Volumetric Capture

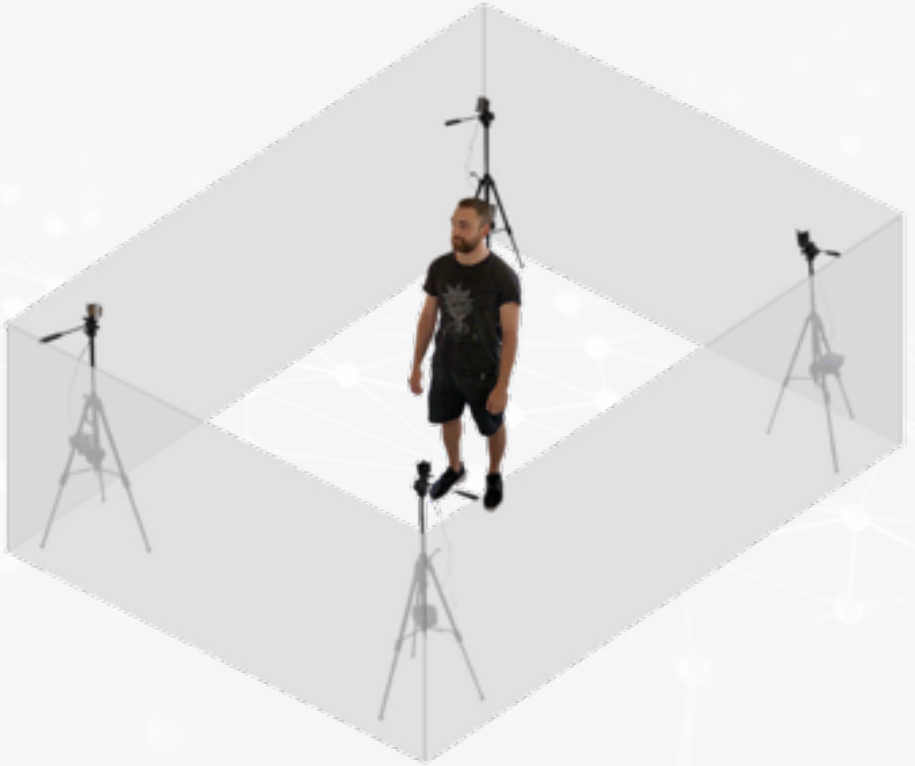
Marcel Bruckner, Kevin Bein, Moiz Sajid

Goal

- Frame Fusion

Setup

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



Setup Problems

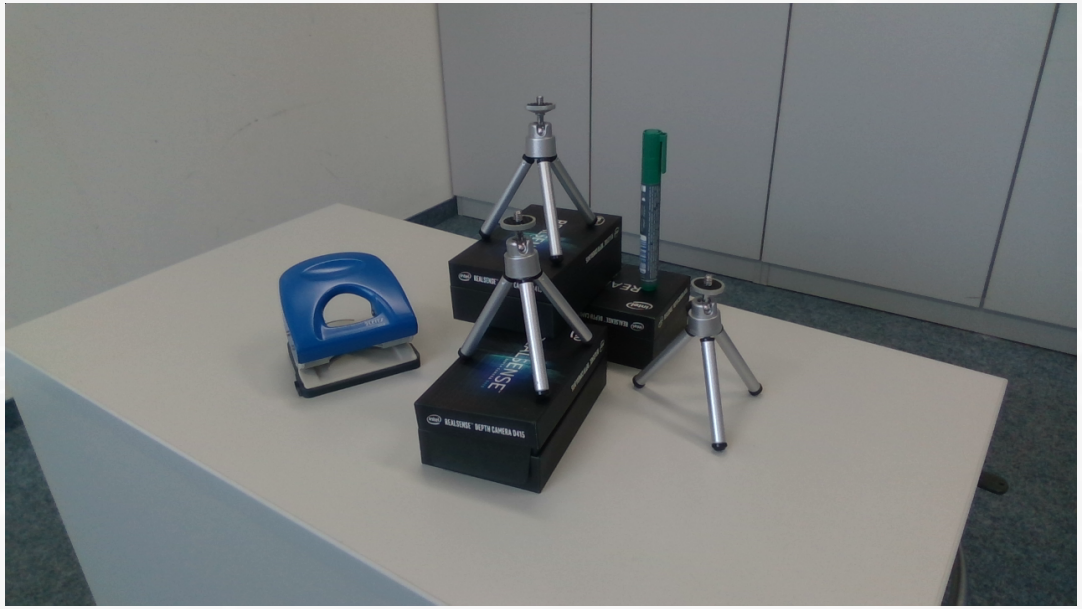
- 4x cameras are not working with single USB-Hub
- Multiple Repeater Cables not working
- Therefore, 3x cameras only

Pipeline Overview

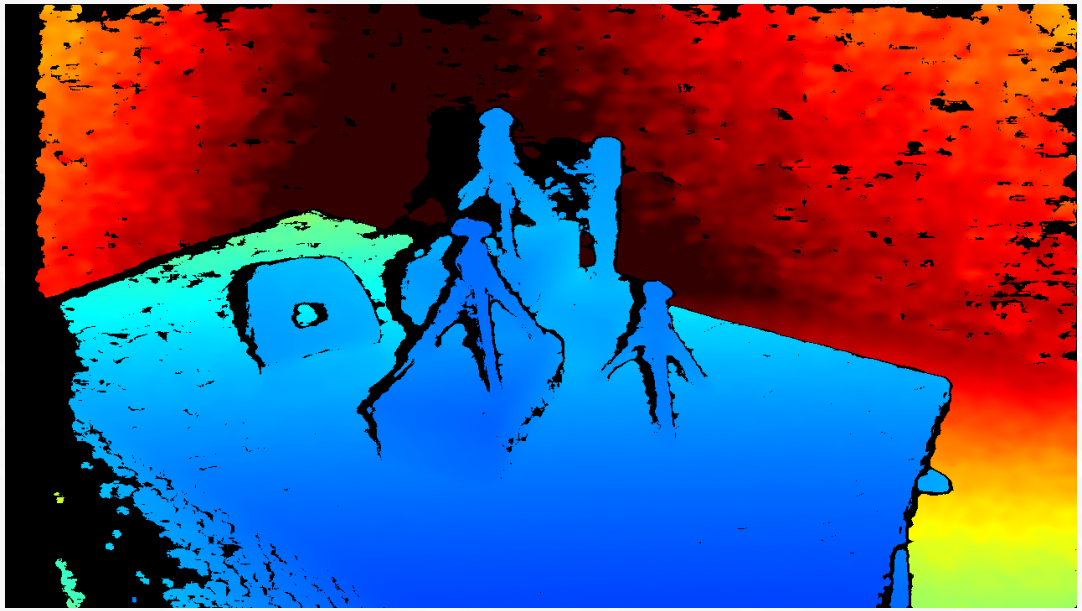
1. RGB and depth frame capture
2. Post-Processing
3. Align RGB and depth frame
4. ChArUco Board detection
5. Callibration via Procrustes, Bundle Adjustment and ICP
6. Fusion into Voxelgrid
7. Compute Marching Cubes

1. RGB and depth frame capture

- 1920x1080 RGB resolution

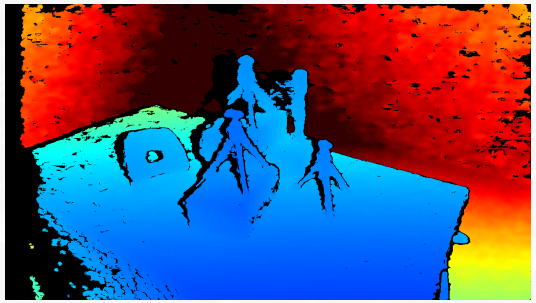


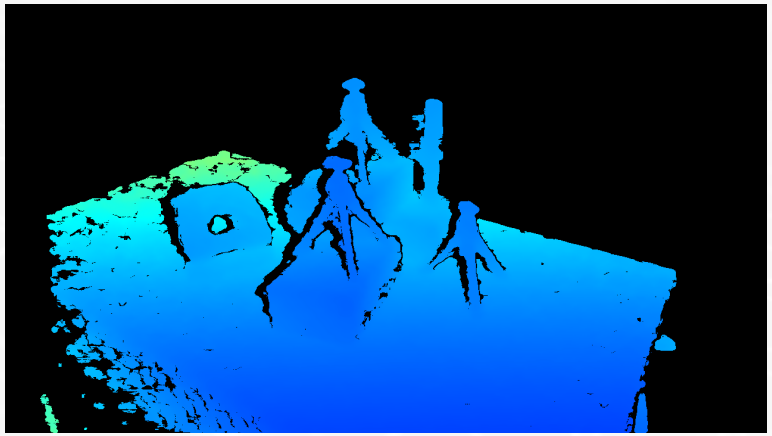
- 1280x720 active stereo depth



2. Post-Processing

2.1 Raw Input ⟶ 2.2 Threshold Filter

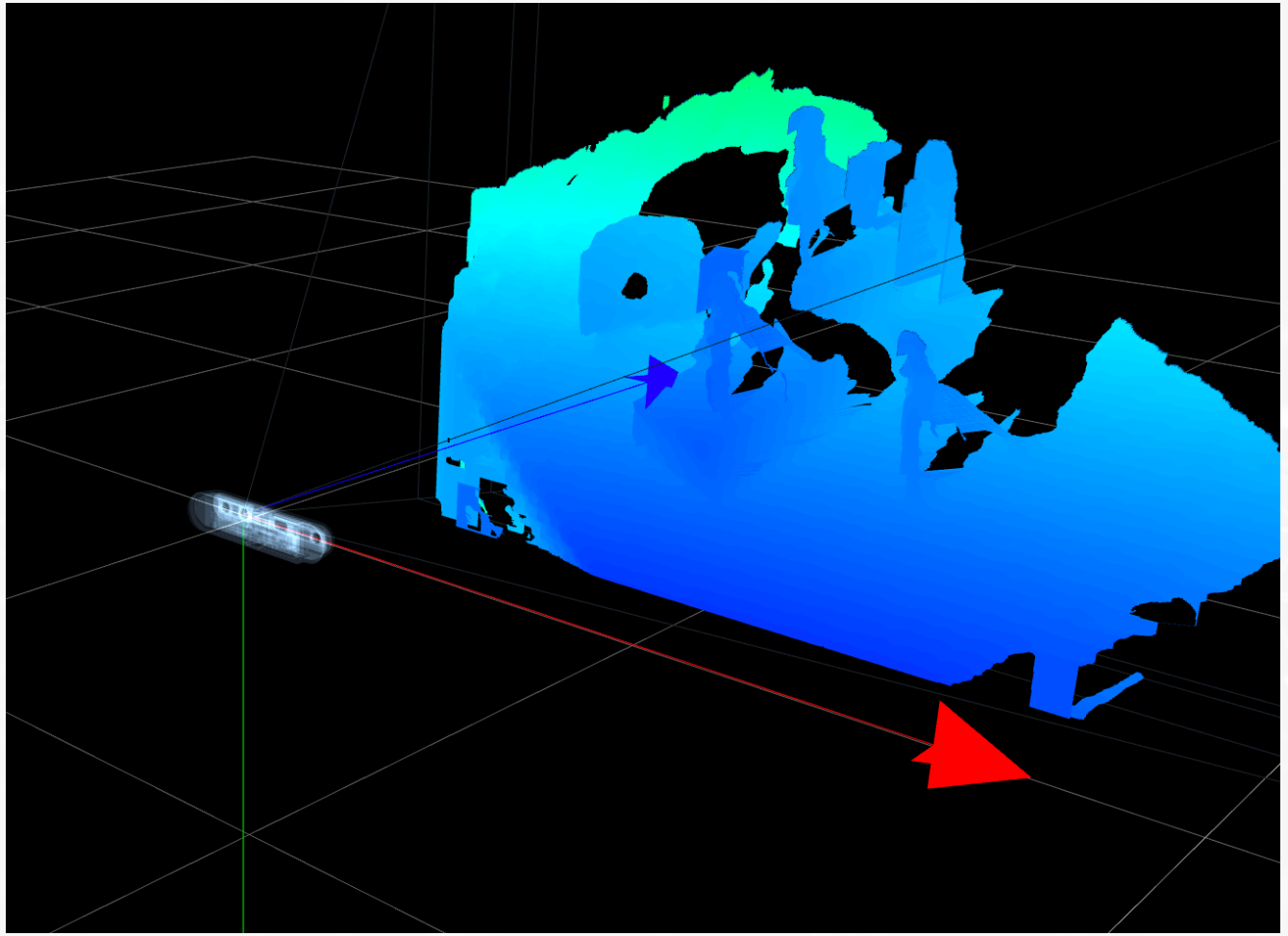




Camera

librealsense

3. Align RGB and depth frame



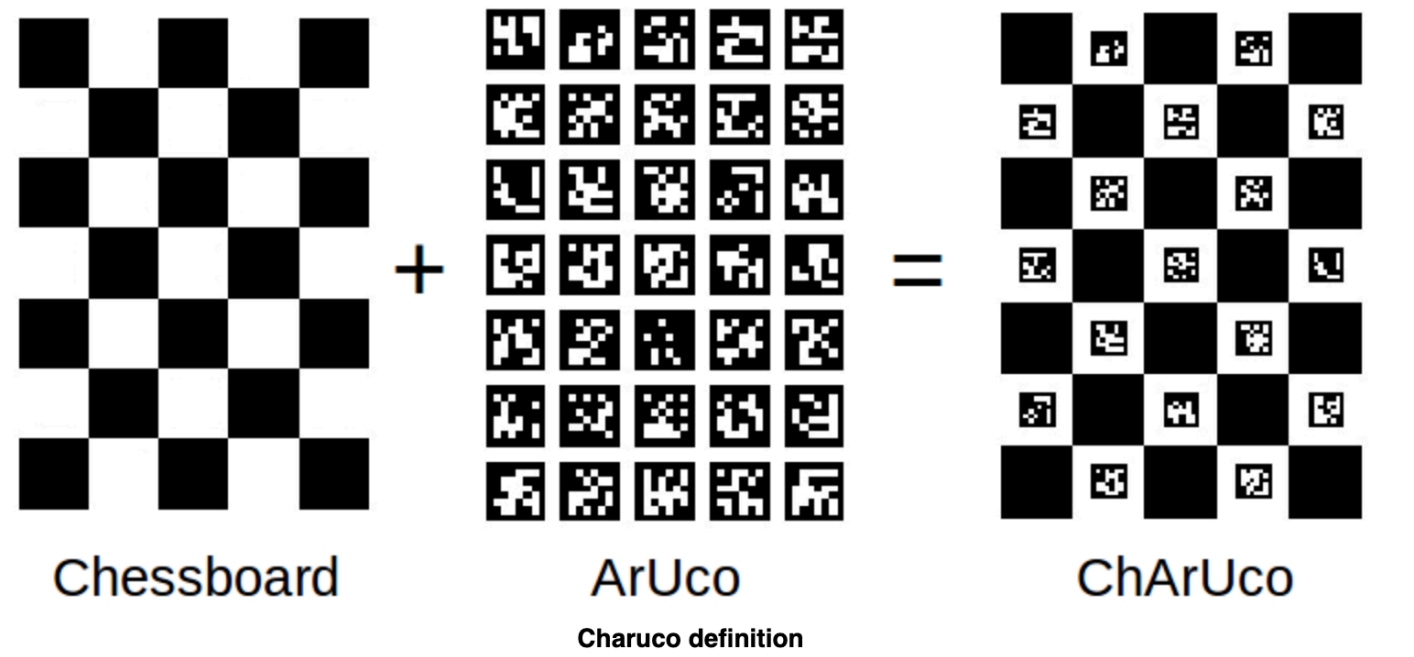
librealsense

4. ArUco markers

- Encodes messages (in our case: 1, 2, 3, 4, …)
- Can be easily detected
- Position of edges are known

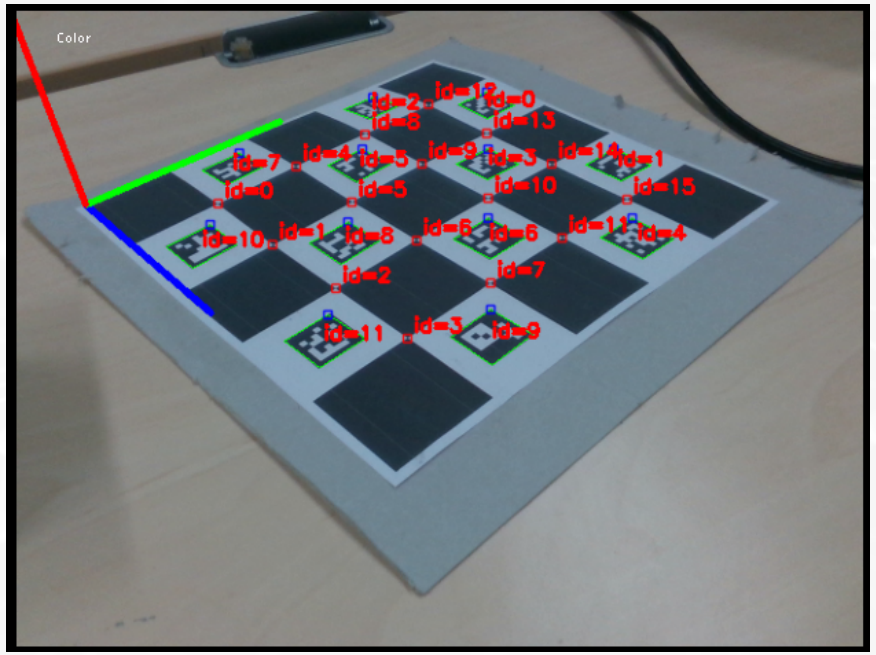
4. ChArUco

- Combination of multiple ArUco markeres on a chessboard = ChArUco

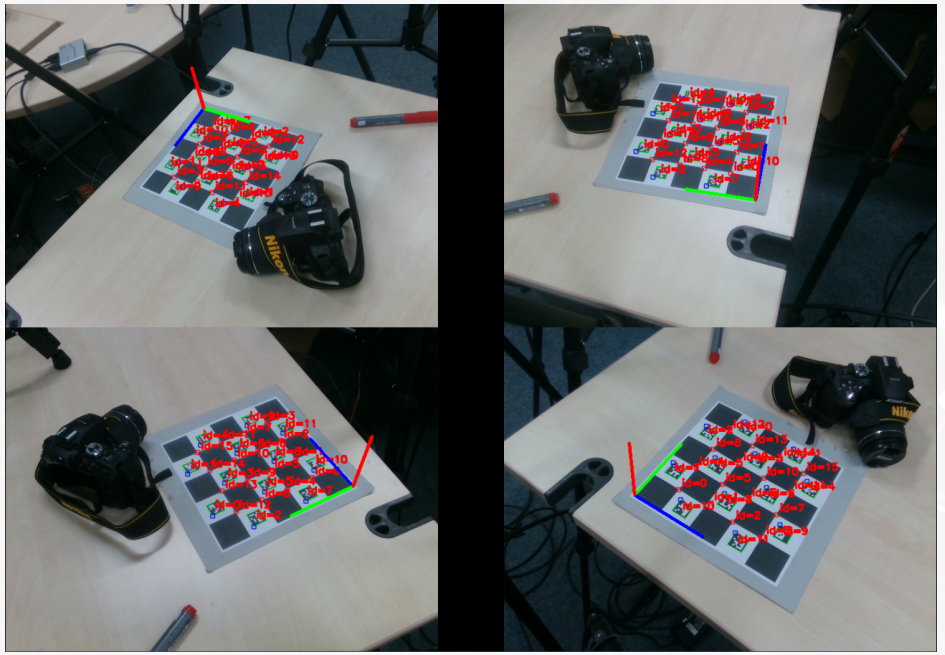


4. ChArUco board detection

- Detect 16 ArUco markers on one board



Single board



All camera streams

4. ChArUco board detection

- Detect 16 ArUco markers on one board
- Low resolutions only
- Noisy depth maps did not work too well during calibration

4. ArUco cube detection

- Ultimately, big cube with single ArUco marker on each side

[big cube picture here]

5. Calibration via Procrustes and Bundle Adjustment

- OpenCV:
  - `cv::aruco::detectMarkers`
  - `cv::aruco::estimatePoseCharucoBoard`
- Returns: Rotation and translation matrix
- Use detected ArUco corners as correspondences to refine rotation and translation matrices

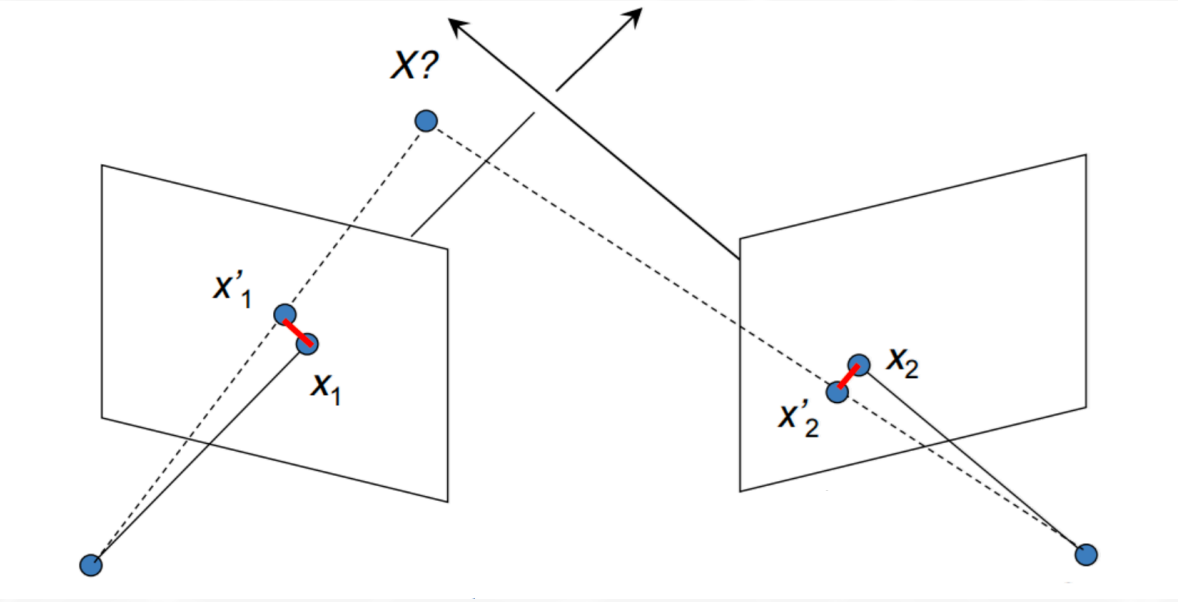
5. Calibration via Procrustes and Bundle Adjustment

- Use detected ArUco corners as correspondences to refine rotation and translation matrices

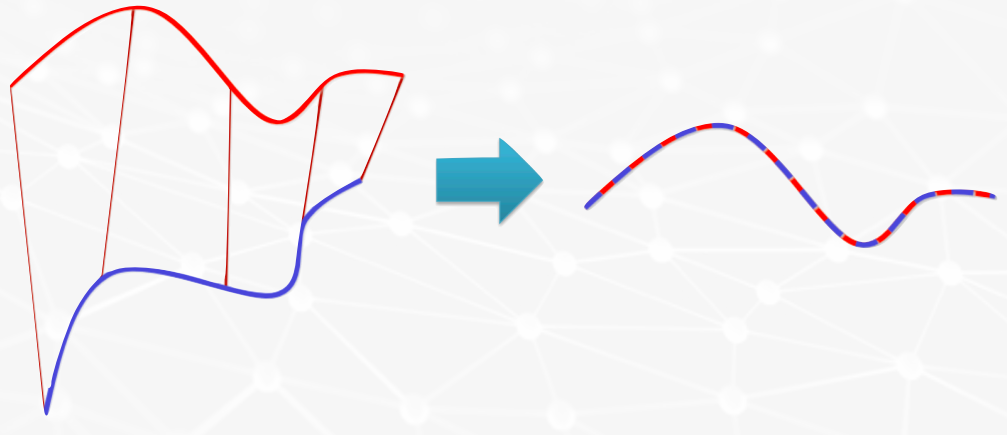
1. Initialize algorithm with Procrustes
2. Minimize re-projection error (Bundle adjustment)

5. Calibration via Procrustes and Bundle Adjustment

1. Initialize algorithm with Procrustes



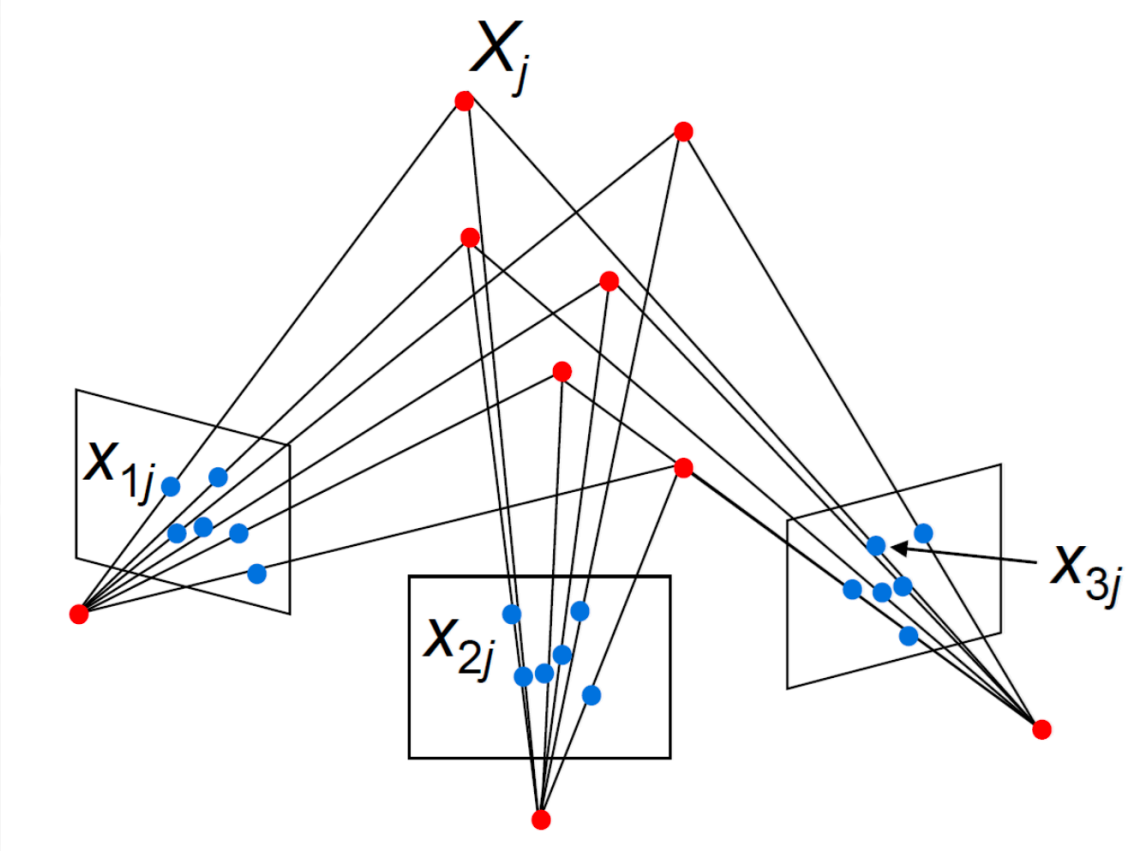
ArUco correspondences



Minimize rotation and translation

5. Calibration via Procrustes and Bundle Adjustment

2. Minimize re-projection error (bundle adjustment)

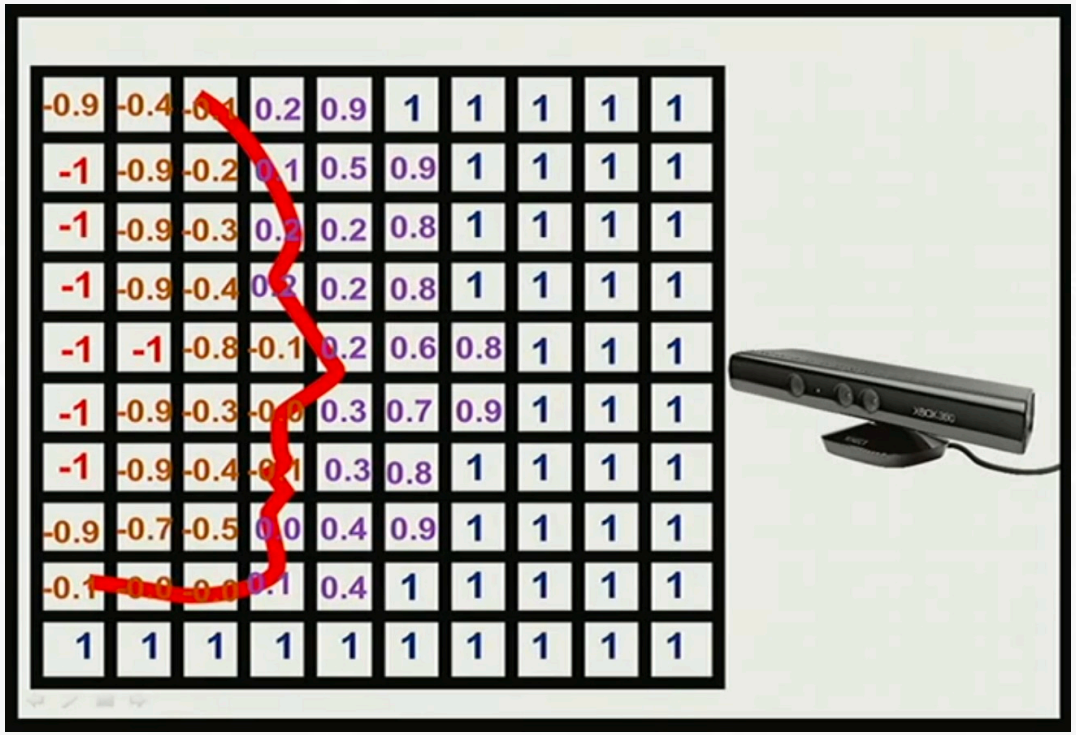


5. Calibration via Procrustes, Bundle Adjustment and ICP

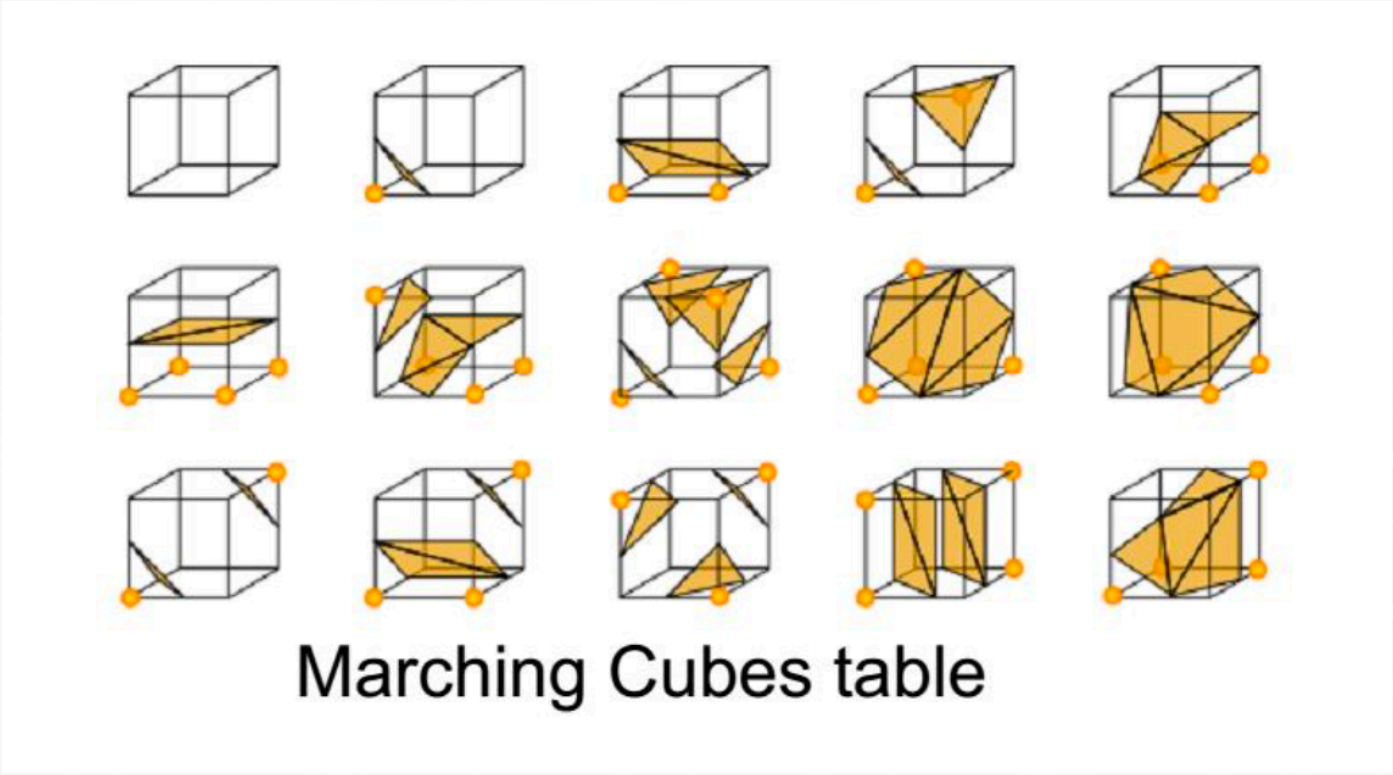
3. Run ICP for further refinement

6. Fusion into Voxelgrid

- Calculate TSDF values
- Fuse aligned points into Voxelgrid



7. Compute Marching Cubes



7. Compute Marching Cubes

Future work

- ICP to further improve Calibration
- Edge Enhancements and artifact reduction

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 and Eigen
- Custom implementation of Marching Cubes

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

Implementation Details

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

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

Live Demo

- Please feel free to join us in our office for a **Live Demonstration**!

Thank you! Questions?

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