TSDF Volume Reconstruction

Martin Herrmann
Simon Trendel
Neeraj Sujan

Technische Universität München
Department of Informatics
Computer Vision Group

October 5, 2015

Outline

1 Introduction

2 Approach

3 Results

Outline

1 Introduction

2 Approach

3 Results

Introduction

- Reconstruct 3D voxel grid from multiple input frames
- Frames consist of color (RGB) and depth images
- Must be fast enough to use in real-time with a Kinect (or a similar sensor)

Introduction

- Reconstruct 3D voxel grid from multiple input frames
- Frames consist of color (RGB) and depth images
- Must be fast enough to use in real-time with a Kinect (or a similar sensor)

Introduction

- Reconstruct 3D voxel grid from multiple input frames
- Frames consist of color (RGB) and depth images
- Must be fast enough to use in real-time with a Kinect (or a similar sensor)

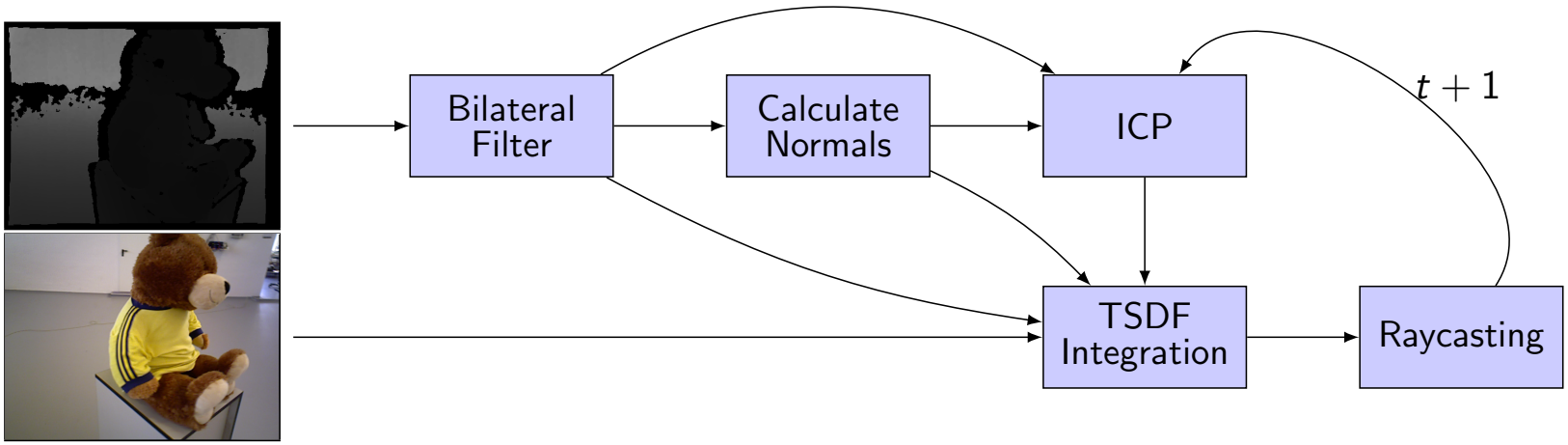
Outline

1 Introduction

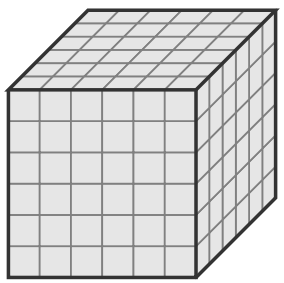
2 Approach

3 Results

The Pipeline

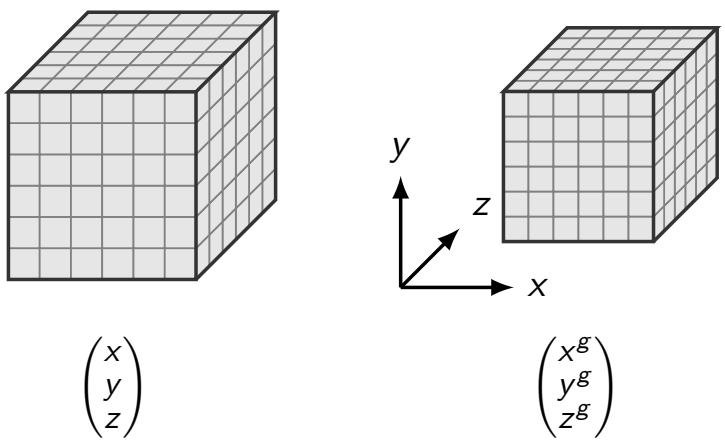


Transformations between the different coordinate systems



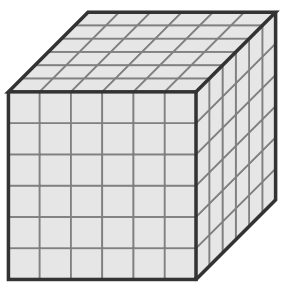
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Transformations between the different coordinate systems

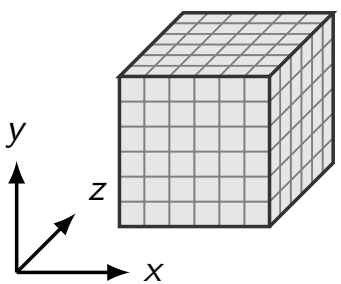


$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} \qquad \begin{pmatrix} x^g \\ y^g \\ z^g \end{pmatrix}$$

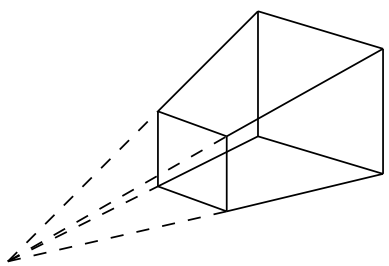
Transformations between the different coordinate systems



$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

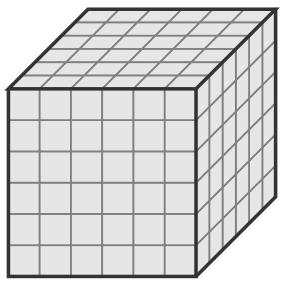


$$\begin{pmatrix} x^g \\ y^g \\ z^g \end{pmatrix}$$

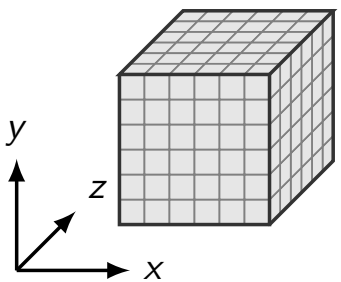


$$\begin{pmatrix} x^c \\ y^c \\ z^c \end{pmatrix}$$

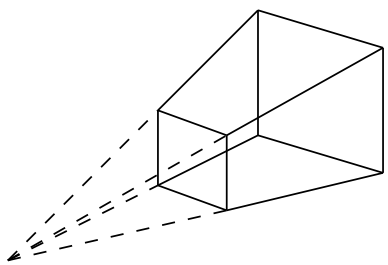
Transformations between the different coordinate systems



$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$



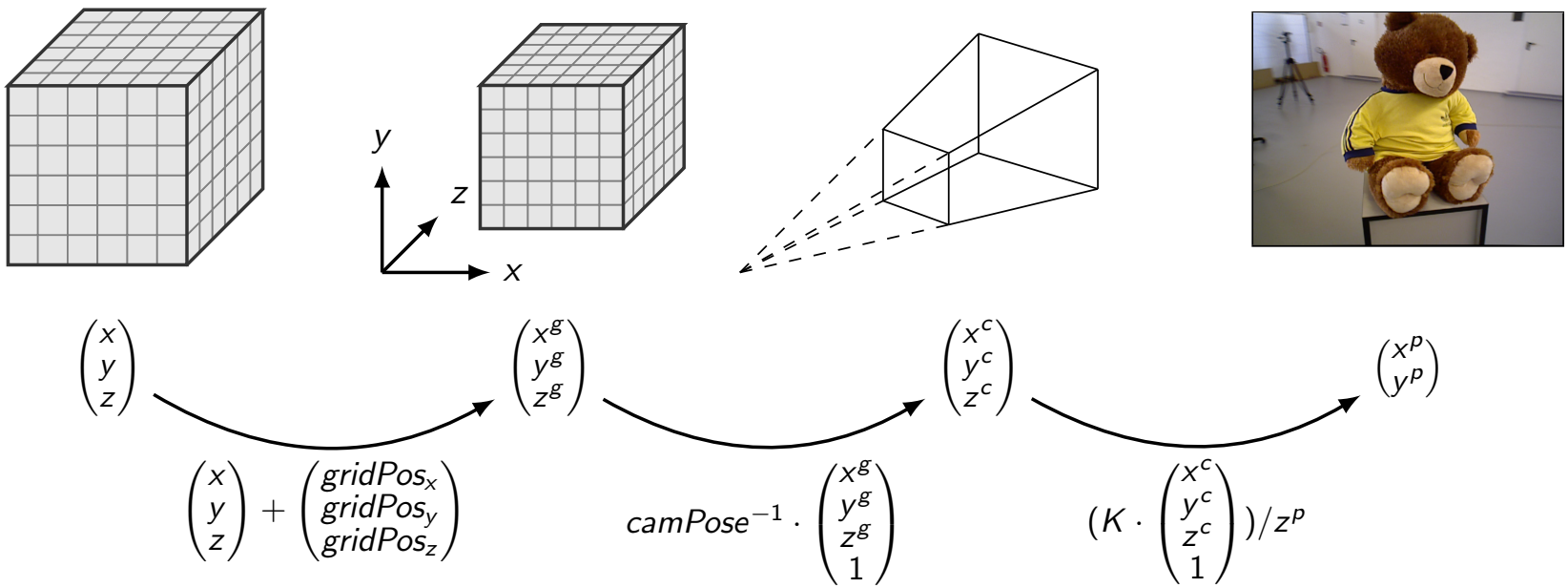
$$\begin{pmatrix} x^g \\ y^g \\ z^g \end{pmatrix}$$



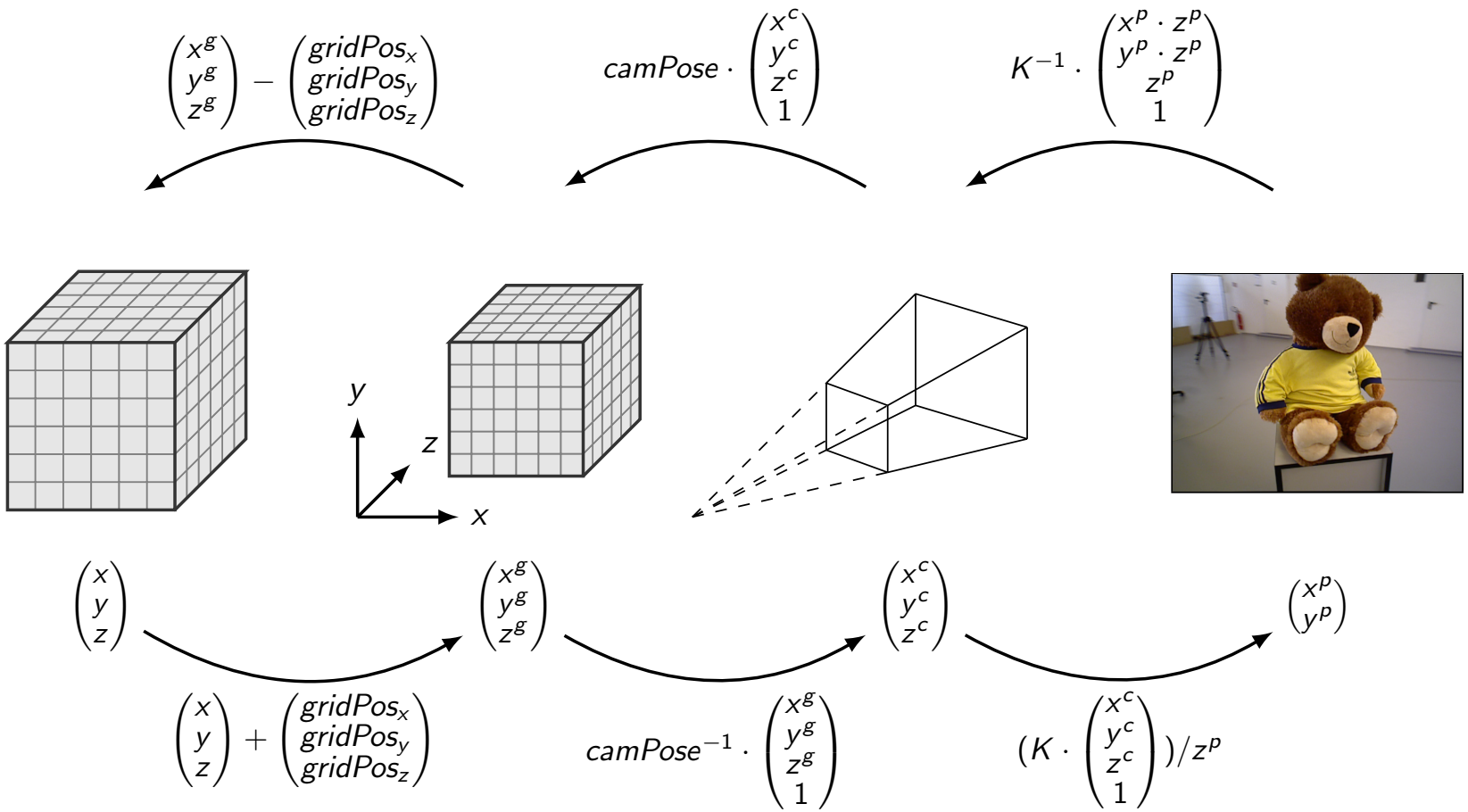
$$\begin{pmatrix} x^c \\ y^c \\ z^c \end{pmatrix}$$

$$\begin{pmatrix} x^p \\ y^p \end{pmatrix}$$

Transformations between the different coordinate systems

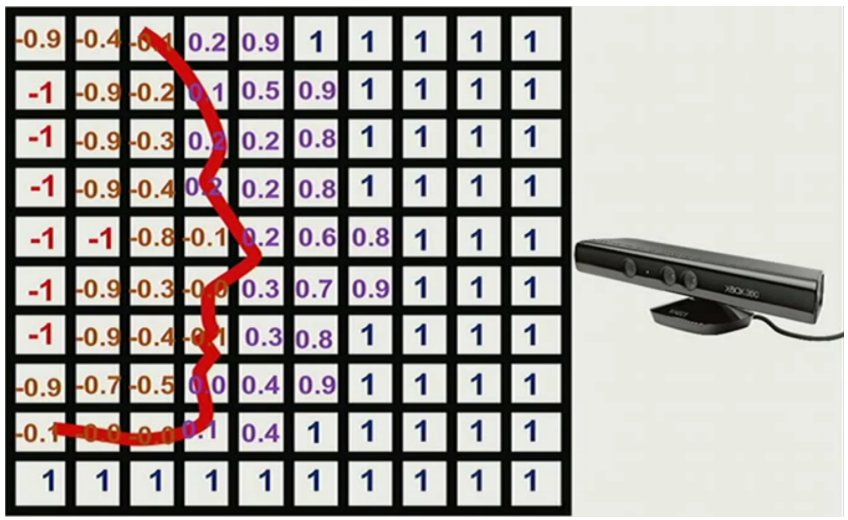


Transformations between the different coordinate systems



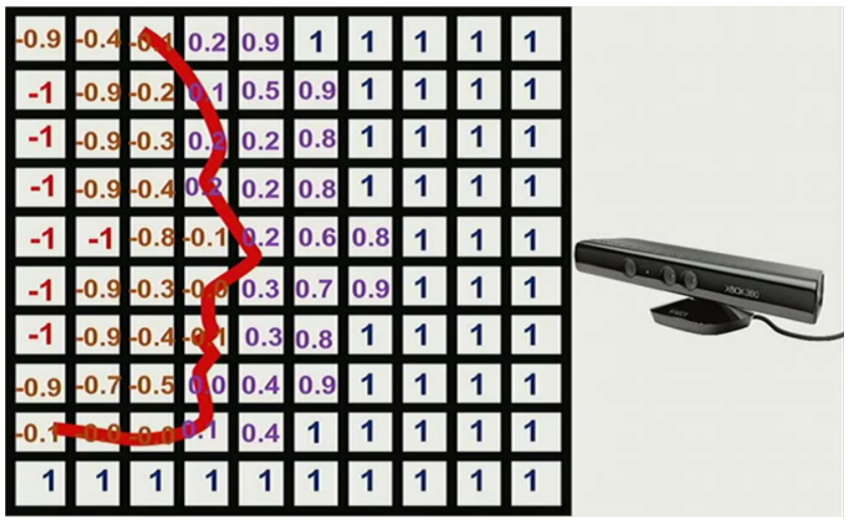
Truncated Signed Distance Function (TSDF)

- Get distance of the corresponding pixel of each voxel within the voxel grid
- Subtract it from the distance of the voxel itself and divide by the truncation threshold
- Update TSDF and color values in global memory



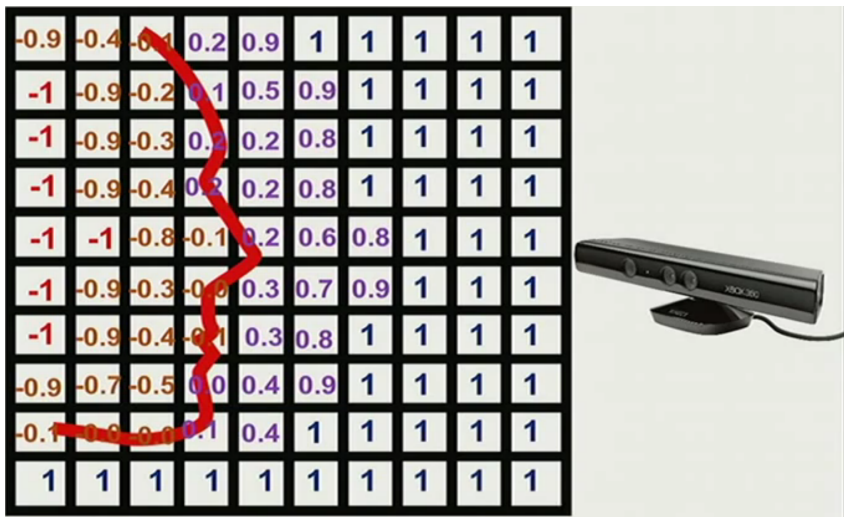
Truncated Signed Distance Function (TSDF)

- Get distance of the corresponding pixel of each voxel within the voxel grid
- Subtract it from the distance of the voxel itself and divide by the truncation threshold
- Update TSDF and color values in global memory



Truncated Signed Distance Function (TSDF)

- Get distance of the corresponding pixel of each voxel within the voxel grid
- Subtract it from the distance of the voxel itself and divide by the truncation threshold
- Update TSDF and color values in global memory



Weighting of Color and Depth Values

- Bad samples exist and must be weighted accordingly
- Idea: use angle of incidence – lower angles usually correspond to better samples
- Implementation: multiply by z-coordinate of normal

Good sample

Bad sample

Weighting of Color and Depth Values

- Bad samples exist and must be weighted accordingly
- Idea: use angle of incidence – lower angles usually correspond to better samples
- Implementation: multiply by z-coordinate of normal

Good sample

Bad sample

Weighting of Color and Depth Values

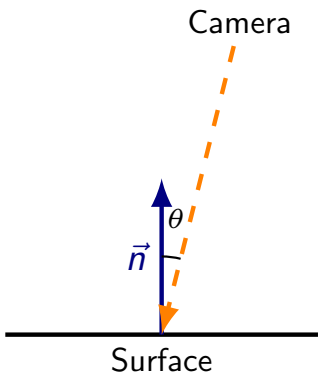
- Bad samples exist and must be weighted accordingly
- Idea: use angle of incidence – lower angles usually correspond to better samples
- Implementation: multiply by z-coordinate of normal

Good sample

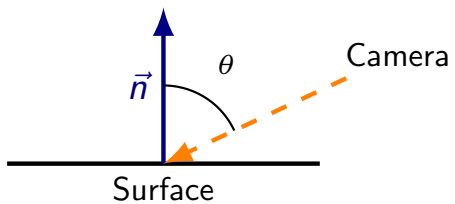
Bad sample

Weighting of Color and Depth Values

- Bad samples exist and must be weighted accordingly
- Idea: use angle of incidence – lower angles usually correspond to better samples
- Implementation: multiply by z-coordinate of normal



Good sample



Bad sample

Adaptive Raycasting

- Cast a ray for each pixel of the picture being rendered
- Take a step in z-direction and transform coordinates to voxel grid
- Check TSDF value (using trilineal interpolation); if zero-crossing (= edge) was detected, use increasingly smaller step size until we are as close to zero as possible
- Write color value (using trilinear interpolation) to picture
- Algorithm does not include lighting or shadows

Adaptive Raycasting

- Cast a ray for each pixel of the picture being rendered
- Take a step in z-direction and transform coordinates to voxel grid
- Check TSDF value (using trilineal interpolation); if zero-crossing (= edge) was detected, use increasingly smaller step size until we are as close to zero as possible
- Write color value (using trilinear interpolation) to picture
- Algorithm does not include lighting or shadows

Adaptive Raycasting

- Cast a ray for each pixel of the picture being rendered
- Take a step in z-direction and transform coordinates to voxel grid
- Check TSDF value (using trilineal interpolation); if zero-crossing (= edge) was detected, use increasingly smaller step size until we are as close to zero as possible
- Write color value (using trilinear interpolation) to picture
- Algorithm does not include lighting or shadows

Adaptive Raycasting

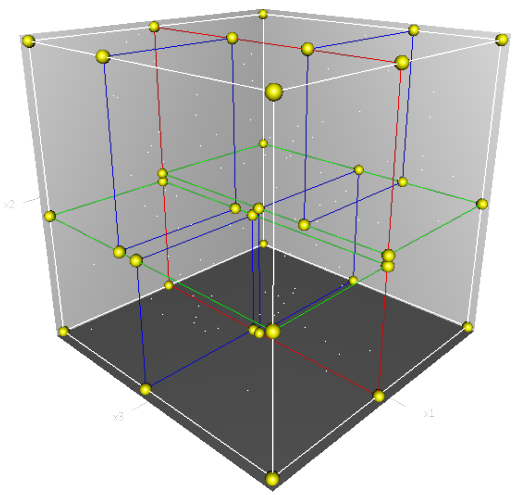
- Cast a ray for each pixel of the picture being rendered
- Take a step in z-direction and transform coordinates to voxel grid
- Check TSDF value (using trilineal interpolation); if zero-crossing (= edge) was detected, use increasingly smaller step size until we are as close to zero as possible
- Write color value (using trilinear interpolation) to picture
- Algorithm does not include lighting or shadows

Adaptive Raycasting

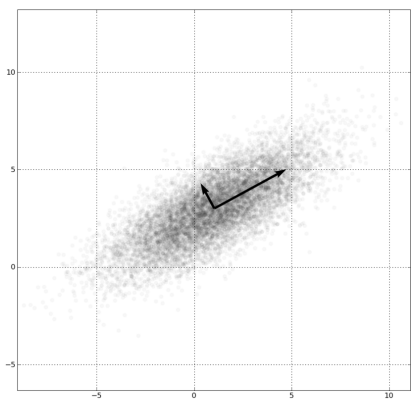
- Cast a ray for each pixel of the picture being rendered
- Take a step in z-direction and transform coordinates to voxel grid
- Check TSDF value (using trilineal interpolation); if zero-crossing (= edge) was detected, use increasingly smaller step size until we are as close to zero as possible
- Write color value (using trilinear interpolation) to picture
- Algorithm does not include lighting or shadows

Normal Calculation using PCA

- Ideal for a more robust approach to calculate normals
- Combination of k-d tree and PCA
- Not used in final version due to large performance hit (CPU implementation) and only negligible improvements



k-d tree



PCA

Robust Normal Estimation

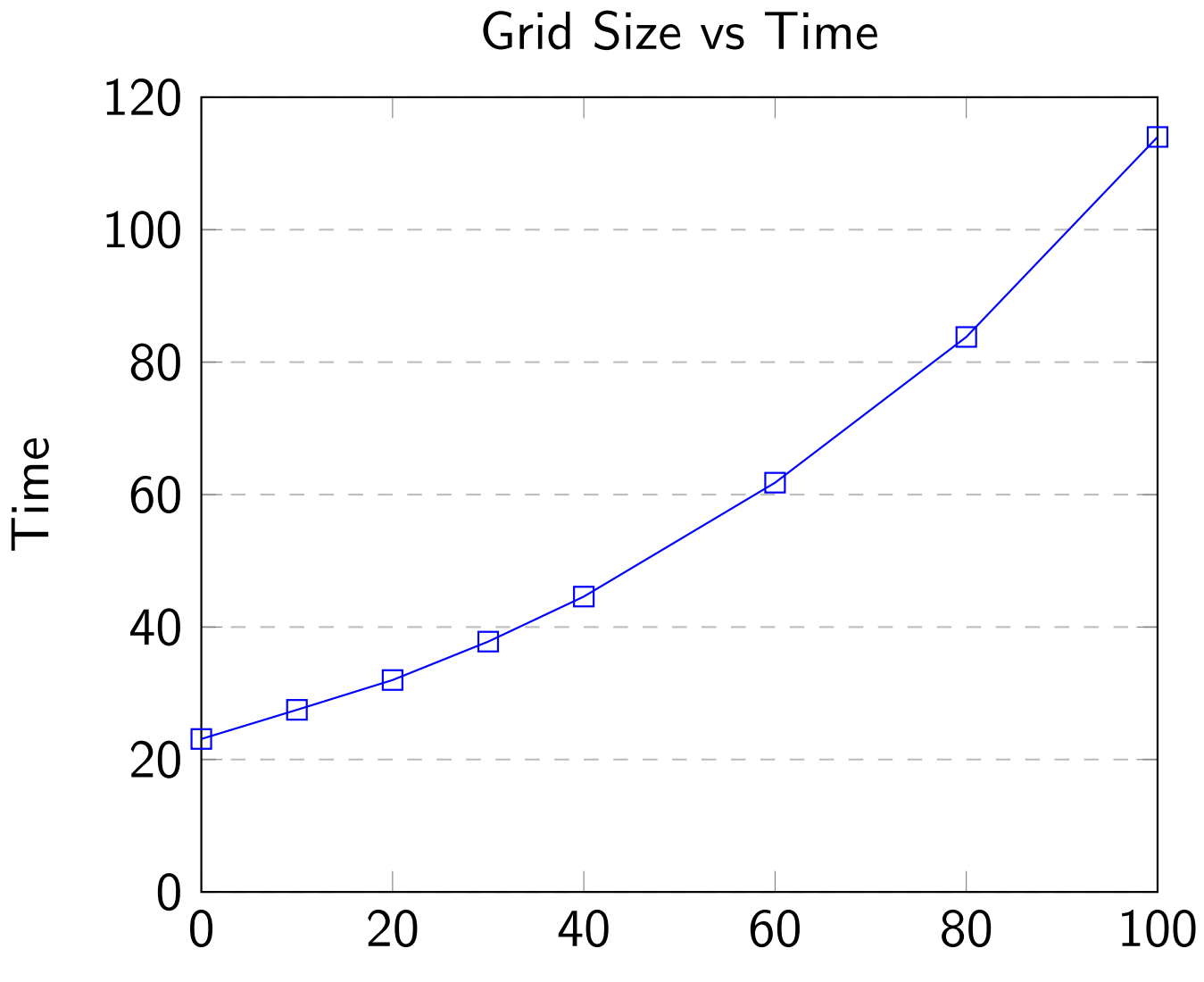
Outline

1 Introduction

2 Approach

3 Results

Results



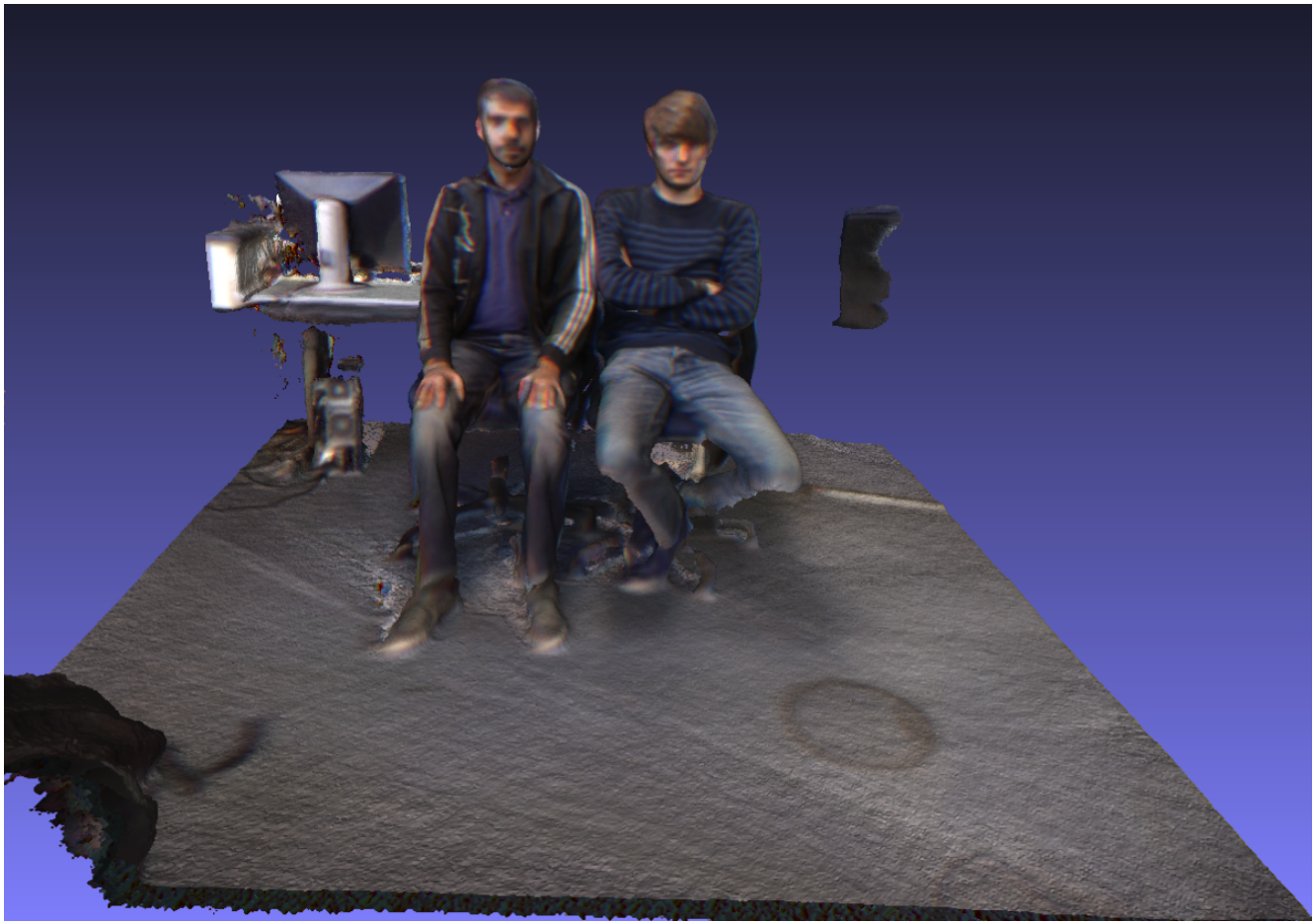
Results – different color weighting methods

Exponential falloff

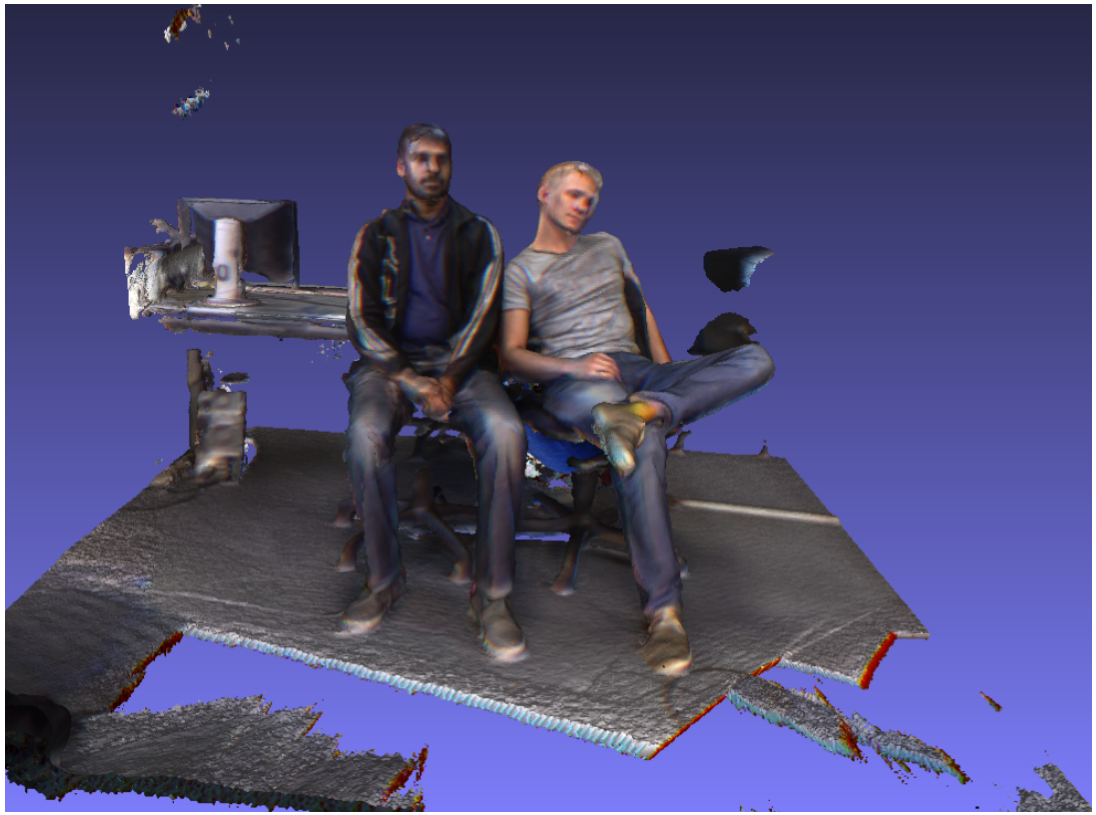
Linear falloff

No falloff

Results



Results



Results

Thank you for your attention!

References

[Curless and Levoy, 1996] Curless, B. and Levoy, M. (1996).
A volumetric method for building complex models from range images.
In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 303–312. ACM.

[Holz et al., 2012] Holz, D., Holzer, S., Rusu, R. B., and Behnke, S. (2012).
Real-time plane segmentation using rgb-d cameras.
In *RoboCup 2011: Robot Soccer World Cup XV*, pages 306–317. Springer.

[Izadi et al., 2011] Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., et al. (2011).
Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera.
In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568. ACM.

[Sturm et al., 2013] Sturm, J., Bylow, E., Kahl, F., and Cremers, D. (2013).
Copyme3d: Scanning and printing persons in 3d.
In *Pattern Recognition*, pages 405–414. Springer.