# Real-time 3D Reconstruction and Localization

## Hauptseminar Computer Vision & Visual Tracking for Robotic Applications SS2012

Robert Maier

Technische Universität München
Department of Informatics
Robotics and Embedded Systems

12.06.2012

# Overview


1. Motivation
2. Existing related work
3. Real-time 3D Reconstruction and Localization
   - Basic approach
   - 3D data aquisition
   - Reconstruction: Volume representation & ray casting
   - Localization
   - Volumetric integration
   - Visualization
4. Examples
5. Conclusion

# Outline

1 Motivation

2 Existing related work

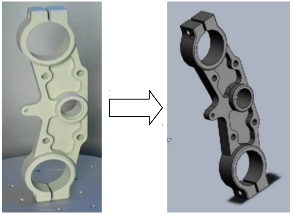
3 Real-time 3D Reconstruction and Localization
- Basic approach
- 3D data aquisition
- Reconstruction: Volume representation & ray casting
- Localization
- Volumetric integration
- Visualization

4 Examples

5 Conclusion

# Motivation

- Reverse Engineering: How to get a CAD model of a work piece? (e.g. for measuring purposes)
  $\rightarrow$ Elaborate manual creation!



- Environment reconstruction for robots

# Motivation: Task

General task: reconstruct 3D models of real world scenes

→ Solution: **Real-time 3D Reconstruction and Localization**

- 3D Reconstruction: build digital 3D model from physical object
- Localization: track camera to fuse different views
- Desired properties:
  - Infrastructure- and marker-free
  - (Single) handheld camera
  - Real-time capability
  - Accurate reconstruction

# Outline

1 Motivation

2 Existing related work

3 Real-time 3D Reconstruction and Localization
- Basic approach
- 3D data aquisition
- Reconstruction: Volume representation & ray casting
- Localization
- Volumetric integration
- Visualization

4 Examples

5 Conclusion

# Existing related work

Approaches:

- SLAM: Simultaneous Localization And Mapping
- PTAM: Parallel Tracking And Mapping

Limitations:

- Only use of sparse (feature-based) depth maps
- Reconstruction of only sparse models, used mostly for tracking

# Outline

1 Motivation

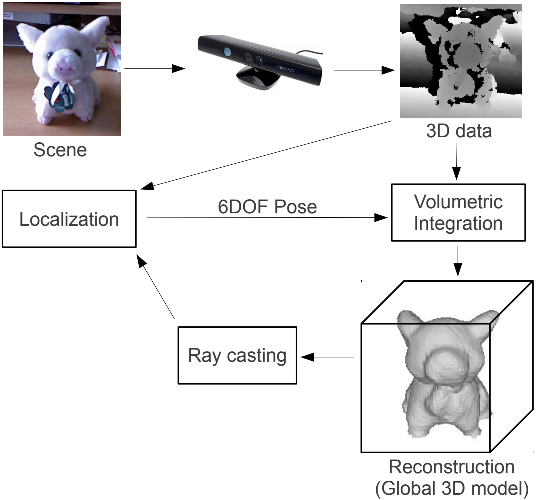
2 Existing related work

3 Real-time 3D Reconstruction and Localization
- Basic approach
- 3D data aquisition
- Reconstruction: Volume representation & ray casting
- Localization
- Volumetric integration
- Visualization

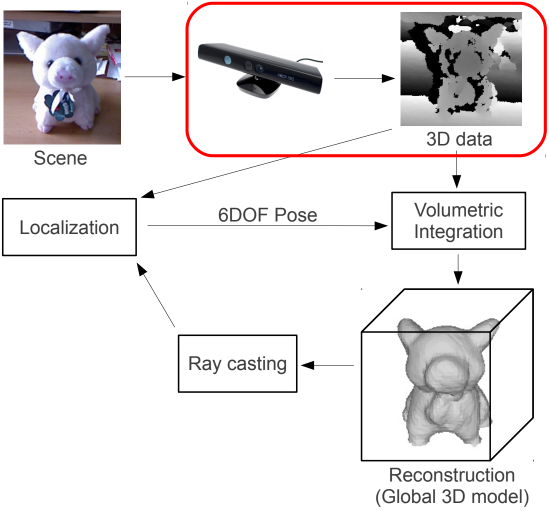
4 Examples

5 Conclusion

# Basic approach



# Basic approach: 3D data aquisition



# Aquisition of depth maps

Dense depth map: 2D image with distances to next surface at *each* pixel

Depth sensors:

- Microsoft Kinect
- Time-of-flight cameras
- 3D laser scanner

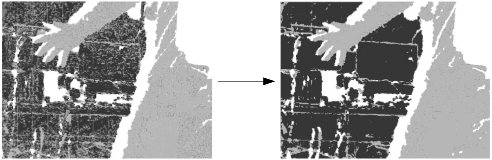

Standard cameras:

- Stereo vision setup
- Single camera: depth from multiple views

# Depth map conversion

Initial data: **depth map**

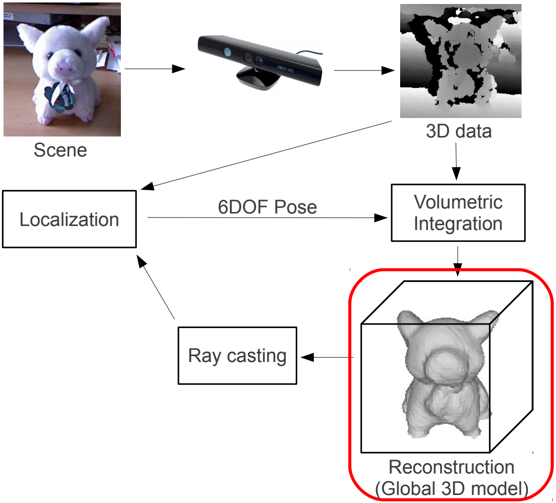
→ Remove noise using bilateral filter: **filtered depth map**



→ Project into 3D camera space: **vertex map** (point cloud)

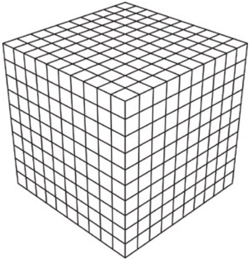
→ Compute surface normals for each vertex: **normal map**

# Basic approach: Volume representation



# Volumetric scene representation

- Goal: Find representation for reconstructed 3D model
- Store all point clouds from all frames → too expensive
- Better: partition physical 3D space into discrete 3D voxel grid

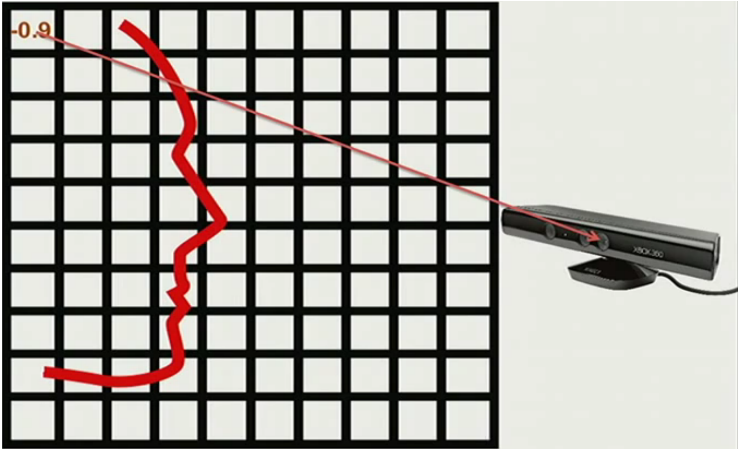


- Fuse all depth maps from different views into this global model
- Implicit surfaces: Truncated Signed Distance Function (TSDF)

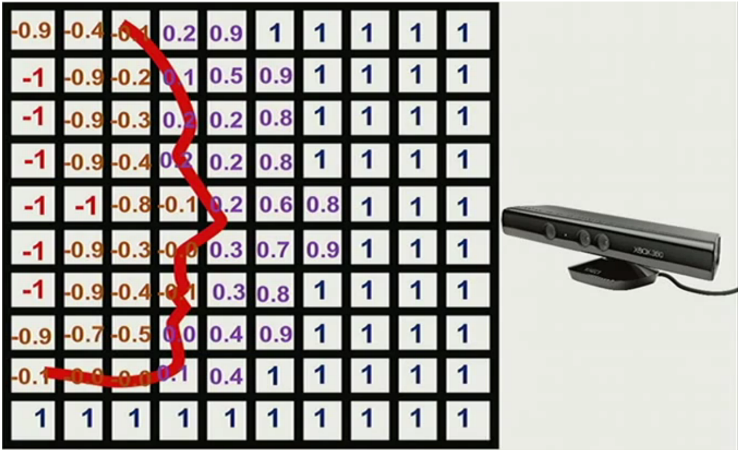
# Truncated Signed Distance Function

- Object lies inside volume
- Assumption: Pose of camera w.r.t. to volume known
- Compute and store for each voxel of the volume:
  - Truncated signed distance: relative distance of each voxel to the surface (between -1.0 and 1.0)
  - Accumulated weight (for volumetric integration)
- Object surface defined by zero-crossings of TSDF

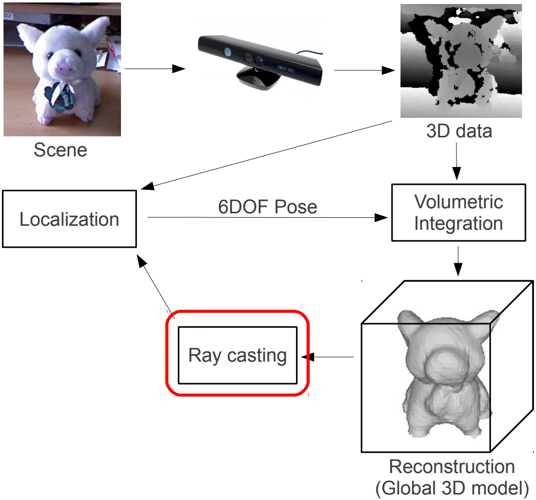
# Truncated Signed Distance Function



# Truncated Signed Distance Function

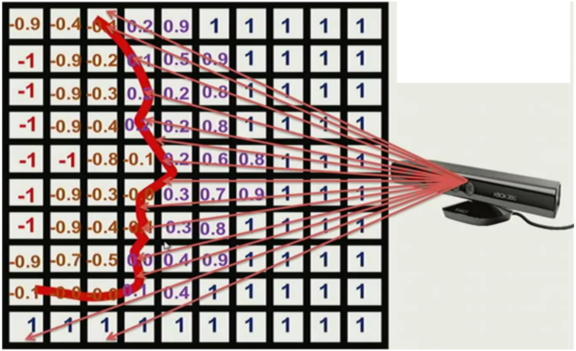


# Basic approach: Ray casting

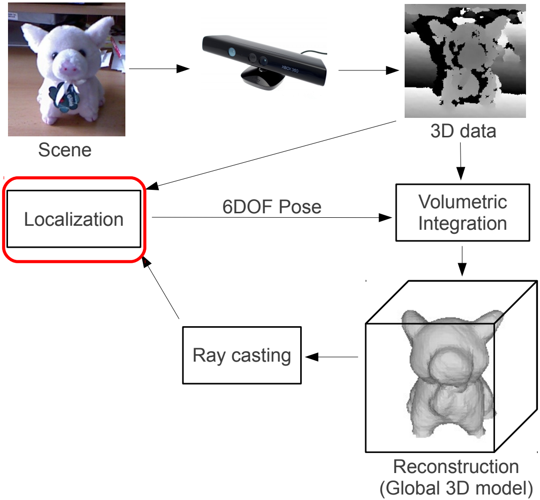


# Ray casting

- Given pose (virtual camera): Shoot rays through volume
- Zero-crossing of TSDF values → surface
- Generate synthetic depth map



# Basic approach: Localization



# Localization: Approaches

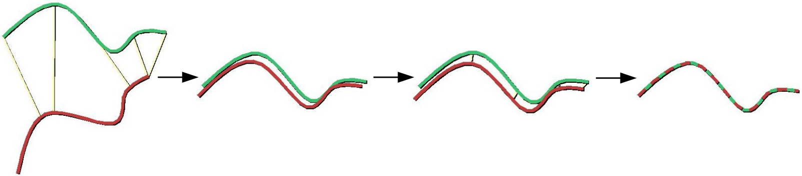
- Estimate camera pose for each new depth image w.r.t. global model frame


- Two different approaches:
  - Scan Alignment (KinectFusion)
  - Image Alignment (DTAM)

# Scan alignment: Iterative Closest Points algorithm

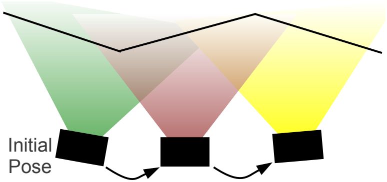
- Align overlapping point clouds (roughly aligned)
- Compute 6DOF pose between these scans
- Use of dense depth maps: more accurate localization

- ICP algorithm: Minimize distances between two point clouds
  - Find closest pairs of points in the two point clouds
  - Minimize distances between all closest points (= align scans)
  - Compute closest points again and minimize distances
  - Repeat steps iteratively until convergence



# Localization using Scan Alignment

- First frame: predefined camera pose w.r.t. global volume
- Then: Align vertex map of current frame to previous frame:
  - Vertex map of previous frame:
    - Raw depth map of camera → significant drift
    - Better: Ray casting of synthetic depth map of global model
  - ICP computes transformation w.r.t. previous frame
  - Camera pose w.r.t. global model: combine consecutive transformations

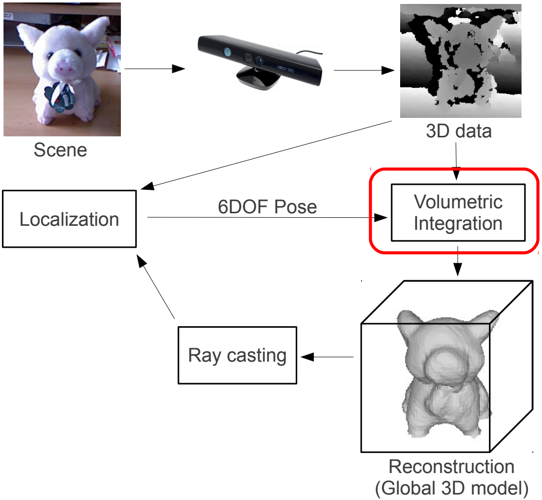



- Camera motion too far: lost tracking → Re-localization

# Localization using Image Alignment

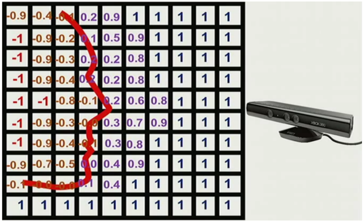
- 2.5D image alignment
- Ray casting from the global model (starting from previous camera pose):
  - synthetic color image (from keyframes)
  - synthetic inverse depth image
- Compare synthetic image with current real image
- Adjust and find virtual camera pose which gives best match between synthetic and real image $\rightarrow$ 6DOF camera pose

# Basic approach: Volumetric integration



# Volumetric integration

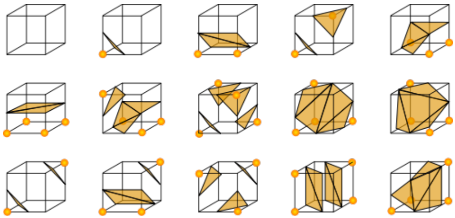
- 6DOF pose (localization) $\rightarrow$ project current vertex map into global volume
- Compute for each voxel using projected vertex map:
  - TSDF values: relative distances




  - Weight: angle between surface normal and viewing direction
- Update voxels of global model:
  - Update only visible voxels
  - Average TSDF values of voxels with new values
  - Add new weight to weight of voxels

# Visualization: Marching cubes algorithm

- Compute polygonal mesh of an isosurface from 3D voxel grid:
  - Take eight neighbor voxels of voxel grid (cube)
  - Scalar values of cube corners: polygons to represent isosurface going through this cube
  - 8 bits: 1, if scalar value is lower than isosurface value (inside the surface), otherwise 0
  - Based on 8 bits: choose polygon representation (lookup table)
  - Place polygon vertices on cube edges (linear interpolation)



# Outline

1 Motivation

2 Existing related work

3 Real-time 3D Reconstruction and Localization
- Basic approach
- 3D data aquisition
- Reconstruction: Volume representation & ray casting
- Localization
- Volumetric integration
- Visualization

4 Examples

5 Conclusion

# KinectFusion

Play time!


*http://reconstructme.net*

# Outline

1 Motivation

2 Existing related work

3 Real-time 3D Reconstruction and Localization
- Basic approach
- 3D data aquisition
- Reconstruction: Volume representation & ray casting
- Localization
- Volumetric integration
- Visualization

4 Examples

5 Conclusion

# Conclusion

Achievements:

- Accurate Reconstruction of full workspaces
- Real-time capability (due to use of GPU)

Limitations:

- Voxel model unflexible
- Limited support for large areas
- Only rigid scenes (no deformations)
- Kinect sensor (only indoor)

→ Current field of research!