Volumetric Capture

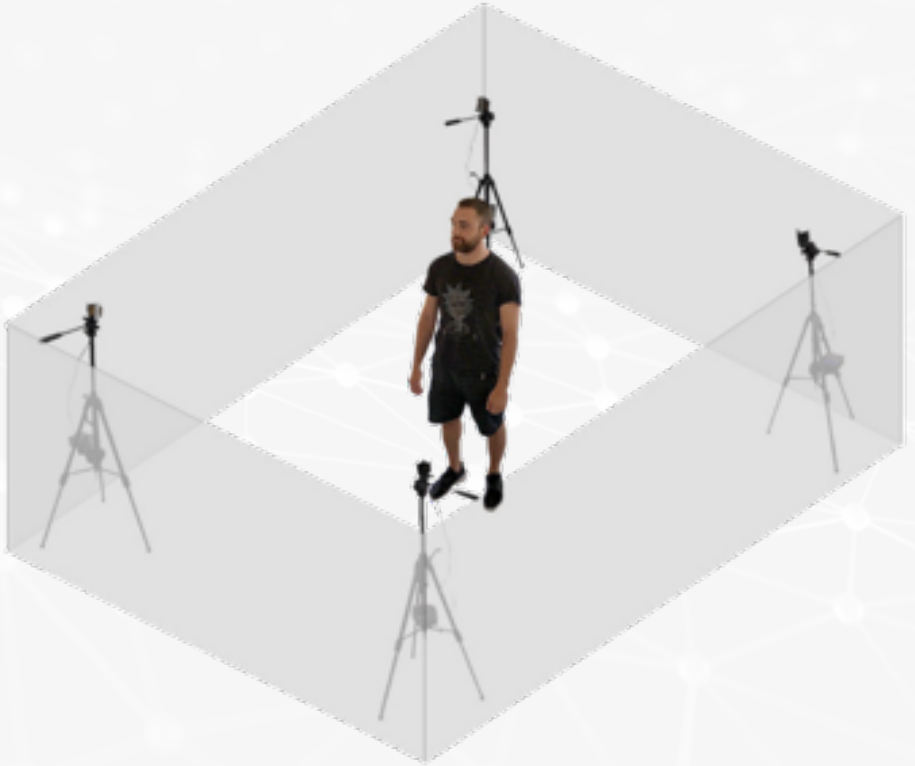
Marcel Bruckner, Kevin Bein, Moiz Sajid

Goal

- Reconstruct volumetric

Setup

- 4x Intel RealSense Depth Camera D415
- 4x Tripods
- USB Hub
- USB Active Repeater Cables

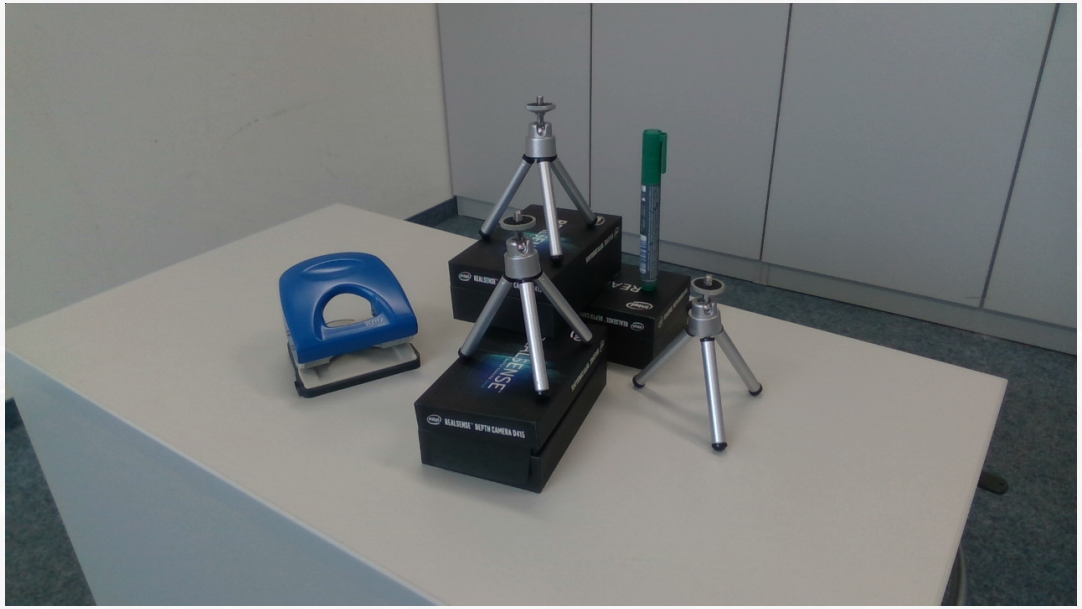


Pipeline Overview

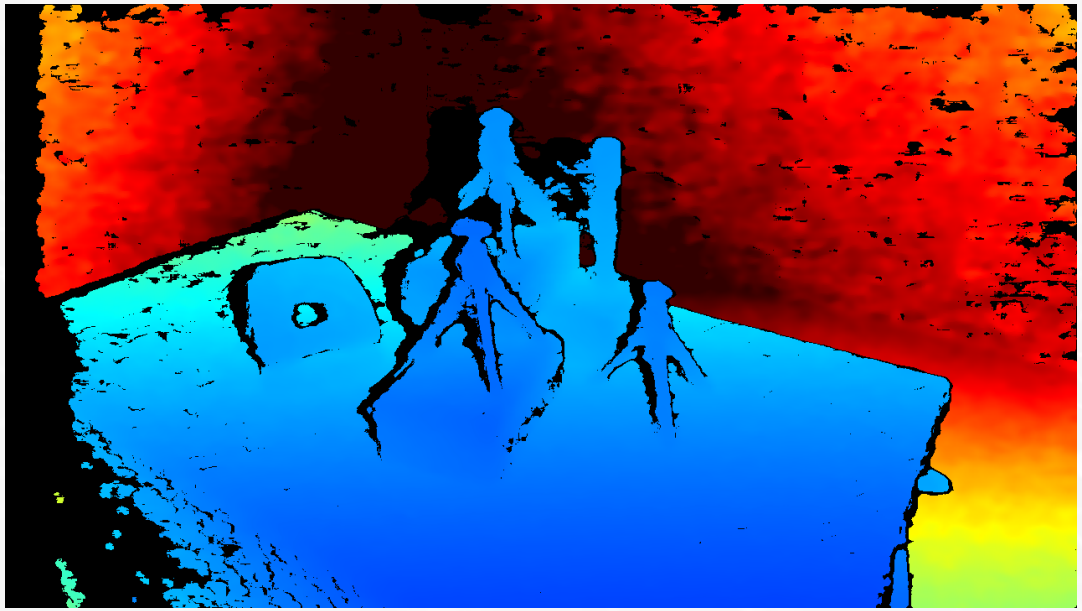
1. RGB and depth frame capture
2. Post-Processing
3. Align RGB and depth frame
4. ChArUco Board detection
5. Callibration via Procrustes, Bundle Adjustment and ICP
6. Fusion into Voxelgrid
7. Compute Marching Cubes

1. RGB and depth frame capture

- 1920x1080 RGB resolution

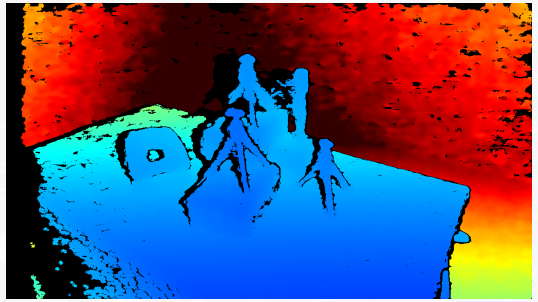


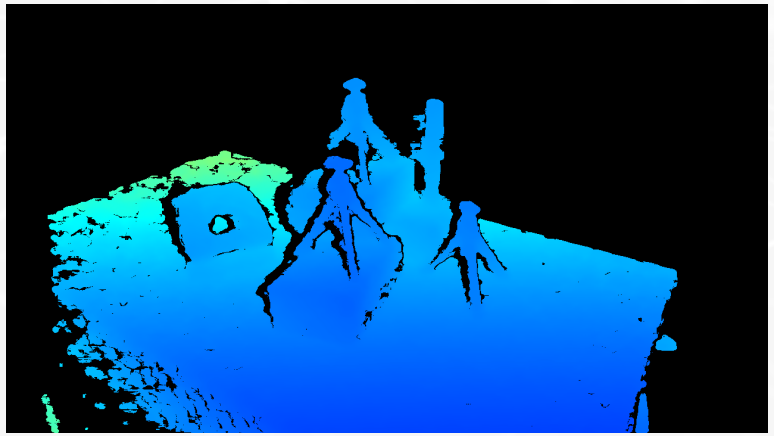
- 1280x720 active stereo depth

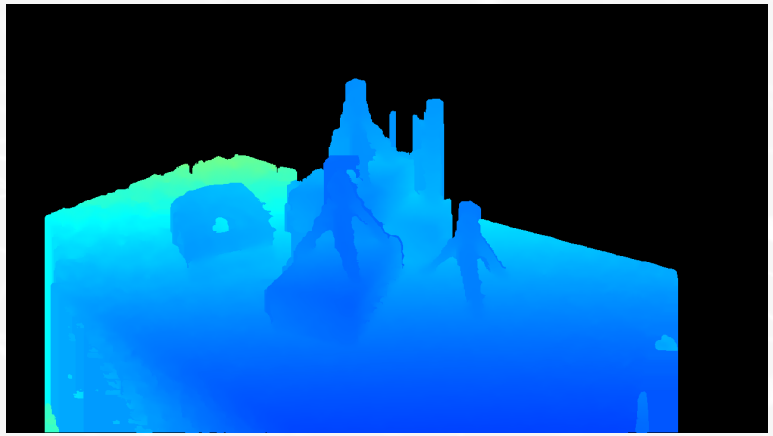


2. Post-Processing

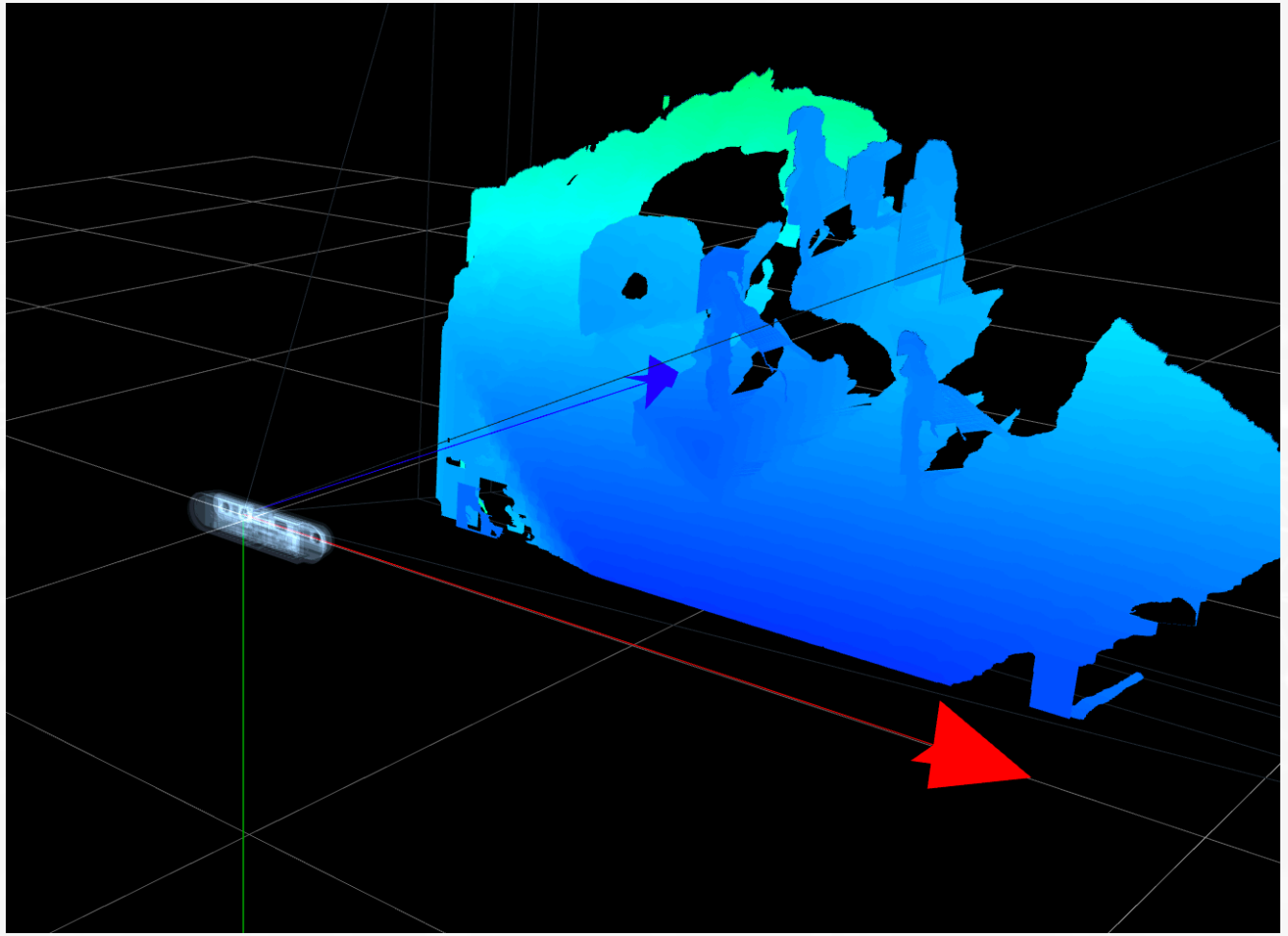
2.1 Raw Input → 2.2 Threshold Filter → 2.3 Hole-Filling Filter





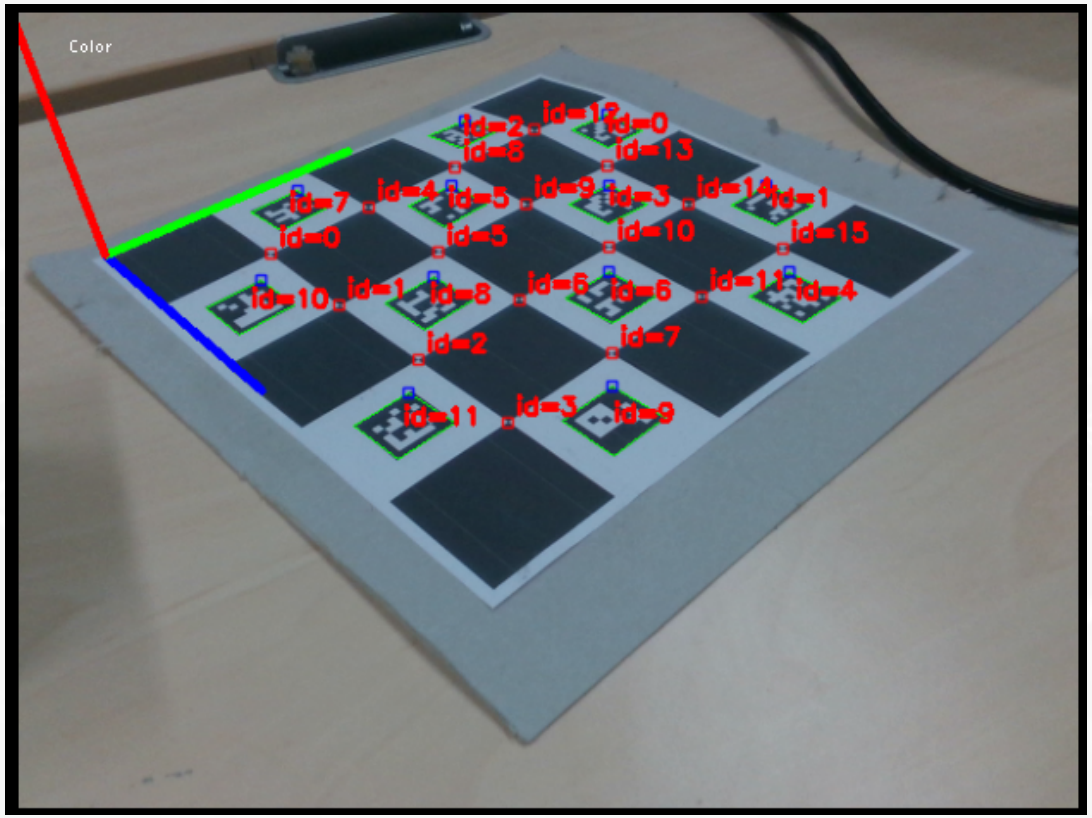


3. Align RGB and depth frame



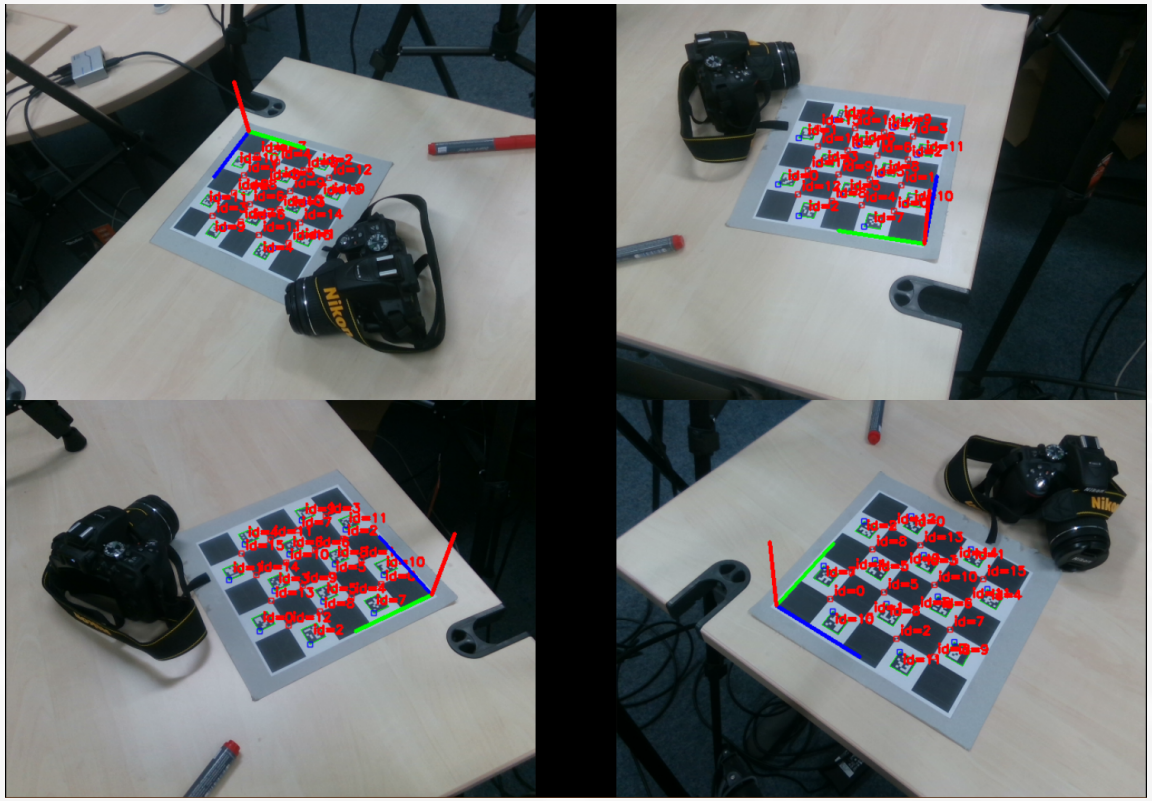
4. ChArUco Board detection

- Detect 12 ChArUco corners in all camera streams



4. ChArUco Board detection

- Detect 12 ChArUco corners in all camera streams



5. Callibration via Procrustes, Bundle Adjustment and ICP

- Use detected ChArUco corners as correspondences
- Initialize algorithm with Procrustes
- Minimze error function
- Run ICP for further refinement

6. Fusion into Voxelgrid

- Calculate TSDF values
- Fuse aligned points into Voxelgrid

7. Compute Marching Cubes

Implementation Details

- Librealsense API
  - Camera communication
  - Post-processing filters: Hole-filling, Depth threshold
  - RGB and Depth alignment
- OpenCV for ChArUco corner detection
- OpenGL Rendering
- OpenGL Shading Language
- Immediate Mode GUI (imgui) for UX

Implementation Details

- Custom implementation of Procrustes, Bundle Adjustment and ICP using Ceres solver
- Custom implementation of Marching Cubes

- Not used:
  - Point Cloud Library (PCL)
  - CUDA
  - DirectX

Live Demo

Thank you! Questions?