the gamma term -8.88 is much smaller than the focal length term 1248, indicating small skew of the sensor. Note that for the Principal Point Offset term 690 and 539 does not match half of the resolution (1440x1080) very well.

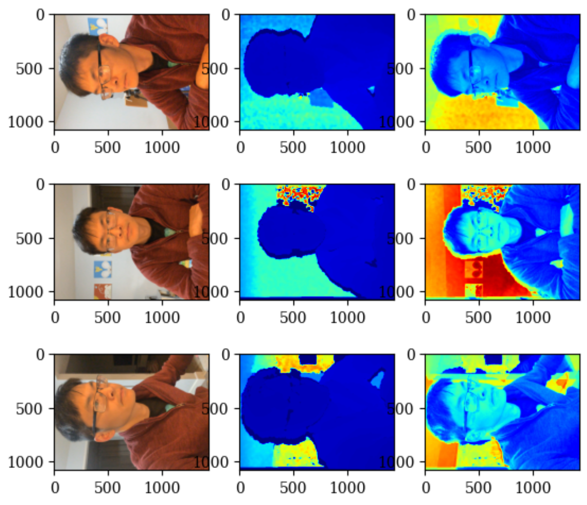


Figure 1: RGBD camera outputs

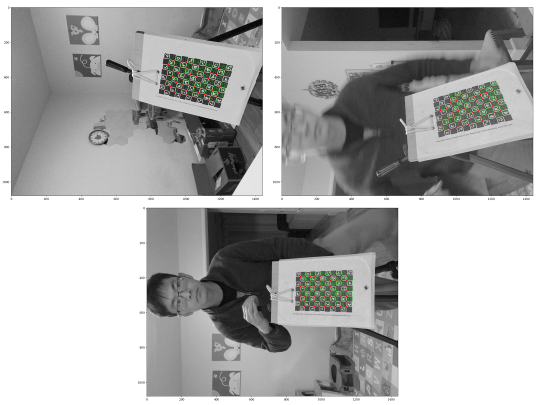


Figure 2: Calibration images

$$K = \begin{bmatrix} 1248.23 & -8.88 & 690.40 \\ 0. & 1248.66 & 539.24 \\ 0. & 0. & 1. \end{bmatrix} \quad (5)$$

The extrinsic matrix of the cameras are extracted as follows.

$$RT_{left} = \begin{bmatrix} 0.923 & 0.041 & -0.383 & 76.159 \\ 0.244 & -0.831 & 0.5 & -23.249 \\ -0.298 & -0.555 & -0.777 & 614.077 \end{bmatrix} \quad (6)$$

$$RT_{right} = \begin{bmatrix} 0.916 & -0.034 & -0.4 & 87.414 \\ -0.277 & -0.776 & -0.567 & 85.985 \\ -0.291 & 0.63 & -0.72 & 517.552 \end{bmatrix} \quad (7)$$

$$RT_{middle} = \begin{bmatrix} 0.92 & 0.02 & -0.391 & 86.976 \\ 0.014 & -1. & -0.018 & 27.184 \\ -0.391 & 0.011 & -0.92 & 677.89 \end{bmatrix} \quad (8)$$

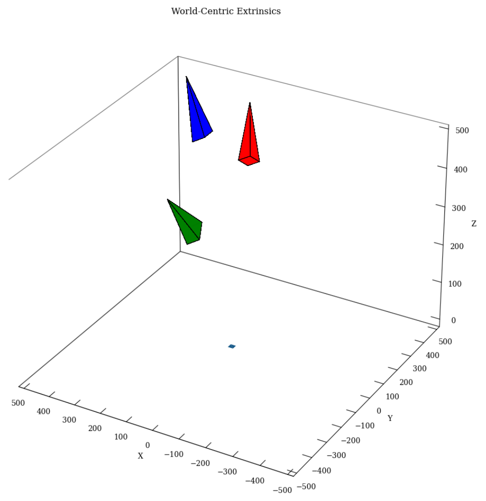


Figure 3: Extrinsic Calibration Illustration

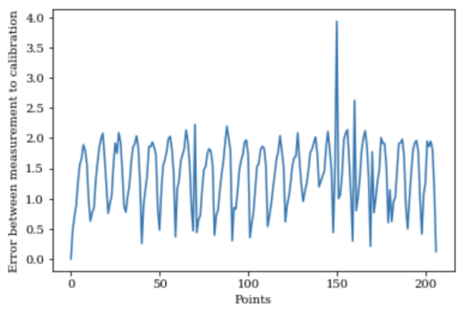
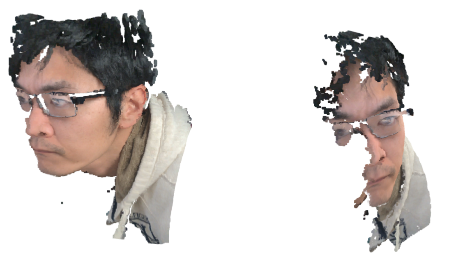


Figure 4: Pixel projection error after calibration

Figure 3 visualizes the extrinsics of the cameras, showing four views from left (blue), right (green), center (blue).

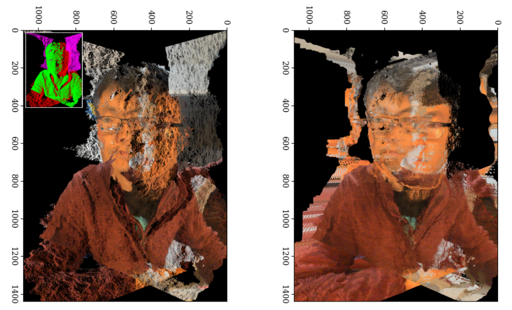
In order to quantify the calibration quality, the pixel projection error is calculated by projecting calibration pattern's world coordinates to the three camera image via calibrated intrinsic and extrinsic parameters; then compared with the measured coordinates. The mean square error of all detected pixels is 2.2.

Note that Initially the extrinsic calibration gives erroneous results. After investigation, it is found that two factors impacts the calibration accuracy. First, converting calibration image to gray scale reduces error. Second and more important, interpolate the checkerboard corners. Given the RGBD camera resolution is rather low (1440x1080), the quantization error of detected checkerboard corner coordinates likely introduces significant error that impacts extrinsic calibration.



(a) from left side (b) from right side

Figure 5: Constructed point cloud illustration



(a) Brute force rendering from all input pixels (b) Starline rendering aprach

Figure 6: Rendering results

4.3. Point Cloud Reconstruction

Figure 5 illustrates the point cloud reconstructed for left camera, rendered by Open3D for visualization.

4.4. Rendering

Figure 6 shows the rendering results. Obvisously the rendering result from this implementation is not very good. First, their is tearing in the faces, and the texture reconstruction is not very smooth. Second, it seems some pixels got fetched to the ouptut view twice, e.g. right side of the face. The following discusses the results. In the brute force rendering approach, the color map included in the figure marks the source of texture, where green pixels marks texture from right view, and red pixels marks texture from left view. It can be seen that the left and right view did not merge very well and a tearing can be seen in the middle of the face. Since there was no additional manipulation in this step, two possible reason can lead to this result. First, the depth map error may lead to incorrect 3D coordinates. This hypothesis can be largely ruled out due to inspection of good single view point cloud reconstruction in the prior section. Second, the camera extrinsic calibration result has error. This hypothesis is more probable since any error in extrinsics will place the point clouds from left and right view in a wrong location.

In terms of the repetitive structure on the right side of the face. It is due to the nature of rendering algorithm. In the ray casting algorithm, the signed distance is calculated as the algorithm march along the ray. Once the signed distance changes sign, it is concluded that the latest guess of depth is correct, and input pixel coordinates is calculated. This algorithm does not protect against ambiguity of search result. It is possible that two 3D points from search result can be projected to the same input view coordinates.

5. Conclusion

In summary, for this final project, algorithms from two papers are implemented and tested. First the camera calibration algorithm by Microsoft's Zhang; second the Starline rendering algorithm by Google. The results show that camera extrinsic calibration limited the reconstruction performance.

There are a few further improvement can be done if more time allowed. As already discussed in the results section, the nuances of camera calibration was the biggest learning and there are various pitfalls that can lead to calibration error. Some potential further improvements include increasing the RGBD camera resolution and increase the size of the calibration pattern. In the rendering step, the depth search can be made adaptive such that when signed distance is closer to zero, the increment in depth can be reduced. There can also be redundancy check, i.e. if an input pixel has already been queried from prior pixel rendering, it is likely one of the pixels would be rendered wrong, we refine both pixel rendering to arrive at more accurate result.

Code repo is located at https://github.com/sheng-sz/sz-starline

References

[1] C. Kuster, N. Ranieri, Agustina, H. Zimmer, J. Bazin, C. Sun, T. Popa, and M. Gross. Towards next generation 3d teleconferencing systems. In *2012 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON)*, pages 1–4, 2012.

[2] J. Lawrence, D. B. Goldman, S. Achar, G. M. Blascovich, J. G. Desloge, T. Fortes, E. M. Gomez, S. Häberling, H. Hoppe, A. Huibers, C. Knaus, B. Kuschak, R. Martin-Brualla, H. Nover, A. I. Russell, S. M. Seitz, and K. Tong. Project starline: A high-fidelity telepresence system. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 40(6), 2021.

[3] A. Maimone, J. Bidwell, K. Peng, and H. Fuchs. Enhanced personal autostereoscopic telepresence system using commodity depth cameras. *Computers Graphics*, 36(7):791–807, 2012. Augmented Reality Computer Graphics in China.

[4] C. Zhang, Q. Cai, P. Chou, Z. Zhang, , and P. A. Chou. Viewport: A fully distributed immersive teleconferencing system with infrared dot pattern. Technical Report MSR-TR-2012-60, April 2012.

[5] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:1330–1334, December 2000. MSR-TR-98-71, Updated March 25, 1999.