RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments

Peter Henry[1], Michael Krainin[1], Evan Herbst[1],
Hao Du[3], Marvin Cheng[1], Xiaofeng Ren[2], and Dieter Fox[1,2]

[1]University of Washington
Computer Science & Engineering

[2]ISTC-Pervasive Computing

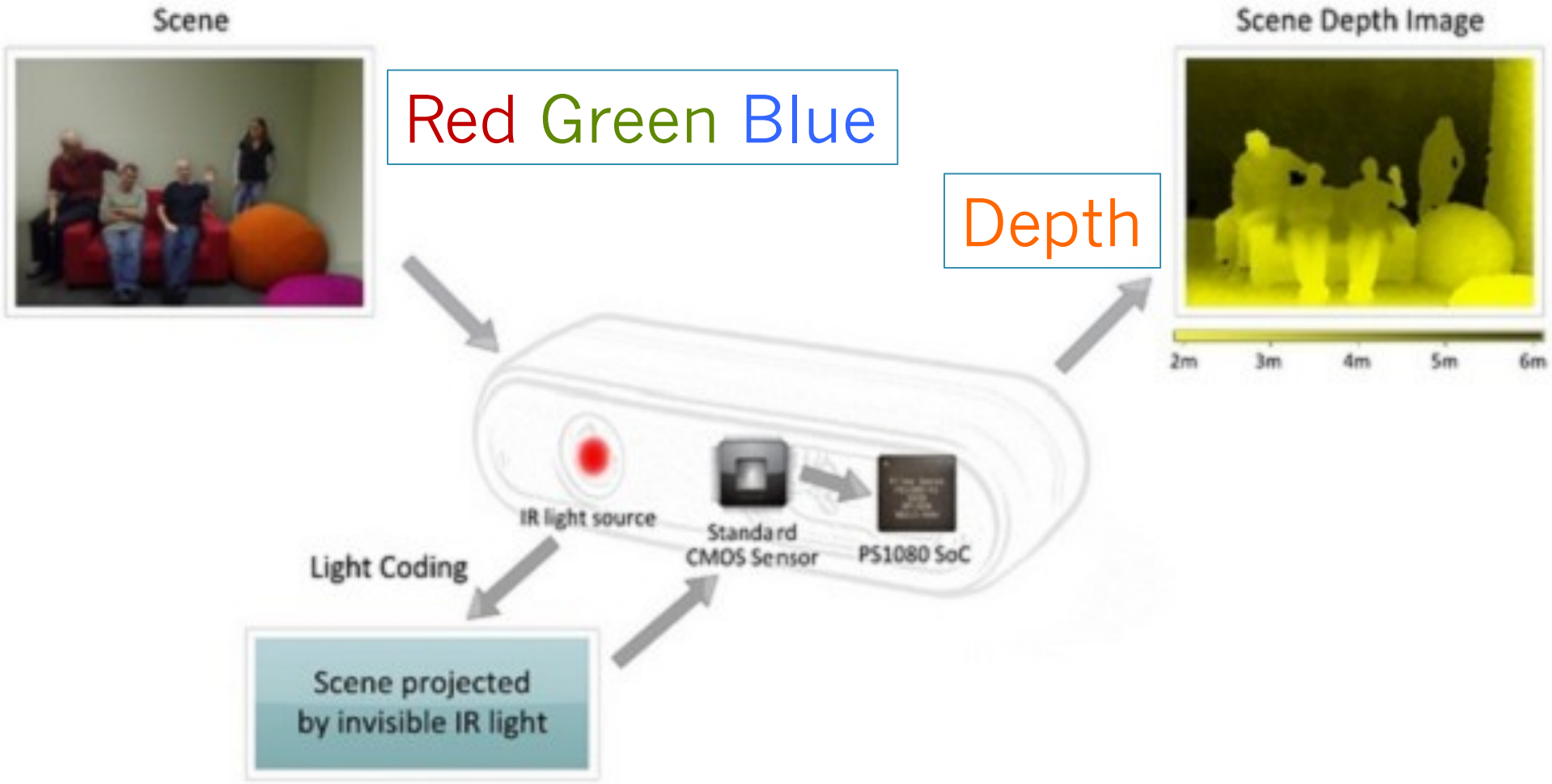
[3]Google

The Kinect

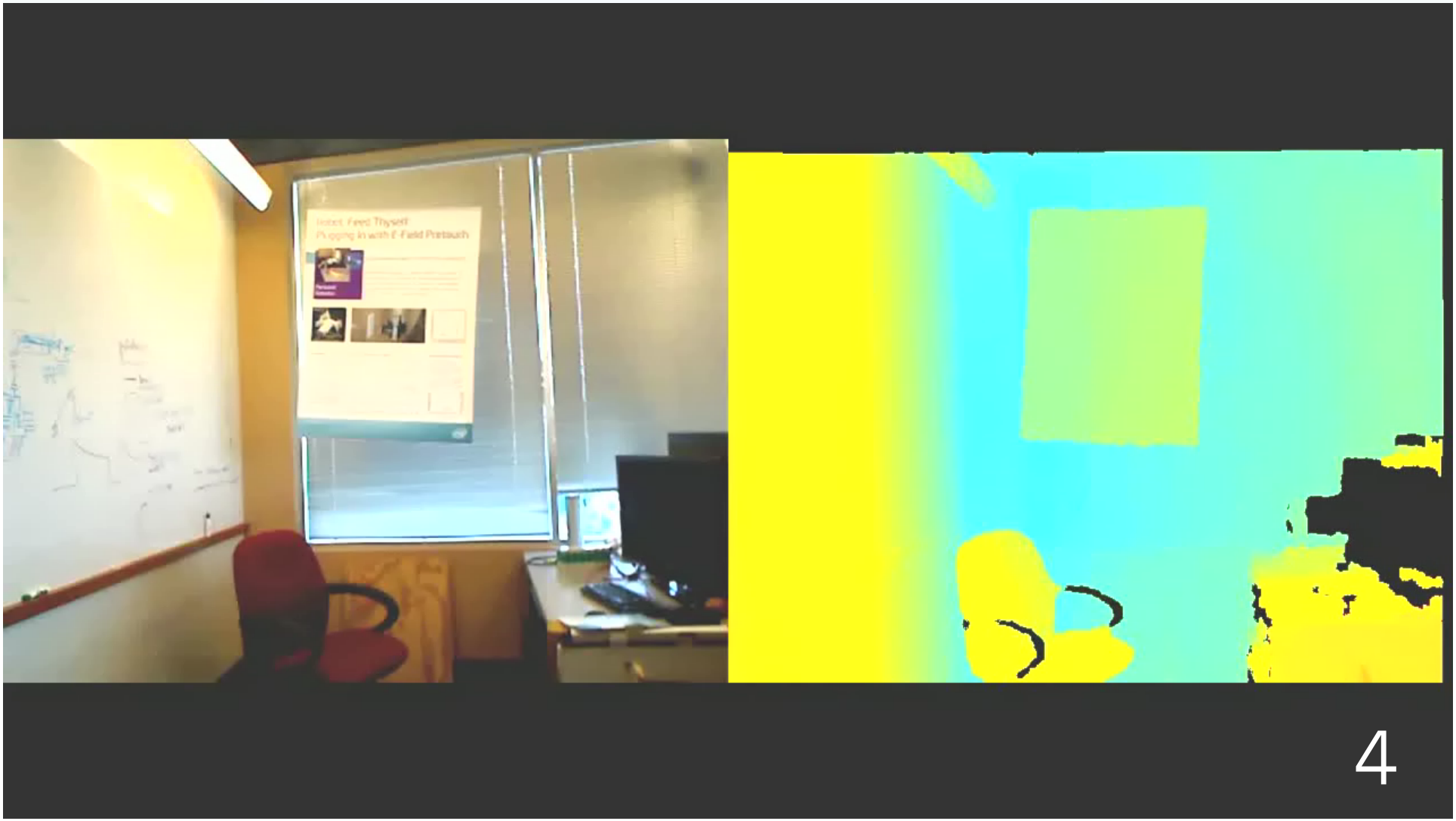
KINECT™
for XBOX 360.

[Slide: PrimeSense]

PrimeSense Technology



RGB-D Data

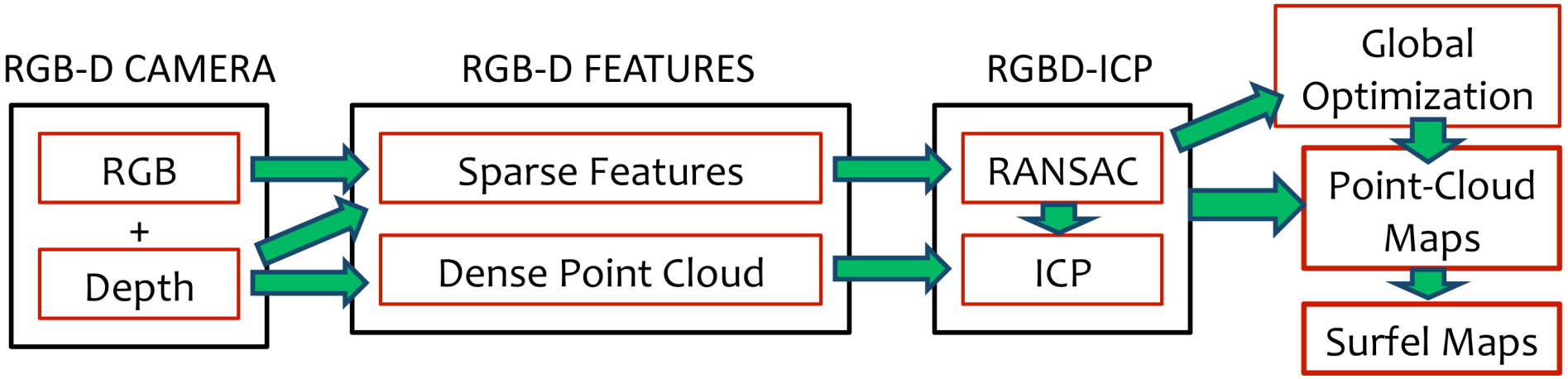


The Goal

Align the “frames” from a Kinect to create a single 3D map (or model) of the environment

Like this…

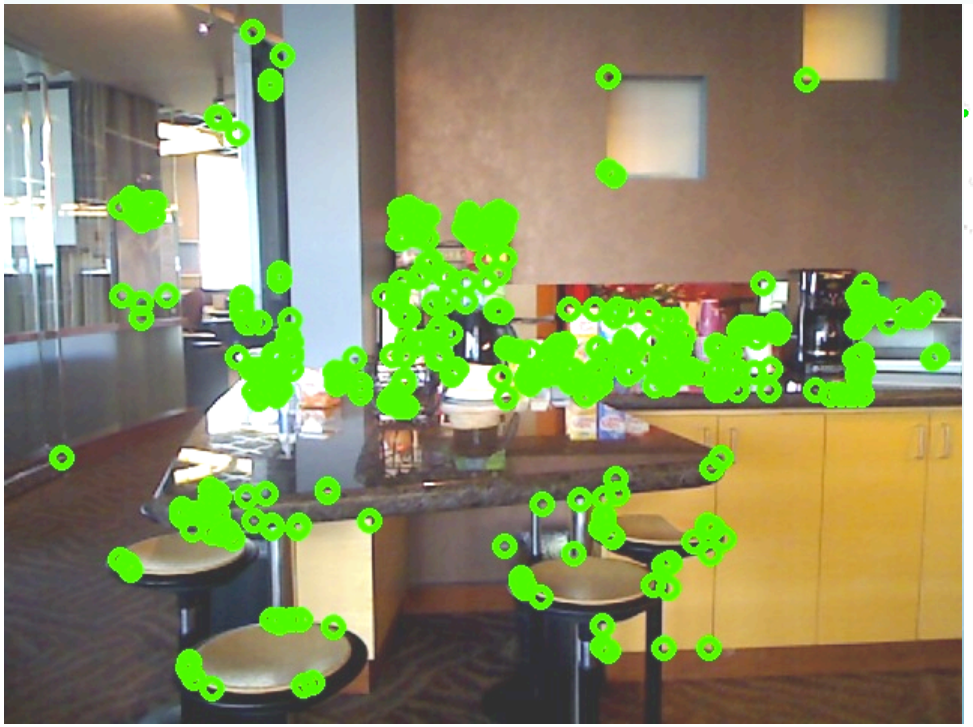
System Overview

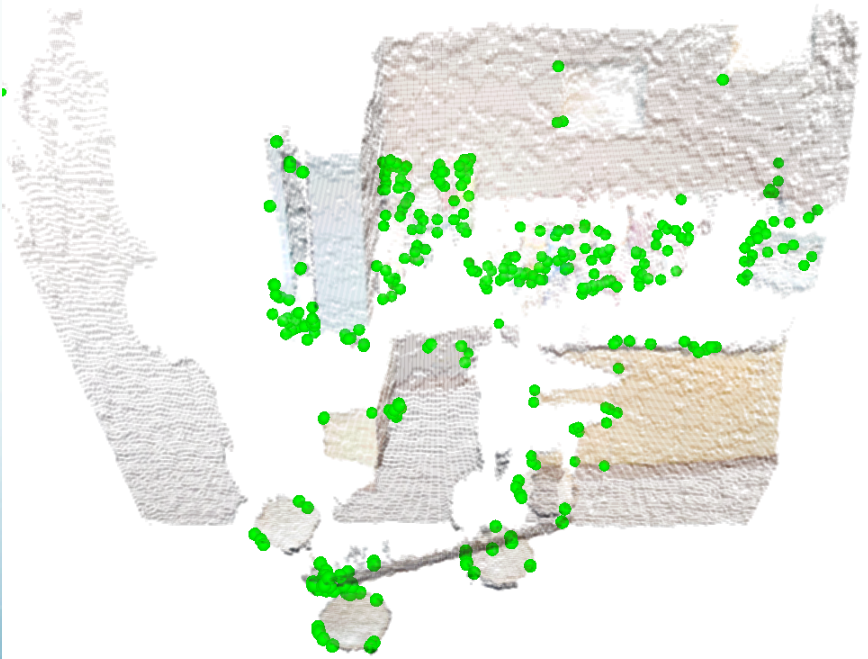


1. Frame-to-frame alignment
2. Global Optimization (Loop Closure)
3. Map representation

RANSAC
(Random Sample Consensus)

- Visual features (from image) in 3D (from depth)
- Figure out how the camera moved by matching these feature



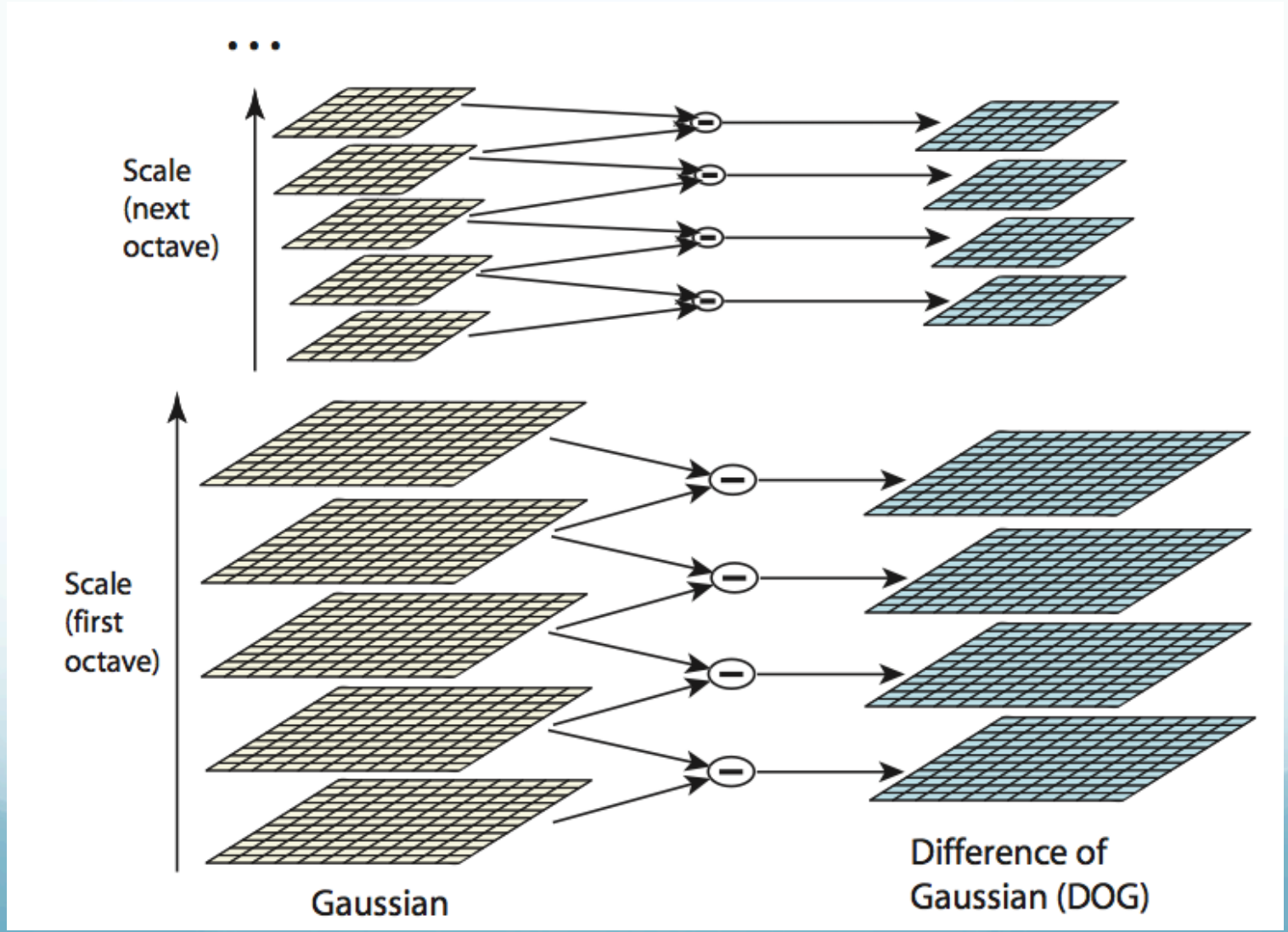


Visual Features

- Detector
  - Repeatable
  - Stable
  - Invariances:
    - Illumination
    - Rotation
    - Scale?
- Descriptor
  - Discriminative
  - Invariant

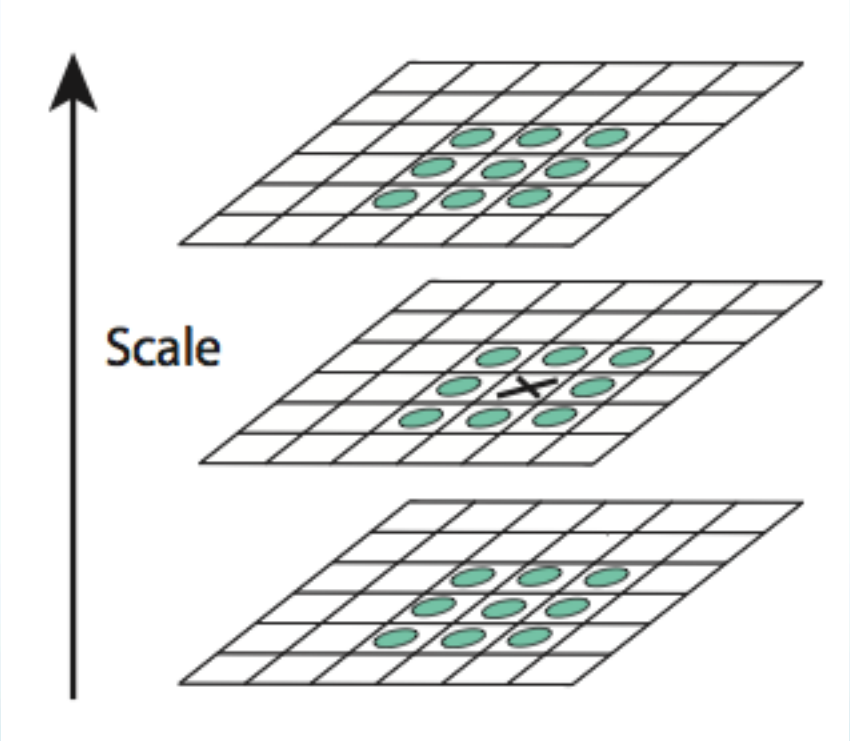
[Slide: David Lowe]

SIFT Detector
(Scale Invariant Feature Transform)

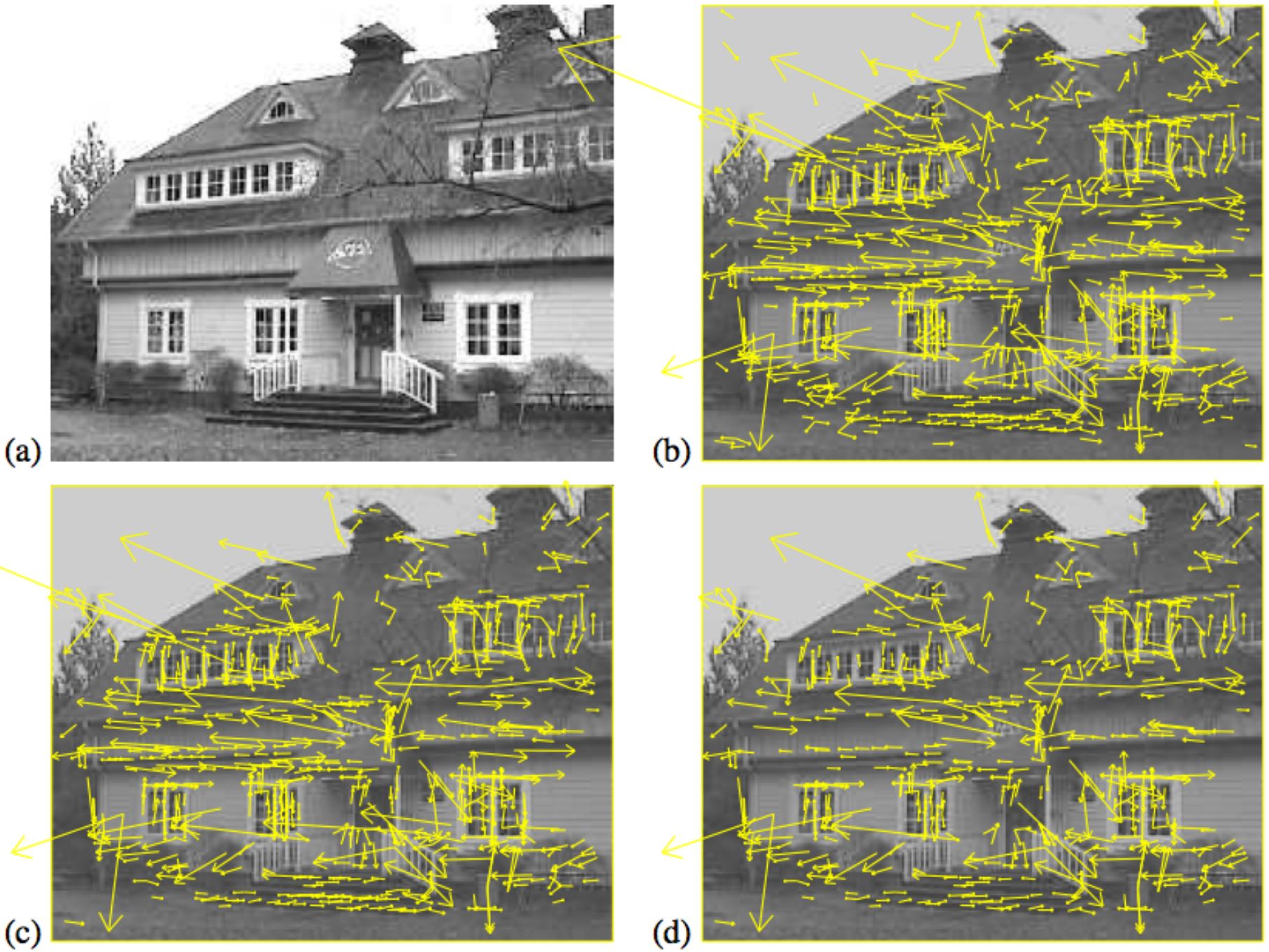


[Slide: David Lowe]

Local Maxima/Minima in Scale Space



[Slide: David Lowe]



[Slide: Edward Rosten]

FAST Detector

(Features from Accelerated Segment Test)

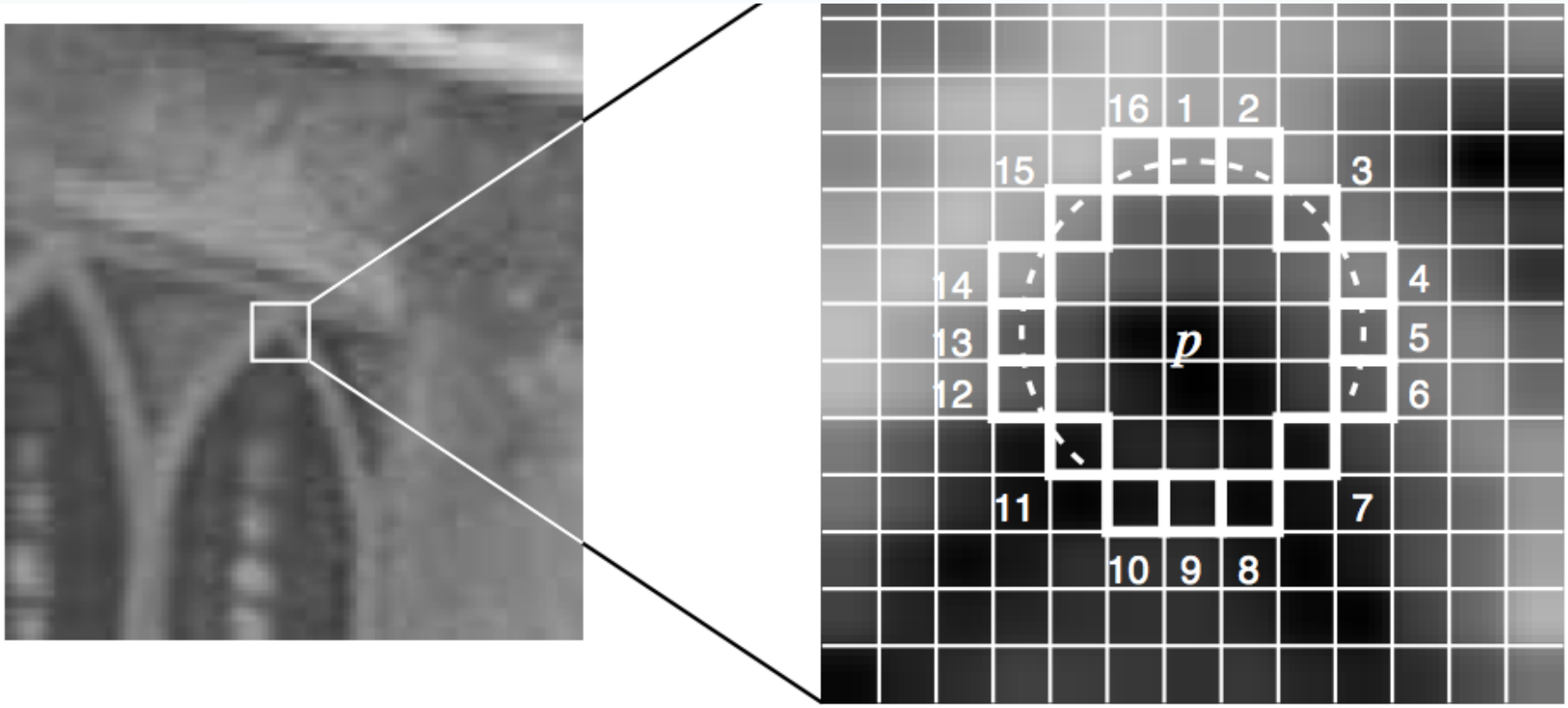


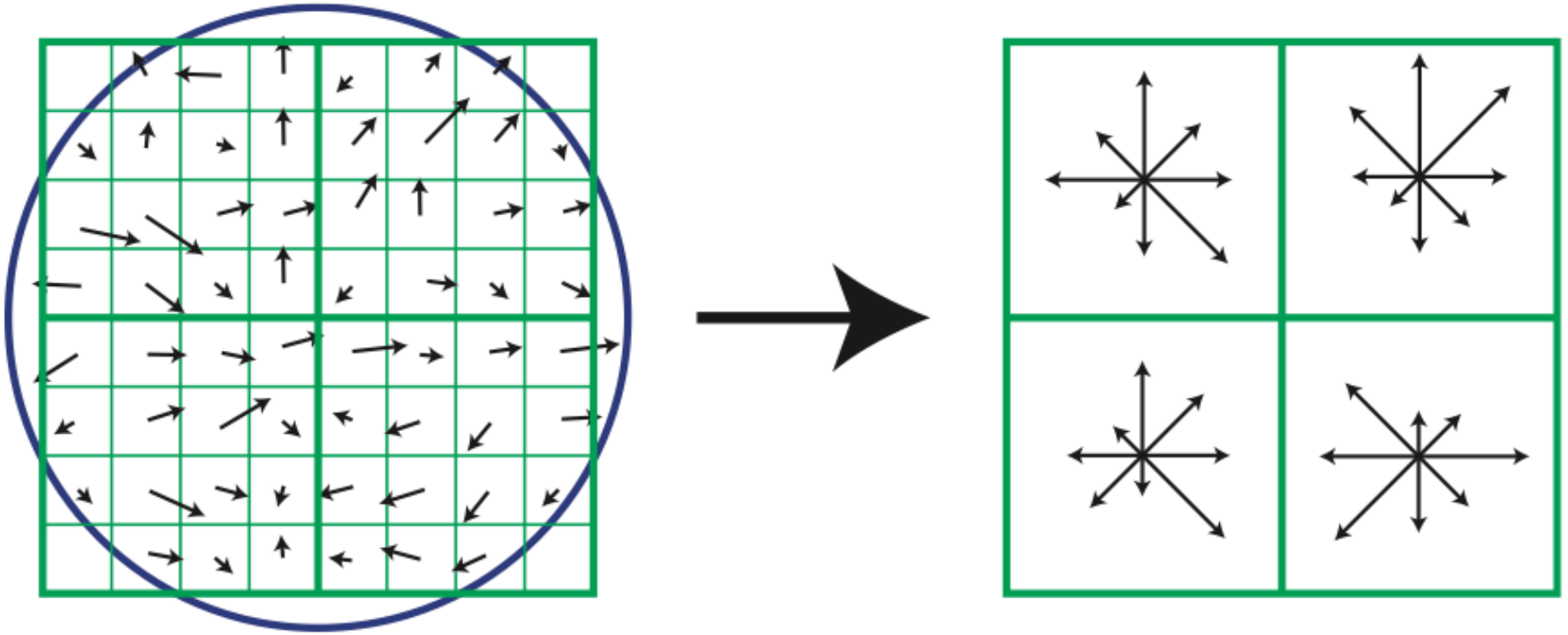
Figure 1. 12 point segment test corner detection in an image patch. The highlighted squares are the pixels used in the corner detection. The pixel at p is the centre of a candidate corner. The arc is indicated by the dashed line passes through 12 contiguous

Detector Properties

- SIFT is invariant to illumination, translation, scale, rotation
- FAST is *not* scale invariant, but is fast (~50x faster)

[Slide: David Lowe]

SIFT Descriptor

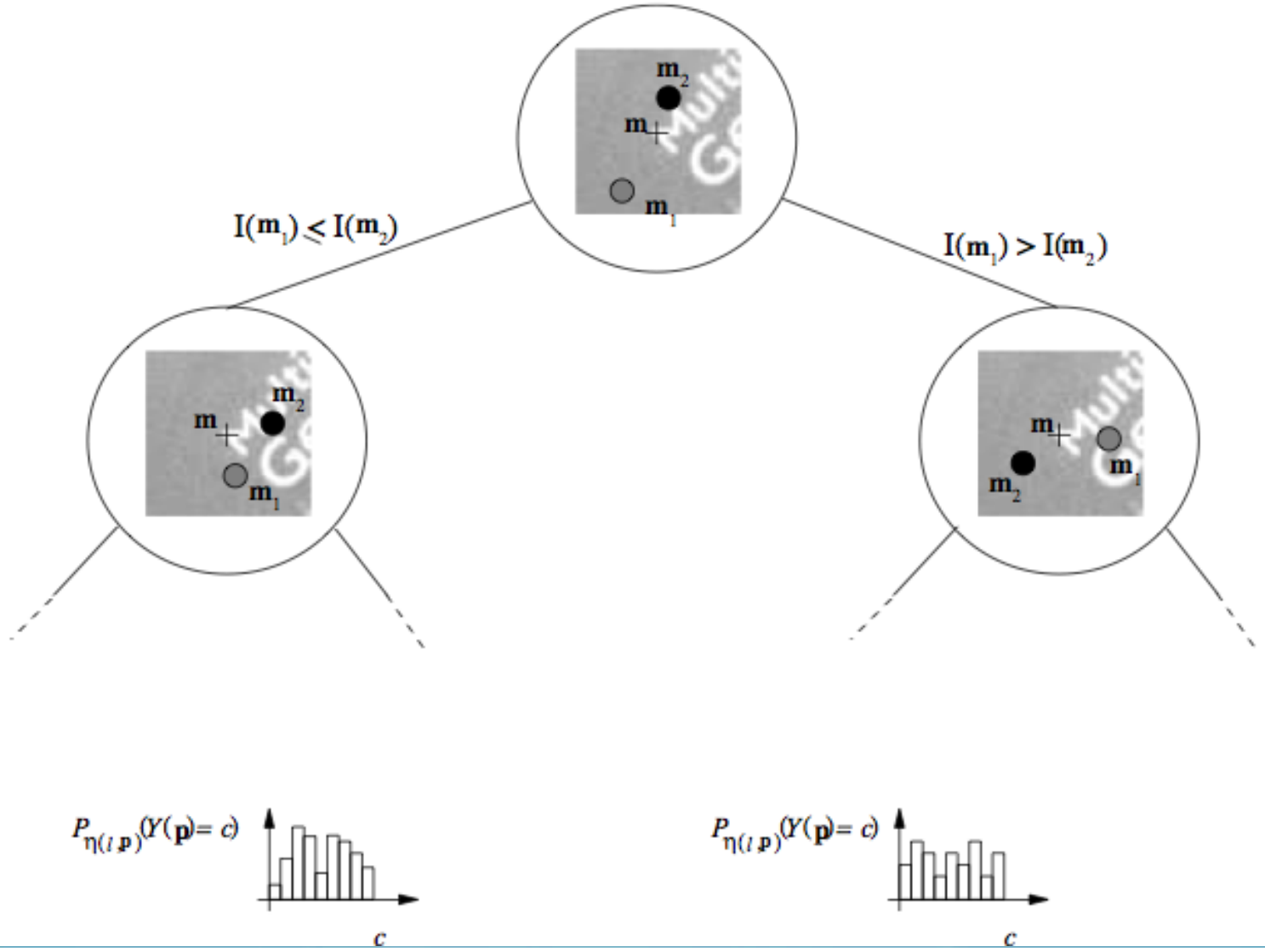


Calonder Randomized Tree Descriptor

- Use training set of keypoints (“base set”)
- Apply distortions in scale, rotation, perspective (“view set”)
- Train randomized decision trees to classify the incoming keypoint
- Descriptor is the distribution across the base set

[Slide: Michael Calonder]

Keypoint Classification



[Slide: Michael Calonder]

Calonder Example

Fig. 3. The leftmost image of each row represents a patch from a test image. The remaining images in the same row represent the patches surrounding the 10 keypoints the test patch looks most similar to according to our Randomized Tree classifier, in decreasing similarity order.

[Slide: Michael Calonder]

Detector/Descriptor Speed

Task	Time
Gaussian pyramid	1434 ms
DOG pyramid	277.4 ms
Feature scales	0.2362 ms
Feature orientations	91.34 ms
Assemble final descriptor	339.2 ms
Total time	2142 ms

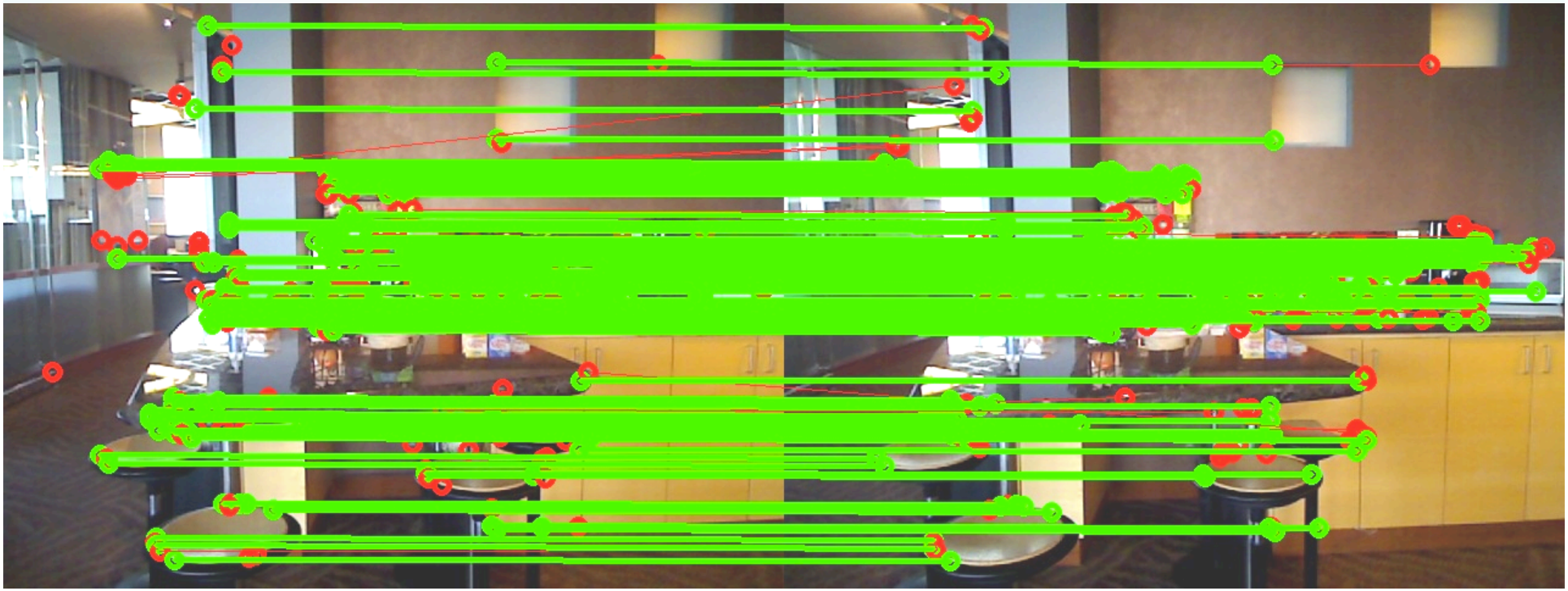
(a) SIFT

Task	Time
Keypoint detection (est'd)	5 ms
Compute base distribution	33.21 ms
Thresholding	1.217 ms
Total time	39.43 ms

(b) GTs

What is RANSAC?

- For each feature point, find the most similar descriptor in the other frame
- Find largest set of consistent matches
- Move the new frame to align these matches



RANSAC Details

- Feature Detector / Descriptor Options
  - SIFT (SiftGPU)
  - SURF (Sped-Up Robust Features)
  - **FAST Detector / Calonder Descriptor**
  - (All available in OpenCV)
- Matching:
  - L2 descriptor distance
  - Frame-to-frame: window matching
  - Loop closure: ratio-test matching

RANSAC Error Function

- Euclidean 3D Error:

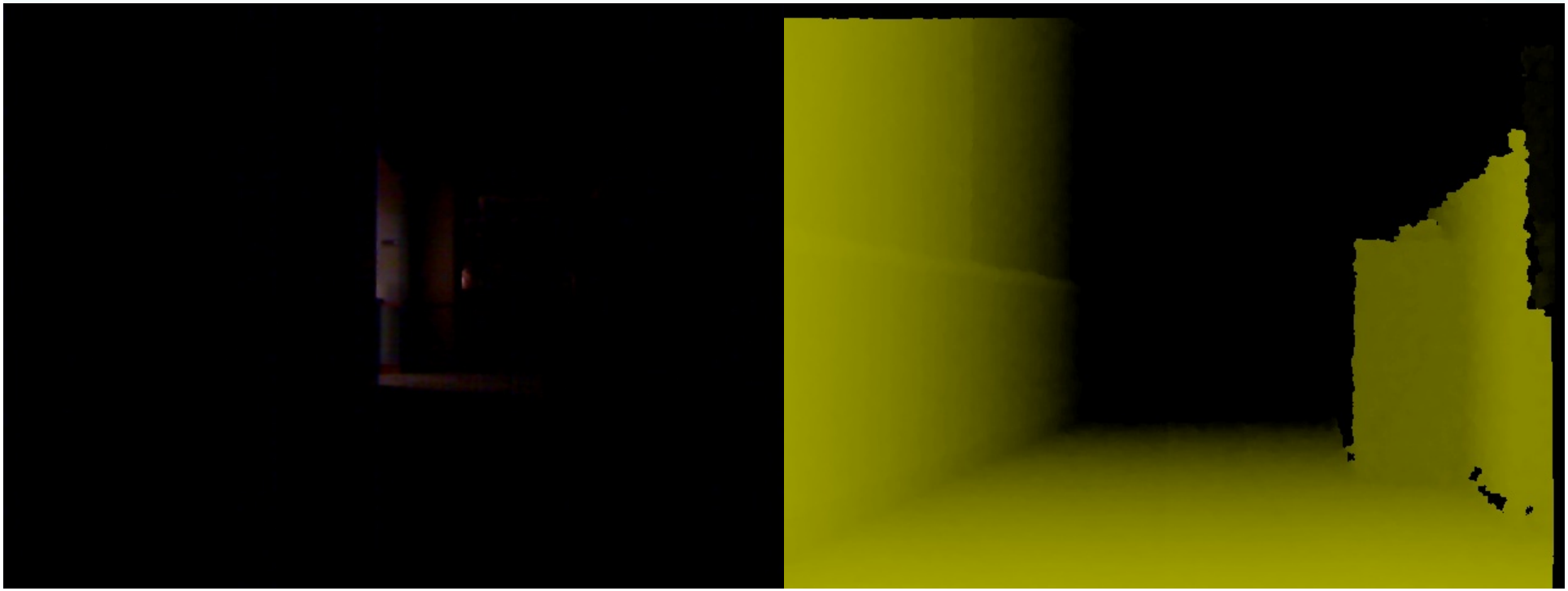
$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \left(\frac{1}{|A_f|} \sum_{i \in A_f} w_i \left| \mathbf{T}(f_s^i) - f_t^i \right|^2 \right)$$

- Reprojection Error (better):

$$\mathbf{T}^* = \underset{\mathbf{T}}{\operatorname{argmin}} \left(\frac{1}{|A_f|} \sum_{i \in A_f} \left| Proj(\mathbf{T}(f_s^i)) - Proj(f_t^i) \right|^2 \right)$$

RANSAC Failure Cases

- Low light
- Lack of visual “texture” or features
- Kinect still provides depth or “shape” information



ICP
(Iterative Closest Point)

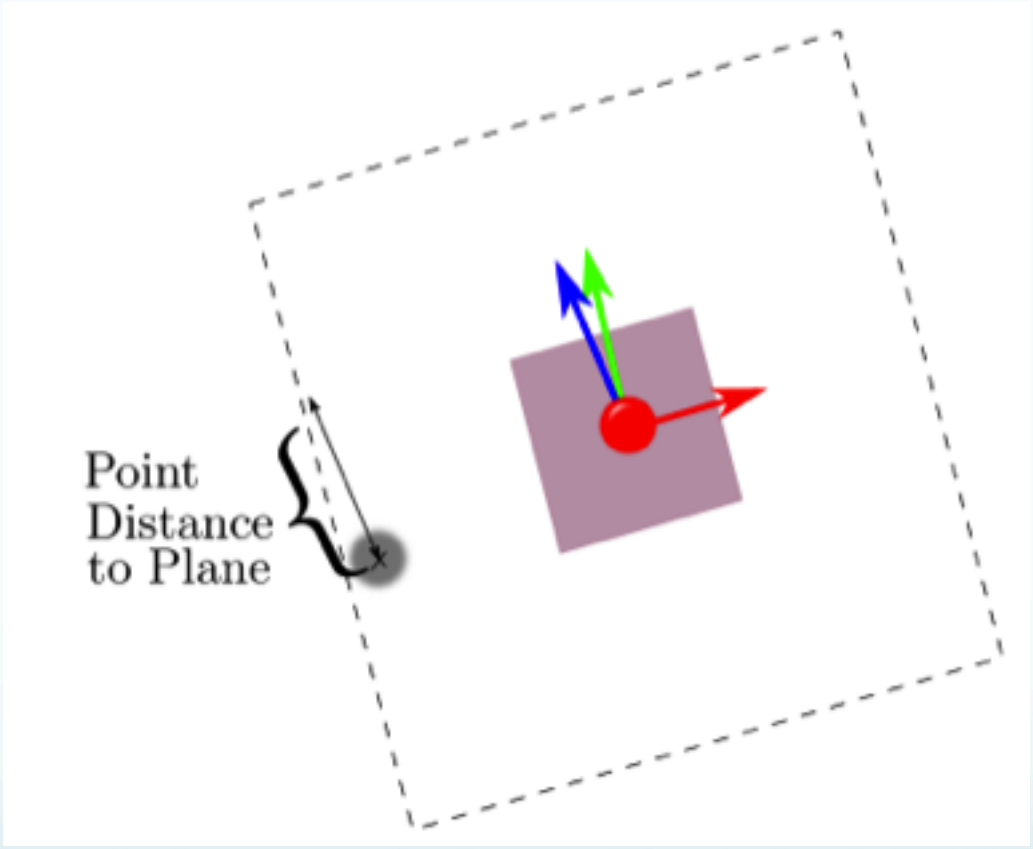
- *Iterative Closest Point* (ICP) uses shape to align frames
- Does *not* require the RGB image
- Does need a good initial "guess"
- Repeat the following two steps:
  - For each point in cloud A, find the closest corresponding point in cloud B
  - Compute the transformation that best aligns this set of corresponding pairs

ICP Variants

- Correspondence
  - Outliers as absolute or percentage
  - No many-to-one correspondences
  - Reject boundary points
  - Normal agreement
- Error metric
  - Point-to-point
  - Point-to-plane
  - Weight by color / normal agreement

[Image: Richard Newcombe]

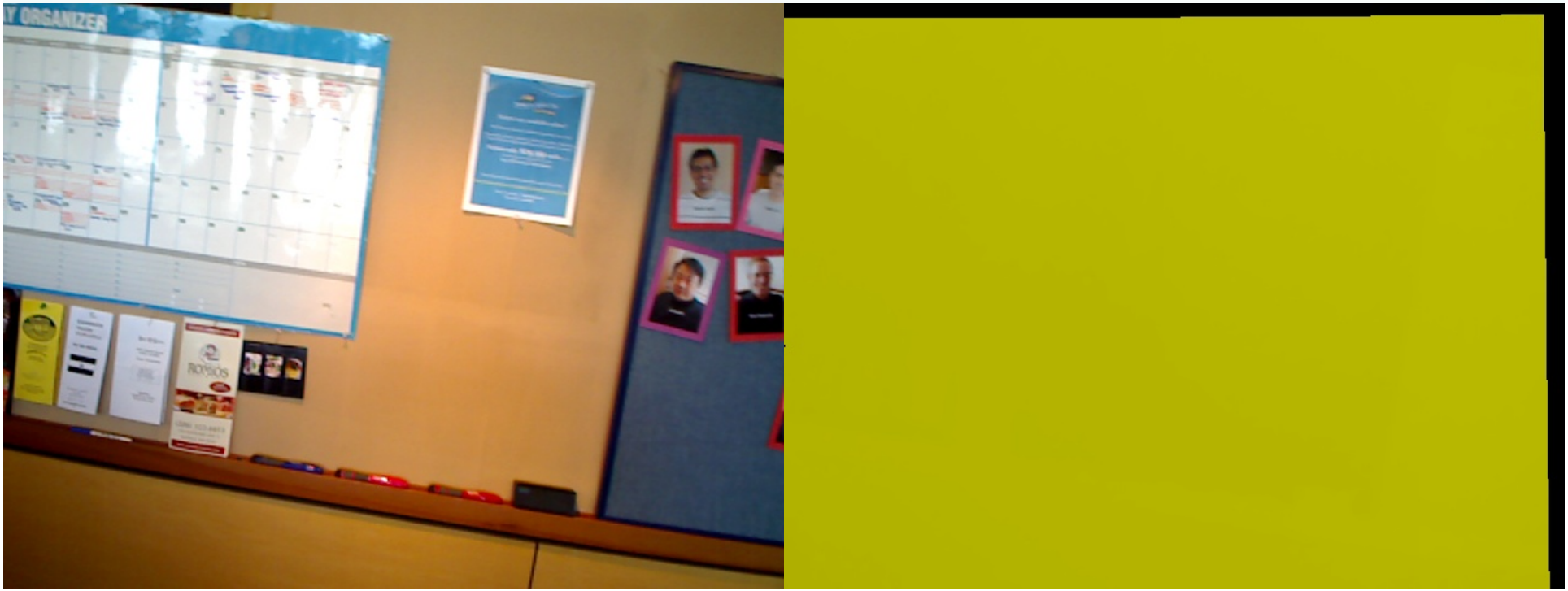
Point-to-plane Error



$$\left(\frac{1}{|A_d|} \sum_{j \in A_d} w_j \left| (\mathbf{T}(p_s^j) - p_t^j) \cdot n_t^j \right|^2 \right)$$

ICP Failure Cases

- Not enough distinctive shape
- Don’t have a close enough initial “guess”
- Here the shape is basically a simple plane…



Joint Optimization (RGBD-ICP)

input : source RGB-D frame P_s, target RGB-D frame P_t, previous transformation $\mathbf{T}_p$
output: Optimized relative transformation $\mathbf{T}^*$

1 $F_s \leftarrow$ `Extract_RGB_Point_Features` (P_s)
2 $F_t \leftarrow$ `Extract_RGB_Point_Features` (P_t)
3 $(\mathbf{T}^*, A_f) \leftarrow$ `Perform_RANSAC_Alignment` (F_s, F_t)
4 **if** $|A_f| < \gamma$ **then**
5 $\quad \mathbf{T}^* = \mathbf{T}_p$
6 $\quad A_f = \emptyset$
7 **end**
8 **repeat**
9 $\quad A_d \leftarrow$ `Compute_Closest_Points` $(\mathbf{T}^*, P_s, P_t)$
10 $\quad \mathbf{T}^* \leftarrow$ `Optimize_Alignment` $(\mathbf{T}^*, A_f, A_d)$
11 **until** (`Change` $(\mathbf{T}^*) \leq \theta$) **or** (Iterations $>$ MaxIterations)
12 **return** $\mathbf{T}^*$

Algorithm 1: RGB-D ICP algorithm for matching two RGB-D frames.

Optimal Transformation

$$\mathbf{t}^* = \underset{\mathbf{t}}{\operatorname{argmin}} \left[\left(\frac{1}{|A_f|} \sum_{i \in A_f} \left| Proj(\mathbf{t}(f_s^i)) - Proj(f_t^i) \right|^2 \right) + \beta \left(\frac{1}{|A_d|} \sum_{j \in A_d} w_j \left| (\mathbf{t}(p_s^j) - p_t^j) \cdot n_t^j \right|^2 \right) \right]$$

Two-Stage Alternative

input : source RGB-D frame P_s and target RGB-D frame P_t
output: optimized relative transformation $\mathbf{T}^*$

1 $F_s \leftarrow$ `Extract_RGB_Point_Features` (P_s)
2 $F_t \leftarrow$ `Extract_RGB_Point_Features` (P_t)
3 $(\mathbf{T}^*, A_f) \leftarrow$ `Perform_RANSAC_Alignment` (F_s, F_t)
4 **if** $|A_f| < \gamma$ **then**
5 $\quad \mathbf{T}^* = \mathbf{T}_p$
6 $\quad A_f = \emptyset$
7 **end**
8 **if** $|A_f| \geq \phi$ **then**
9 $\quad$ **return** $\mathbf{T}^*$
10 **else**
11 $\quad$ **repeat**
12 $\quad\quad A_d \leftarrow$ `Compute_Closest_Points` $(\mathbf{T}^*, P_s, P_t)$
13 $\quad\quad \mathbf{T}^* \leftarrow$ `Optimize_Alignment` $(\mathbf{T}^*, A_f, A_d)$
14 $\quad$ **until** *(*`Change` $(\mathbf{T}^*) \leq \theta$*)* **or** *(*Iterations $>$ MaxIterations*)*
15 $\quad$ **return** $\mathbf{T}^*$
16 **end**

Algorithm 2: Two-Stage RGB-D ICP algorithm for more quickly matching two RGB-D frames.

Experiments

- Reprojection error is better for RANSAC:

	EE-RANSAC	RE-RANSAC
Mean inliers per frame	60.3	116.7

- Errors for variations of the algorithm:

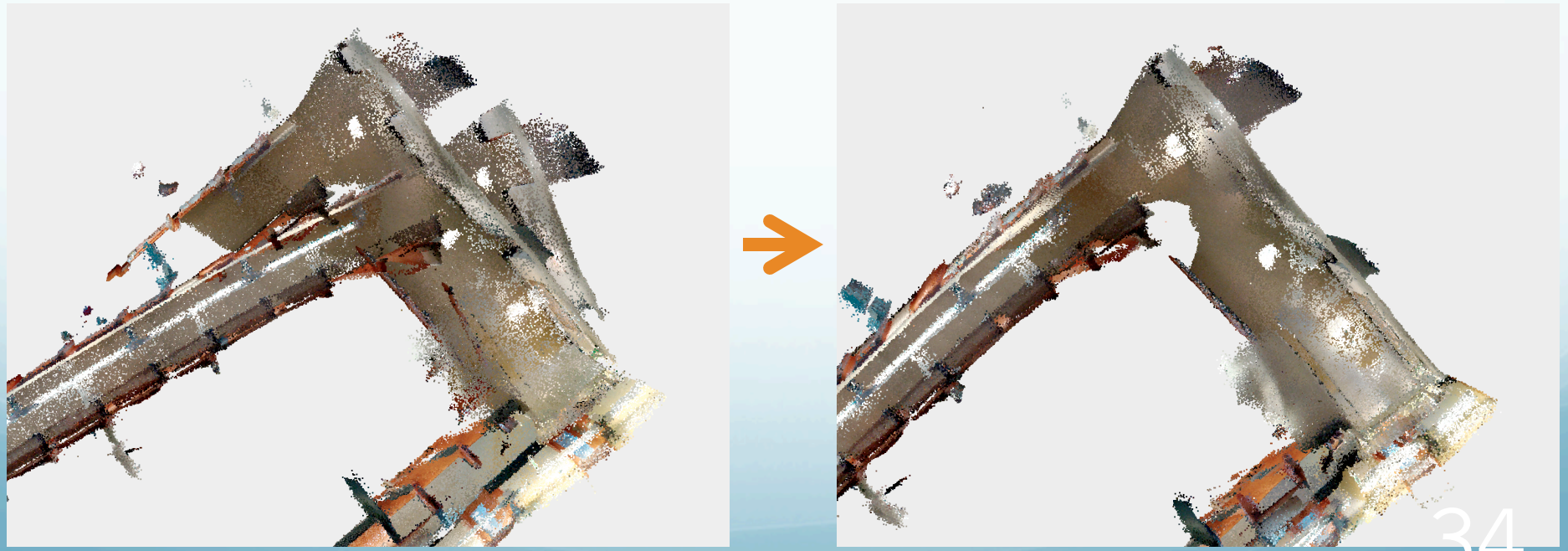
	RE-RANSAC	EE-RANSAC	ICP	RGBD-ICP	Two-Stage RGBD-ICP
Intel-Day	0.11 (±0.05)	0.16 (±0.07)	0.15 (±0.05)	**0.10** (±**0.04**)	0.11 (±0.05)
Intel-Night	1.09 (±0.88)	1.15 (±0.89)	0.17 (±0.06)	**0.15** (±**0.08**)	0.15 (±0.09)

- Timing for variations of the algorithm:

	RE-RANSAC	EE-RANSAC	ICP	RGBD-ICP	Two-Stage RGBD-ICP
Intel-Day	0.21 (±0.03)	0.20 (±0.05)	0.72 (±0.73)	0.48 (±0.10)	0.21 (±0.03)
Intel-Night	0.20 (±0.05)	0.20 (±0.05)	0.43 (±0.64)	0.57 (±0.47)	0.37 (±0.63)

Loop Closure

- Sequential alignments accumulate error
- Revisiting a previous location results in an inconsistent map

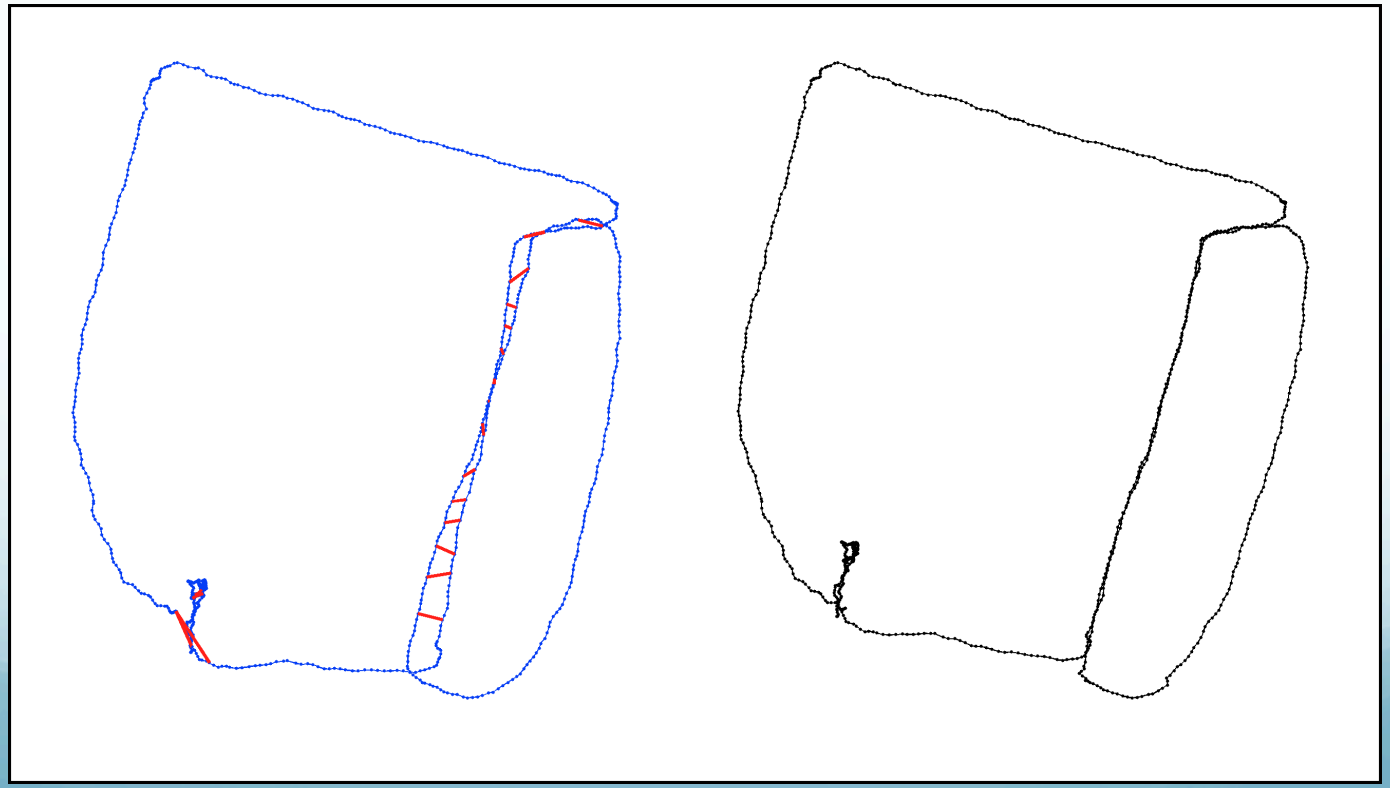


Loop Closure Detection

- Detect by running RANSAC against previous frames
- Pre-filter options (for efficiency):
  - Only a subset of frames (*keyframes*)
  - Only keyframes with similar estimated 3D pose
  - Place recognition using vocabulary tree
    - *Scalable recognition with a vocabulary tree*, David Nister and Henrik Stewenius, 2006
- Post-filter (avoid false positives)
  - Estimate maximum expected drift and reject detections changing pose too greatly

Loop Closure Correction (TORO)

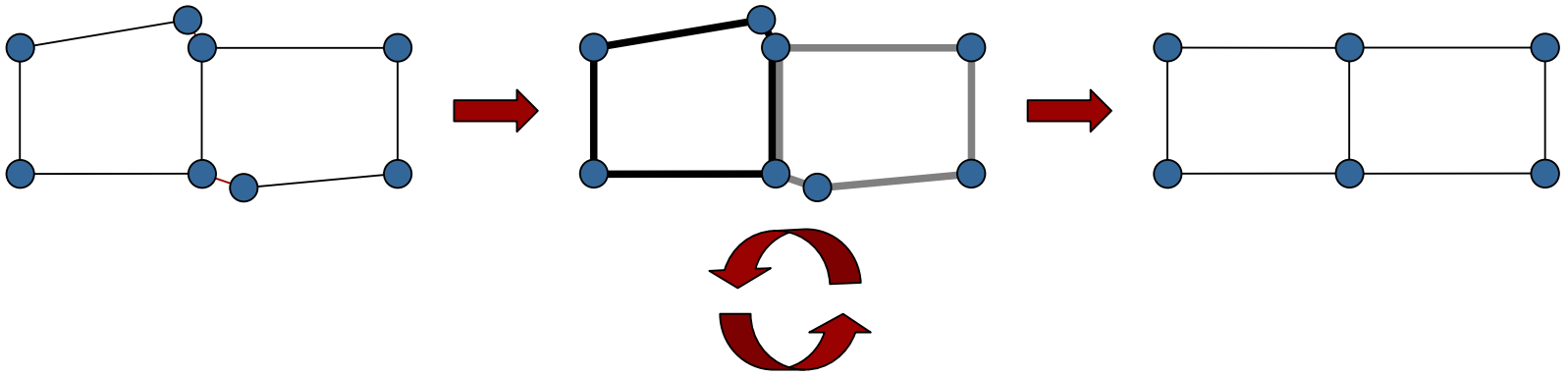
- TORO [Grisetti 2007, 2009]:
  - Constraints between camera locations in *pose graph*
  - Maximum likelihood global camera poses



[Slide: Cyrill Stachniss]

Stochastic Gradient Descent

- Minimize the error individually for each constraint (Decomposition of the problem into sub-problems)
- Solve one step of each sub-problem
- Solutions might be contradictory
- The magnitude of the correction decreases with each iteration
- The individual solutions are merged and via the learning rate converge to an equilibrium

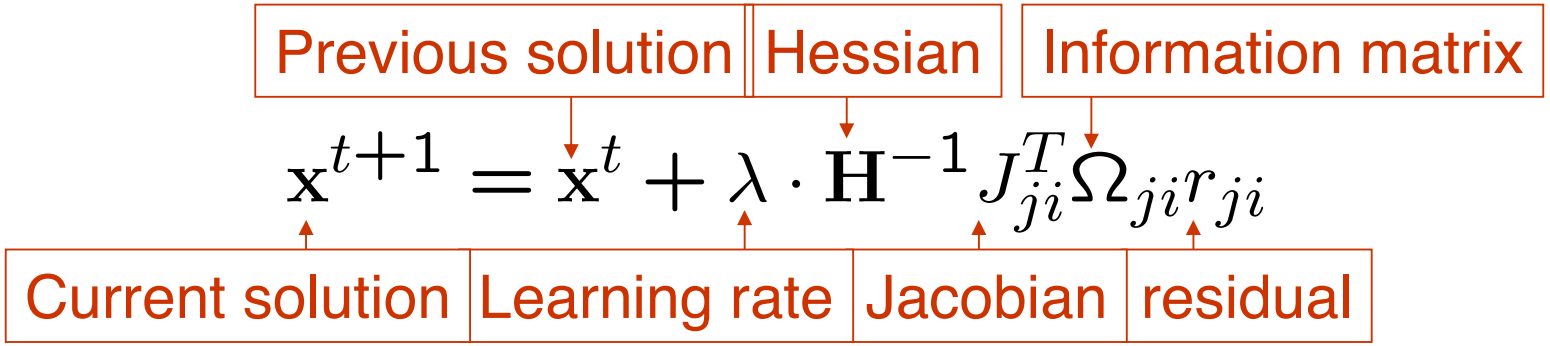


[First introduced in the SLAM community by Olson et al., '06]

[Slide: Cyrill Stachniss]

Preconditioned SGD

- Minimize the error individually for each constraint (Decomposition of the problem into sub-problems)
- Solve one step of each sub-problem
- Solutions might be contradictory
- A solution is found when an equilibrium is reached
- Update rule for a single constraint:



[First introduced in the SLAM community by Olson et al., '06]

[Slide: Cyrill Stachniss]

Node Parameterization

- How to represent the node in the graph?
- **Key question: which parts need to be updated for a single constraint update?**
- This are to the "sub-problems" in SGD
- Transform the problem into a different space so that:
  - the structure of the problem is exploited.
  - the calculations become easier and faster.

parameters

poses

$$\mathbf{x} = g(\mathbf{p}) \leftrightarrow \mathbf{p} = g^{-1}(\mathbf{x})$$

Mapping function

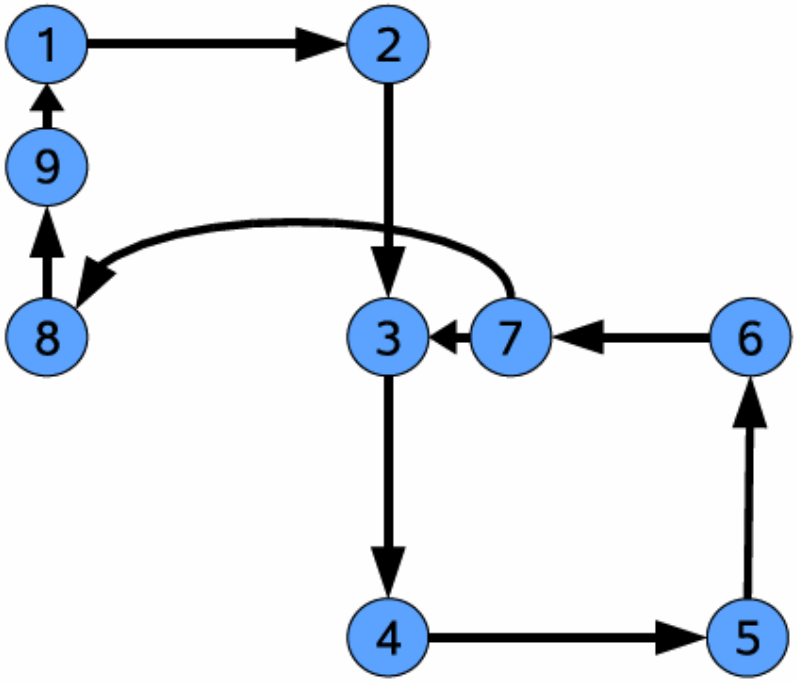
$$\mathbf{x}^* = \operatorname{argmin}_{\mathbf{x}} \sum_{i,j} e'_{ji}(\mathbf{x})^T \Omega_{ji} e'_{ji}(\mathbf{x})$$

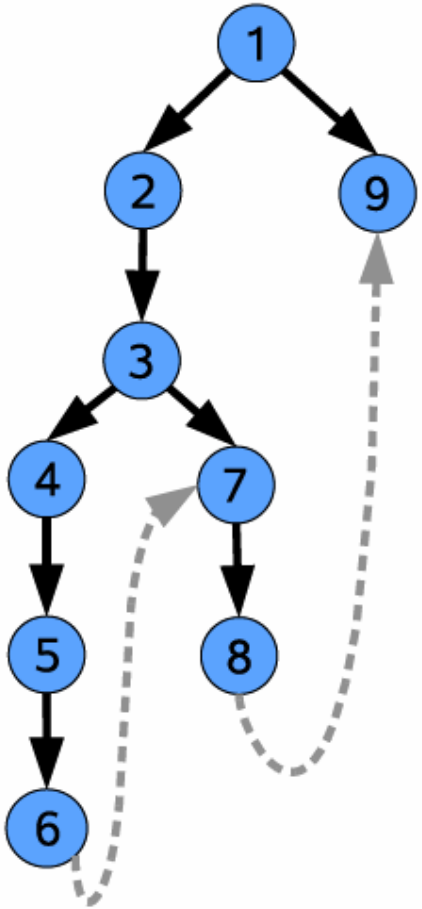
transformed problem

[Slide: Cyrill Stachniss]

Tree Parameterization

- Use a spanning tree!

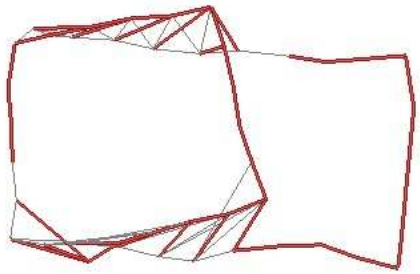


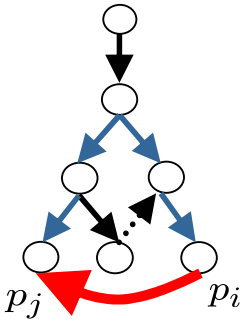


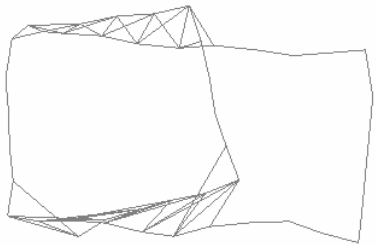
[Slide: Cyrill Stachniss]

Stochastic Gradient Descent using the Tree Parameterization

- Using a tree parameterization we decompose the problem in many small sub-problems which are either:
  - constraints on the tree ("open loop")
  - constraints not in the tree ("a loop closure")
- Each SGD equation independently solves one sub-problem at a time
- The solutions are integrated via the learning rate







[Slide: Cyrill Stachniss]

Computation of the Update Step

- 3D rotations lead to a highly nonlinear system.
  - Update the poses directly according to the SGD equation may lead to poor convergence.
  - This effect increases with the connectivity of the graph.
- Key idea in the SGD update:

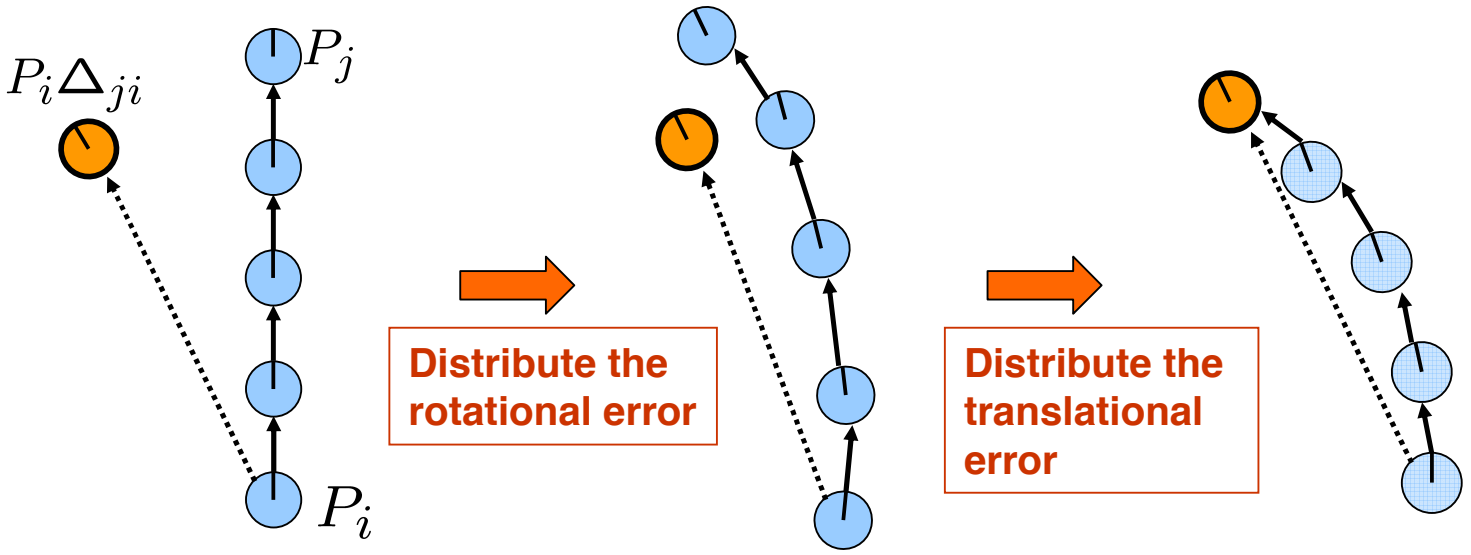
$$\mathbf{\Delta x} = \lambda \cdot \mathbf{H}^{-1} J_{ji}^T \Omega_{ji} r_{ji}$$

Distribute a fraction of the residual along the parameters so that the error of that constraint is reduced.

[Slide: Cyrill Stachniss]

Computation of the Update Step

Update in the "spirit" of the SGD:
Smoothly deform the path along the constraints so that the error is reduced.



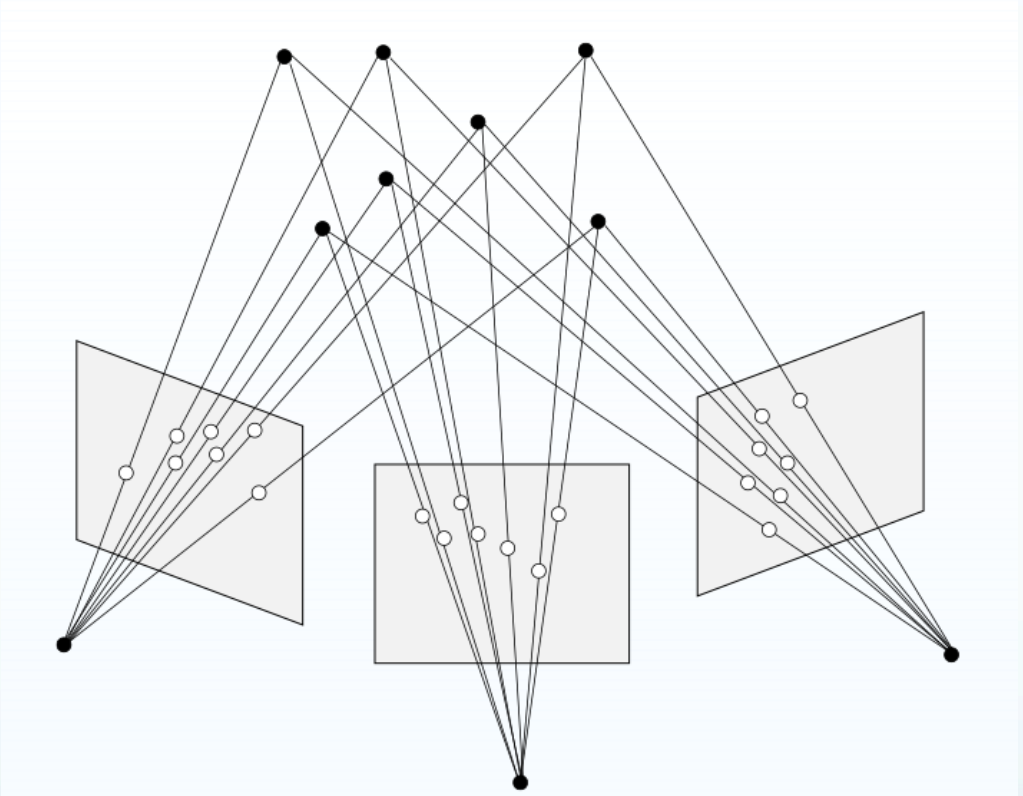
[Slide: Cyrill Stachniss]

Summary of the Algorithm

- Decompose the problem according to the tree parameterization
- Loop
  - Select a constraint
    - Randomly
    - Alternative: sample inverse proportional to the number of nodes involved in the update
  - Compute the nodes involved in the update
    - Nodes according to the parameterization tree
  - Reduce the error for this sub-problem
    - Reduce the rotational error (slerp)
    - Reduce the translational error

[Image: Manolis Lourakis]

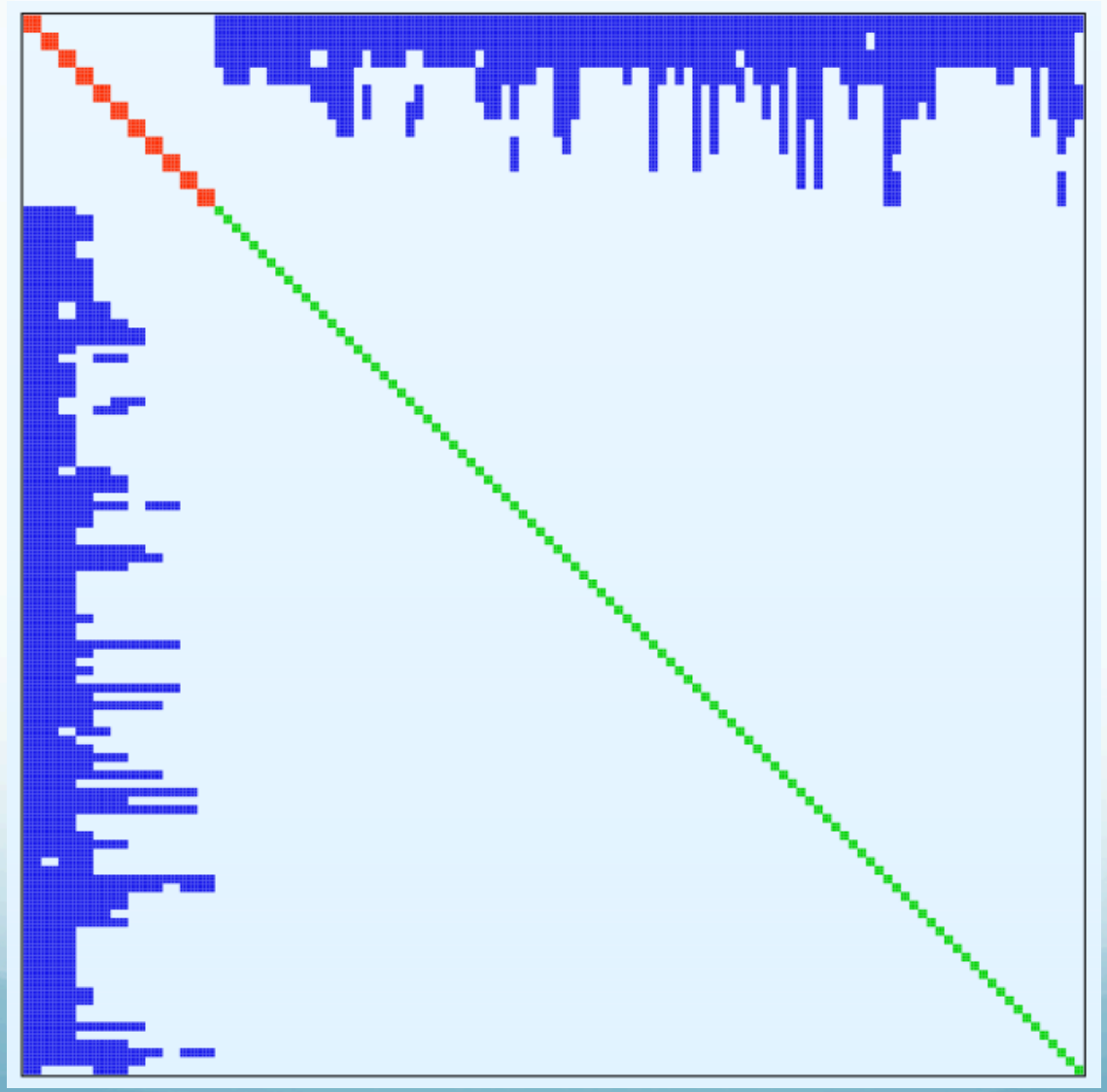
Bundle Adjustment



$$\sum_{c_i \in C} \sum_{p_j \in P} v_{ij} \left| Proj(c_i, p_j) - (\bar{u}, \bar{v}, \bar{d}) \right|^2$$

[Image: Manolis Lourakis]

Sparse Bundle Adjustment



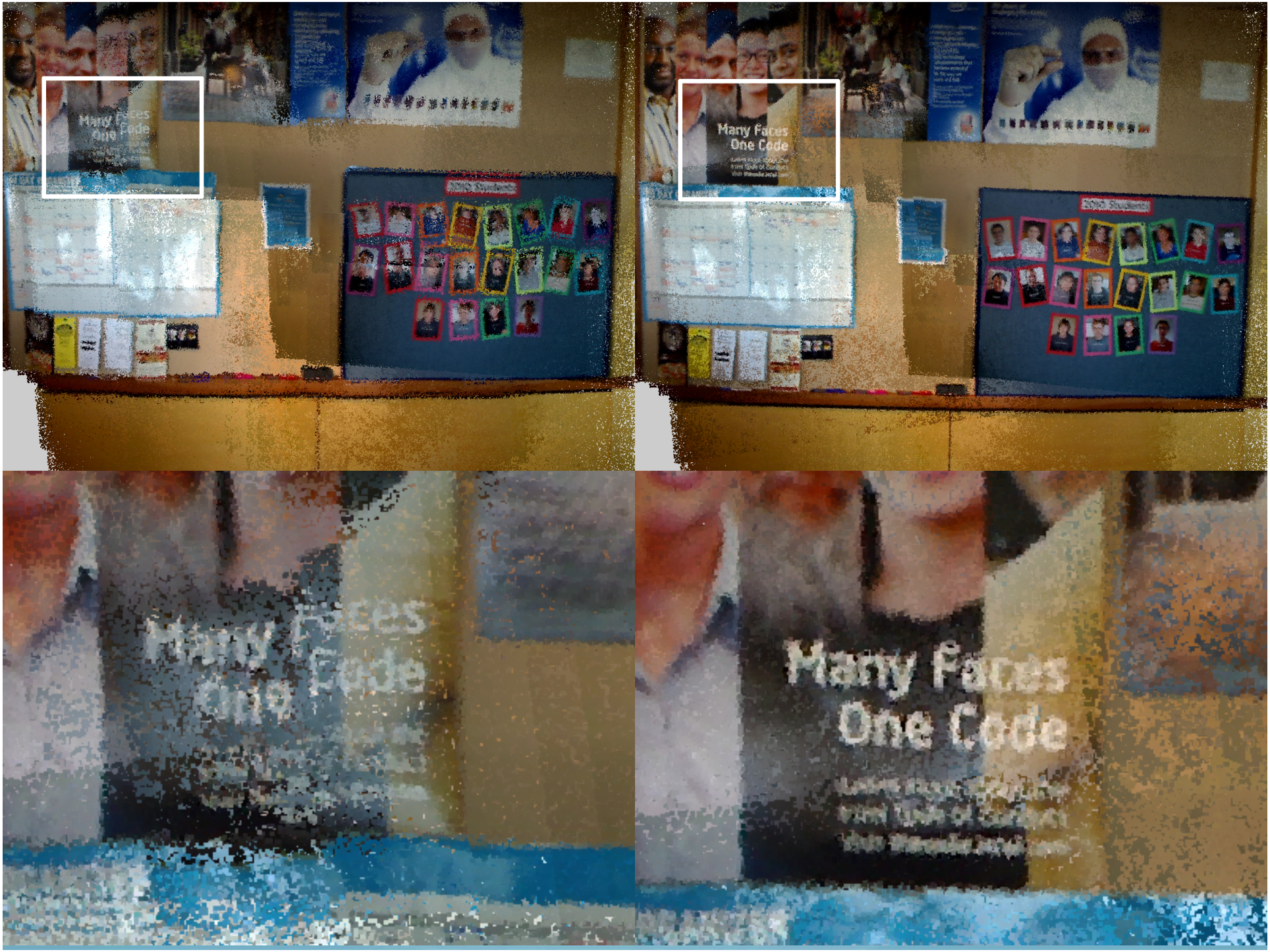
Sparse Sparse Bundle Adjustment Kurt Konolige, 2010

- Two kinds of sparsity:
  - Primary structure (connections only between cameras and points)
  - Secondary structure (connections between cameras are sparse as well)
- Integrates an efficient method for setting up the linear subproblem with recent advances in direct sparse Cholesky solvers
- Implementation available in ROS
- Nonlinear least squares, optimized iteratively

Bundle Adjustment with RGB-D ICP

- ICP-only constraints and RGB-D ICP constraints don't use (only) common feature points
- Solution:
  - Sample points in a grid in image space
  - Corresponding points in other frame according to relative pose
  - Filter based on distance and normal angle threshold
  - Add remaining points as though they were feature points
- Avoids disconnected SBA system

SBA Points

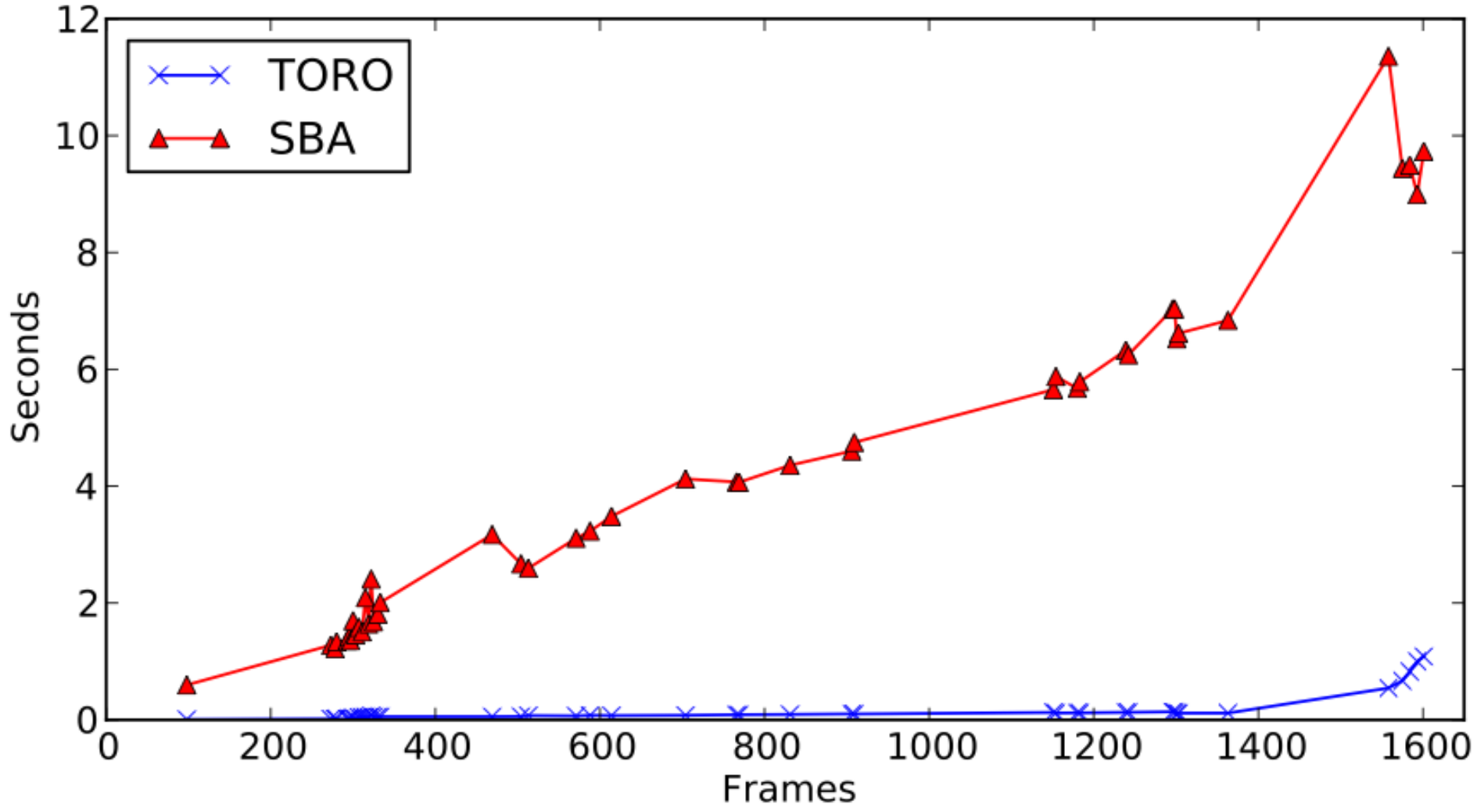

Many Faces
One Code
Many Faces
One Code
Many Faces
One Code

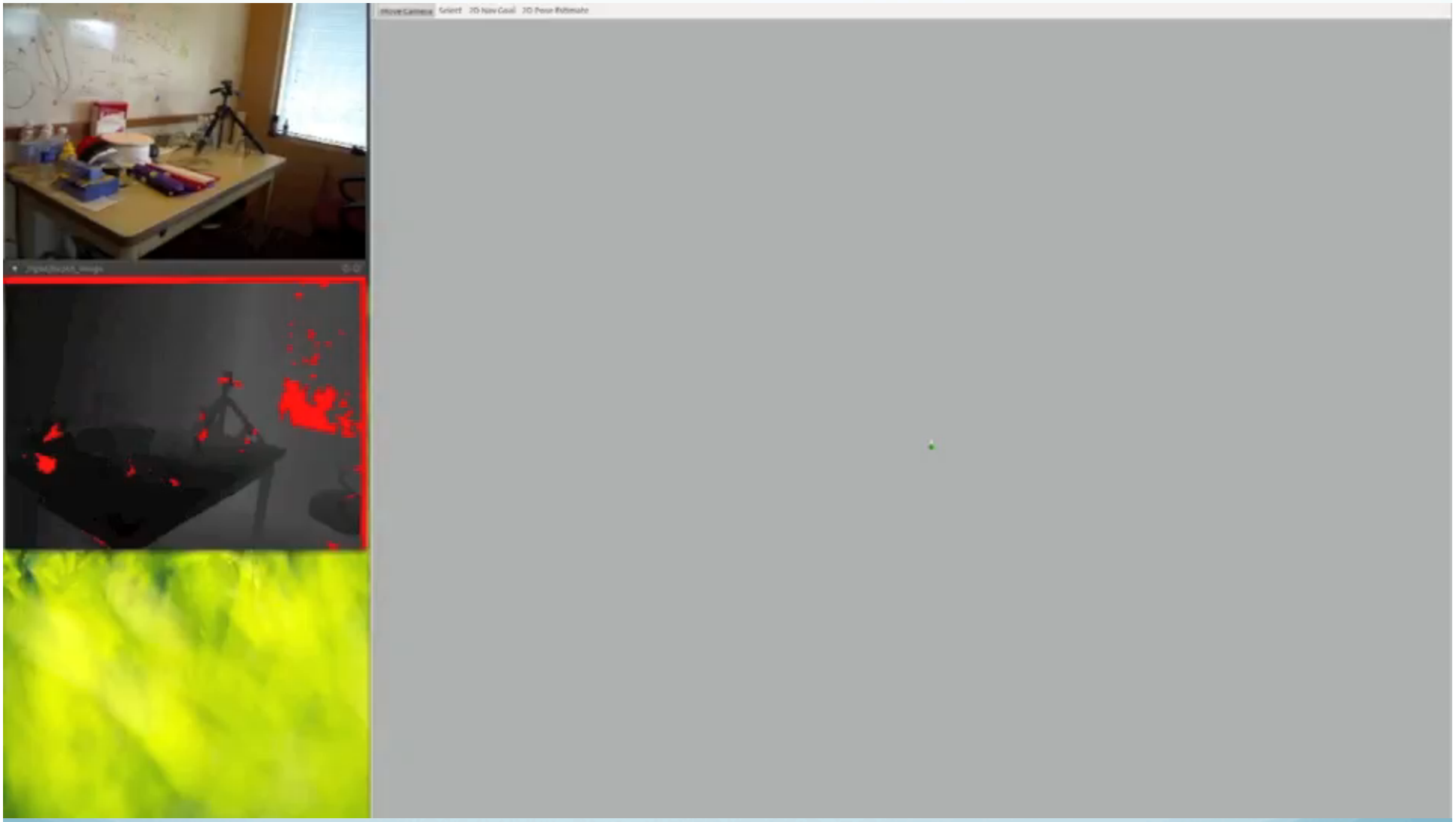
A Second Comparison

TORO

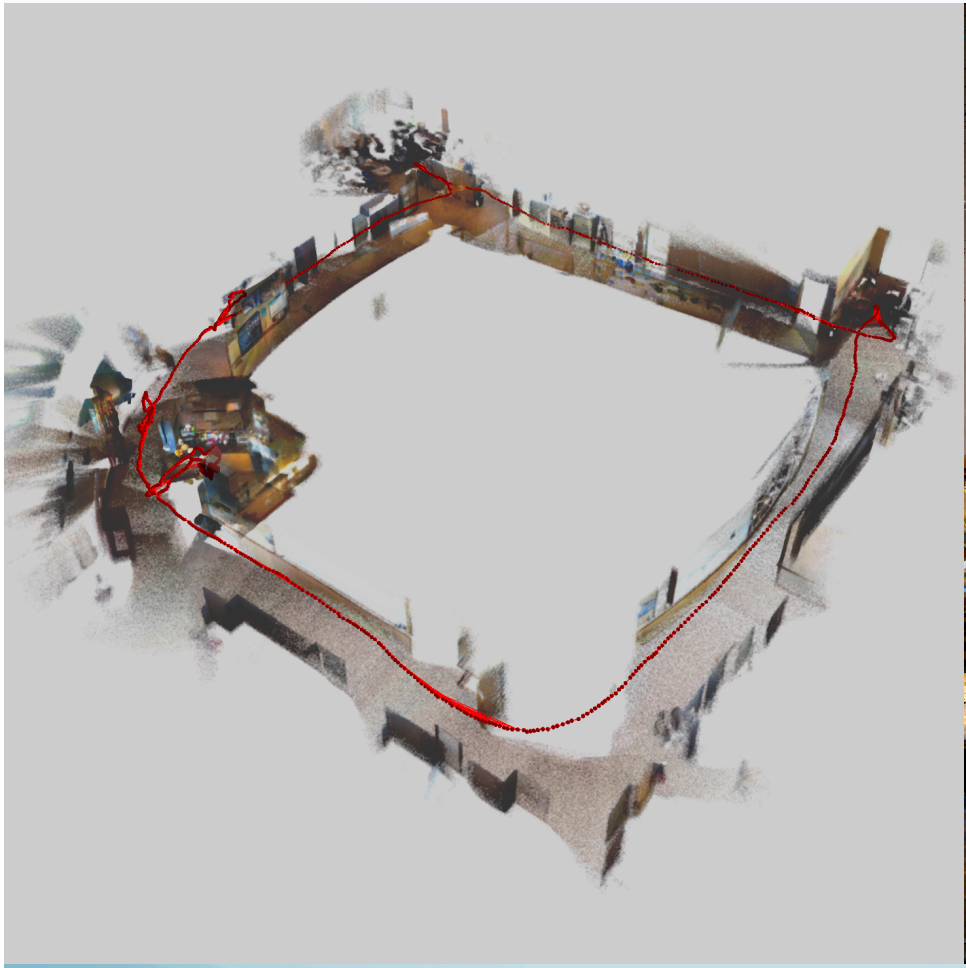
SBA

Timing

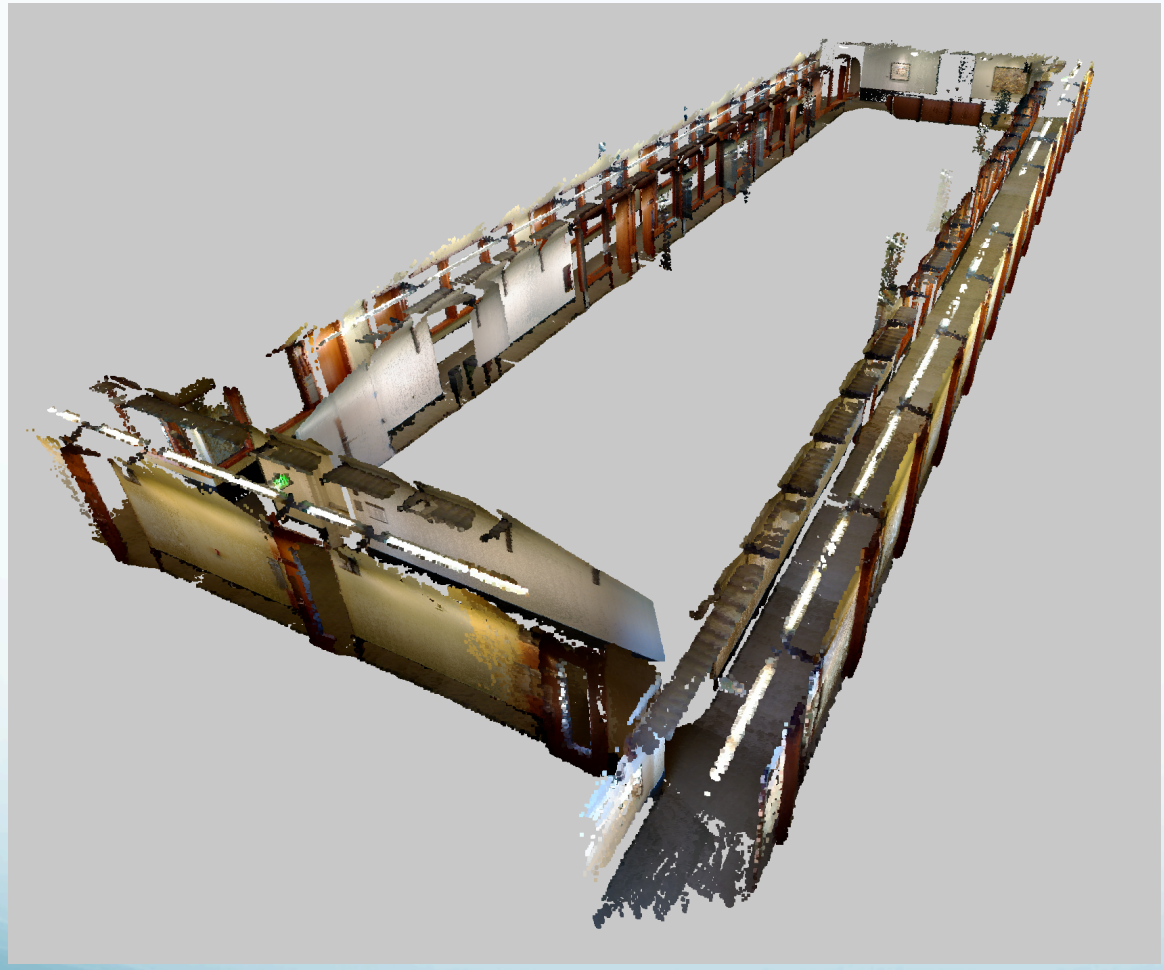

TORO
SBA
Seconds
Frames
0
2
4
6
8
10
12
0
200
400
600
800
1000
1200
1400
1600

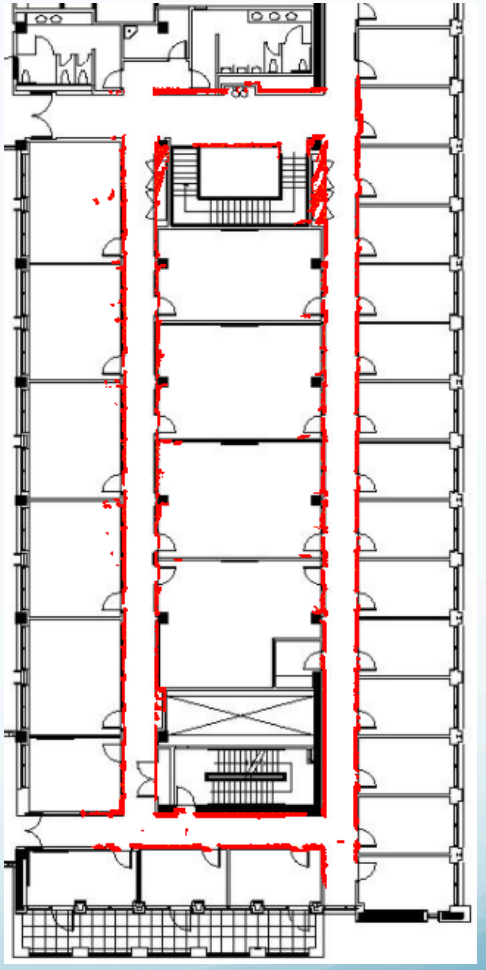


Resulting Map

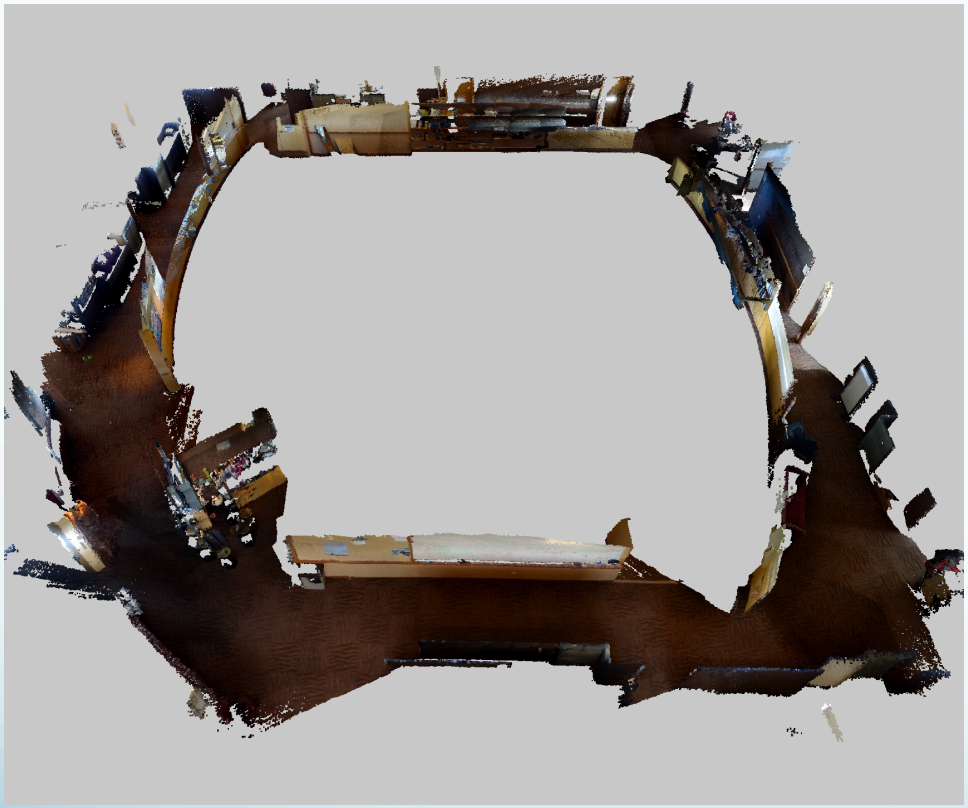


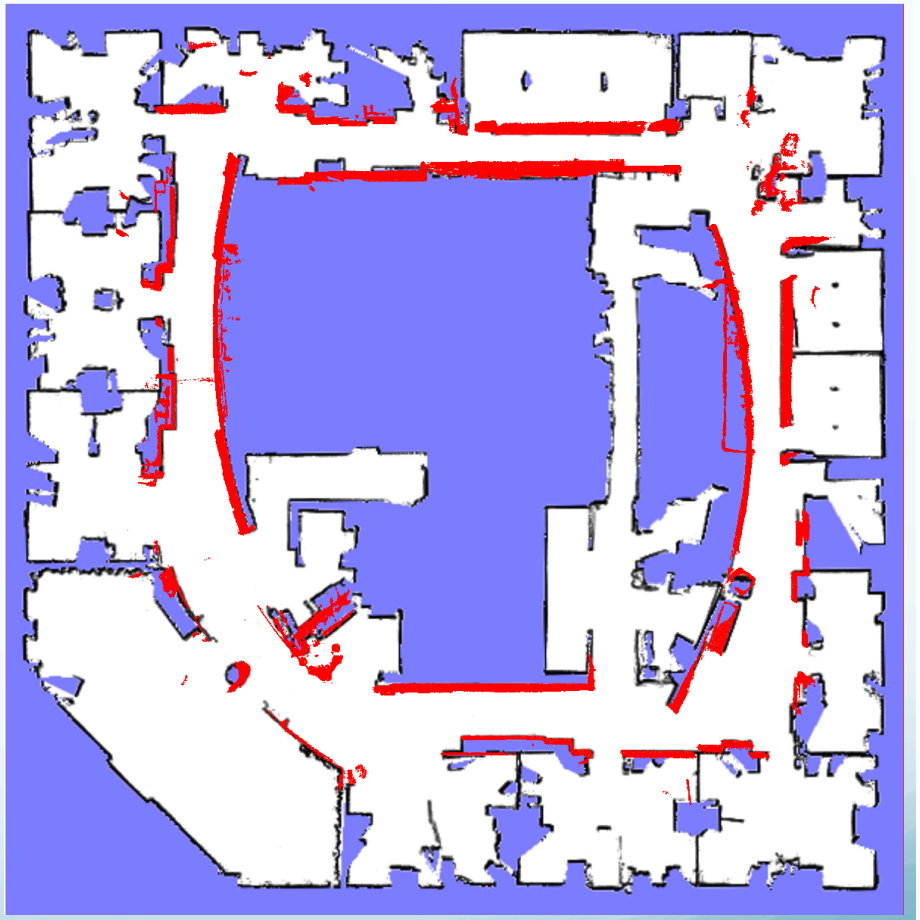
Experiments: Overlay 1





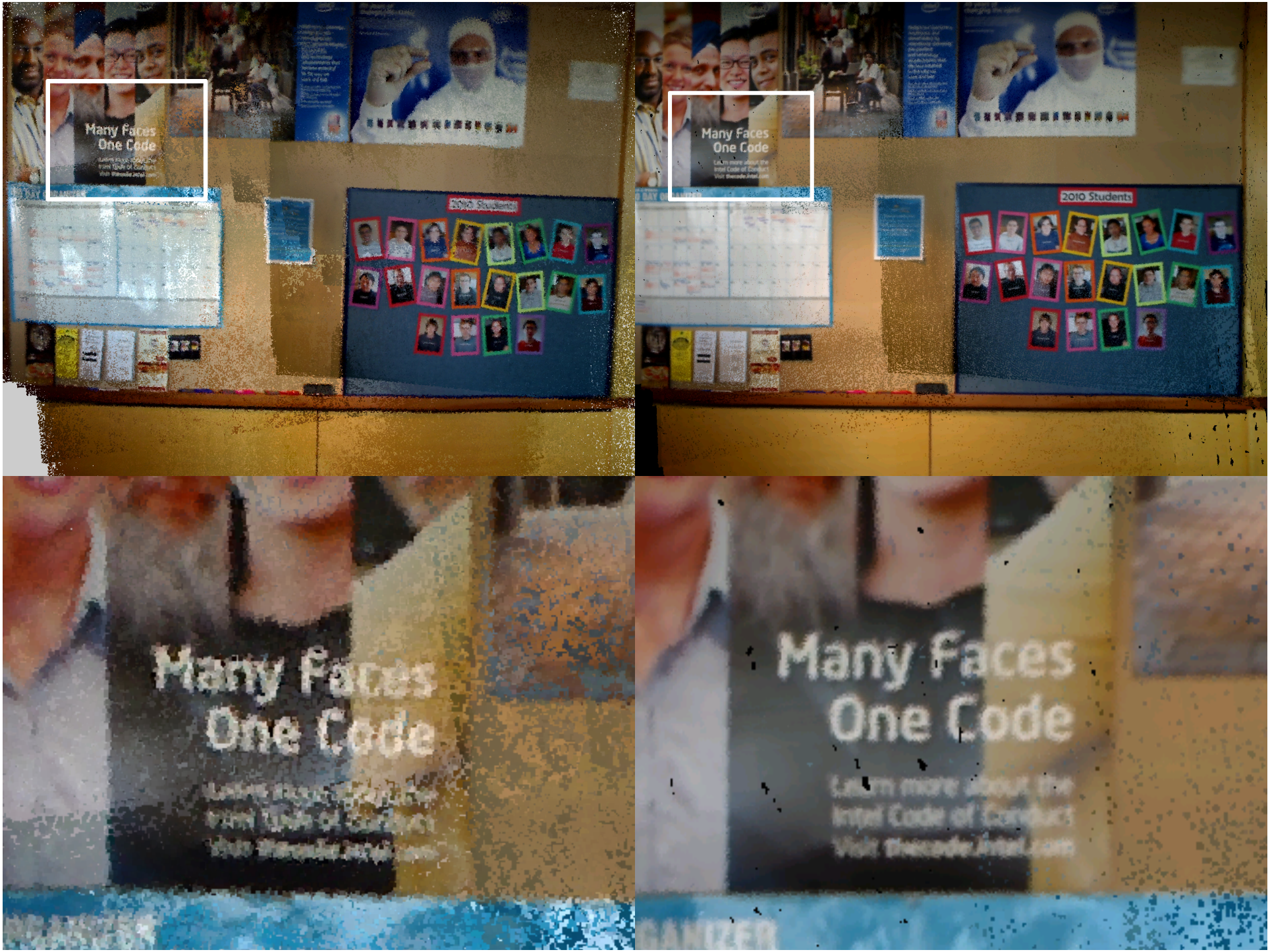
Experiments: Overlay 2





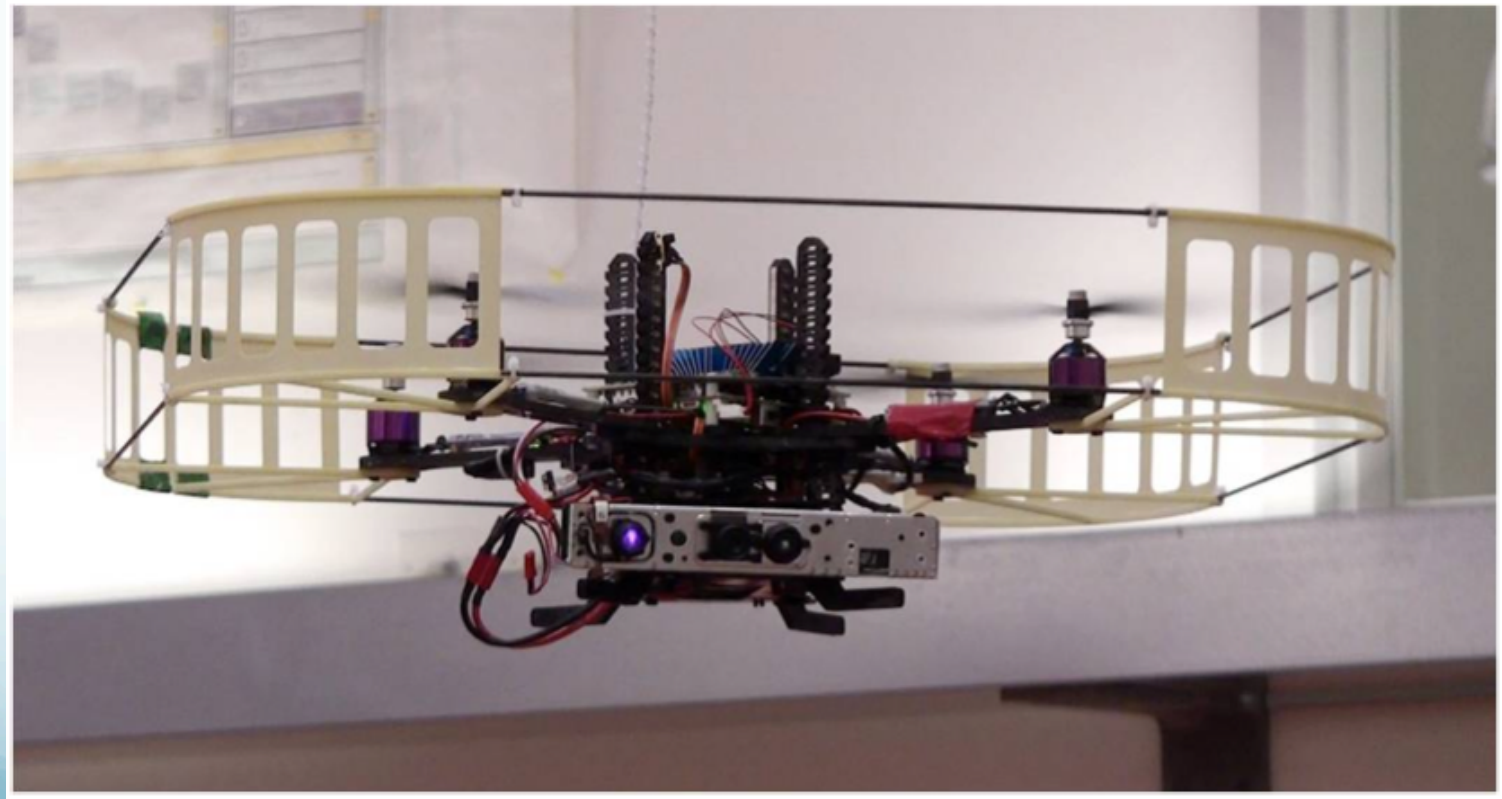
Map Representation: Surfels

- *Surface Elements* [Pfister 2000, Weise 2009, Krainin 2010]
- Circular surface patches
- Accumulate color / orientation / size information
- Incremental, independent updates
- Incorporate occlusion reasoning
- 750 million points reduced to 9 million surfels

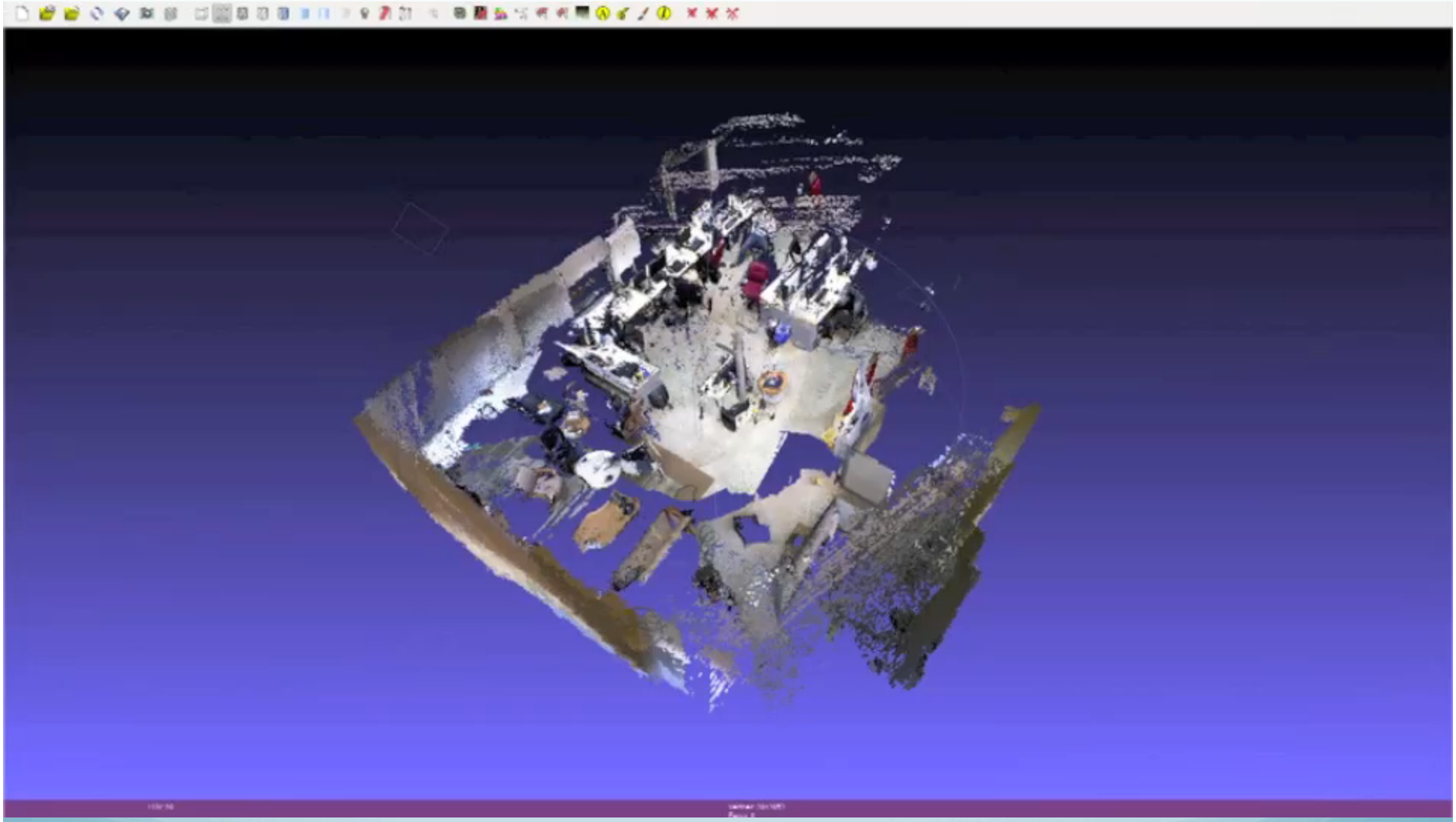

Many Faces
One Code
2010 Students
Many Faces
One Code
2010 Students
Many Faces
One Code
Many Faces
One Code

Application: Quadrocopter

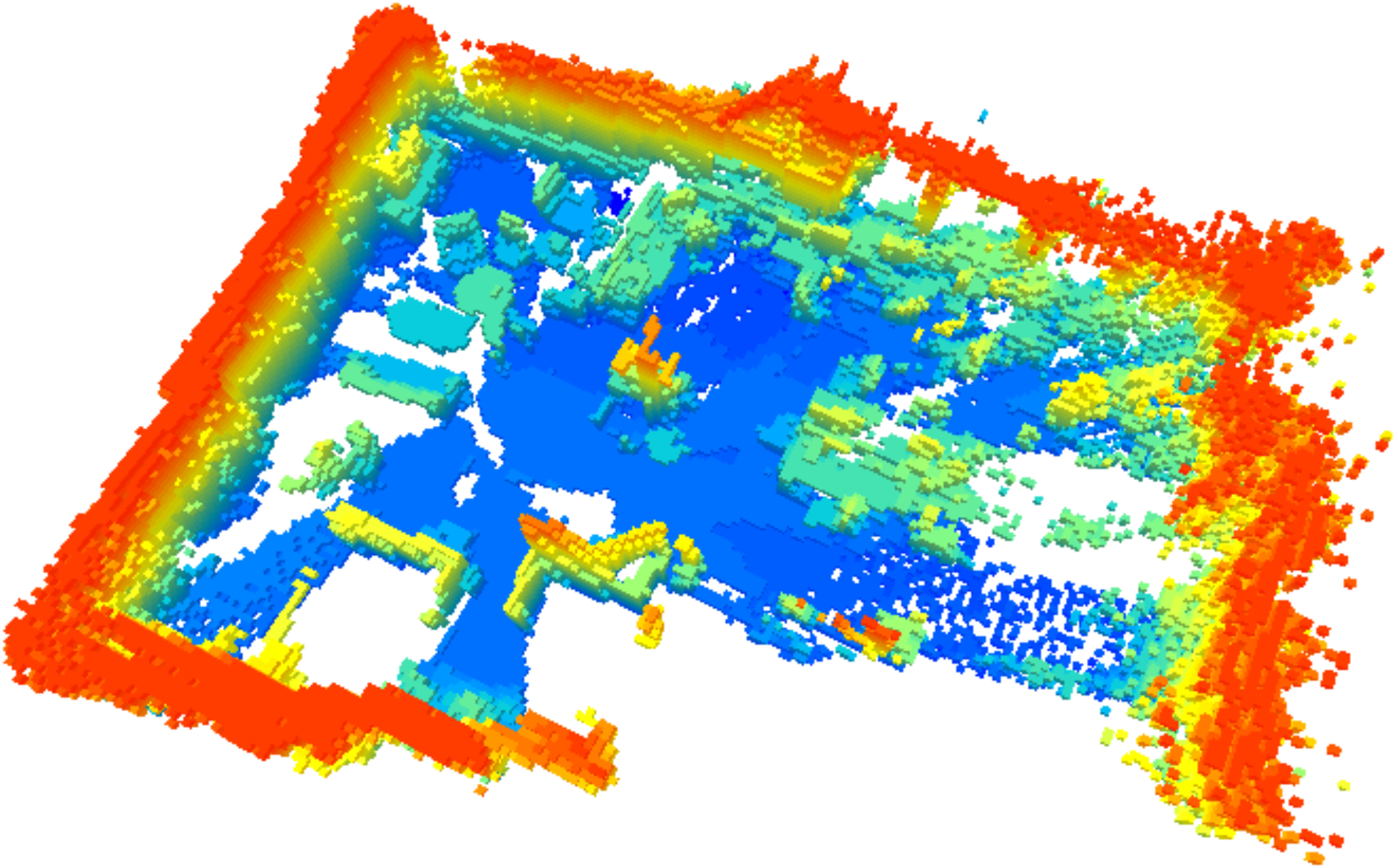
- Collaboration with Albert Huang, Abe Bacharach, and Nicholas Roy from MIT

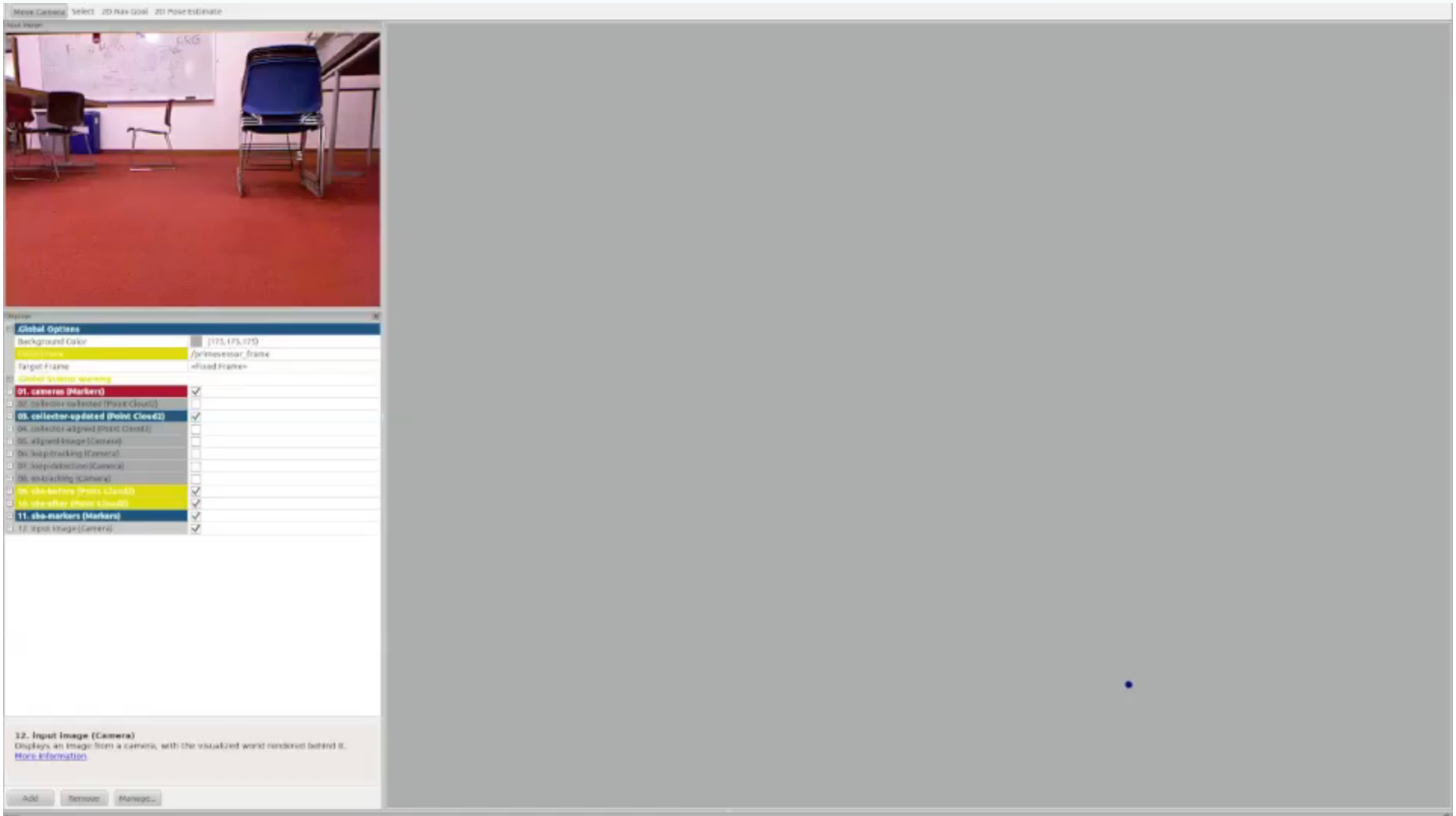


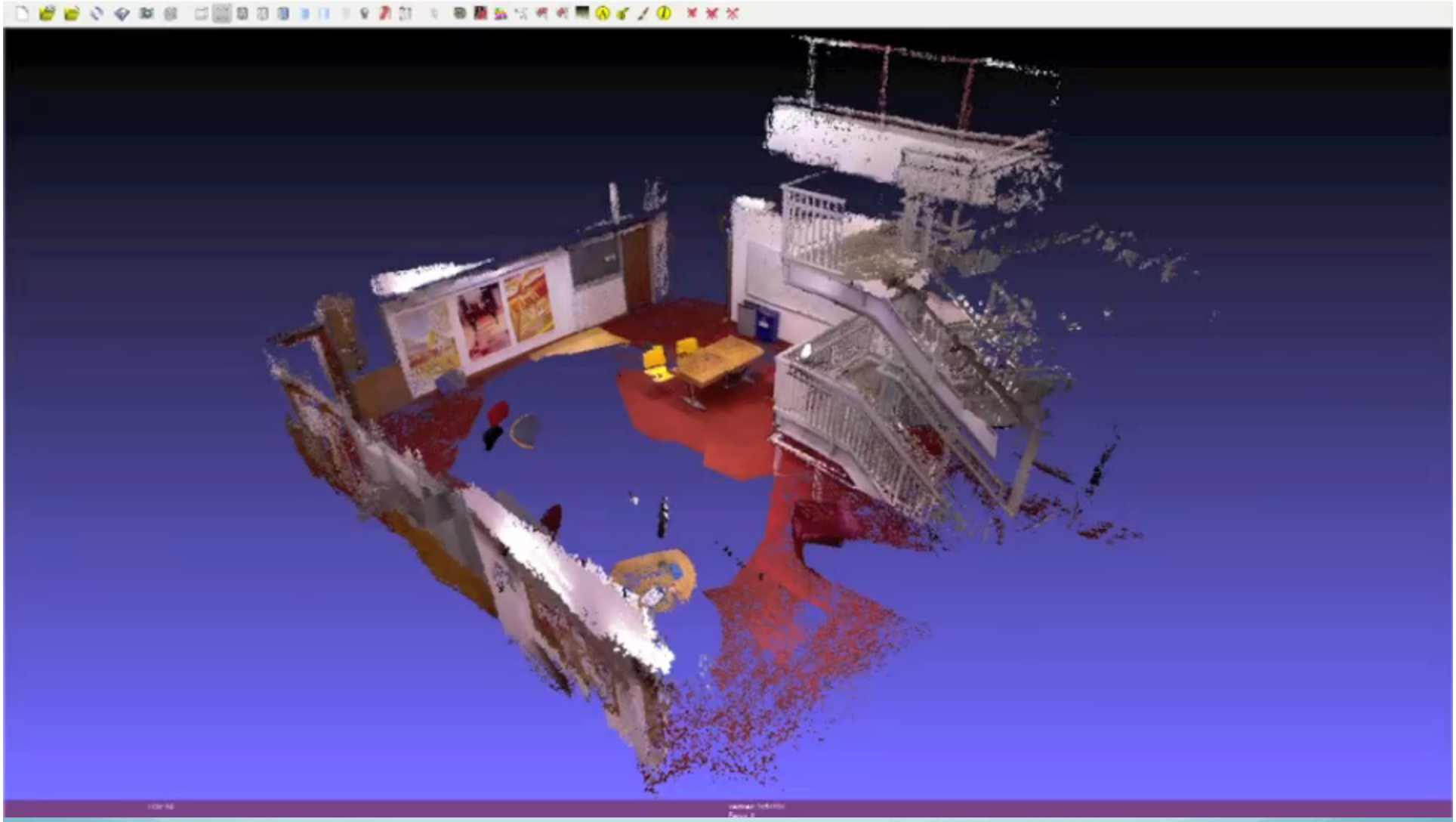

No Image
Global Options
Background Color
Target Frame
01. cameras (Markers)
11. sba markers (Markers)
Image Topic
Alpha
Transport Hint
12. input image (Camera)
Displays an image from a camera, with the visualized world rendered behind it. More information.
Add
Remove
Manage...

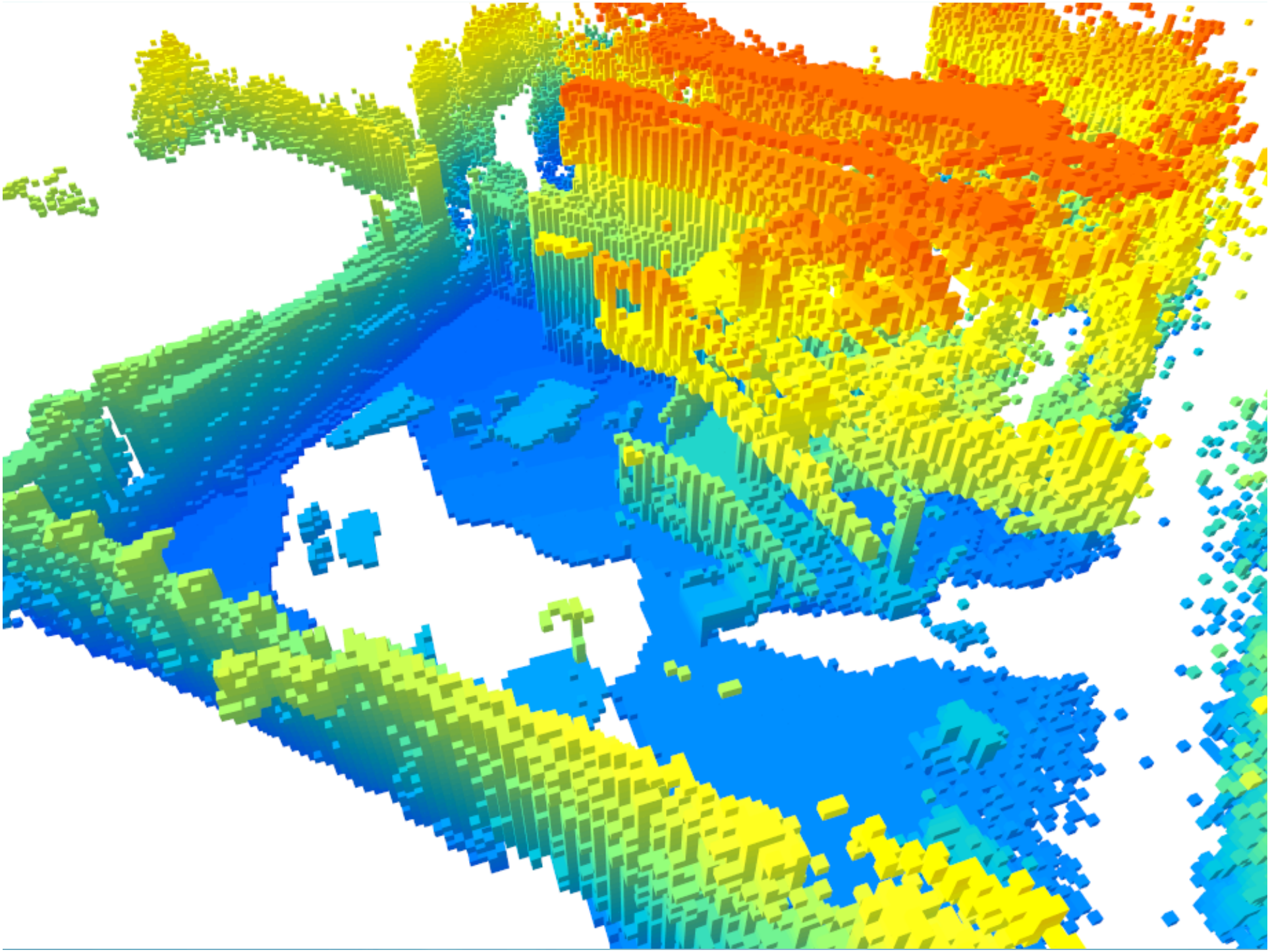


Occupancy Map




Move Camera
Select
2D Nav Goal
2D Pose Estimate
Global Options
Background Color
Target Frame
<Fixed Frame>
01. cameras (Markers)
03. collector-updated (Point Cloud2)
11. sba-markers (Markers)
12. Input Image (Camera)
Displays an image from a camera, with the visualized world rendered behind it.
More Information
Add
Remove
Manage...



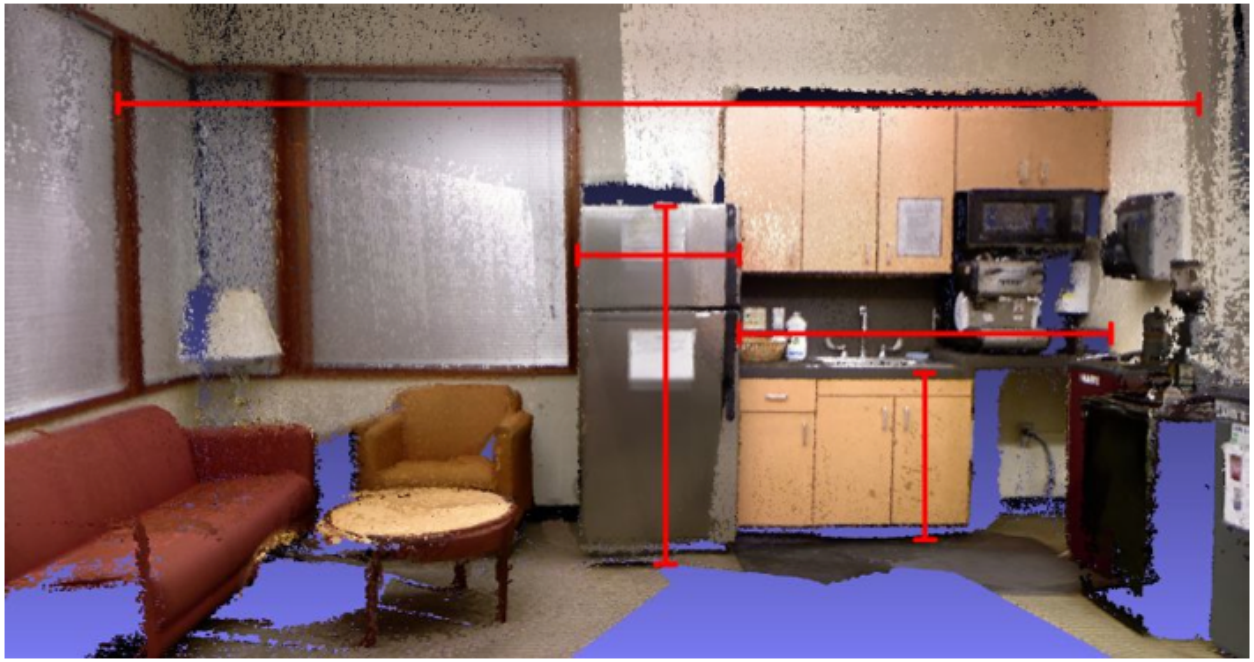


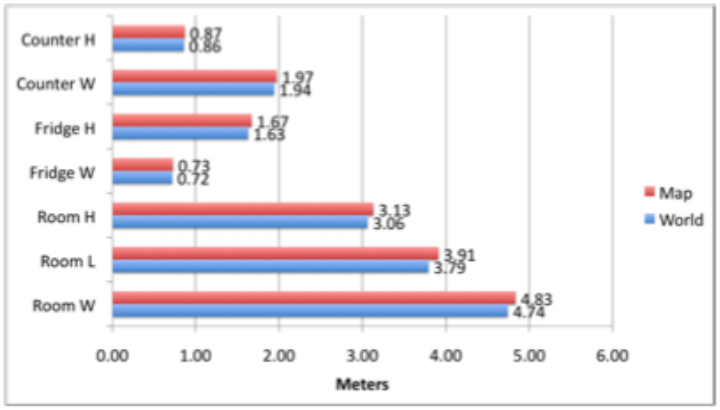
Application: Interactive Mapping

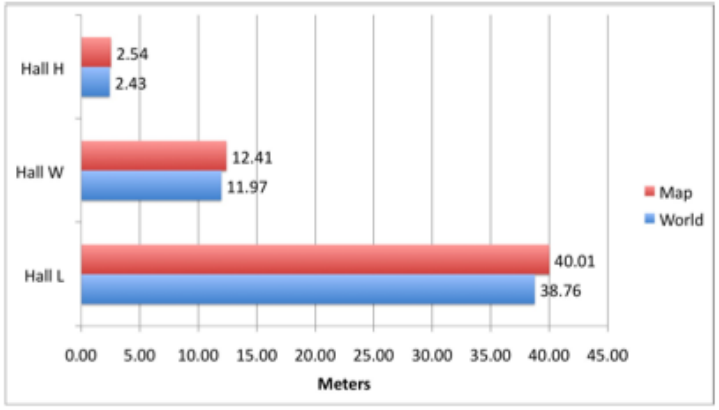
- Allow anyone to construct maps with a Kinect
- Uses for these maps
  - Localization
  - Measurements
  - Remodeling
  - Buy new furniture
  - Video game levels???


Point Cloud View
Latest Frame : 3
Health : Poor
RGB-D Camera View

Application: Measurements







Conclusion

- Kinect-style depth cameras have recently become available as consumer products
- RGB-D Mapping can generate rich 3D maps using these cameras
- RGBD-ICP combines visual and shape information for robust frame-to-frame alignment
- Global consistency achieved via loop closure detection and optimization (RANSAC, TORO, SBA)
- Surfels provide a compact map representation
- ROS + OpenCV are powerful tools to enable these applications

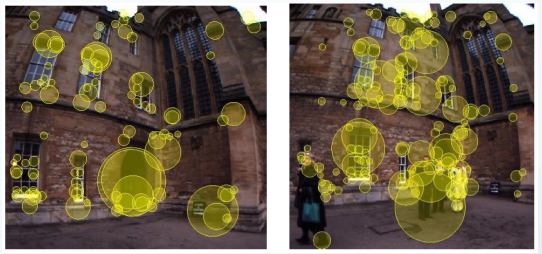
Open Questions

- Which are the best features to use?
- How to find more loop closure constraints between frames?
- What is the right representation (point clouds, surfels, meshes, volumetric, geometric primitives, objects)?
- How to generate increasingly photorealistic maps?
- Autonomous exploration for map completeness?
- Can we use these rich 3D maps for semantic mapping?
- Global optimization with g2o
  - *g2o : A General Framework for Graph Optimization.* Kummerle et al., ICRA, 2011.
  - Allows both pose-graph and SBA optimization, along with possible new error functions

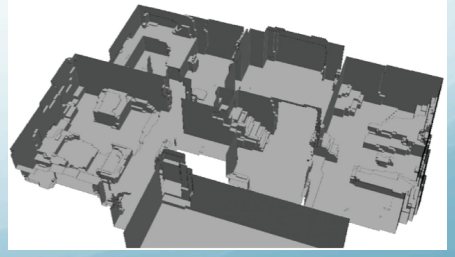
Related Work

- SLAM
  - [Davison et al, PAMI 2007] (monocular)
  - [Konolige et al, IJR 2010] (stereo)
  - [Pollefeys et al, IJCV 2007] (multi-view stereo)
  - [Borrmann et al, RAS 2008] (3D laser)
  - [May et al, JFR 2009] (ToF sensor)

- Loop Closure Detection
  - [Nister et al, CVPR 2006]
  - [Paul et al, ICRA 2010]



- Photo collections
  - [Snavely et al, SIGGRAPH 2006]
  - [Furukawa et al, ICCV 2009]



References (Ours)

- *RGB-D Mapping: Using Kinect-style Depth Cameras for Dense 3D Modeling of Indoor Environments.* Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, Dieter Fox. IJRR, 2012 (to appear).
- *Interactive 3D Modeling of Indoor Environments with a Consumer Depth Camera.* Hao Du, Peter Henry, Xiaofeng Ren, Marvin Cheng, Dan B Goldman, Steven Seitz, Dieter Fox. UbiComp, 2011.
- *Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera.* Albert S. Huang, Abraham Bachrach, Peter Henry, Michael Krainin, Daniel Maturana, Dieter Fox. ISRR, 2011.
- *RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments.* Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, Dieter Fox. ISER, 2010.

References (Others)

- SIFT:
  - Distinctive image features from scale-invariant keypoints. David Lowe, IJCV 2004
- FAST:
  - Machine learning for high-speed corner detection. Edward Rosten, Tom Drummond, ECCV 2006
- Calonder Descriptor:
  - Keypoint signatures for fast learning and recognition. Michael Calonder, Vincent Lepetit, Pascal Fua, ECCV 2008
- TORO:
  - A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. Giorgio Grisetti, Cyrill Stachniss, Slawomir Grzonka, Wolfram Burgard, RSS 2007
  - Nonlinear Constraint Network Optimization for Efficient Map Learning. Giorgio Grisetti, Cyrill Stachniss, Wolfram Burgard, Transactions on Intelligent Transportation Systems, 2009
- sSBA:
  - Sparse Sparse Bundle Adjustment, Kurt Konolige, BMVC 2010

Links

- www.cs.washington.edu/robotics/projects/rgbd-3d-mapping/
- www.ros.org
- The following have nice ROS integration but also work separately:
  - http://opencv.willowgarage.com/wiki/
  - http://www.pointclouds.org/
- www.cs.washington.edu/homes/peter/
- peter@cs.washington.edu