3D Reconstruction with Kinect & Kinect Fusion

Sema Berkiten
Princeton University
February 2014
3D Sensor

MS Kinect
3D Sensor

MS Kinect

Prime Sense

Capri
(for mobile)
**What is kinect fusion?**

- Using low-cost sensor like MS Kinect (~100$)
What is Kinect Fusion?

- Using low-cost sensor like MS Kinect (~100$)
- 3D object scanning
What is Kinect Fusion?

- Using low-cost sensor like MS Kinect (~100$)
  
  - 3D object scanning
  
  - Model creation
What is Kinect Fusion?

- Using low-cost sensor like MS Kinect (~100$)
  - 3D object scanning
  - model creation
  - interact with the scene
What is Kinect Fusion?

- Using low-cost sensor like MS Kinect (~100$)
- 3D object scanning
- Model creation
- Interact with the scene
- Can be in real time by using GPU
DEMO

Skanect: http://skanect.manctl.com
I. Kinect Fusion

II. Kinect Fusion extension

III. Keyframe based approach

IV. Deformable and non-rigid alignment
I. Kinect Fusion

II. Kinect Fusion extension

III. Keyframe based approach

IV. Deformable and non-rigid alignment
KinectFusion: Real-Time Dynamic 3D Surface Reconstruction and Interaction

Shahram Izadi 1, Richard Newcombe 2, David Kim 1,3, Otmar Hilliges 1, David Molyneaux 1,4, Pushmeet Kohli 1, Jamie Shotton 1, Steve Hodges 1, Dustin Freeman 5, Andrew Davison 2, Andrew Fitzgibbon 1

1 Microsoft Research Cambridge 2 Imperial College London 3 Newcastle University 4 Lancaster University 5 University of Toronto
on a regular grid, we compute, using a cross product between neighbors, to obtain a discontinuity preserved depth map with reduced noise $D_{\text{pixel}}$. We back-project the filtered depth values into the sensor's frame by a rigid body transformation matrix:

$$r_k = \exp(p_k) \cdot r_k$$

The sensor pose estimation includes dehomogenisation to obtain the sensor coordinate frame at time $3$, which maps the camera coordinate frame at time $3$. We represent the live 6DOF camera pose estimated for a frame at $3$.

**3.1 Preliminaries**

A multi-scale ICP alignment between the reference sensor measurement. Our GPU based implementation uses all levels of the depth map pyramid produced, improving the data association required in tracking the scene model maintained with a volumetric, truncated signed distance function, where given the pose determined by tracking the depth data at each image frame, the surface measurement is integrated into the estimated frame to provide a dense surface prediction.

**3.2 Surface Measurement**

Sensor pose estimation is computed for each frame and used to integrate the current surface measurement into the globally fused surface. We compute an average of predicted and measured surface data using a bilateral filter. First a depth map pyramid $D_k$ is computed. Setting the bottom level of the pyramid to the global frame point in the 3D volume to be resolved, each level is stored in global GPU memory where all processing will reside. From here on we assume a fixed bijective mapping between voxel/memory elements and the continuous TSDF representation. The TSDF resides. Subsequently each level in a vertex and normal map pyramid is computed at each image frame, which which are aligned into a global frame is a global surface fusion. In the image domain a multi-scale representation of the surface is stored in global GPU memory where all processing will reside.

**4.0 KINECT FUSION: Summary**

Like frame-to-frame pose estimation as described in Section 3.5, a depth map pyramid is produced, improving the data association required in tracking the scene model maintained with a volumetric, truncated signed distance function, where given the pose determined by tracking the depth data at each image frame, the surface measurement is integrated into the estimated frame to provide a dense surface prediction.

**Figure 3: Overall system workflow.**
Kinect Fusion
[Newcombe et al., 2011]

- Measurement
- ICP (Iterative Closest Point)
- TSDF Integration (Truncated Signed Distance Function)
- Ray Casting
- Measurement
- ICP (Iterative Closest Point)
- TSDF Integration (Truncated Signed Distance Function)
- Ray Casting
Kinect Fusion - Measurement

IZADI et al., 2011

Raw Data → Bilateral Filter → Filtered Data
Kinect Fusion - Measurement
[IZADI ET AL., 2011]

Raw Data → Bilateral Filter → Filtered Data

Vertex Map

[x y depth] in pixels
**Kinect Fusion - Measurement**

[IZADI et al., 2011]

**Raw Data** → **Bilateral Filter** → **Filtered Data**

Vertex Map

\[[x \ y \ \text{depth}] \text{ in pixels}\]

*project to 3D multiply with K*

\[[X \ Y \ Z] \text{ in metrics}\]
Kinect Fusion - Measurement [IZADI et al., 2011]

- Raw Data → Bilateral Filter → Filtered Data
- Vertex Map
- Normal Map
- [x y depth] in pixels
- [X Y Z] in metrics

Figure 2: RGB image of scene (A). Extracted normals (B) and surface reconstruction (C) from a single bilateral filtered Kinect.
Kinect Fusion - Measurement

IZADI et al., 2011

- Raw Data → Bilateral Filter → Filtered Data
- Vertex Map
- Normal Map

[x y depth] in pixels

project to 3D multiply with K

[X Y Z] in metrics

cross product

RELATED WORK

Interactive rates (e.g. the Kinect-based system of...) achieve real-time interactive rates for robot while creating a map of the surrounding physical environment, while simultaneously segmenting, reconstructing camera tracking and reconstruction of a static background scene. We illustrate handheld scanner, and present novel interactive methods for surface-based representation. A novel GPU pipeline allows camera and fuses new viewpoints of the scene into a global representation or support reconstruction in the context of SfM systems (e.g.)... No explicit feature detection, this section is structured... RELATED WORK

7

The remainder of this paper is structured into two parts: The scanning of small physical objects (e.g. recent Kinect RGBD systems e.g.)... One final requirement is to support whole environments (e.g. dynamic scenes. We show how a real-time 3D model can be leveraged for accurate camera tracking and surface reconstruction at interactive rates... Dynamic interaction assumed which more accurately approximate real-world geometry. We explore tracking and interact in front of the camera. While there has been work on the full depth maps acquired from the Kinect sensor. Our system also avoids the reliance on RGB (used in works on the full depth maps acquired from the Kinect sensor)...
Kinect Fusion - Measurement

[IZADI ET AL., 2011]

Raw Data → Bilateral Filter → Filtered Data

Vertex Map

[x y depth] in pixels

project to 3D multiply with K

[X Y Z] in metrics

cross product

Normal Map

N
Kinect Fusion - Measurement
[IZADI ET AL., 2011]

Raw Data → Bilateral Filter → Filtered Data

Data Pyramid to speed-up pose tracking

Kinect Fusion
[Newcombe et al., 2011]

- Measurement
- ICP (Iterative Closest Point)
- TSDF Integration (Truncated Signed Distance Function)
- Ray Casting
Align two partially-overlapping meshes given initial guess for relative transform
If correct correspondences are known, can find correct relative rotation/translation
... and iterate to find alignment

- Iterative Closest Points (ICP)  [Besl & McKay 92]

Converges if starting position “close enough”
ICP - Basic algorithm
by Szymon Rusinkiewicz, 2011

Select e.g. 1000 random points

Match each to closest point on other scan, using data structure such as k-d tree

Reject pairs with distance > k times median

Construct error function:

Minimize (closed form solution in [Horn 87])

\[ E = \sum |Rp_i + t - q_i|^2 \]
Variants on the following stages of ICP have been proposed:

1. **Selecting** source points (from one or both meshes)
2. **Matching** to points in the other mesh
3. **Weighting** the correspondences
4. **Rejecting** certain (outlier) point pairs
5. Assigning an **error metric** to the current transform
6. **Minimizing** the error metric w.r.t. transformation
Using point-to-plane distance instead of point-to-point lets flat regions slide along each other [Chen & Medioni 91]
ICP - POINT-TO-PLANE ERROR
BY KOK-LIM LOW, 2004

destination point
\( d_1 \)

unit normal
\( l_1 \)

source point
\( s_1 \)

tangent plane

\( n_1 \)

\( d_2 \)

\( n_2 \)

\( l_2 \)

\( d_3 \)

\( n_3 \)

\( l_3 \)

source surface

destination surface
Error function:

\[ E = \sum [(R\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i) \cdot \mathbf{n}_i]^2 \]

where \( R \) is a rotation matrix, \( \mathbf{t} \) is translation vector

Linearize (i.e. assume that \( \sin \theta \approx \theta, \cos \theta \approx 1 \)):

\[ E \approx \sum ((\mathbf{p}_i - \mathbf{q}_i) \cdot \mathbf{n}_i + \mathbf{r} \cdot (\mathbf{p}_i \times \mathbf{n}_i) + \mathbf{t} \cdot \mathbf{n}_i)^2, \quad \text{where} \quad \mathbf{r} = \begin{pmatrix} r_x \\ r_y \\ r_z \end{pmatrix} \]

Result: overconstrained linear system
Overconstrained linear system

\[ \mathbf{Ax} = \mathbf{b}, \]

\[
\mathbf{A} = \begin{pmatrix}
\vec{p}_1 \times \mathbf{n}_1 & \vec{p}_1 & \mathbf{n}_1 \\
\vec{p}_2 \times \mathbf{n}_2 & \vec{p}_2 & \mathbf{n}_2 \\
\vdots & \vdots & \vdots
\end{pmatrix},
\quad
\mathbf{x} = \begin{pmatrix}
\mathbf{n}_x \\
\mathbf{n}_y \\
\mathbf{n}_z \\
t_x \\
t_y \\
t_z
\end{pmatrix},
\quad
\mathbf{b} = \begin{pmatrix}
-(\vec{p}_1 \cdot \mathbf{n}_1) \mathbf{n}_1 \\
-(\vec{p}_2 \cdot \mathbf{n}_2) \mathbf{n}_2 \\
\vdots
\end{pmatrix}
\]

Solve using least squares

\[
\mathbf{A}^\mathbf{T} \mathbf{Ax} = \mathbf{A}^\mathbf{T} \mathbf{b}
\]

\[
\mathbf{x} = \left(\mathbf{A}^\mathbf{T} \mathbf{A}\right)^{-1} \mathbf{A}^\mathbf{T} \mathbf{b}
\]
ICP - Data Projection
by Szymon Rusinkiewicz, 2011

Idea: use a simpler algorithm to find correspondences
For range images, can simply project point [Blais 95]

- Constant-time
- Does not require precomputing a spatial data structure
- ICP algorithm with
  - projection-based correspondences,
  - point-to-plane matching

can align meshes in a few tens of ms.
Each iteration is one to two orders of magnitude faster than closest-point.

Result: can align two range images in a few milliseconds, vs. a few seconds.
Frame-to-Model Alignment

- Frame-to-frame alignment:
  - Unstable
  - Prune to incremental drift
Frame-to-Model Alignment

- Frame-to-frame alignment:
  - Unstable
  - Prune to incremental drift

- Frame-to-model alignment:
  - Inherently prevents drift
  - More stable
Kinect Fusion

[Newcombe et al., 2011]

- Measurement
- ICP (Iterative Closest Point)
- TSDF Integration (Truncated Signed Distance Function)
- Ray Casting
SDF : signed distance function

- Distance from a point \( x \) to the boundary of a region
- Sign determined by whether \( x \) is in region
Update only the close voxels to the boundary
TSDF: truncated signed distance function

Update only the close voxels to the boundary

http://graphics.stanford.edu/papers/volarep/paper_2_levels/node3.html
Figure 8 - How a TSDF is created

The resulting TSDF is used to reconstruct the surface of the current, refined model. To do so, ray casting is performed. Ray casting is a surface visualization method used in computer graphics. It consists of choosing a point of view, the camera’s focal point, and casting rays in the scene from it. A ray is basically a traveling line, it starts from the point of view with zero length and travels in a chosen direction until it intersects something. When the ray intersects a surface, the pixel it corresponds to in the image space of the camera is assigned to the 3D intersection point. This makes ray casting, with its output-centric focus, different from typical object-centered visualization methods such as vertex and segment rendering. Ray casting is often used in the visualization of translucent and semi-transparent objects, since it is easy to perform reflection and refraction by changing the direction of the traveling rays. It is also considered one of the better 3D visualization methods, resulting in high-quality images. Ray casting is also used for physics checks in virtual reality environments, as well for other distance-based computations such as 3D user-picking.

In Kinect Fusion ray casting is used for this surface reconstruction step. Ray casting is performed from the global camera focal point by intersecting the zero level set of the TSDF. A prediction of the current global surface is obtained, with vertex data and estimated normal data. The refined, ray-casted model is the one that will be used in the next ICP step for alignment. By doing so instead of using just the last frame point cloud as the source for alignment, a less noisy model is obtained, placing the focus on finding a smooth surface. The process is sped up by using ray skipping, that is by advancing the rays at discrete steps. Ray casting is also used for the final visualization step. The end result is a 3D surface representing the acquired scene (see Figure 2).

Conclusion and notes

Kinect Fusion allows reconstruction of 3D scenes in real-time with ease thanks to its assumptions and the heavy parallelization. However, it still presents a few issues. The algorithm requires high computational power and a powerful GPU to work, due to the assumption of small movements between two frames. In addition, even at a high frame rate, the motion of the Kinect sensor must be kept in check, as a sudden tilt or translation may break the assumption. The consequence of breaking the assumption is that the ICP alignment may not converge to a correct match. In this case, the algorithm will discard the new cloud and fall back to an older pose. However, as we have experienced with the KinFu implementation (see the next section), a different problem may arise. Due to sudden movements and thus due to the breaking of the assumption, the ICP alignment step may erroneously converge to a wrong match. In this case, the algorithm is not able to detect that tracking has been lost and instead will merge two different clouds together, breaking both the tracking and the reconstruction. We found that the major reason of loss of tracking in the scene was the...
Representing the surface in memory

- Truncated Surface Distance Function (TSDF)
  - Makes handheld scanning on personal computers feasible
    - Faster than other high-accuracy methods
    - Allows for continuous refinement of the model
  - When scanning, a (usually cubic) virtual volume is defined. The real-world target object is reconstructed within this volume.
  - The volume is subdivided into a grid of many smaller cubes, called voxels.
  - Voxels are volumetric picture elements.
TSDF representation

- Each voxel is assigned a distance to the surface
- Negative is behind, positive is in front
- Each distance is also assigned a weight (not shown)
- Weights represent an estimate of accuracy for a voxel’s distance
- For example: a surface facing the camera is likely more accurate than a surface at an angle, so those measurements are given a higher weight
Step 6: Calculating TSDF Values

- Line from each vertex to camera through voxel grid.
- Intersected voxels near the surface are updated.
- Distance from vertex to voxel center is distance value for a given voxel.
- Assign a weight for the measurement based on the voxel orientation.
- Use the measurement weight and distance to update the voxel’s current value.
- Repeat for other intersected voxels.
- Repeat for each vertex.
TSDF Refinement

As the camera moves around and captures more vertices, the TSDF is continuously updated and refined.

Since TSDF voxels contain distances and are not just representative of the edge of a surface, the surface is represented far more accurately than the actual voxel resolution.
Each voxel stores:

- \( F \): signed distance
- \( W \): accumulated weight

Moving average at time \( k \):

\[
F_k(p) = \frac{W_{k-1}(p)F_{k-1}(p) + W_{R_k}(p)F_{R_k}(p)}{W_{k-1}(p) + W_{R_k}(p)}
\]

\[
W_k(p) = W_{k-1}(p) + W_{R_k}(p)
\]
Figure 4: A slice through the truncated signed distance volume showing the truncated function $F > \mu$ (white), the smooth distance field around the surface interface $F = 0$ and voxels that have not yet had a valid measurement (grey) as detailed in eqn. 9.
Kinect Fusion
[Newcombe et al., 2011]

- Measurement
- ICP (Iterative Closest Point)
- TSDF Integration (Truncated Signed Distance Function)
- Ray Casting
Ray casting [Newcombe et al., 2011]

- Cast a ray from each pixel in the image through the focal point of the virtual camera
RAY casting

- Find “zero-crossings” in tsdf volume
- p: extracted grid position by trilinear interpolation
RAY CASTING

- Find “zero-crossings” in tsdf volume
- \( p \): extracted grid position by trilinear interpolation
- Uniform sampling for voxel traversal
RAY casting

- Find “zero-crossings” in tsdf volume
- p: extracted grid position by trilinear interpolation
- Uniform sampling for voxel traversal

Uniform sampling
RAY casting

- Find “zero-crossings” in tsdf volume
- p: extracted grid position by trilinear interpolation
- DDA for fast voxel traversal [Amanatides&Woo’87]
RAY casting

- Find “zero-crossings” in tsdf volume
- \( p \): extracted grid position by trilinear interpolation
- \( v \): convert grid position \( p \) to 3D global position
- \( n \): surface gradient at \( p \)
RAY casting

Ray-casting on the TSDF

Ray-casting on the TSDF

Surface

Normals
KINECT Fusion: RECAP
[IZADI ET AL., 2011]
At the end of the day:
Model Extraction

Marching Cubes on TSDF
I. Kinect Fusion

II. Kinect Fusion extensions

III. Keyframe based approach

IV. Deformable and non-rigid alignment
KinectFusion extensions

- Problems with Kinect Fusion:
  - Limited integration volume size
Kinect Fusion extensions

- Problems with Kinect Fusion:
  - Limited integration volume size
  - Loop closure

http://www.ccs.neu.edu/research/gpc/
http://robotics.usc.edu/~ahoward/projects/eadme.php
KinectFusion extensions

- Problems with Kinect Fusion:
  - Limited integration volume size
  - Loop closure
  - Texturing

http://www.ccs.neu.edu/research/gpc/
KinectFusion extensions

- Circular buffer for space extension [Kintinuous’12]
- Loop closure [Kintinuous Loop Closure ’13]
- Texturing [3DSelfPortrait’13]
- Large Scale [LargeScale KinectFusion’13]
KinectFusion extensions

- Circular buffer for space extention [Kintinuous’12]
- Loop closure [Kintinuous Loop Closure ’13]
- Texturing [3DSelfPortrait’13]
- Large Scale [LargeScale KinectFusion’13]
- Using circular buffer
- Moving TSDF volume
- Real-time surface extraction & triangulation
- Store in the system memory not in GPU
CIRCULAR (RING) BUFFER

- Data structure
  - uses a single, fixed-size buffer
  - virtually connected end-to-end
Circular buffer makes space-extended scanning possible.

Fig. 2. The four main steps of the Kintinuous algorithm are shown above; (i) Camera motion exceeds movement threshold (black arrow); (ii) Volume slice (red) is raycast (orthogonal directions shown in blue arrows) for point extraction and reset; (iii) Point cloud extracted and fed into greedy mesh triangulation algorithm [19]; (iv) New spatial region enters volume (blue).

Whelan et. al, 2013
Kintinuous: Spatially Extended Kinect Fusion

Thomas Whelan, John McDonald
National University of Ireland Maynooth, Ireland

Michael Kaess, Maurice Fallon, Hordur Johannsson, John J. Leonard
Computer Science and Artificial Intelligence Laboratory, MIT, USA
KinectFusion extensions

- Circular buffer for space extension [Kintinuous’12]
- Loop closure [Kintinuous Loop Closure ’13]
- Texturing [3DSelfPortrait’13]
- Large Scale [LargeScale KinectFusion’13]
What happens in loop closure?

How to correct accumulated registration error?
Kintinuous Loop Closure
Whelan et. al, 2013

Kintinuous 2.0
Real-time large scale dense loop closure with volumetric fusion mapping

Thomas Whelan*, Michael Kaess’, John J. Leonard’, John McDonald*

* Computer Science Department, NUI Maynooth
' Computer Science and Artificial Intelligence Laboratory, MIT
- Place Recognition
- SURF: Speeded Up Robust Feature
- Bag of Words (DBoW)
- Strict validation to eliminate outliers
- ICP
- Space Deformation
- Deformation graph
- Space Deformation
- Deformation graph

Before

After
→ Space Deformation

→ \( w_{rot} E_{rot} + w_{reg} E_{reg} + w_{con_P} E_{con_P} + w_{surf} E_{surf} \)
Space Deformation
\[ w_{rot} E_{rot} + w_{reg} E_{reg} + w_{con_P} E_{con_P} + w_{surf} E_{surf} \]

Map deformation considering:
- rigidity, \( E_{rot} = \sum_l \|N_{lR}^T N_{lR} - I\|_F^2 \)
KINTINIOUS LOOP CLOSURE
Whelan et. al, 2013

- Space Deformation
  \[ w_{\text{rot}} E_{\text{rot}} + w_{\text{reg}} E_{\text{reg}} + w_{\text{con}} E_{\text{con}} + w_{\text{surf}} E_{\text{surf}} \]

- Map deformation considering:
  - rigidity,
  - regularization for smooth deformation across the graph,

  \[ E_{\text{reg}} = \sum_{r} \sum_{n \in \mathcal{N}(N_i)} \left\| N_{i_r}(N_{ng} - N_{ig}) + N_{ig} + N_{it} - (N_{ng} + N_{nt}) \right\|_2^2 \]
Space Deformation

\[ w_{rot} E_{rot} + w_{reg} E_{reg} + w_{con_p} E_{con_p} + w_{surf} E_{surf} \]

- Map deformation considering:
  - rigidity,
  - regularization for smooth deformation across the graph,
  - user constraints,

\[ E_{con} = \sum_p \| \phi(v) - U_p \|_2^2 \]
Space Deformation

\[ w_{rot}E_{rot} + w_{reg}E_{reg} + w_{conP}E_{conP} + w_{surf}E_{surf} \]

- Map deformation considering:
  - rigidity,
  - regularization for smooth deformation across the graph,
  - user constraints,
  - surface orientation

\[ E_{surf} = \sum_{q} \| \phi((P_{iR}V_q) + P_t) - ((P'_{iR}V_q) + P'_t) \|_2^2 \]
KINTINIOUS LOOP CLOSURE
Whelan et. al, 2013

Fig. 8. Large indoor and outdoor dataset made up of over five million vertices. Insets show the high fidelity of small scale features in the map.
KinectFusion extensions

- Circular buffer for space extension [Kintinuous’12]
- Loop closure [Kintinuous Loop Closure ’13]
- Texturing [3DSelfPortrait’13]
- Large Scale [LargeScale KinectFusion’13]
3D SELF PORTRAITS
LI ET. AL, 2013

Texturing

Direct Mapping
3D Self Portraits
Li et. al, 2013

- Texturing by
  - Albedo recovery (variation of SIRFS: shape, reflection, and illumination recovery from shading)

Direct Mapping

Image of a person playing the guitar.
Abstract
We address the problem of recovering shape, albedo, and illumination from a single grayscale image of an object, using shading as our primary cue. Because this problem is fundamentally underconstrained, we construct statistical models of albedo and shape, and define an optimization problem that searches for the most likely explanation of a single image. We present two priors on albedo which encourage local smoothness and global sparsity, and three priors on shape which encourage flatness, outward-facing orientation at the occluding contour, and local smoothness. We present an optimization technique for using these priors to recover shape, albedo, and a spherical harmonic model of illumination. Our model, which we call SAIFS (shape, albedo, and illumination from shading) produces reasonable results on arbitrary grayscale images taken in the real world, and outperforms all previous grayscale "intrinsic image"-style algorithms on the MIT Intrinsic Images dataset.

1. Introduction
We wish to take only a single grayscale image of an object and estimate the shape, albedo, and illumination that produced that image (Figure 1). This "inverse optics" problem is terribly underconstrained: the space of albedos, shapes, and illumination that reproduce an image is vast. But of course, not all albedos and shapes are equally likely. Past work has demonstrated that simple statistics govern natural images [8, 23], and we will construct models of the similar statistics that can be found in natural albedo and shape. Our algorithm is simply an optimization problem in which we recover the most likely shape, albedo, and illumination under to our statistical model, such that a single image is exactly reproduced. Our priors are effective enough that shape, albedo, and illumination can be recovered from real-world images, and are general enough that they work across a variety of objects: a single model learned on teabags and squirrels can be applied to images of coffee cups and turtles. Our model can be seen in Figures 1, 2.
3D Self Portraits
Li et al, 2013

- Texturing by
  - Albedo recovery (variation of SIRFS: shape, reflection, and illumination recovery from shading)

Direct Mapping

SIRFS

Monday, February 17, 14
3D Self Portraits
Li et al, 2013

- Texturing by
  - Albedo recovery (variation of SIRFS: shape, reflection, and illumination recovery from shading)
  - Poisson Blending

Direct Mapping  SIRFS  SIRFS + Poisson Blending

Images showing before and after texturing results.
Texturing by

- Albedo recovery (variation of SIRFS: shape, reflection, and illumination recovery from shading)
- Poisson Blending

Direct Mapping  SIRFS  SIRFS + Poisson Blending
KinectFusion extensions

- Circular buffer for space exention [Kintinuous’12]
- Loop closure [Kintinuous Loop Closure ’13]
- Texturing [3DSelfPortrait’13]
- Scalable [LargeScale KinectFusion’13]
Scalable
Chen et. al, 2013

Same reconstruction algorithm to obtain fine details
Same reconstruction algorithm

**Figure 2:** High level 3D reconstruction pipeline.
Same reconstruction algorithm

With a smart hierarchical data structure

Figure 2: High level 3D reconstruction pipeline.

Figure 3: Logical view of hierarchical data structure.
Scalable
Chen et. al, 2013

regular grid (12mm$^3$),
512$^3$, 512 MB

hierarchy (3mm$^3$)
2048$^3$, 174 MB
We address the fundamental challenge of volumetric surface reconstruction methods. We design a memory-efficient, hierarchical data structure for commodity graphics hardware. This structure runs entirely in RAM, allowing for unbounded reconstructions. Our pipeline, which fuses depth maps from a moving depth camera into a single volumetric representation particularly for cultural heritage, special effects, gaming, and computer graphics and computer vision, with many practical applications.

**Keywords:**
- Generation—Digitizing and scanning;
- Scalable;
- Real-time;
- Volumetric;
- Fusion;
- Surface extraction;
- Streaming;
- Out-of-core;
- Adapting;
- Multiresolution;
- Interactive;
- Quality;
- Memory;
- Speed;
- Future work;
- Limitations;
- Limitations in mobility (our current system relies on having a powerful server within wireless range) and exploring the use of our data structure for real-world needs to be combined live with the virtual and rendered environments particularly for cultural heritage, special effects, gaming, and computer graphics and computer vision, with many practical applications.

**Abstract**

We make real-time depth sensing a commodity. This has naturally led to an explosion of applications such as indoor mapping, participating in augmented reality, and exploring the use of our data structure for real-world needs to be combined live with the virtual and rendered environments particularly for cultural heritage, special effects, gaming, and computer graphics and computer vision, with many practical applications.

**References**

[Chen et al. 2013] has 12mm voxels which have minimal error and we are able to close the loop. Track-model tracking. In another comparison, at the same physical extent (6m) necessary to capture an office scene, a regular grid (12mm), 512³, 512 MB leaves unobserved regions empty. Our approach permits a much larger active region compared to a regular grid. We compare an implementation of our data structure to the method of Curless and Levoy and demonstrated compelling live reconstructions from noisy Kinect depth maps, which were applied in long “forward-only” paths such as exploring the use of our data structure for real-world needs to be combined live with the virtual and rendered environments particularly for cultural heritage, special effects, gaming, and computer graphics and computer vision, with many practical applications.

**7 Summary**

We thank Mat Cook, Andreas Georgiou, Steven Johnston, James Newcombe et al. for insightful discussions, and Emily Whiting and Jaakko Lehtinen for GPU programming wizardry; Christopher Henze, Dionissios Manatides, and Iason Oikonomidis for insights and support this level of scalability, we design a fast, compact volumetric reconstruction method which fuses depth maps from a moving depth camera into a single volumetric representation particularly for cultural heritage, special effects, gaming, and computer graphics and computer vision, with many practical applications.

**6.4 Limitations and future work**

In mobility (our current system relies on having a powerful server within wireless range) and exploring the use of our data structure for real-world needs to be combined live with the virtual and rendered environments particularly for cultural heritage, special effects, gaming, and computer graphics and computer vision, with many practical applications.

**512³, 512 MB hierarchy (3mm³)**

2048³, 174 MB
I. Kinect Fusion

II. Kinect Fusion extension

III. Keyframe based approach:

IV. Deformable and non-rigid alignment
KEYFRAME BASED METHODS

- Super-Resolution 3D Tracking and Mapping.
- On unifying key-frame and voxel-based dense visual SLAM at large scales.
- 3D High Dynamic Range dense visual SLAM and its application to real-time object re-lighting.
KEYFRAME BASED METHODS

- Super-Resolution 3D Tracking and Mapping.
- On unifying key-frame and voxel-based dense visual SLAM at large scales.
- 3D High Dynamic Range dense visual SLAM and its application to real-time object re-lighting.
- Low resolution images

- warped, combined -> to estimate high-resolution target

The image degradation pipeline (forward compositional). One or more low resolution images are transformed to a common reference frame. The images are up-sampled and then inverse blurred to estimate high-resolution images. Classical modern optimization approaches with image priors.

The main contribution of this paper is based on how the LR images are combined to form a SR image. Classic SR methods distill the LR images and use motion models for warping and up-sampling. Our approach is based on the idea of inverse shading. The relationship between the SR and the LR image is then obtained by the following homographic combinations:

\[ \begin{align*}
\tilde{I}(t) &= w(T_t, V^*; \mathcal{K}) \\
I^w(t) &= w(S) \\
\beta(B^{-1}) &= \text{Inverse Blur} \\
I^* &= \text{High Resolution Depth and color images}
\end{align*} \]

\[ I(t) \rightarrow \text{Geometric/photometric warping} \rightarrow \text{Upsampling} \rightarrow \text{Inverse Blur} \rightarrow I^* \]
Super Resolution SLAM
Meilland et. al, 2013

- High resolution images
- to estimate 6D pose and 3D mapping
3D Tracking:

- Optimizing both photometric and depth data:

\[
\begin{align*}
\mathbf{e}_I &= \sum_{t=0}^{N} \mathbf{C}(t) \left( \mathbf{I}^* - \mathbf{I} \left( w \left( \hat{T}_t \mathbf{T}(\mathbf{x}_t), \mathbf{V}^*, \mathbf{S} \right), t \right) \right) \\
\mathbf{e}_D &= \sum_{t=0}^{N} \tilde{\mathbf{C}}(t) \left( \mathbf{D}^* - \mathbf{D} \left( w \left( \hat{T}_t \mathbf{T}(\mathbf{x}_t), \mathbf{V}^*, \mathbf{S} \right), t \right) \right)
\end{align*}
\]

- Ability to keep tracking when
  - depth camera is totally occluded,
  - too close to the scene
- Weighting:

- Image distance function:

\[ C_{ii}(t) = (\| (T_t - T_o v_i^* \| + \epsilon)^{-1} \]

The current frame
Weighting:

- Image distance function:

\[ C_{ii}(t) = \left( \| (T_t - T_o) \vec{v}_i^* \| + \epsilon \right)^{-1} \]

The current frame

- Depth weights:

\[ \tilde{C}_{ii}(t) = \frac{fb}{\sigma_d} D(p_i, t)^{-2} \]
Super-Resolution 3D Tracking and Mapping

Maxime Meillard and Andrew L. Comport
KEYFRAME BASED METHODS

- Super-Resolution 3D Tracking and Mapping.

- On unifying key-frame and voxel-based dense visual SLAM at large scales.

- 3D High Dynamic Range dense visual SLAM and its application to real-time object re-lighting.
Combines advantages of two methods:

- Voxel grid representation
- Image-based key-frame based
**Key Frame Fusion**  
*Meilland et. al, 2013*

- Voxel grid representation  
  - Easy and efficient defining mathematical operations  
  - Global noise reduction  
  - Hard to handle loop closure  
  - Requires a lot of memory

- Image-based key-frame based  
  - Hard  
  - Not inherent  
  - Easy to handle drifts and loop closure  
  - Requires less memory
Unified system:

- Key Frame Fusion
  Meilland et. al, 2013

- 3D Model
  Key-frames/Voxel-grid

- Prediction
  $I^*, V^*$

- Estimation
  $x = (\hat{T}, \hat{I}, \hat{V})^T$

- Learning/Update

- RGB-D Sensor
  $I, V$

- Warping
  $I^w, V^w$
KEY FRAME FUSION
MEILLAND ET. AL, 2013

- Pose Estimation:
  - ICP + photometric constraint

\[
e(x) = \begin{bmatrix}
  I \left( w(\hat{T}T(x), V^*) \right) - I^*(P^*) \\
  \hat{R}R(x)N^{*\top} \left( V^\top - \Pi\hat{T}T(x)V^{*\top} \right)^\top
\end{bmatrix}
\]
 Pose Estimation:

- ICP + photometric constraint

\[
\begin{align*}
e(x) &= \begin{bmatrix}
I \left( w(\hat{T}T(x), V^*) \right) - I^*(P^*) \\
\hat{R}R(x)N^* \left( V^T - \Pi \hat{T}T(x)V^* \right)^T
\end{bmatrix} \\
& \text{Nonlinear! (Gauss Newton)}
\end{align*}
\]
Key Frame Fusion
Meillard et. al, 2013

- Multi-frame fusion:
  - Find closest $M$ frames
  - Rasterize each frame and blend from current view

$$S^* = \sum_{i=1}^{M} f \left( S \left( \Gamma (P^*, E, w(V_i, \hat{T}^{-1}T_i, K^* )) \right) \right)$$

- Integrate as in Super Resolution SLAM
On unifying key-frame and voxel-based dense visual SLAM at large scales

With local bundle adjustment, large scale loop closure and high dynamic range mapping

Maxime Meillard and Andrew I. Comport
KEYFRAME BASED METHODS

- Super-Resolution 3D Tracking and Mapping.
- On unifying key-frame and voxel-based dense visual SLAM at large scales.
- 3D High Dynamic Range dense visual SLAM and its application to real-time object re-lighting.
Main Contributions:

- Registering HDR images to 3D scene
- Real-time HDR reflections on virtual objects
- High Dynamic range registration

- LDR:

\[
\mathbf{e}(\mathbf{x})_{ldr} = \left[ \mathbf{I} \left( w(\hat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \mathbf{V}^*) \right) - \mathbf{I}^* (\mathbf{P}^*) \right]
\]
High Dynamic range registration

- LDR:

\[ e(x)_{ldr} = \left[ I \left( w(\hat{TT}(x), V^*) \right) - I^*(P^*) \right] \]

- HDR:

\[ e(x)_{hdr} = \left[ I \left( w(\hat{TT}(x), V^*) \right) - (\hat{\alpha} + \alpha)I^*_{hdr}(P^*) \right] \]
HDR SLAM
Meilland et. al, 2013

- High Dynamic range registration

- LDR:

\[ e(x)_{ldr} = \left[ I \left( w(\hat{T}T(x), V^*) \right) - I^*(P^*) \right] \]

- HDR:

\[ e(x)_{hdr} = \left[ I \left( w(\hat{T}T(x), V^*) \right) - (\hat{\alpha} + \alpha)I^*_{hdr}(P^*) \right] \]

Unknown shutter estimate

Solve \([x \alpha]\): Nonlinear - Gauss-Newton!
3D High dynamic range mapping:

- Key-frame image: $I^*_{\text{hdr}}$
- Cumulative weights: $C^*_{\text{hdr}}$
- Updated rule between time $t - 1$ and time $t$:

$$C^*_{\text{hdr}}(p, t) \leftarrow C^*_{\text{hdr}}(p, t - 1) + f(I^w(p, t))$$

$$I^*_{\text{hdr}}(p, t) \leftarrow \frac{f(I^w(p, t))I^w_{\text{hdr}}(p, t) + C^*_{\text{hdr}}(p, t - 1)I^*_{\text{hdr}}(p, t - 1)}{C^*_{\text{hdr}}(p, t)}$$
Real-time HDR reflections on virtual objects

- HDR SLAM
  - Meillard et al., 2013

Diagram:
- Cube maps generation
  - Sec. 3.1.1
  - Input: HDR key-frame based 3D model
- Light sources detection
  - Sec. 3.1.2
  - Output: HDR reflection maps
  - Input: Virtual objects world positions
  - Recursive rendering (inter-object reflections)
- Shadow-maps pre-computation
  - Sec. 3.1.3
  - Input: 3D lights positions
  - Output: Light depth-maps

HDR reflection maps
- Object 1
- Object 2
- Object 3

Light depth-maps
- Light 1
- Light 2
- Light 3

HDR reflection maps
- 3D lights positions
- Lights depth-maps
3D High Dynamic Range Dense Visual SLAM and its application to real-time object re-lighting

ISMAR 2013

Maxime Meilland, Christian Barat and Andrew I. Comport
I. Kinect Fusion

II. Kinect Fusion extension

III. Keyframe based approach:

IV. Deformable and non-rigid alignment
Deformable & non-rigid alignment

- Robust Single-View Geometry And Motion Reconstruction.

- 3D Self-Portraits.

- Elastic Fragments for Dense Scene Reconstruction.
Deformable & non-rigid alignment

- Robust Single-View Geometry And Motion Reconstruction.
- 3D Self-Portraits.
- Elastic Fragments for Dense Scene Reconstruction.
Robust Single-View Geometry and Motion Reconstruction
Li et. al, 2009

Reconstruction from a dynamic scene
Pipeline:

1. Static Acquisition
2. Dynamic Acquisition
3. Partial Scans
4. Detail Estimation
5. Non-Rigid Registration
6. Detail Coefficients
7. Detail Aggregation
8. Reconstructed Shape
Align the template to the each frame by creating deformation graph.
Robust Single-View Geometry and Motion Reconstruction

Li et. al, 2009

Aggregating detail over temporally adjacent frames propagates detail
Deformable & non-rigid alignment

- Robust Single-View Geometry And Motion Reconstruction.
- 3D Self-Portraits.
- Elastic Fragments for Dense Scene Reconstruction.
Both portions are described in the next section. Online processing requires no user intervention. The details of this online portion of our algorithm, illustrated in Figure 3, take about two minutes total. It is followed by about 12 minutes of offline processing, requiring the user to stand as still as possible for about four seconds until the sensor signals the beginning of the acquisition process and warns the user to stand as still as possible for about four seconds. The user is given five seconds to turn roughly 45 degrees clockwise. A second sound notifies the user that capture of the current view is complete, and the user is given five seconds to turn roughly 45 degrees clockwise again. This process is repeated about eight times, until a 360-degree capture of the subject is completed. This online portion of our algorithm makes minimal prior assumptions about the subject’s shape or structure priors, and thus requires minimal user input.

Approach to body capture that makes minimal prior assumptions about the subject’s shape or structure priors. Our system takes a similar approach to body capture that makes minimal prior assumptions about the subject’s shape or structure priors. This approach to body capture is purely geometric and thus reliable for textureless subjects and does not involve human shape priors. Moreover, our non-rigid alignment techniques based on the multiview registration framework of Pulli work well for subjects standing or lying on the ground, allowing for the acquisition of data from different viewpoints. We review the relevant recent advances in 3D shape reconstruction methods for static objects. Using rigid registration techniques for human capture registration techniques have been introduced to handle different types of deformations such as quasi-articulated motions. Due to the potential complexity of the deformations, these methods generally require good coverage, e.g., multiple input scans. To align scans of deformable subjects, pairwise non-rigid registration techniques have been introduced to handle different types of deformations. Because these techniques generally require good coverage, e.g., multiple input scans, they are not suitable for textureless subjects or for acquisition of data from different viewpoints. Instead, we use a purely geometric alignment technique that is robust to noise and can handle arbitrary poses and props.

Registration algorithms for capturing multiple body views. A number of different multi-view registration algorithms have been introduced to handle different types of deformations. These algorithms generally require good coverage, e.g., multiple input scans. However, these methods are not suitable for textureless subjects or for acquisition of data from different viewpoints. Instead, we use a purely geometric alignment technique that is robust to noise and can handle arbitrary poses and props. We review the relevant recent advances in 3D shape reconstruction methods for deformable objects. Using rigid registration techniques for human capture registration techniques have been introduced to handle different types of deformations such as quasi-articulated motions. Due to the potential complexity of the deformations, these methods generally require good coverage, e.g., multiple input scans. To handle the loop closure problem in a real-time 3D scanning setting, we rely on texture features for matching.

3.2 Alignment.

Pipeline:

1. Raw input data
2. Per-view fusion
3. Rigid alignment
4. Non-rigid alignment
5. Output reconstruction
6. Textured reconstruction
7. 3D print
Aggregation & per-view fusion of data

For ~8 views

per-view aggregation of raw frames  per-view fusion
After initial rigid ICP

Nonrigid ICP (solves loop closure as well)
Deformable & non-rigid alignment

- Robust Single-View Geometry And Motion Reconstruction.
- 3D Self-Portraits.
- Elastic Fragments for Dense Scene Reconstruction.
Elastic Fragments for Dense Scene Reconstruction
zhou et. al, 2013

Handles the deformations caused by:
High-frequency errors,
Low-frequency distortion

(a) Extended KinectFusion
(b) Zhou and Koltun
(c) Our approach
Elastic Fragments for Dense Scene Reconstruction
Zhou et. al, 2013

- Fragment construction
- Initial alignment
- Elastic Registration
- Integration
- Fragment construction

- Partition into $k(50 \text{ or } 100)$ segments

- Use frame-to-model registration & integration as in the kinect fusion
Elastic Fragments for Dense Scene Reconstruction

- Initial alignment between fragments
- SLAM for rough alignment
- ICP stability to validate correspondences
Elastic Registration by iteratively optimizing:

- Distances between fragments
- Elastic strain energy to preserve shape of each fragment
Elastic Fragments for Dense Scene Reconstruction

Zhou et al., 2013

Comparison to rigid methods

(a) Extended KinectFusion
(b) Zhou and Koltun
(c) Mocap trajectory
(d) Our approach
REFERENCES

Raycasting
[Izadi et al., 2011]

Listing 3 Raycasting to extract the implicit surface, composite virtual 3D graphics, and perform lighting operations.

```plaintext
for each pixel \( u \in \text{output image} \) in parallel do
    \( \text{ray}^{\text{start}} \leftarrow \text{back project } [u, 0]; \text{convert to grid pos} \)
    \( \text{ray}^{\text{next}} \leftarrow \text{back project } [u, 1]; \text{convert to grid pos} \)
    \( \text{ray}^{\text{dir}} \leftarrow \text{normalize } (\text{ray}^{\text{next}} - \text{ray}^{\text{start}}) \)
    \( \text{ray}^{\text{len}} \leftarrow 0 \)
    \( g \leftarrow \text{first voxel along } \text{ray}^{\text{dir}} \)
    \( m \leftarrow \text{convert global mesh vertex to grid pos} \)
    \( m^{\text{dist}} \leftarrow ||\text{ray}^{\text{start}} - m|| \)
while voxel \( g \) within volume bounds do
    \( \text{ray}^{\text{len}} \leftarrow \text{ray}^{\text{len}} + 1 \)
    \( \text{g}^{\text{prev}} \leftarrow g \)
    \( g \leftarrow \text{traverse next voxel along } \text{ray}^{\text{dir}} \)
if zero crossing from \( g \) to \( g^{\text{prev}} \) then
    \( p \leftarrow \text{extract trilinear interpolated grid position} \)
    \( v \leftarrow \text{convert } p \text{ from grid to global 3D position} \)
    \( n \leftarrow \text{extract surface gradient as } \nabla \text{tsdf} (p) \)
    shade pixel for oriented point \((v, n)\) or
    follow secondary ray (shadows, reflections, etc)
if \( \text{ray}^{\text{len}} > m^{\text{dist}} \) then
    shade pixel using inputed mesh maps or
    follow secondary ray (shadows, reflections, etc)
```

Our rendering pipeline shown in Figure [raycasting for rendering and tracking](#).

Raycasting for Rendering and Tracking
[izadi et al., 2011]