3D Reconstruction with Kinect & Kinect Fusion

Sema Berkiten
Princeton University
February 2014

3D Sensor

MS Kinect

3D Sensor

MS Kinect

Prime Sense

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

OUTLINE

I. Kinect Fusion

II. Kinect Fusion extension

III. Keyframe based approach

IV. Deformable and non-rigid alignment

OUTLINE

I. Kinect Fusion

II. Kinect Fusion extension

III. Keyframe based approach

IV. Deformable and non-rigid alignment

KINECT FUSION SUMMARY

[IZADI ET AL., 2011]

SIGGRAPH Talks 2011

KinectFusion:

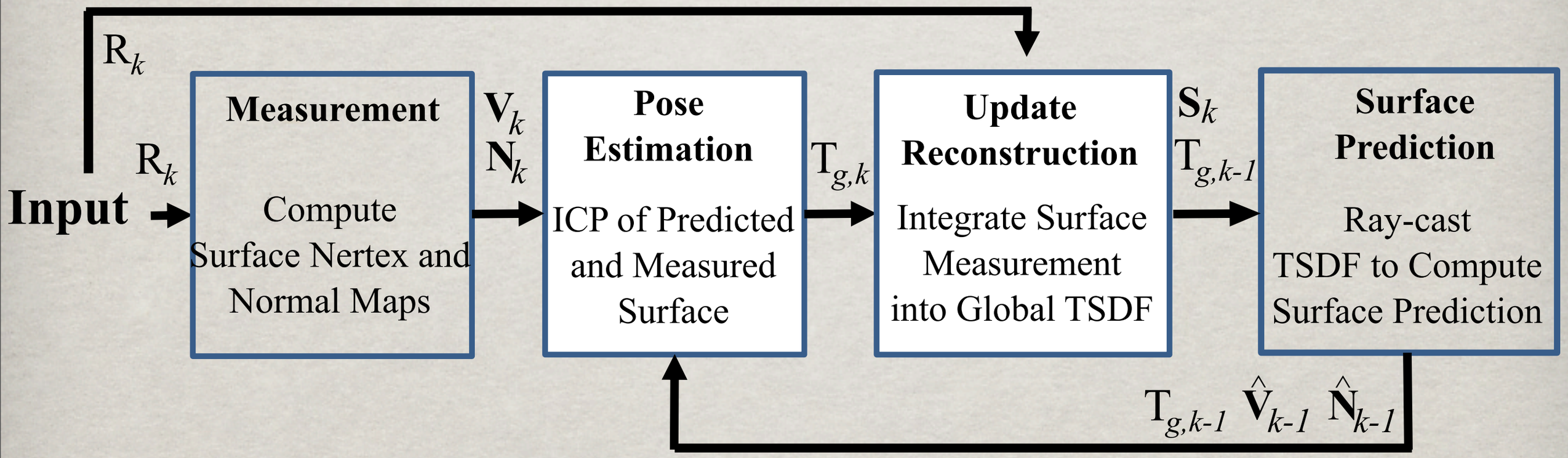
Real-Time Dynamic 3D Surface Reconstruction and Interaction

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

KINECT FUSION: SUMMARY

[NEWCOMBE ET AL., 2011]



Kinect Fusion

[Newcombe et al., 2011]

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

Kinect Fusion

[Newcombe et al., 2011]

- 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 depth] in pixels

project to 3D
multiply with K

[X Y Z] in metrics

Kinect Fusion - Measurement

[Izadi et al., 2011]

Raw Data → Bilateral Filter → Filtered Data

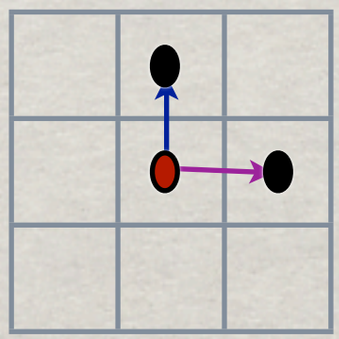
Vertex Map

Normal Map

[x y depth] in pixels

project to 3D
multiply with K

[X Y Z] in metrics



Kinect Fusion - Measurement

[Izadi et al., 2011]

Raw Data → Bilateral Filter → Filtered Data

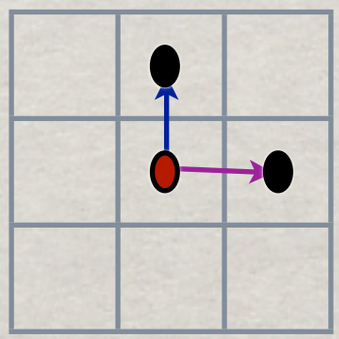
Vertex Map

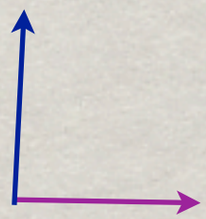
Normal Map

[x y depth] in pixels

project to 3D
multiply with K

[X Y Z] in metrics





cross product

Kinect Fusion - Measurement

[Izadi et al., 2011]

Raw Data → Bilateral Filter → Filtered Data

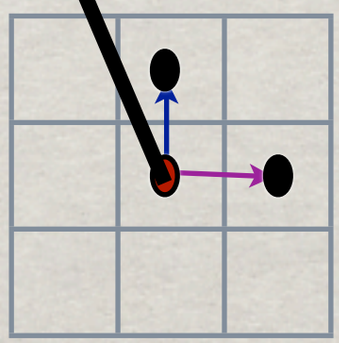
Vertex Map

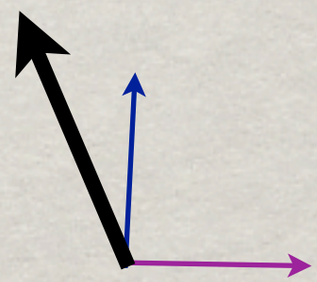
Normal Map

[x y depth] in pixels

project to 3D
multiply with K

[X Y Z] in metrics





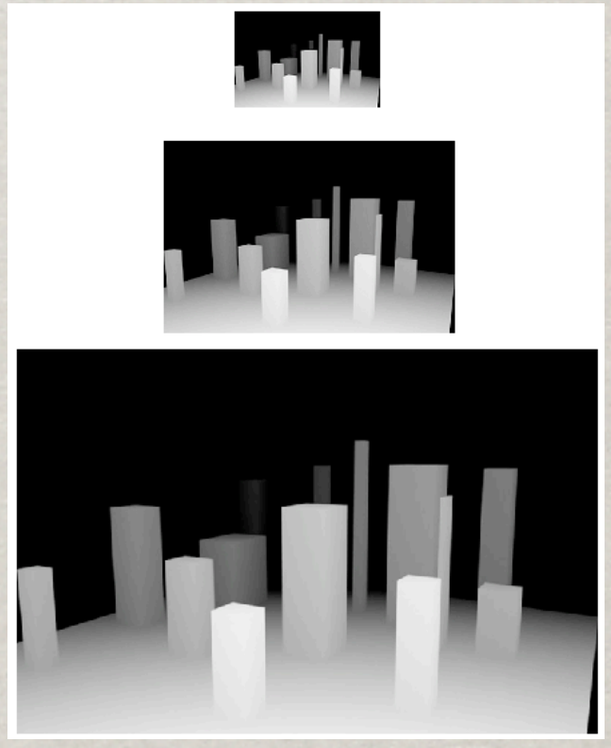
cross product

Kinect Fusion - Measurement

[Izadi et al., 2011]

Raw Data → Bilateral Filter → Filtered Data

Data Pyramid to speed-up pose tracking



http://razorvision.tumblr.com/post/15039827747/how-kinect-and-kinect-fusion-kinfu-work

Kinect Fusion

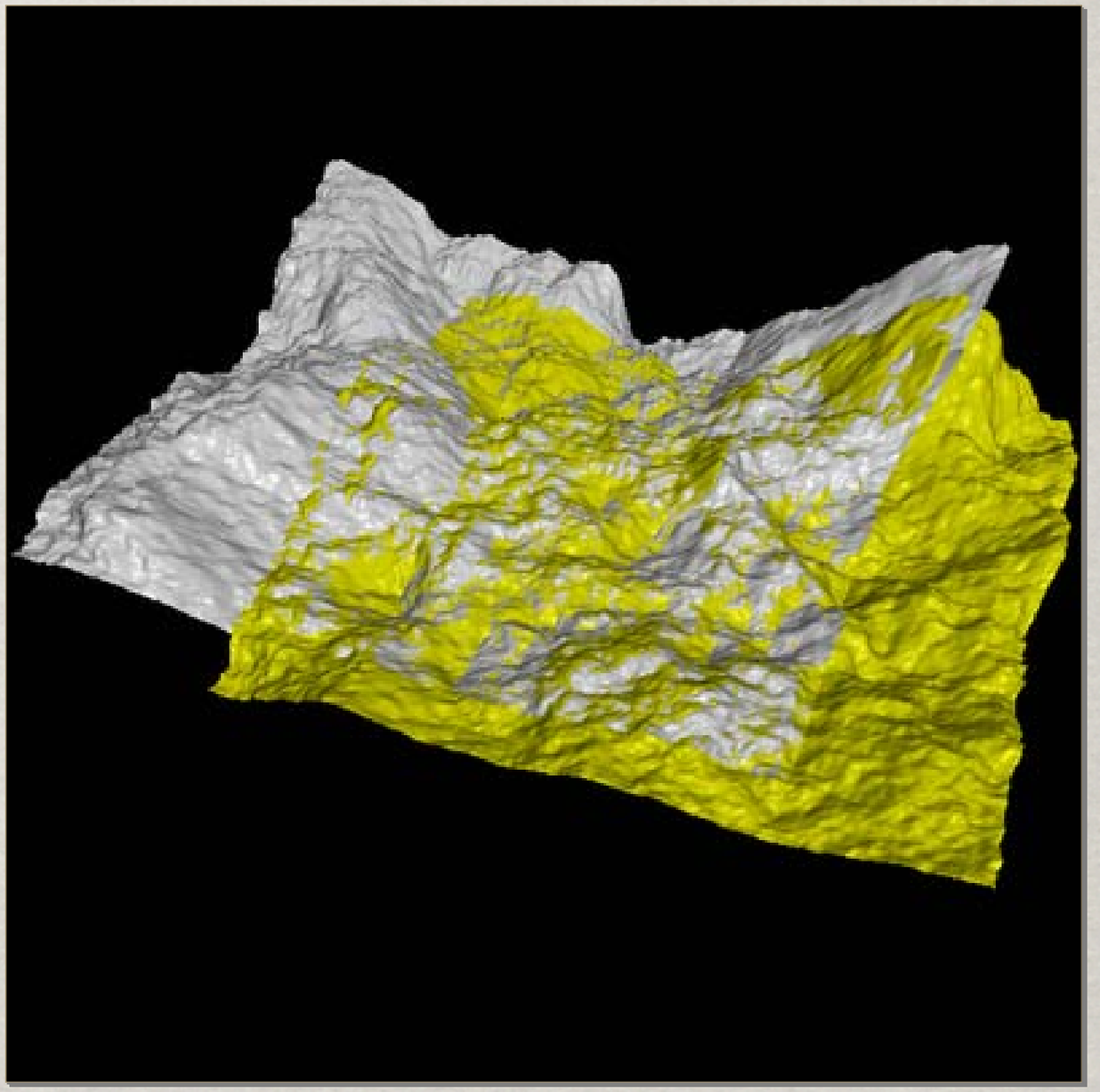
[Newcombe et al., 2011]

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

ICP - Problem

by Szymon Rusinkiewicz, 2011

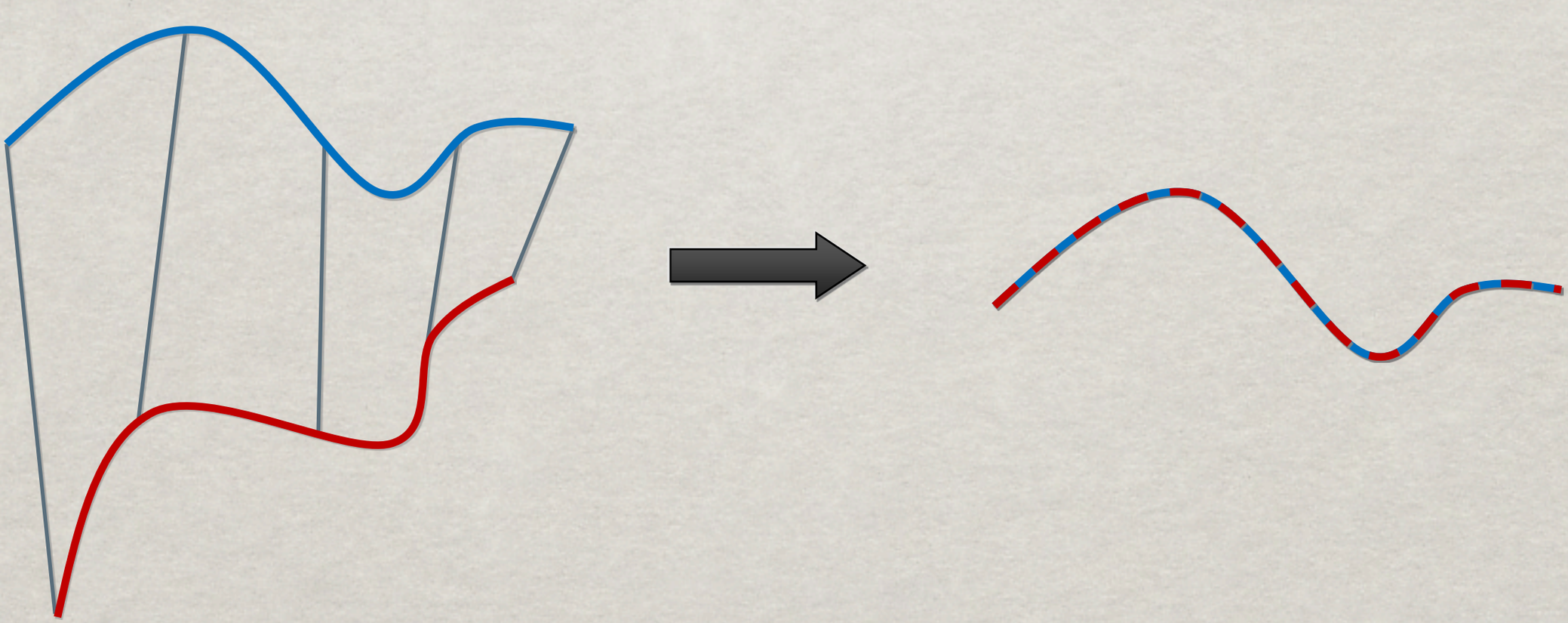
Align two partially-overlapping meshes given initial guess for relative transform



ICP - Aligning 3D Data

by Szymon Rusinkiewicz, 2011

If correct correspondences are known, can find correct relative rotation/translation



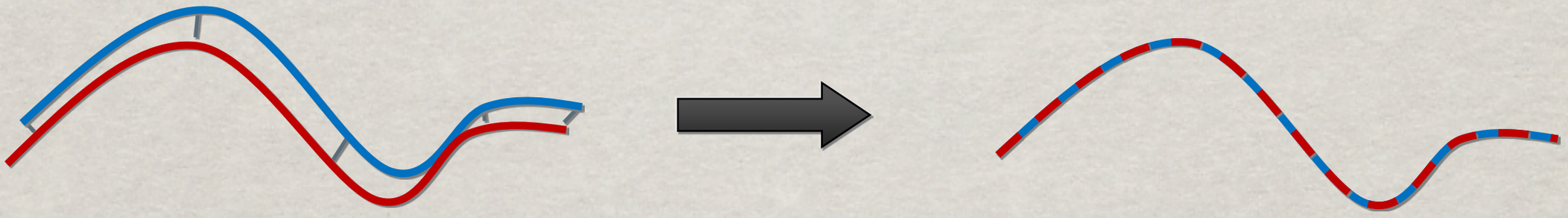
ICP - Aligning 3D Data

by Szymon Rusinkiewicz, 2011

… 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])

$$E = \sum \left\| \mathbf{R}\mathbf{p}_i + \mathbf{t} - \mathbf{q}_i \right\|^2$$

ICP Variants

by Szymon Rusinkiewicz, 2011

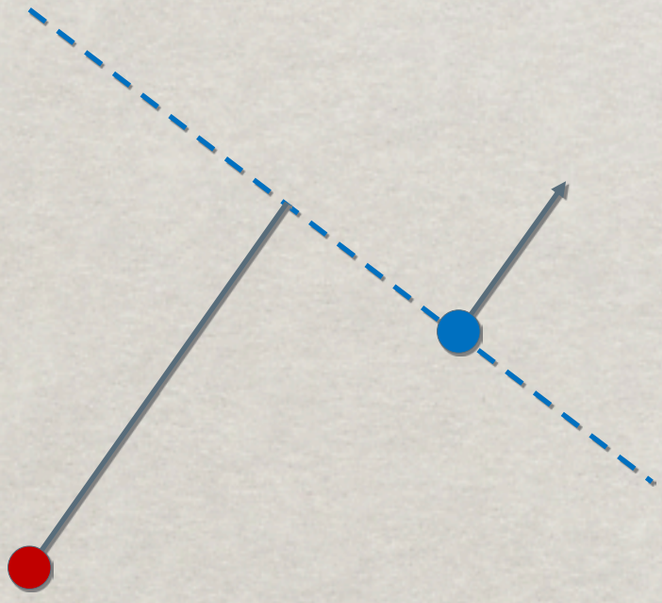
Variants on the following stages of ICP have been proposed:

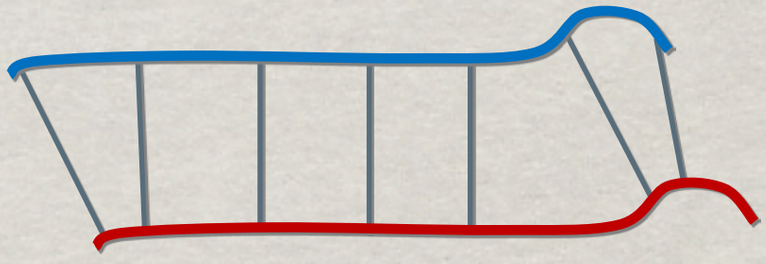
1. Selecting source points (from one or both meshes)
2. Matching to points in the other mesh
3. Weighting the correspondences
4. Rejecting certain (outlier) point pairs
5. Assigning an error metric to the current transform
6. Minimizing the error metric w.r.t. transformation

ICP - POINT-TO-PLANE ERROR

BY SZYMON RUSINKIEWICZ, 2011

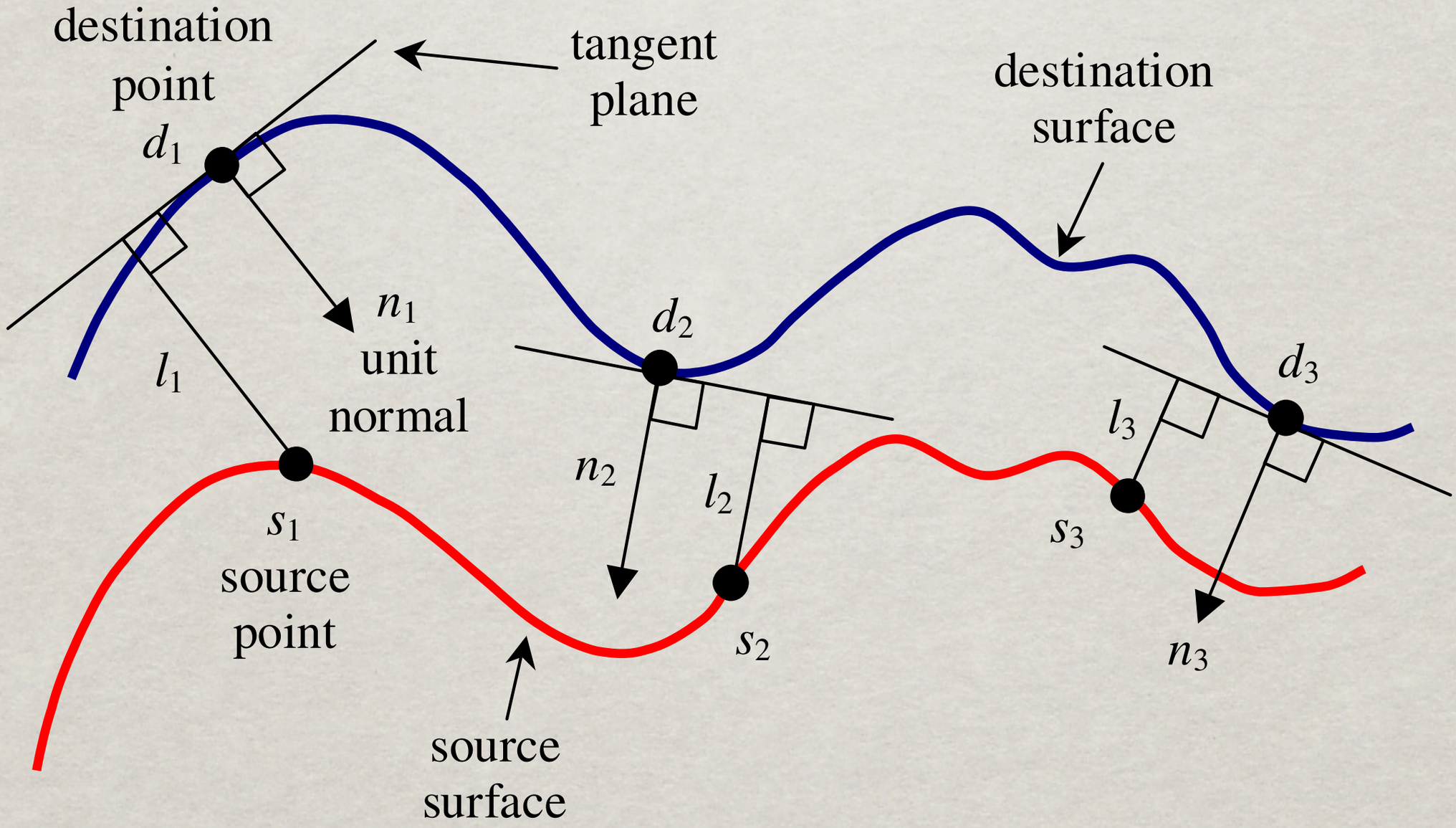
Using point-to-plane distance instead of point-to-point lets flat regions slide along each other [Chen & Medioni 91]





ICP - POINT-TO-PLANE ERROR

BY KOK-LIM LOW, 2004



ICP - POINT-TO-PLANE ERROR

BY SZYMON RUSINKIEWICZ, 2011

Error function:

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

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

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

$$E \approx \sum \left((\mathbf{p}_i - \mathbf{q}_i) \cdot \mathbf{n}_i + \mathbf{r} \cdot (\mathbf{p}_i \times \mathbf{n}_i) + \mathbf{t} \cdot \mathbf{n}_i \right)^2, \qquad \text{where } \mathbf{r} = \begin{pmatrix} \mathbf{r}_x \\ \mathbf{r}_y \\ \mathbf{r}_z \end{pmatrix}$$

Result: overconstrained linear system

ICP - POINT-TO-PLANE ERROR

BY SZYMON RUSINKIEWICZ, 2011

Overconstrained linear system

$$\mathbf{Ax} = \mathbf{b},$$

$$\mathbf{A} = \begin{pmatrix} \leftarrow & \mathbf{p}_1 \times \mathbf{n}_1 & \rightarrow & \leftarrow & \mathbf{n}_1 & \rightarrow \\ \leftarrow & \mathbf{p}_2 \times \mathbf{n}_2 & \rightarrow & \leftarrow & \mathbf{n}_2 & \rightarrow \\ & \vdots & & & \vdots & \end{pmatrix}, \qquad \mathbf{x} = \begin{pmatrix} r_x \\ r_y \\ r_z \\ t_x \\ t_y \\ t_z \end{pmatrix}, \qquad \mathbf{b} = \begin{pmatrix} -(\mathbf{p}_1 - \mathbf{q}_1) \cdot \mathbf{n}_1 \\ -(\mathbf{p}_2 - \mathbf{q}_2) \cdot \mathbf{n}_2 \\ \vdots \end{pmatrix}$$

Solve using least squares

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

$$\mathbf{x} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$$

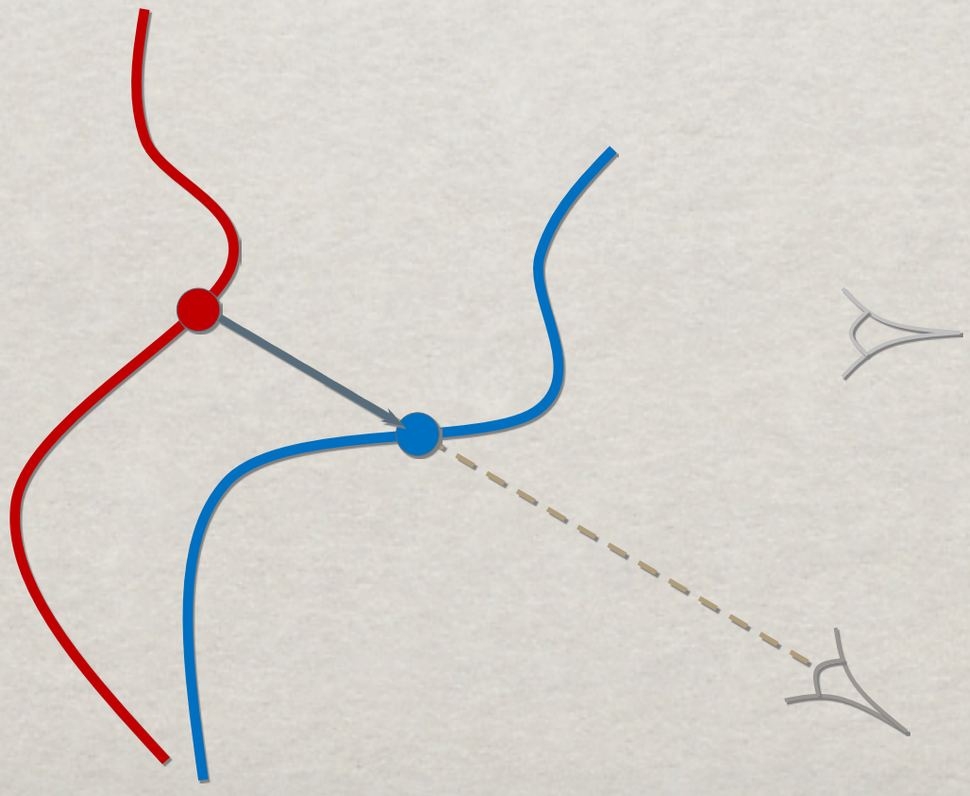
ICP - Data Projection

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



EFFICIENT ICP

BY SZYMON RUSINKIEWICZ & MARC LEVOY, 2001

- ICP algorithm with
  - projection-based correspondences,
  - point-to-plane matching

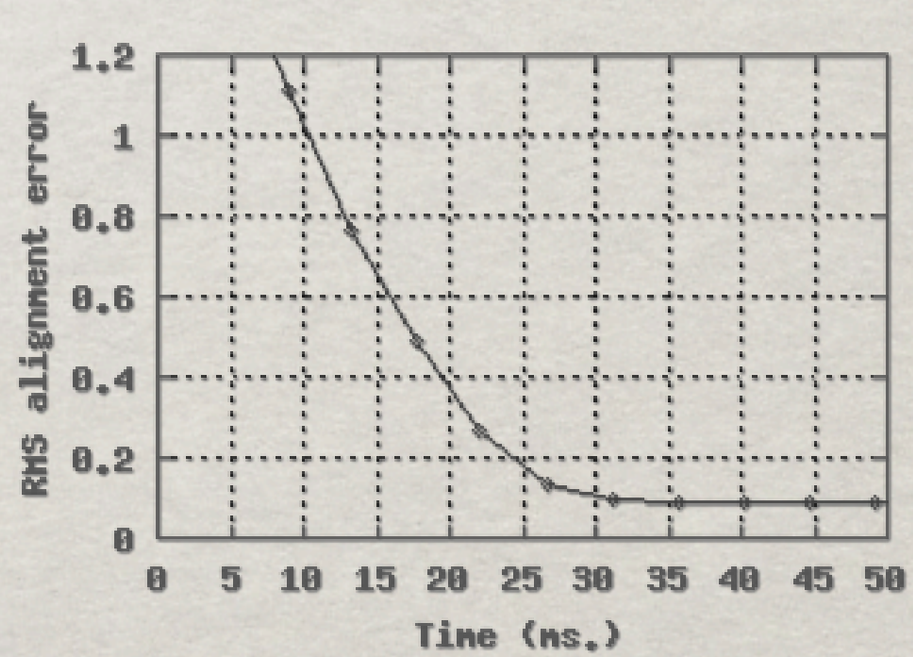
can align meshes in a few tens of ms.

EFFICIENT ICP

BY SZYMON RUSINKIEWICZ & MARC LEVOY, 2001

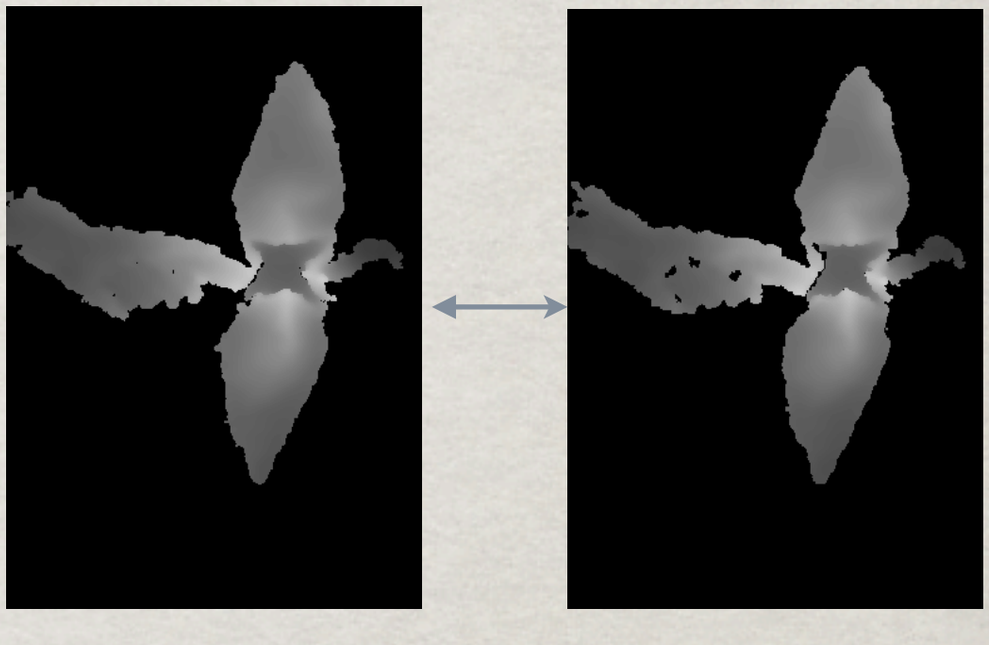
Each iteration is one to two orders of magnitude faster than closest-point

Result: can align two range images in a few milliseconds, vs. a few seconds



Frame-to-Model Alignment

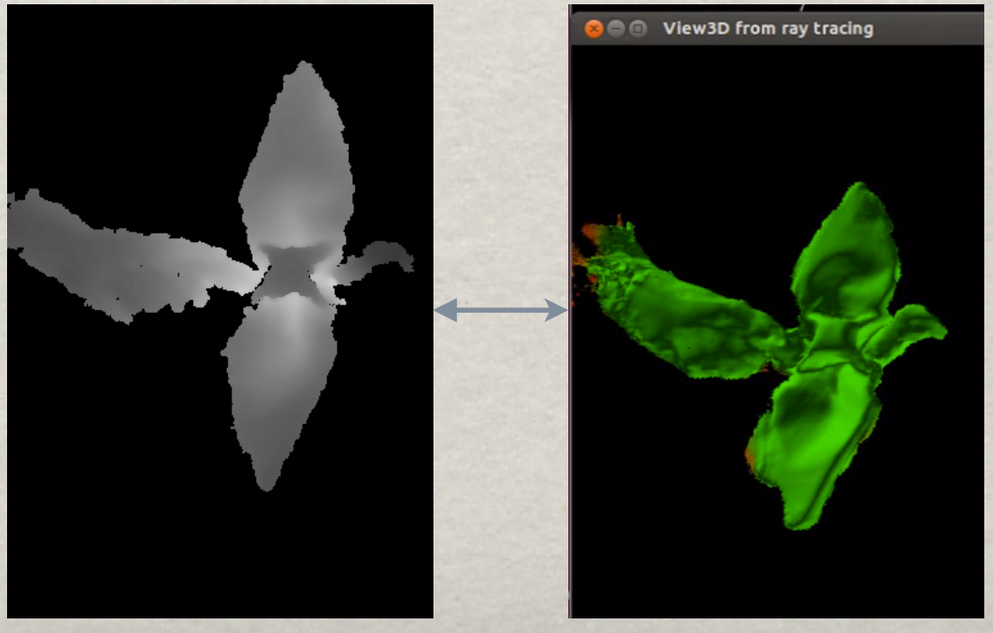
- Frame-to-frame alignment:
  - Unstable
  - Prune to incremental drift



Frame-to-Model Alignment

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

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



Kinect Fusion

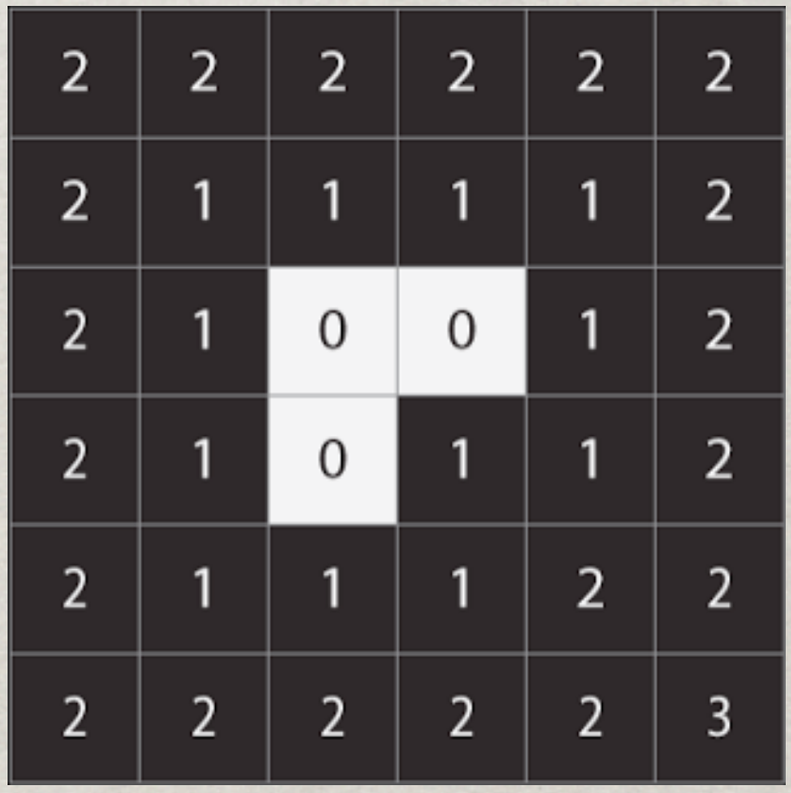
[Newcombe et al., 2011]

- 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

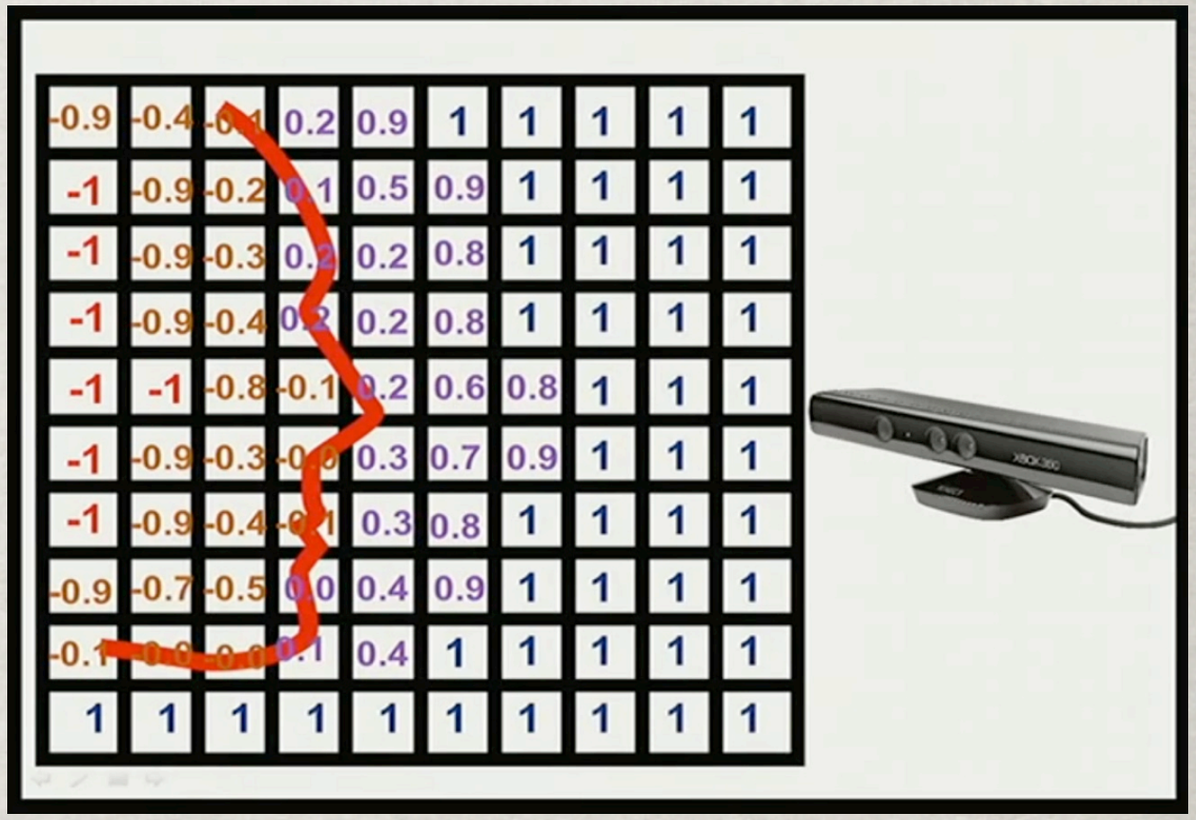
http://www.paraview.org/Wiki/ITK/Examples/ImageProcessing/SignedDanielssonDistanceMapImageFilter



http://2.bp.blogspot.com/-bQanRezjlk4/TpdpSpMq5rI/AAAAAAAAAWw/_8lQsmOJHWM/s320/SignedDistanceField.png

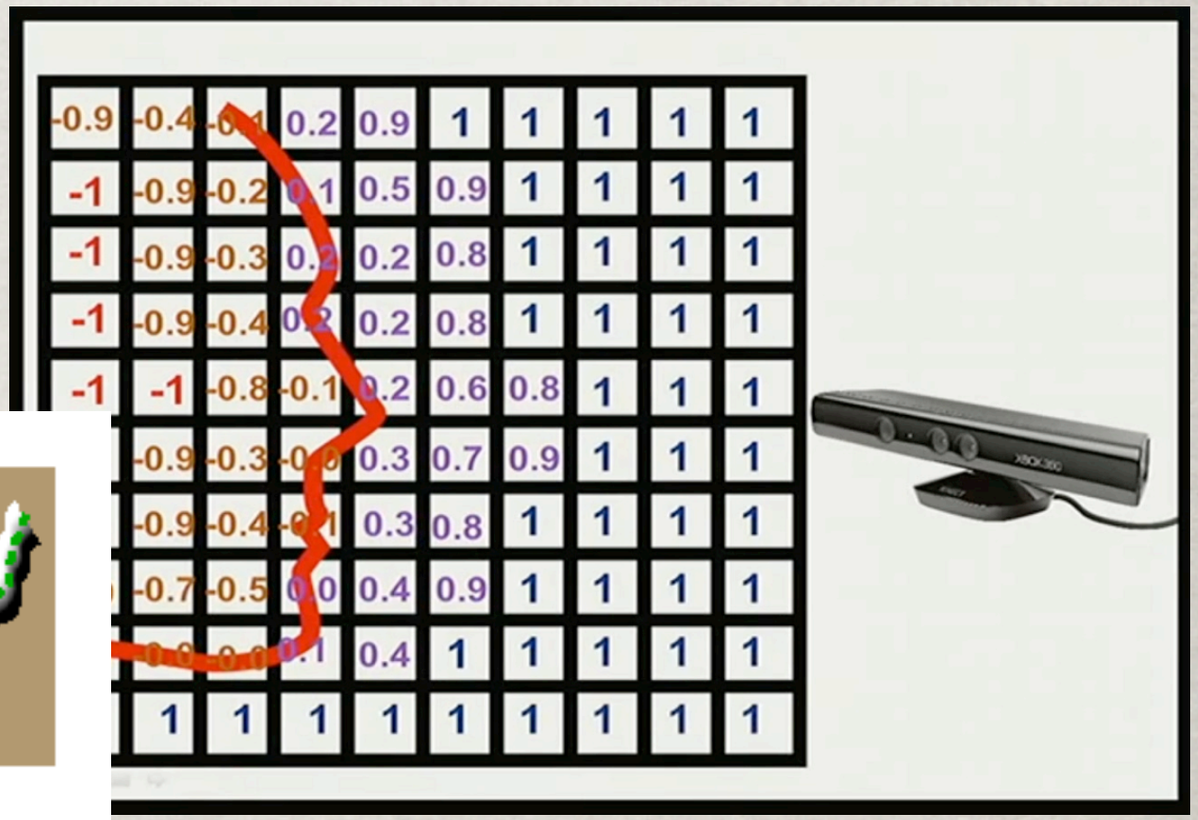
TSDF : TRUNCATED SIGNED DISTANCE FUNCTION

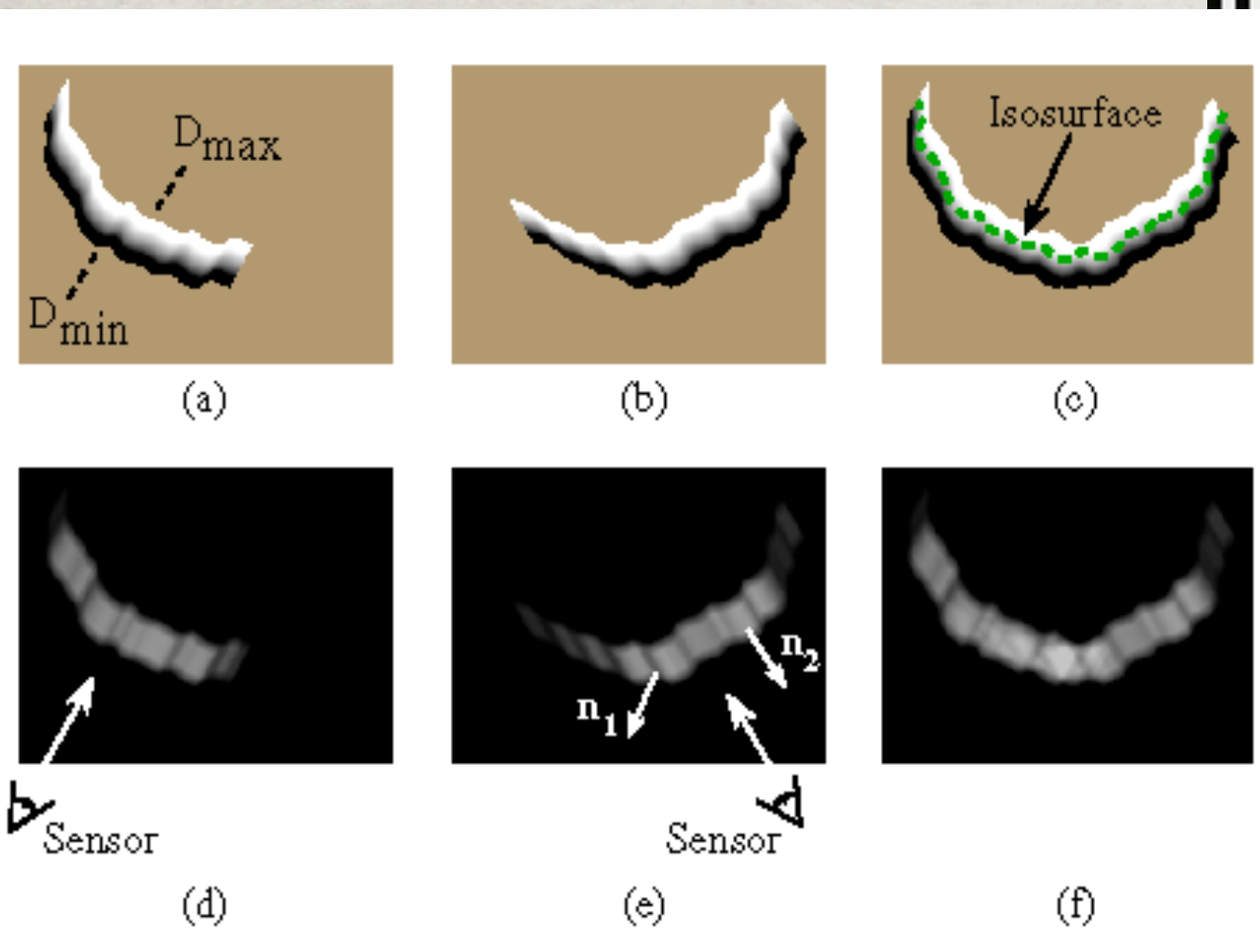
Update only the close voxels to the boundary



TSDF : TRUNCATED SIGNED DISTANCE FUNCTION

Update only the close voxels to the boundary

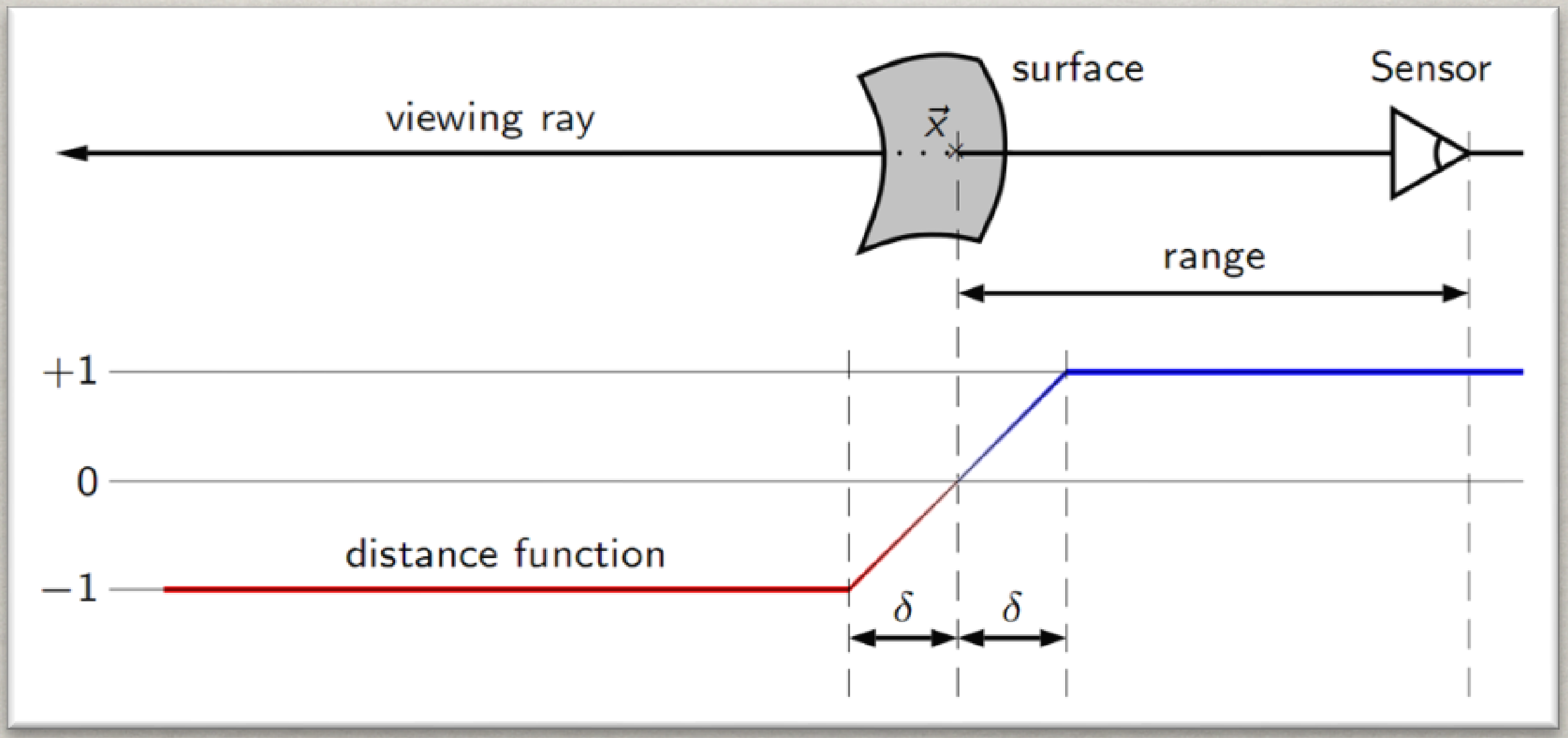




http://graphics.stanford.edu/papers/volrange/paper_2_levels/node3.html

TSDF : Truncated Signed Distance Function

by Michele Pirovano, 2011

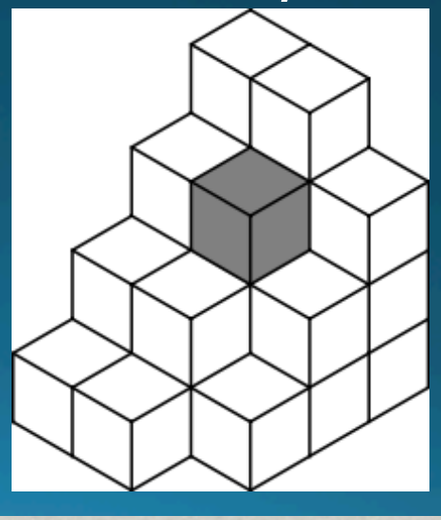


TSDF integration

by Milwaukee 3D Printing Meetup & Voxel Metric

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.

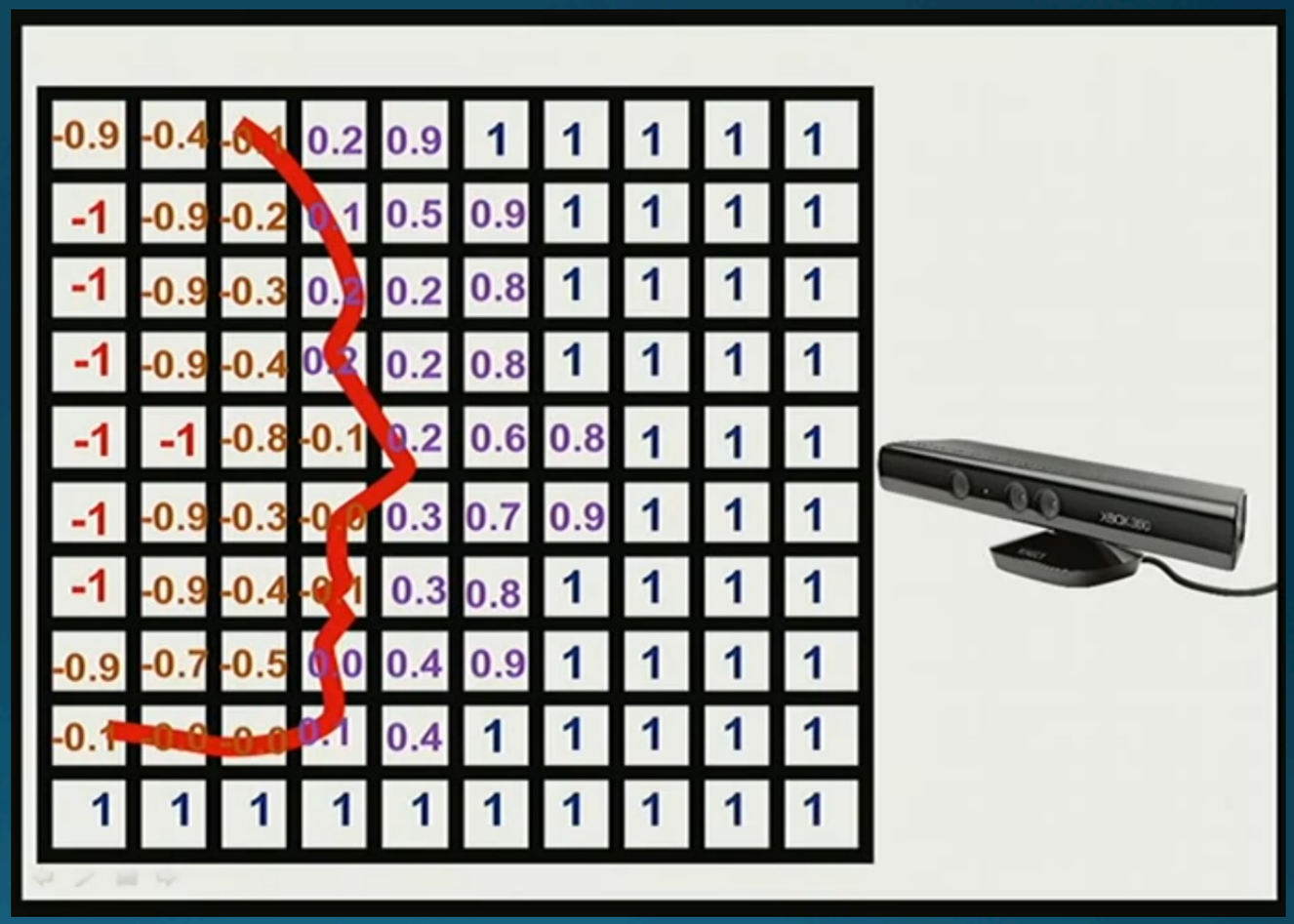


TSDF integration

by Milwaukee 3D Printing Meetup & Voxel Metric

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



TSDF INTEGRATION

BY MILWAUKEE 3D PRINTING MEETUP & VOXEL METRIC

Step 6: Calculating TSDF Values

Line from each vertex to camera through voxel grid.

Intersected voxels near the surface are updated

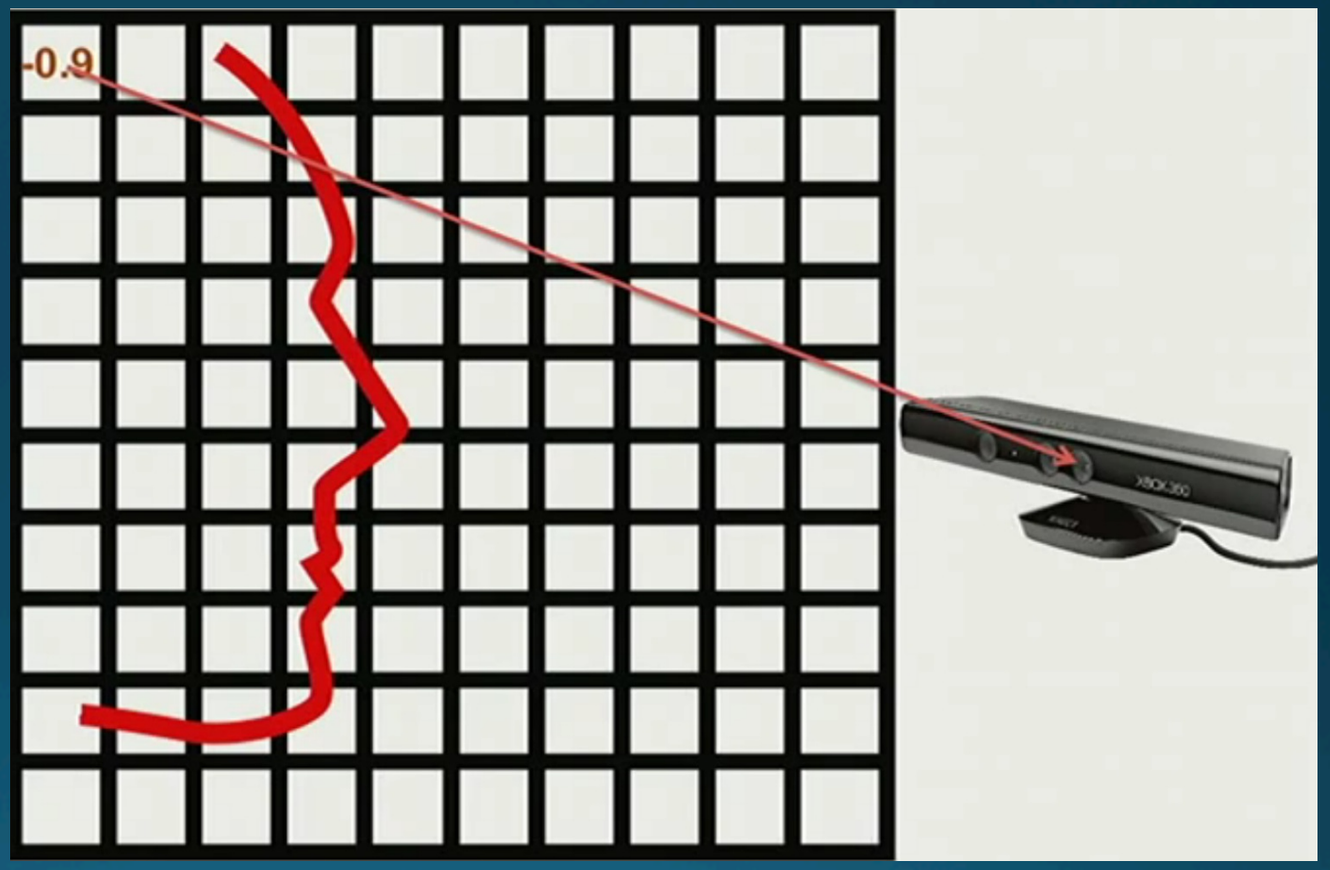
Distance from vertex to voxel center is distance value for a given voxel

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.



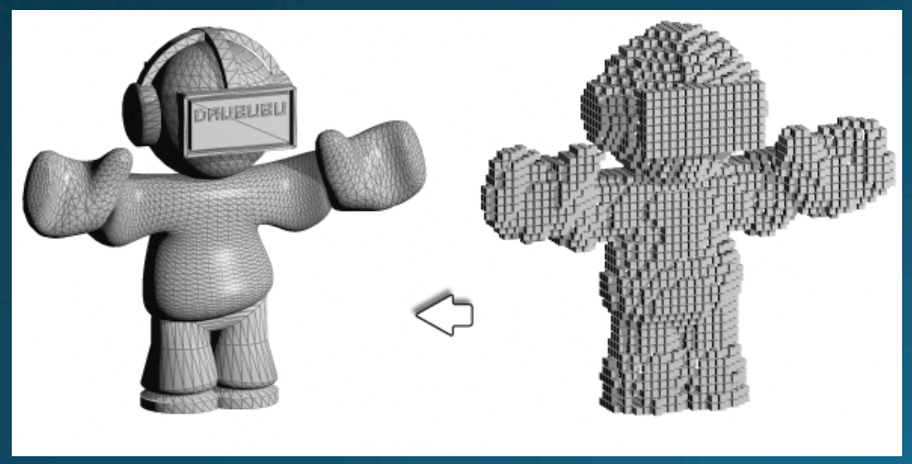
TSDF integration

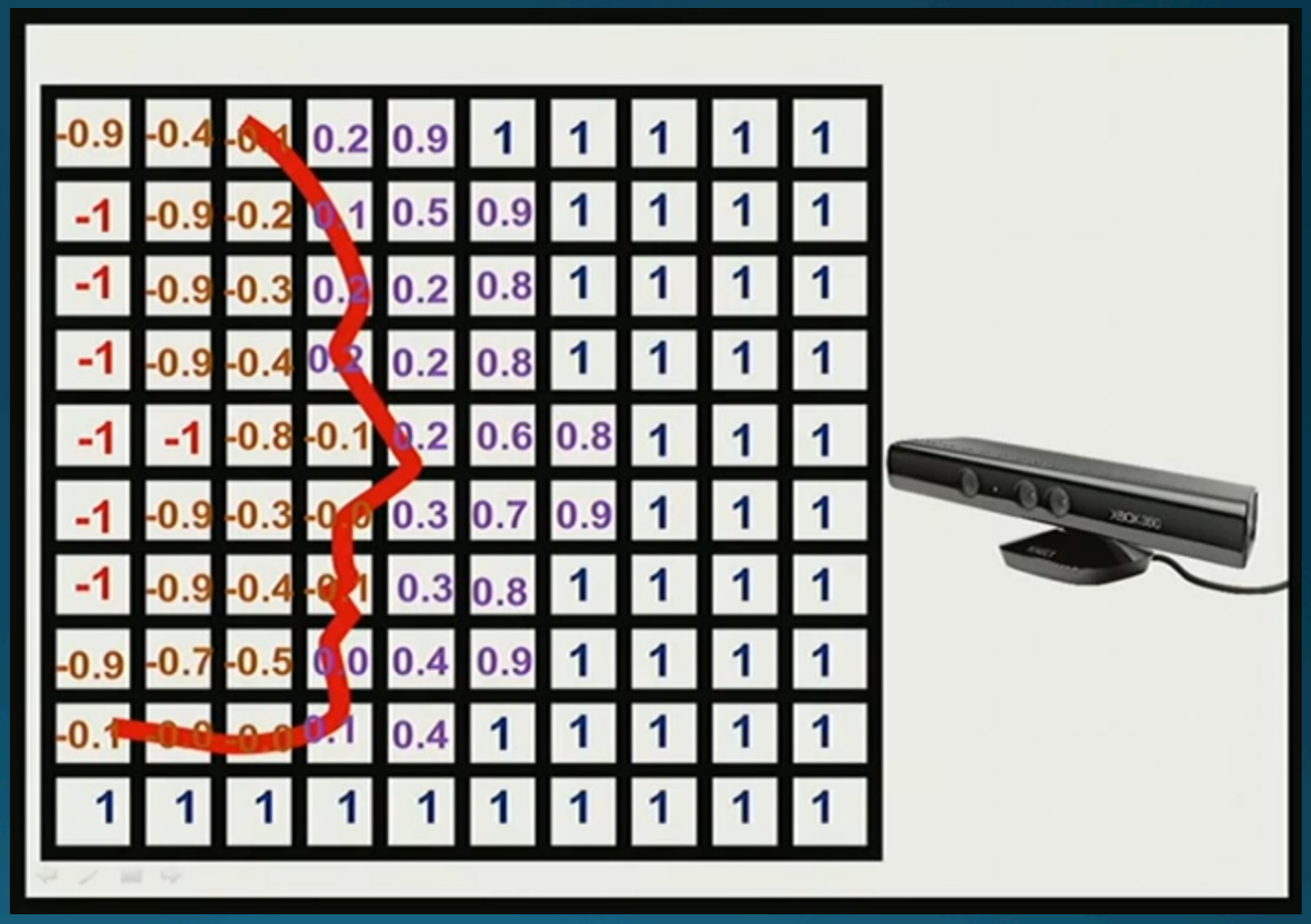
by Milwaukee 3D Printing Meetup & Voxel Metric

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.





TSDF integration
[Newcombe et al., 2011]

- Each voxel stores:
  - F: signed distance
  - W: accumulated weight

Moving average at time k:

$$\mathrm{F}_k(\mathbf{p}) = \frac{\mathrm{W}_{k-1}(\mathbf{p})\mathrm{F}_{k-1}(\mathbf{p}) + \mathrm{W}_{\mathrm{R}_k}(\mathbf{p})\mathrm{F}_{\mathrm{R}_k}(\mathbf{p})}{\mathrm{W}_{k-1}(\mathbf{p}) + \mathrm{W}_{\mathrm{R}_k}(\mathbf{p})}$$

$$\mathrm{W}_k(\mathbf{p}) = \mathrm{W}_{k-1}(\mathbf{p}) + \mathrm{W}_{\mathrm{R}_k}(\mathbf{p})$$

TSDF integration

[Newcombe et al., 2011]

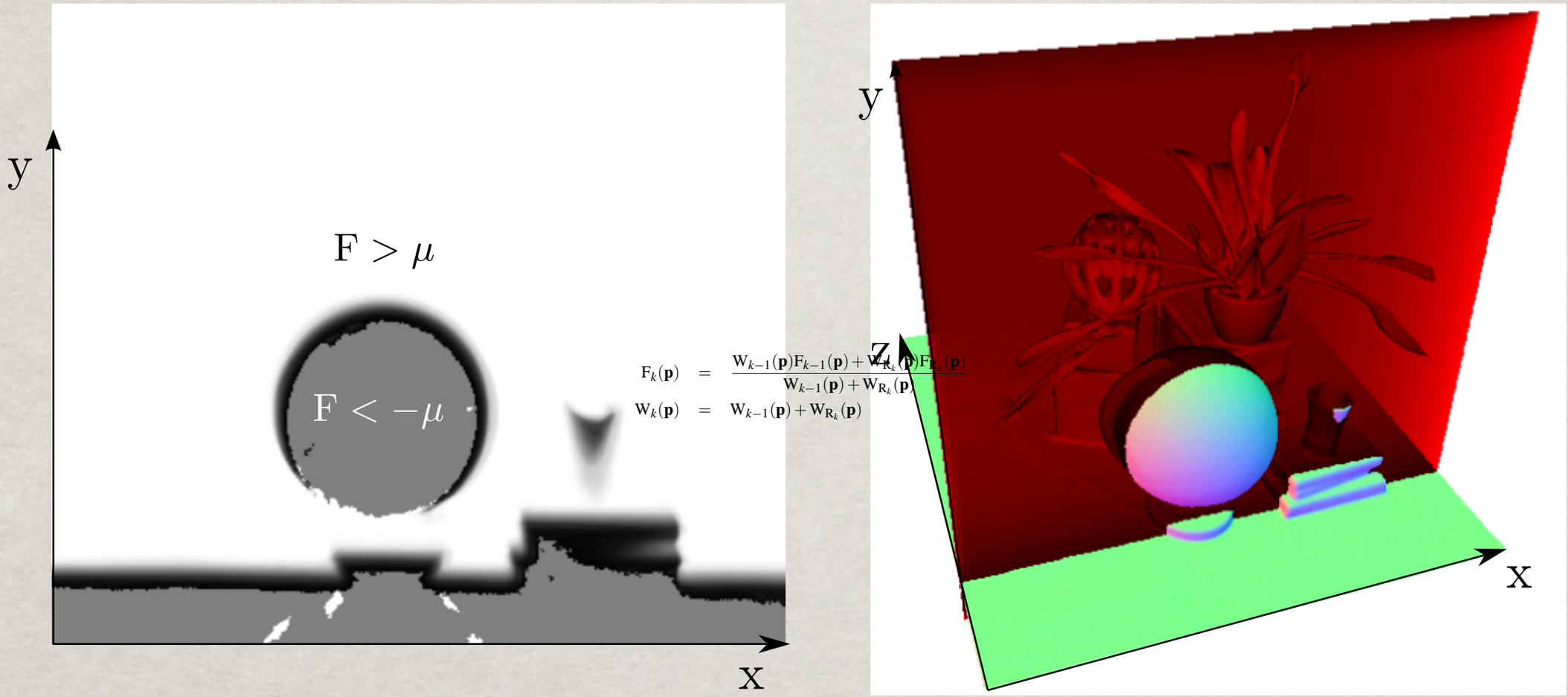


Figure 4: A slice through the truncated signed distance volume showing the truncated function $F > \mu$ (white), the smooth distance field around the surface interface $F = 0$ and voxels that have not yet had a valid measurement(grey) as detailed in eqn. 9.

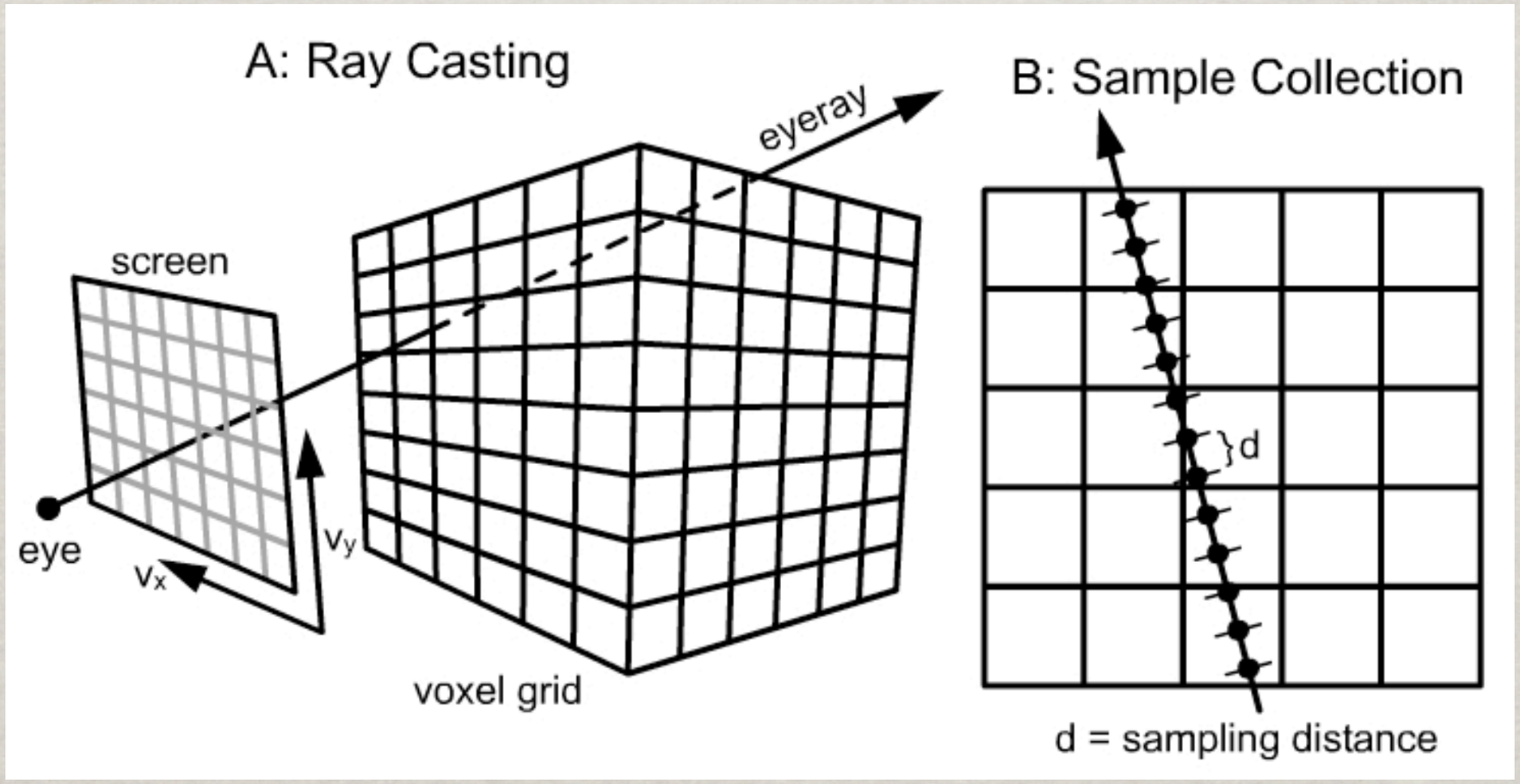
Kinect Fusion

[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



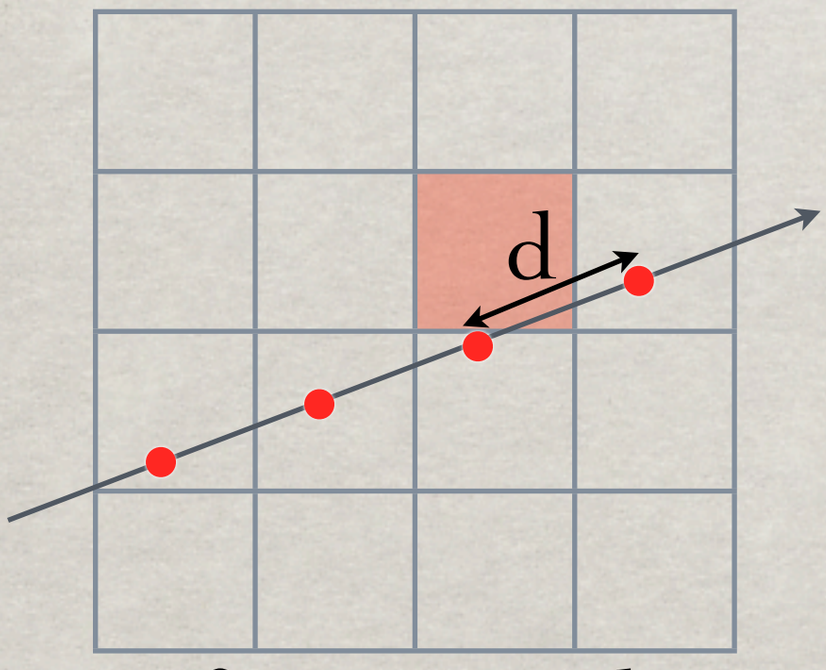
http://johnrichie.com/V2/richie/isosurface/volume.html

Ray casting

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

Ray casting

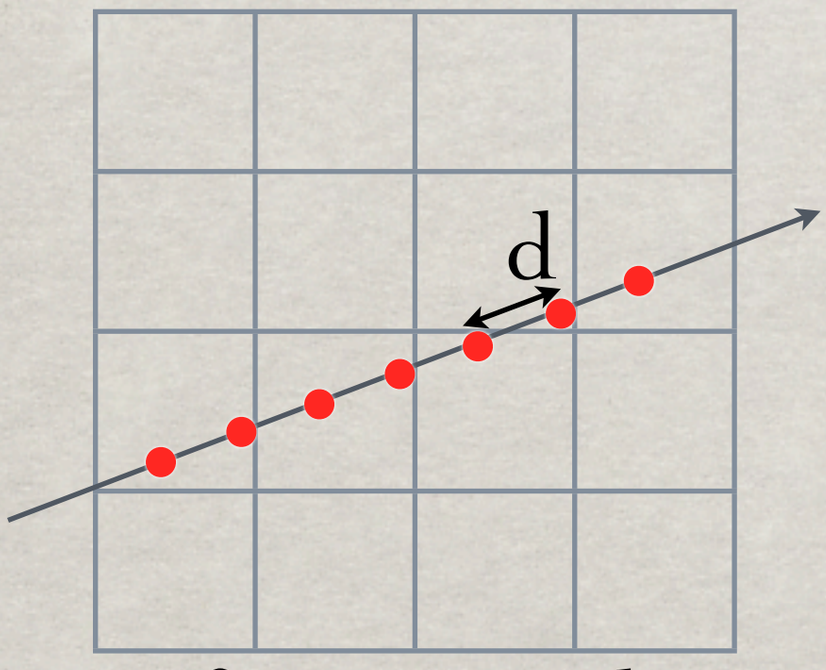
- Find “zero-crossings” in tsdf volume
- p: extracted grid position by trilinear interpolation
- Uniform sampling for voxel traversal



Uniform sampling

RAy casting

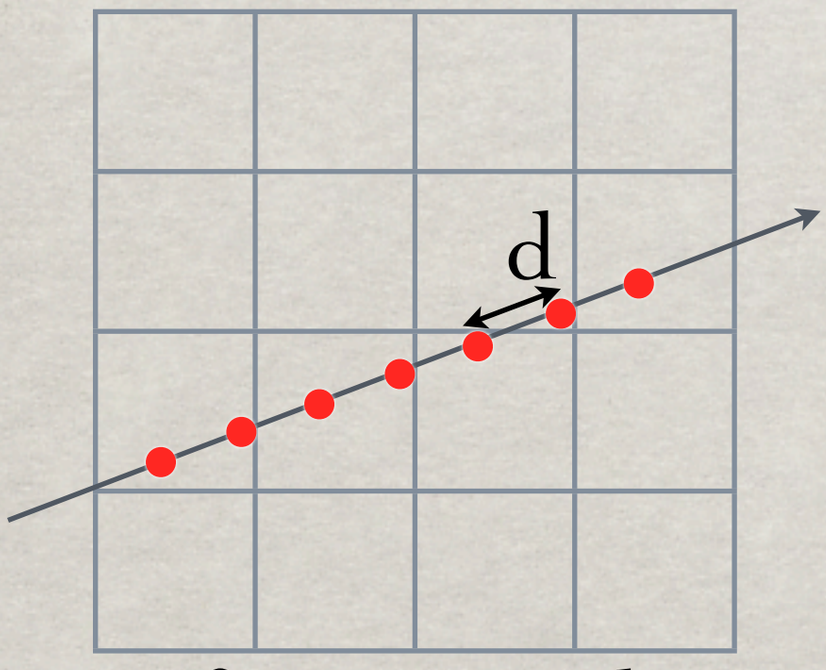
- Find “zero-crossings” in tsdf volume
- p: extracted grid position by trilinear interpolation
- Uniform sampling for voxel traversal



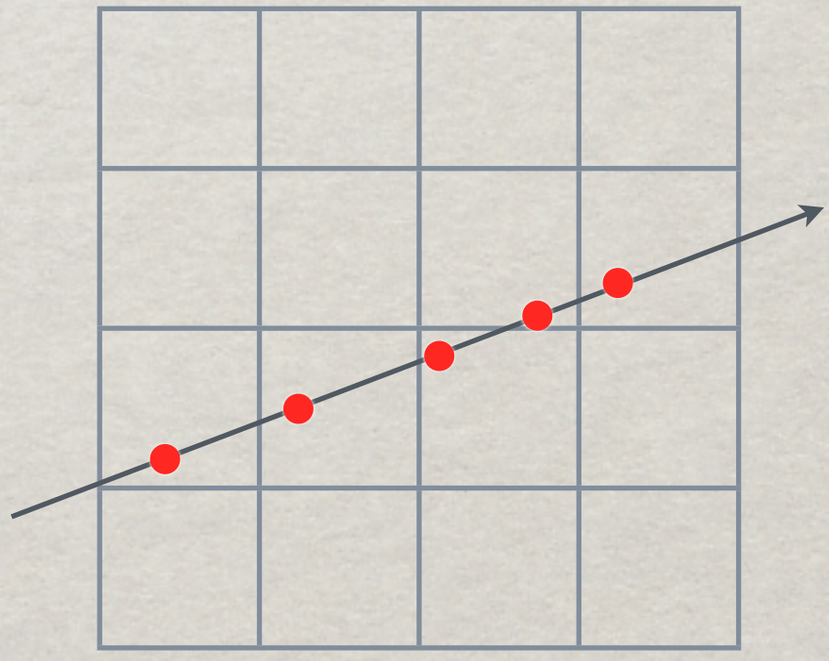
Uniform sampling

Ray casting

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



Uniform sampling



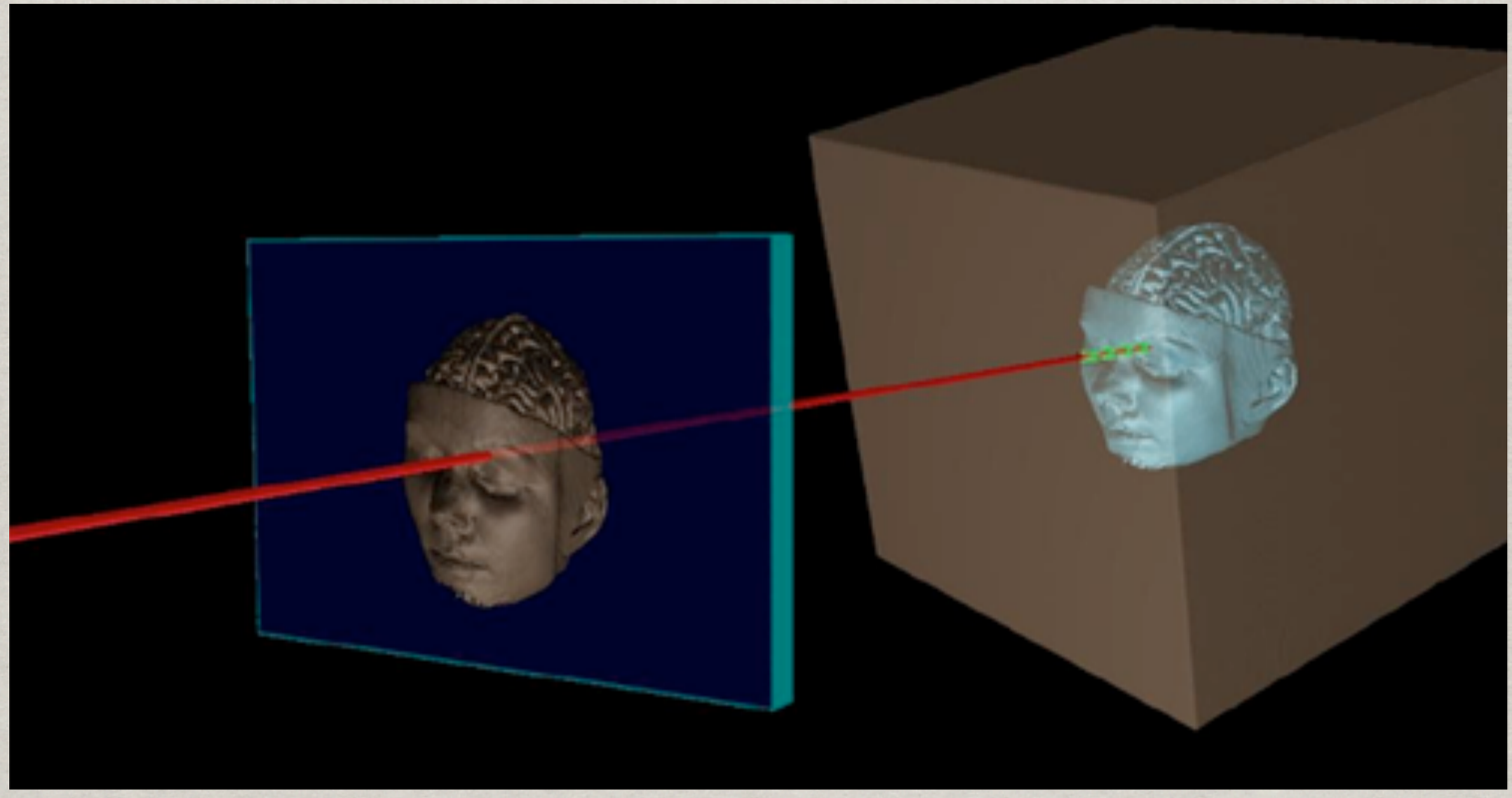
DDA sampling

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

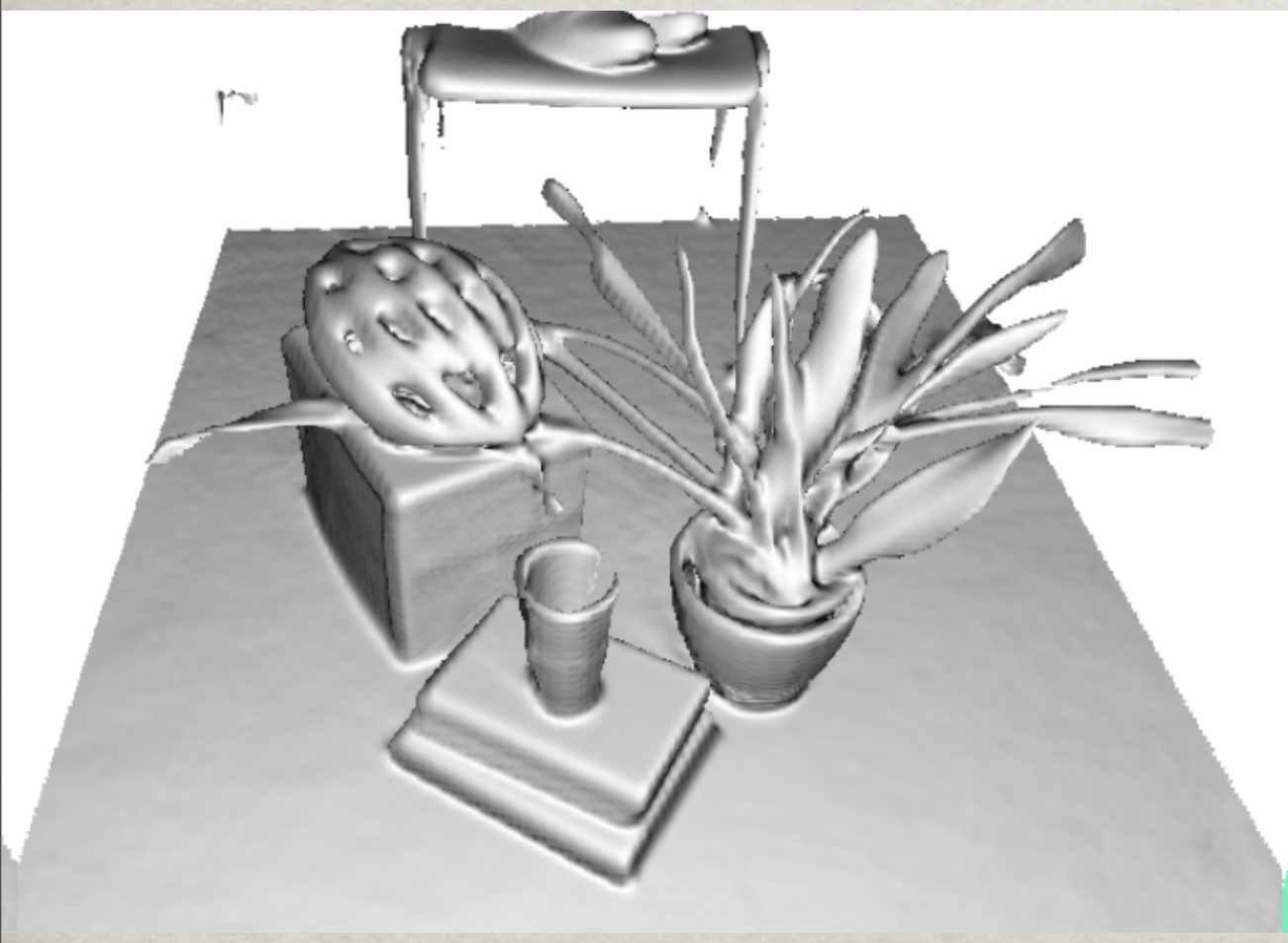


http://razorvision.tumblr.com/post/15039827747/how-kinect-and-kinect-fusion-kinfu-work

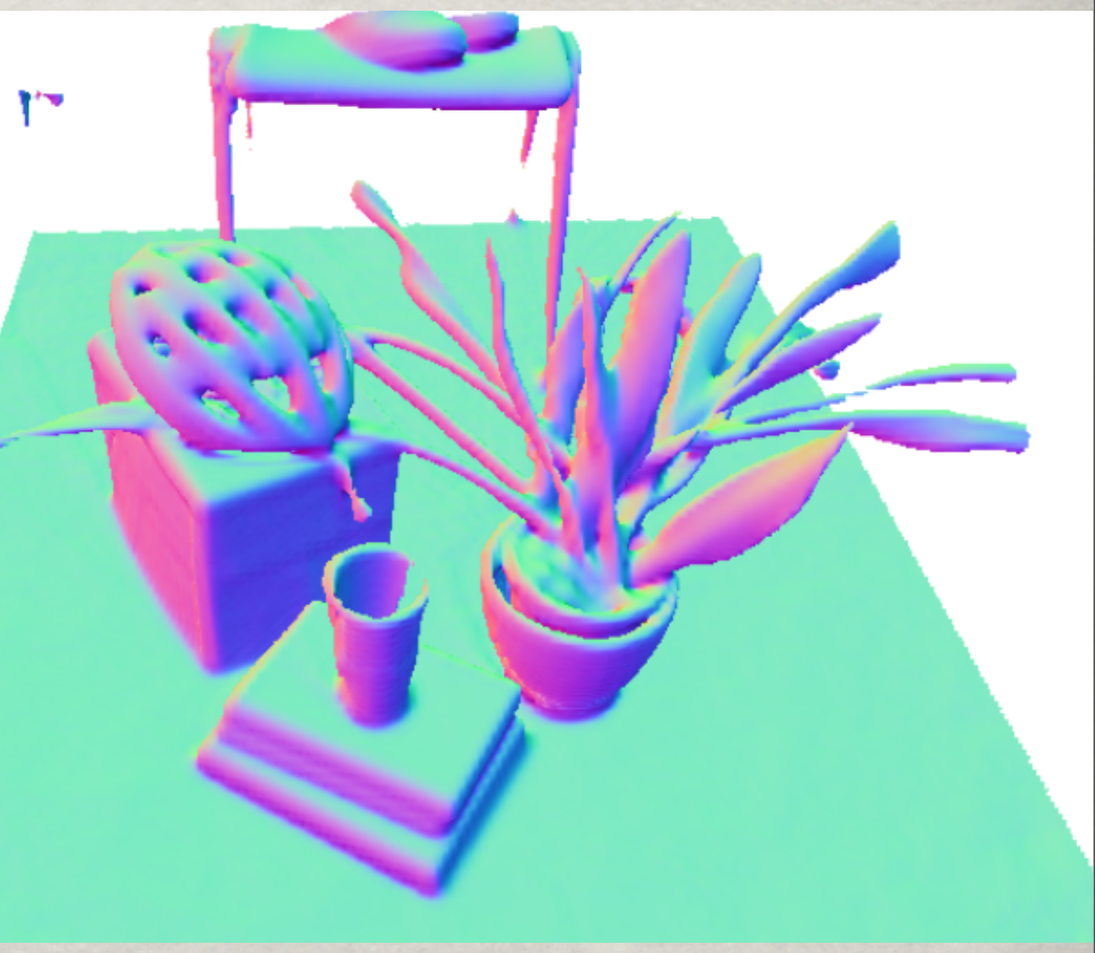
RAy casting
[Newcombe et al., 2011]

Ray-casting on the TSDF

Surface

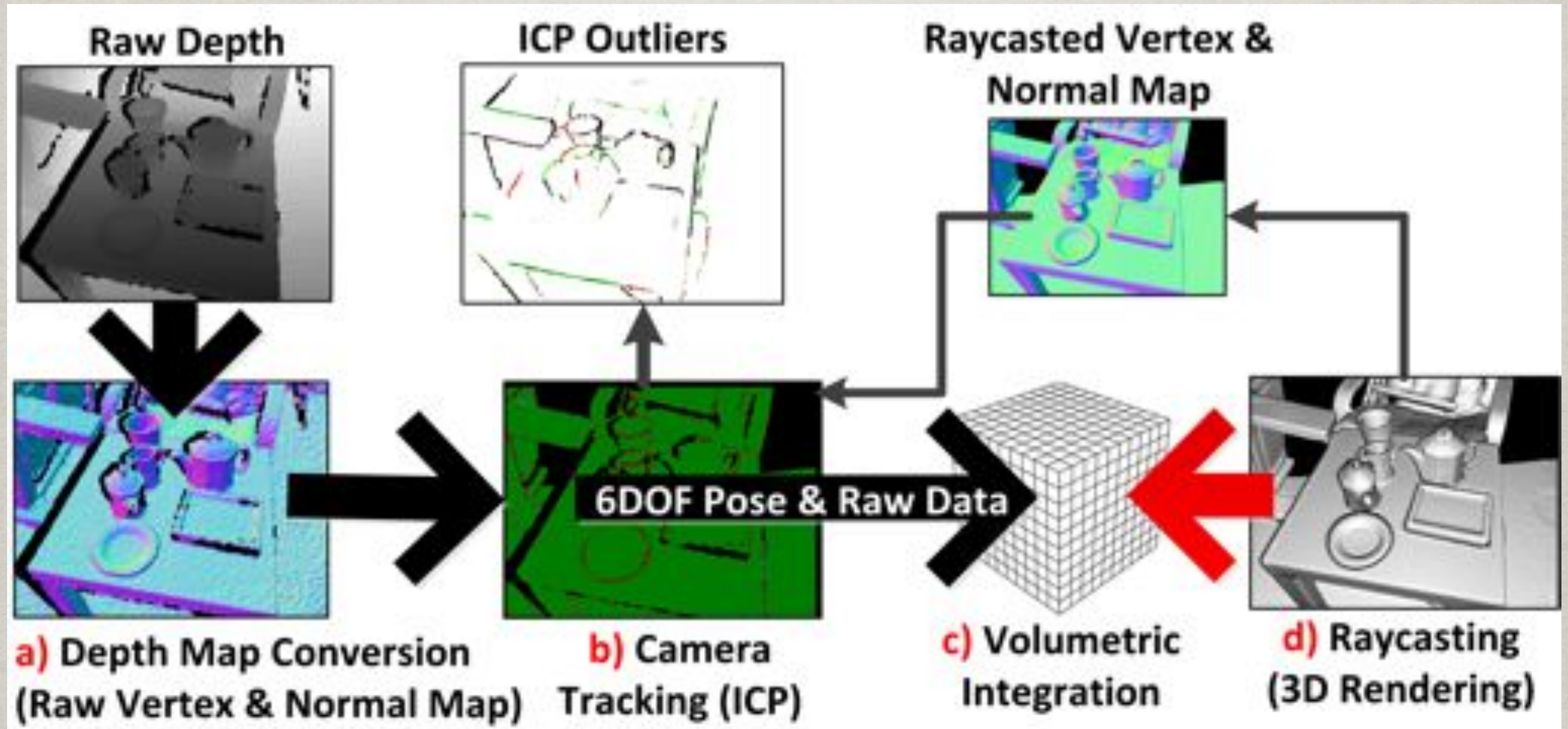


Normals



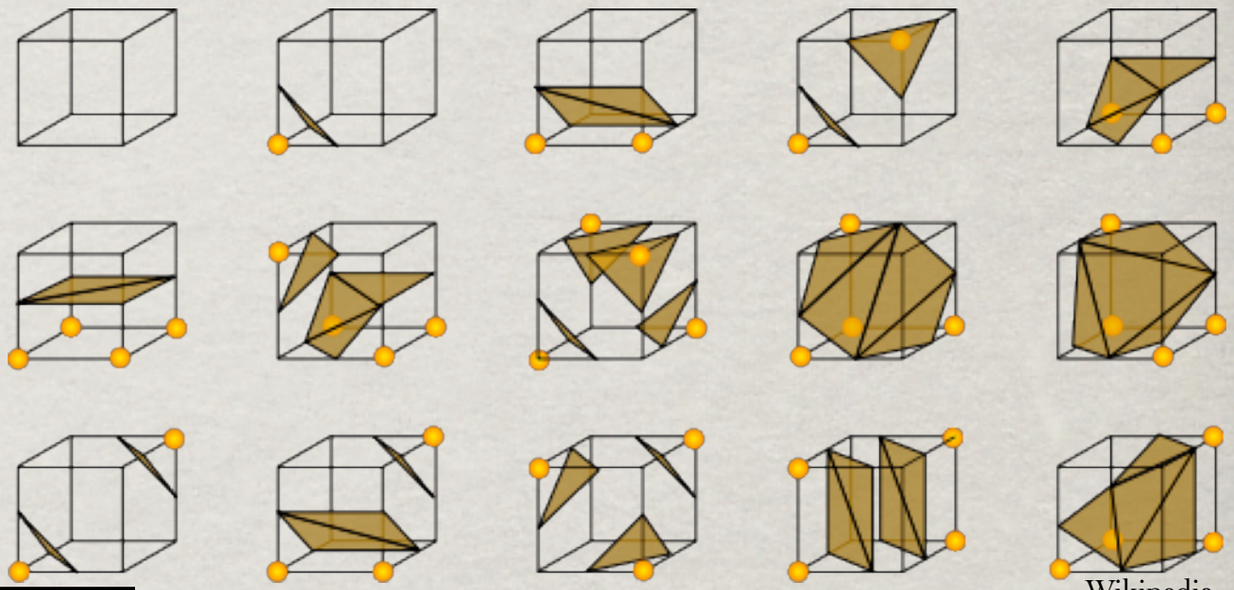
KINECT FUSION: RECAP

[IZADI ET AL., 2011]

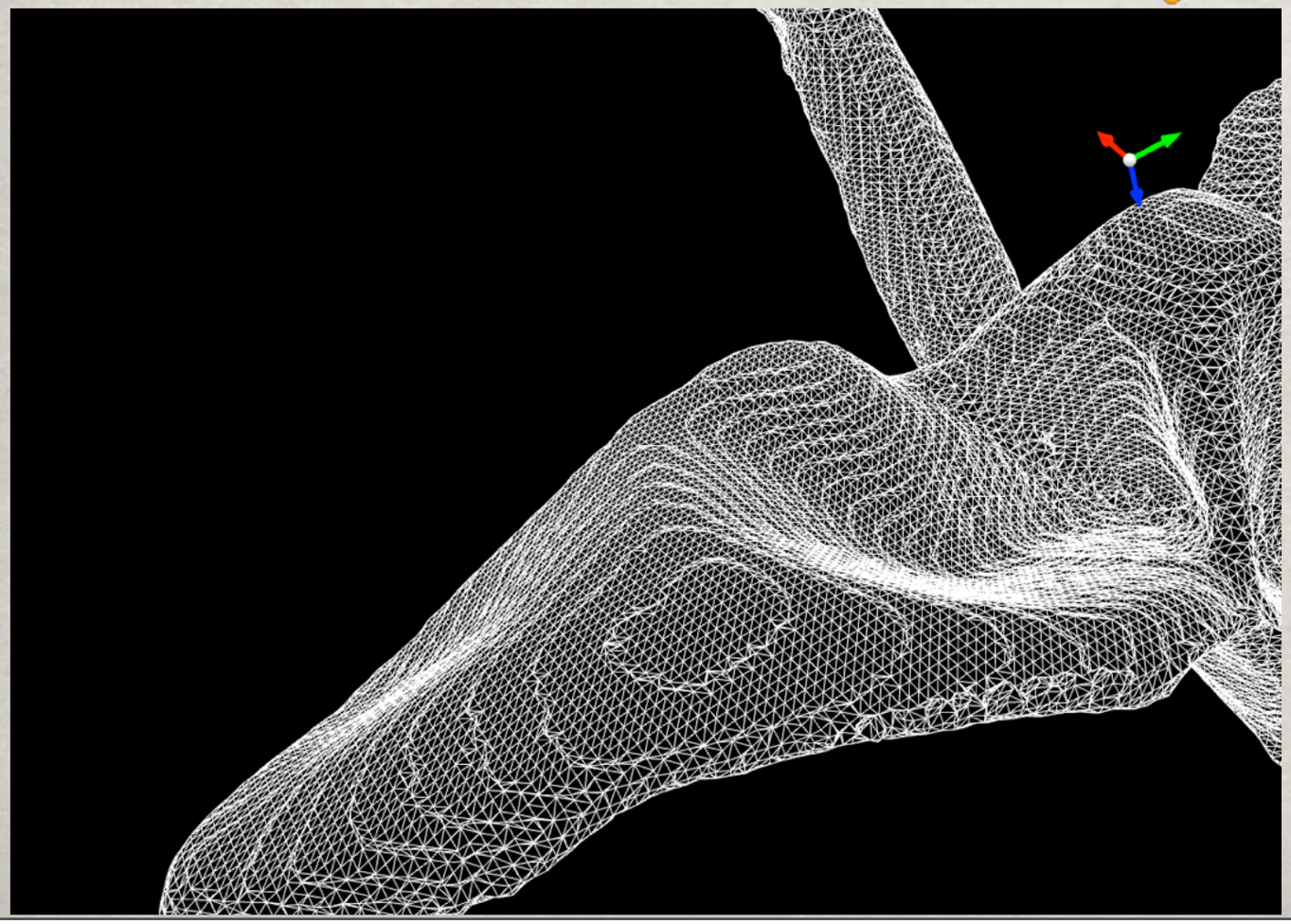


At the end of the day: Model Extraction

Marching Cubes on TSDF



Wikipedia



OUTLINE

I. Kinect Fusion

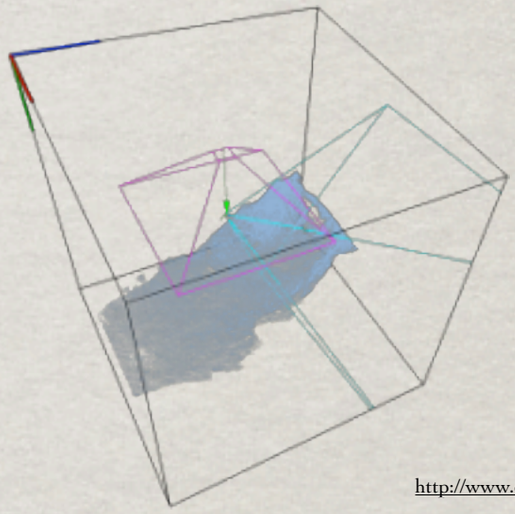
II. Kinect Fusion extensions

III. Keyframe based approach

IV. Deformable and non-rigid alignment

KinectFusion extensions

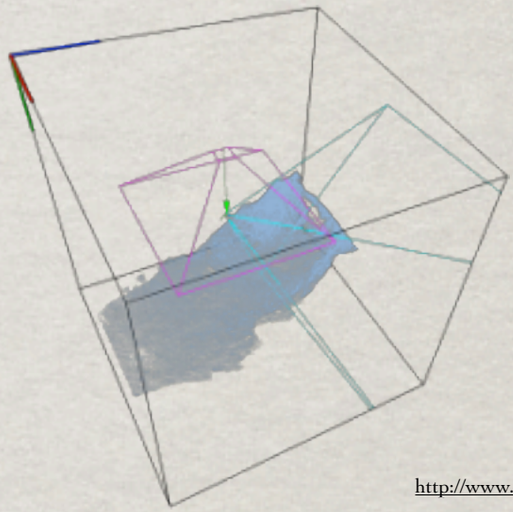
- Problems with Kinect Fusion:
  - Limited integration volume size



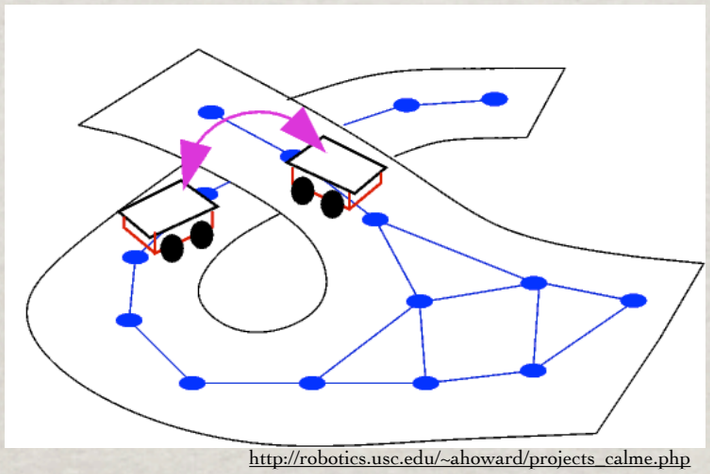
http://www.ccs.neu.edu/research/gpc/

KinectFusion extensions

- Problems with Kinect Fusion:
  - Limited integration volume size



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

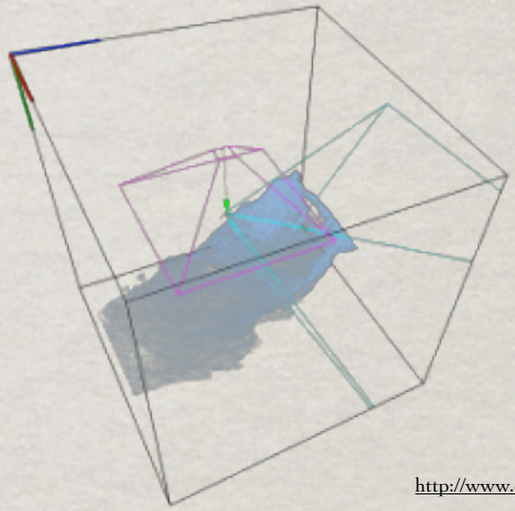


http://robotics.usc.edu/~ahoward/projects_calme.php

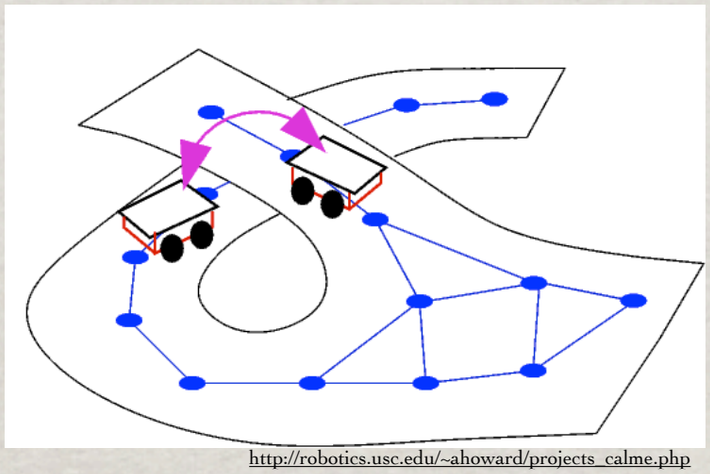
  - Loop closure

KinectFusion extensions

- Problems with Kinect Fusion:
  - Limited integration volume size

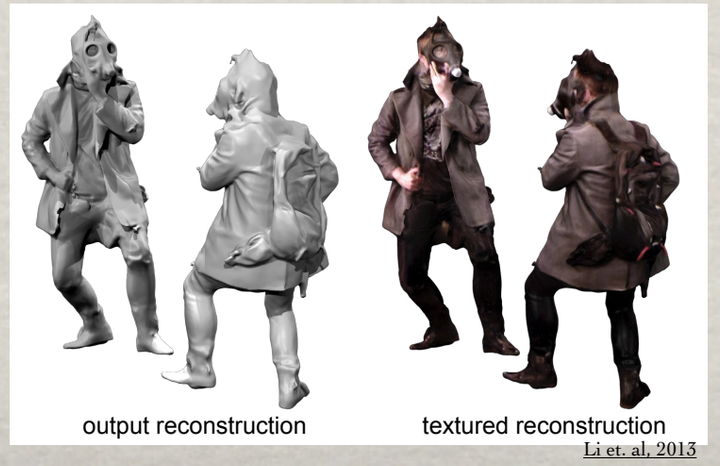


http://www.ccs.neu.edu/research/gpc/



http://robotics.usc.edu/~ahoward/projects_calme.php

- Loop closure
- Texturing



Li et. al, 2013

KinectFusion extensions

- Circular buffer for space extention [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]

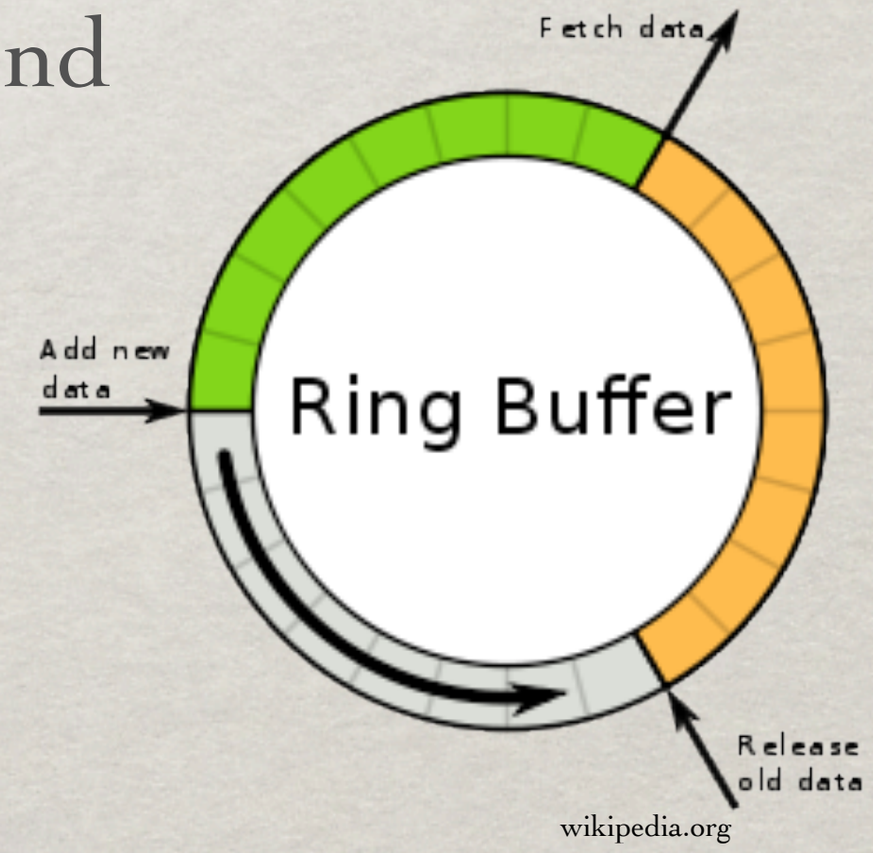
KINTINUOUS

Whelan et. al, 2012

- 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 (RING) BUFFER

- Circular buffer makes space-extended scanning possible

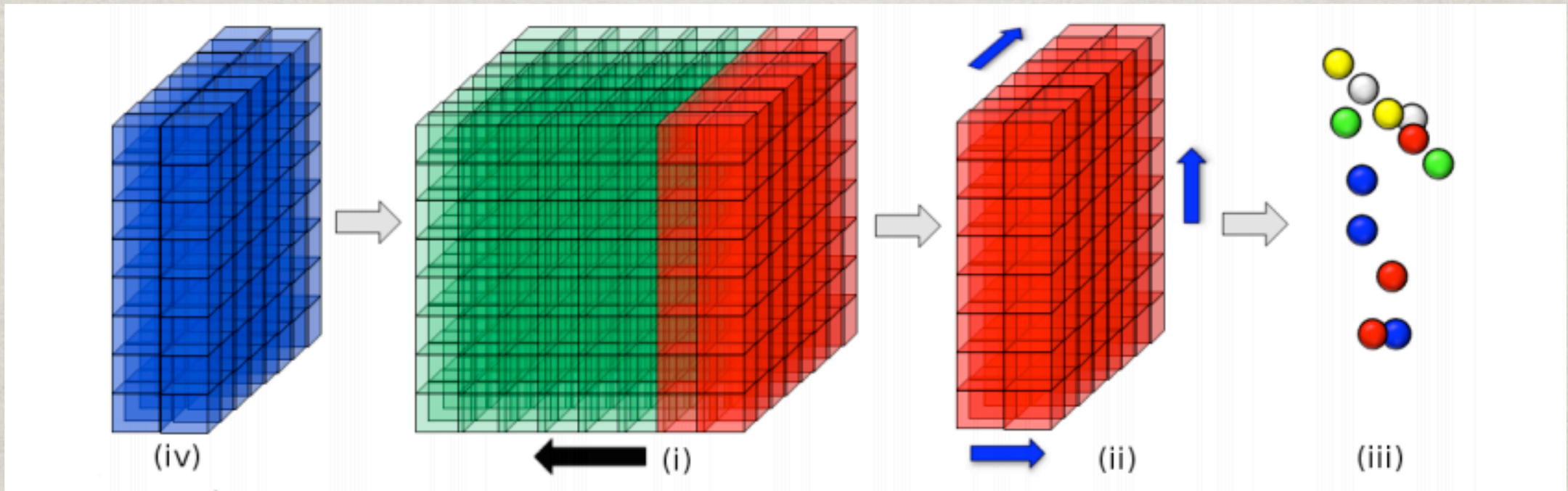


Fig. 2. The four main steps of the Kintinuous algorithm are shown above; (i) Camera motion exceeds movement threshold (black arrow); (ii) Volume slice (red) is raycast (orthogonal directions shown in blue arrows) for point extraction and reset; (iii) Point cloud extracted and fed into greedy mesh triangulation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

Whelan et. al, 2013

KINTINUOUS

Whelan et. al, 2012

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

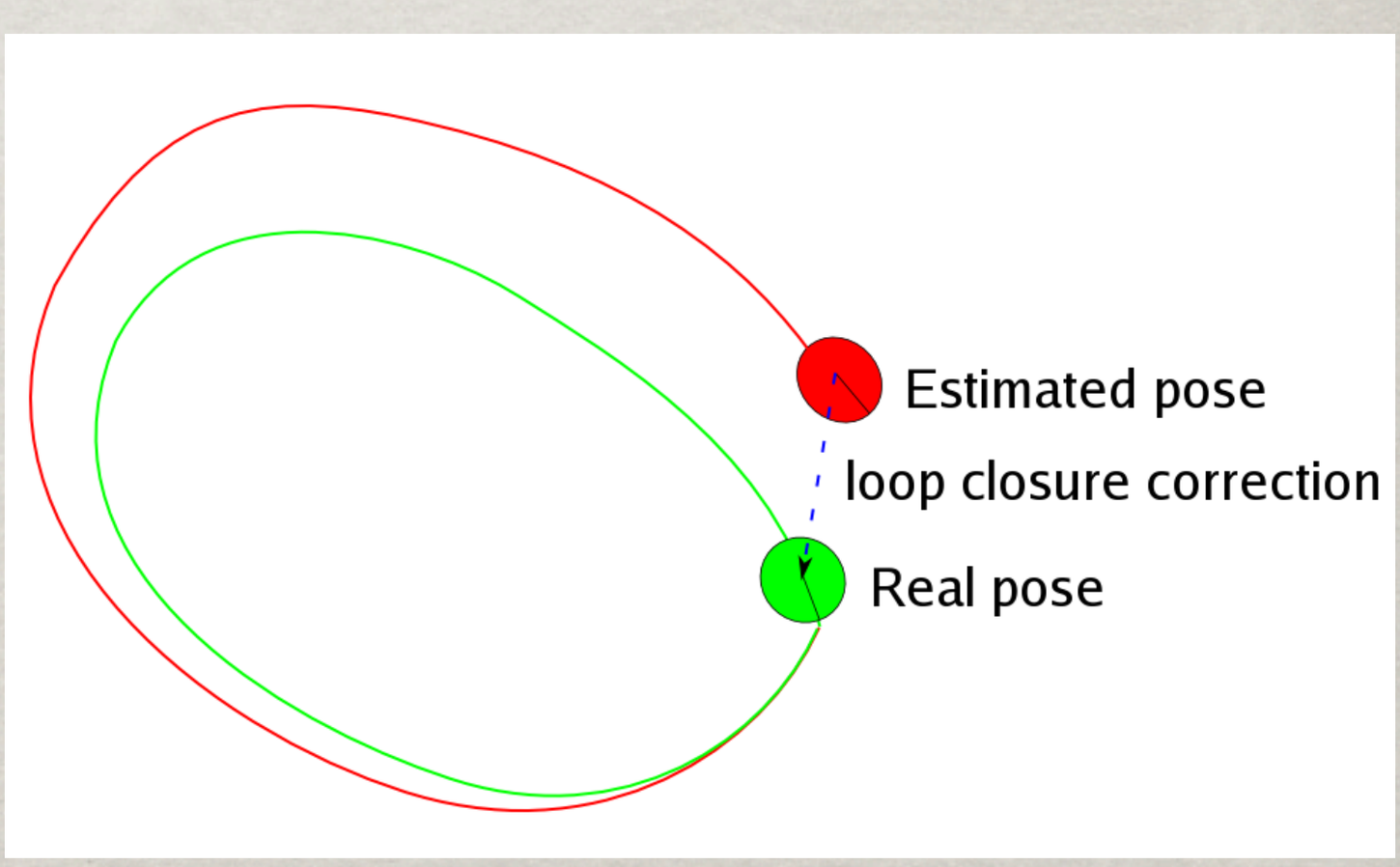
KinectFusion extensions

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

KINTINUOUS LOOP CLOSURE

WHELAN ET. AL, 2013

- 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

KINTINUOUS LOOP CLOSURE

WHELAN ET. AL, 2013

- Place Recognition
  - SURF : Speeded Up Robust Feature
  - Bag of Words (DBoW)
  - Strict validation to eliminate outliers
  - ICP

KINTINUOUS LOOP CLOSURE

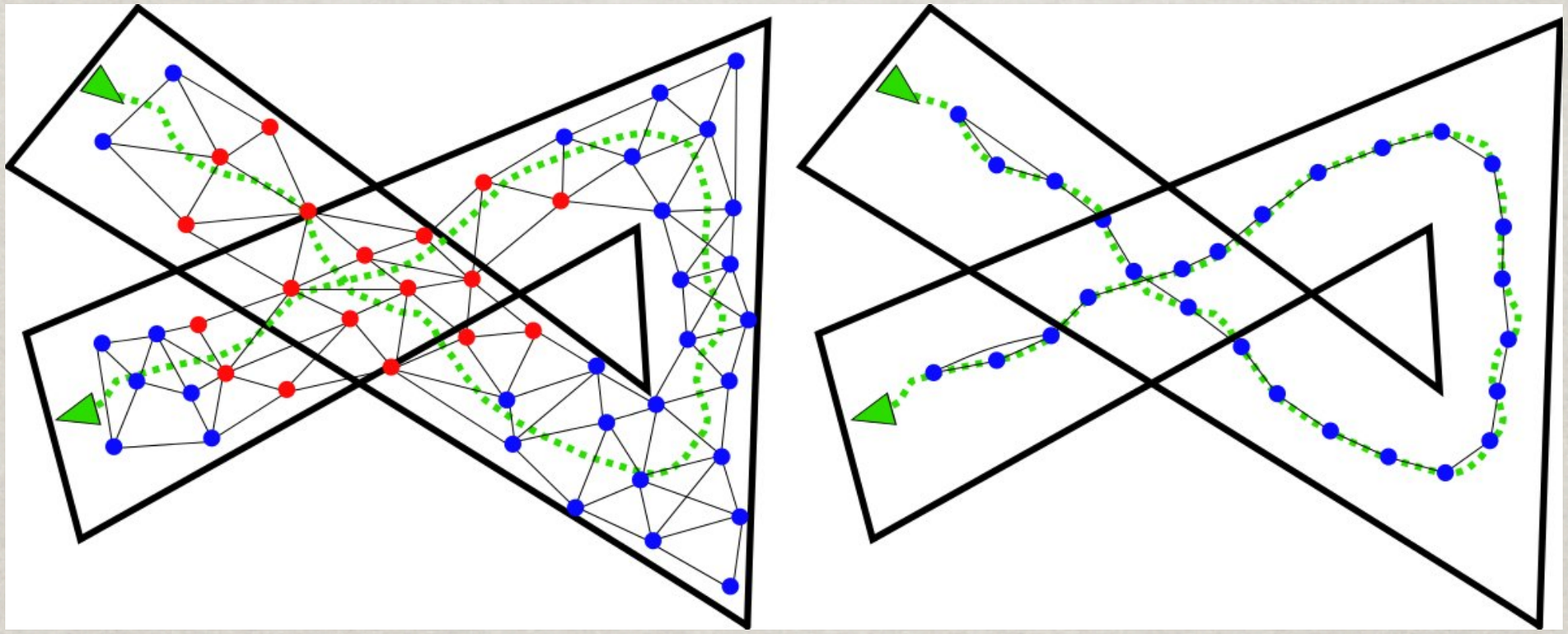
WHELAN ET. AL, 2013

- Space Deformation
  - Deformation graph

KINTINUOUS LOOP CLOSURE

WHELAN ET. AL, 2013

- Space Deformation
  - Deformation graph



Before

After

KINTINUOUS LOOP CLOSURE

WHELAN ET. AL, 2013

- Space Deformation
  - $w_{rot}E_{rot} + w_{reg}E_{reg} + w_{con_P}E_{con_P} + w_{surf}E_{surf}$

KINTINUOUS LOOP CLOSURE

WHELAN ET. AL, 2013

- Space Deformation

$$w_{rot}E_{rot} + w_{reg}E_{reg} + w_{con_P}E_{con_P} + w_{surf}E_{surf}$$

  - Map deformation considering:

    - rigidity, $E_{rot} = \sum_{l} \left\| N_{l_{\mathbf{R}}}^{\top} N_{l_{\mathbf{R}}} - \mathbf{I} \right\|_F^2$

KINTINUOUS LOOP CLOSURE

WHELAN ET. AL, 2013

- 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,

$$E_{reg} = \sum_{l} \sum_{n \in \mathcal{N}(N_l)} \left\| N_{l_{\mathbf{R}}}(N_{n_{\mathbf{g}}} - N_{l_{\mathbf{g}}}) + N_{l_{\mathbf{g}}} + N_{l_{\mathbf{t}}} - (N_{n_{\mathbf{g}}} + N_{n_{\mathbf{t}}}) \right\|_2^2$$

KINTINUOUS LOOP CLOSURE

WHELAN ET. AL, 2013

- 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, $E_{con} = \sum_p \|\phi(\mathbf{v}) - U_p\|_2^2$

KINTINUOUS LOOP CLOSURE

WHELAN ET. AL, 2013

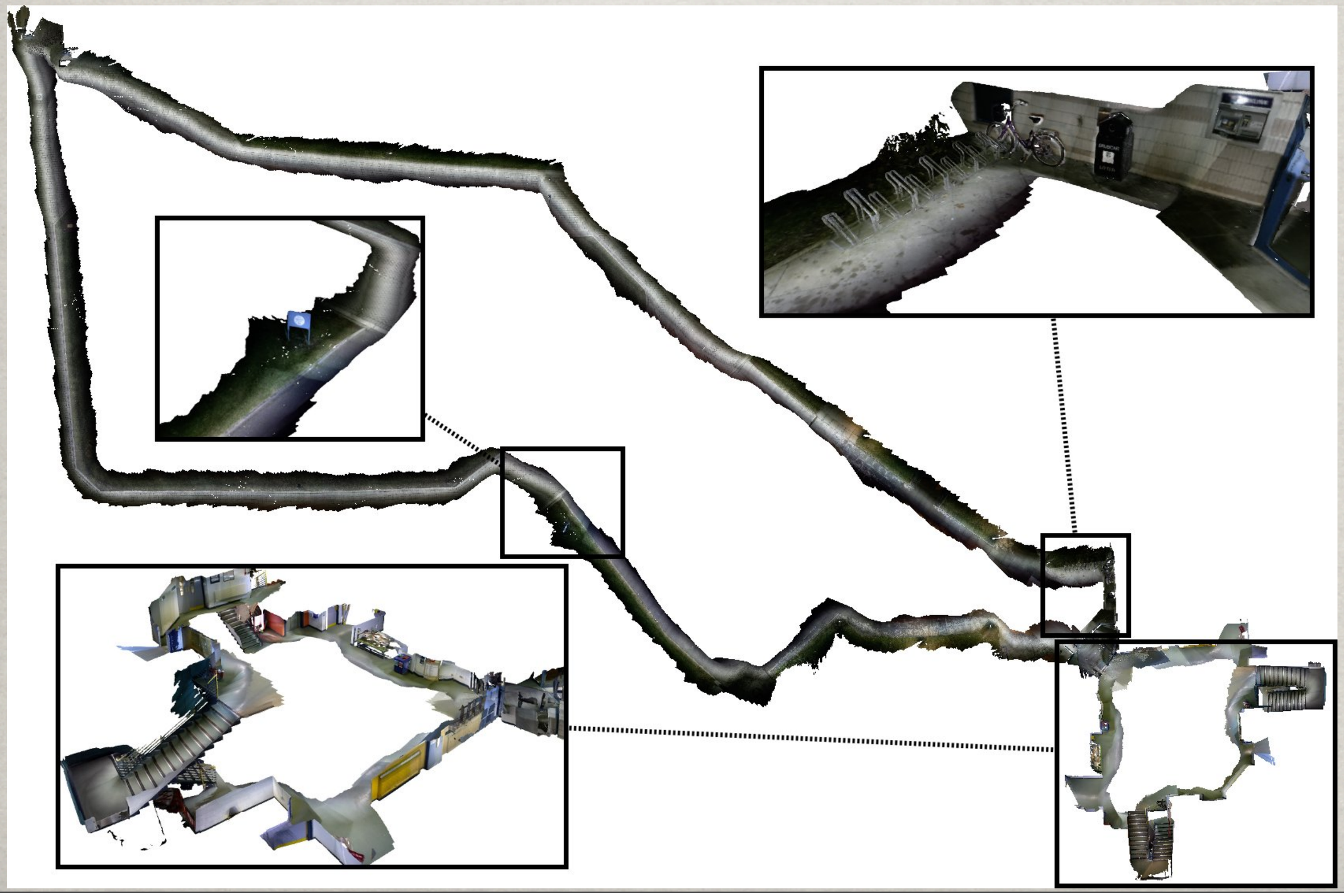
- Space Deformation

$$w_{rot}E_{rot} + w_{reg}E_{reg} + w_{con_P}E_{con_P} + w_{surf}E_{surf}$$

  - Map deformation considering:
    - rigidity,
    - regularization for smooth deformation across the graph,
    - user constraints,
    - surface orientation $E_{surf} = \sum_q \left\| \phi((P_{i_{\mathbf{R}}} V_q) + P_{i_{\mathbf{t}}}) - ((P'_{i_{\mathbf{R}}} V_q) + P'_{i_{\mathbf{t}}}) \right\|_2^2$

kintinuous Loop Closure

Whelan et. al, 2013



KinectFusion extensions

- Circular buffer for space extention [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

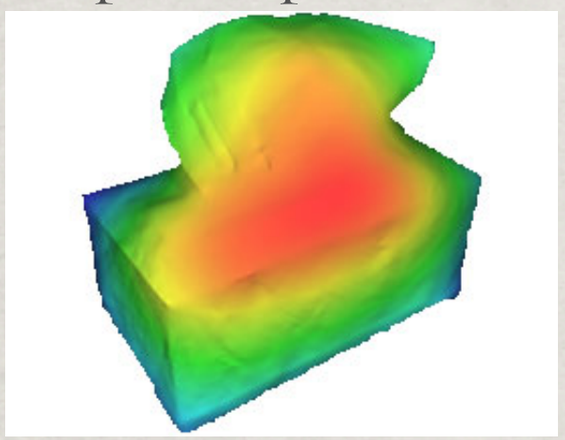
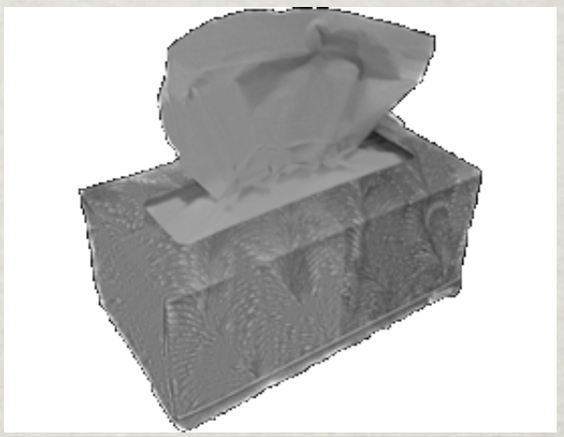
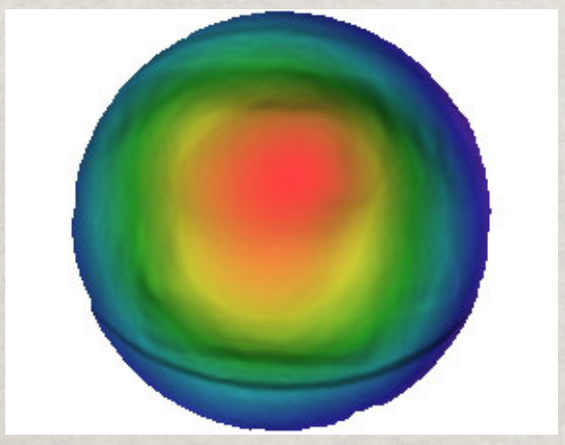
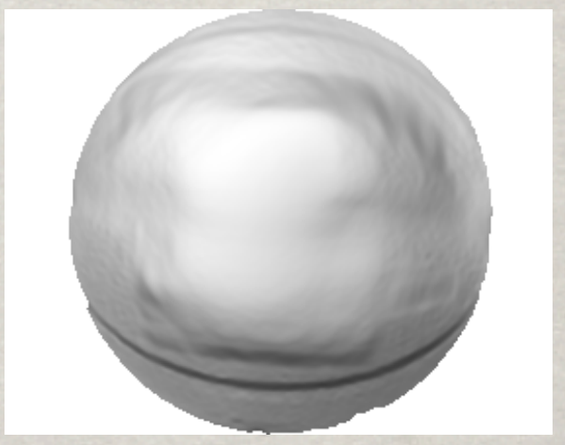
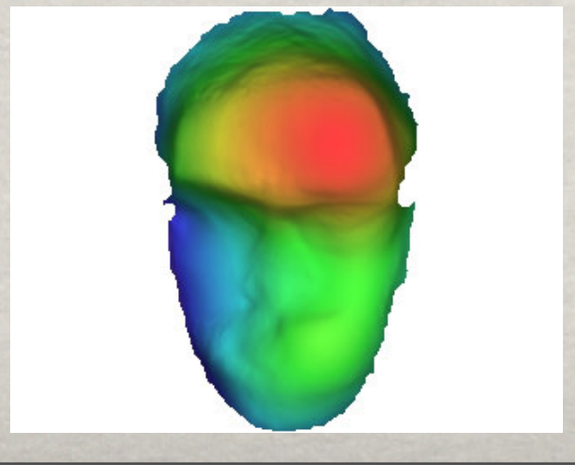
- Texturing by
  - Albedo recovery (variation of SIRFS: shape, reflection, and illumination recovery from shading)

Direct Mapping

SIRFS

Barron & Malik, 2012

From **single** RGB image using priors:

Gray Image	Depth Map	Albedo	Shading Image
			
			
			

3D Self portraits

Li et. al, 2013

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

Direct Mapping

SIRFS

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

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

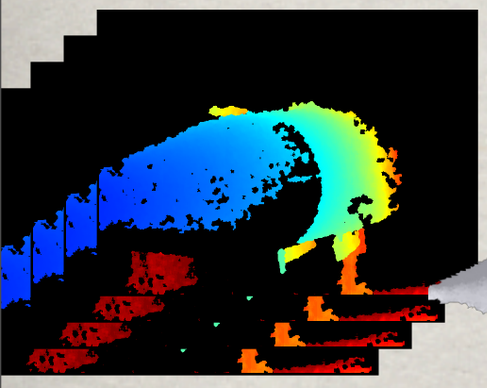
KinectFusion extensions

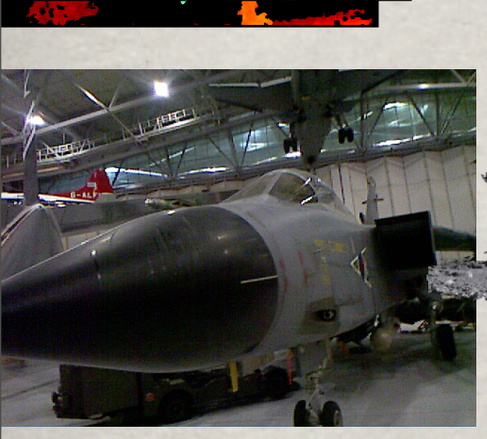
- Circular buffer for space extention [Kintinuous'12]
- Loop closure [Kintinuous Loop Closure '13]
- Texturing [3DSelfPortrait'13]
- Scalable [LargeScale KinectFusion'13]

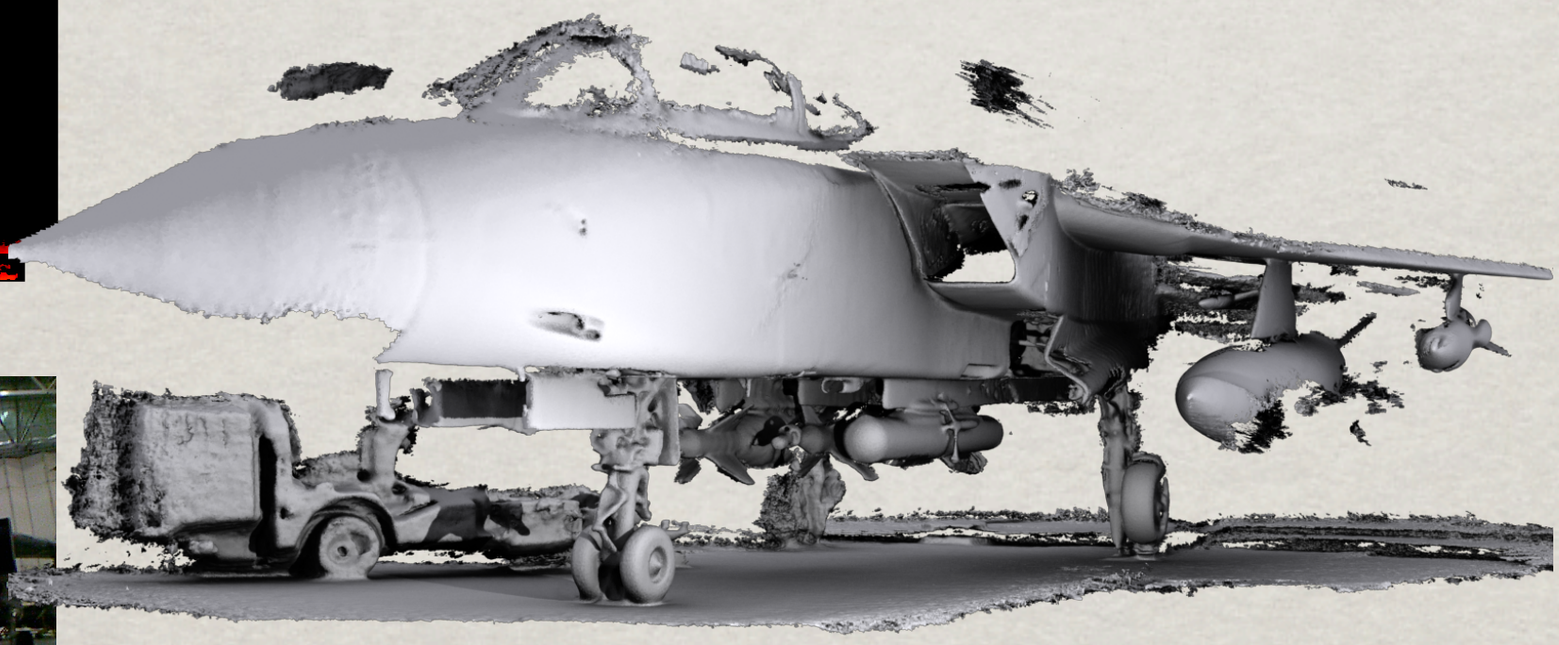
Scalable

Chen et. al, 2013

Same reconstruction algorithm to obtain fine details







Scalable

Chen et. al, 2013

Same reconstruction algorithm

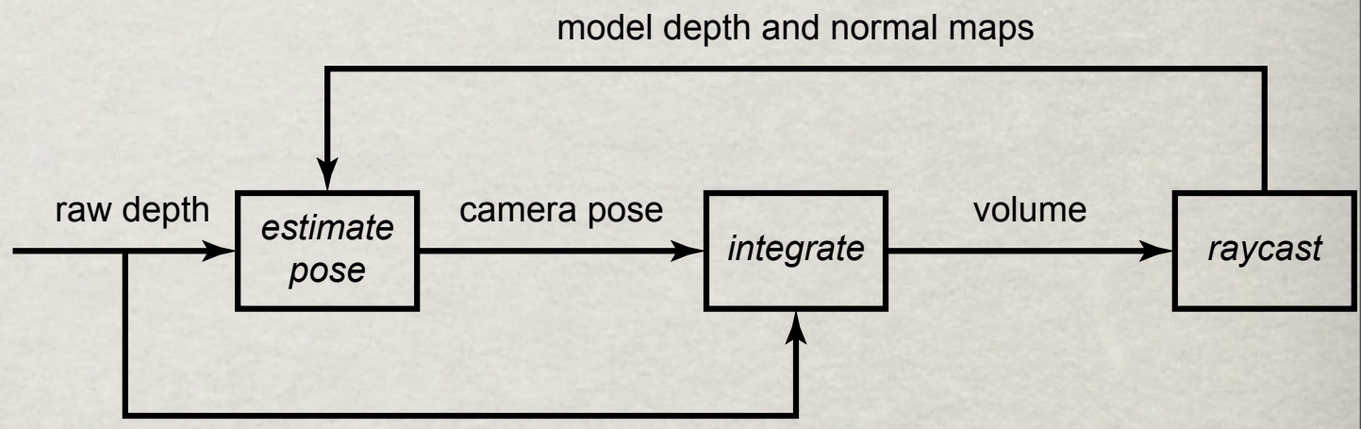


Figure 2: *High level 3D reconstruction pipeline.*

Scalable

Chen et. al, 2013

Same reconstruction algorithm

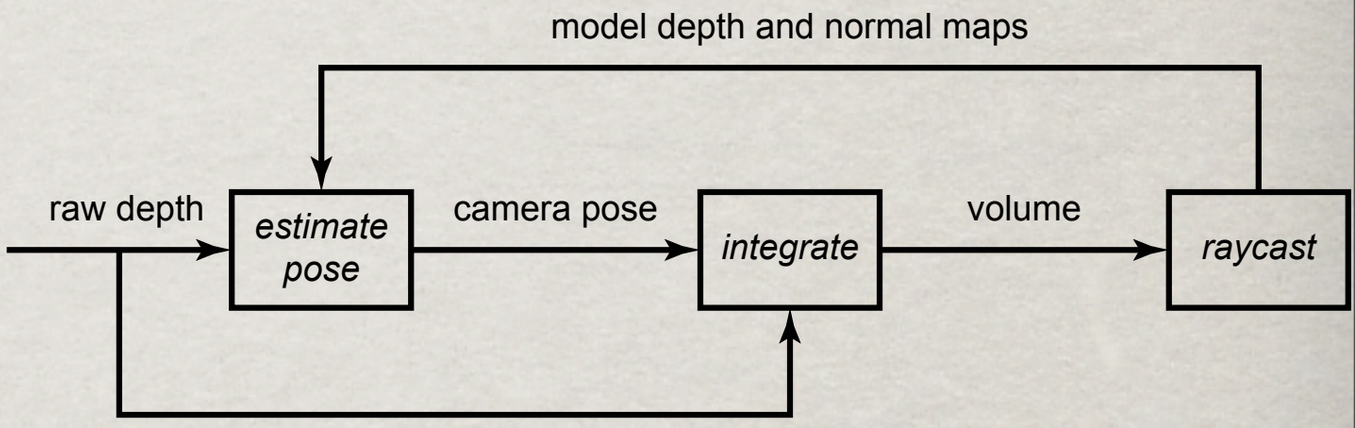


Figure 2: *High level 3D reconstruction pipeline.*

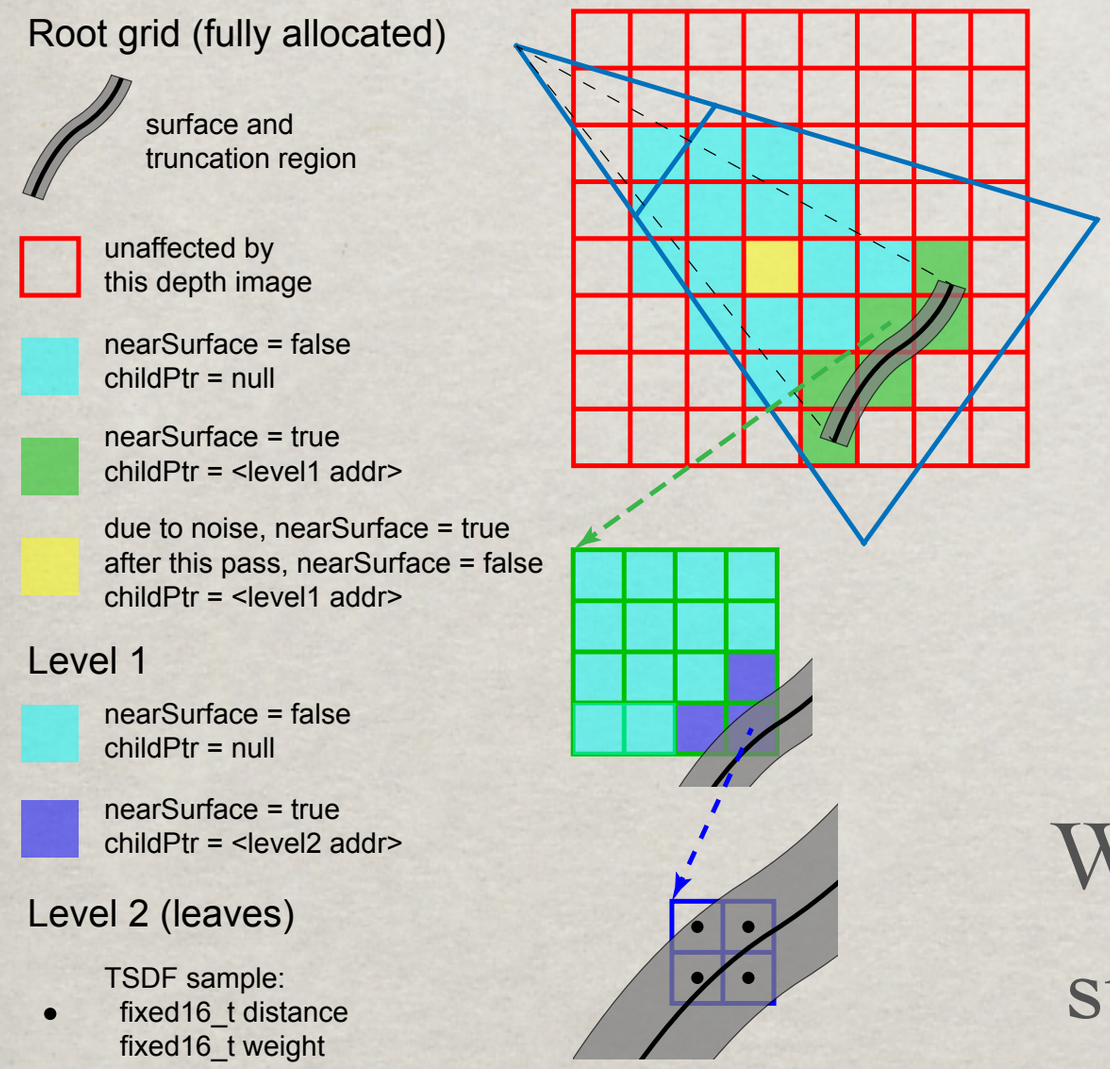
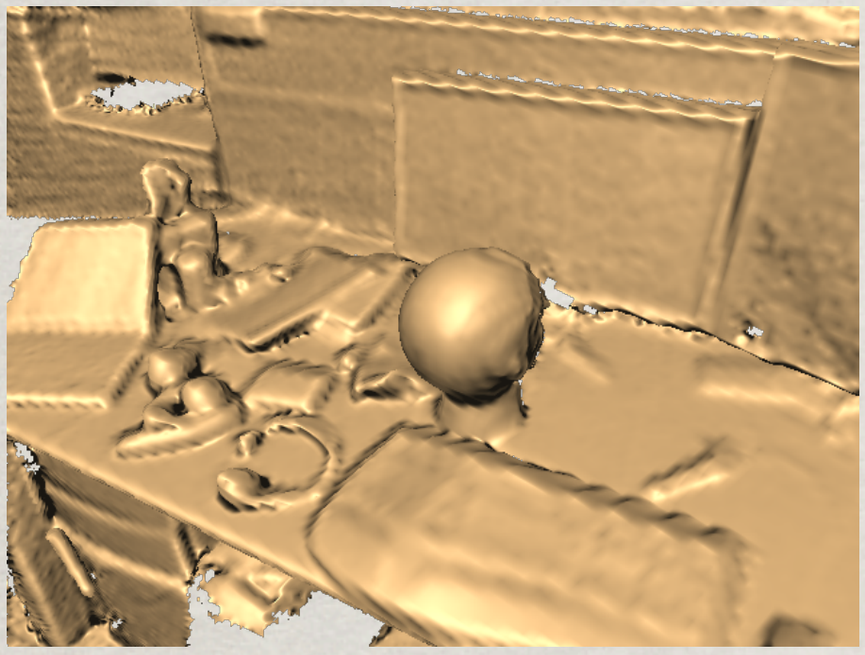


Figure 3: *Logical view of hierarchical data structure.*

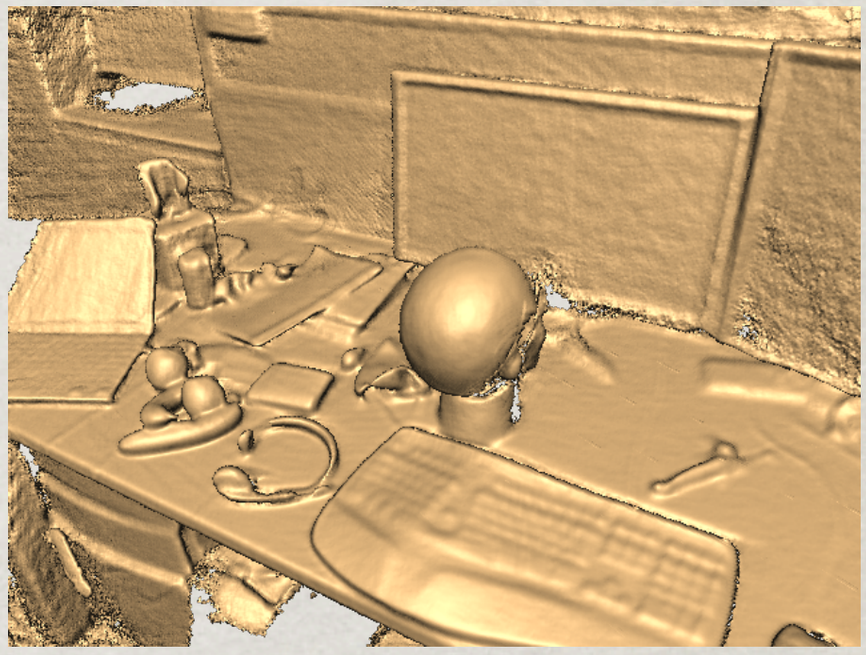
With a smart hierarchical data structure

Scalable

Chen et. al, 2013



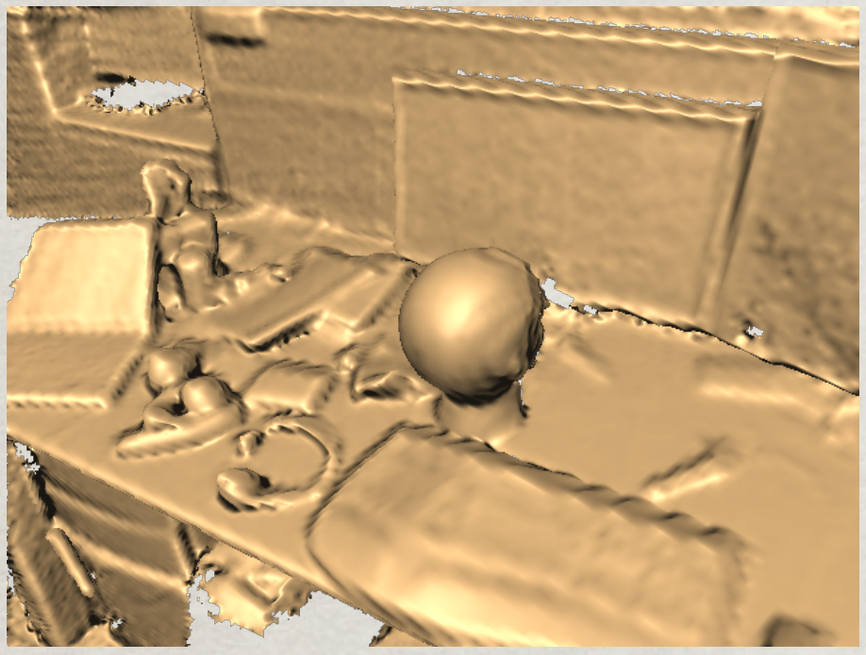
regular grid ($12mm^3$),
512^3, 512 MB



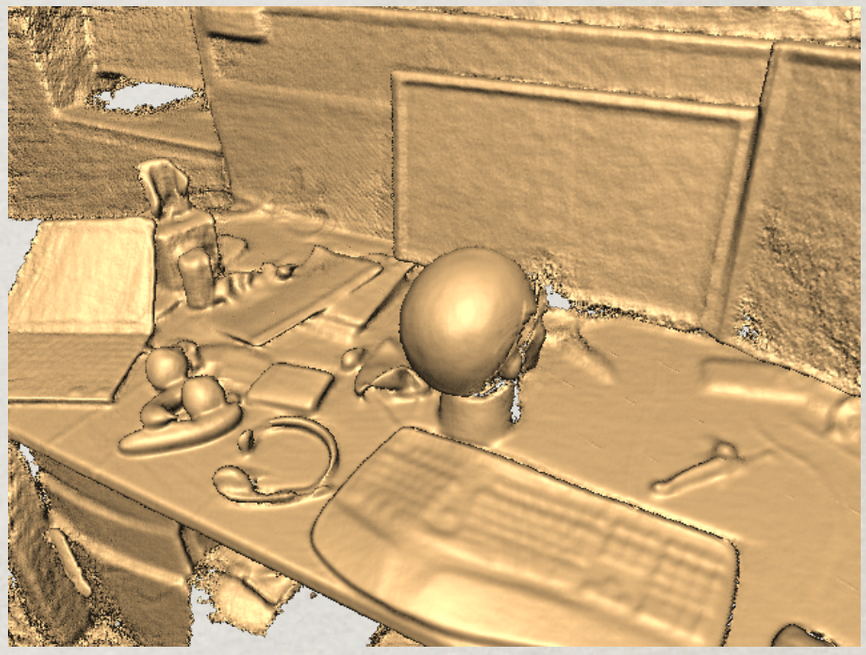
hierarchy ($3mm^3$)
2048^3, 174 MB

Scalable

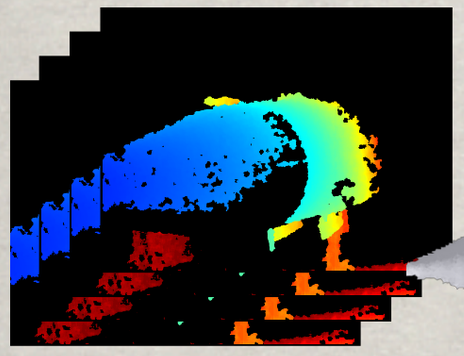
Chen et. al, 2013



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



hierarchy ($3mm^3$)
2048^3, 174 MB



OUTLINE

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.

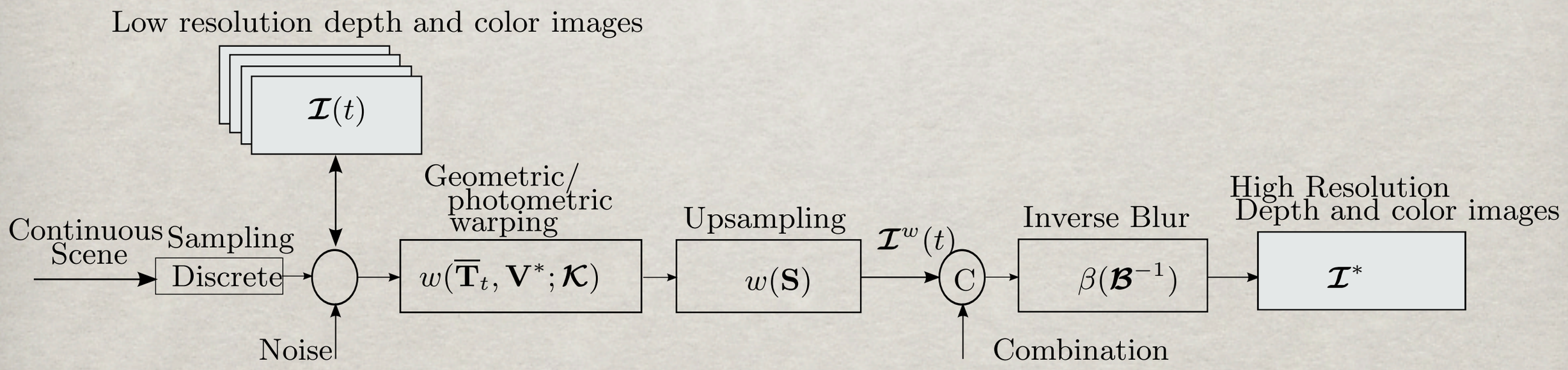
KEYFRAME BASED METHODS

- 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.

Super Resolution SLAM

Meilland et. al, 2013

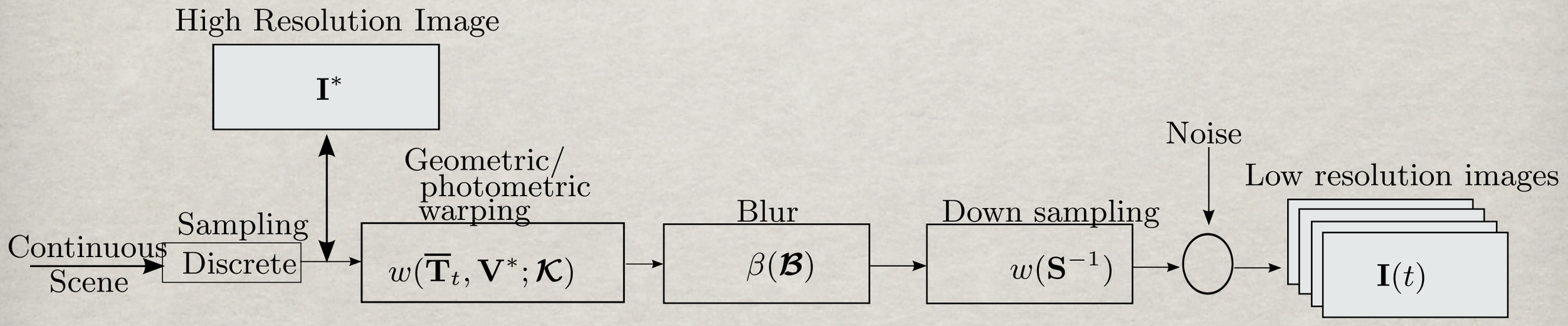
- Low resolution images
  - warped, combined -> to estimate high-resolution target



Super Resolution SLAM

Meilland et. al, 2013

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



Super Resolution SLAM

Meilland et. al, 2013

- 3D Tracking:
  - Optimizing both photometric and depth data:

$$\mathbf{e_I} = \sum_{t=0}^{N} \mathbf{C}(t) \left(\mathbf{I}^* - \mathbf{I} \left(w \left(\widehat{\mathbf{T}}_t \mathbf{T}(\mathbf{x}_t), \mathbf{V}^*, \mathbf{S} \right), t \right) \right)$$

$$\mathbf{e_D} = \sum_{t=0}^{N} \tilde{\mathbf{C}}(t) \left(\mathbf{D}^* - \mathbf{D} \left(w \left(\widehat{\mathbf{T}}_t \mathbf{T}(\mathbf{x}_t), \mathbf{V}^*, \mathbf{S} \right), t \right) \right)$$

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

Super Resolution SLAM

Meilland et. al, 2013

- Weighting:
  - Image distance function:

virtual camera frame

$$\mathbf{C}_{ii}(t) = \left(\left\| (\mathbf{T}_t - \mathbf{T}_o) \overline{\mathbf{v}}_i^* \right\| + \epsilon \right)^{-1}$$

The current frame

Super Resolution SLAM

Meilland et. al, 2013

- Weighting:
  - Image distance function:

virtual camera frame

$$\mathbf{C}_{ii}(t) = \left(\left\|(\mathbf{T}_t - \mathbf{T}_o)\overline{\mathbf{v}}_i^*\right\| + \epsilon\right)^{-1}$$

The current frame

  - Depth weights:

$$\tilde{\mathbf{C}}_{ii}(t) = \frac{fb}{\sigma_d}\mathbf{D}(\mathbf{p}_i, t)^{-2}$$

Super Resolution SLAM

Meilland et. al, 2013

Super-Resolution 3D Tracking and Mapping

Maxime Meilland 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.

Key Frame Fusion

Meilland et. al, 2013

- Combines advantages of two methods:
  - Voxel grid representation
  - Image-based key-frame based

Key Frame Fusion

Meilland et. al, 2013

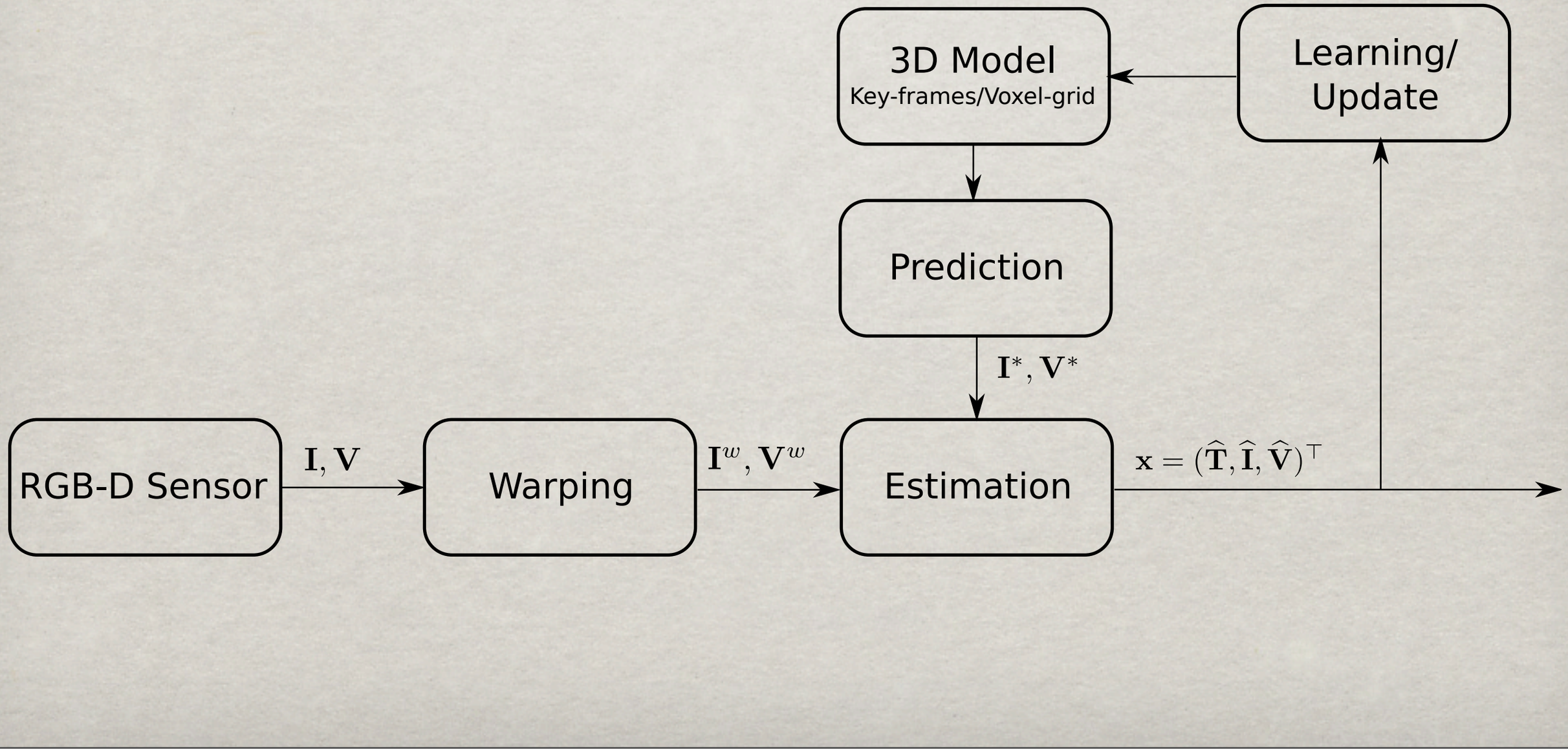
- Voxel grid representation
  - + Easy and efficient defining mathematical operations
  - + Global noise reduction
  - - Hard to handle loop closure
  - - Requires a lot of memory

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

Key Frame Fusion

Meilland et. al, 2013

- Unified system:



Key Frame Fusion

Meilland et. al, 2013

- Pose Estimation:
  - ICP + photometric constraint

$$\mathbf{e}(\mathbf{x}) = \begin{bmatrix} \mathbf{I}\left(w(\hat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \mathbf{V}^*)\right) - \mathbf{I}^*(\mathbf{P}^*) \\ \hat{\mathbf{R}}\mathbf{R}(\mathbf{x})\mathbf{N}^{*\top}\left(\mathbf{V}^\top - \mathbf{\Pi}\hat{\mathbf{T}}\mathbf{T}(\mathbf{x})\mathbf{V}^{*\top}\right)^\top \end{bmatrix}$$

Key Frame Fusion

Meilland et. al, 2013

- Pose Estimation:
  - ICP + photometric constraint

$$\mathbf{e}(\mathbf{x}) = \begin{bmatrix} \mathbf{I}\left(w(\hat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \mathbf{V}^*)\right) - \mathbf{I}^*(\mathbf{P}^*) \\ \hat{\mathbf{R}}\mathbf{R}(\mathbf{x})\mathbf{N}^{*\top}\left(\mathbf{V}^\top - \mathbf{\Pi}\hat{\mathbf{T}}\mathbf{T}(\mathbf{x})\mathbf{V}^{*\top}\right)^\top \end{bmatrix}$$

- Nonlinear! (Gauss Newton)

Key Frame Fusion

Meilland et. al, 2013

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

$$\boldsymbol{\mathcal{S}}^* = \sum_{i=1}^{M} f\left(\boldsymbol{\mathcal{S}}\left(\Gamma\left(\mathbf{P}^*, \mathbf{E}, w(\mathbf{V}_i, \widehat{\mathbf{T}}^{-1}\mathbf{T}_i, \mathbf{K}^*)\right)\right)\right)$$

  - Integrate as in Super Resolution SLAM

Key Frame Fusion

Meilland et. al, 2013

Key Frame Fusion

Meilland et. al, 2013

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 Meilland and Andrew I. Comport

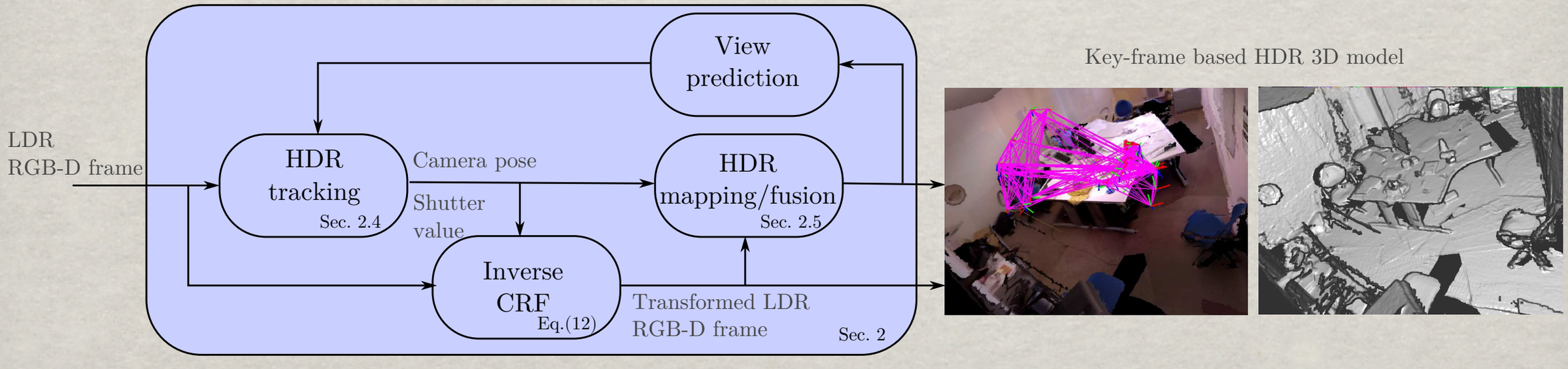
KEYFRAME BASED METHODS

- 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.

HDR SLAM

Meilland et. al, 2013

- Main Contributions:
  - Registering HDR images to 3D scene
  - Real-time HDR reflections on virtual objects



HDR SLAM

Meilland et. al, 2013

- High Dynamic range registration
  - LDR:

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

HDR SLAM

Meilland et. al, 2013

- High Dynamic range registration
  - LDR:

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

  - HDR:

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

HDR SLAM

Meilland et. al, 2013

- High Dynamic range registration
  - LDR:

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

  - HDR:

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

Unknown shutter estimate

Solve [x α]: Nonlinear - Gauss-Newton!

HDR SLAM

Meilland et. al, 2013

- 3D High dynamic range mapping:
  - Key-frame image: I^*_{hdr}
  - Cumulative weights: C^*_{hdr}
- Updated rule between time t – 1 and time t:

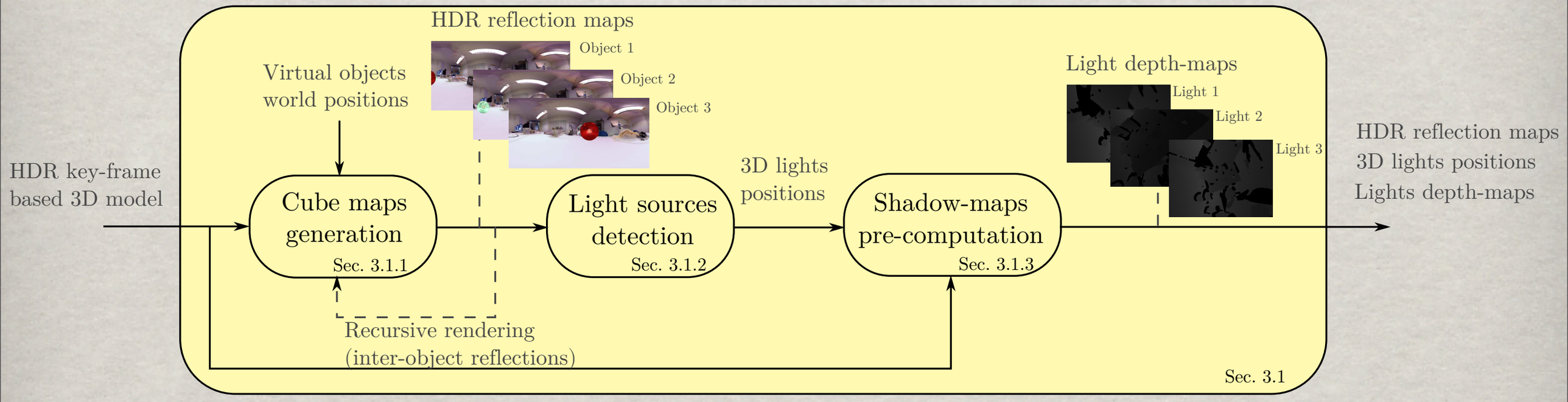
$$\mathbf{C}^*_{hdr}(\mathbf{p},t) \leftarrow \mathbf{C}^*_{hdr}(\mathbf{p},t-1) + f(\mathbf{I}^w(\mathbf{p},t))$$

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

HDR SLAM

Meilland et. al, 2013

- Real-time HDR reflections on virtual objects



HDR SLAM

Meilland et. al, 2013

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

OUTLINE

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

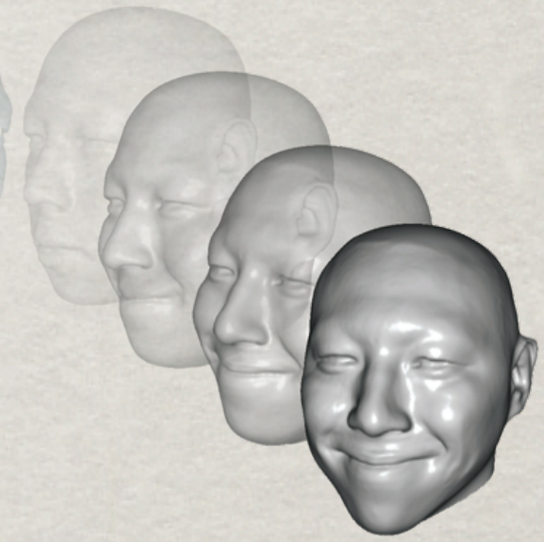
Li et. al, 2009

Reconstruction from a dynamic scene

input scans

reconstruction

input scans



reconstruction

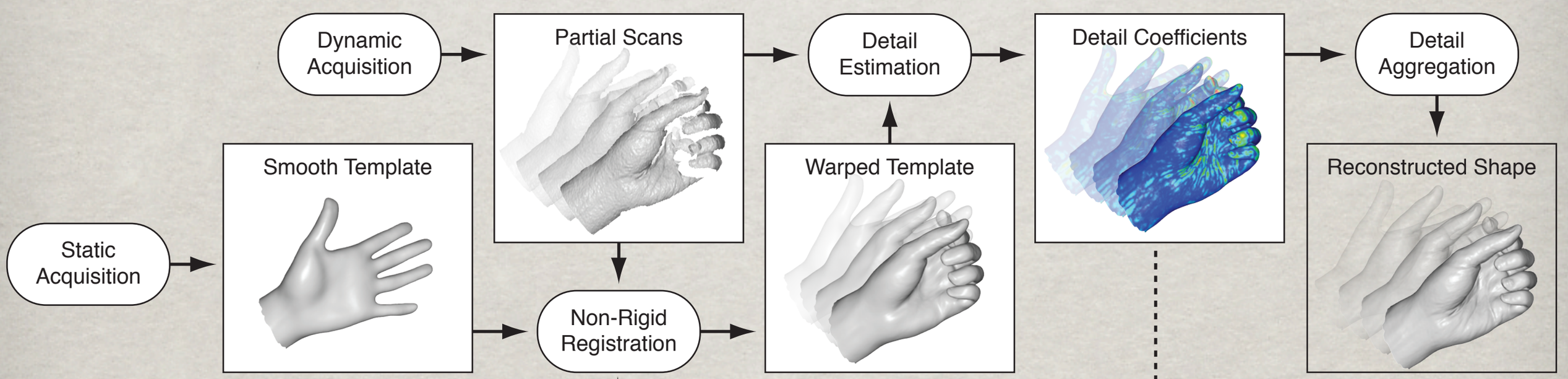
input scans

reconstruction

Robust Single-View Geometry and Motion Reconstruction

Li et. al, 2009

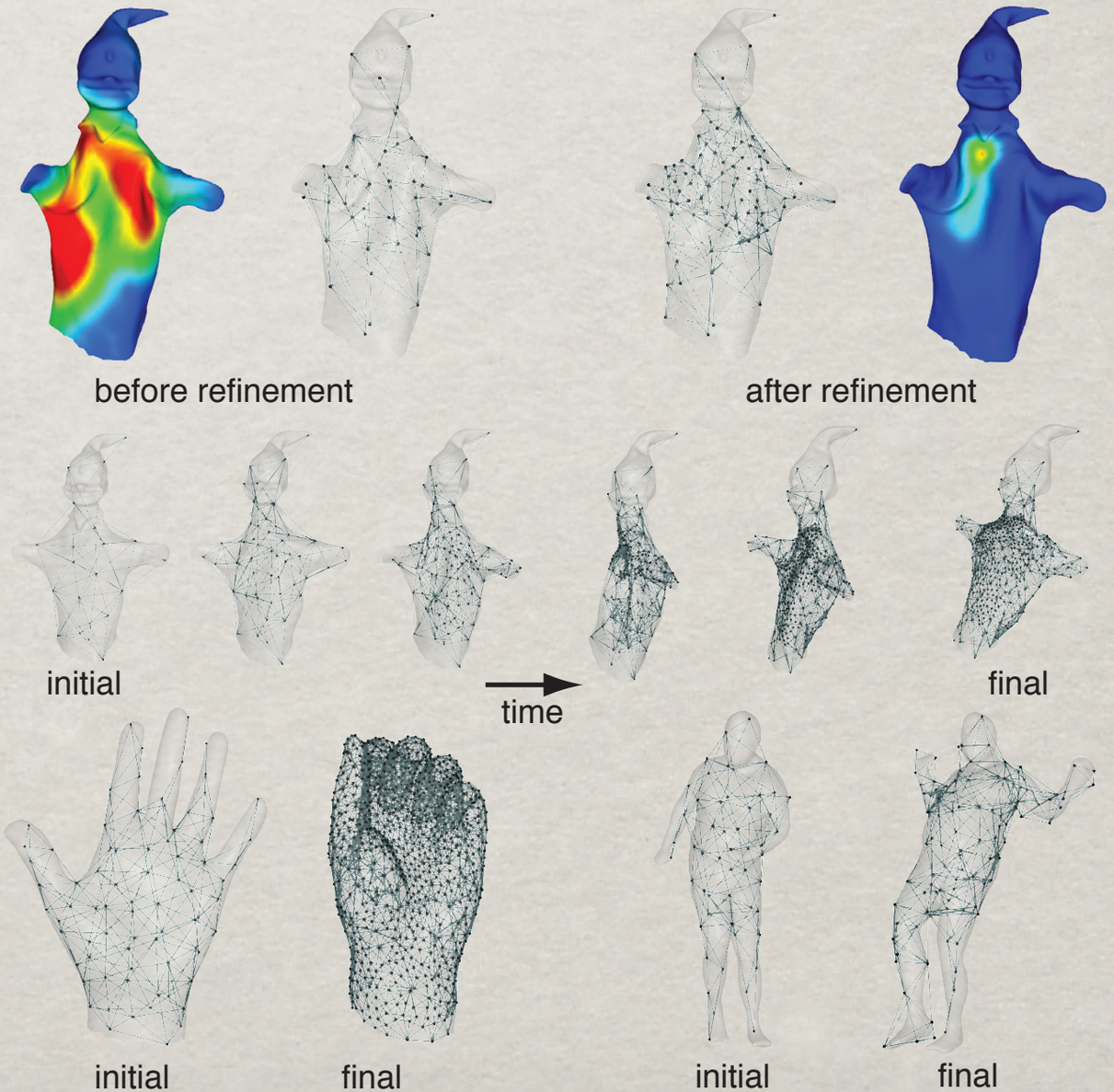
Pipeline:



Robust Single-View Geometry and Motion Reconstruction

Li et. al, 2009

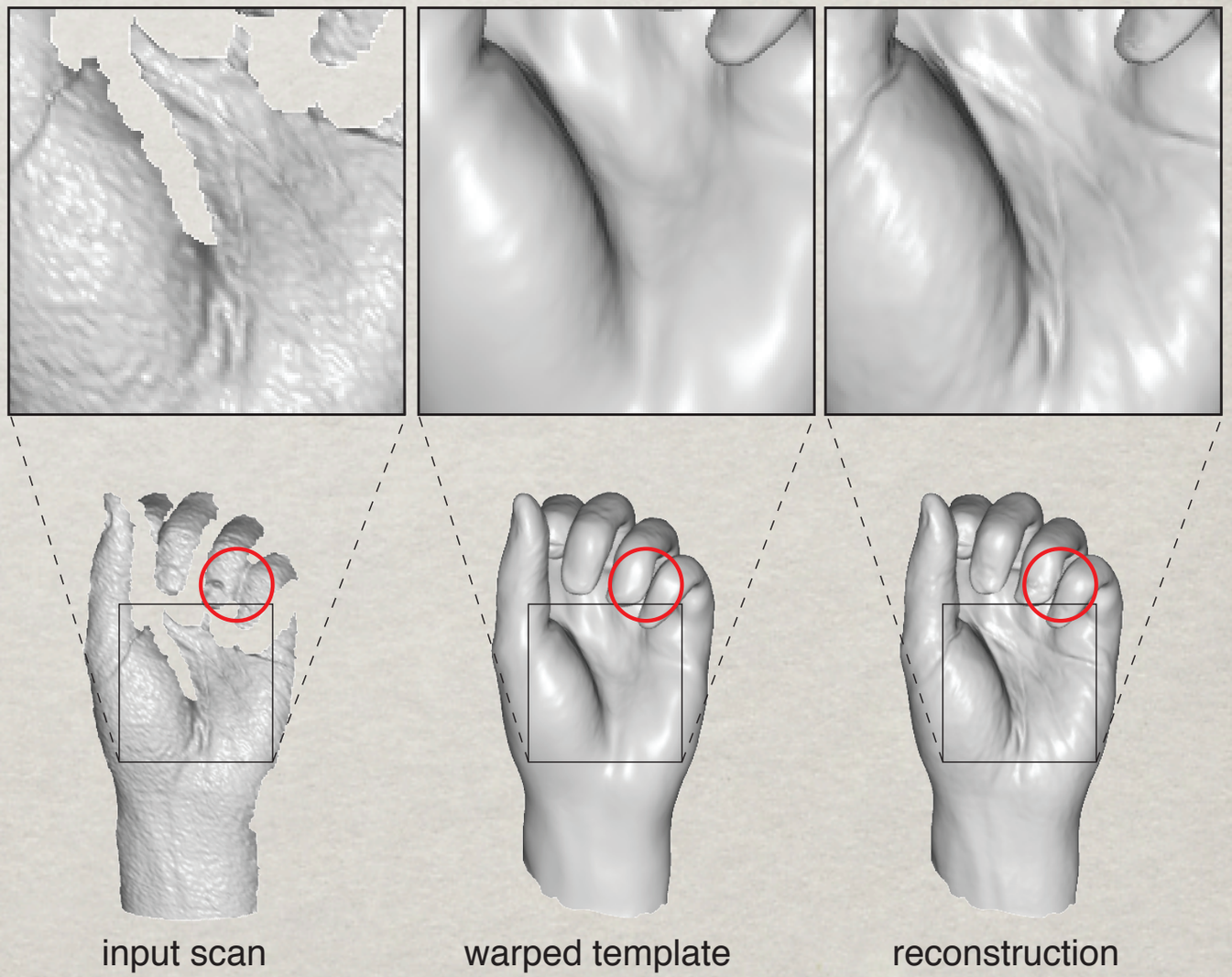
Align the template to the each frame by creating **deformation graph**



Robust Single-View Geometry and Motion Reconstruction

Li et. al, 2009

Aggregating detail over temporally adjacent frames propagates detail



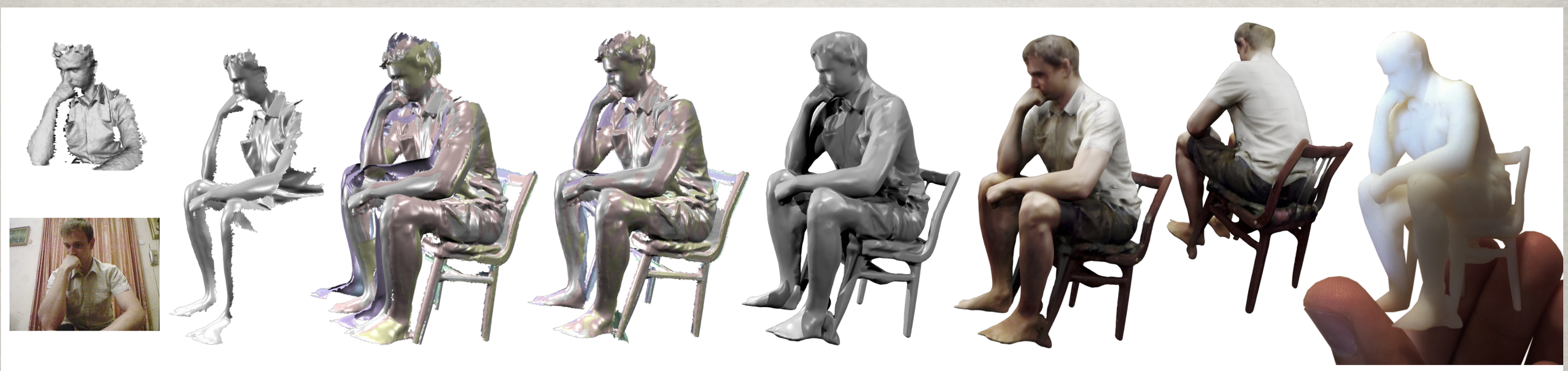
Deformable & non-rigid alignment

- Robust Single-View Geometry And Motion Reconstruction.
- 3D Self-Portraits.
- Elastic Fragments for Dense Scene Reconstruction.

3D Self portraits

Li et. al, 2013

Pipeline:



raw input data | per-view fusion | rigid alignment | non-rigid alignment | output reconstruction | textured reconstruction | 3D print

3D Self portraits

Li et. al, 2013

Aggregation & per-view fusion of data

For ~8 views

per-view aggregation of raw frames

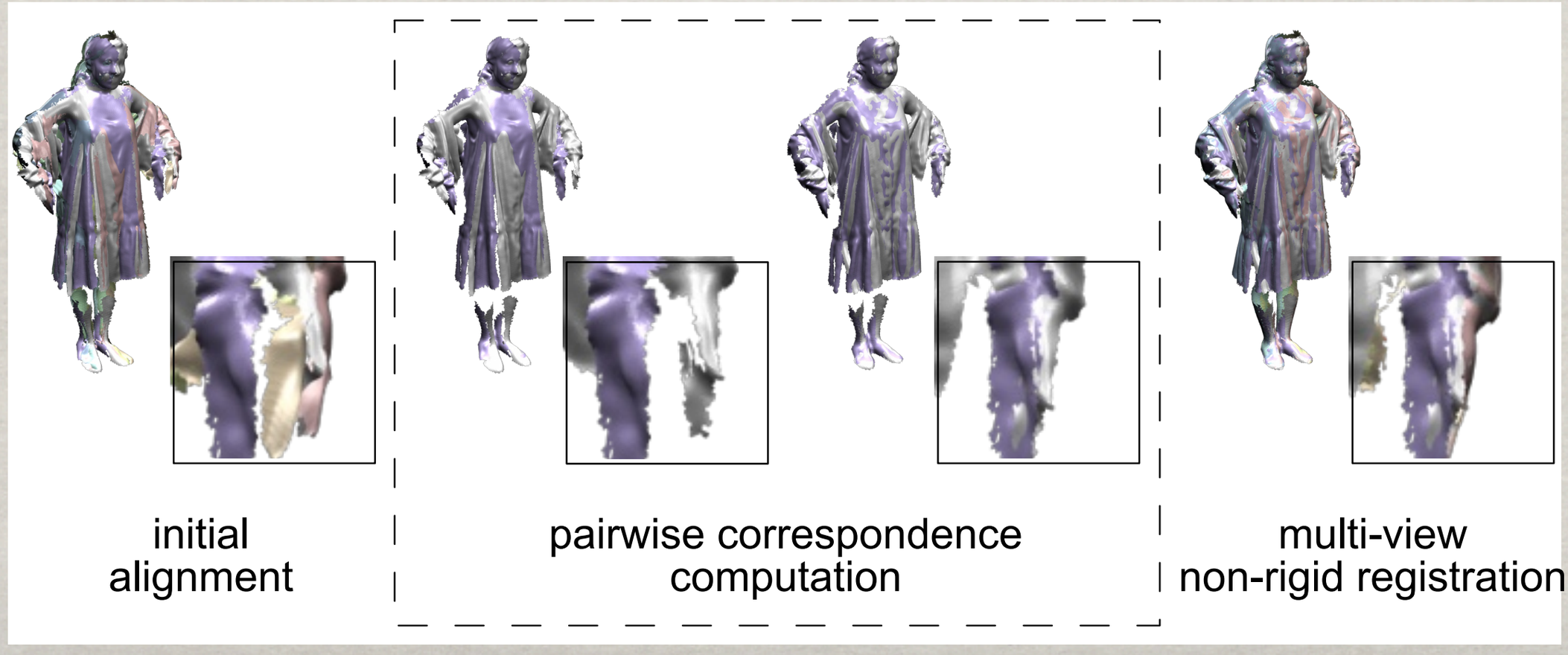
per-view fusion

3D Self portraits

Li et. al, 2013

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.

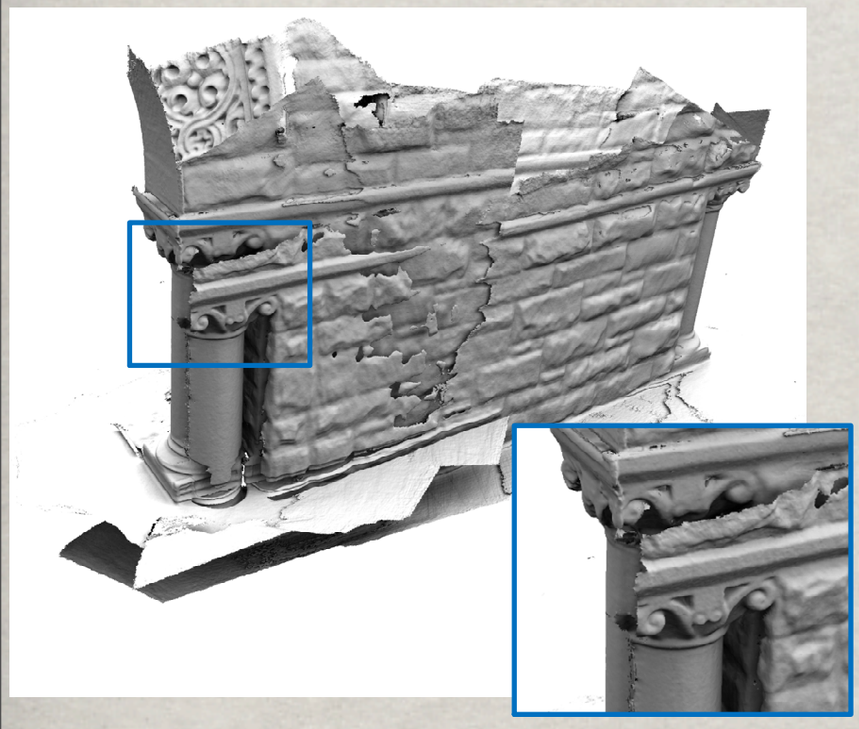
Elastic Fragments for Dense Scene Reconstruction

ZHOU ET. AL, 2013

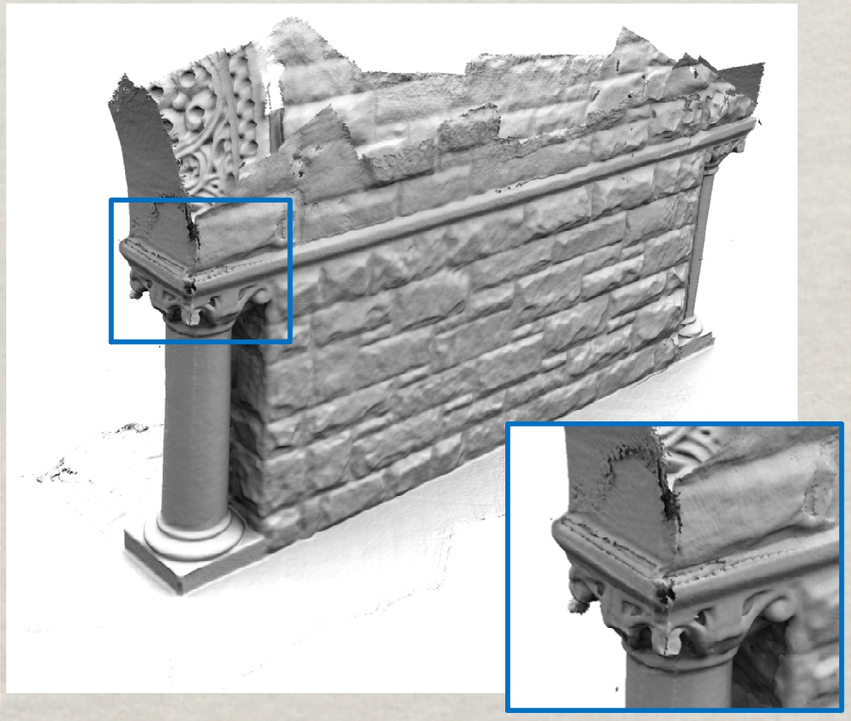
Handles the deformations caused by:

High-frequency errors,

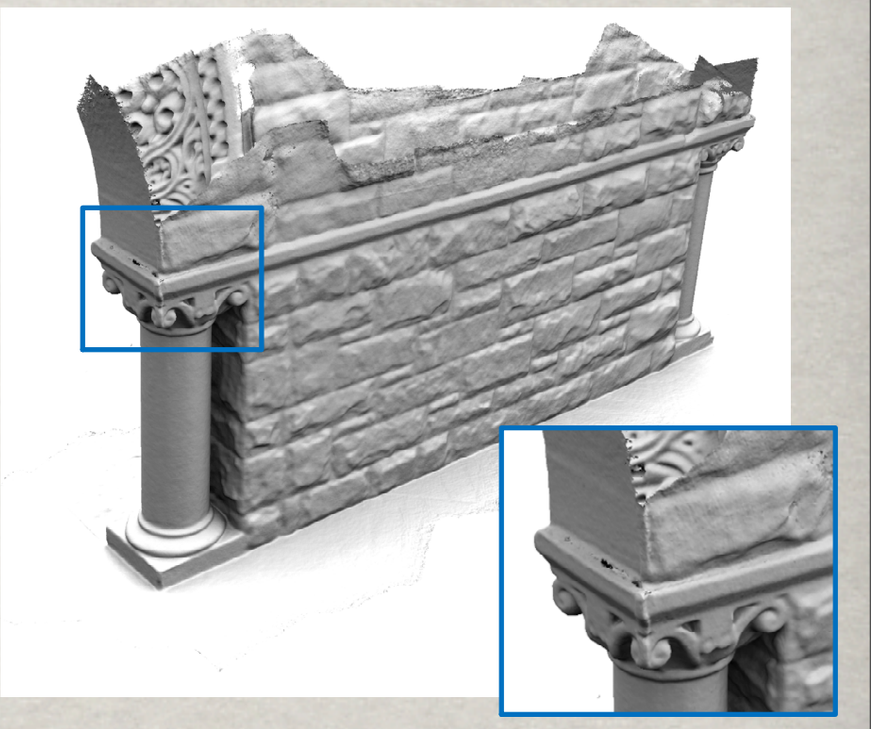
Low-frequency distortion



(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

Elastic Fragments for Dense Scene Reconstruction

ZHOU ET. AL, 2013

- Fragment construction
  - Partition into k(50 or 100) segments
  - Use frame-to-model registration & integration as in the kinect fusion

Elastic Fragments for Dense Scene Reconstruction

ZHOU ET. AL, 2013

- Initial alignment between fragments
- SLAM for rough alignment
- ICP stability to validate correspondences

Elastic Fragments for Dense Scene Reconstruction

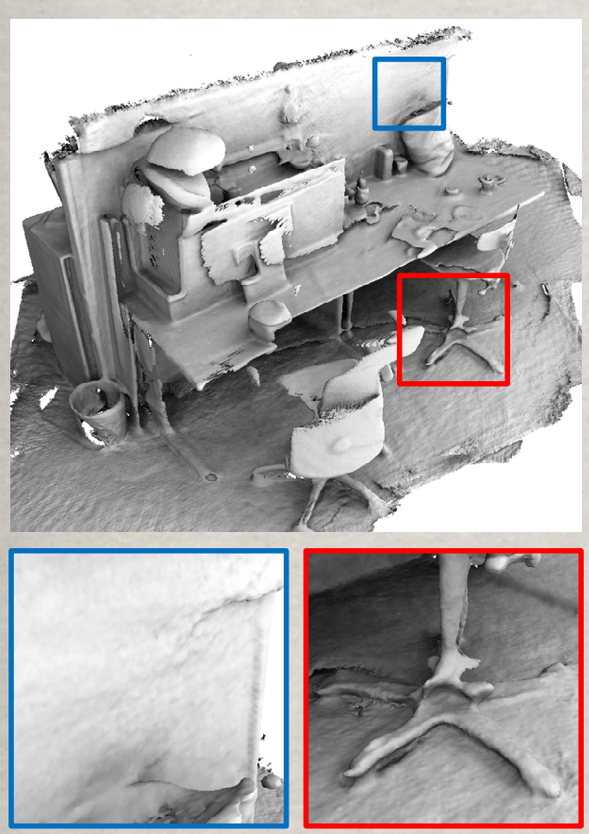
ZHOU ET. AL, 2013

- Elastic Registration by iteratively optimizing:
  - Distances between fragments
  - Elastic strain energy to preserve shape of each fragment

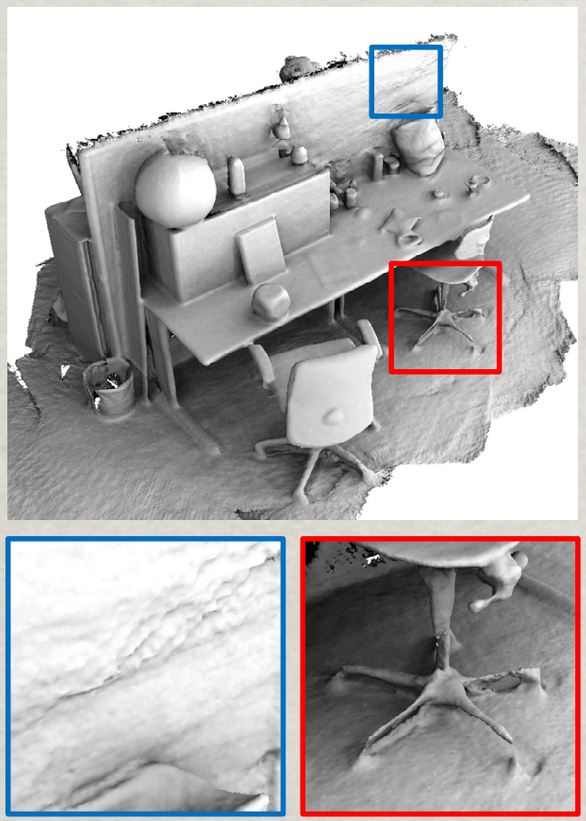
Elastic Fragments for Dense Scene Reconstruction

ZHOU ET. AL, 2013

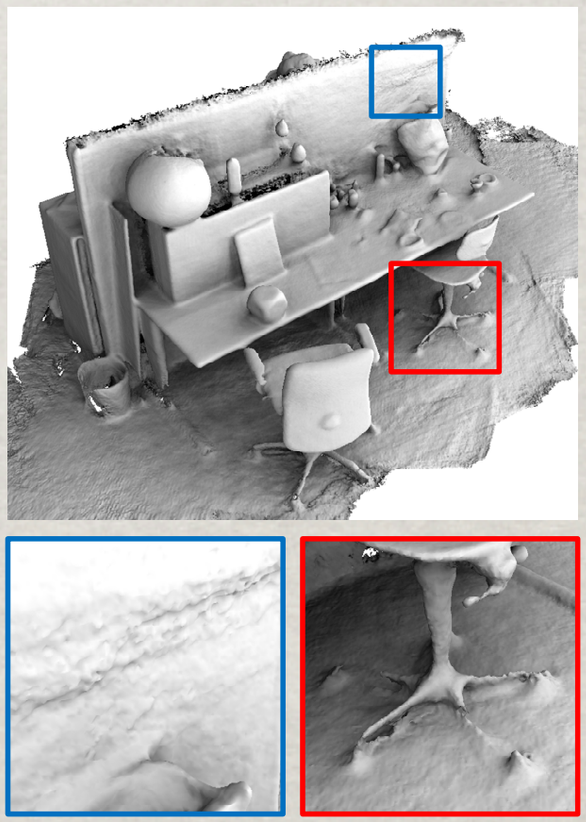
Comparison to rigid methods



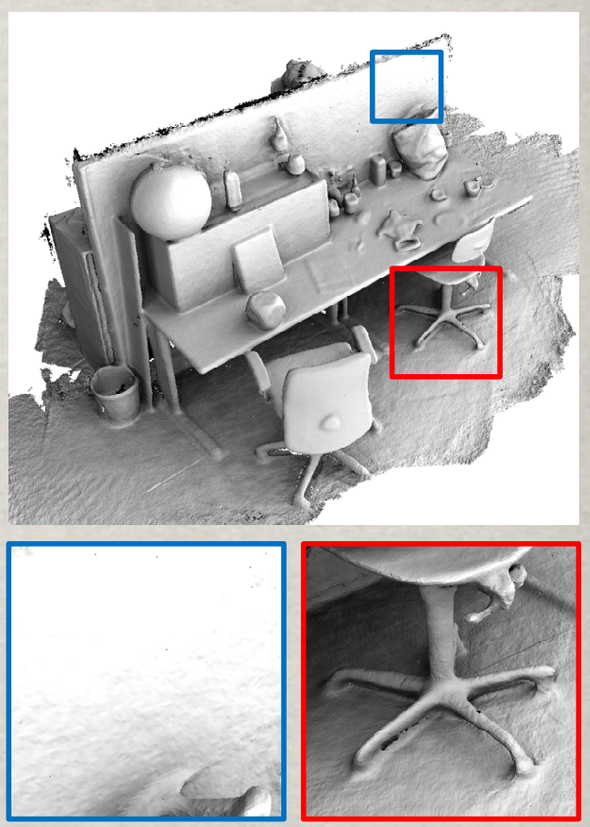
(a) Extended KinectFusion



(b) Zhou and Koltun



(c) Mocap trajectory



(d) Our approach

REFERENCES

- Chen, J., Bautembach, D., and Izadi, S. 2013. Scalable real-time volumetric surface reconstruction. ACM Transactions on Graphics (TOG) 32, 4, 113.
- Izadi, S., Newcombe, R.A., Kim, D., et al. 2011. KinectFusion: real-time dynamic 3D surface reconstruction and interaction. SIGGRAPH '11: SIGGRAPH 2011 Talks, 1.
- Leonard, J. and Robotics, M. 2012. Kintinuous: Spatially Extended KinectFusion.
- Li, H., Vouga, E., Gudym, A., Luo, L., Barron, J.T., and Gusev, G. 2013. 3D Self-Portraits. ACM Transactions on Graphics 32, 6.
- Meilland, M. and Comport, A.I. 2013a. On unifying key-frame and voxel-based dense visual SLAM at large scales. IEEE, 3677–3683.
- Meilland, M. and Comport, A.I. 2013b. Super-resolution 3D tracking and mapping. IEEE, 5717–5723.
- Meilland, M., Barat, C., and Comport, A. 2013. 3D High Dynamic Range dense visual SLAM and its application to real-time object re-lighting. Mixed and Augmented Reality (ISMAR), 2013 IEEE International Symposium on, 143–152.
- Newcombe, R.A., Izadi, S., Hilliges, O., et al. 2011. KinectFusion: Real-time dense surface mapping and tracking. IEEE Computer Society, 127–136.
- Whelan, T., Kaess, M., and Leonard, J. 2013. Deformation-based loop closure for large scale dense rgb-d slam. IEEE/RSJ Intl Conf
- Zhou, Q.Y., Miller, S., and Koltun, V. 2013. Elastic Fragments for Dense Scene Reconstruction. ICCV.

RAY CASTING

[IZADI ET AL., 2011]

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

```
1:  for each pixel u ∈ output image in parallel do
2:      ray^start ← back project [u, 0]; convert to grid pos
3:      ray^next ← back project [u, 1]; convert to grid pos
4:      ray^dir ← normalize (ray^next − ray^start)
5:      ray^len ← 0
6:      g ← first voxel along ray^dir
7:      m ← convert global mesh vertex to grid pos
8:      m^dist ← ||ray^start − m||
9:      while voxel g within volume bounds do
10:         ray^len ← ray^len + 1
11:         g^prev ← g
12:         g ← traverse next voxel along ray^dir
13:         if zero crossing from g to g^prev then
14:             p ← extract trilinear interpolated grid position
15:             v ← convert p from grid to global 3D position
16:             n ← extract surface gradient as ∇tsdf(p)
17:             shade pixel for oriented point (v, n) or
18:             follow secondary ray (shadows, reflections, etc)
19:         if ray^len > m^dist then
20:             shade pixel using inputed mesh maps or
21:             follow secondary ray (shadows, reflections, etc)
```