A Lightning Tour of Dense Real-Time Reconstruction Systems

Two key enabling technologies

Structured light projection

General purpose GPU computing

Microsoft Kinect

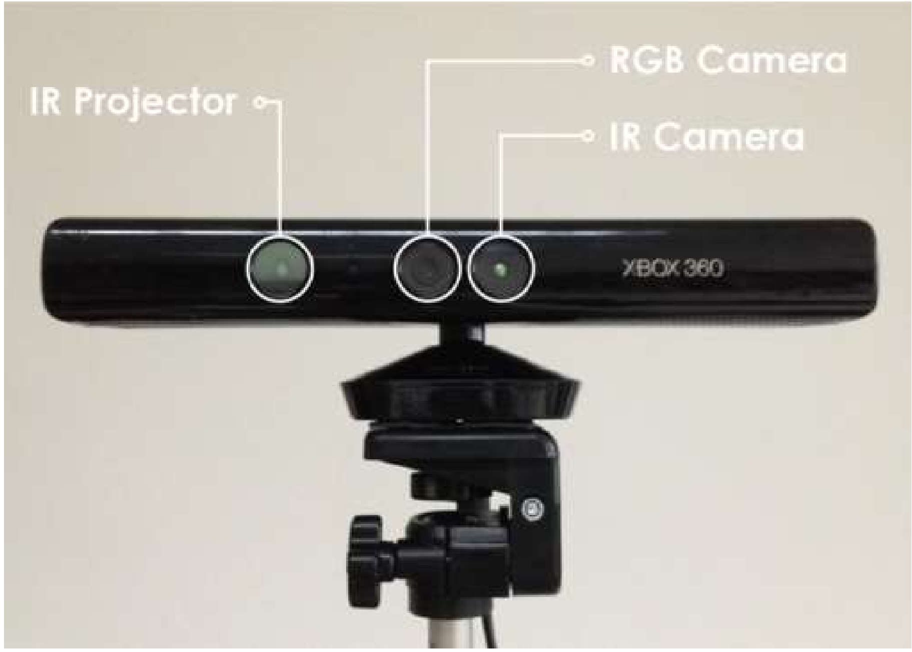
- Released Nov 2010

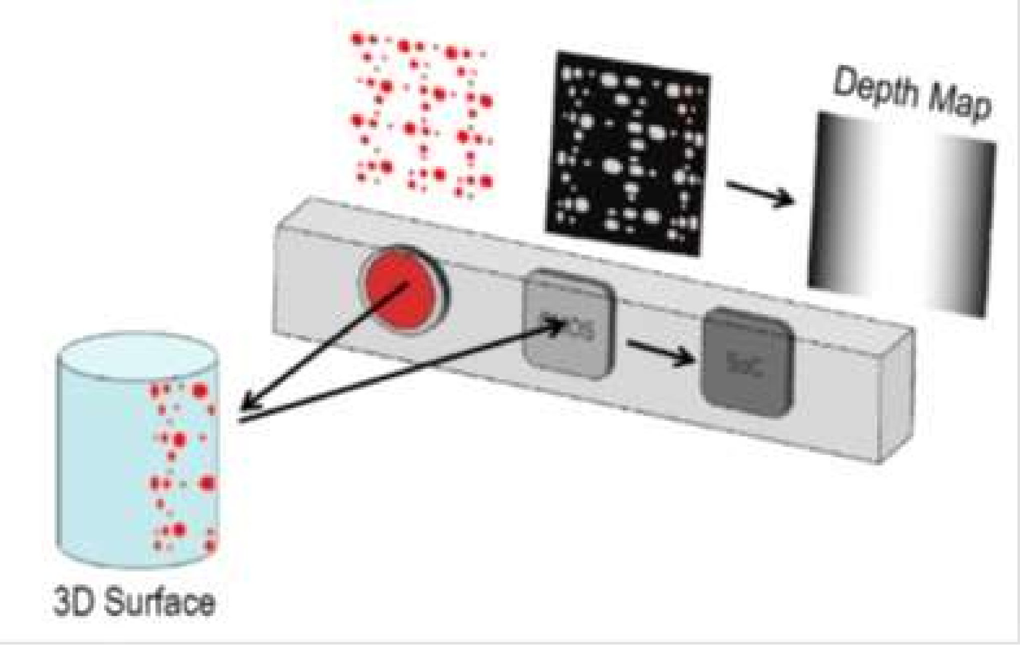
- Provides *dense, real-time RGB + D*:
  - 640 x 480 RGB @ 30 Hz
  - 640 x 480 IR (=> depth) @ 30 Hz

- Cost: ~$100

Microsoft Kinect: Principle of operation

Main idea: *Dense depth* via *structured light projection (active stereo)*

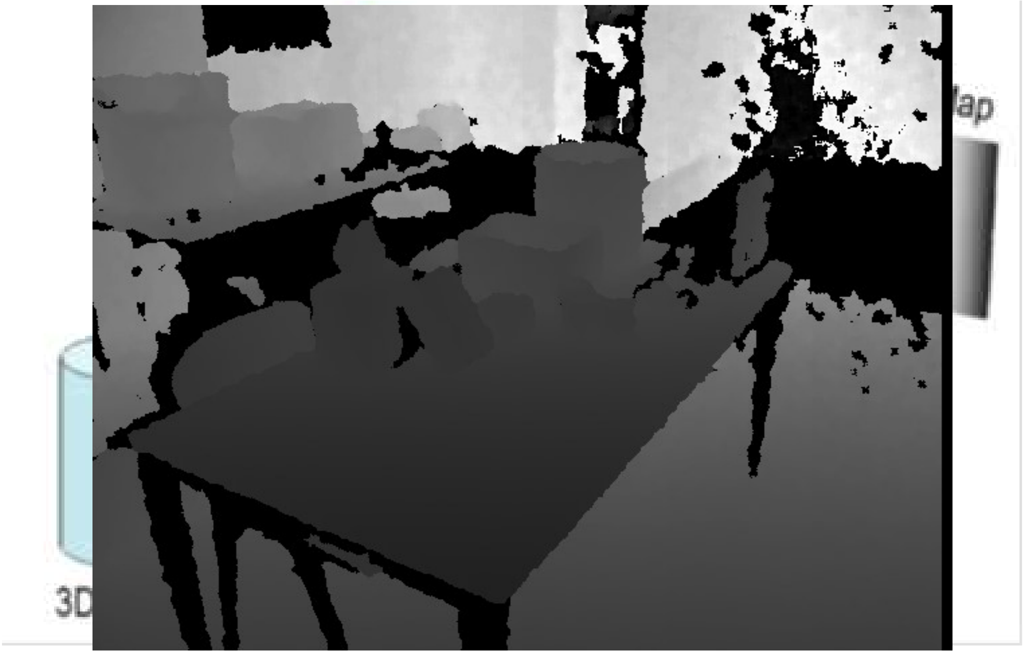


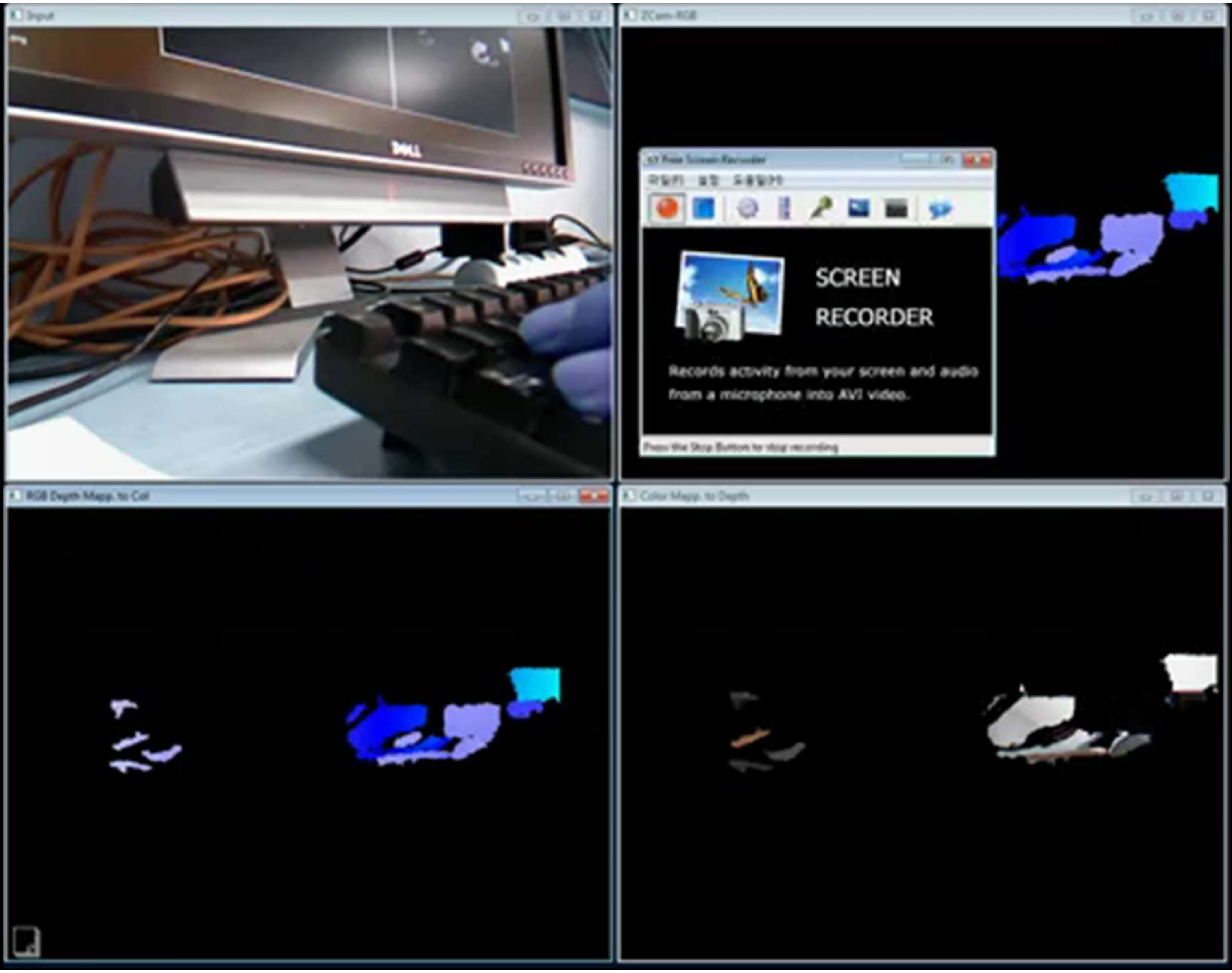


Microsoft Kinect: Principle of operation

Main idea: *Dense depth* via structured light projection *(active stereo)*

1. Project IR dots
2. Capture IR image
3. Identify dots in IR image via *local pattern*
4. Stereo depth estimation





Records activity from your screen and audio from a microphone into AVI video.

Press the Stop Button to stop recording

Microsoft Kinect

- Released Nov 2010

- Provides *dense, real-time RGB + D*:
  - 640 x 480 RGB @ 30 Hz
  - 640 x 480 IR (=> depth) @ 30 Hz

- Cost: ~$100

- **But:** How can we *utilize* all of this data?

General purpose GPU programming

CUDA:

- Programming language for *parallel computing* on NVIDIA graphical processing units (GPUs)

- Released by NVIDIA June 2007

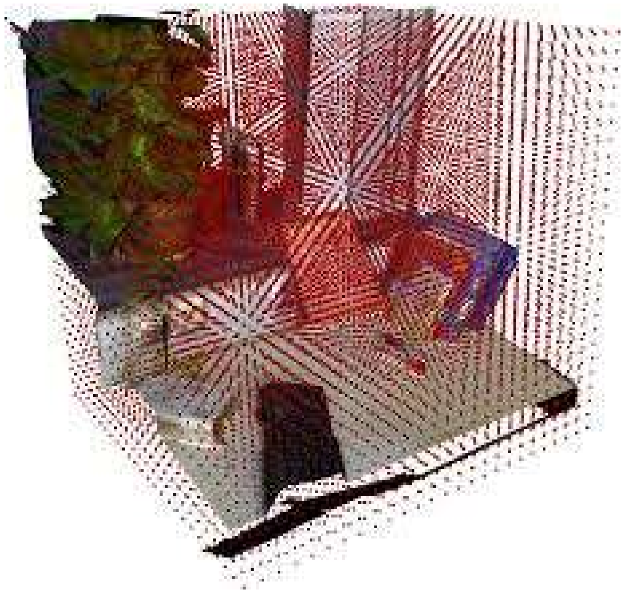
KinectFusion

Main idea: Dense depth data (Kinect) + massive parallelism (GPU) = **WIN!**

Major design points:

- Volumetric (TSDF) environment model, stored on the GPU
  ⇒ *Fast* (*parallel*) data fusion

- Kinect pose estimation via iterative closest point (ICP)



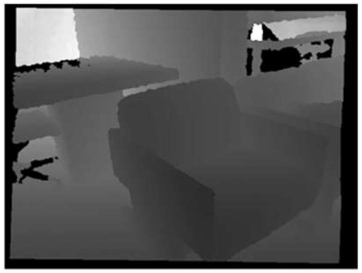
[Image credit: E. Bylow et al.]

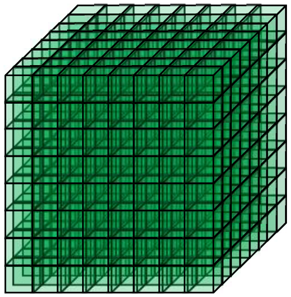
[R.A. Newcombe et al., "KinectFusion: Real-Time Dense Surface Mapping and Tracking", ISMAR, 2011]

KinectFusion: System architecture

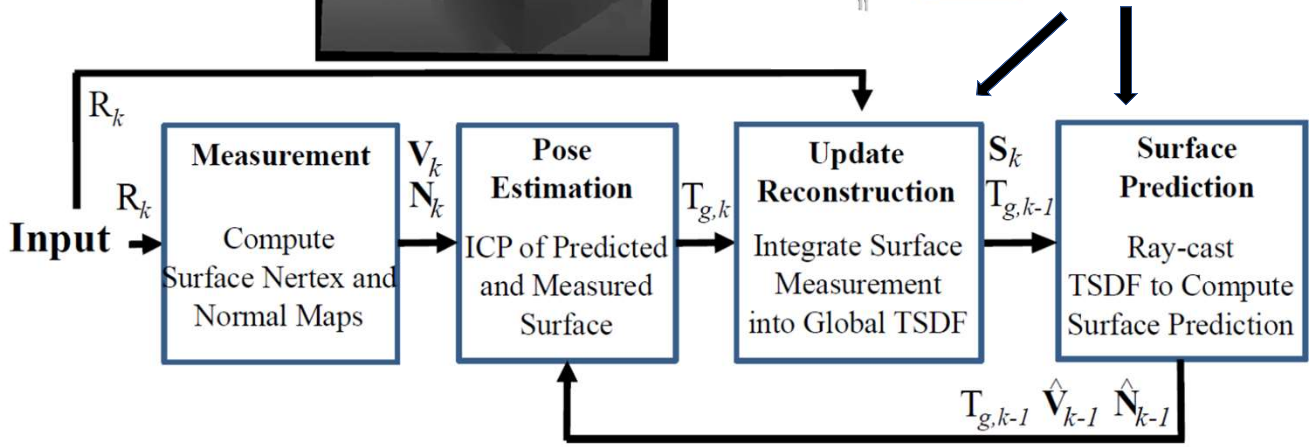
Basic algorithm: *interleave* camera tracking (ICP) and TSDF fusion

Depth map





TSDF



Input $\rightarrow$ R_k

Measurement

Compute Surface Nertex and Normal Maps

V_k N_k

Pose Estimation

ICP of Predicted and Measured Surface

$T_{g,k}$

Update Reconstruction

Integrate Surface Measurement into Global TSDF

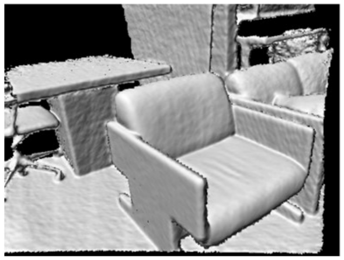
S_k $T_{g,k-1}$

Surface Prediction

Ray-cast TSDF to Compute Surface Prediction

R_k

$T_{g,k-1}$ $\hat{V}_{k-1}$ $\hat{N}_{k-1}$



Surface model

[R.A. Newcombe et al., "KinectFusion: Real-Time Dense Surface Mapping and Tracking", ISMAR, 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

KinectFusion

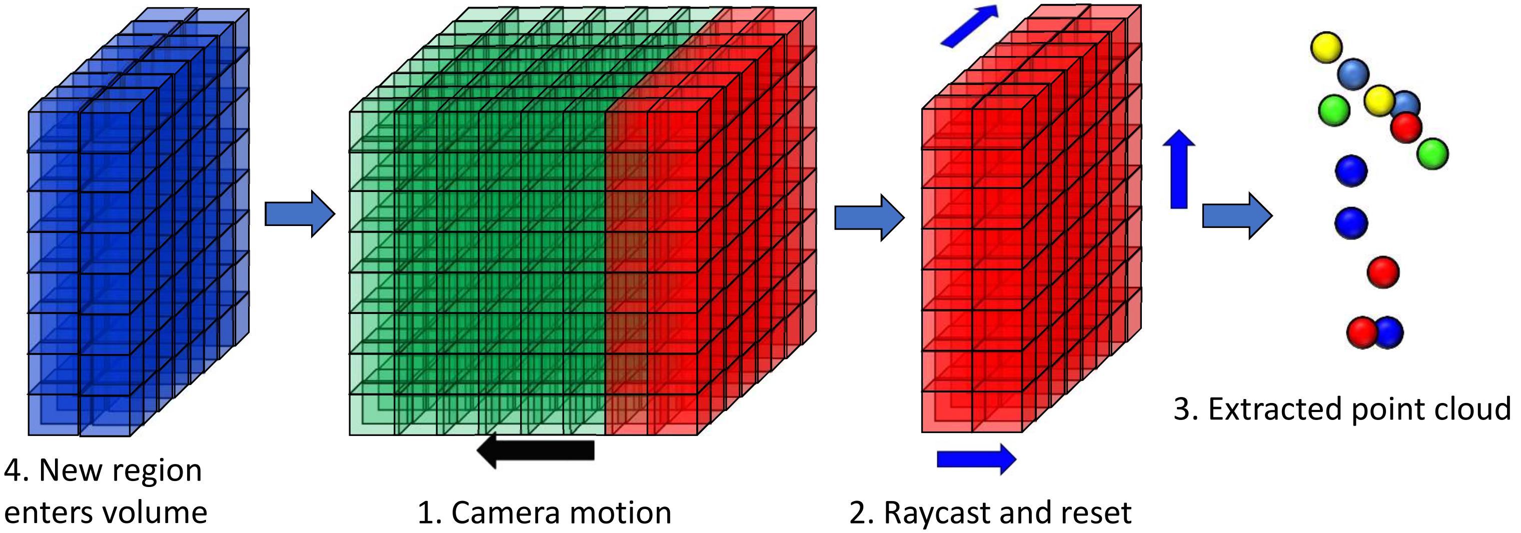
First real-time, dense SLAM system

But: (naïve) TSDF can only represent a *fixed volume*
⇒ Can only map inside a *fixed area*

[R.A. Newcombe et al., "KinectFusion: Real-Time Dense Surface Mapping and Tracking", ISMAR, 2011]

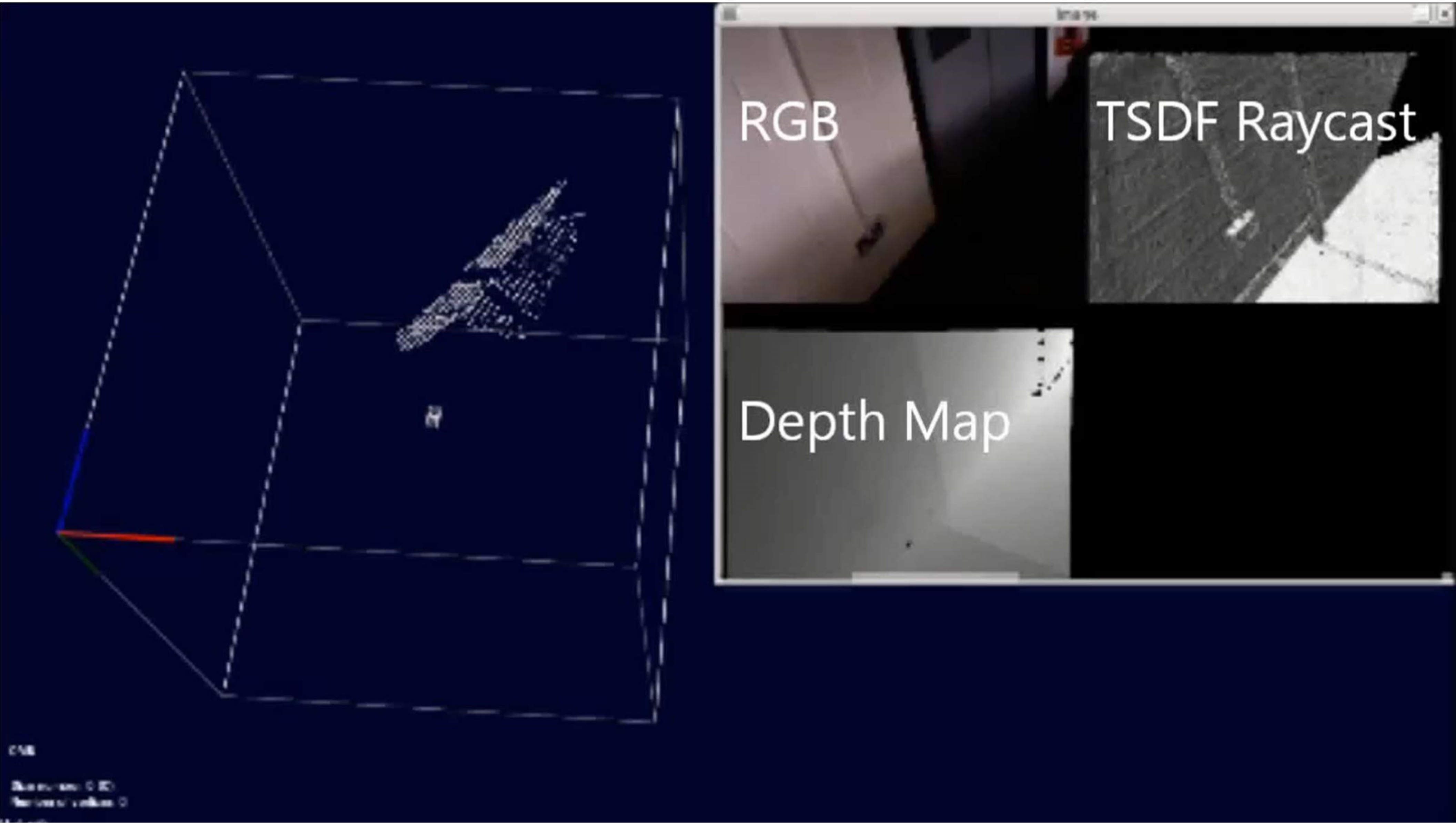
Kintinuous: Spatially-extended KinectFusion

Main idea: Treat the voxel grid used for the TSDF as a *cyclical buffer*



Payoff: Can scale to *arbitrarily large* environments!

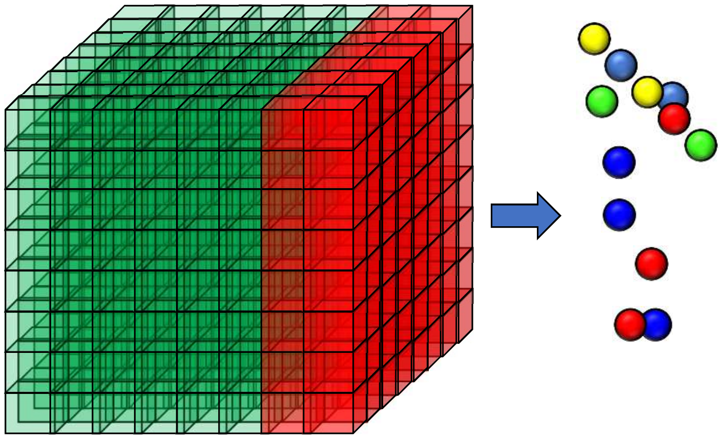
[T. Whelan et al., "Kintinuous: Spatially-Extended KinectFusion", RSS, 2012]


RGB
TSDF Raycast
Depth Map

Kintinuous: Spatially-extended KinectFusion

Main idea: Treat the voxel grid used for the TSDF as a *cyclical buffer*

Payoff: Can scale to *arbitrarily large* environments!



But: Surface geometry is *fixed* once it's extracted the TSDF

⇒ How do we deal with *drift*?

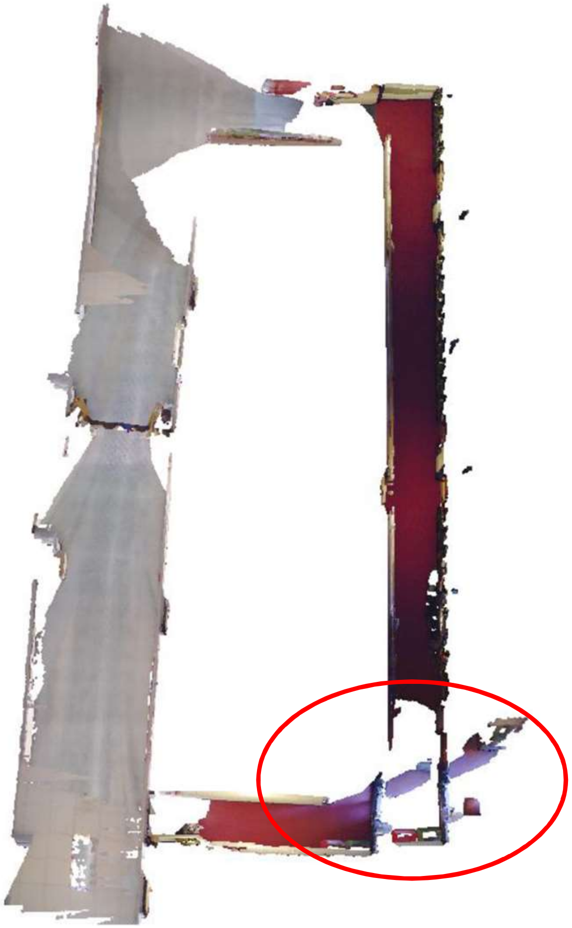
The problem of drift

Problem: Kintinuous' odometric camera-pose estimate drifts over time

⇒ This can lead to inconsistency when closing long loops

What to do?

⇒ Apply ***SLAM***



[T. Whelan et al., "Deformation-Based Loop Closure for Large-Scale Dense RGBD-SLAM", IROS 2013]

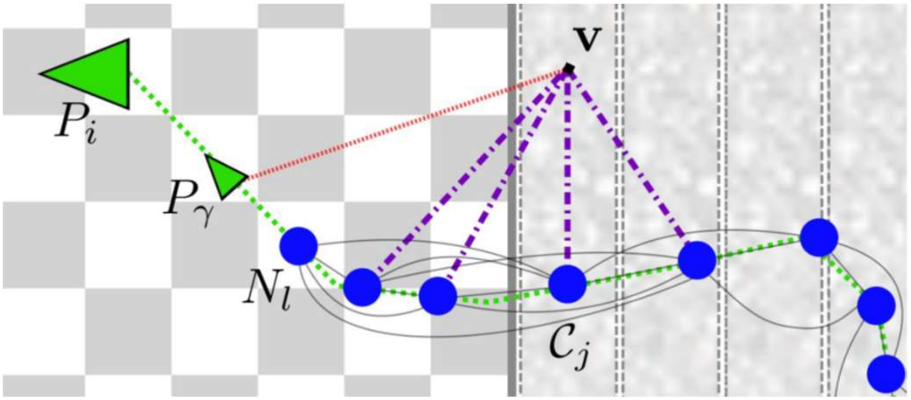
Kintinuous 2.0

Main idea: Employ *pose-graph SLAM* to correct geometry by *warping space*

Major design points:

- Maintain “cloud slice” ⇔ camera pose correspondence
- Apply pose-graph SLAM to joint camera-pose + vertex surface model

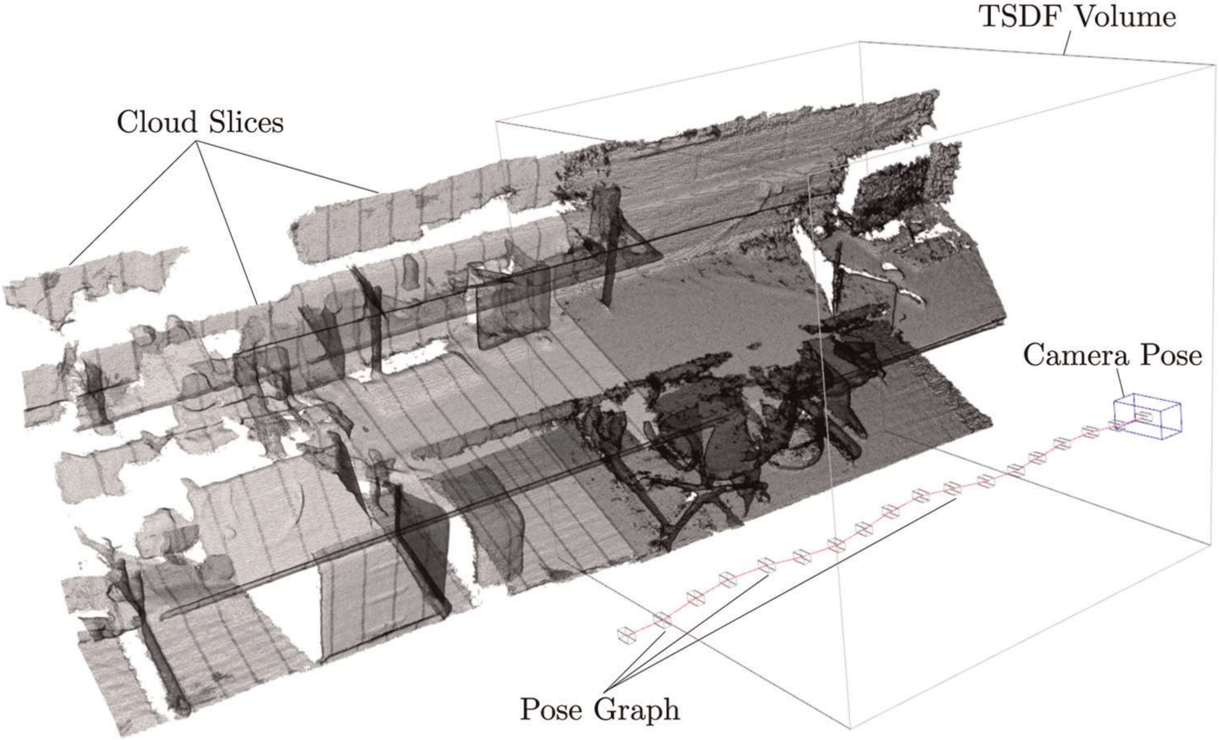
⇒ This has the effect of *warping space*



[T. Whelan et al., “Deformation-Based Loop Closure for Large-Scale Dense RGBD-SLAM”, IROS 2013]

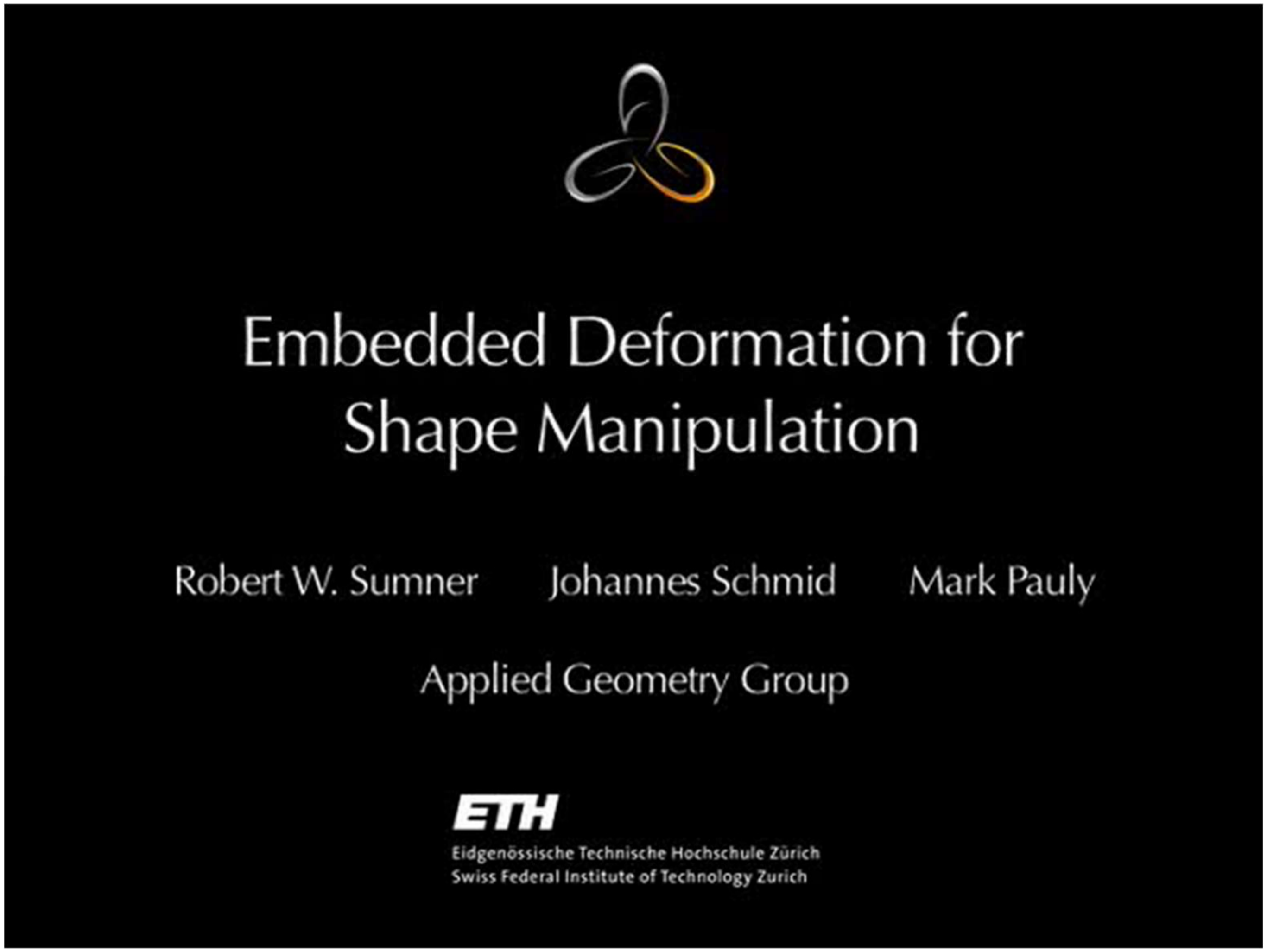
Kintinuous 2.0

Main idea: Employ *pose-graph SLAM* to correct geometry by *warping space*



[T. Whelan et al., “Real-Time Large-Scale Dense RGB-D SLAM with Volumetric Fusion”, IJRR 2014]

Inspiration: Mesh deformation



Embedded Deformation for Shape Manipulation

Robert W. Sumner Johannes Schmid Mark Pauly

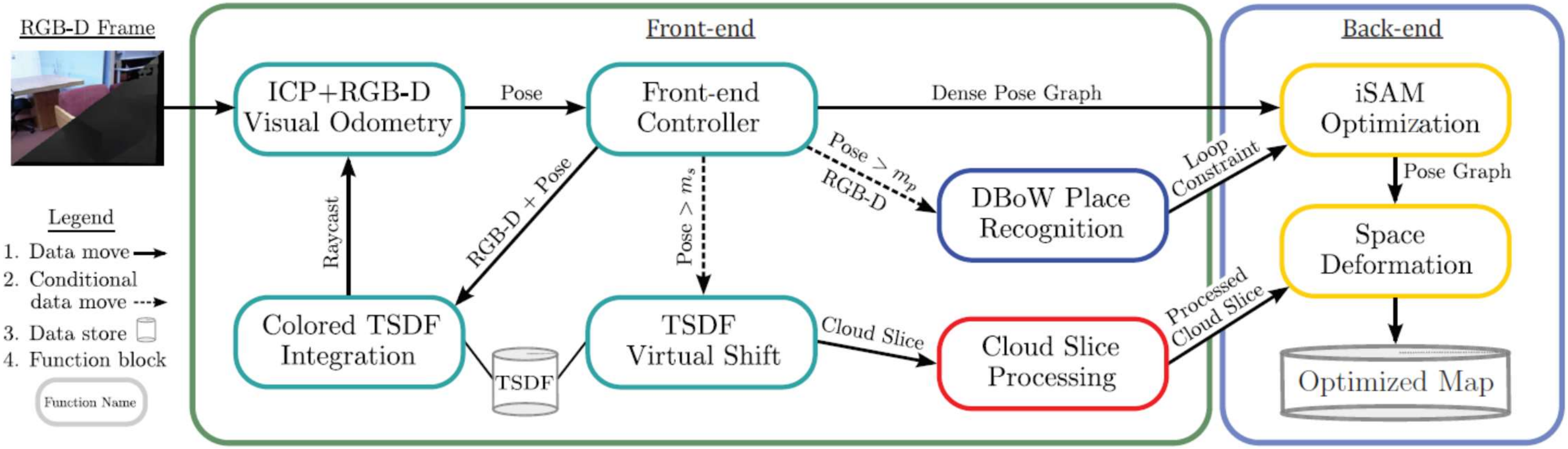
Applied Geometry Group

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Kintinuous 2.0: System architecture

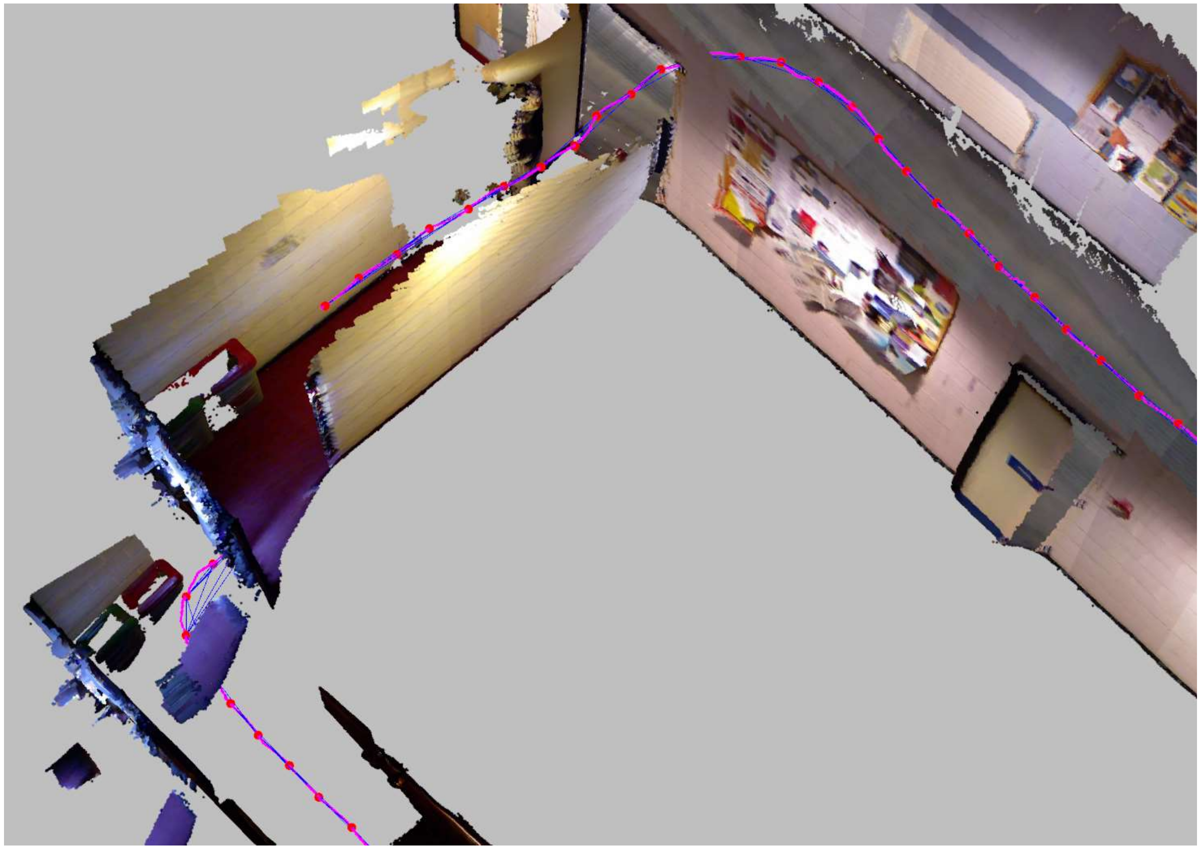
Main idea: Employ *pose-graph SLAM* to correct geometry by *warping space*



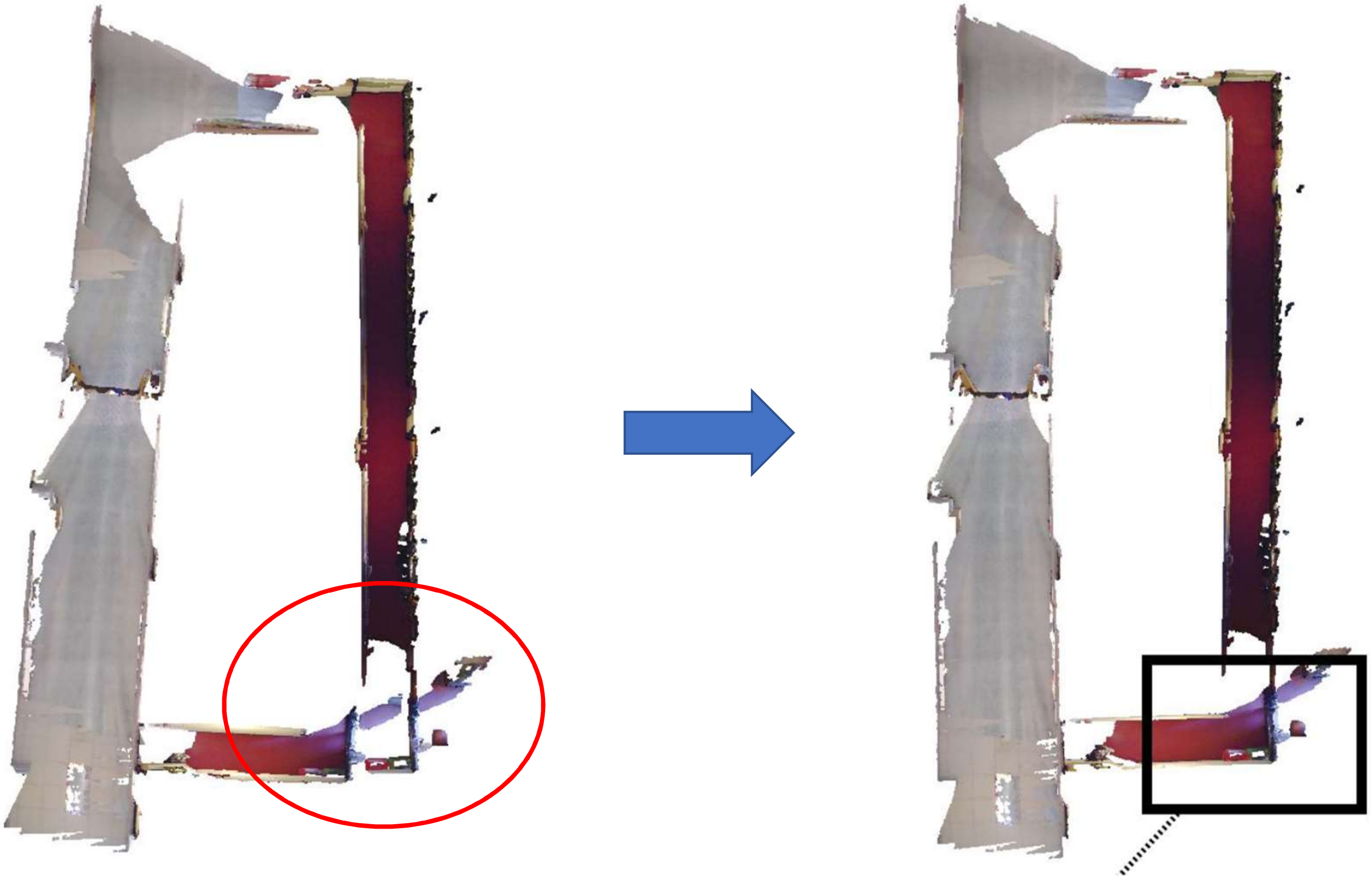
⇒Combines *pose-graph SLAM* + *dense reconstruction*

[T. Whelan et al., "Real-Time Large-Scale Dense RGB-D SLAM with Volumetric Fusion", IJRR 2014]

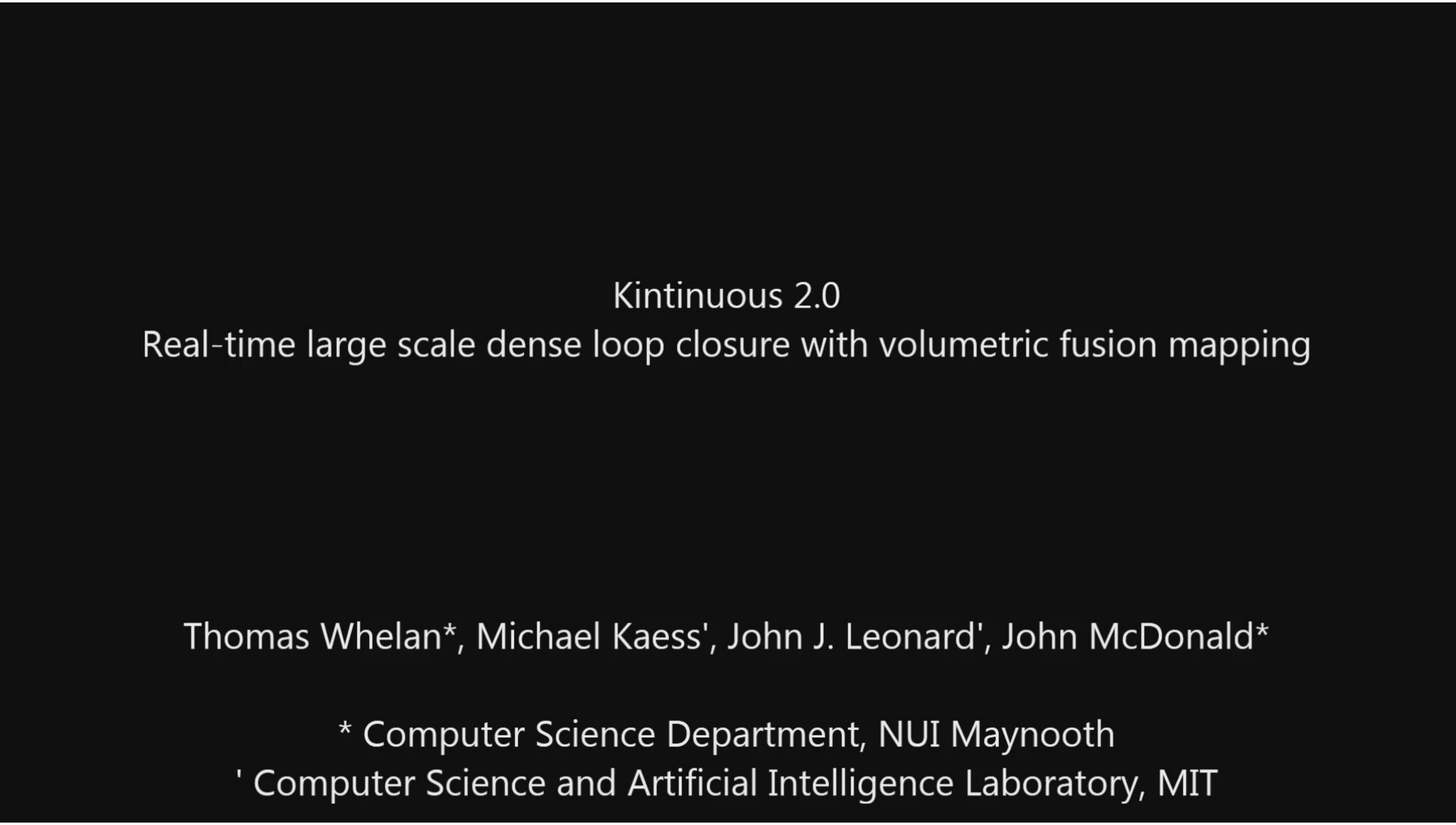
Embedded deformation graph



Correcting drift via mesh deformation



[T. Whelan et al., “Deformation-Based Loop Closure for Large-Scale Dense RGBD-SLAM”, IROS 2013]



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

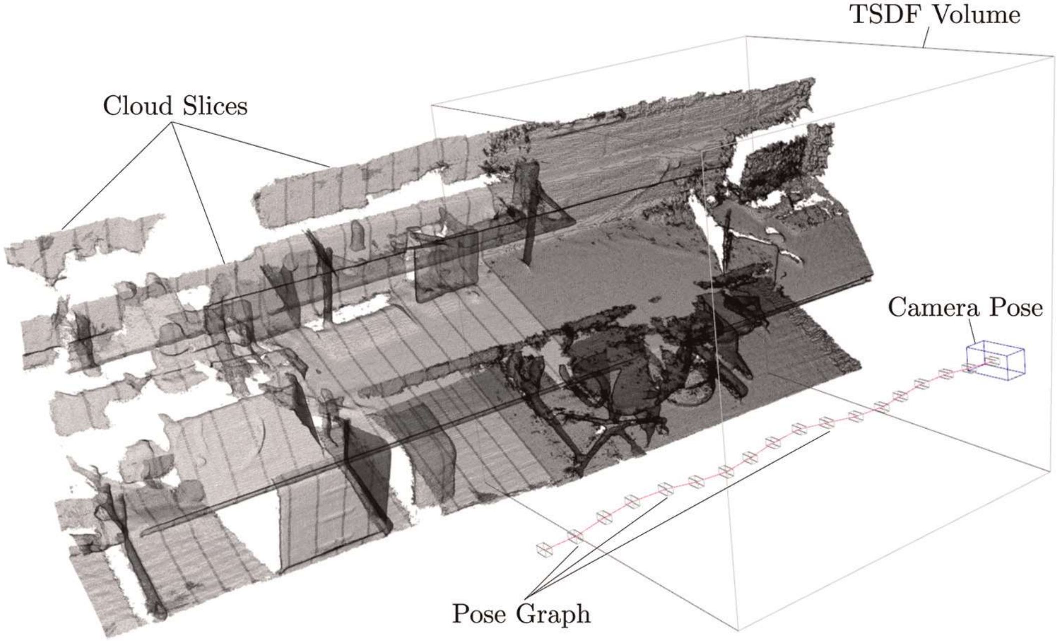
Thomas Whelan*, Michael Kaess', John J. Leonard', John McDonald*

* Computer Science Department, NUI Maynooth
' Computer Science and Artificial Intelligence Laboratory, MIT

Kintinuous 2.0

Main idea: Employ *pose-graph SLAM* to correct geometry by *warping space*

- Combines *pose-graph SLAM* + *dense reconstruction*
- Employs *space deformation* to correct dense surface geometry

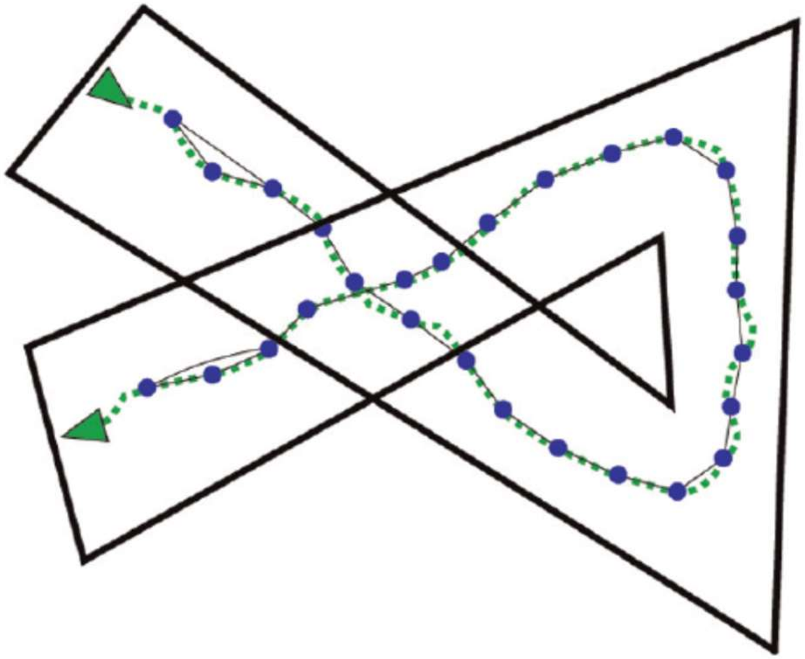


[T. Whelan et al., "Real-Time Large-Scale Dense RGB-D SLAM with Volumetric Fusion", IJRR 2014]

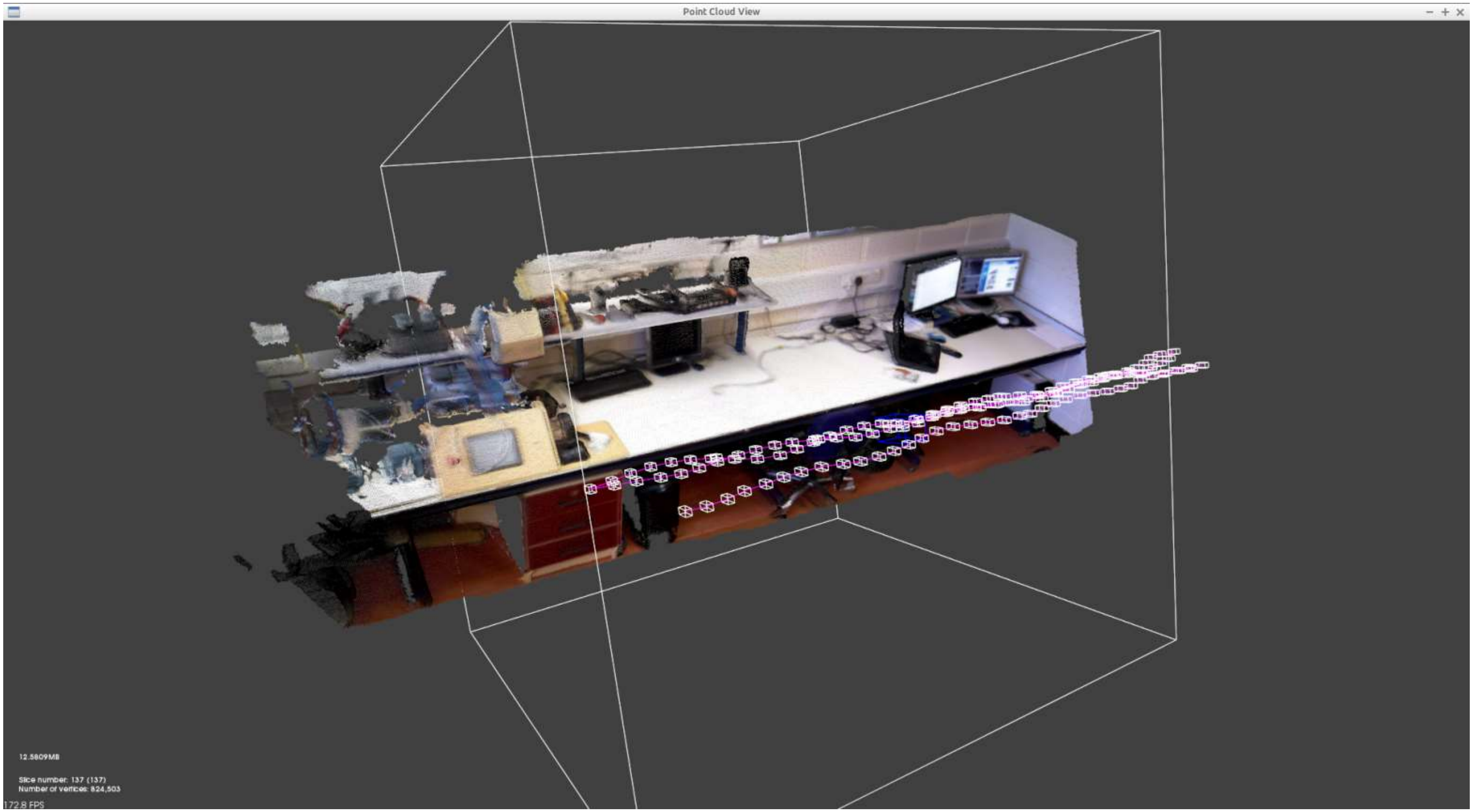
Surface aliasing

Recall: Kintinuous 2.0 attaches *dense structure* to a *pose-graph*

What happens if we revisit the same area many times?



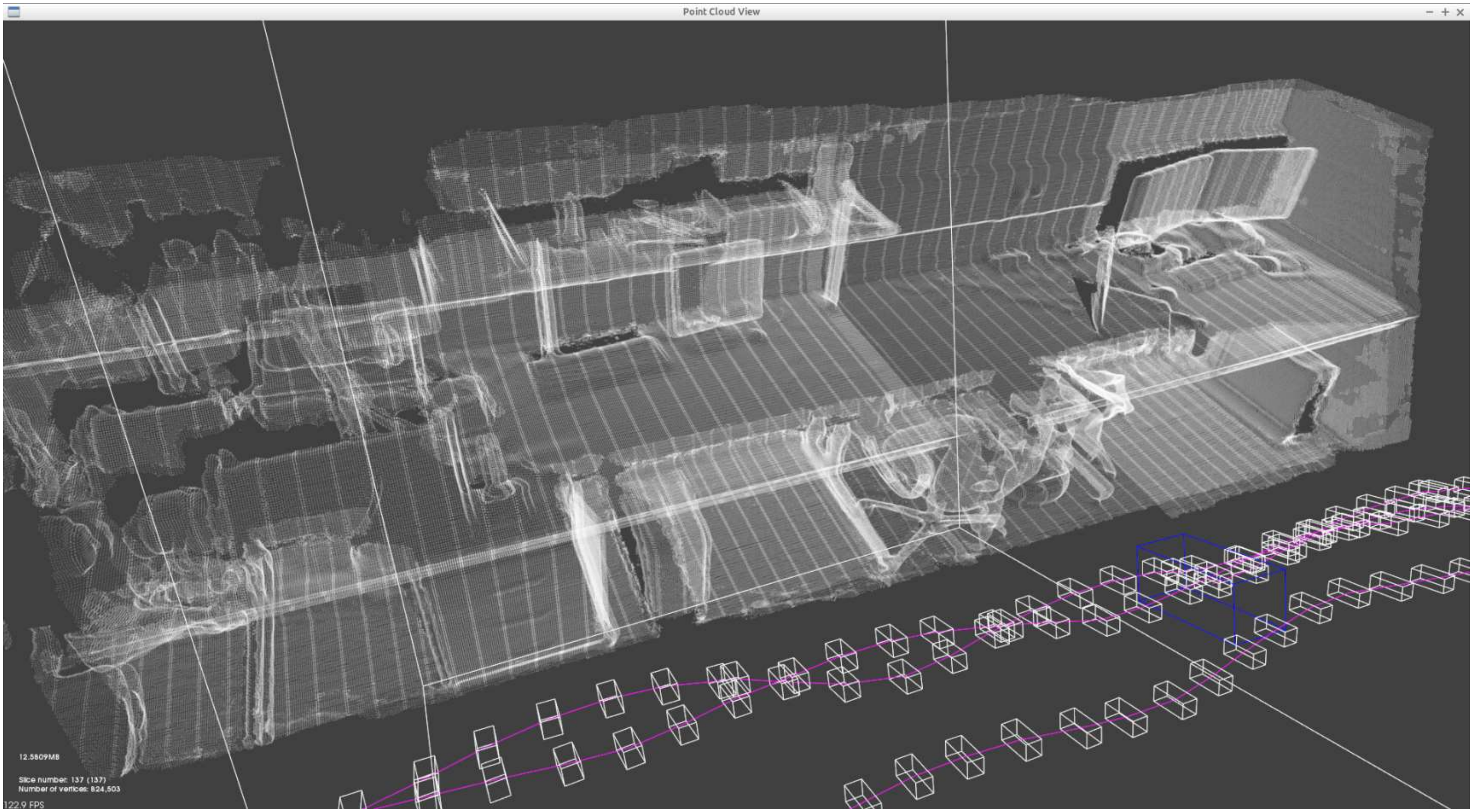
Surface aliasing



```
12.5809MB

Slice number: 137 (137)
Number of vertices: 824,503
172.8 FPS
```

Surface aliasing



```
12.5809MB

Slice number: 137 (137)
Number of vertices: 824,503
122.9 FPS
```

ElasticFusion: Dense SLAM without a pose graph

Main idea: *Eliminate pose-graph*; perform *direct model-model* correspondence

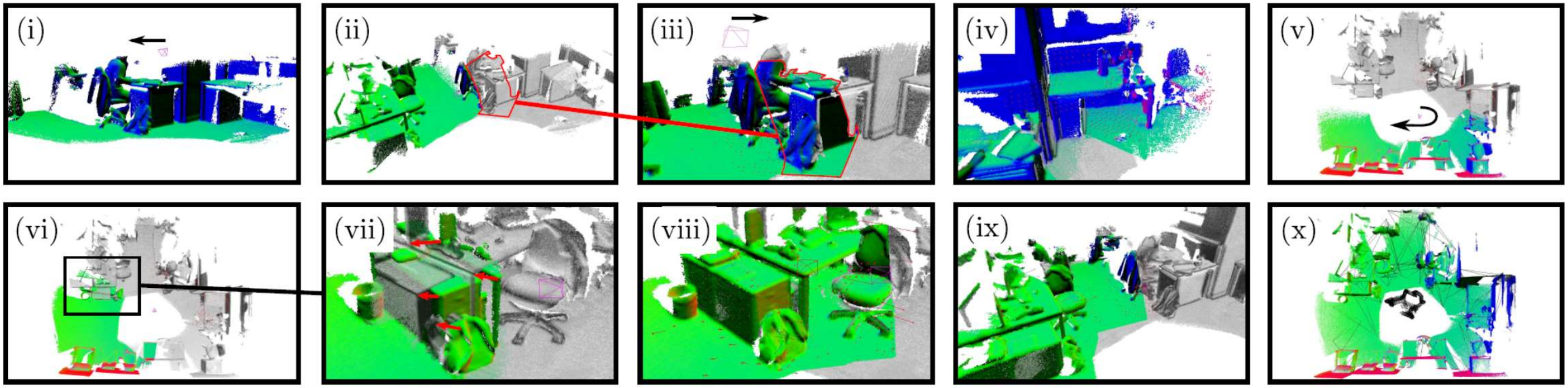
Major design points:

- Employs a *surfel-based* representation
- Maintains a (*local*) *active model* that tracks camera pose
- Loop closure via *active-model registration*

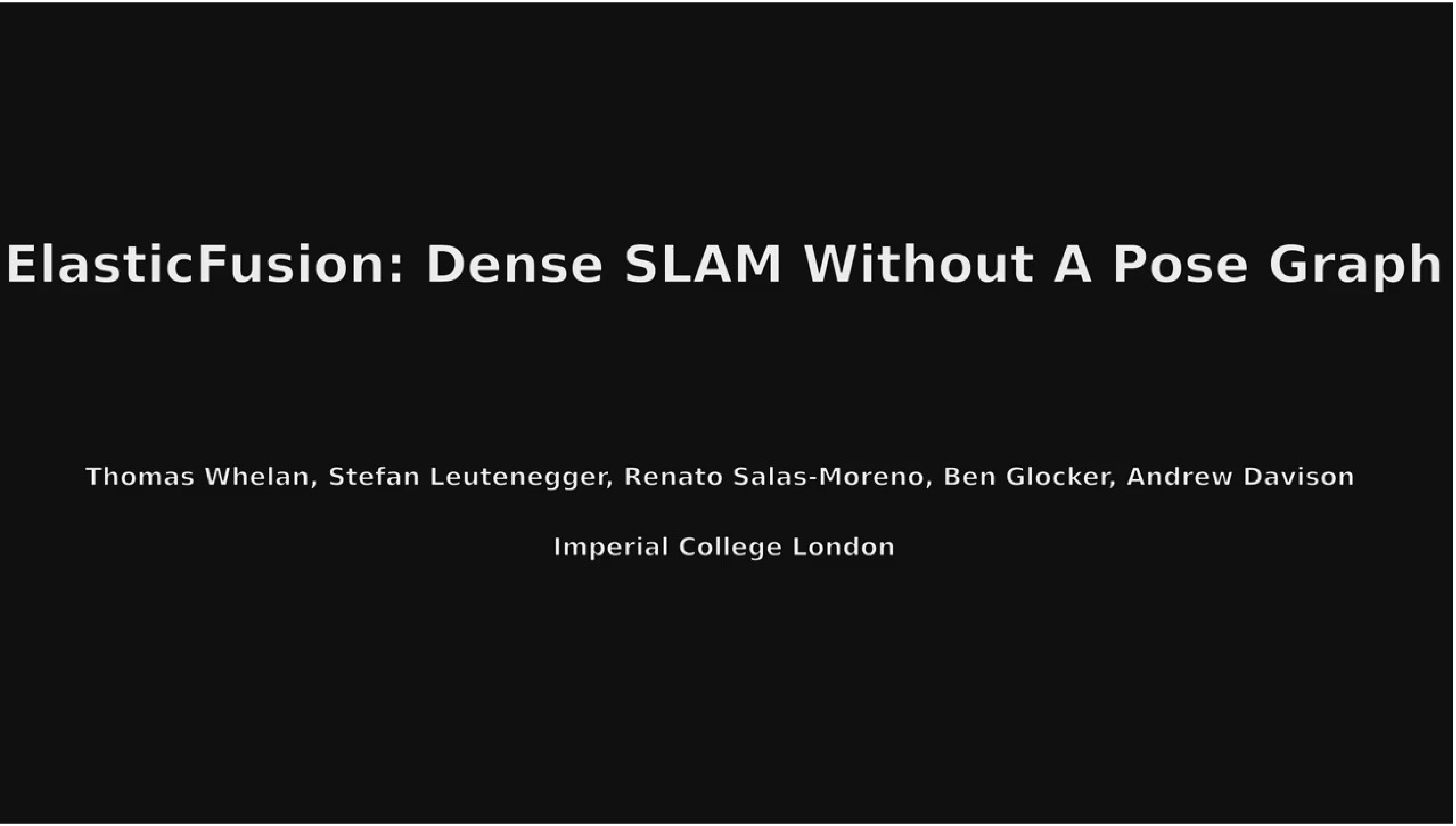
Payoff: Can directly generate *model-model* correspondence

[T. Whelan et al., "ElasticFusion: Dense SLAM without a Pose-Graph", RSS, 2012]

ElasticFusion: Correspondence generation



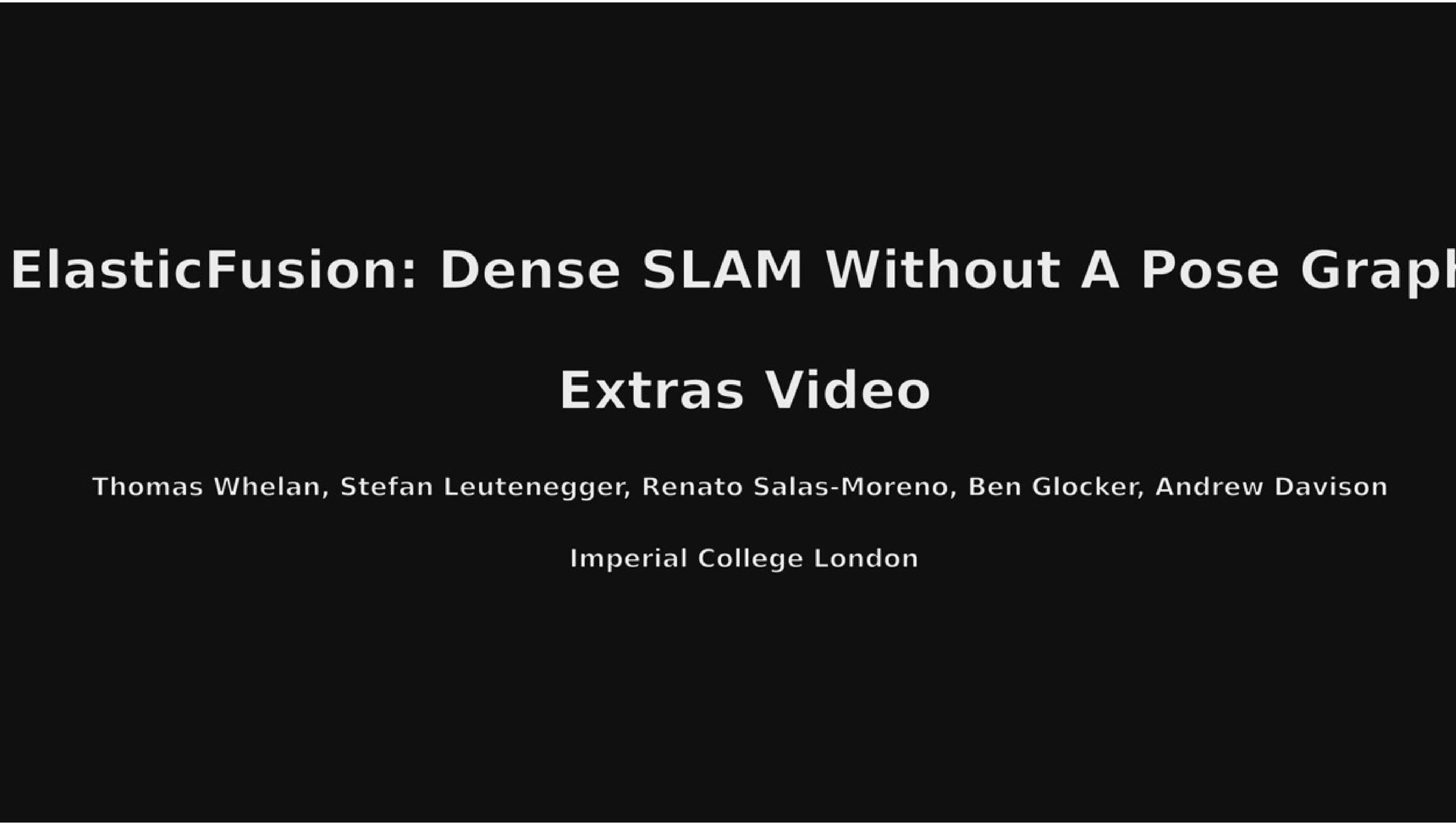
[T. Whelan et al., "ElasticFusion: Dense SLAM without a Pose-Graph", RSS, 2012]



ElasticFusion: Dense SLAM Without A Pose Graph

Thomas Whelan, Stefan Leutenegger, Renato Salas-Moreno, Ben Glocker, Andrew Davison

Imperial College London



ElasticFusion: Dense SLAM Without A Pose Graph

Extras Video

Thomas Whelan, Stefan Leutenegger, Renato Salas-Moreno, Ben Glocker, Andrew Davison

Imperial College London

The Takeaway

- The combination of *inexpensive*, *real-time*, *dense RGB-D* + *inexpensive*, *massively parallel GPGPU computation* has enabled a **revolution** in dense real-time reconstruction.

- **LOTS** of high-quality, open-source systems available for *commodity hardware*

- **Dive in ☺!**