KinectFusion: Real-Time Dense Surface Mapping and Tracking

Richard A. Newcombe[+⊤] Shahram Izadi[⊤] Otmar Hilliges[⊤] David Molyneaux[⊤] David Kim[⊤] Andrew J. Davison[+] Pushmeet Kohli[⊤] Jamie Shotton[⊤] Andrew Fitzgibbon[⊤]

[+]Imperial College, London

[⊤]Microsoft Research, Cambridge

October 28, 2011

Outline

Table of Contents

Need for *infrastructure free* tracking and *surface* mapping

Joint Tracking of a sensor pose and Mapping of scene geometry also called simultaneous localisation and mapping (SLAM) is at the Core of robotics and AR/MR applications.

Mixed and Augmented Reality

A first requirement of augmented reality is the requirement to track a camera pose accurately. Increasing predictive quality depends on building and keeping up to date a model of the environments geometry, illumination and surface material properties.

Robotics: Scene interaction vs. Obstacle avoidance/navigation

A robot needs sense of its surrounding surfaces if it is to competently interact with it. This is quite a different challenge to modelling the scene for navigation purposes alone.

Real-time Motivation

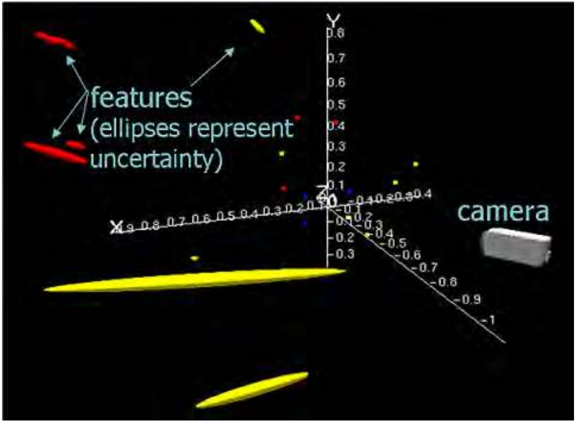
Live incremental scene reconstruction vs. Offline batch methods

There are a number of reasons why an incremental approach is required, but more importantly there are a number of useful constraints when thinking about dense reconstruction with an embodied live stream instead of an unordered collection of still frames.

Table of Contents

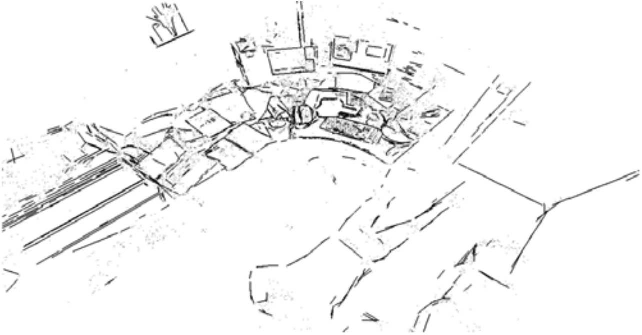
Real time, commodity SLAM system evolution

2003 Davison's Monoslam: importance of a cheap comodity sensor



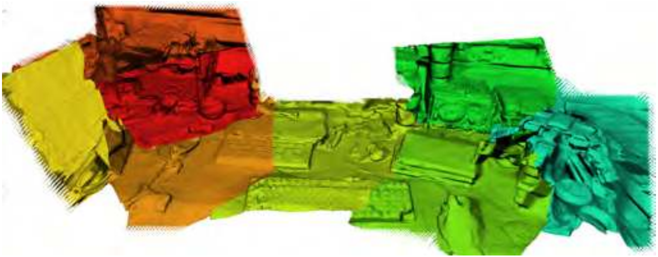
Real time, commodity SLAM system evolution

2007,2008 Klein and Murray's PTAM, also passive, optimised software using features of the CPU. Maps are much denser than monoSLAM, but still not surfaces.



Real time, commodity SLAM system evolution

2010 Newcombe and Davison, augmenting the sparse tracking and mapping with dense surface estimation method. Utilising GPU power, live but not real-time and no way to correct grossly wrong geometry.



Research Live *dense* reconstruction from a passive camera is gathering pace (see upcoming Workshop at ICCV this year). However, passive methods will always fail when light levels are too low.

Real time, commodity SLAM system evolution

Now, KinectFusion: Dense real-time surface geometry and robust tracking even in complete darkness.

Real time, commodity SLAM system evolution

Now, KinectFusion: Dense real-time surface geometry and robust tracking even in complete darkness.

What's changed?

Depth cameras have become commodity along with the massive parallel processing capabilities now available.

Amazing commodity hardware capabilities

Kinect camera:
Real-time depth measurement

GPGPU:
Massive processing capabilities

This pairing of New technology changes what makes a solution scalable or elegant for SLAM.

Key Technology (1)

Commodity Depth Sensor

Real-time high quality depth maps from Kinect sensor. Vertex and normal maps. One of the most exciting prospects of this technology is that it's active! So low/dynamic lighting conditions are much less of problem.

- No computational cost to user.
- Given known camera intrinsics, K, a depth map at time k provides a *scale correct* 3D point measurement at each pixel; a vertex map $\mathbf{V}_k$.
- Using a cross product on neighbouring points we can compute an estimate of the surface normal at each depth pixel; normal map $\mathbf{N}_k$.

Key Technology (2)

Powerful GPGPU processing

Liberates us from worrying (too much) about efficiency before understand the core approaches possible.

- e.g. MonoSLAM/PTAM struggles with 100s/1000s of point features but now we can integrate and track millions of points per second.
- Representation is important: a surface measurement is not *just* a point cloud — it's much richer.
- Computational requirement hockey stick: once we get to a certain capability, certain representations are feasible that enable integration of all data all of the time.

- CUDA and OpenCL provide higher level languages with which to program the GPU. For many implementations that trivially map, the code can look nearly identical to normal C/C++.

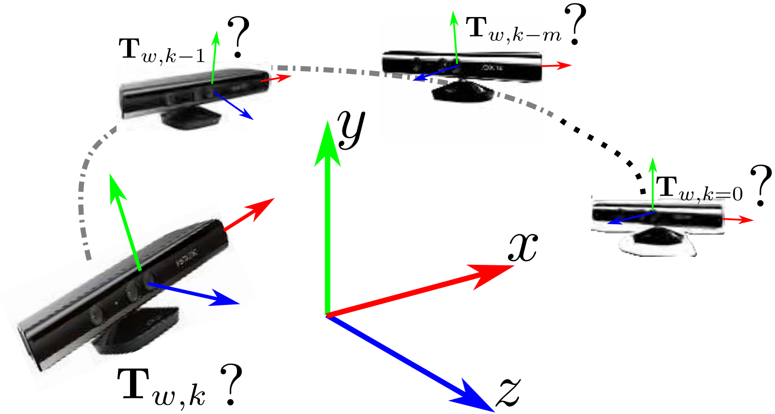
Table of Contents

What is KinectFusion?

Two *simple* interleaved components

1. Building a dense surface model from a set of depth frames with estimated camera poses.
2. Given a dense surface model, estimate the current camera pose by aligning the depth frame in the dense model.

Joint Estimation Problem: What is the camera motion and surface geometry?



Camera Motion: pose over time

For frame *k* the pose of the camera (this refers in this case to the infra-red sensor of the Kinect camera) is given by the six degree of freedom rigid body transform:

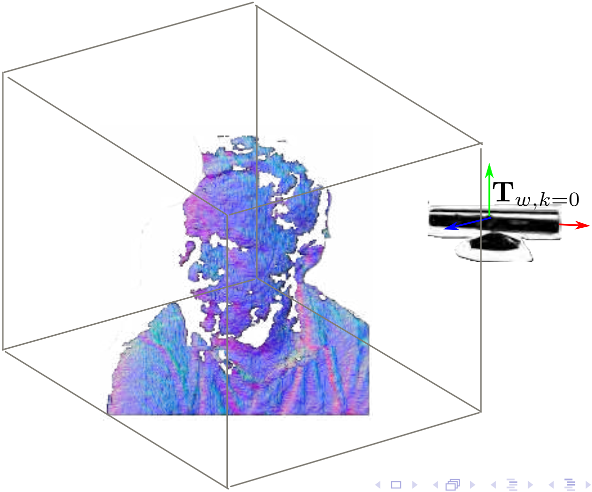
$$\mathbf{T}_{w,k} = \begin{bmatrix} \mathbf{R}_{w,k} & \mathbf{t}_{w,k} \\ \mathbf{0}^\top & 1 \end{bmatrix} \in \mathbb{SE}_3$$

$$\mathbb{SE}_3 := \{\mathbf{R}, \mathbf{t} \mid \mathbf{R} \in \mathbb{SO}_3, \mathbf{t} \in \mathbb{R}^3\}$$

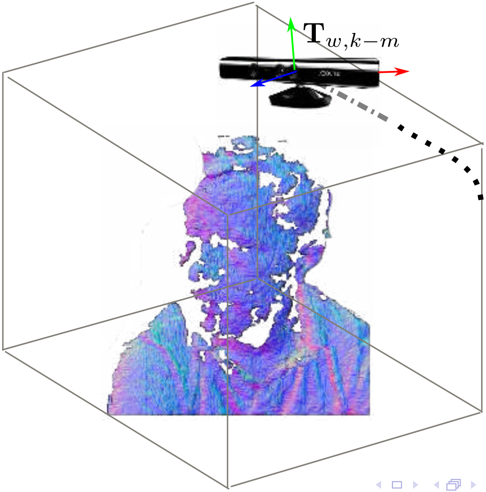
Depth map to Dense 3D surface measurement

We can transform any depth map from its local frame depth map into a global frame surface measurement.

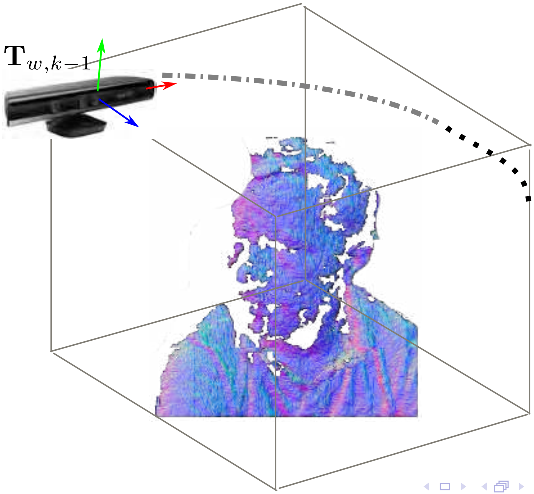
Knowing camera motion, enables model reconstruction...



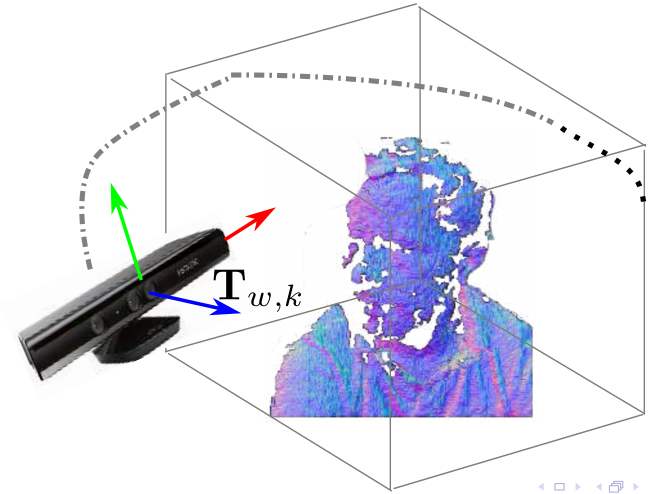
Knowing camera motion...



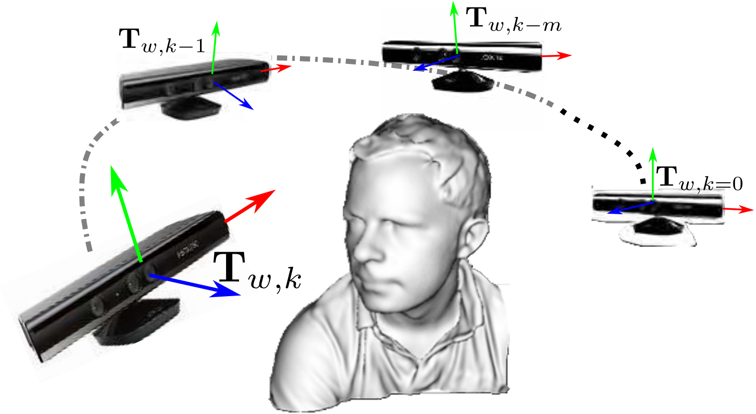
Knowing camera motion...



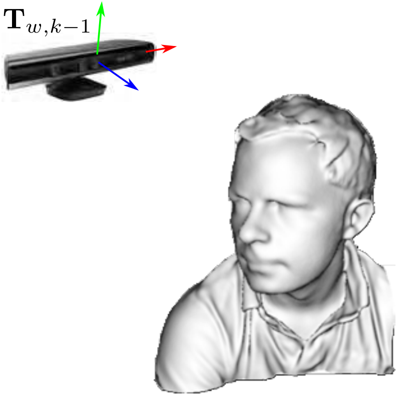
Knowing camera motion...



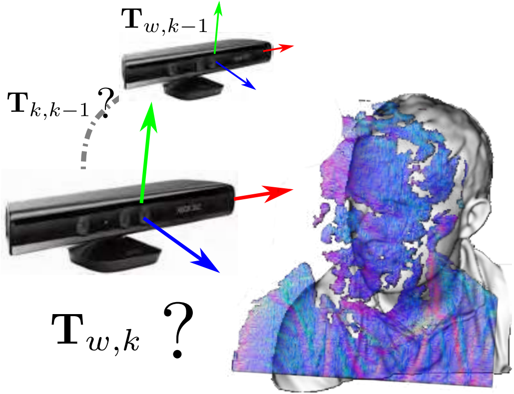
...enables measurement fusion (surface reconstruction)...



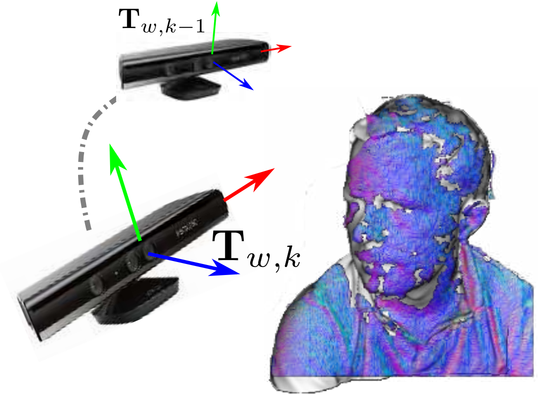
...also, given a known model...



...we can align a new surface measurement...



...minimising the predicted surface measurement error...



...giving us a best current pose estimate, enabling fusion.

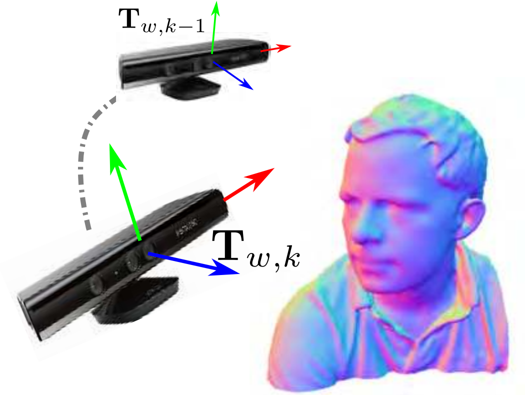


Table of Contents

Dense Mapping as Surface Reconstruction

- There are many techniques from computer vision and graphics for taking a noisy point cloud and turning it into a complete surface estimate.
- Representation is important, we don't want to restricted in surface topology or precision.
- We want to use all the data available.

Use all data

We want to integrate over $640 \times 480 \times 30 \approx 9.2$ Million depth measurements per second on commodity hardware.

- Point clouds are *not* surfaces. Meshes or parametric patches have problems with merging different topologies.

Signed Distance Function surface representations

We use a *truncated signed distance* function representation, $\mathrm{F}(\vec{x}) : \mathbb{R}^3 \mapsto \mathbb{R}$ for the estimated surface where $\mathrm{F}(\vec{x}) = 0$.

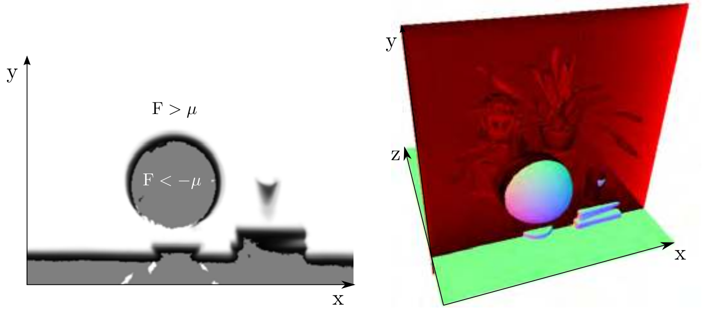
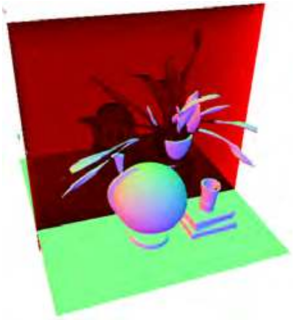
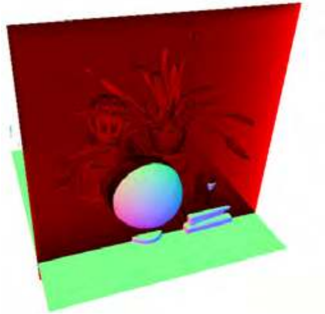


Figure: A cross section through a 3*D* Signed Distance Function of the surface shown.

Signed Distance Function surfaces

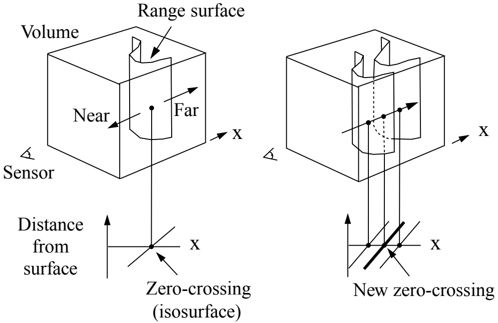


Signed Distance Function surfaces

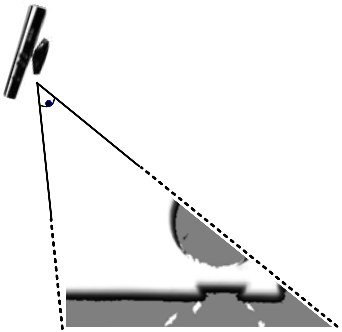


Surface reconstruction via depth map fusion

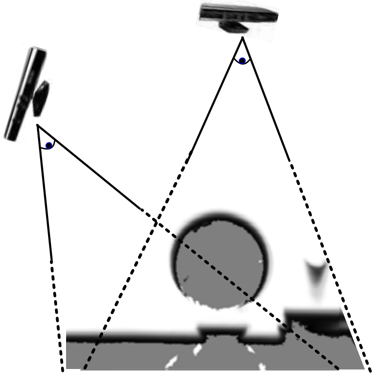
Curless and Levoy (1996) introduced very simple method for fusing depth maps into a global surface using the signed distance function representation.



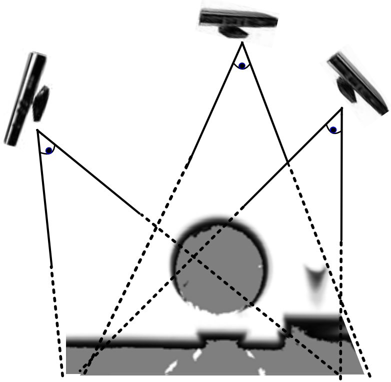
SDF Fusion



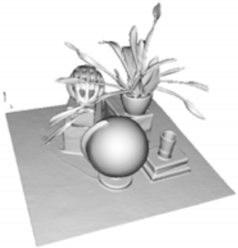
SDF Fusion



SDF Fusion

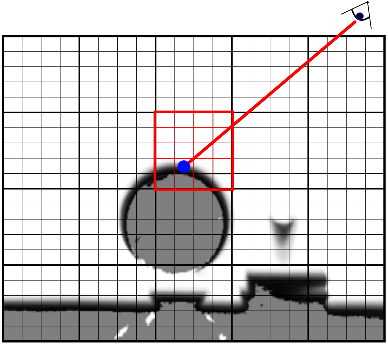


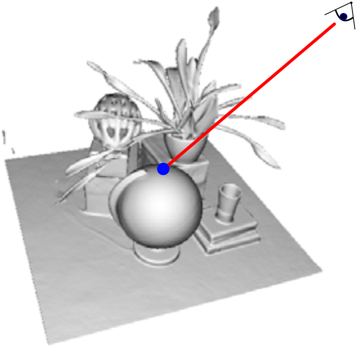
SDF Fusion



Reconstruction by averaging signed distance function versions of depth measurements along measurement ray lines. Equivalent to volumetric denoising of the SDF under an $\mathcal{L}_2$ norm data-cost with no regularisation: Can be computed online as data comes in using weighted average.

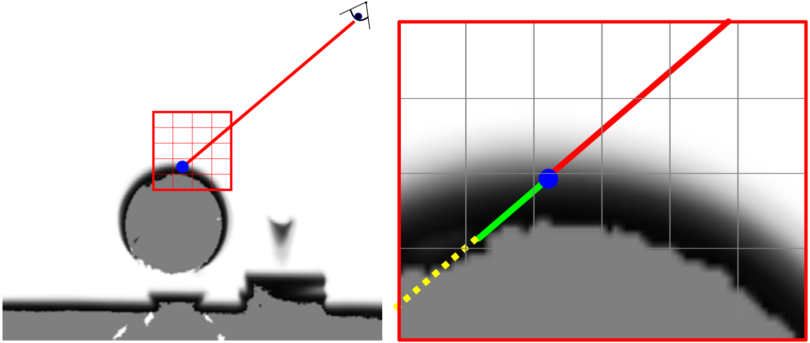
Rendering a surface represented in SDF





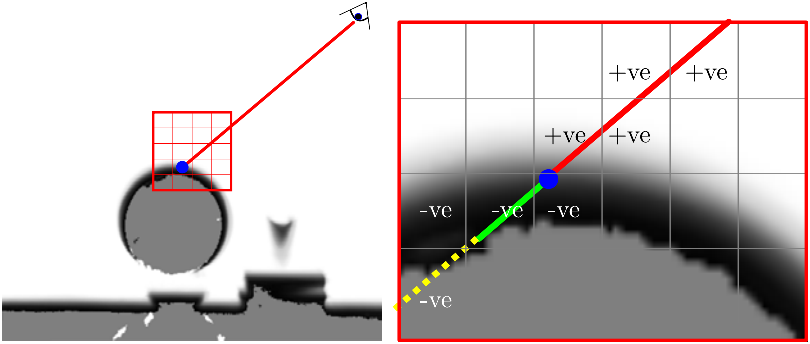
A regular grid holds a discretistion of the SDF. Ray-casting of iso-surfaces (S. Parker et al. 1998) is an established technique in graphics.

Rendering a surface represented in SDF



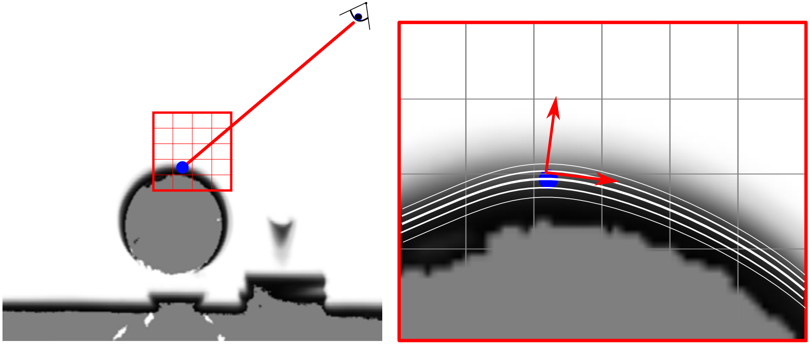
A regular grid holds a discretistion of the SDF. Ray-casting of iso-surfaces S. (Parker et al. 1998) is an established technique in graphics.

Rendering a surface represented in SDF



Interpolation reduces quantisation artefacts, and we can use the SDF value in a given voxel to skip along the ray if we are far from a surface.

Rendering a surface represented in SDF



Near the level sets near the zero crossing are parallel. The SDF field implicitly represents the surface normal.

Dense Mapping as Surface Reconstruction

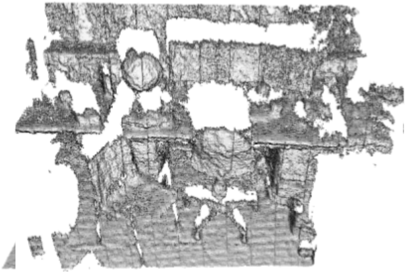
Dense Mapping Algorithm

Given depth map R_k and pose $\mathbf{T}_{k,w}$, For each voxel $\mathbf{p}$ within frustum of frame k update the Truncated Signed Distance function:

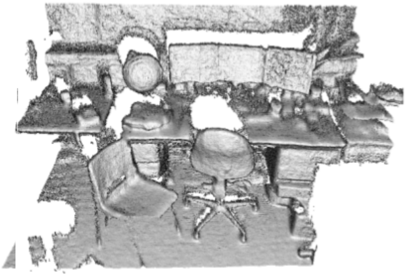
1. Project voxel into frame k: $\mathbf{x} = \pi(\mathrm{K}\mathbf{T}_{k,w}\mathbf{p})$
2. Compute signed distance between $\lambda^{-1}\|\mathbf{p} - \mathbf{t}_{w,k}\|$ and depth for this pixel $R_k(\mathbf{x})$
3. Truncate the signed distance.
4. Update the weighted average TSDF value for this voxel.

Using this approach we can integrate over $640 \times 480 \times 30 \approx 9.2$ Million depth measurements per second on high end laptop grade GPGPU.

TSDF Fusion



TSDF Fusion



TSDF Fusion

Table of Contents

Tracking as Depth Map to Dense surface alignment

- Use all available depth data.
- Using only depth data, we can use Iterated Closest Point (ICP) based surface alignment introduced by P. Besl and N. McKay (1992).

Surface Alignment Outline

1. Obtain correspondences between a surface measurement and the surface model
2. Find the transform for the surface measurement that minimises the surface-model correspondence distance (we use the point-plane metric by Y. Chen and G. Medioni, 1992).

Camera Tracking using a predicted depth map

Point-Plane ICP optimisation

We align the live vertex map onto the previous frame predicted view using a point-plane based ICP (iterated closest point), minimising the following whole image cost for the desired transform $\mathrm{T}_{g,k} \in \mathbb{SE}(3)$:

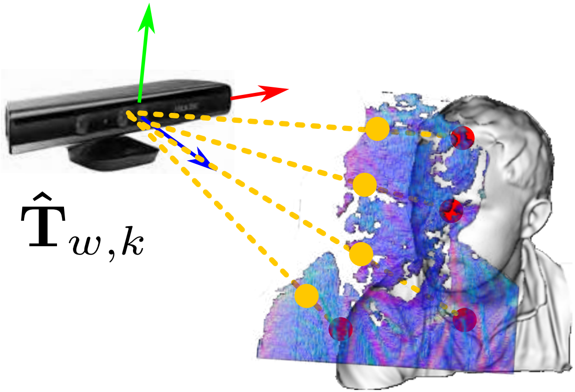
$$\mathbf{E}(\mathrm{T}_{g,k}) = \sum_{\substack{\mathbf{u}\in\mathcal{U} \\ \Omega_k(\mathbf{u})\neq \mathrm{null}}} \left\| \left(\mathrm{T}_{g,k}\dot{\mathbf{V}}_k(\mathbf{u}) - \hat{\mathbf{V}}^g_{k-1}(\hat{\mathbf{u}}) \right)^\top \hat{\mathbf{N}}^g_{k-1}(\hat{\mathbf{u}}) \right\|_2 ,$$

The optimisation is embedded in a coarse to fine scheme and requires

data-association between the predicted and live vertex data.

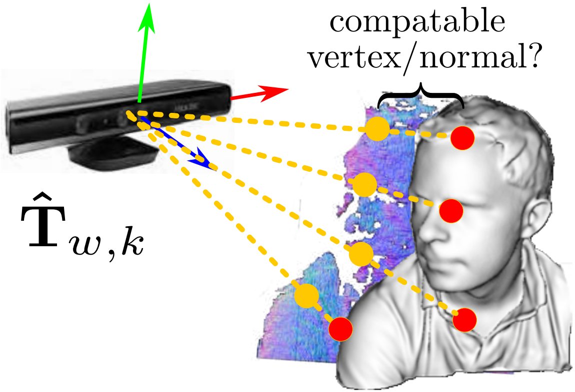
We use projective data-association (G. Blais and M. D. Levine. 1995) to obtain fast dense correspondences.

Projective Data Association

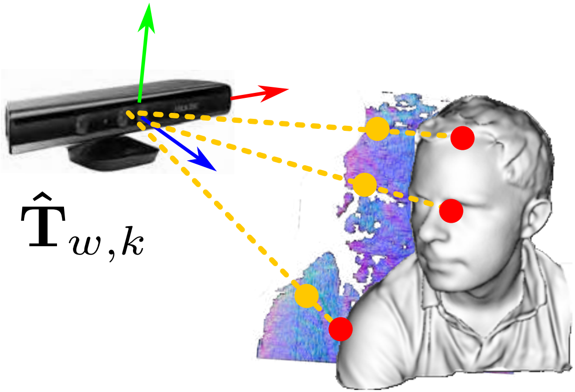


$\hat{\mathrm{T}}_{w,k}$

Projective Data Association



Projective Data Association



Example Data Association

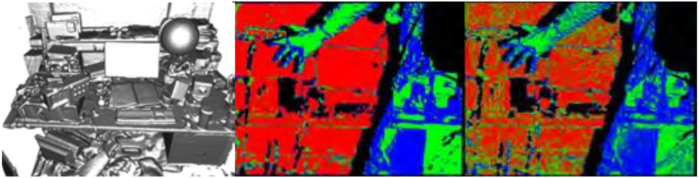
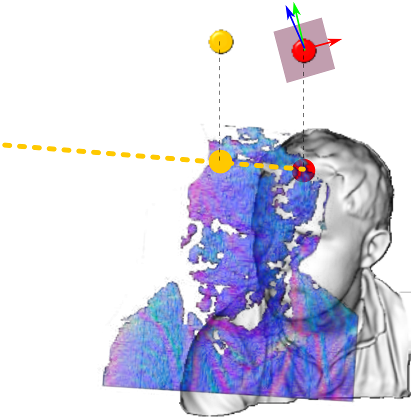
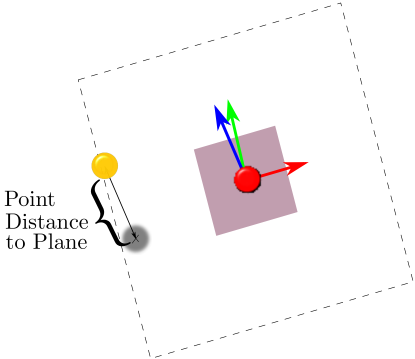


Figure: ICP compatibility testing on the current surface model (Left). *with* bilateral filtering on the vertex/normal map measurement (Middle), using raw vertex/normal map (Right).

Point Plane Metric



Point Plane Metric



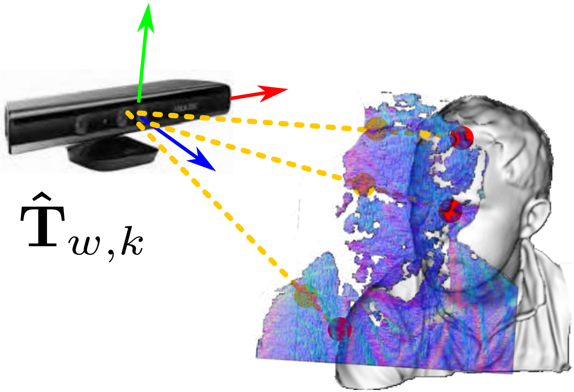
Point-plane metric allows surfaces to *slide* over each other and compliments the projective data-association method.

Tracking as Depth Map to Dense surface alignment

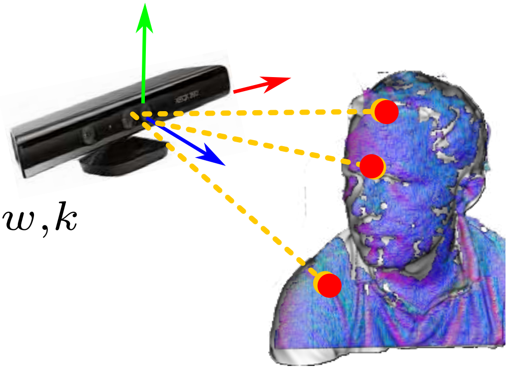
Dense Tracking Algorithm

1. Initialise current pose estimate with previous pose: $\hat{\mathbf{T}}_{k',w} \leftarrow \hat{\mathbf{T}}_{k-1,w}$
2. Compute current surface measurement from depth map R_k
3. Predict surface into estimated previous camera pose $\mathbf{T}_{k-1,w}$
4. Projective data associate vertices from predicted surface with the measured surface using current pose estimate $\hat{\mathbf{T}}_{k',w}$.
5. Find incremental transform $\mathbf{T}_{k,k'}$ that minimises the point-plane metric over the associated surface points.
6. Update current pose estimate $\hat{\mathbf{T}}_{k,w} \leftarrow \mathbf{T}_{k,k'} \hat{\mathbf{T}}_{k',w}$

Minimising the point plane error



Minimising the point plane error



$$\mathrm{T}_{w,k}$$

Table of Contents

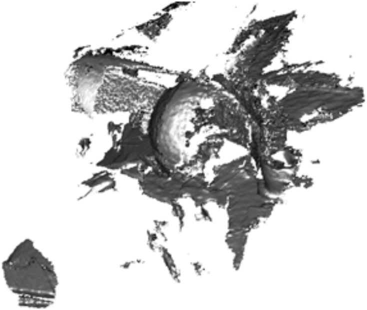
Useful properties

We performed a number of experiments to investigate useful properties of the system.

- Drift free tracking
- Scalable dense tracking and mapping
- Joint tracking/mapping convergence

Frame-Frame vs. Frame-Model Tracking

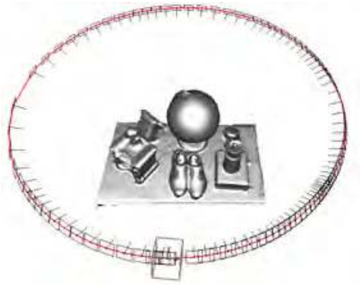
Frame-Frame tracking results in drift as pose errors are continuous integrated into the next frame.

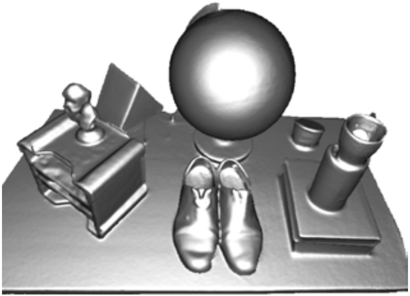


Frame-Frame vs. *Frame-Model* Tracking

Drift Free Tracking with KinectFusion

Frame-Model tracking provides drift free, higher accuracy tracking than Frame-Frame (Scan matching).

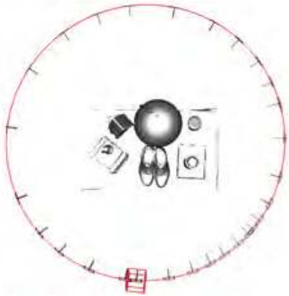




Scalability

Scalability and Robustness

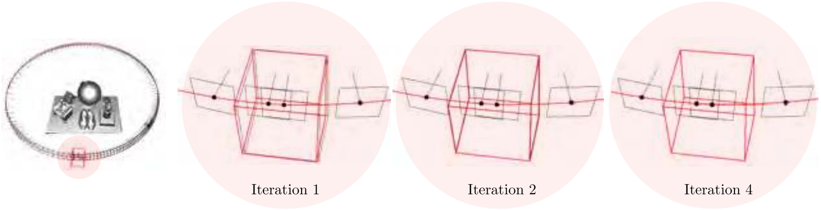
System scales elegantly for limited hardware: frame dropping and reduction in voxel resolution: example $1/64^{th}$ memory and keeping every 6^{th} frame.



Alternating Joint optimisation

Geometry/Tracking Convergence

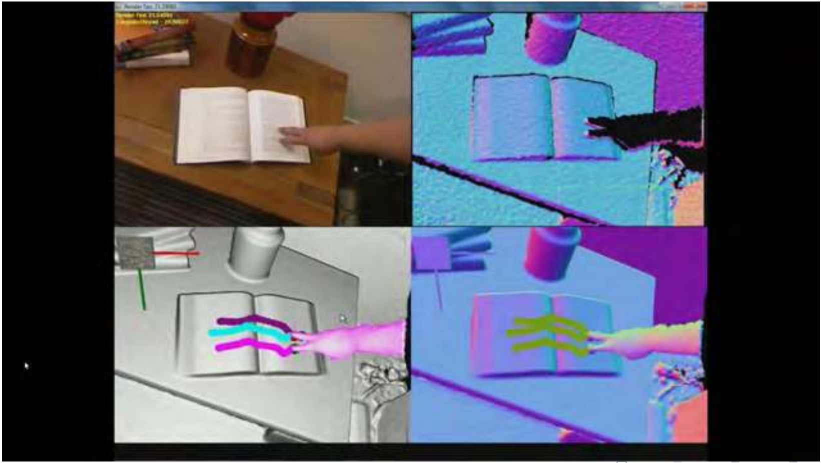
Joint Convergence without explicit joint optimisation. To a minimum of point plane and joint reconstruction error (although the point of convergence may not be the global minimum).



Issues

- Drift is still possible for long exploratory loops as there is no explicit loop closure.
- Sufficient surface geometry required to lock down all degrees of freedom in the point-plane system, e.g. Viewing a single plane leaves 3DOF nullspace.
- Regular grid discretisation of the SDF does not scale for larger spaces. Instead there is a lot of sparsity in the volume that we can exploit using octree style SDF.

A new AR/MR Platform?



Thanks!

Demonstration/Questions?