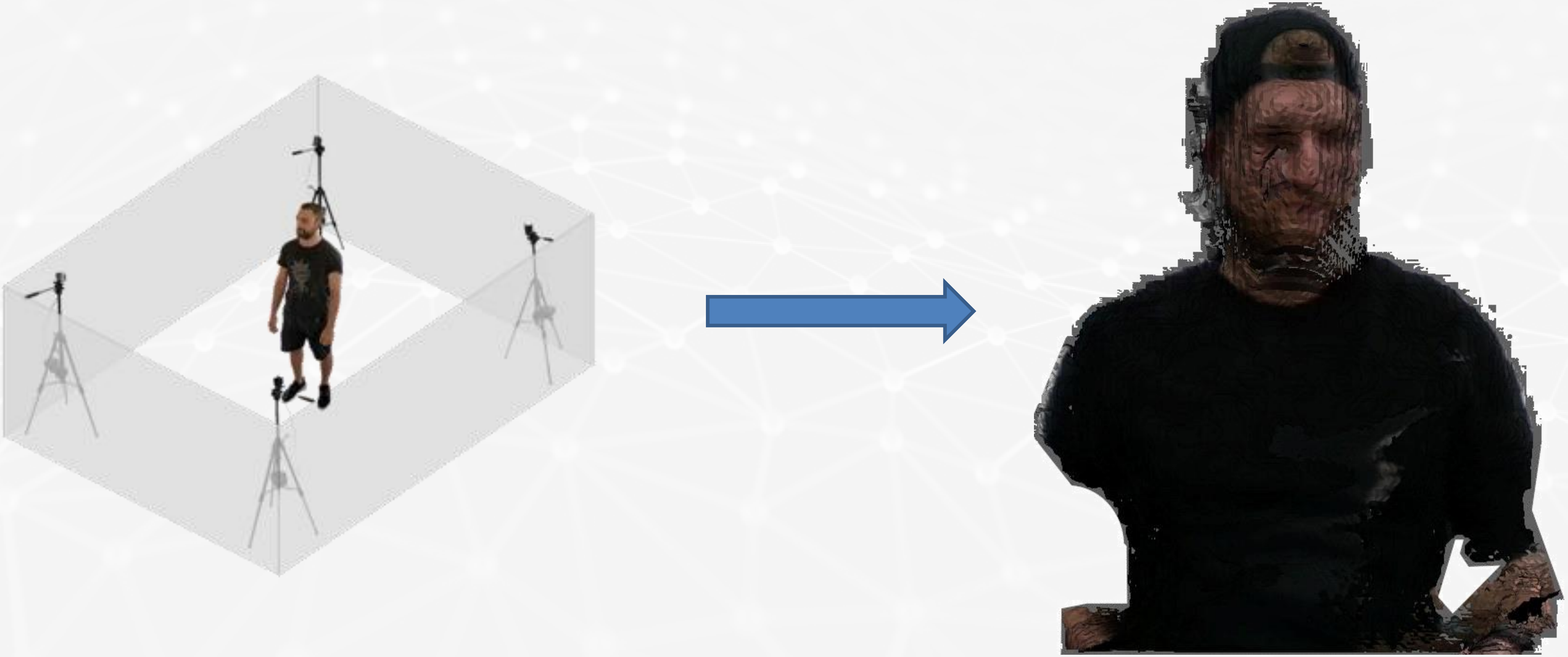
3D Scanning and Spatial Learning
Volumetric Capture

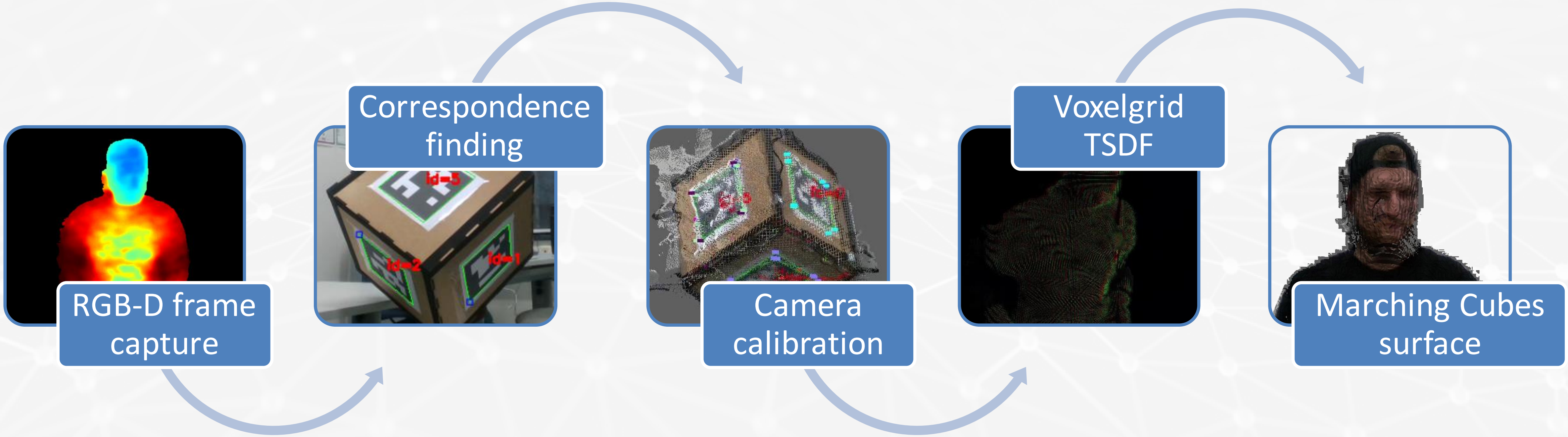
Marcel Bruckner, Kevin Bein, Moiz Sajid

Volumetric Capture

- Realtime 3D reconstruction using multiple cameras



Reconstruction Pipeline



RGB-D frame capture

Correspondence finding

Camera calibration

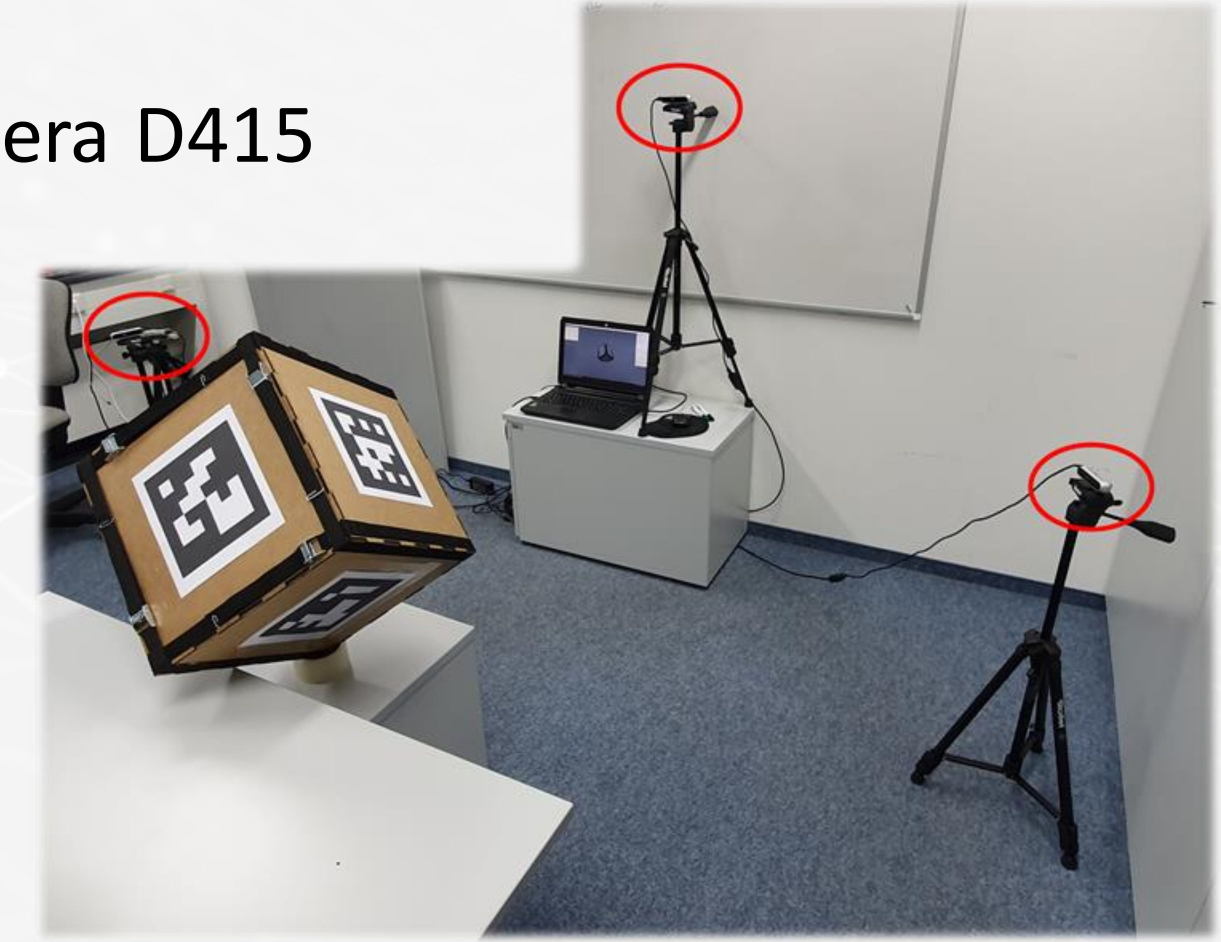
Voxelgrid TSDF

Marching Cubes surface

RGB-D Frame Capture

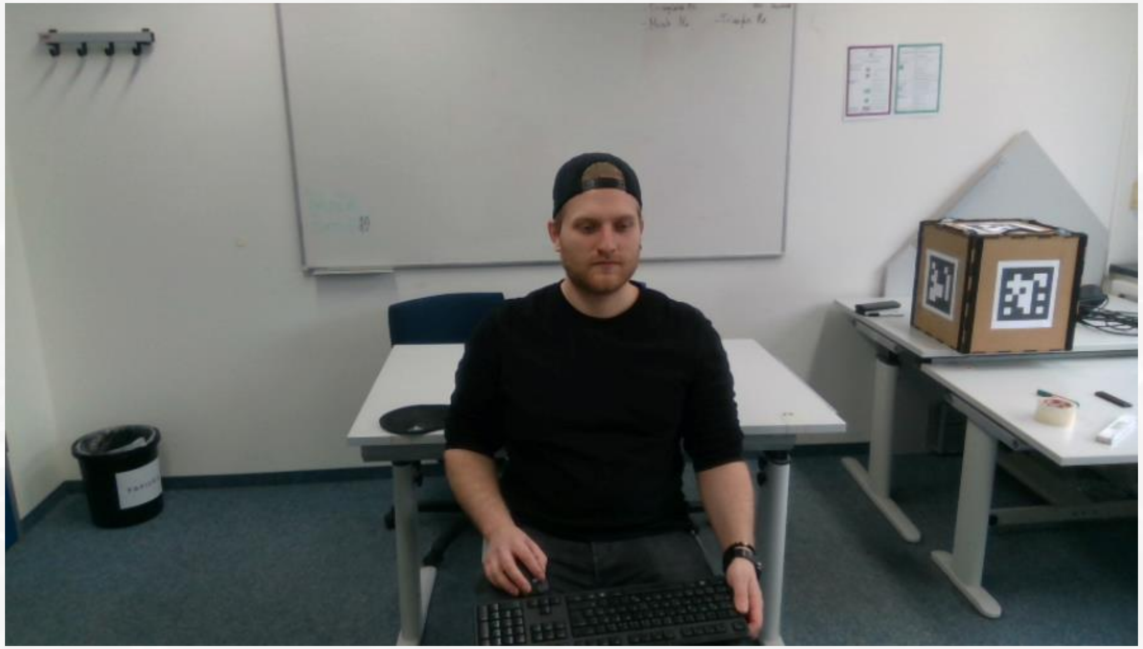
Hardware setup

- 3 Intel RealSense Depth Camera D415

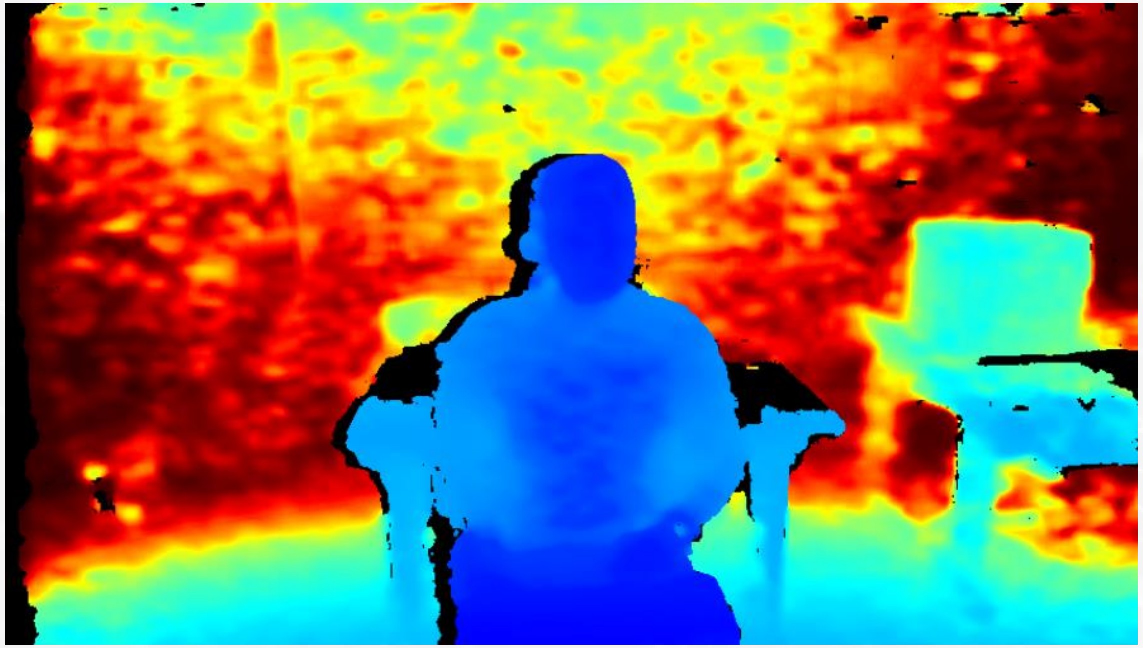


RGB-D Frame Capture

Frame aquisition



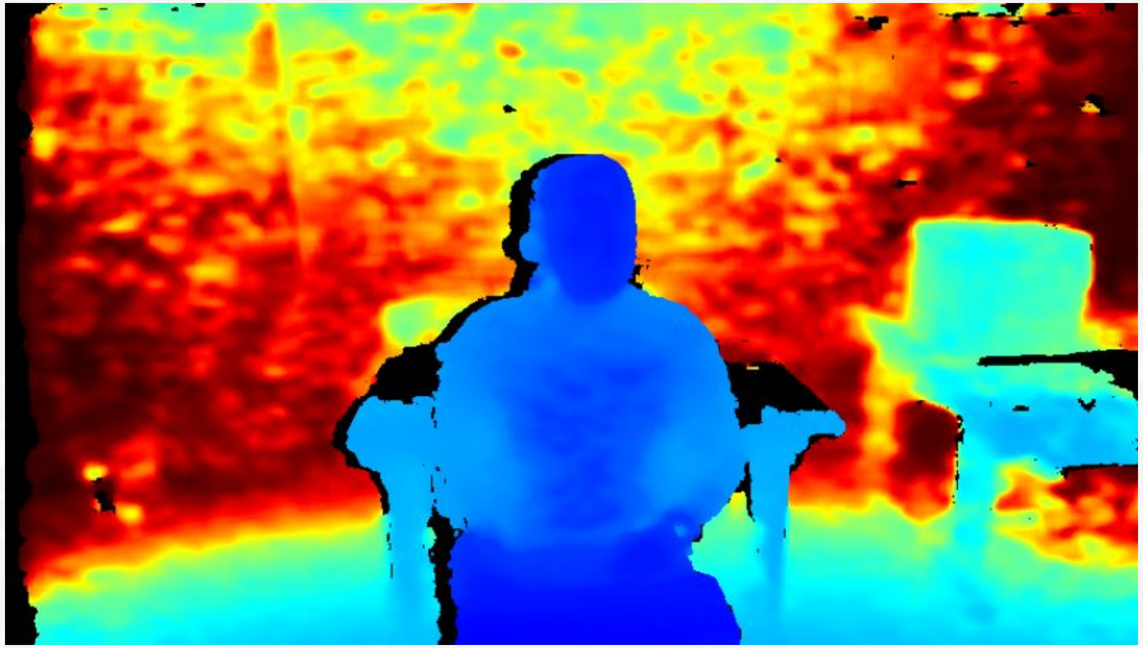
1920x1080 RGB resolution



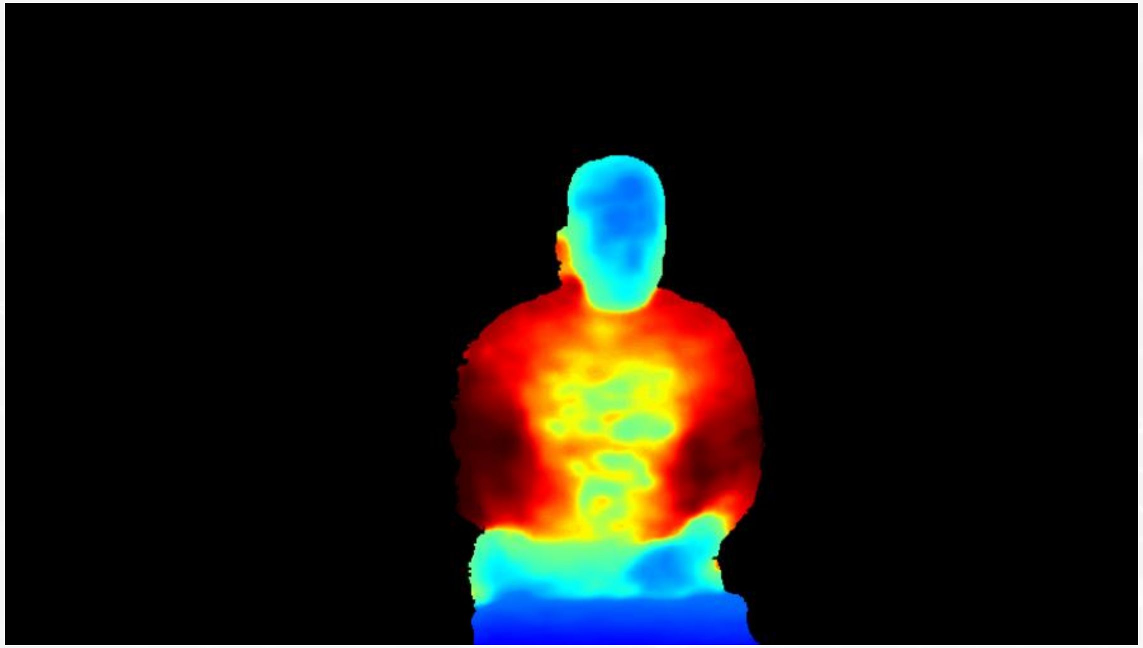
1280x720 active stereo depth

RGB-D Frame Capture

Depthmap preprocessing

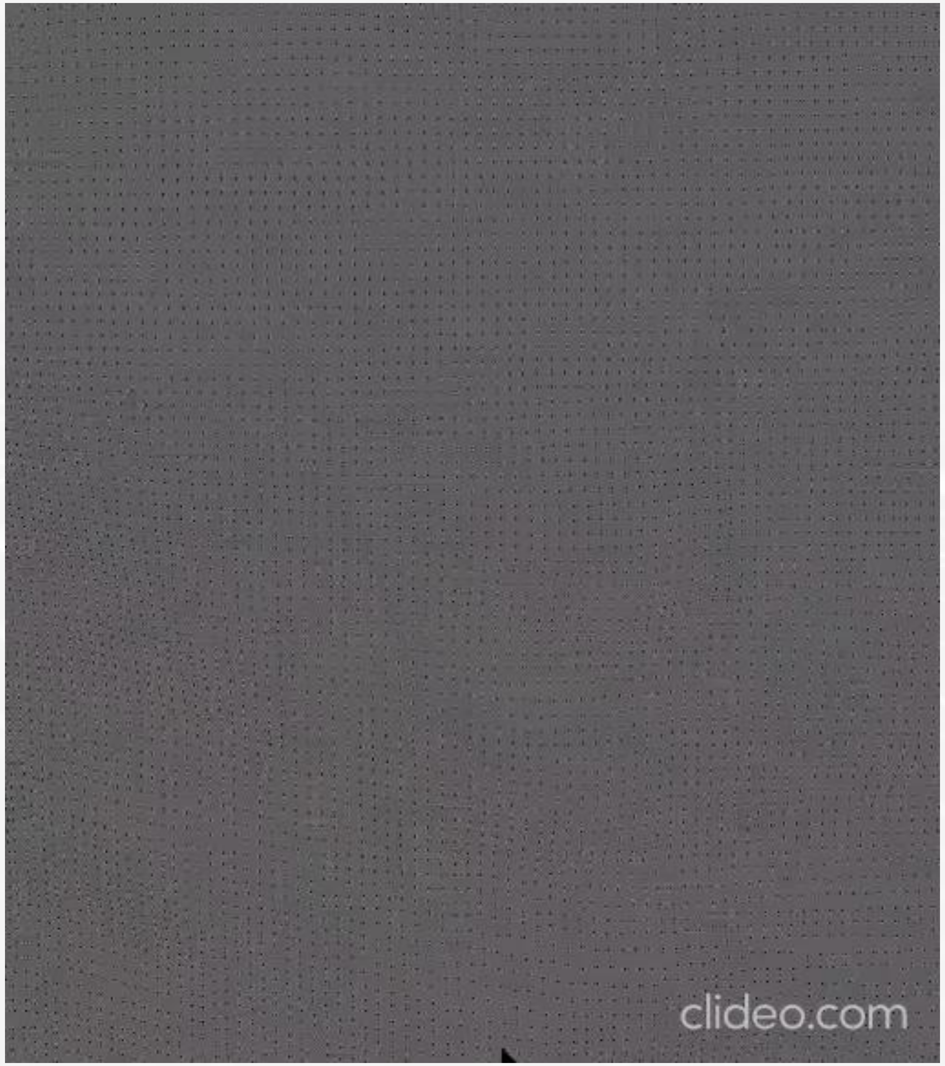


Original



Threshold filtered

RGB-D Frame Capture



RGB-D Frame Capture

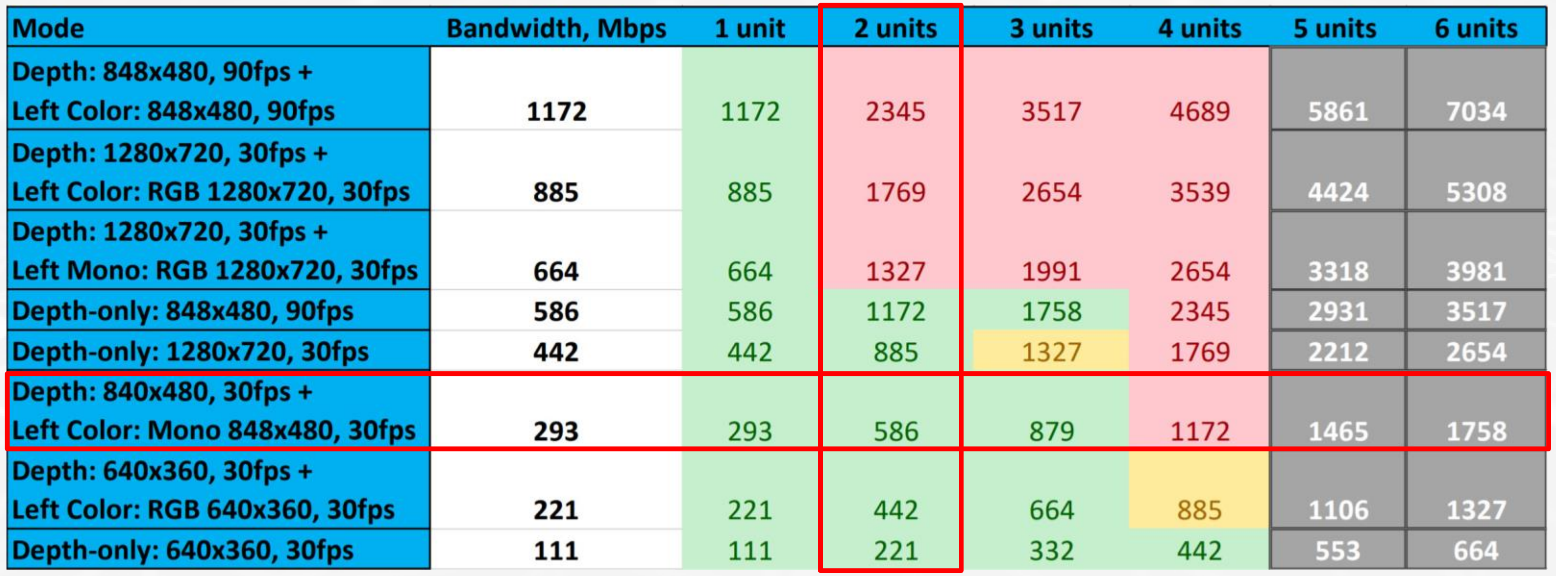
Tried approaches

- Holefilling filter
- Spatial filter
- Edge enhancement filter

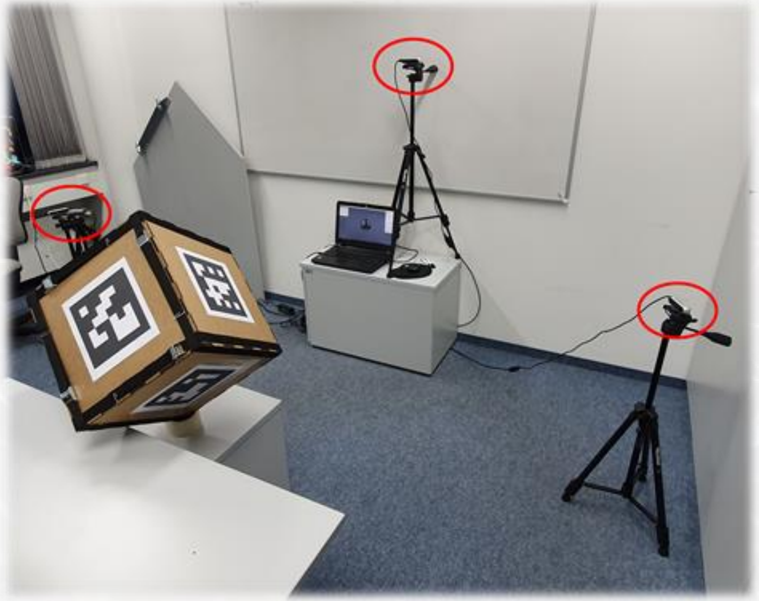
RGB-D Frame Capture

Difficulties

- Bandwidth limitations



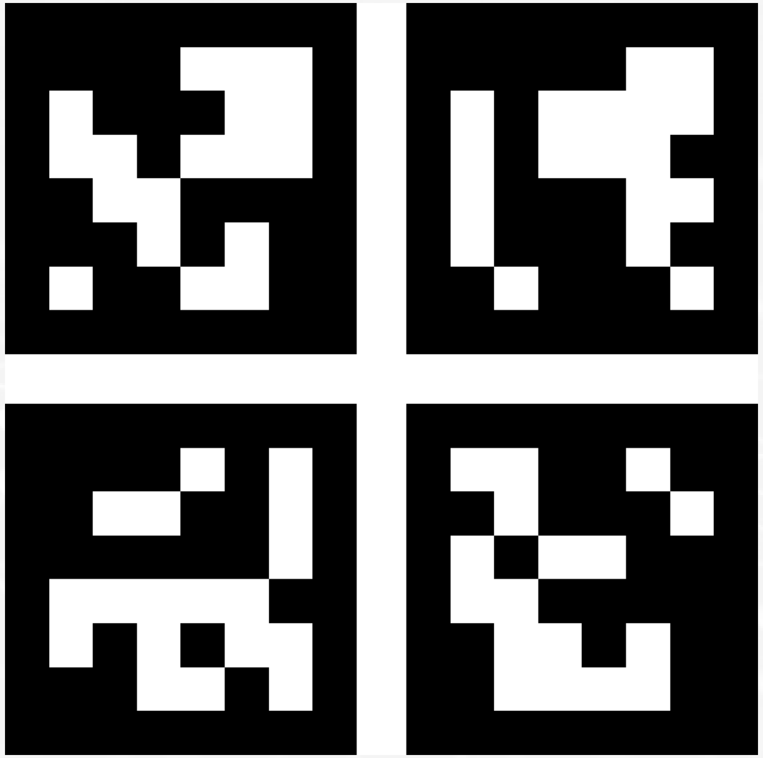
Mode	Bandwidth, Mbps	1 unit	2 units	3 units	4 units	5 units	6 units
Depth: 848x480, 90fps + Left Color: 848x480, 90fps	1172	1172	2345	3517	4689	5861	7034
Depth: 1280x720, 30fps + Left Color: RGB 1280x720, 30fps	885	885	1769	2654	3539	4424	5308
Depth: 1280x720, 30fps + Left Mono: RGB 1280x720, 30fps	664	664	1327	1991	2654	3318	3981
Depth-only: 848x480, 90fps	586	586	1172	1758	2345	2931	3517
Depth-only: 1280x720, 30fps	442	442	885	1327	1769	2212	2654
Depth: 840x480, 30fps + Left Color: Mono 848x480, 30fps	293	293	586	879	1172	1465	1758
Depth: 640x360, 30fps + Left Color: RGB 640x360, 30fps	221	221	442	664	885	1106	1327
Depth-only: 640x360, 30fps	111	111	221	332	442	553	664



Correspondence Finding

ArUco markers

- Square markers with unique ids
- *Easy detection* using OpenCV
- Subpixel perfect corners in color stream



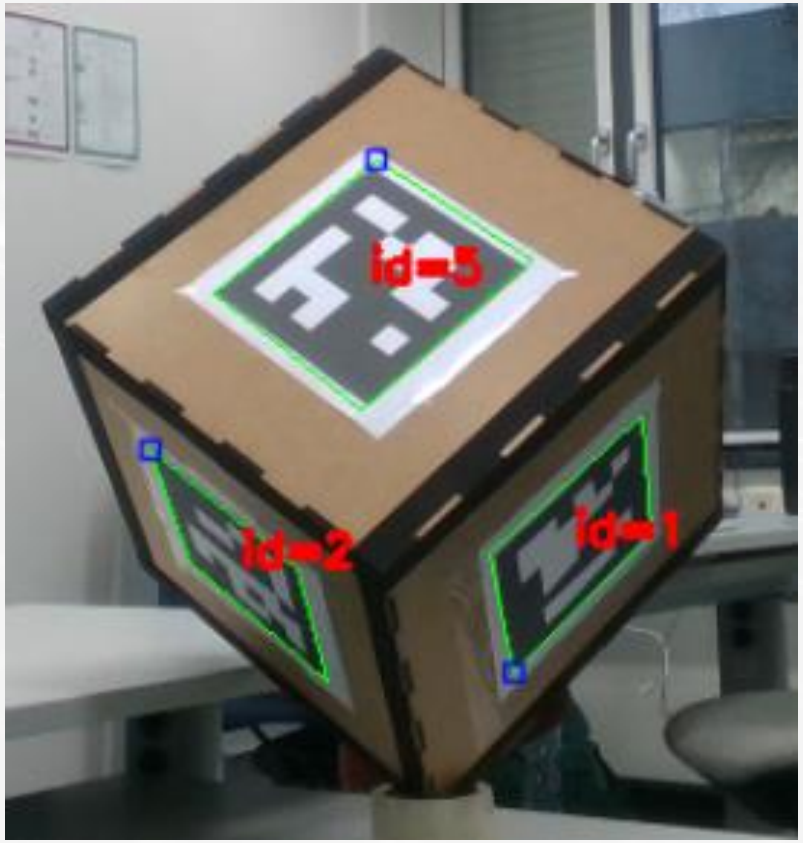
Correspondence Finding

Marker cube

- 6 sides, one marker per side
- Easily detectable even from steep angles
- Overlap of detected markers used for pose estimation

Correspondence Finding

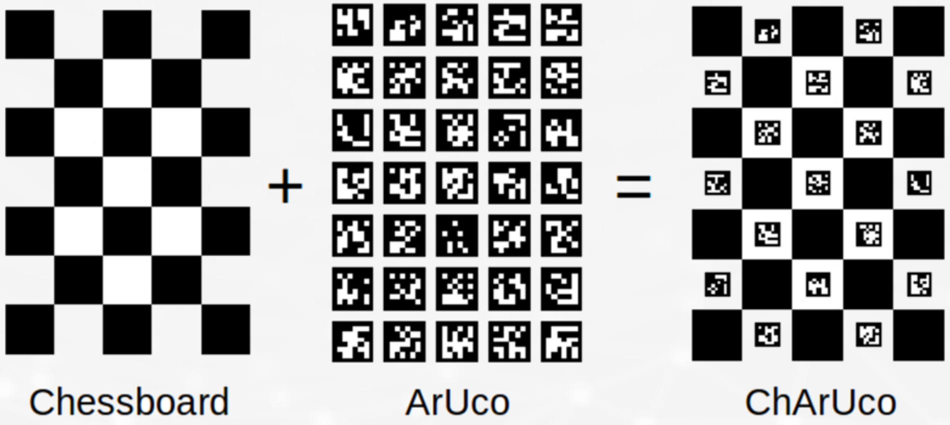
Marker cube



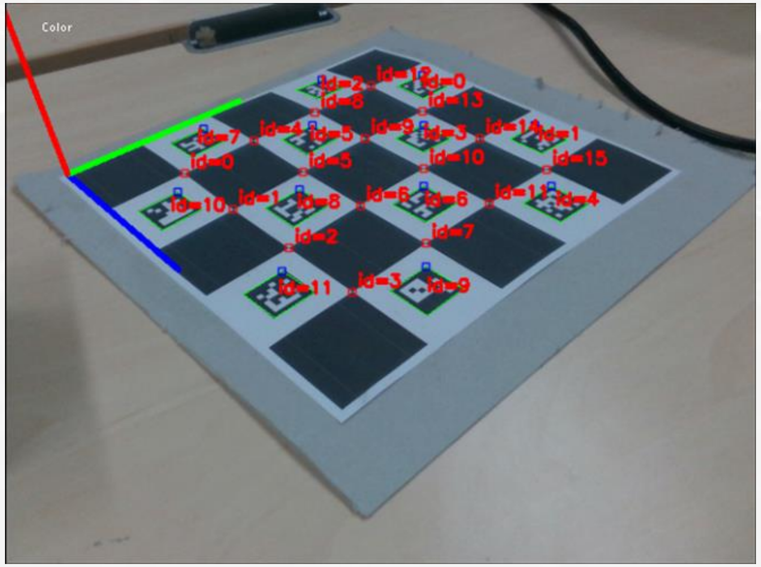
Correspondence Finding

Tried approaches

- ChArUco marker board
- Simultaneous marker detection and pose estimation
- Tradeoff between number of markers and marker size
- *Not robust enough*



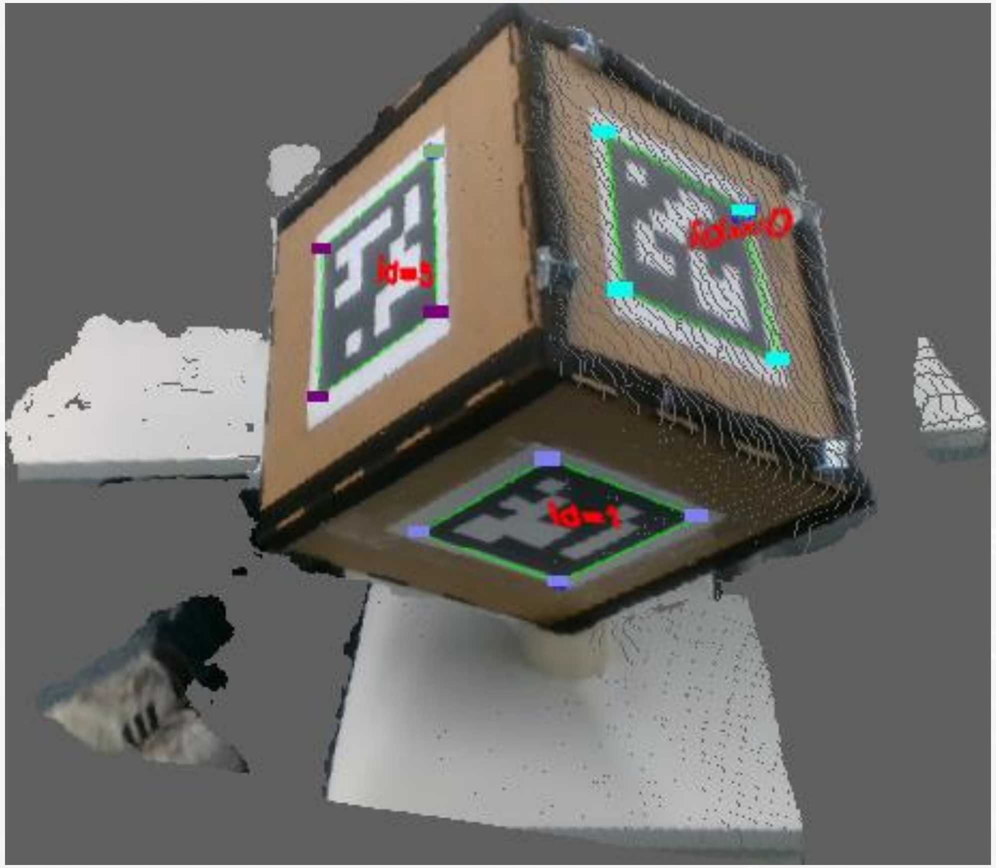
Charuco definition



3D Pose Estimation

3D marker positions

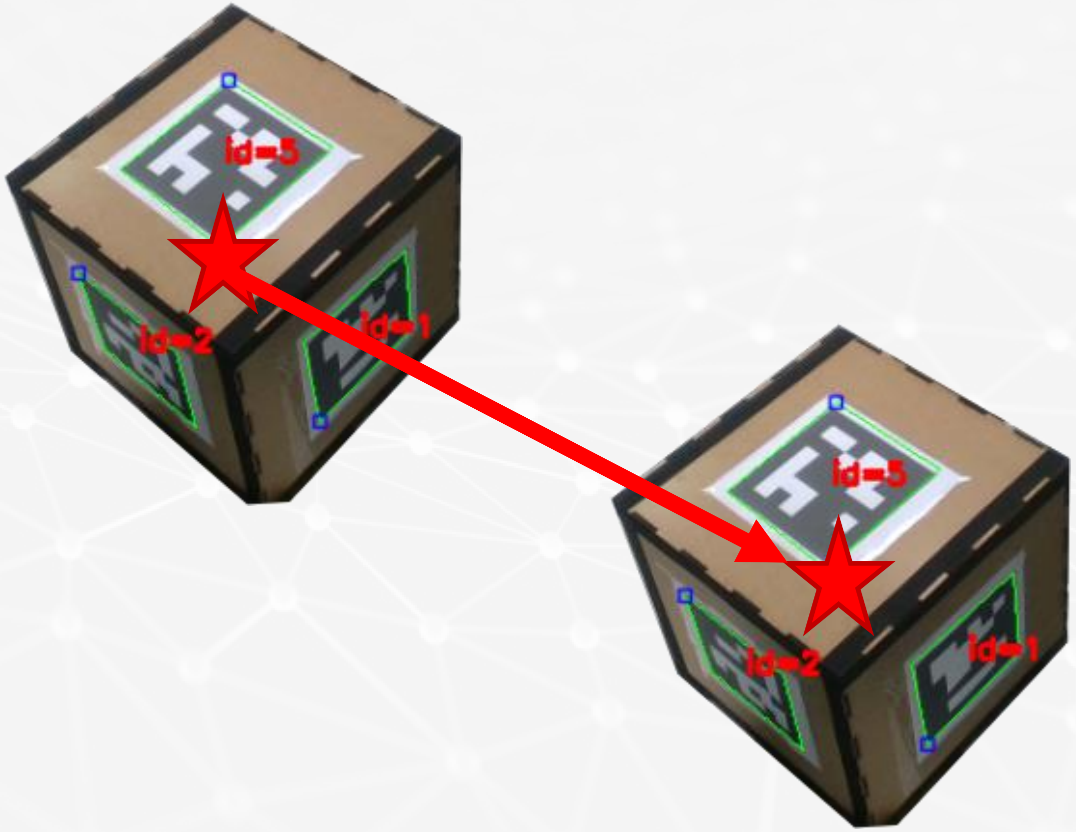
- Backprojection of ArUco markers into 3D pointcloud
- Subpixel perfect pixel location allows robust backprojection
- *Align 3D marker positions of all cameras*



3D Pose Estimation

Procrustes

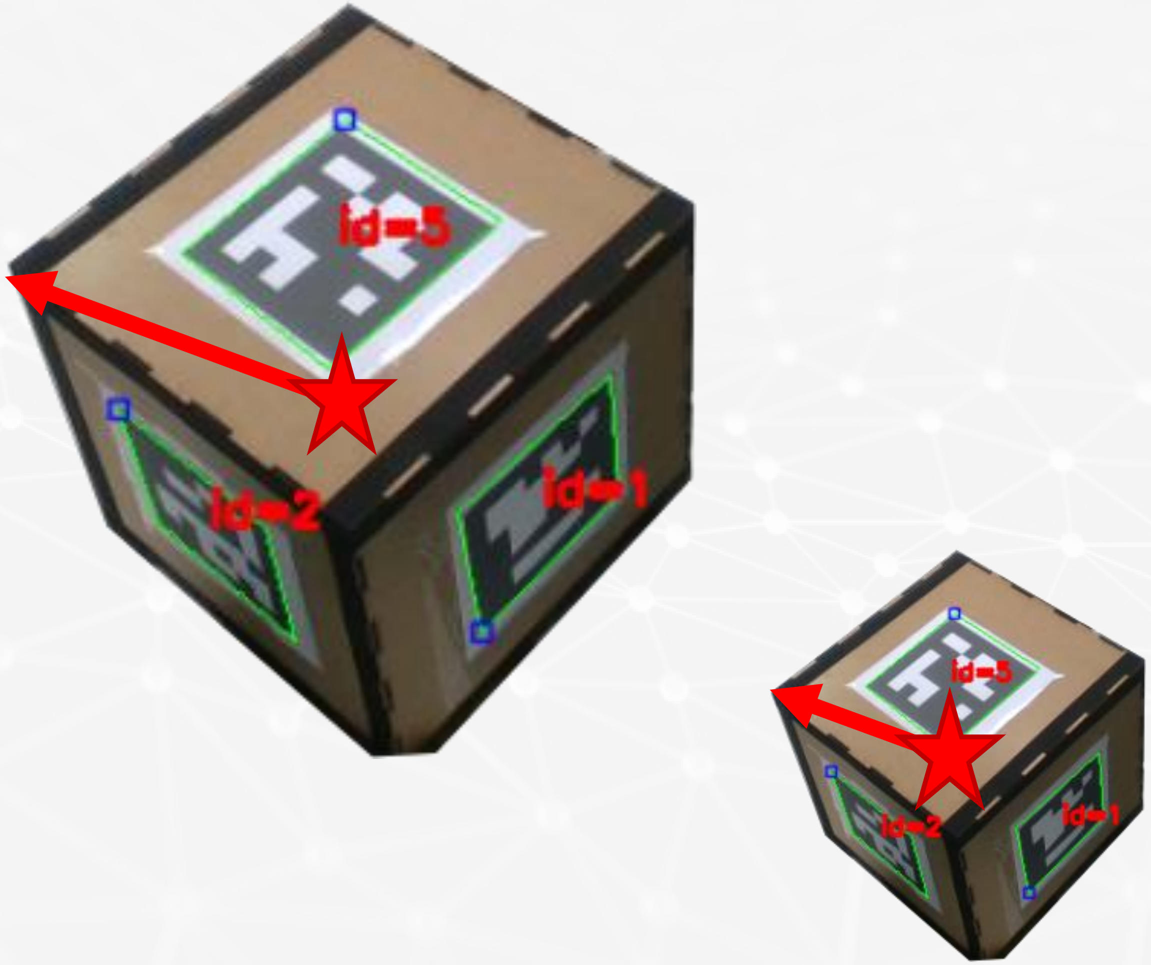
- Align two objects using known correspondences
- *Translation:* Vector between centers of gravity



3D Pose Estimation

Procrustes

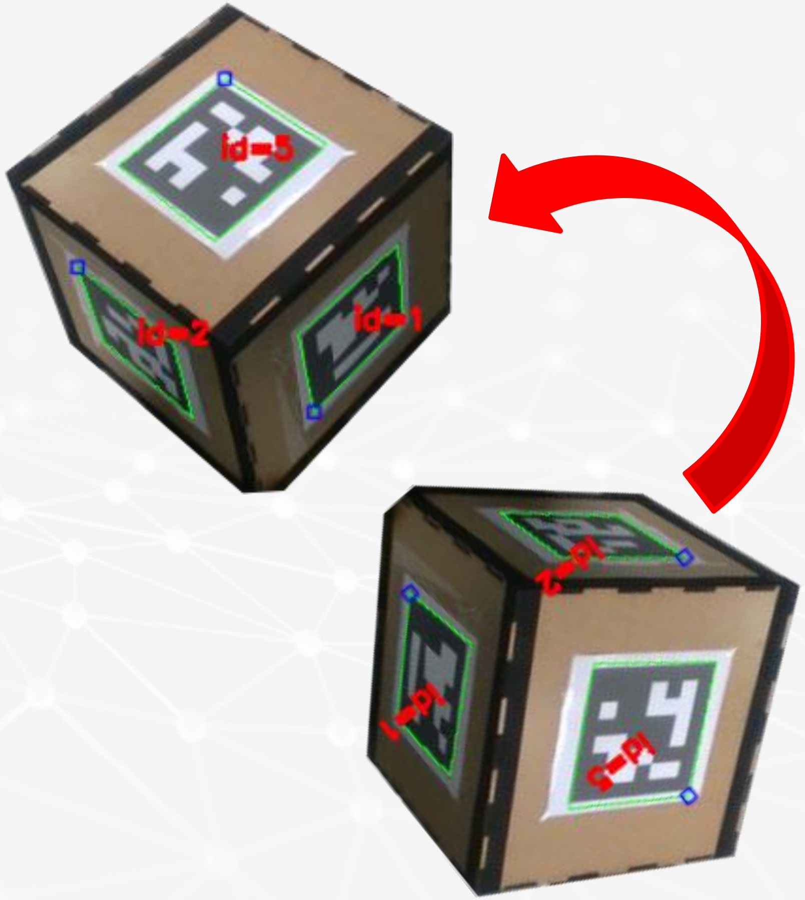
- Align two objects using known correspondences
- *Scale:* Avgerage distance to center of gravity



3D Pose Estimation

Procrustes

- Align two objects using known correspondences
- *Rotation:*
  - Minimize $\sum_i \|x_i - R * \hat{x}_i\| = \left\|X - \hat{X}R^T\right\|$
  - *Compute SVD of* $X^T X = USV^T$
  - *Final rotation:* $R = UV^T$



3D Pose Estimation

Difficulties

- Procrustes is not robust enough
- Mean squared error after alignment still fairly high

3D Pose Estimation

Point to Point Error

- Use procrustes as an initialization
- Optimize:

$$\sum_{i}\sum_{j}\sum_{k}\left\|X_{ik} - T_j R_j S_j * X_{jk}\right\|$$

Frames Relative frames Correspondences

3D Pose Estimation

Camera Calibration

Difficulties

- Visualization of calibration results
- Ceres for Point to Point optimization

Further work

- Nearest Neighbor Search for correspondence matching
- ICP for further alignment

Voxelgrid

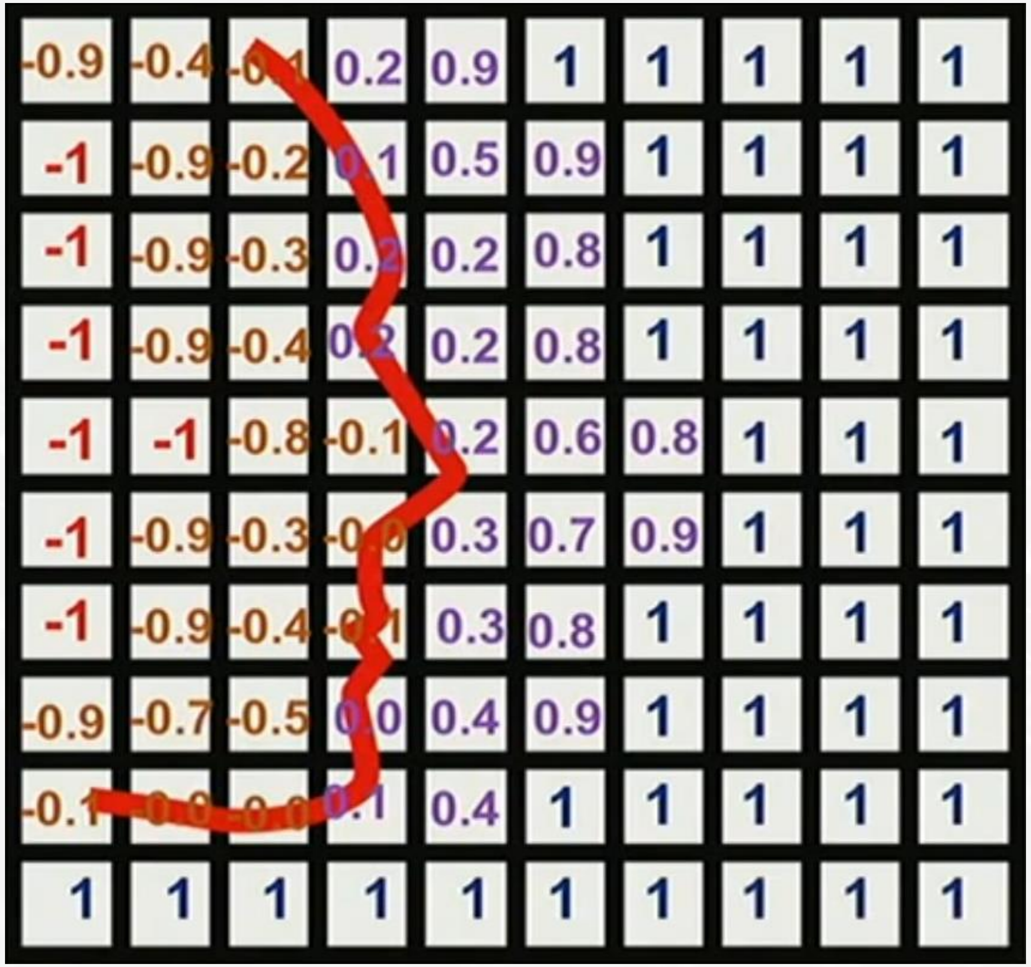
- Fuse aligned frames into voxelgrid
- Calculate TSDF of the integrated frames:

$$tsdf = z_{voxel} - z_{depthmap}$$

- Weighted averaging of the TSDF values:

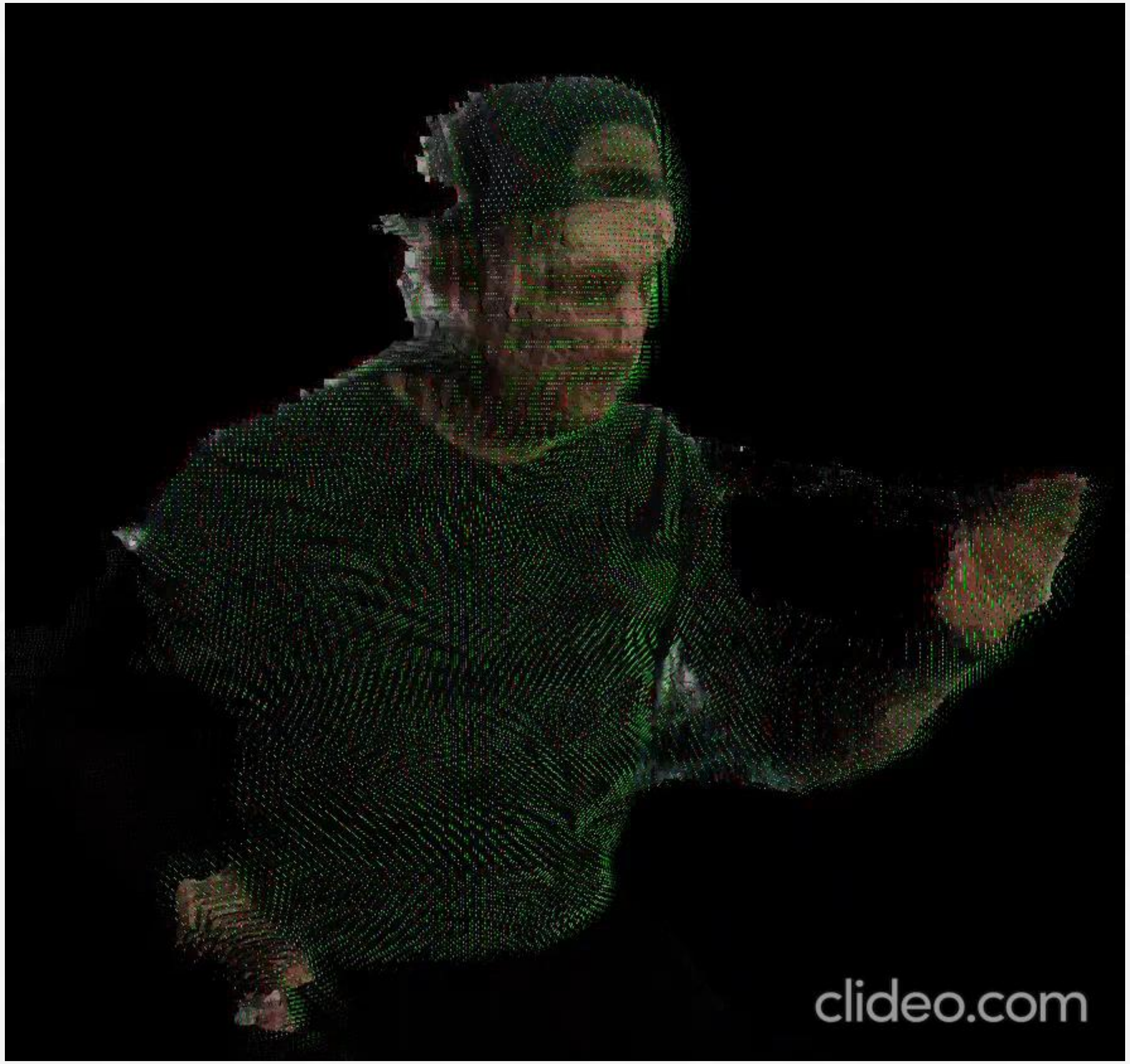
$$tsdf_{i+1} = \frac{tsdf_i * weight}{weight + 1}$$

→ *Implicit surface representation*



-0.9	-0.4	-0.1	0.2	0.9	1	1	1	1	1
-1	-0.9	-0.2	0.1	0.5	0.9	1	1	1	1
-1	-0.9	-0.3	0.2	0.2	0.8	1	1	1	1
-1	-0.9	-0.4	0.2	0.2	0.8	1	1	1	1
-1	-1	-0.8	-0.1	0.2	0.6	0.8	1	1	1
-1	-0.9	-0.3	-0.0	0.3	0.7	0.9	1	1	1
-1	-0.9	-0.4	-0.1	0.3	0.8	1	1	1	1
-0.9	-0.7	-0.5	0.0	0.4	0.9	1	1	1	1
-0.1	-0.0	-0.0	0.1	0.4	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1

Voxelgrid



Voxelgrid

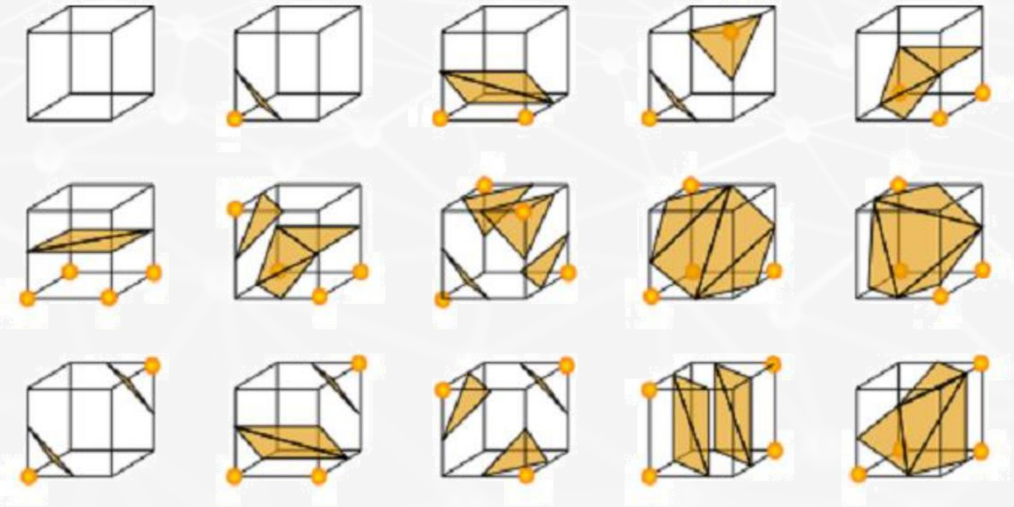
Difficulties

- *Parallelize* all calculations on GPU
- OpenGL compute shader programming
- Optimizing for *realtime*

- Tradeoff between *resolution* and *frame rate*
- Find good *truncation distance* to lower artifacts

Marching Cubes

- Converts an implicit surface to a polygonal mesh
- Voxelgrid representing the implicit surface
- Determine zero crossings for every grid cell

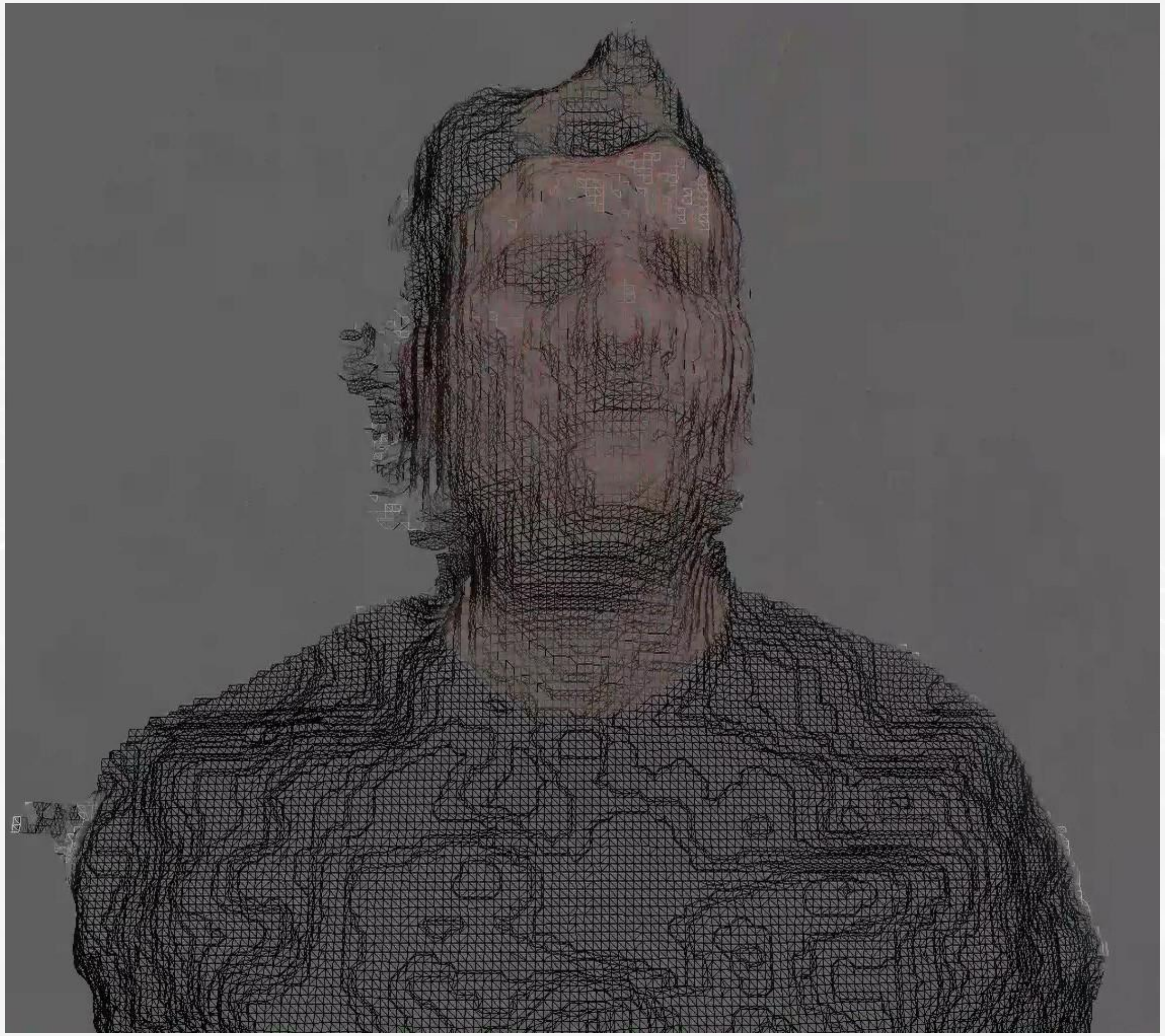


Marching Cubes table

Marching Cubes

Result

- Triangulated mesh representing the implicit surface



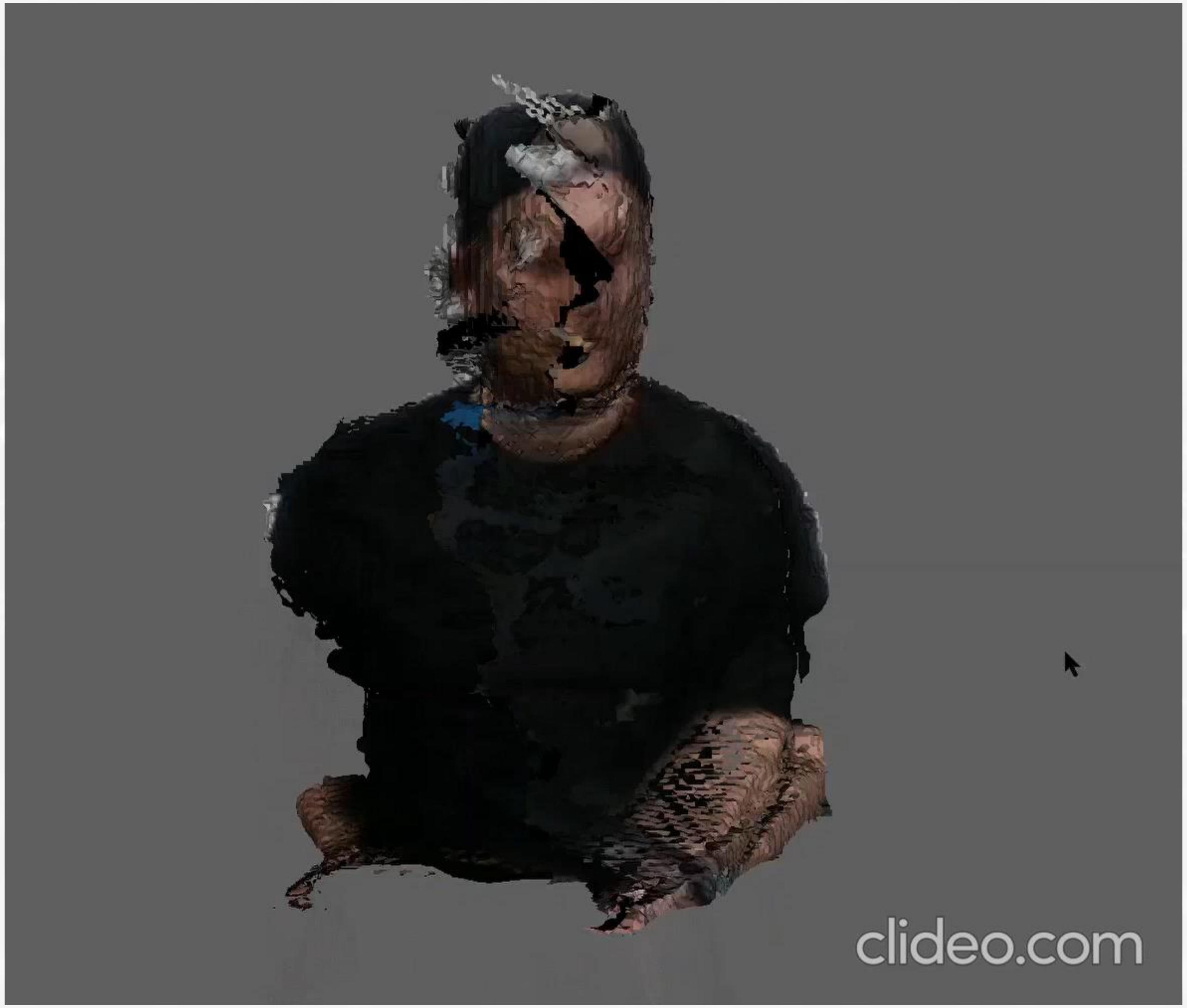
Marching Cubes

Difficulties

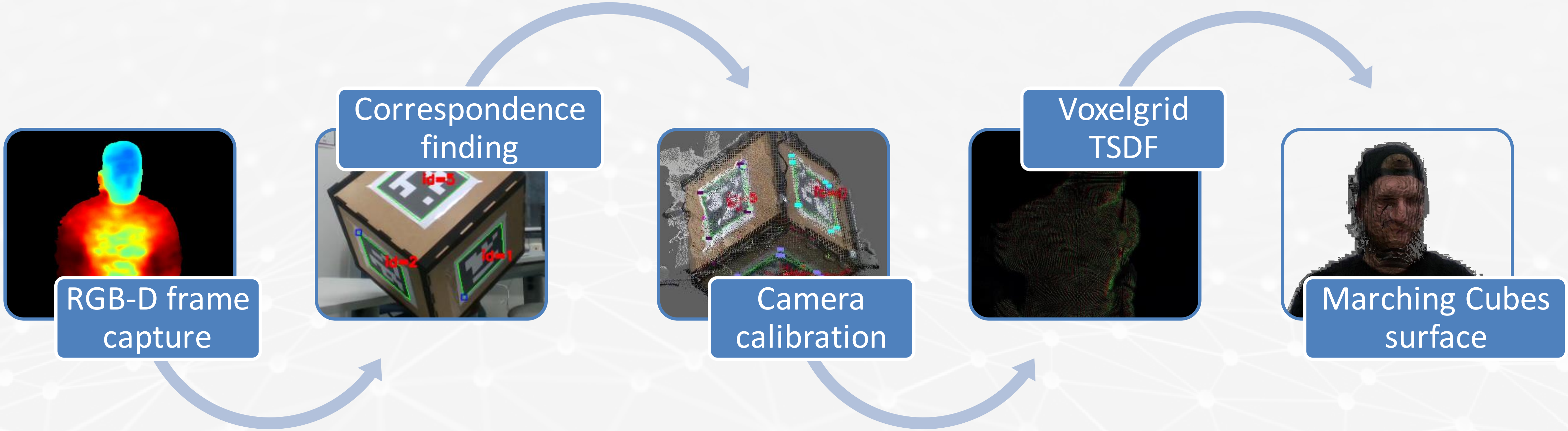
- *Parallelize* all calculations on GPU
- OpenGL compute shader programming
- Optimizing for *realtime*

- Voxels outside of camera view
- Two pass compute shader
  1. Count triangles
  2. Generate triangles

Final results



Volumetric Capture



RGB-D frame capture

Correspondence finding

Camera calibration

Voxelgrid TSDF

Marching Cubes surface

If you are interested in a live reconstruction of yourself, we invite you to come to our office in 02.07.39!