



# Robot Perception

## RGBD Sensors and Algorithms

Dr. Chen Feng

[cfeng@nyu.edu](mailto:cfeng@nyu.edu)

ROB-GY 6203, Fall 2022



# Overview

---

- + Depth from Stereo (Stereo Calibration and Stereo Matching)
- + Monocular Depth Estimation by CNN
- \* ICP
  - ++ Procrustes Analysis
- + TSDF & Kinect Fusion
- + Fast Plane Extraction (PEAC)

\*: know how to code

++: know how to derive

+: know the concept



# References

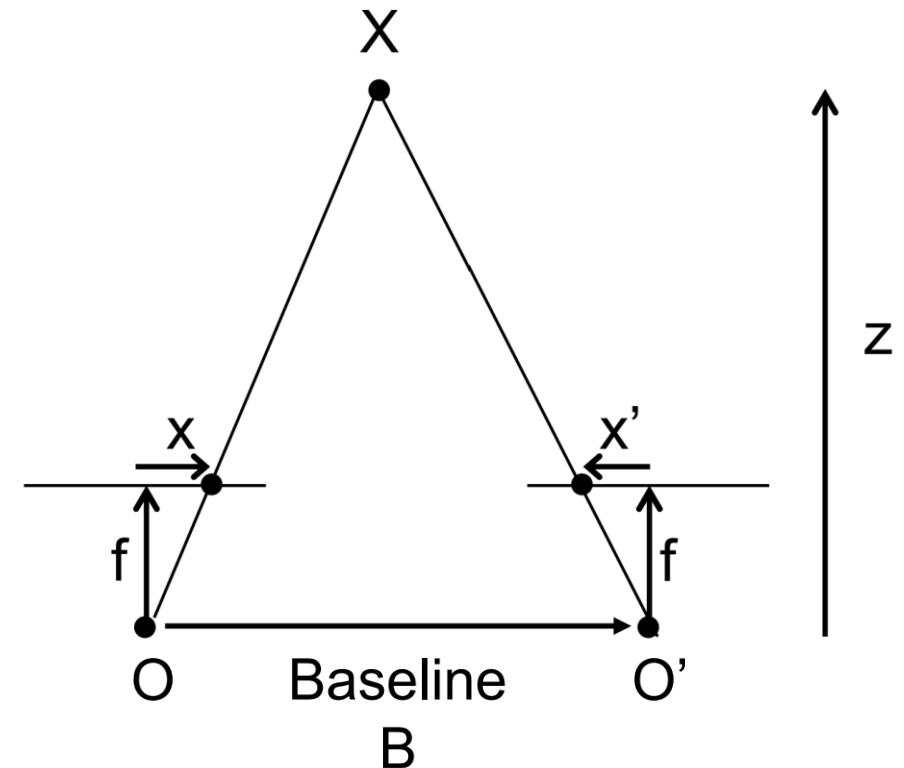
- Co2011:
  - Section 14.3
- Sz2022:
  - Chapter 12
- FP2011:
  - Chapter 7, Section 12.1, 14.3
- HZ2003:
  - Section 11.12
- Umeyama, S. (1991). Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (4), 376-380.
- Low, K.L., 2004. Linear least-squares optimization for point-to-plane icp surface registration. *University of North Carolina Chapel Hill*, 4(10).
- Garg, Ravi, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. "Unsupervised cnn for single view depth estimation: Geometry to the rescue." In European conference on computer vision, pp. 740-756. Springer, Cham, 2016.
- Feng, C., Taguchi, Y. and Kamat, V.R., (2014). Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6218-6225).
- Park, J., Zhou, Q. Y., & Koltun, V., (2017). Colored point cloud registration revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 143-152).



# Depth from Stereo Images

- Disparity and depth
  - Inversely related
- Stereo camera infers pixel depth from disparity
- Longer baseline==better depth accuracy

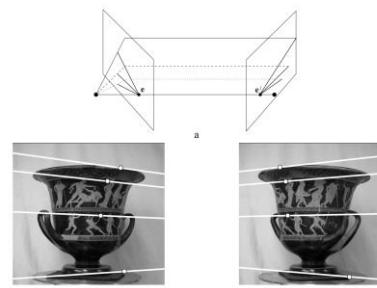
$$\frac{x - x'}{O - O'} = \frac{f}{z}$$



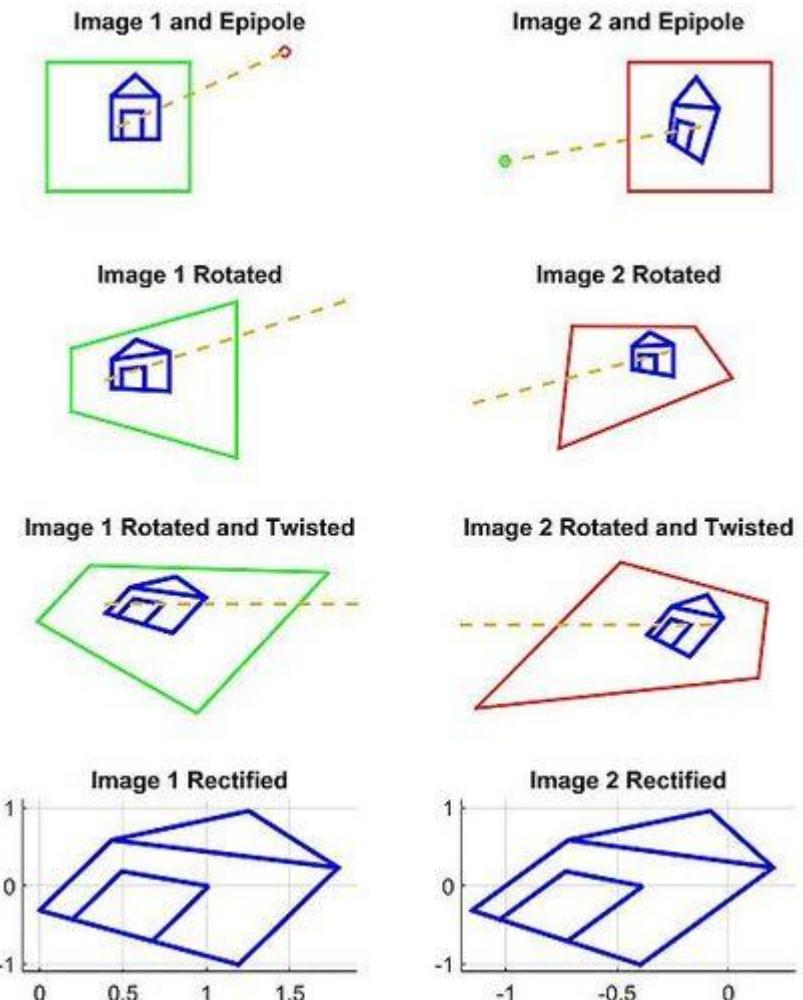
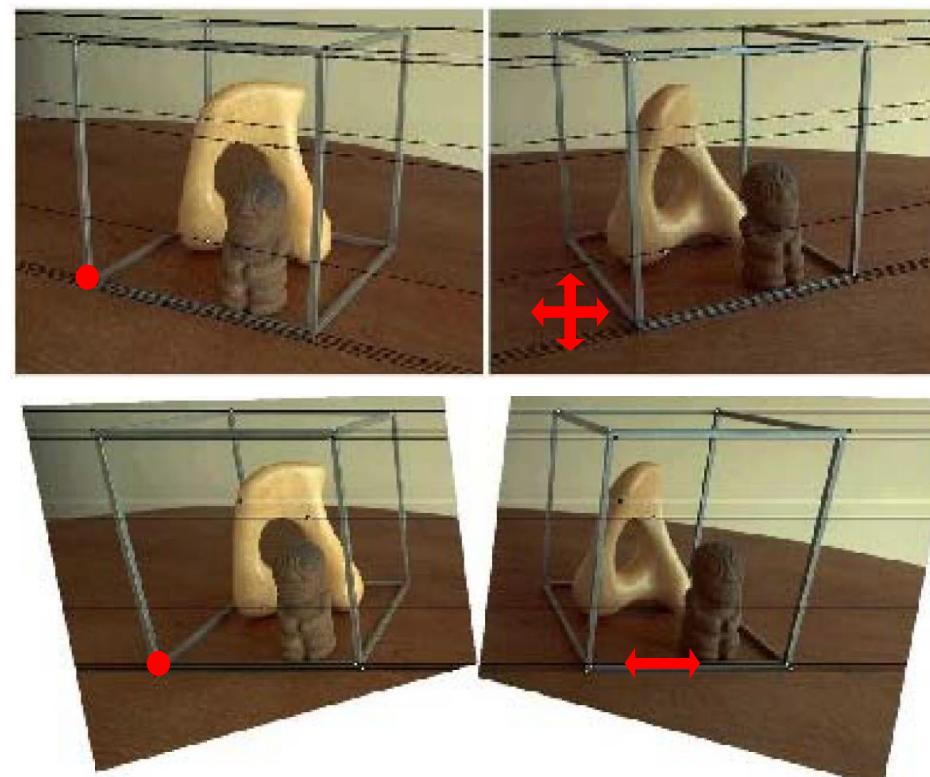
$$\text{disparity} = x - x' = \frac{B \cdot f}{z} \quad z = \frac{B \cdot f}{x - x'}$$

# Depth from Stereo Images

- Epipolar geometry: stereo calibration
  - Not only the intrinsic parameters
  - But also the extrinsics, R and t



HZ2003



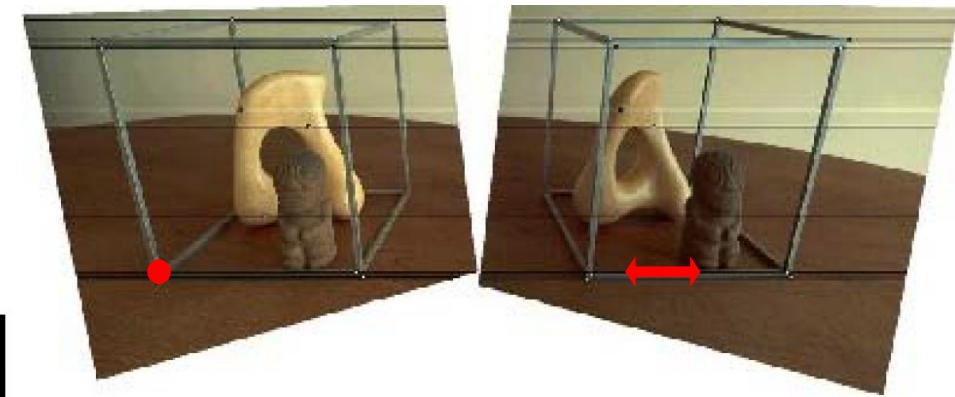
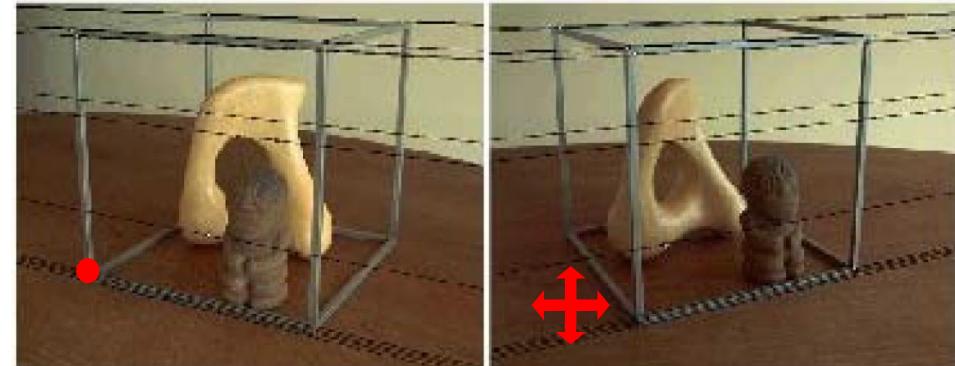
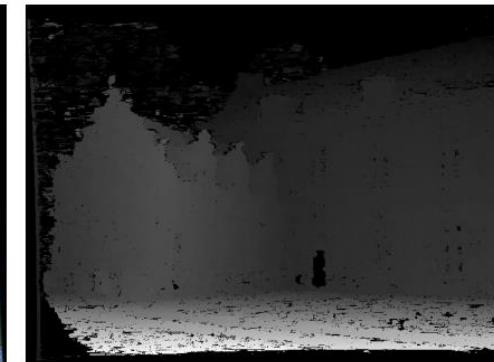
stereo image rectification

[https://en.wikipedia.org/wiki/Image\\_rectification](https://en.wikipedia.org/wiki/Image_rectification)



# Depth from Stereo Images

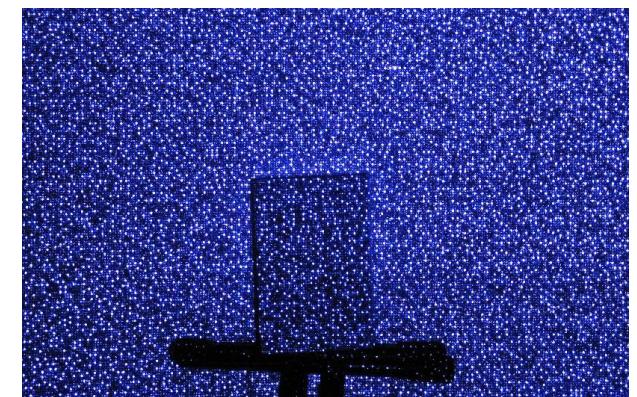
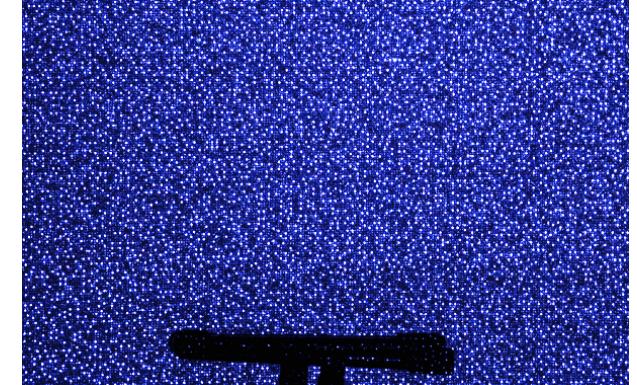
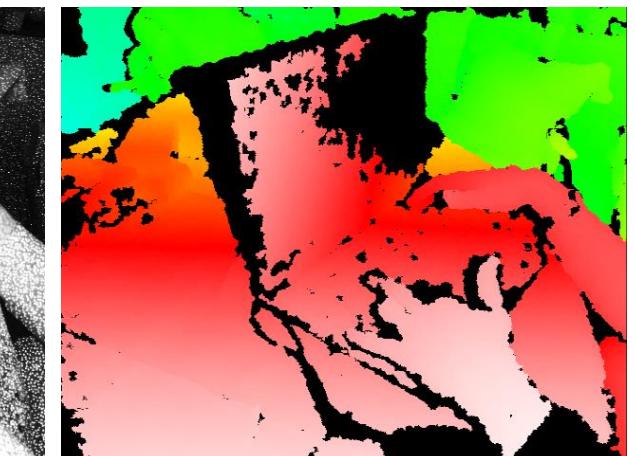
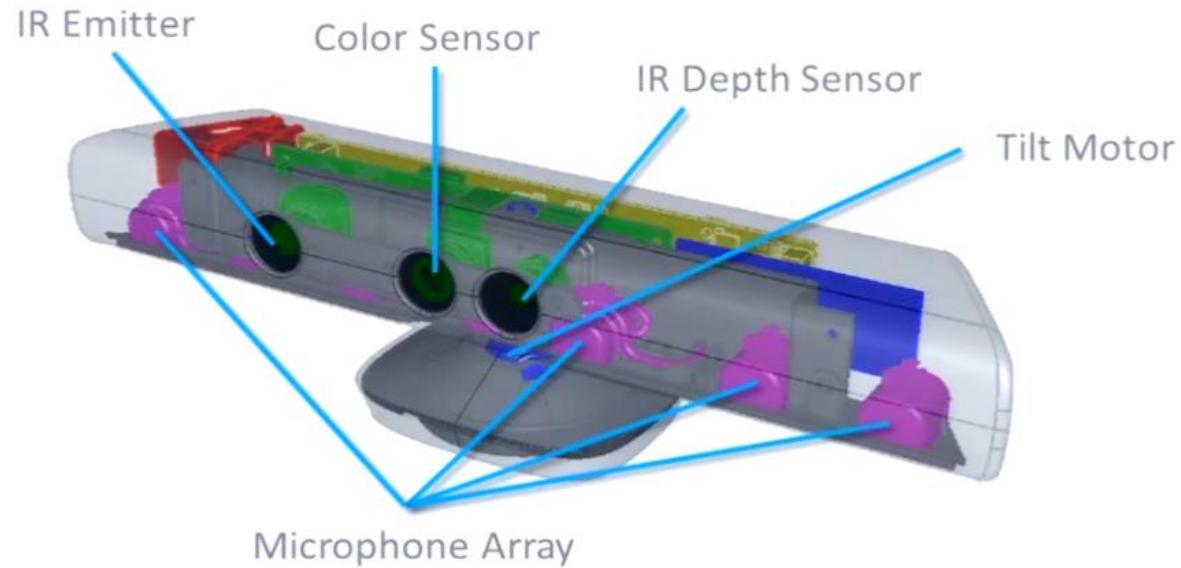
- Epipolar geometry: stereo calibration
- Finding disparity by searching along epipolar lines
  - Low texture: bad
  - Repeated texture: bad





# Depth from Structured Light

- projector + camera
  - Microsoft Kinect
  - Texture in the environment: generally not important
  - But cannot be used under direct sunlight





# Other RGBD Sensor Alternatives

---

- Intel RealSense
- Asus Xtion Pro
- Microsoft Kinect V2
- Structure Sensor
- If you do not have RGBD sensor
  - <http://www.michaelfirman.co.uk/RGBDdatasets/>





# Monocular Depth Estimation

---

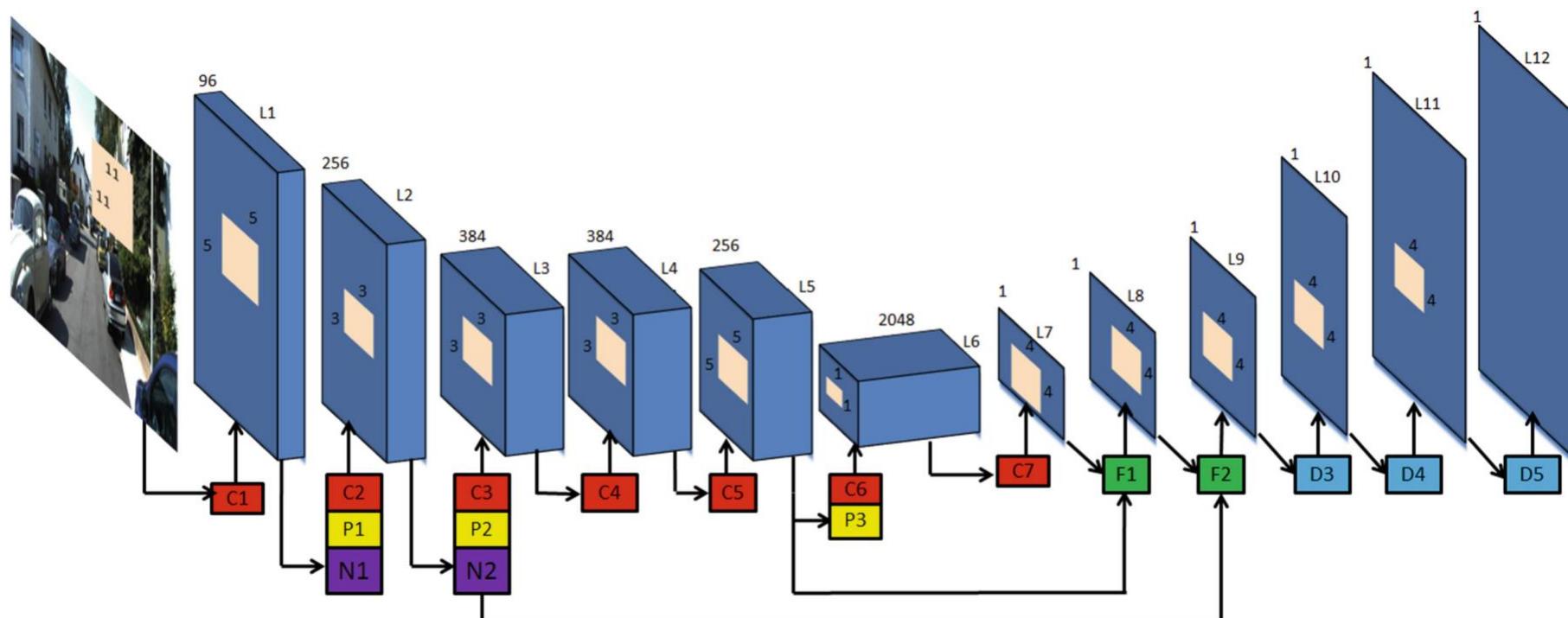
- Can we estimate depth from only a single image?





# Monocular Depth Estimation

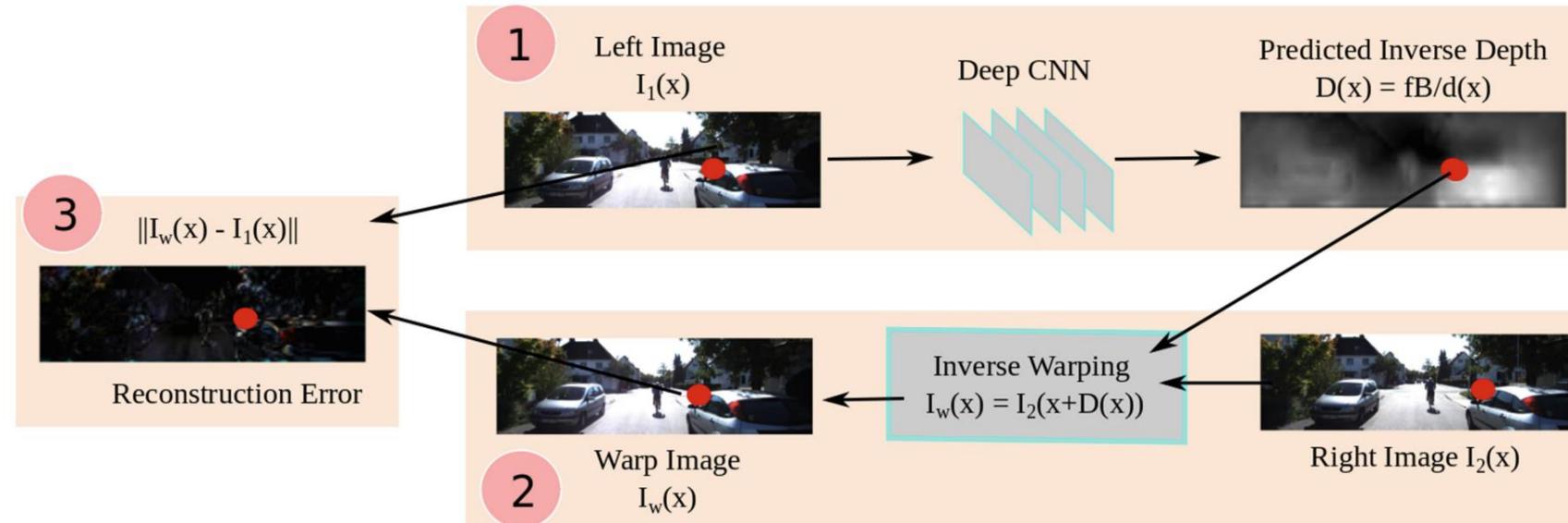
- Estimating depth from a single-view image **based on experience**
  - We could use supervised learning with regression loss (L2/MSE or L1 loss)
  - Need a large-scale dataset of paired RGB and *ground truth* Depth images





# Monocular Depth Estimation

- Estimating depth from a single-view image **based on experience**
  - A better way is to follow a self-supervised/unsupervised paradigm
  - Easily train monocular depth estimation from stereo image pairs

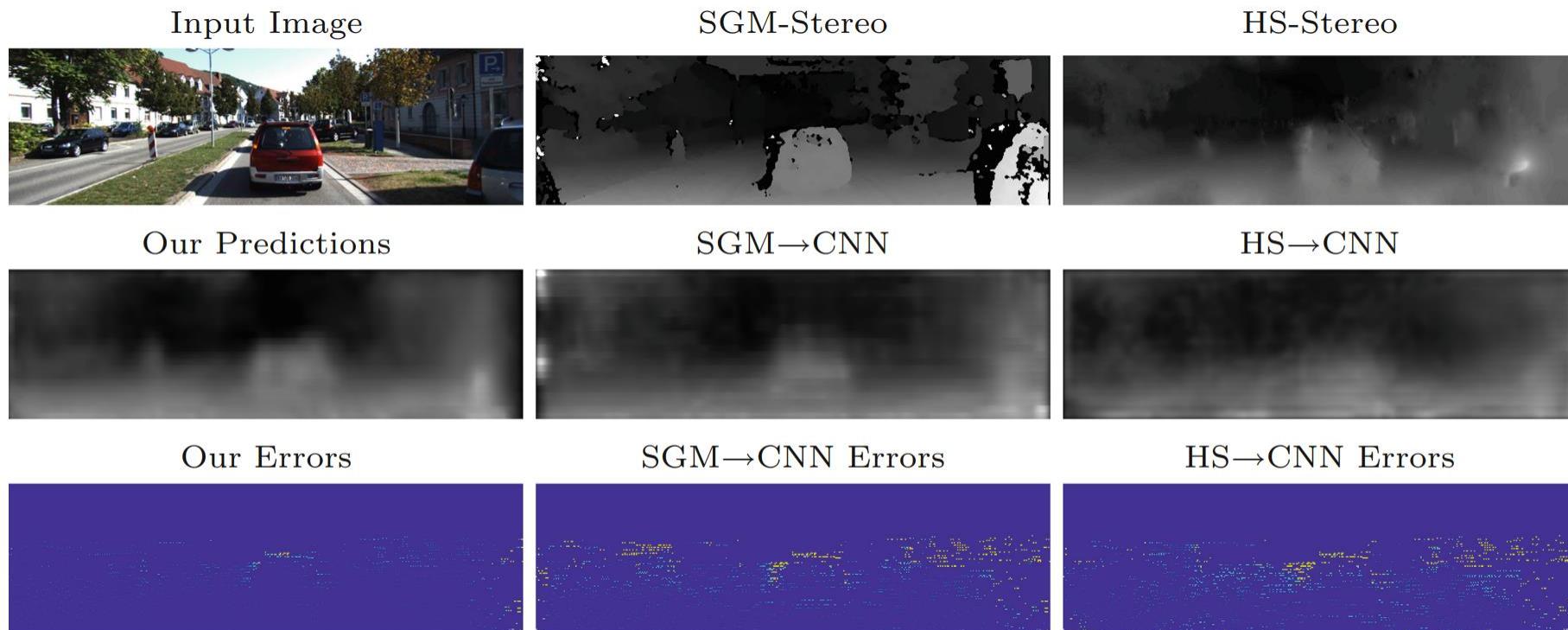


$$E_{recons}^i = \int_{\Omega} \|I_w^i(x) - I_1^i(x)\|^2 dx = \int_{\Omega} \|I_2^i(x + \underbrace{D^i(x)}_{fB/d^i(x)}) - I_1^i(x)\|^2 dx$$

$$E_{smooth}^i = \|\nabla D^i(x)\|^2$$

# Monocular Depth Estimation

- Estimating depth from a single-view image **based on experience**
  - A better way is to follow a self-supervised/unsupervised paradigm
  - Easily train monocular depth estimation from stereo image pairs

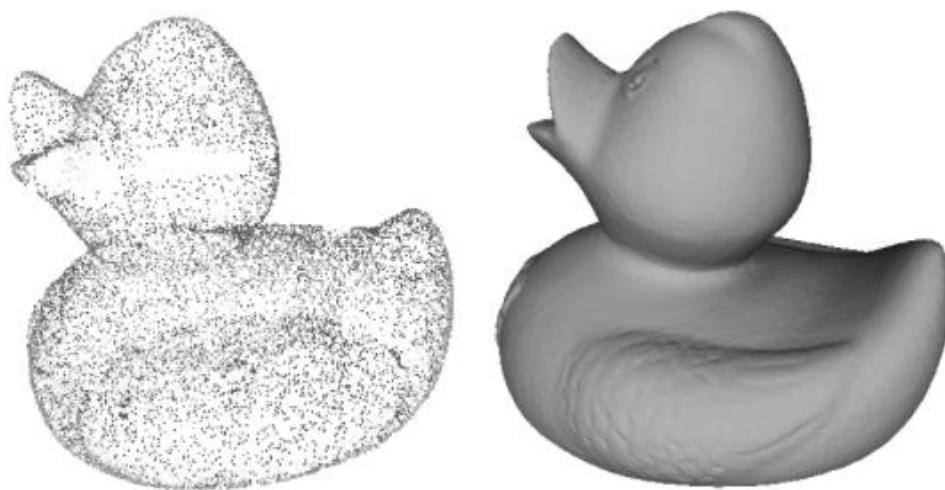




# 3D Data Representations

## Point Cloud/Mesh

- ✓ Raw format/Efficiency
- Explicit representation
- ✗ Unorganized/Unordered



[https://elmoatazbill.users.greyc.fr/  
point\\_cloud/reconstruction.png](https://elmoatazbill.users.greyc.fr/point_cloud/reconstruction.png)

## Voxel

- Implicit representation
- ✗ Resolution/Scalability
- ✗ Discretization artifact



[https://www.planetminecraft.com/  
project/giant-snowman-1638162/](https://www.planetminecraft.com/project/giant-snowman-1638162/)

## Primitives

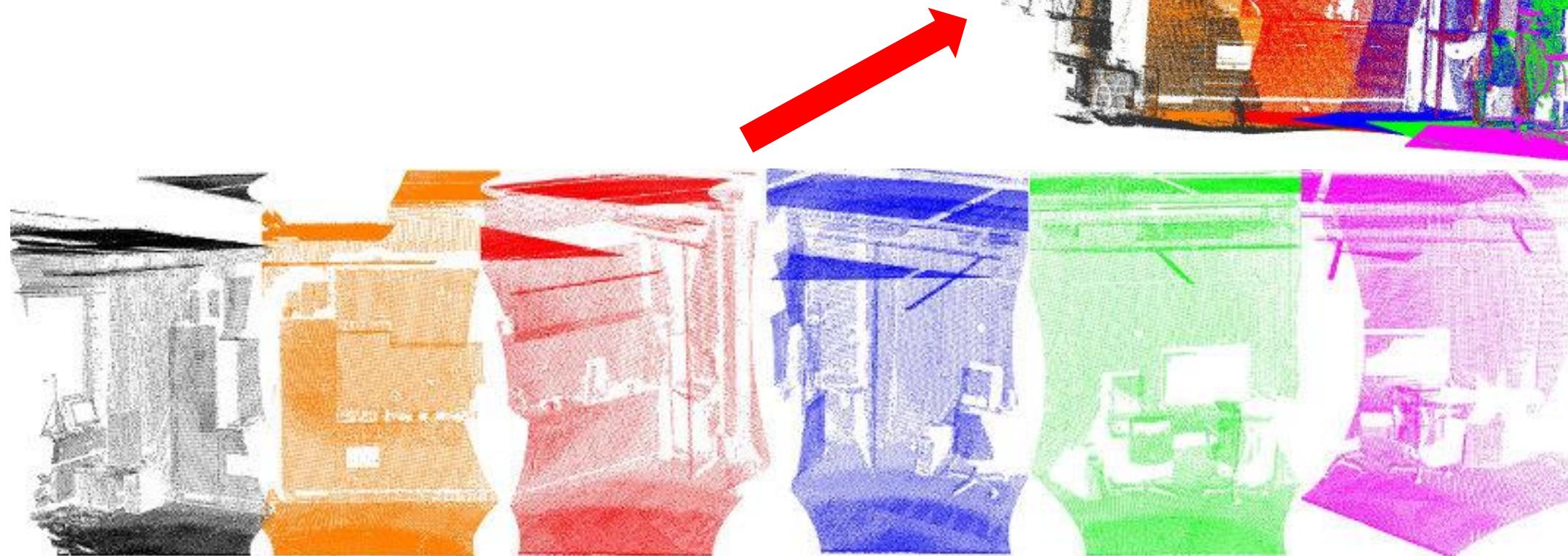
- ✓ Compact
- ✓ Ready for interaction
- Complex shapes?



<http://pointclouds.org/gsoc/>



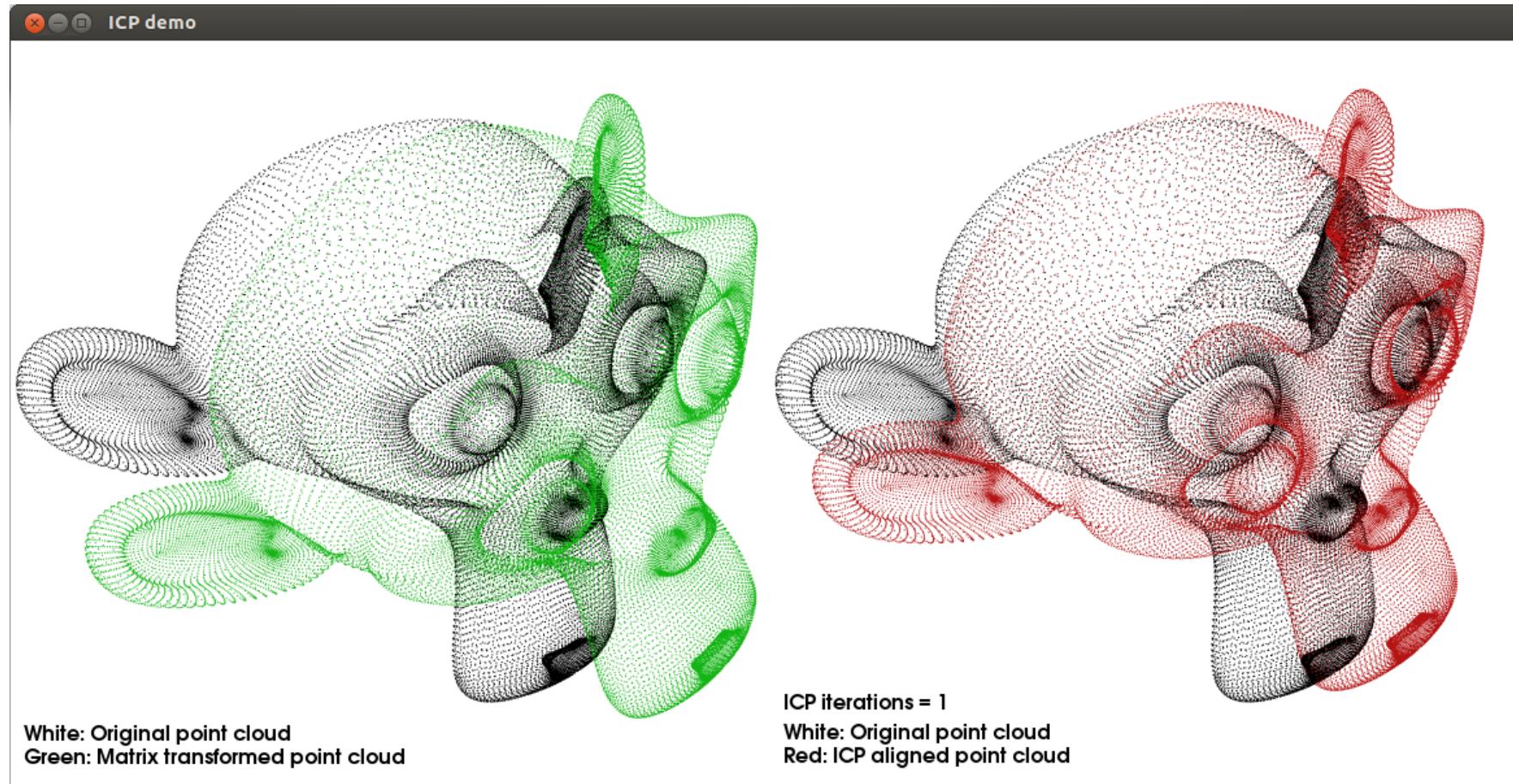
# Scan Registration



[http://pointclouds.org/documentation/tutorials/registration\\_api.php](http://pointclouds.org/documentation/tutorials/registration_api.php)



# Iterative Closest Point (ICP)

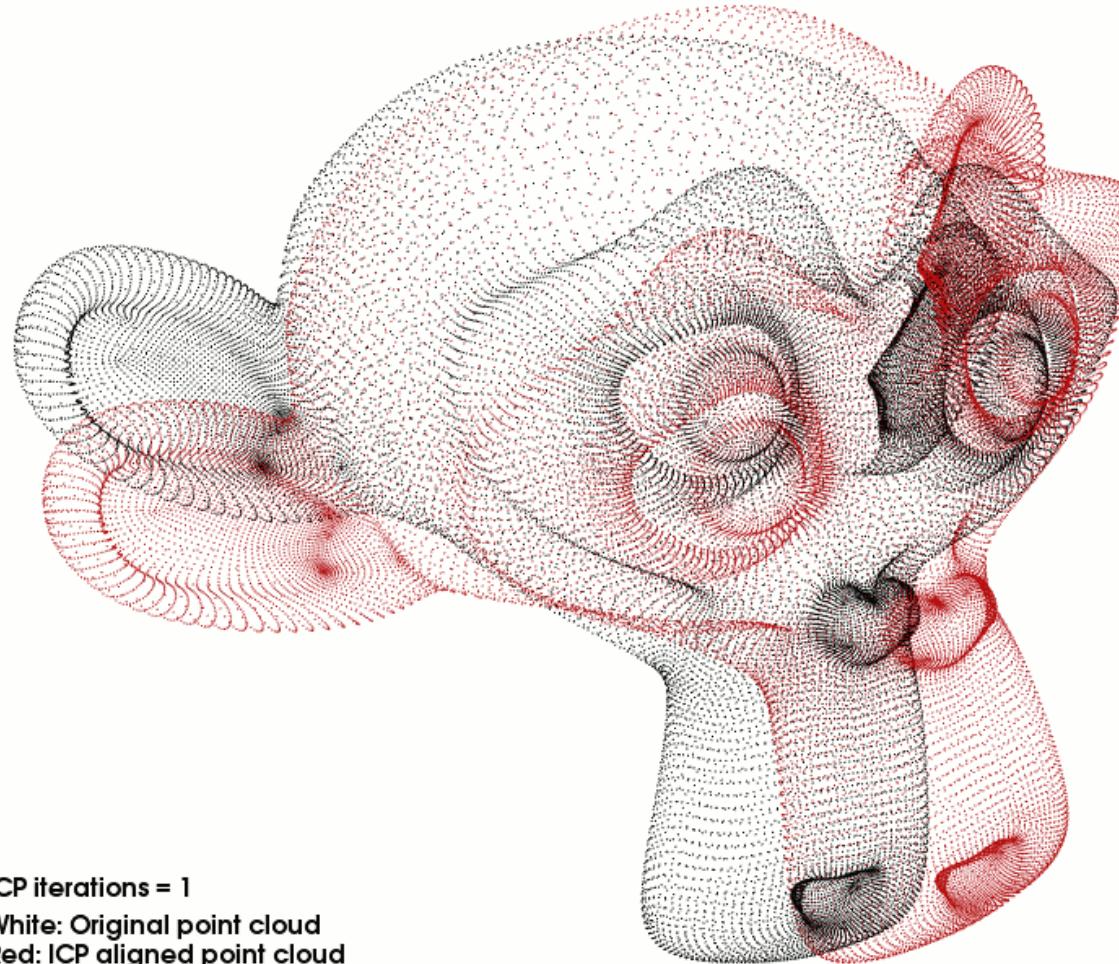


[http://pointclouds.org/documentation/tutorials/interactive\\_icp.php#interactive-icp](http://pointclouds.org/documentation/tutorials/interactive_icp.php#interactive-icp)



# ICP Process

---



ICP iterations = 1

White: Original point cloud

Red: ICP aligned point cloud

[http://pointclouds.org/documentation/tutorials/interactive\\_icp.php#interactive-icp](http://pointclouds.org/documentation/tutorials/interactive_icp.php#interactive-icp)



## Core Problem Definition

- Given two sets of **corresponding** points
  - Source:  $S = \{S_1, S_2, \dots, S_n\}$
  - Destination D =  $\{D_1, D_2, \dots, D_n\}$
- Find the translation and rotation that optimize the cost function
  - $C(R, t) = \sum_i \|D_i - RS_i - t\|^2$
- A fancy name of the problem: procrustes analysis
  - We've seen a simpler version of this
  - Hand-eye calibration
  - Vanishing-point-based camera orientation estimation



## Solve Translation First

---

- Idea: assume the optimal rotation has been found
- Set partial derivative w.r.t. translation to zero, solve the optimal translation
- Result
  - $t^* = (\sum_i D_i - R \sum_i S_i)/n$



# Solve Rotation



- Idea: substitute the optimal translation back into the cost function
- $C(R, t) = \sum_i \|D_i - RS_i - (\sum_i D_i - R \sum_i S_i)/n\|^2$
- Define new sets of corresponding points
  - Source:  $s = \{s_i\} = \{S_i - \sum_i S_i / n\}$
  - Destination d =  $\{d_i\} = \{D_i - \sum_i D_i / n\}$
  - A common term of this process: **demean, i.e. move data center to origin**
- $C(R, t) = \sum_i \|d_i - Rs_i\|^2$
- Solve this by the Orthogonal Procrustes Problem
  - Polar decomposition on the 3x3 covariance matrix  $\sum_i d_i s_i^T$  by SVD



# Wait, but I Do Not Know Correspondences!

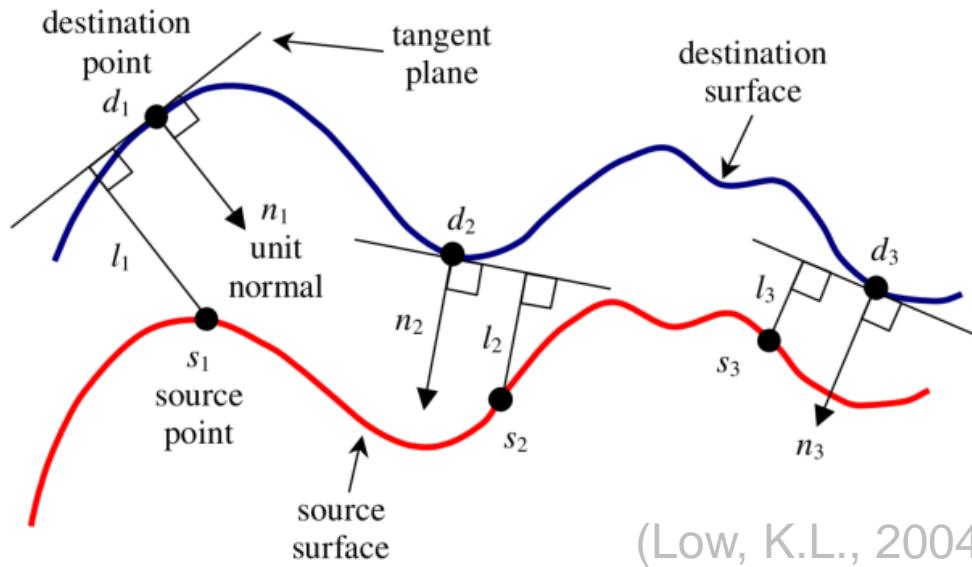
---

- Approximate correspondences
    - Simply pick the closest point
  - Then iterate until converge
- 
- ICP
    - Find correspondences of **S** in **D** by searching closest points (Brute-Force, KdTree)
    - Solve the current optimal translation and rotation by procrustes analysis
    - If cost decreased, apply the current solution back to **S**
    - If cost stopped changing or smaller than a threshold, stop.
    - Otherwise repeat the above steps.



## ICP variants - Change Cost Function

- Point-to-Point:  $C(R, t) = \sum_i \|D_i - RS_i - t\|^2$
- Point-to-Plan:  $C(R, t) = \sum_i \|N_i^T(D_i - RS_i - t)\|^2$ ,  $N_i$  being  $D_i$ 's unit normal
  - Often significantly better convergence rate than point-to-point ICP
  - Linearization under small rotation assumption



(Low, K.L., 2004)

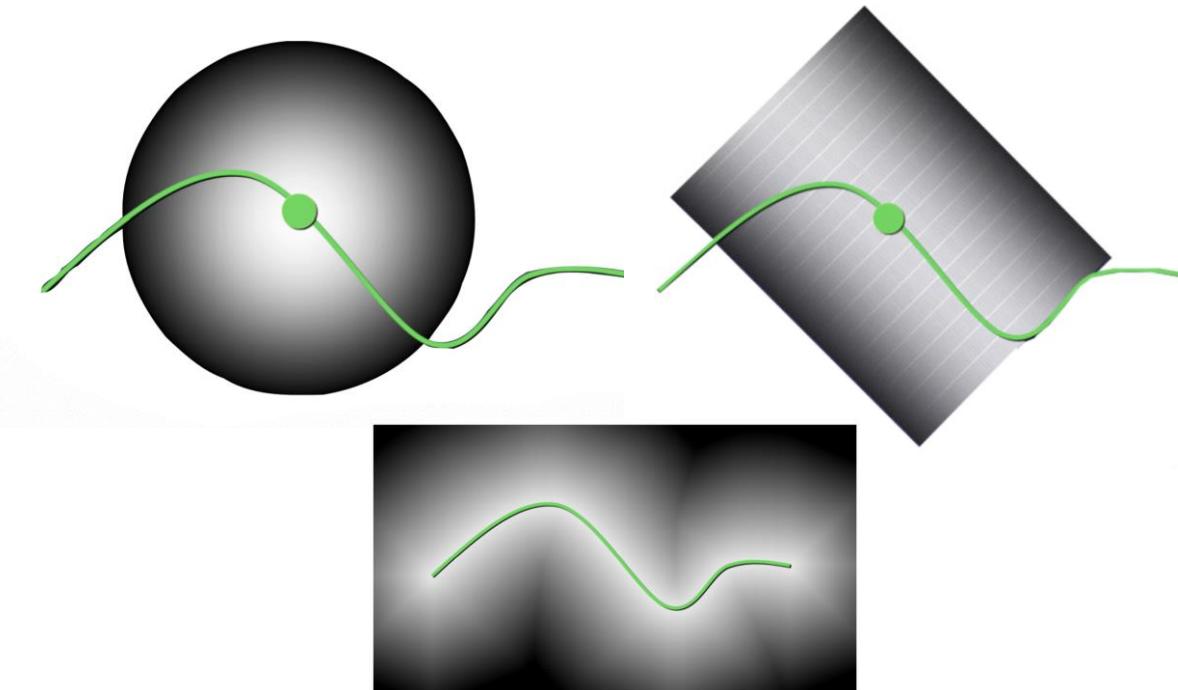
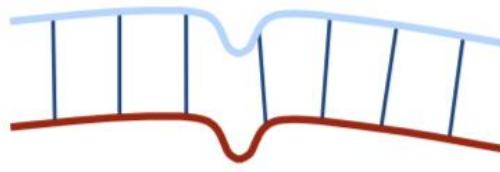


Image from Hao Li.

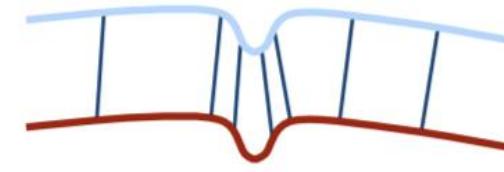


# ICP variants – Sampling Source Points

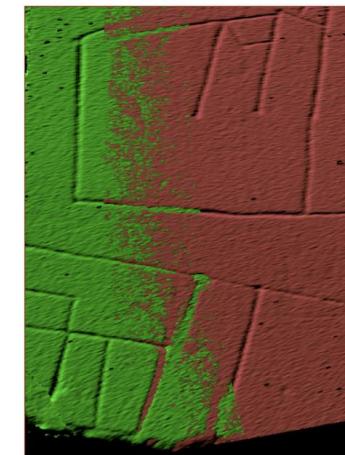
- Normal-space sampling is better for mostly smooth areas with sparse features



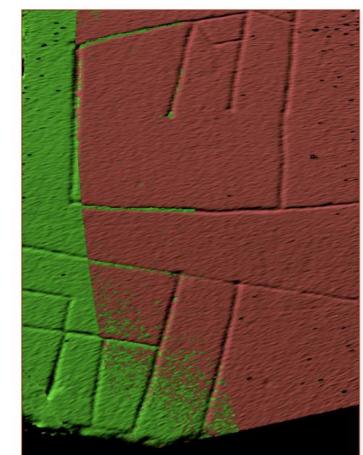
uniform sampling



normal-space sampling



Random sampling



Normal-space sampling

(Rusinkiewicz et al., 2001)

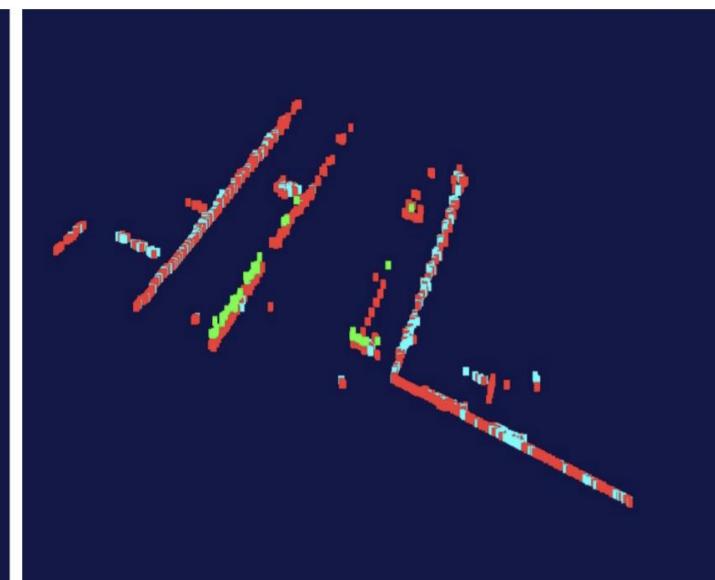


# ICP variants – Sampling Source Points

- Feature-based Sampling
  - Find 3D keypoints
  - Often with higher efficiency and higher accuracy
  - May require RANSAC to remove outliers



3D Scan (~200.000 Points)



Extracted Features (~5.000 Points)



# ICP variants - Change Data Association

- Beyond closest point search

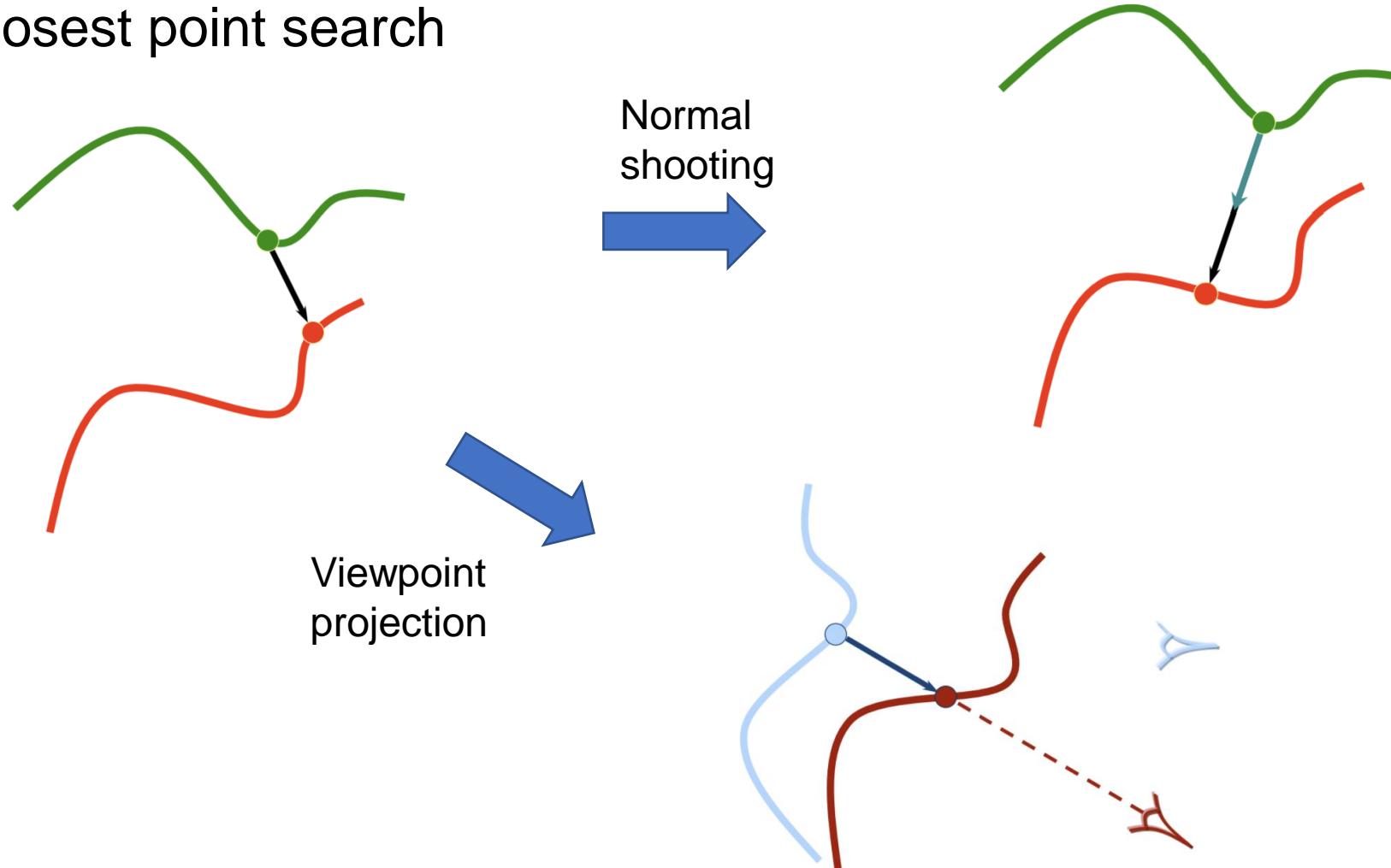


Image from Burgard, et al.



# ICP for Degenerated Cases

- Solution: use color-based ICP

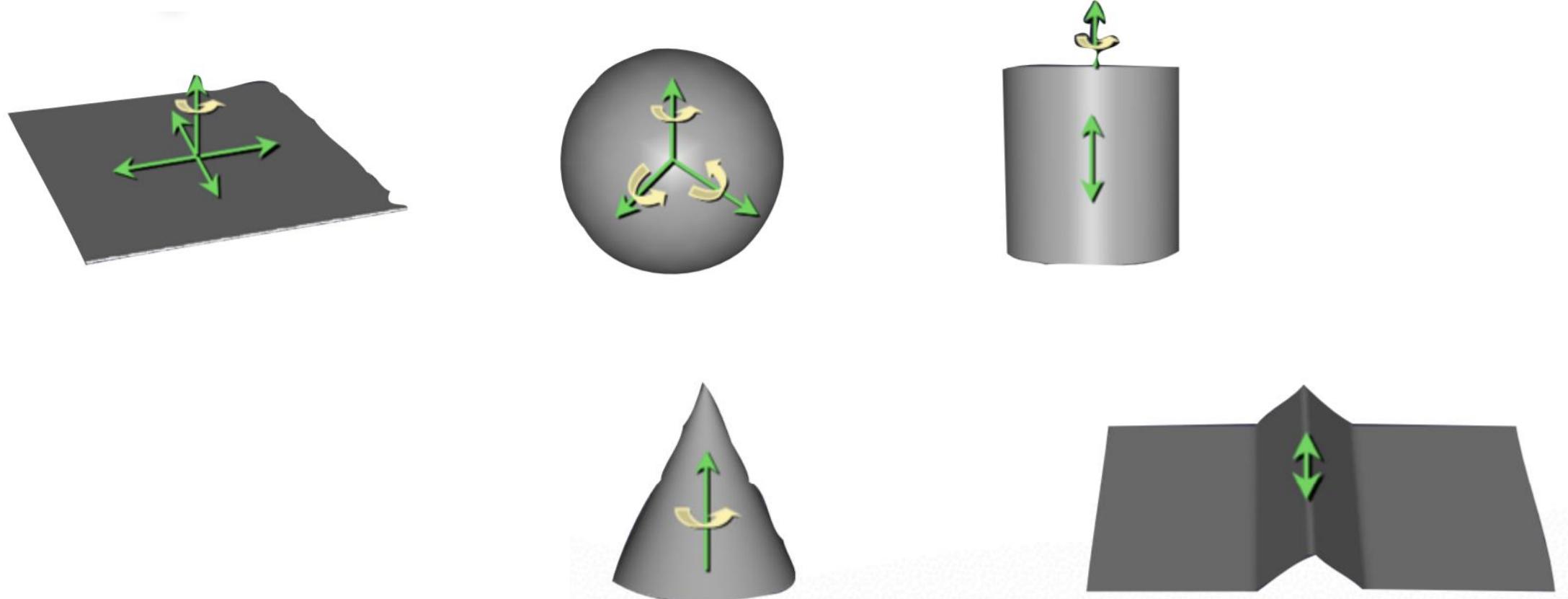


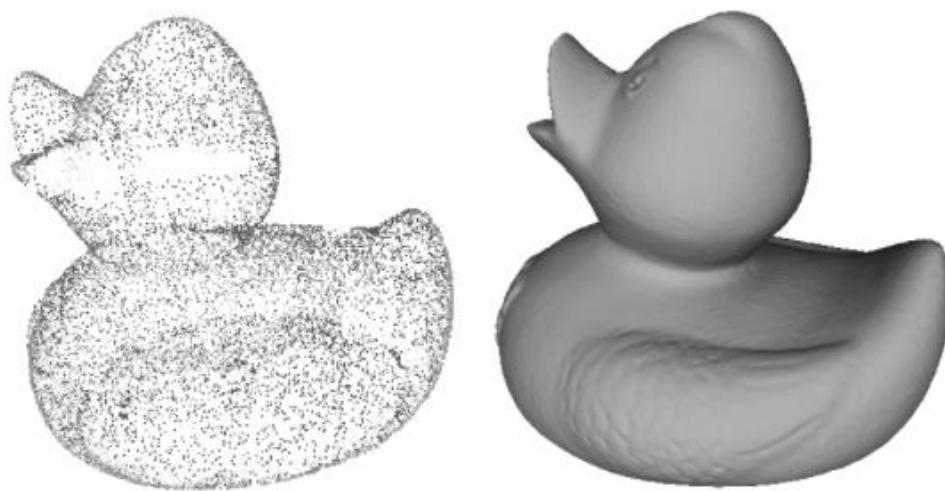
Image from Hao Li.



# 3D Data Representations

## Point Cloud/Mesh

- ✓ Raw format/Efficiency
- Explicit representation
- ✗ Unorganized/Unordered



[https://elmoatazbill.users.greyc.fr/  
point\\_cloud/reconstruction.png](https://elmoatazbill.users.greyc.fr/point_cloud/reconstruction.png)

## Voxel

- Implicit representation
- ✗ Resolution/Scalability
- ✗ Discretization artifact



[https://www.planetminecraft.com/  
project/giant-snowman-1638162/](https://www.planetminecraft.com/project/giant-snowman-1638162/)

## Primitives

- ✓ Compact
- ✓ Ready for interaction
- Complex shapes?

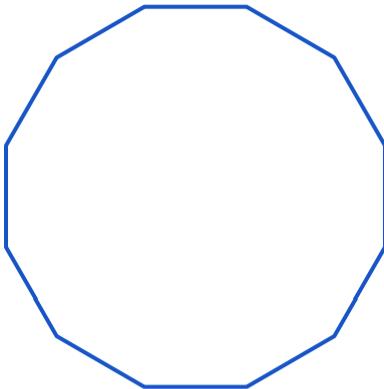


<http://pointclouds.org/gsoc/>



# Explicit Surface Representation

- Geometry/Shape is stored explicitly as a list of points, triangles, or other geometric fragments
  - Point clouds, mesh, ...



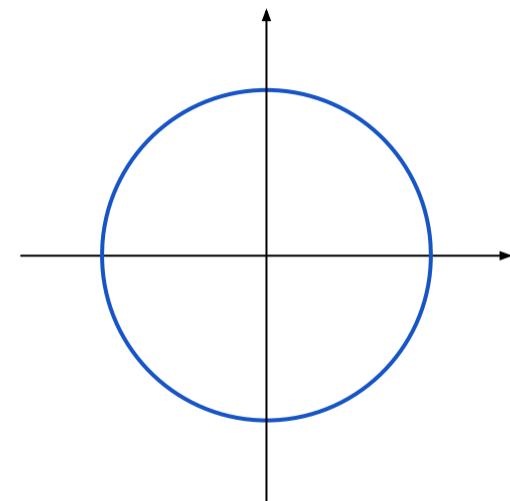
Vertices: [  $(x_0, y_0, z_0), (x_1, y_1, z_1), \dots, (x_n, y_n, z_n)$  ]

Indices: [  $(i_0, i_1), (i_2, i_3), \dots, (i_{n-1}, i_n)$  ]



# Implicit Surface Representation

- Geometry/Shape is not stored explicitly but rather defined as a level set of a function defined over the space in which the geometry is embedded
  - Some are **parametric**

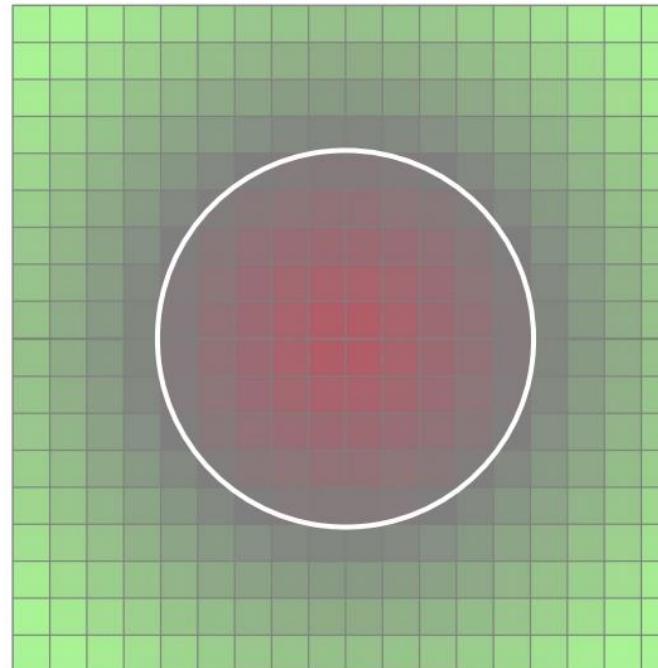


$$f(x, y) = x^2 + y^2 - r^2$$



# Implicit Surface Representation

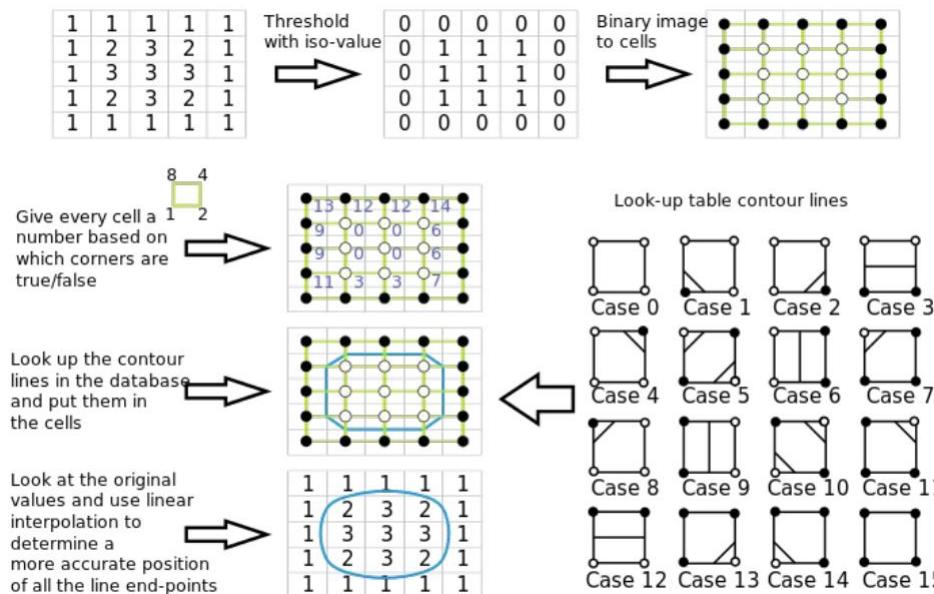
- Geometry/Shape is not stored explicitly but rather defined as a level set of a function defined over the space in which the geometry is embedded
  - Some are **non-parametric: Signed Distance Function/Field (SDF)**



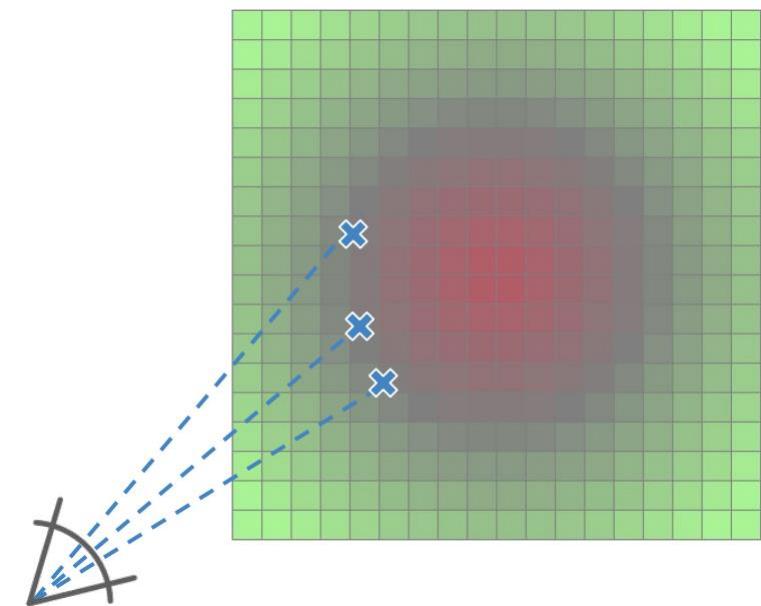


# Implicit-to-Explicit Conversions

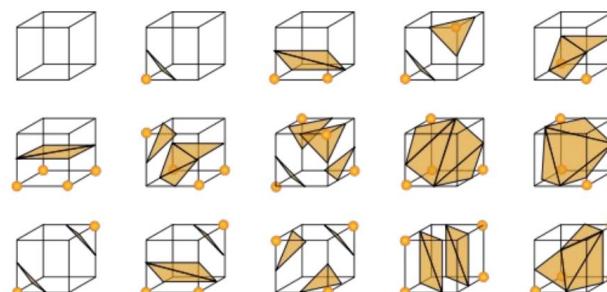
## Marching Squares (2D)



## Raycasting for partial conversion



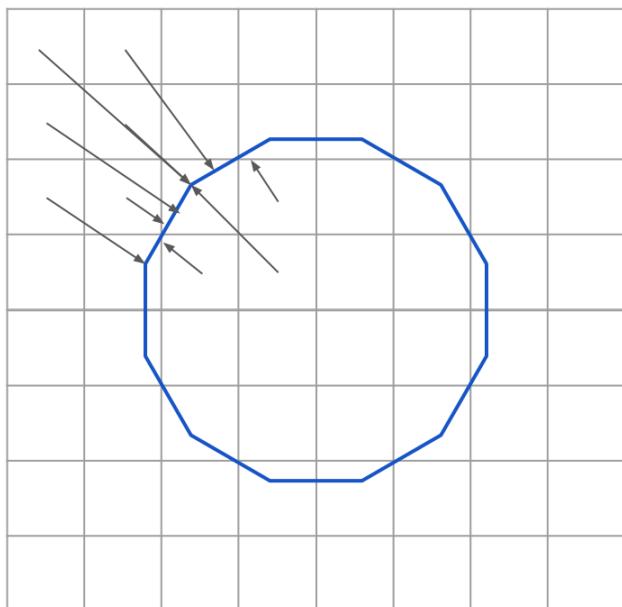
## Marching Cubes (3D)



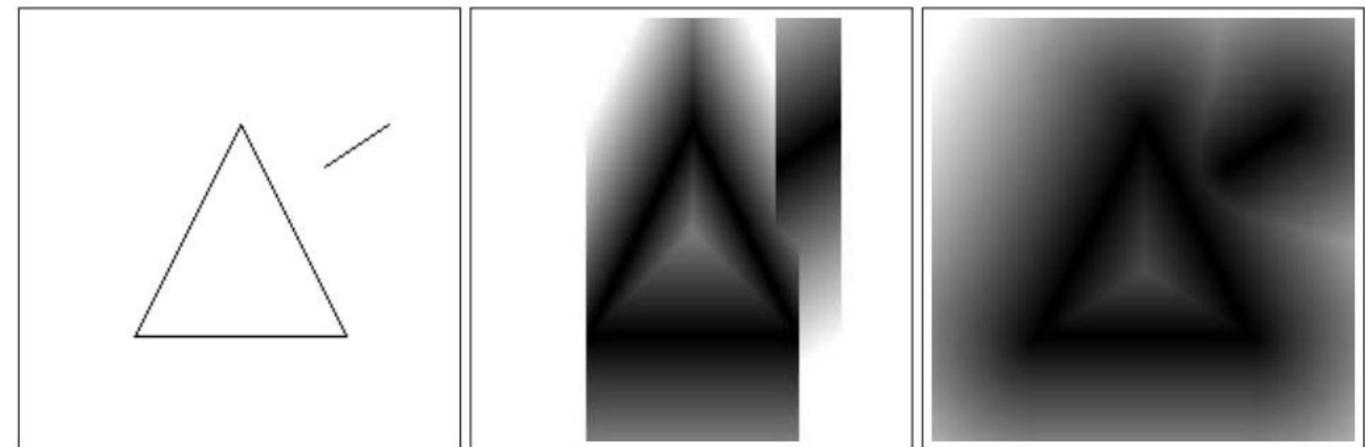


# Explicit-to-Implicit Conversions

By finding closest points to the surface



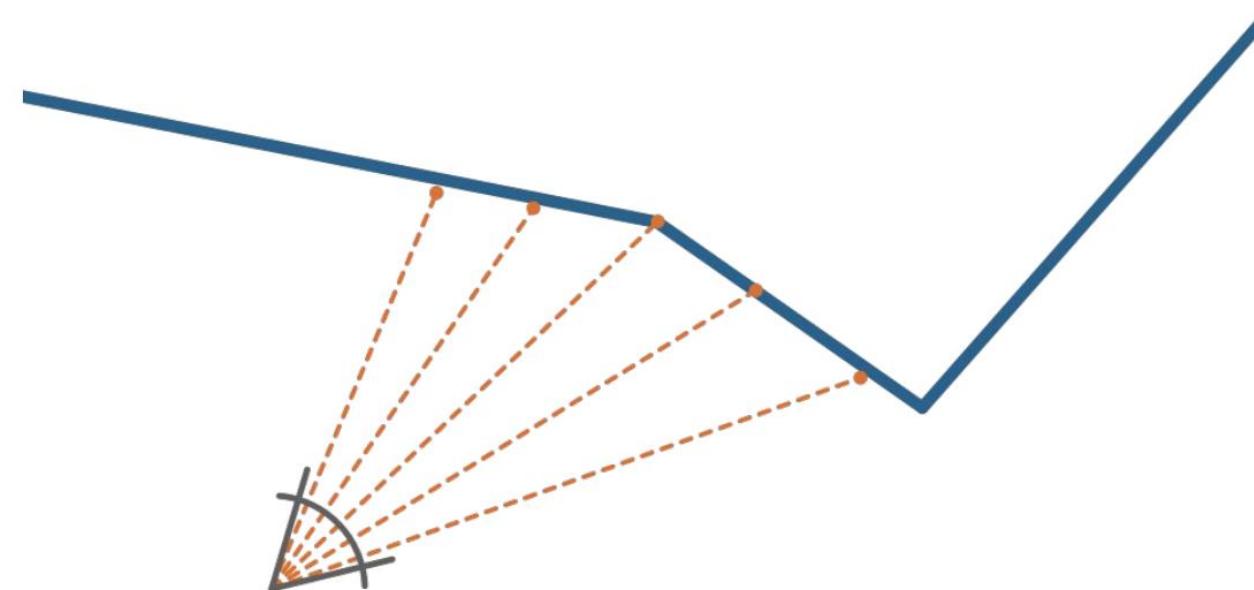
Distance Transform (2D)





# Projective SDF and Truncated SDF (TSDF)

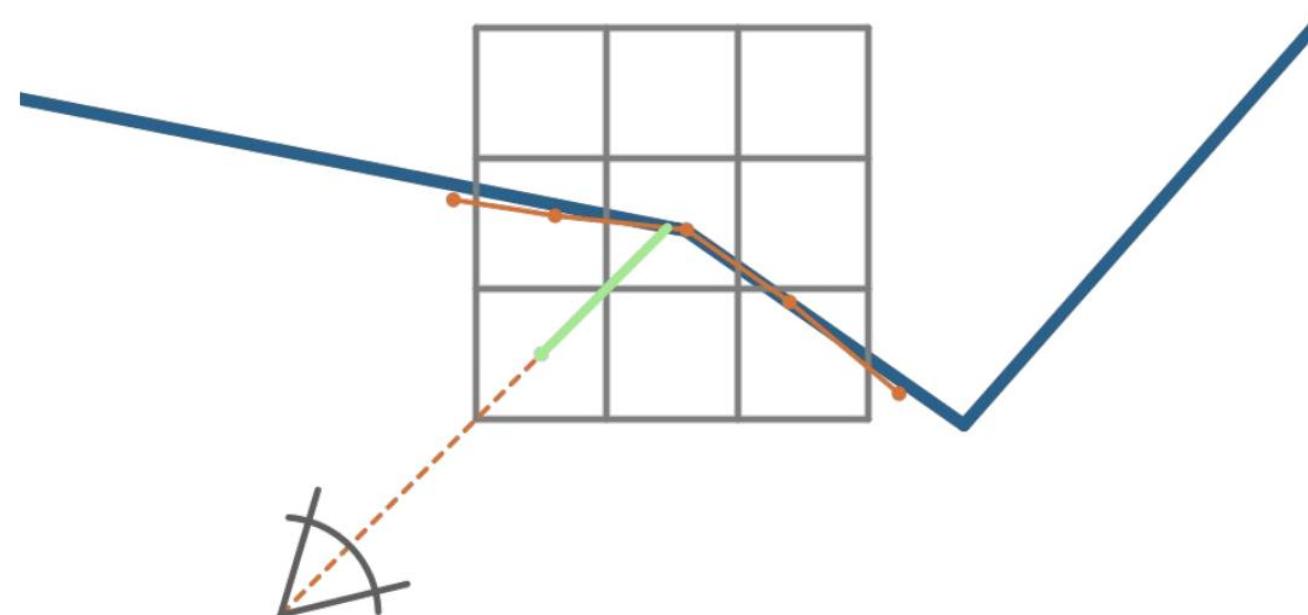
- Computing SDF requires a closed surface
- **What if I have only partial observation of the surface?**





# Projective SDF and Truncated SDF (TSDF)

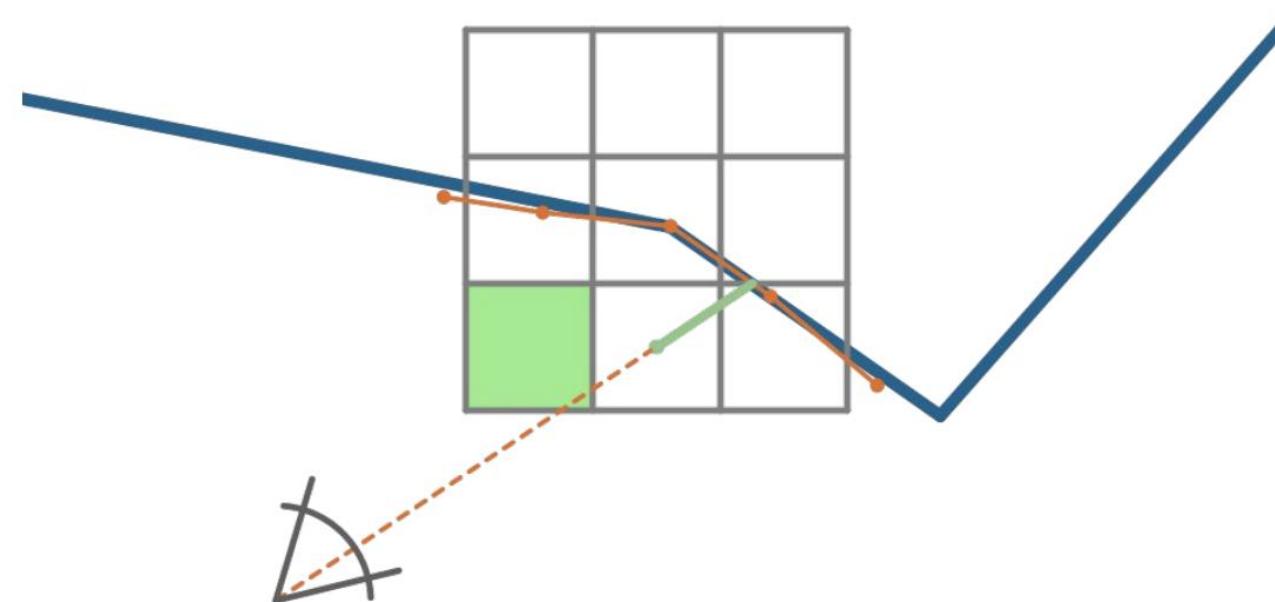
- What if I have only partial observation of the surface?
- We can define projective SDF:





# Projective SDF and Truncated SDF (TSDF)

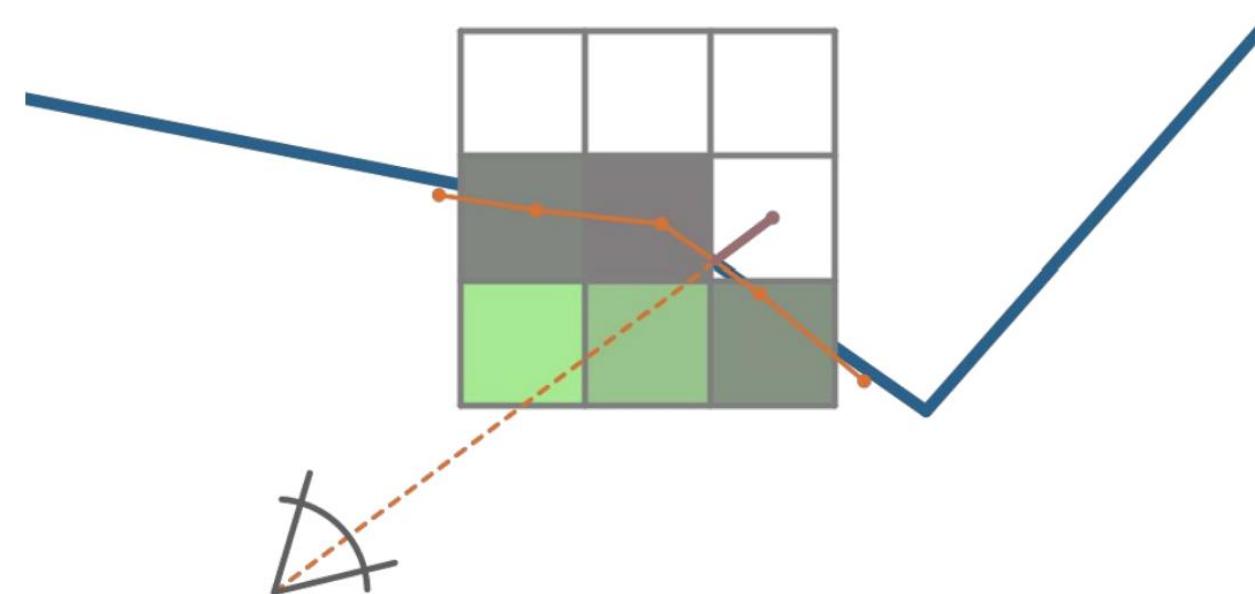
- What if I have only partial observation of the surface?
- We can define projective SDF:





# Projective SDF and Truncated SDF (TSDF)

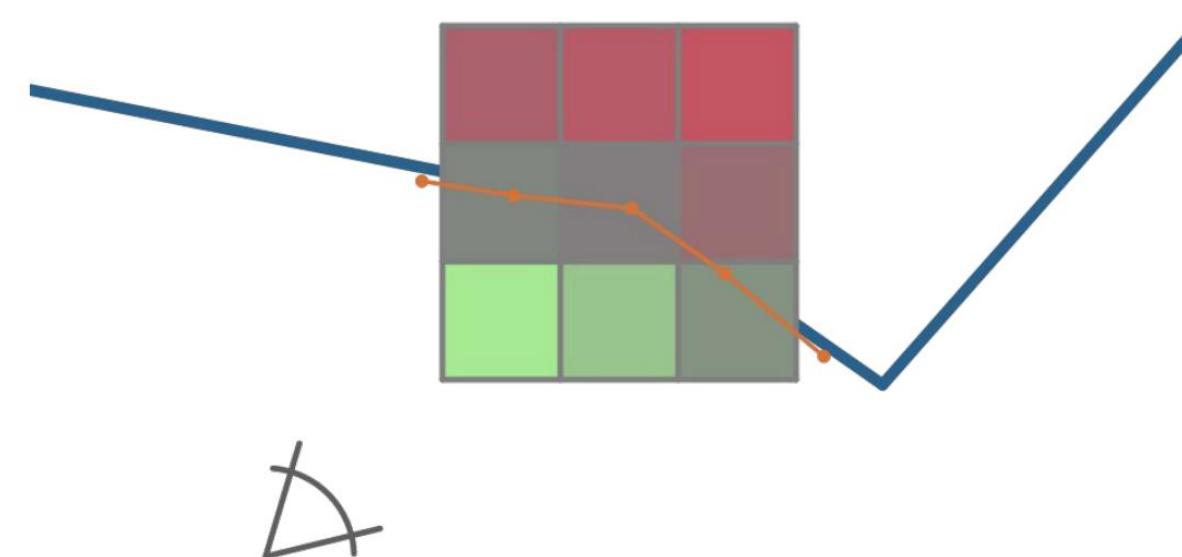
- What if I have only partial observation of the surface?
- We can define projective SDF:





# Projective SDF and Truncated SDF (TSDF)

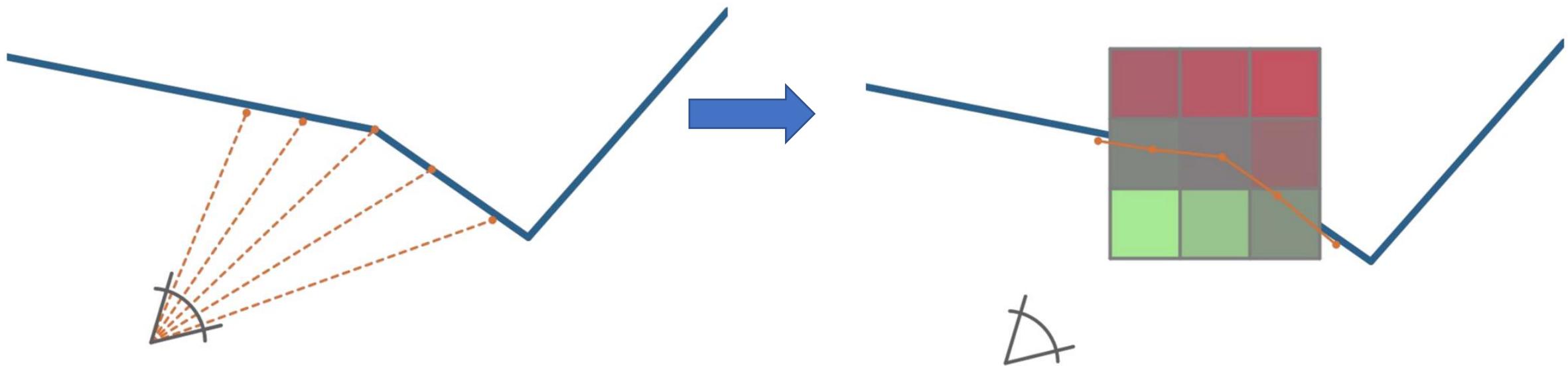
- What if I have only partial observation of the surface?
- Since we only compute the distance near both sides of the surface, we call this projective SDF as a **Truncated SDF**, or **TSDF**.





# Projective SDF and Truncated SDF (TSDF)

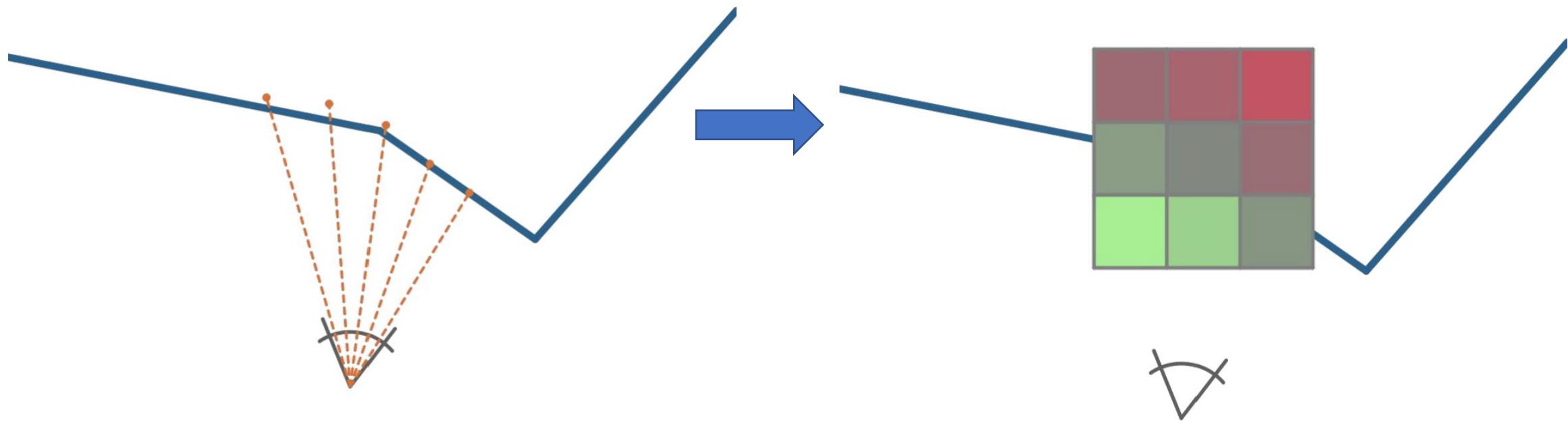
- TSDF is view dependent





# Projective SDF and Truncated SDF (TSDF)

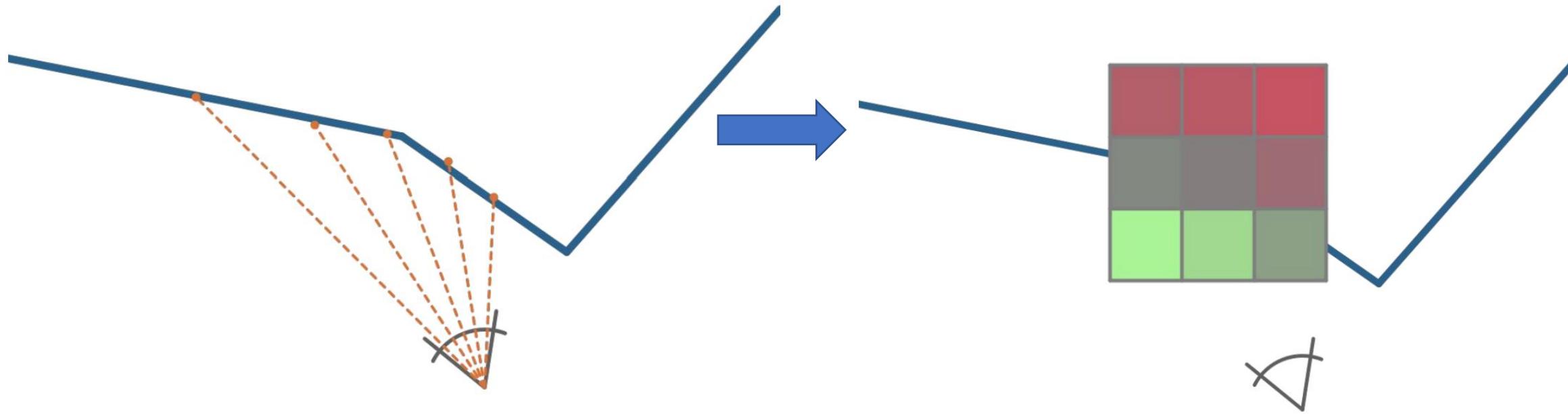
- TSDF is view dependent





# Projective SDF and Truncated SDF (TSDF)

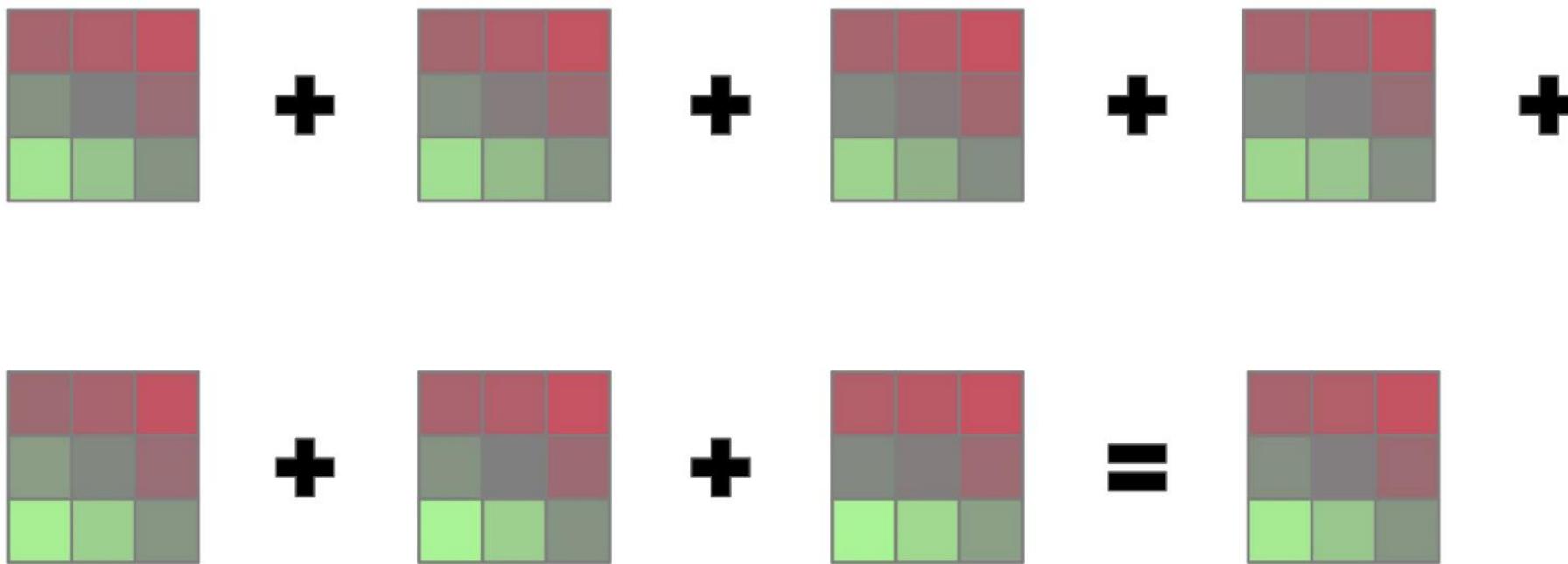
- TSDF is view dependent





# Projective SDF and Truncated SDF (TSDF)

- Fusing multiple TSDF gives a good approximation of the true SDF





# Scan Registration before Fusing Multiple TSDF

- Fusing TSDF requires a consistent coordinate frame





# Scan Registration using Point-Plane ICP

- Recall Point-Plane ICP

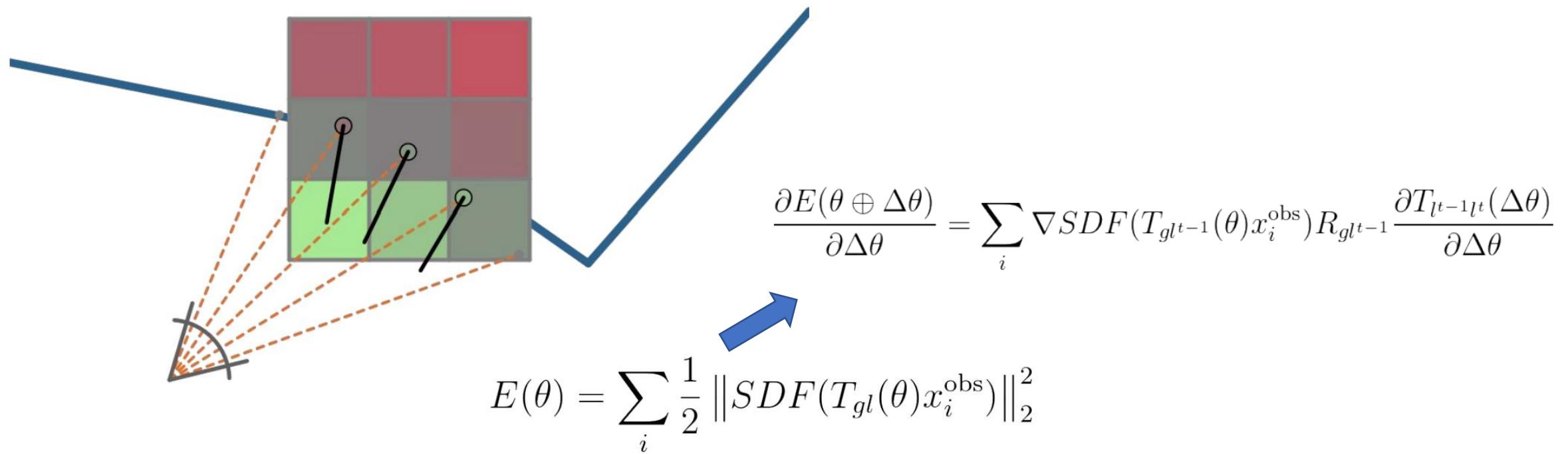
$$E(\theta) = \sum_i \frac{1}{2} \left\| n_i^{\text{pred}} \cdot (T_{gl}(\theta)x_i^{\text{obs}} - \underline{x_i^{\text{pred}}}) \right\|_2^2$$

$$\frac{\partial E(\theta \oplus \Delta\theta)}{\partial \Delta\theta} = \sum_i n_i^{\text{pred}} \cdot \frac{\partial T_{gl}(\theta \oplus \Delta\theta)}{\partial \Delta\theta} x_i^{\text{obs}}$$
$$\frac{\partial E(\theta)}{\partial \theta} = \sum_i n_i^{\text{pred}} \cdot \frac{\partial T_{gl}(\theta)}{\partial \theta} x_i^{\text{obs}}$$

$$\frac{\partial}{\partial \Delta\theta} T(\theta \oplus \Delta\theta) \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & z & -y \\ 0 & 1 & 0 & -z & 0 & x \\ 0 & 0 & 1 & y & -x & 0 \end{bmatrix}$$
$$T(\theta \oplus \Delta\theta) = e^{\Delta\theta} T(\theta)$$

# Scan Registration using TSDF Directly

- Tracking requires the fused SDF volume for all frames up to the current frame
- We must maintain a running average SDF value at each cell





# Scan Registration using TSDF Directly

- Tracking requires the fused SDF volume for all frames up to the current frame
- We must maintain a running average SDF value at each cell

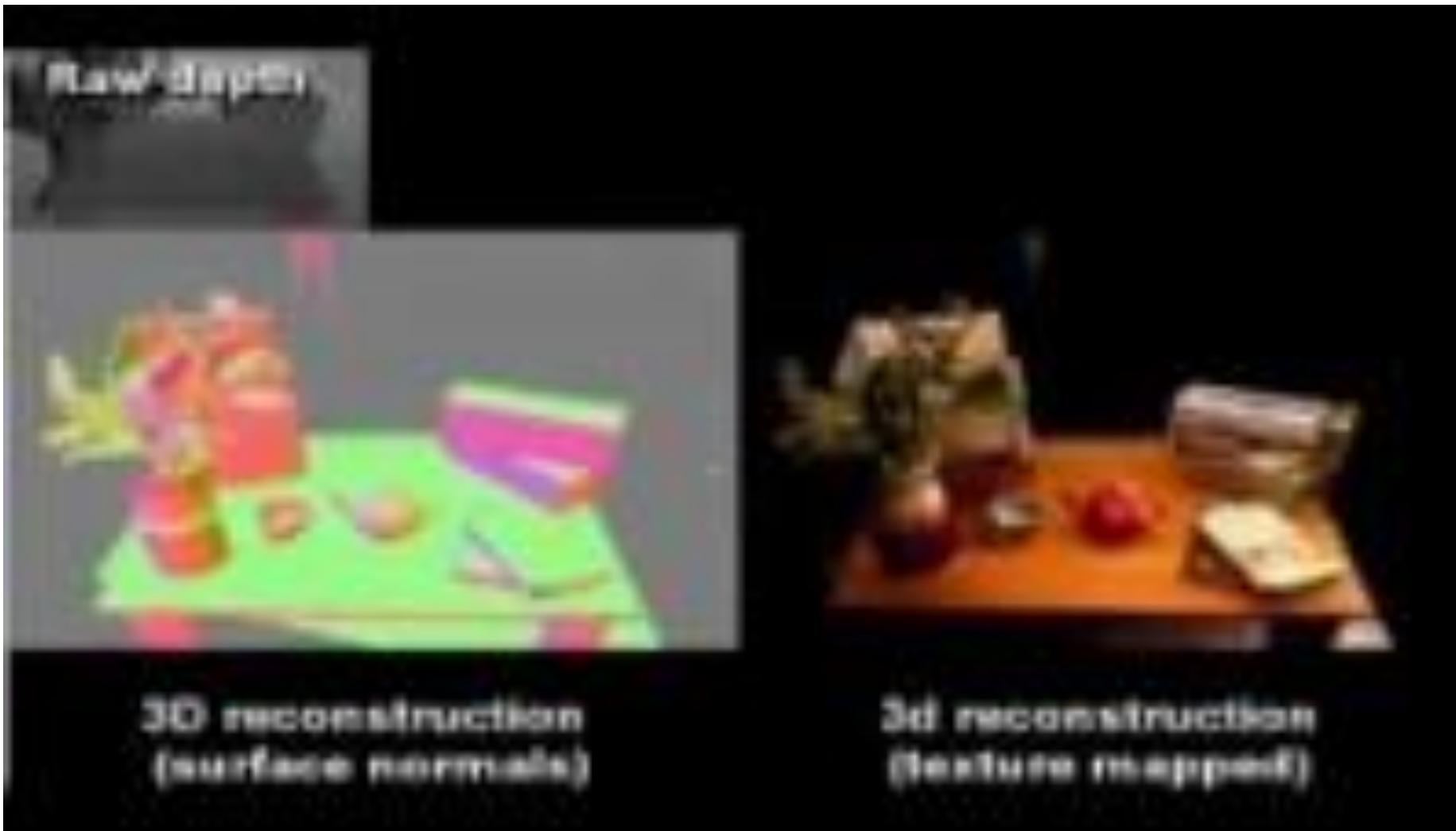
$$SDF^{0:t} \leftarrow \frac{SDF^{0:t-1}w^{0:t-1} + SDF^t w^t}{w^{0:t-1} + w^t}$$

$$w^{0:t} \leftarrow w^{0:t-1} + w^t$$





# Application: KinectFusion

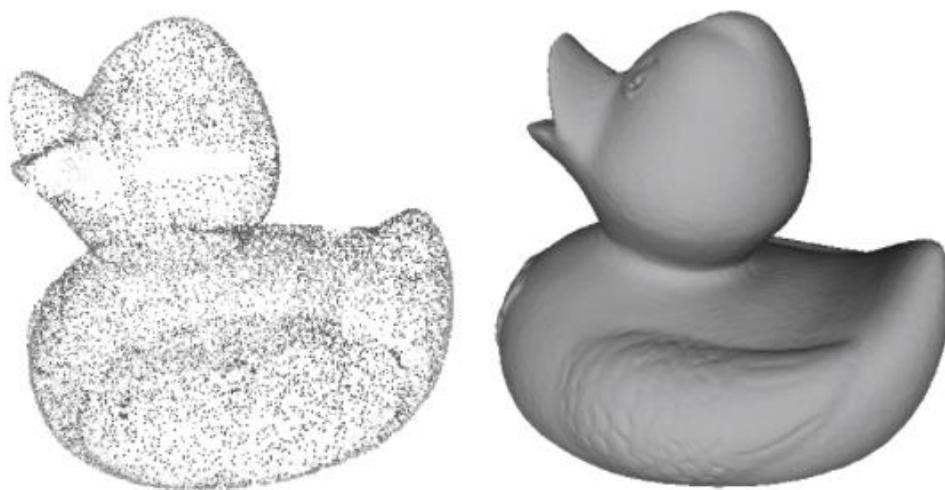




# 3D Data Representations

## Point Cloud/Mesh

- ✓ Raw format/Efficiency
- Explicit representation
- ✗ Unorganized/Unordered



[https://elmoatazbill.users.greyc.fr/  
point\\_cloud/reconstruction.png](https://elmoatazbill.users.greyc.fr/point_cloud/reconstruction.png)

## Voxel

- Implicit representation
- ✗ Resolution/Scalability
- ✗ Discretization artifact



[https://www.planetminecraft.com/  
project/giant-snowman-1638162/](https://www.planetminecraft.com/project/giant-snowman-1638162/)

## Primitives

- ✓ Compact
- ✓ Ready for interaction
- Complex shapes?



<http://pointclouds.org/gsoc/>

# Fast plane extraction

- Refer to the PEAC slides
  - Feng, C., Taguchi, Y. and Kamat, V.R., 2014. Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 6218-6225).



# Fast Plane Extraction in Organized Point Clouds Using Agglomerative Hierarchical Clustering

Chen Feng<sup>1)</sup>, Yuichi Taguchi<sup>2)</sup>, and Vineet R. Kamat<sup>1)</sup>

1) University of Michigan, USA

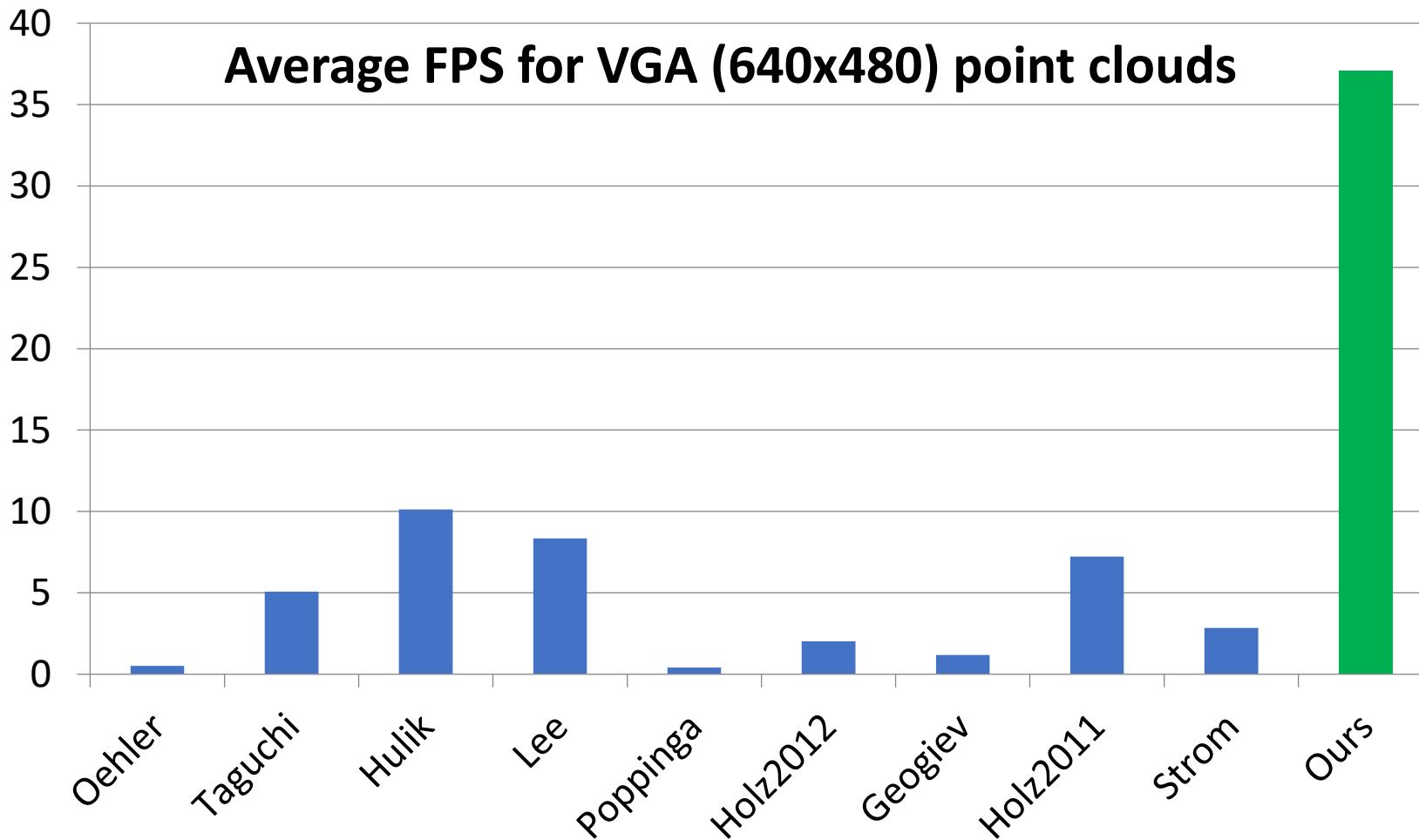
2) Mitsubishi Electric Research Labs, USA

June 4, 2014

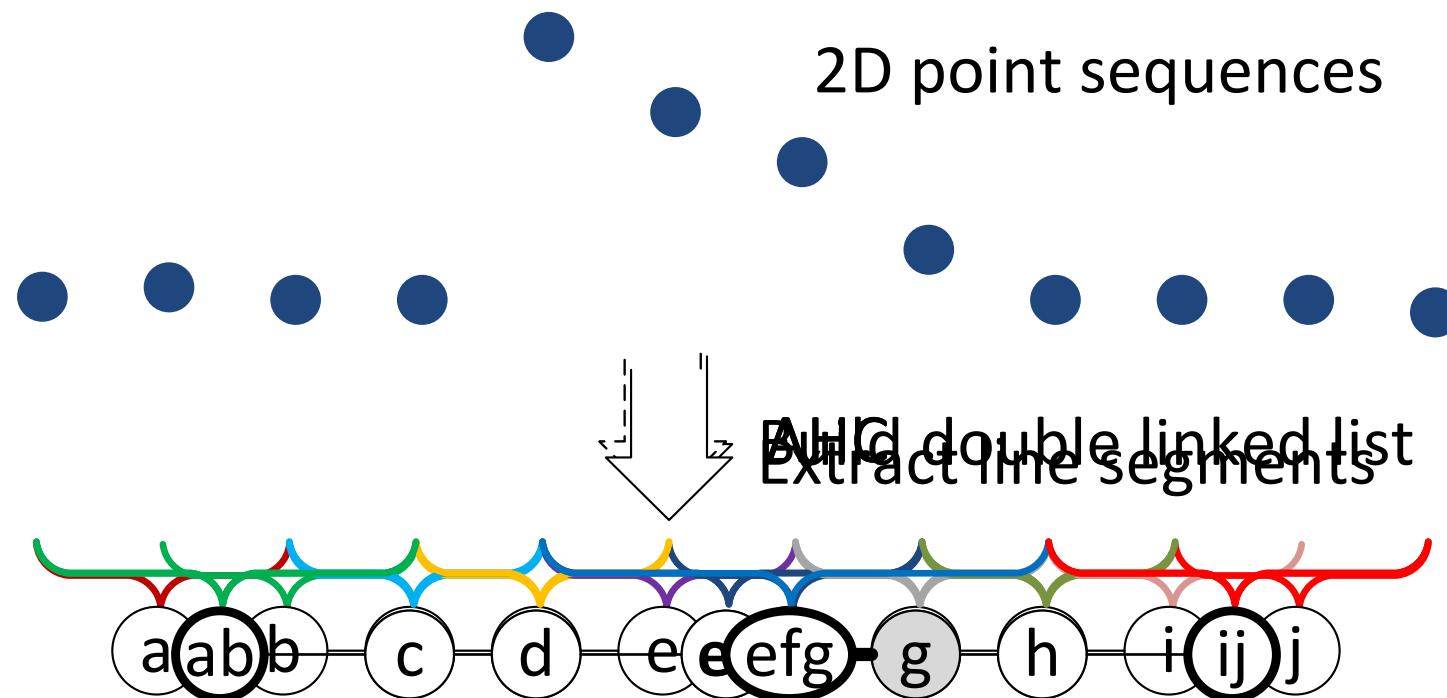
- Real-time plane extraction is crucial to various applications in robotics, computer vision, and 3D modeling:
  - Table-top object manipulation
  - Landmarks for SLAM
  - Compact and semantic scene modeling
- We present an efficient and reliable fast plane extraction algorithm applicable to organized point clouds, such as depth maps obtained by Kinect sensors.

- Previous Work
  - RANSAC-based
    - “Surfels” from Hough Transform (Oehler et al. 2011)
    - RANSAC on local region (Taguchi et al. 2013; Hulik et al. 2012; Lee et al. 2012)
  - Region-grow-based
    - Point-plane distance/MSE threshold (Hahnel et al. 2003; Poppinga et al. 2008)
    - Surface normal deviation threshold (Holz & Behnke 2012)
    - Line segments grow (Georgiev et al. 2011)
  - Graph-based (Strom et al. 2010; Wang et al. 2013)
  - Other
    - Normal space clustering (Holz et al. 2011)
    - Gradient-of-depth feature (Enjarini et al. 2012)

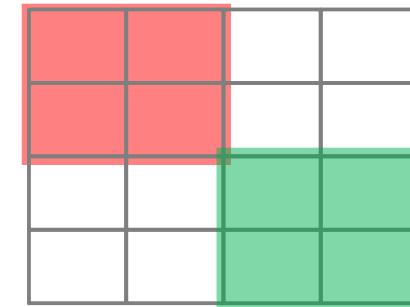
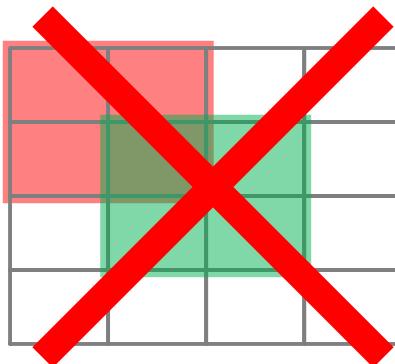
- Previous Work



- **Analogy to Line Regression** (Nguyen et al. 2005; April Robotics Toolkit, 2010)
  - Exploit the neighborhood information given by the order of points

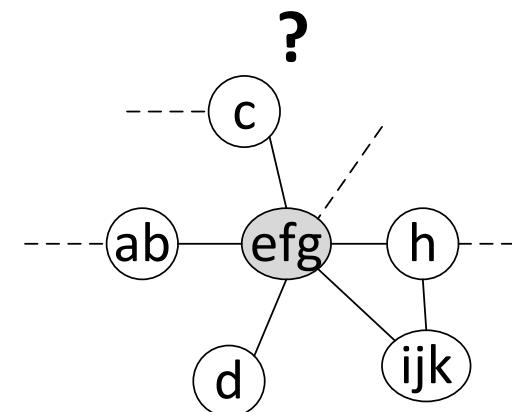
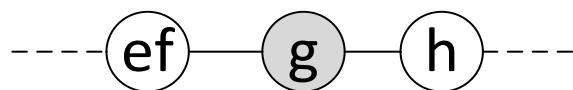


- Non-trivial Generalization to 3D
  - None-overlapping nodes

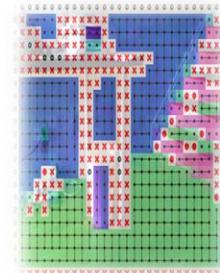
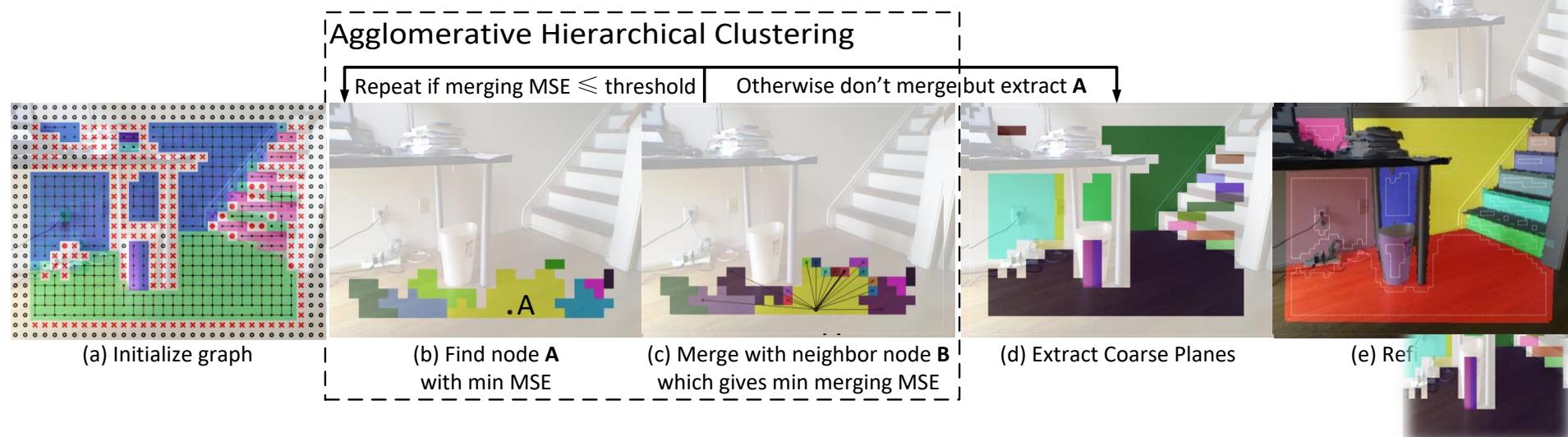


- Number of merging attempts

$\leq 2$

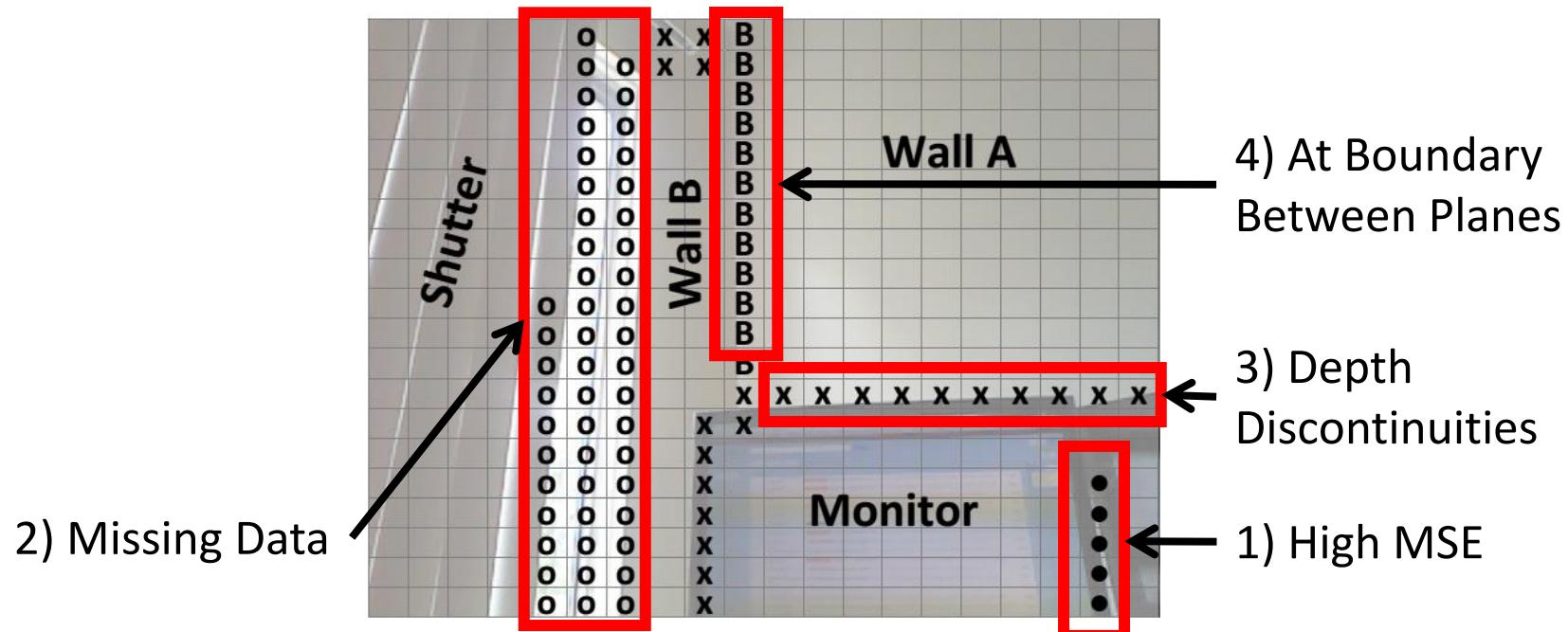


## • Algorithm Overview

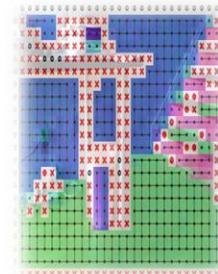


- Graph Initialization

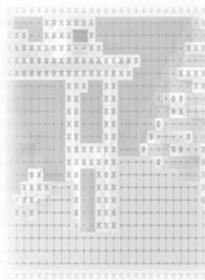
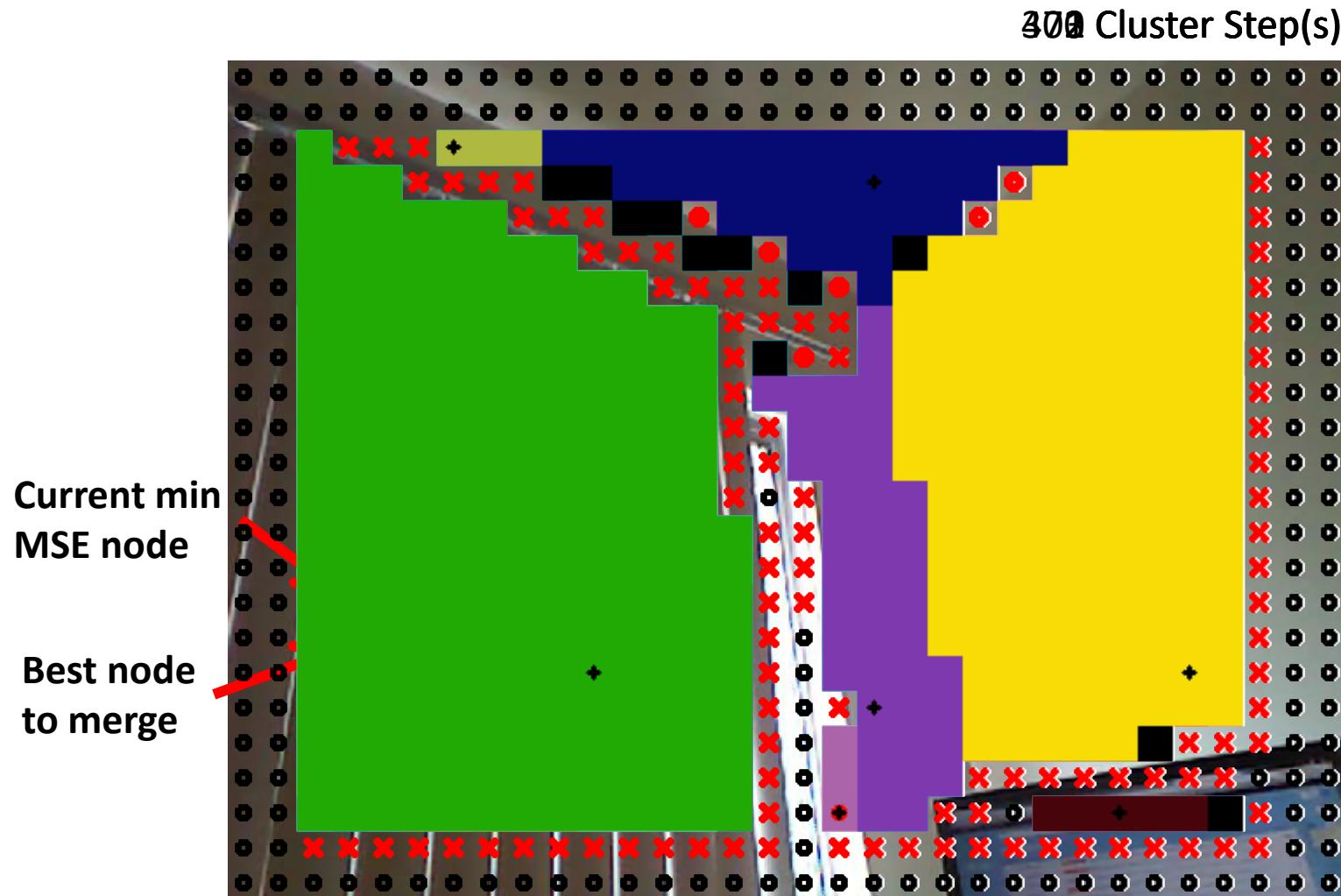
- Non-overlapping node initialization
- Rejecting “bad” nodes



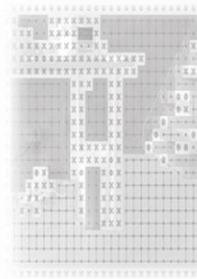
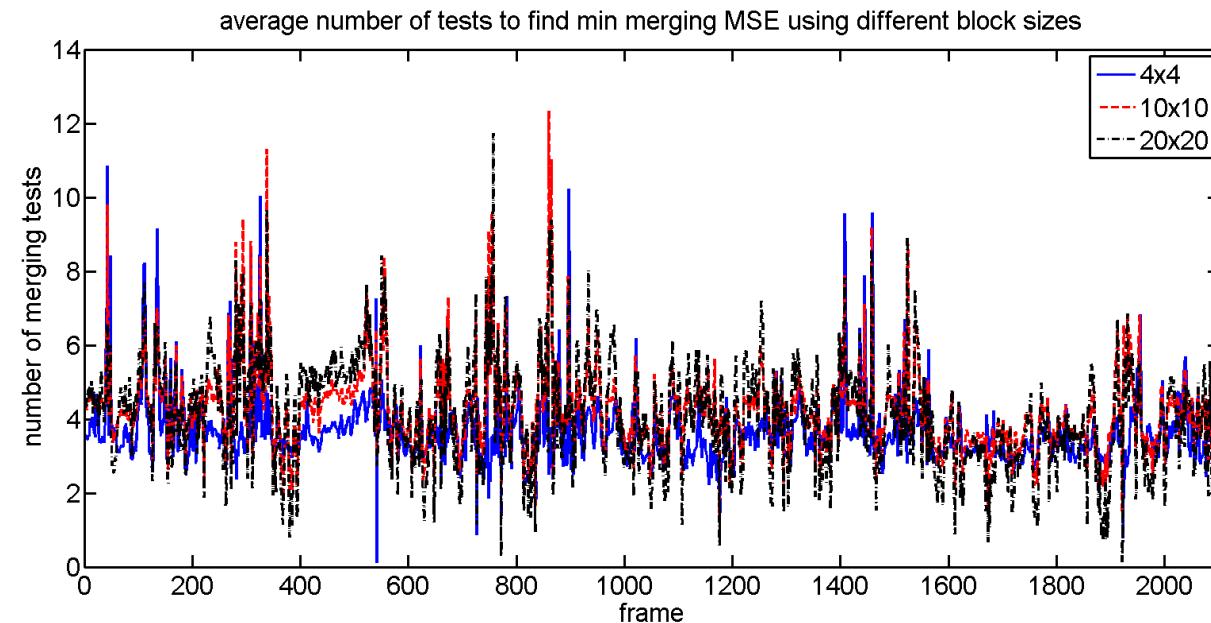
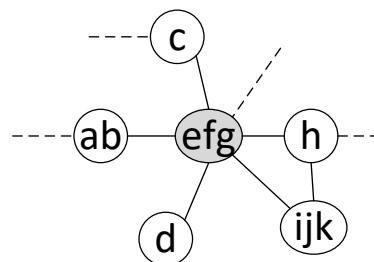
- Good! Avoid per-point normal estimation



- Agglomerative Hierarchical Clustering

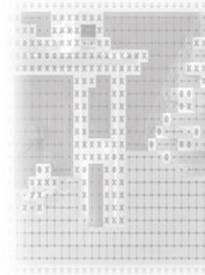
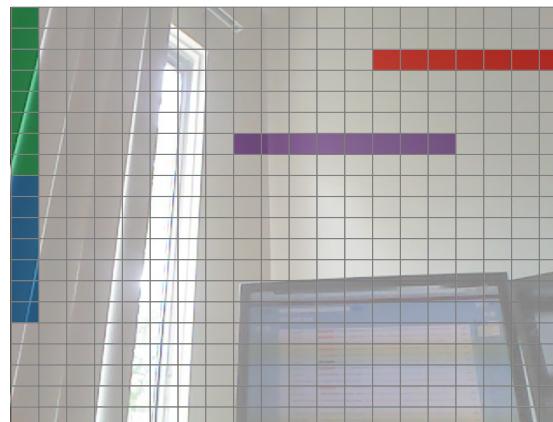


- Average Number of Merging Attempts
  - Small irrespective of initial number of nodes
  - Planar graph! Average node degree < 6
  - Merging is empirically a constant-time operation
    - $O(n \log n)$ , only arise from maintaining the min-heap



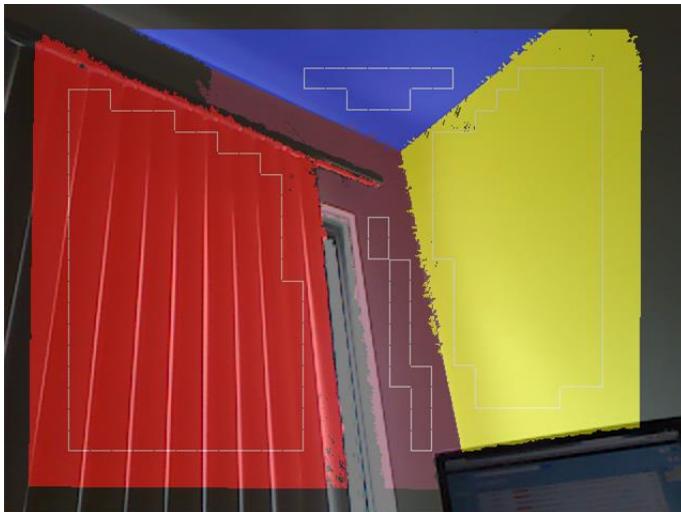
- Implementation Details

- Disjoint set
- Min-heap
- Second-order statistics
- Depth discontinuity/MSE threshold  
(Holzer et al. IROS 2012; Khoshelham & Elberink, 2012)
- Avoid strip-like initial node shape

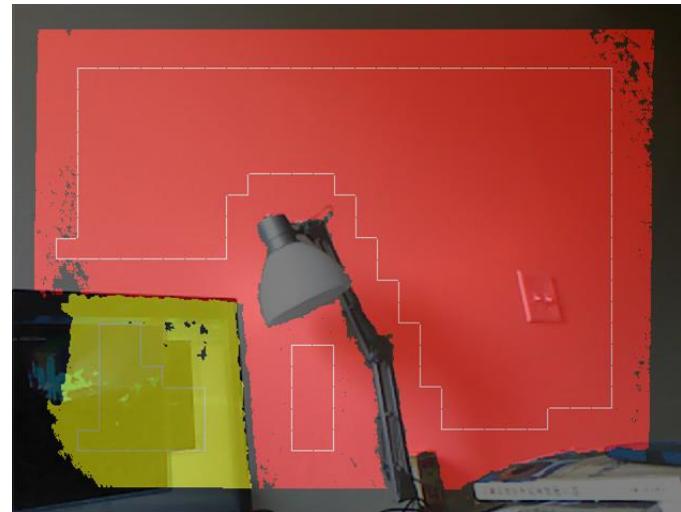


- Segmentation Refinement

- Artifacts

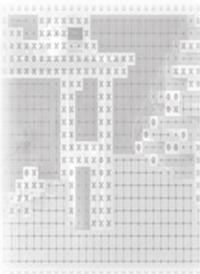


Sawtooth



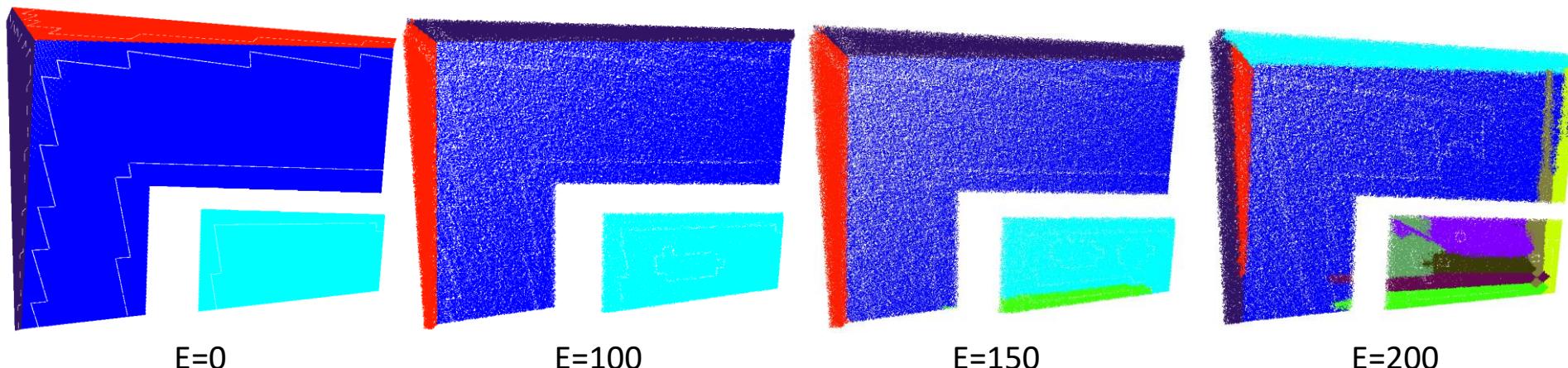
Over-segmentation

- Pixel-wise region-grow refinement
  - Only check boundary blocks and points

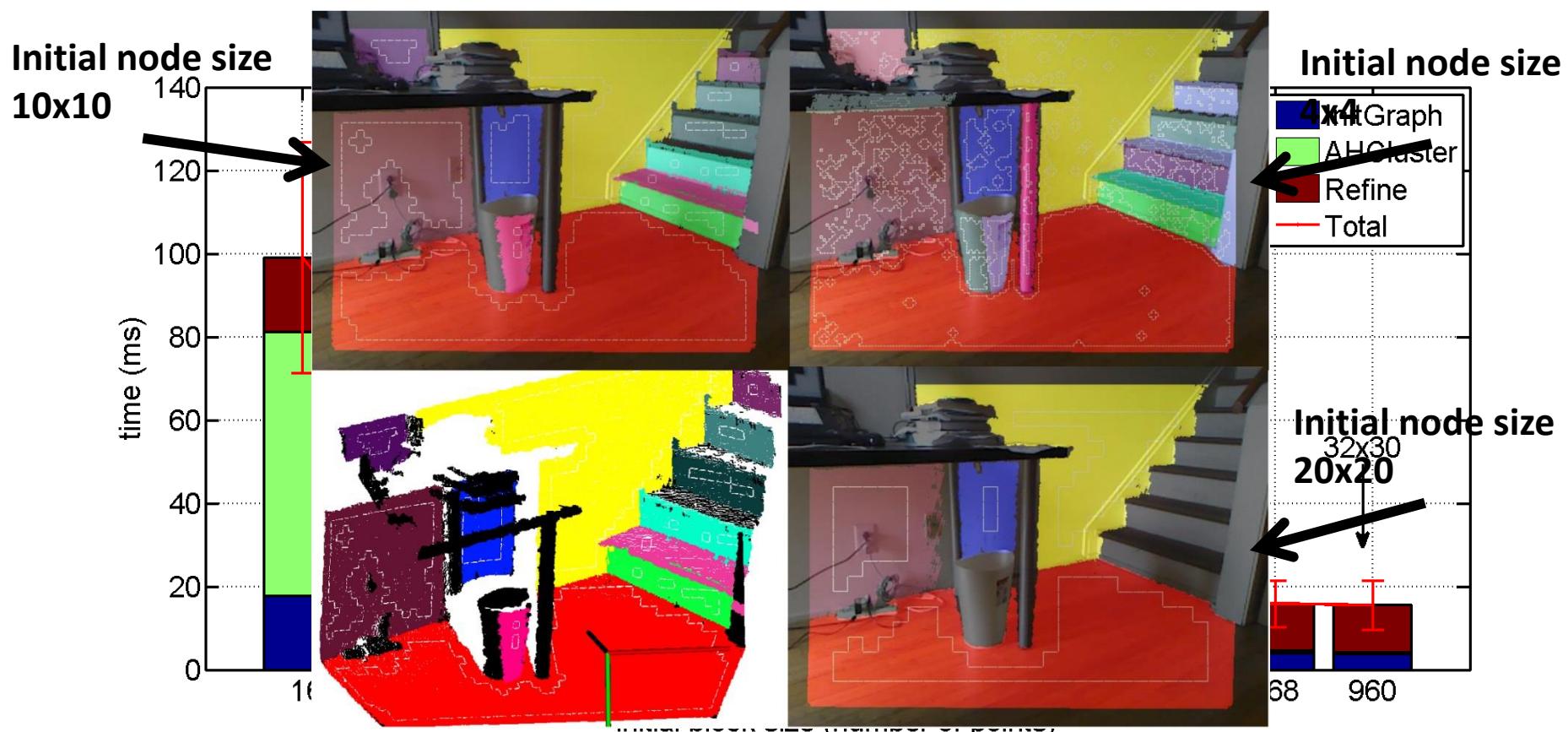


- Simulated Data

- Robustness to uniformly distributed depth noise (Georgiev et al., IROS 2011)
- Noise magnitude  $E = 0, 10, \dots, 200\text{mm}$
- Ground truth depth ranges from 1396mm to 3704mm



- Real-World Kinect Data
  - 2102 frames of an indoor scene
  - $640 \times 480$  pixel/frame



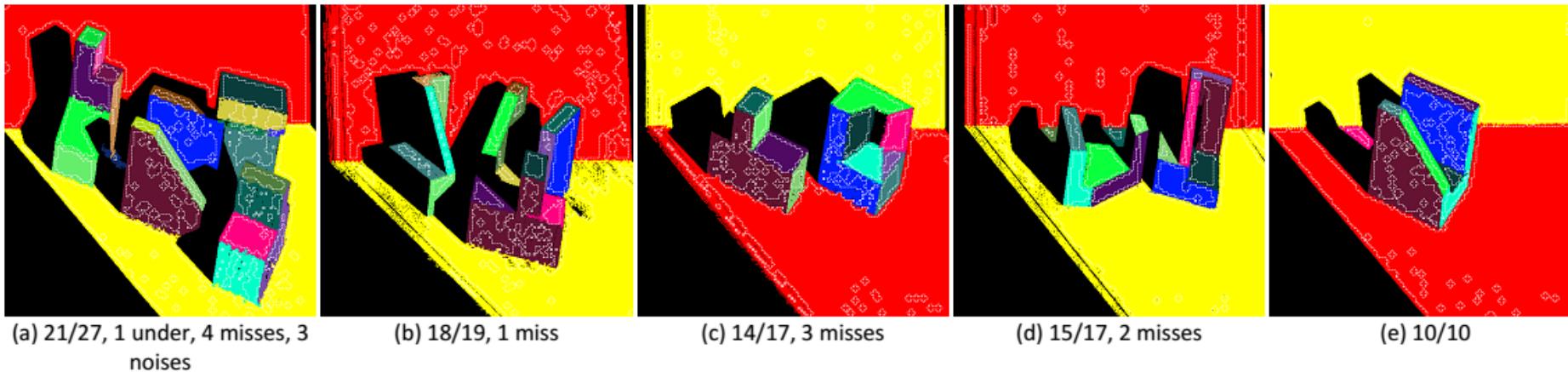
- Real-World Kinect Data

## Algorithm Breakdown

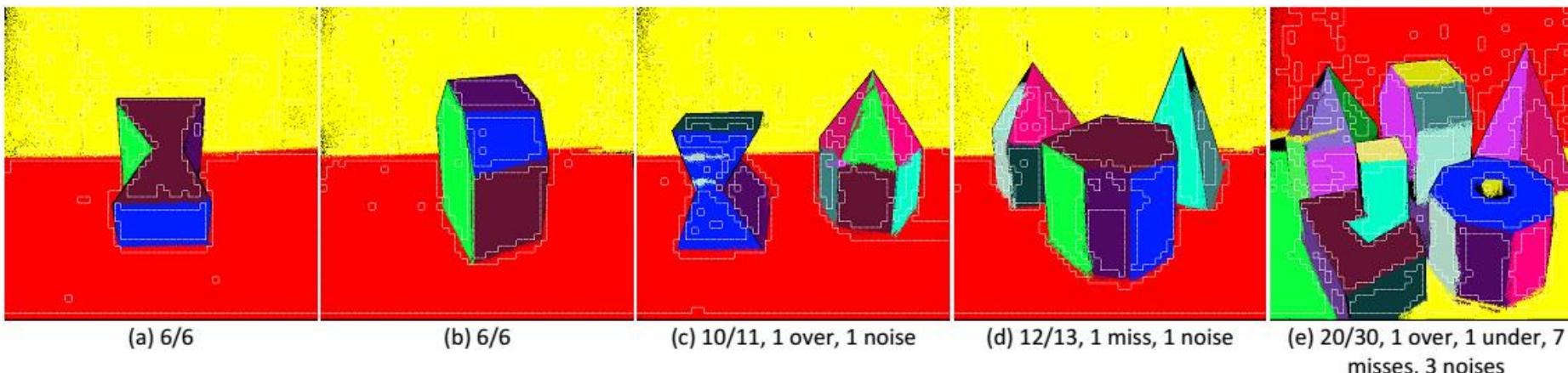
### 1) Graph Initialization

- SegComp Datasets (Hoover et al. PAMI 1996)

- ABW-TEST



- PERCEPTRON-TEST



- SegComp Benchmark (Gotardo et al. CVPR 2003; Oehler et al. ICIRA 2011; Holz & Behnke IAS 2012)

Approach	Regions in ground truth	Correctly detected	Orientation deviation (°)	Over-segmented	Under-segmented	Missed (not detected)	Noise (non-existent)
SegComp ABW data set (30 test images) by Hoover et al. [26], assuming 80% pixel overlap as in [27]							
USF [27]	15.2	12.7 (83.5%)	1.6	0.2	0.1	2.1	1.2
WSU [27]	15.2	9.7 (63.8%)	1.6	0.5	0.2	4.5	2.2
UB [27]	15.2	12.8 (84.2%)	1.3	0.5	0.1	1.7	2.1
UE [27]	15.2	13.4 (88.1%)	1.6	0.4	0.2	1.1	0.8
OU [27]	15.2	9.8 (64.4%)	–	0.2	0.4	4.4	3.2
PPU [27]	15.2	6.8 (44.7%)	–	0.1	2.1	3.4	2.0
UA [27]	15.2	4.9 (32.2%)	–	0.3	2.2	3.6	3.2
UFPR [27]	15.2	13.0 (85.5%)	1.5	0.5	0.1	1.6	1.4
Oehler et al. [2]	15.2	11.1 (73.0%)	1.4	0.2	0.7	2.2	0.8
Holz et al. [8]	15.2	12.2 (80.1%)	1.9	1.8	0.1	0.9	1.3
<b>Ours</b>	<b>15.2</b>	<b>12.8 (84.2%)</b>	<b>1.7</b>	<b>0.1</b>	<b>0.0</b>	<b>2.4</b>	<b>0.7</b>
SegComp PERCEPTRON data set (30 test images) by Hoover et al. [26], assuming 80% pixel overlap as in [27]							
USF [27]	14.6	8.9 (60.9%)	2.7	0.4	0.0	5.3	3.6
WSU [27]	14.6	5.9 (40.4%)	3.3	0.5	0.6	6.7	4.8
UB [27]	14.6	9.6 (65.7%)	3.1	0.6	0.1	4.2	2.8
UE [27]	14.6	10.0 (68.4%)	2.6	0.2	0.3	3.8	2.1
UFPR [27]	14.6	11.0 (75.3%)	2.5	0.3	0.1	3.0	2.5
Oehler et al. [2]	14.6	7.4 (50.1%)	5.2	0.3	0.4	6.2	3.9
Holz et al. [8]	14.6	11.0 (75.3%)	2.6	0.4	0.2	2.7	0.3
<b>Ours</b>	<b>14.6</b>	<b>8.9 (60.9%)</b>	<b>2.4</b>	<b>0.2</b>	<b>0.2</b>	<b>5.1</b>	<b>2.1</b>

## Next Week

---

++ Marker-based SfM

\* Bundle adjustment

+ Error/uncertainty propagation

++ Feature-based SfM

\* COLMAP

\*: know how to code (or how to use tools)

++: know how to derive (more than just the concept)

+: know the concept

# References for Next Week

---

- Sz2022:
  - Chapter 11
- HZ2003:
  - Section 5.2, 18.1, A6.6
- <https://colmap.github.io/>