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

- Bilateral Filter
- Calculate Normals
- ICP
- TSDF Integration
- Raycasting

$t + 1$
Transformations between the different coordinate systems

\[
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
\]

\[
\begin{pmatrix}
x_g \\
y_g \\
z_g
\end{pmatrix}
\]

\[
\begin{pmatrix}
x_c \\
y_c \\
z_c
\end{pmatrix}
\]

\[
\begin{pmatrix}
x_p \\
y_p \\
z_p
\end{pmatrix}
\]

\[
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
+ \begin{pmatrix}
\text{gridPos}_x \\
\text{gridPos}_y \\
\text{gridPos}_z
\end{pmatrix}
\]

\[
\text{camPose}_{-1} \cdot \begin{pmatrix}
x_g \\
y_g \\
z_g
\end{pmatrix}
\]

\[
K \cdot \begin{pmatrix}
x_c \\
y_c \\
z_c
\end{pmatrix}/z_p
\]

\[
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
- \begin{pmatrix}
\text{gridPos}_x \\
\text{gridPos}_y \\
\text{gridPos}_z
\end{pmatrix}
\]

\[
\text{camPose}_{-1} \cdot \begin{pmatrix}
x_c \\
y_c \\
z_c
\end{pmatrix}
\]

\[
K \cdot \begin{pmatrix}
x_p \\
y_p \\
z_p
\end{pmatrix}/z_p
\]
Transformations between the different coordinate systems

\[
\begin{pmatrix}
  x \\
  y \\
  z
\end{pmatrix}
\]

\[
\begin{pmatrix}
  x^g \\
  y^g \\
  z^g
\end{pmatrix}
\]

\[
\begin{pmatrix}
  x^c \\
  y^c \\
  z^c
\end{pmatrix}
\]

\[
\begin{pmatrix}
  x^p \\
  y^p \\
  z^p
\end{pmatrix}
\]

\[
K \cdot \begin{pmatrix}
  x^c \\
  y^c \\
  z^c
\end{pmatrix}
= \begin{pmatrix}
  x^p \\
  y^p \\
  z^p
\end{pmatrix} / z_p
\]

\[
K^{-1} \cdot \left( x^g y^g z^g \right)
- \left( \begin{array}{c}
  x^g \\
  y^g \\
  z^g
\end{array} \right)
\]

\[
\text{camPose} - 1 \cdot \left( \begin{array}{c}
  x^g \\
  y^g \\
  z^g
\end{array} \right)
\]

\[
\text{gridPos} \cdot \left( \begin{array}{c}
  x^g \\
  y^g \\
  z^g
\end{array} \right)
\]

\[
\text{volPose} \cdot \left( \begin{array}{c}
  x^c \\
  y^c \\
  z^c
\end{array} \right)
\]
Transformations between the different coordinate systems

\[
\begin{pmatrix}
x \\
y \\
z
\end{pmatrix}
\rightarrow
\begin{pmatrix}
x_g \\
y_g \\
z_g
\end{pmatrix}
\rightarrow
\begin{pmatrix}
x_c \\
y_c \\
z_c
\end{pmatrix}
\rightarrow
\begin{pmatrix}
x_p \\
y_p \\
z_p
\end{pmatrix}
\]

\[
\begin{pmatrix}
x_g \\
y_g \\
z_g
\end{pmatrix}
1
\cdot
\begin{pmatrix}
x_c \\
y_c \\
z_c
\end{pmatrix}

K
\cdot
\begin{pmatrix}
x_c \\
y_c \\
z_c
\end{pmatrix}
/z_p
\]

\[
\begin{pmatrix}
x_g \\
y_g \\
z_g
\end{pmatrix}
- \begin{pmatrix}
\text{gridPos}_x \\
\text{gridPos}_y \\
\text{gridPos}_z
\end{pmatrix}
\cdot
\begin{pmatrix}
x_c \\
y_c \\
z_c
\end{pmatrix}

K
-1
\cdot
\begin{pmatrix}
x_p \\
y_p \\
z_p
\end{pmatrix}
\]
Transformations between the different coordinate systems

\[
\mathbf{x} = \mathbf{x}_c + \mathbf{gridPos}_{\mathbf{c}} - 1 \cdot \begin{bmatrix} x_g \\ y_g \\ z_g \end{bmatrix} + \mathbf{K}^{-1} \begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} / z_p
\]
Transformations between the different coordinate systems

\[
\begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} \text{gridPos}_x \\ \text{gridPos}_y \\ \text{gridPos}_z \end{pmatrix} = \begin{pmatrix} x^g \\ y^g \\ z^g \end{pmatrix}
\]

\[
camPose^{-1} \cdot \begin{pmatrix} x^g \\ y^g \\ z^g \end{pmatrix} = \begin{pmatrix} x^c \\ y^c \\ z^c \end{pmatrix}
\]

\[
(K \cdot \begin{pmatrix} x^c \\ y^c \\ z^c \\ 1 \end{pmatrix}) / z^p
\]
Transformations between the different coordinate systems

\[
\begin{align*}
\begin{pmatrix} x^g \\ y^g \\ z^g \end{pmatrix} & - \begin{pmatrix} \text{gridPos}_x \\ \text{gridPos}_y \\ \text{gridPos}_z \end{pmatrix} \\
\begin{pmatrix} x \\ y \\ z \end{pmatrix} & + \begin{pmatrix} \text{gridPos}_x \\ \text{gridPos}_y \\ \text{gridPos}_z \end{pmatrix} \\
\begin{pmatrix} x^c \\ y^c \\ z^c \end{pmatrix} & = \text{camPose} \cdot \begin{pmatrix} x^c \\ y^c \\ z^c \end{pmatrix} \\
\begin{pmatrix} x^p \\ y^p \end{pmatrix} & = \left( \frac{K \cdot \begin{pmatrix} x^c \\ y^c \\ z^c \end{pmatrix}}{z^p} \right) / z^p
\end{align*}
\]

Martin Herrmann  Simon Trendel  Neeraj Sujan: TSDF Volume Reconstruction
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
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

![Diagram showing good and bad samples with angle of incidence and normal vector](image)
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
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

![Diagram showing good and bad samples with camera and surface](image-url)
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 trilinear 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 trilinear 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 trilinear 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 trilinear 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 trilinear 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
Outline

1. Introduction

2. Approach

3. Results
Results

Grid Size vs Time

Time vs Grid Size
Results – different color weighting methods

(a) Exponential falloff

(b) Linear falloff

No falloff
Results
Results
Results
Thank you for your attention!
References


