NYU TANDON SCHOOL OF ENGINEERING

AI4CE

Robot Perception

RGBD Sensors and Algorithms

Dr. Chen Feng

cfeng@nyu.edu

ROB-GY 6203, Fall 2023

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

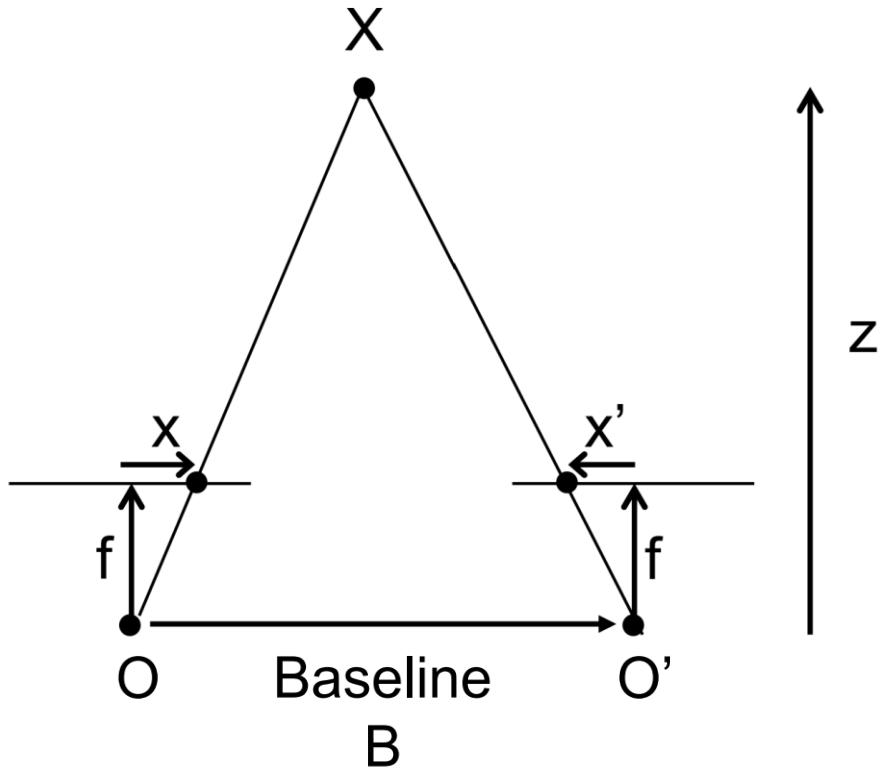
References

- 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

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

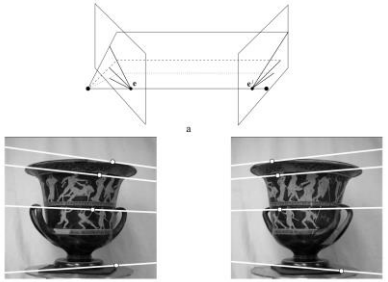


$$disparity = x - x' = \frac{B \cdot f}{z} \qquad z = \frac{B \cdot f}{x - x'}$$

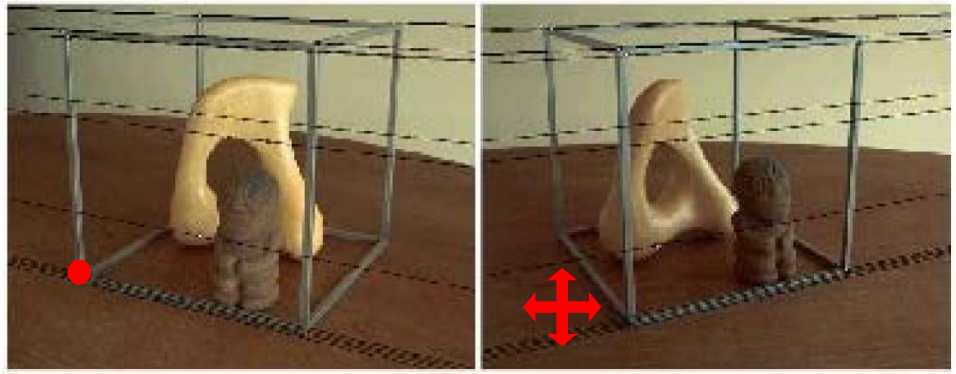
http://pages.cs.wisc.edu/~dyer/cs534/slides/17_kinect.pdf

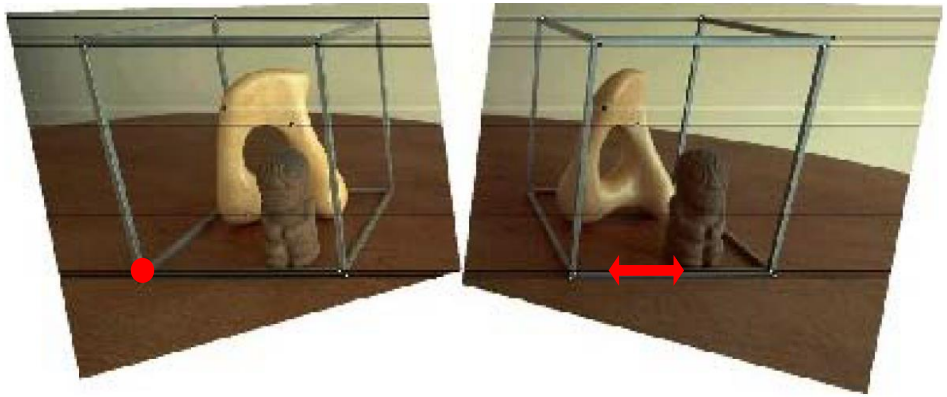
Depth from Stereo Images

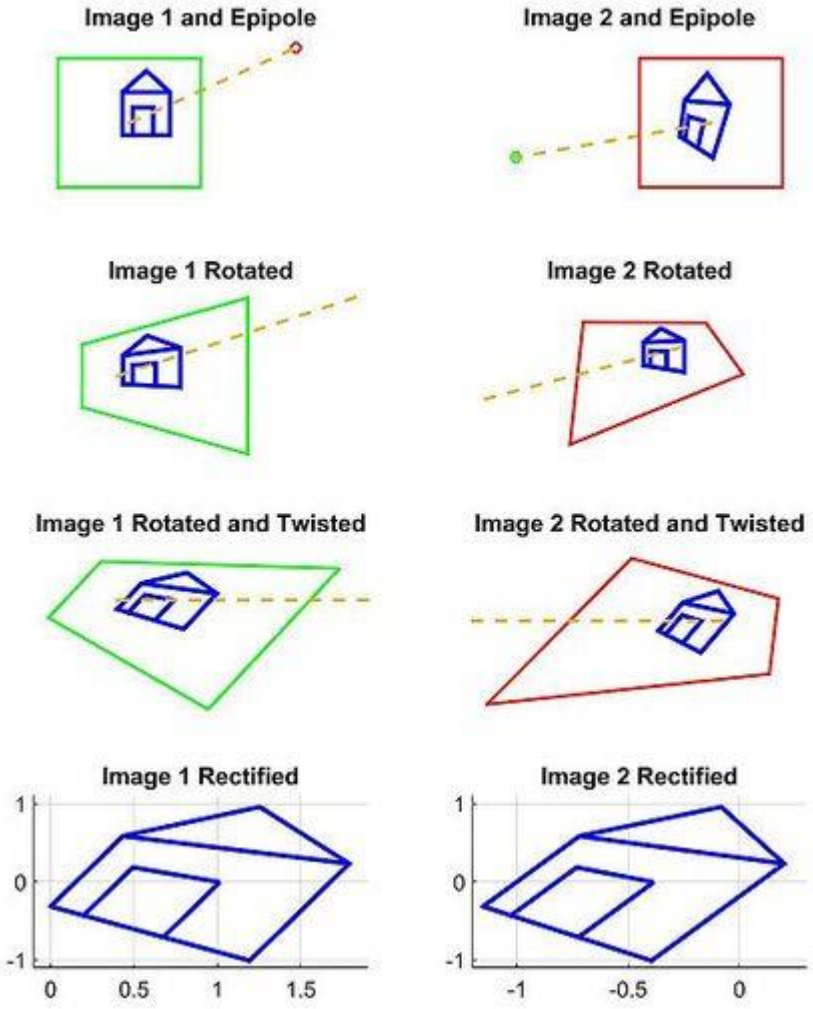
- Epipolar geometry: stereo calibration
  - Not only the intrinsic parameters
  - But also the extrinsics, R and t



HZ2003





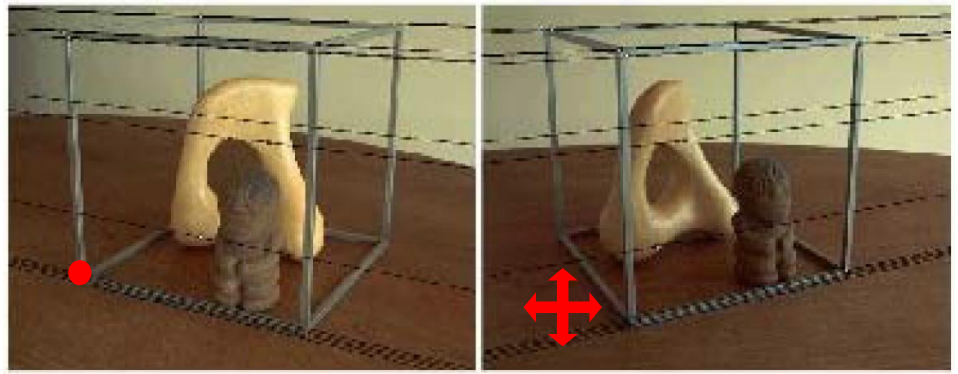


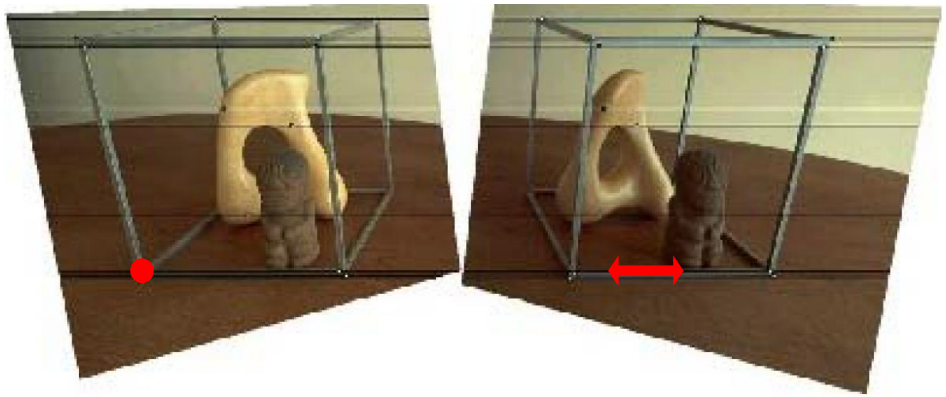
stereo image rectification

https://en.wikipedia.org/wiki/Image_rectification

Depth from Stereo Images

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

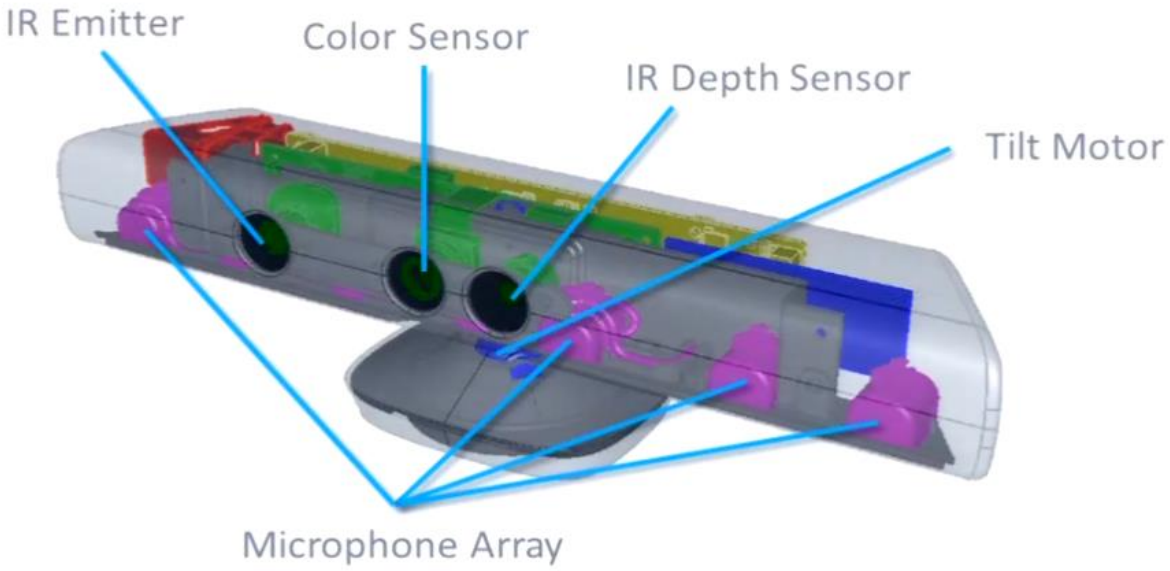


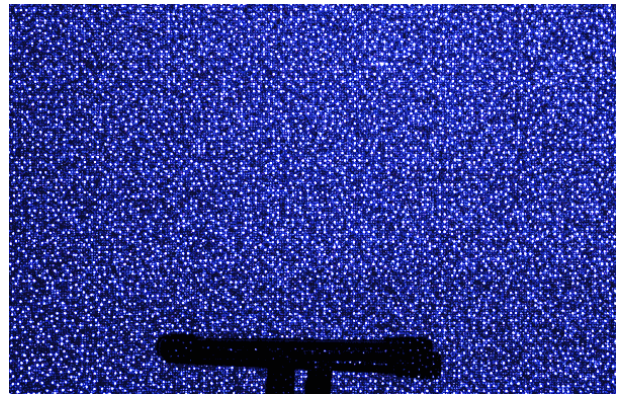


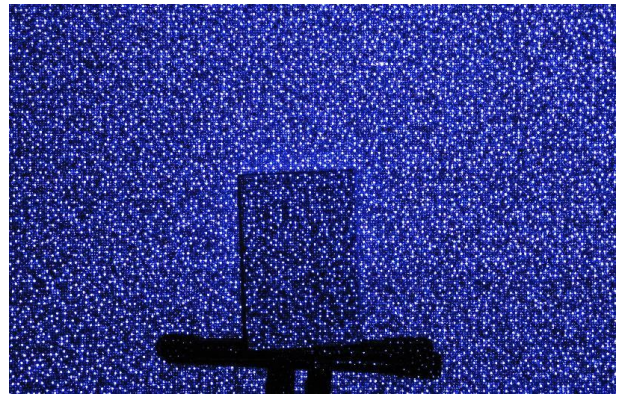
Szeliski 2011

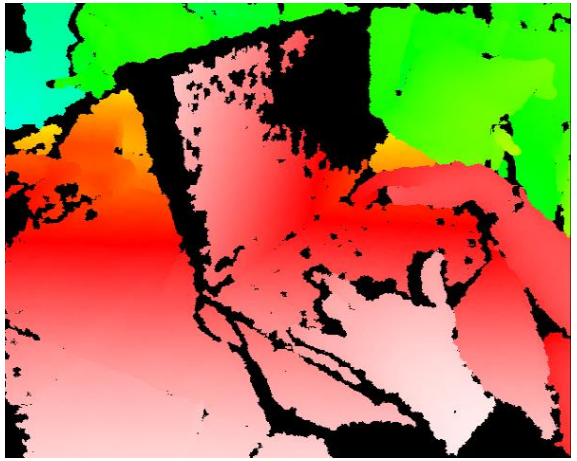
Depth from Structured Light

- 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

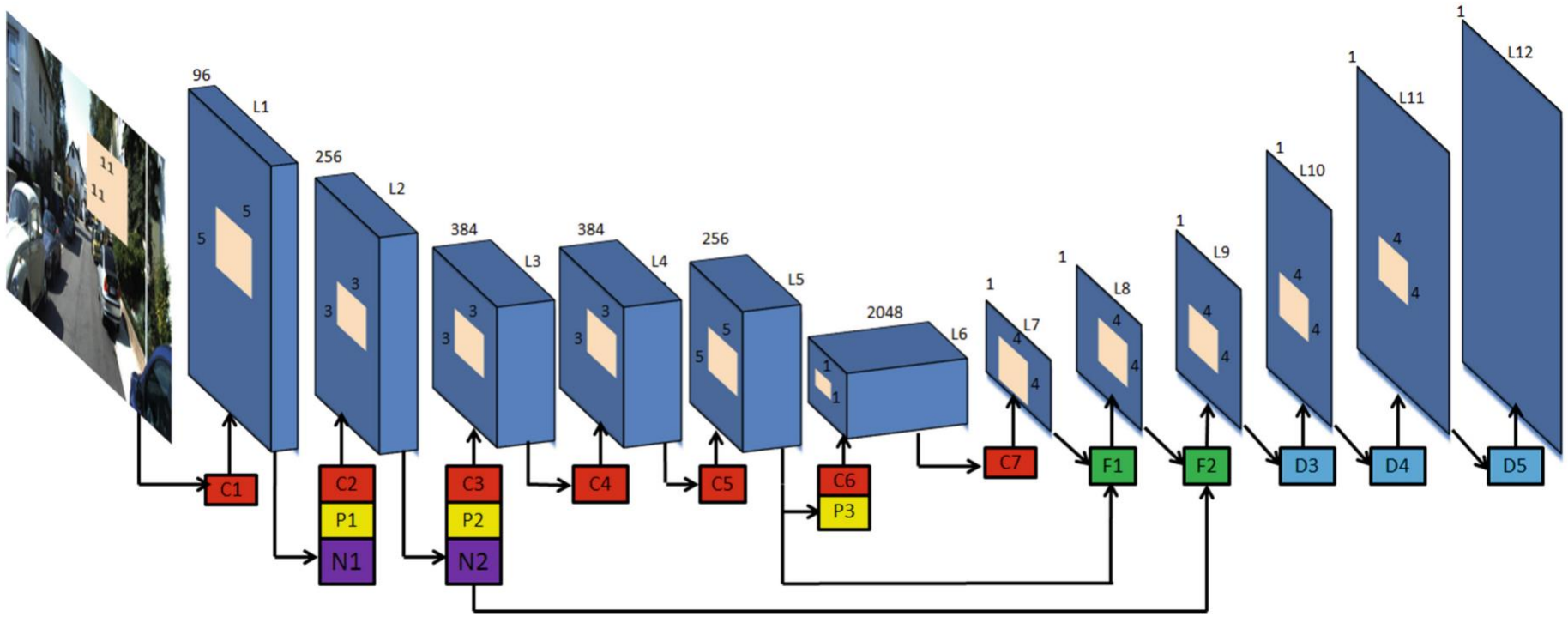
- If you do not have RGBD sensor
  - http://www.michaelfirman.co.uk/RGBDdatasets/

Monocular Depth Estimation

- Can we estimate depth from only a single image?

Monocular Depth Estimation

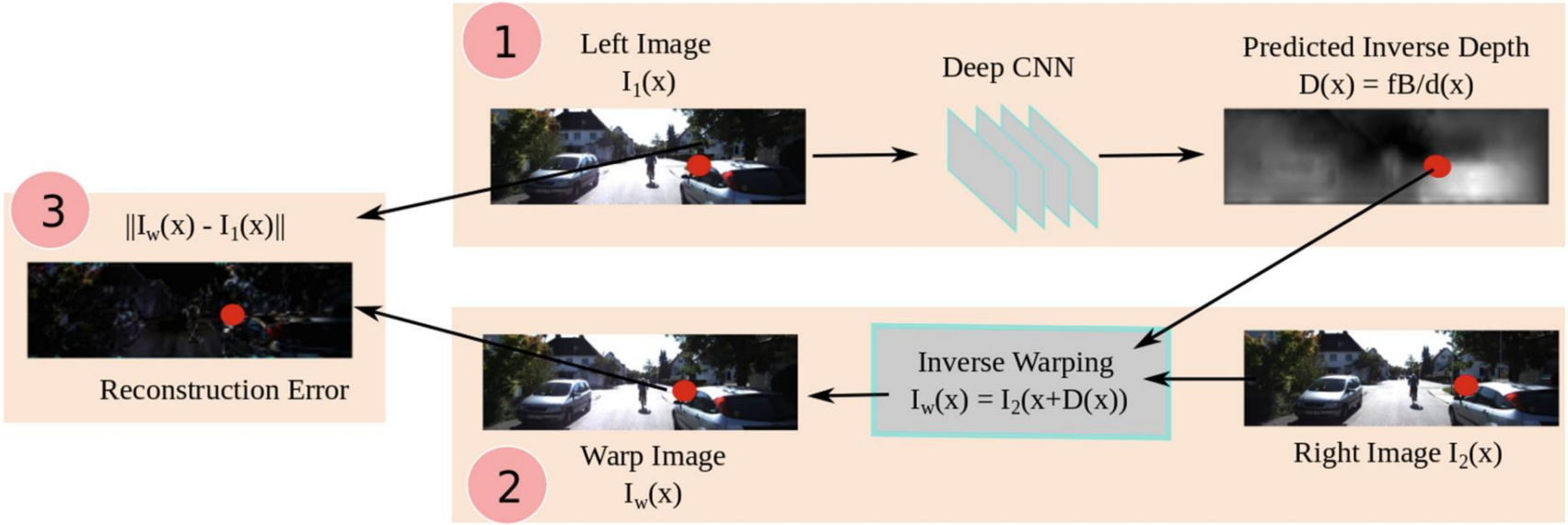
- Estimating depth from a single-view image **based on experience**
  - We could use supervised learning with regression loss (L2/MSE or L1 loss)
  - Need a large-scale dataset of paired RGB and *ground truth* Depth images



Garg, Ravi, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. "Unsupervised cnn for single view depth estimation: Geometry to the rescue." ECCV'16.

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



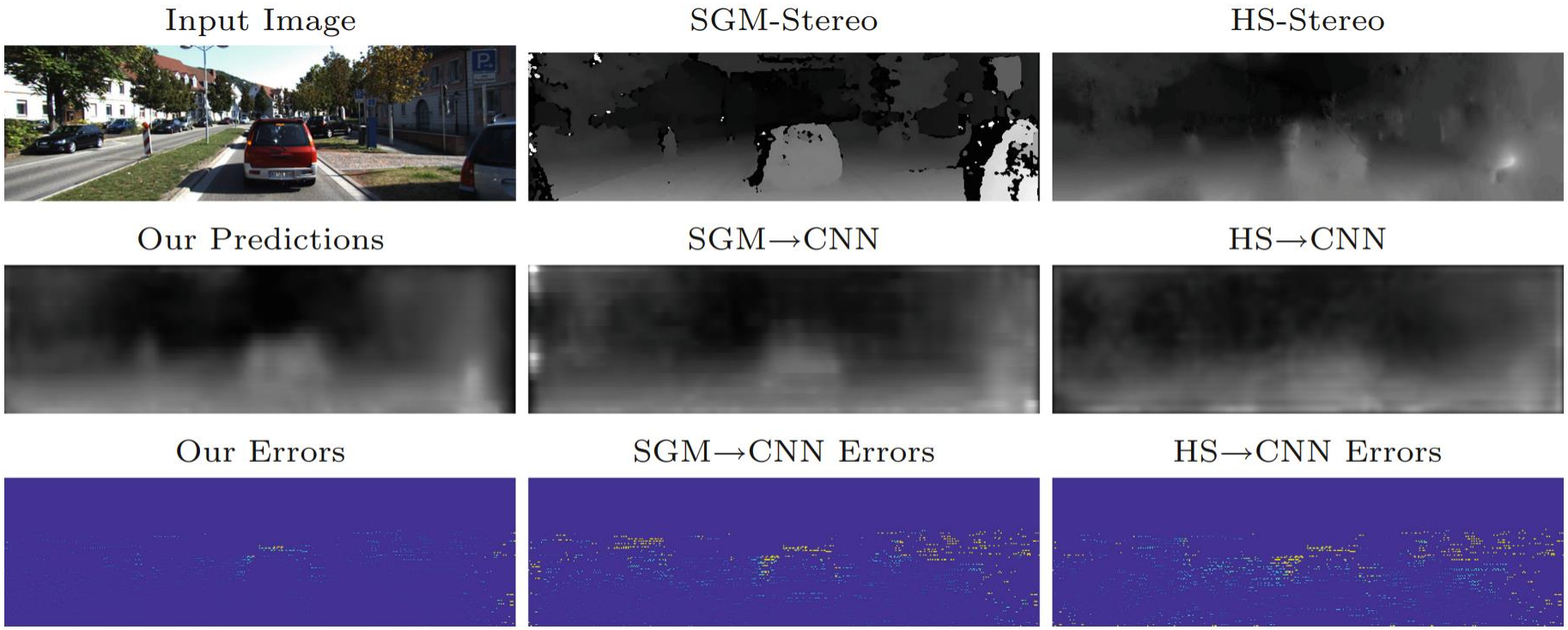
$$E^i_{recons} = \int_\Omega \|I^i_w(x) - I^i_1(x)\|^2 dx = \int_\Omega \|I^i_2(x + \underbrace{D^i(x)}_{fB/d^i(x)}) - I^i_1(x)\|^2 dx$$

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

Garg, Ravi, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. "Unsupervised cnn for single view depth estimation: Geometry to the rescue." ECCV'16.

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

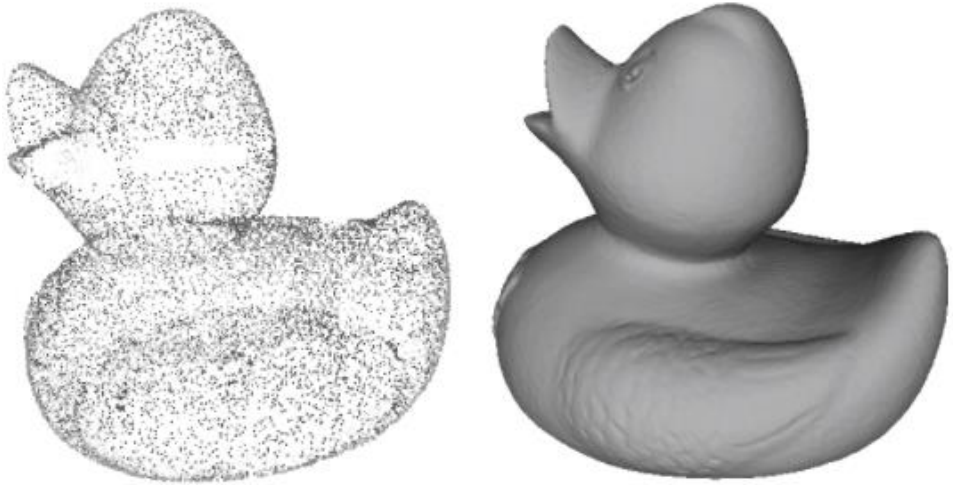


Garg, Ravi, Vijay Kumar Bg, Gustavo Carneiro, and Ian Reid. "Unsupervised cnn for single view depth estimation: Geometry to the rescue." ECCV'16.

3D Data Representations

Point Cloud/Mesh

- ✓ Raw format/Efficiency
- • Explicit representation
- × Unorganized/Unordered



https://elmoatazbill.users.greyc.fr/point_cloud/reconstruction.png

Voxel

- • Implicit representation
- × Resolution/Scalability
- × Discretization artifact

https://www.planetminecraft.com/project/giant-snowman-1638162/

Primitives

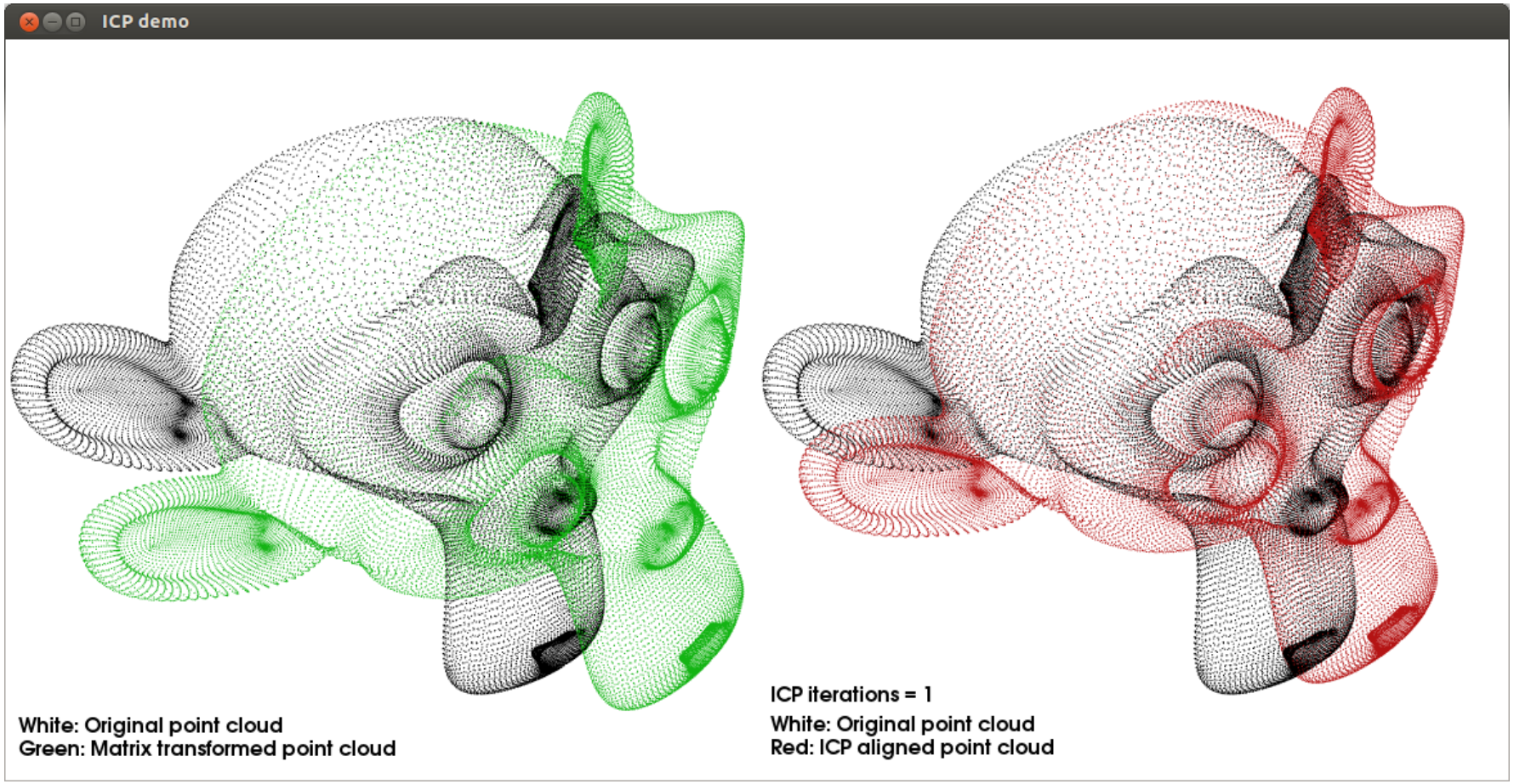
- ✓ Compact
- ✓ Ready for interaction
- • Complex shapes?

http://pointclouds.org/gsoc/

Scan Registration

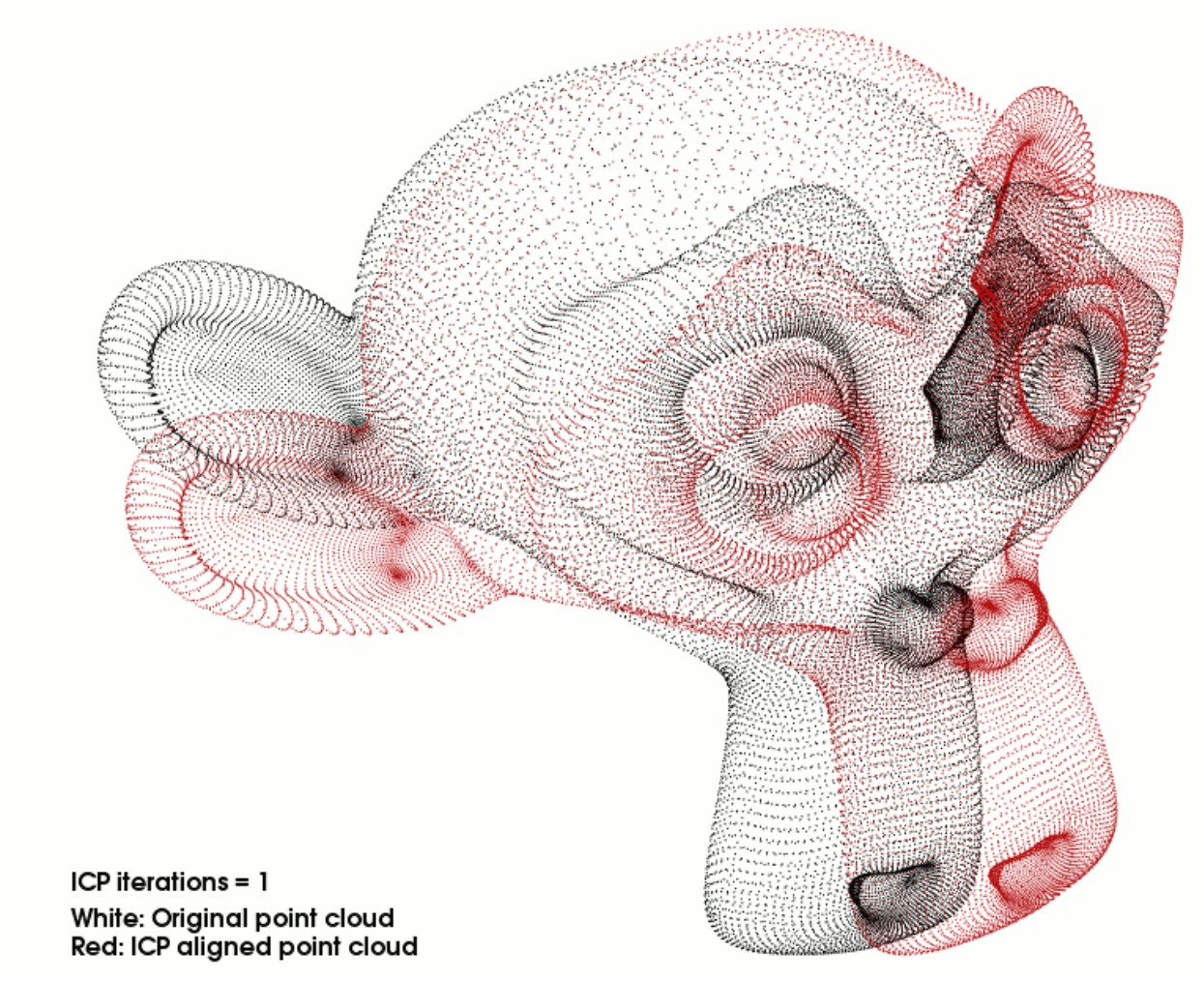
http://pointclouds.org/documentation/tutorials/registration_api.php

Iterative Closest Point (ICP)



https://pcl.readthedocs.io/projects/tutorials/en/latest/interactive_icp.html

ICP Process



https://pcl.readthedocs.io/projects/tutorials/en/latest/interactive_icp.html

Core Problem Definition

- Given two sets of **corresponding** points
  - Source: $S = \{S_1, S_2, \cdots, S_n\}$
  - Destination $\mathrm{D} = \{D_1, D_2, \cdots, D_n\}$

- 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
- $C(R,t) = \sum_i \left\| D_i - RS_i - (\sum_i D_i - R\sum_i S_i)/n \right\|^2$

- 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

- Basically: **Center** point clouds, then **rotate** to align further. Repeat as necessary.

Wait, but I Do Not Know Correspondences!

- Approximate correspondences
  - Simply pick the closest point
- Then iterate until converge

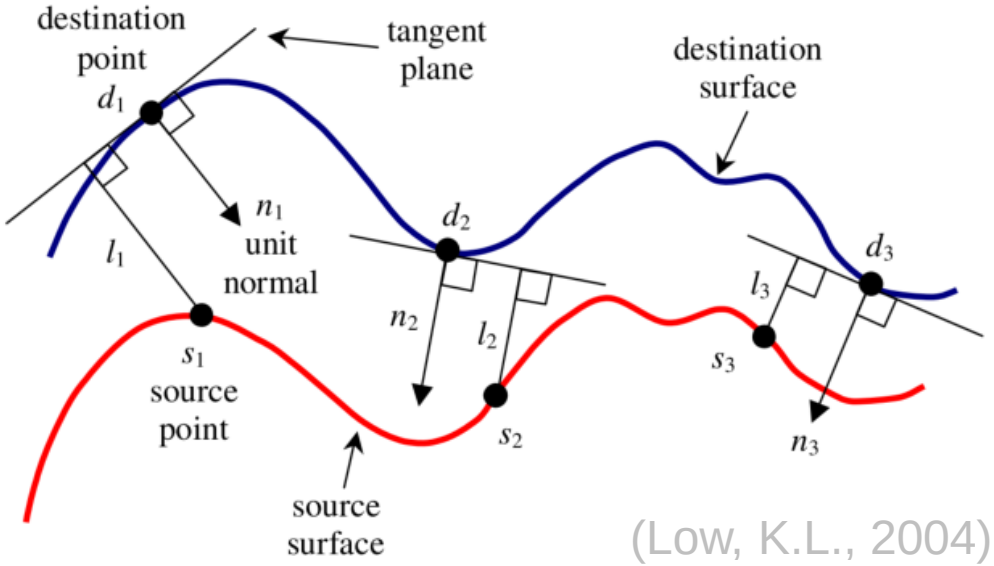
- ICP step-by-step
  1. **Find Correspondences**:
     Find correspondences of **S** in **D** by searching closest points (Brute-Force, KdTree)

  2. **Compute Center and Rotation**:
     Solve the current optimal translation and rotation by Procrustes analysis

  3. **Align via Centering and Rotating**:
     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: $C(R,t) = \sum_i \left\|N_i^T (D_i - RS_i - t)\right\|^2$, 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)

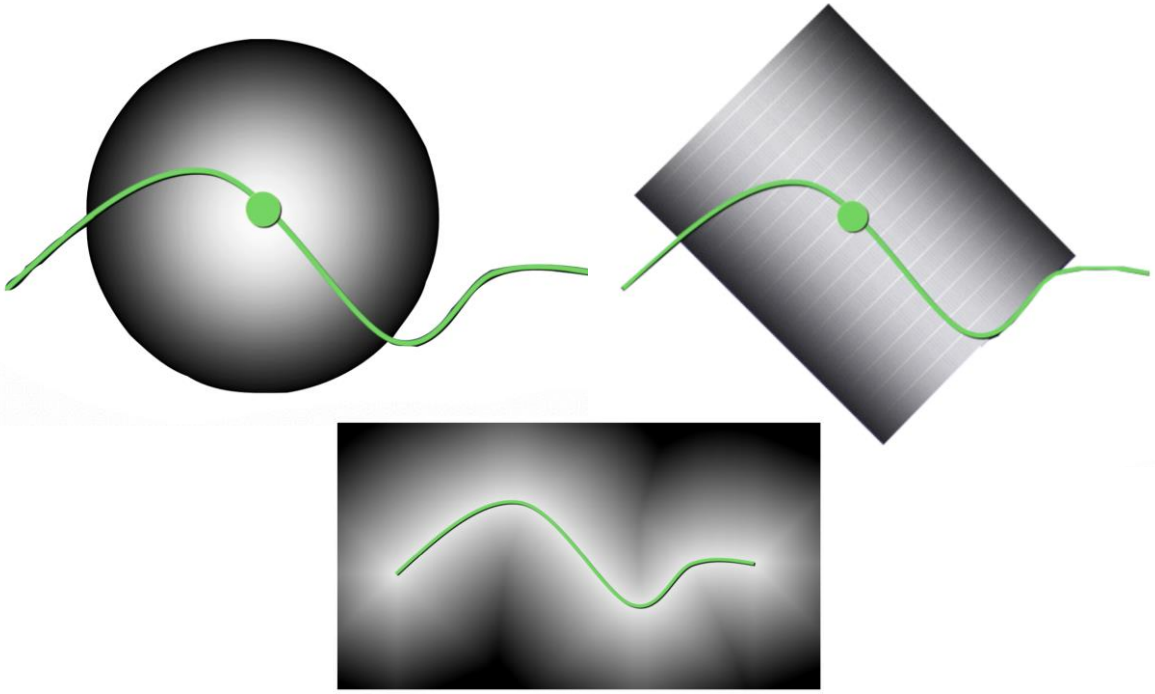
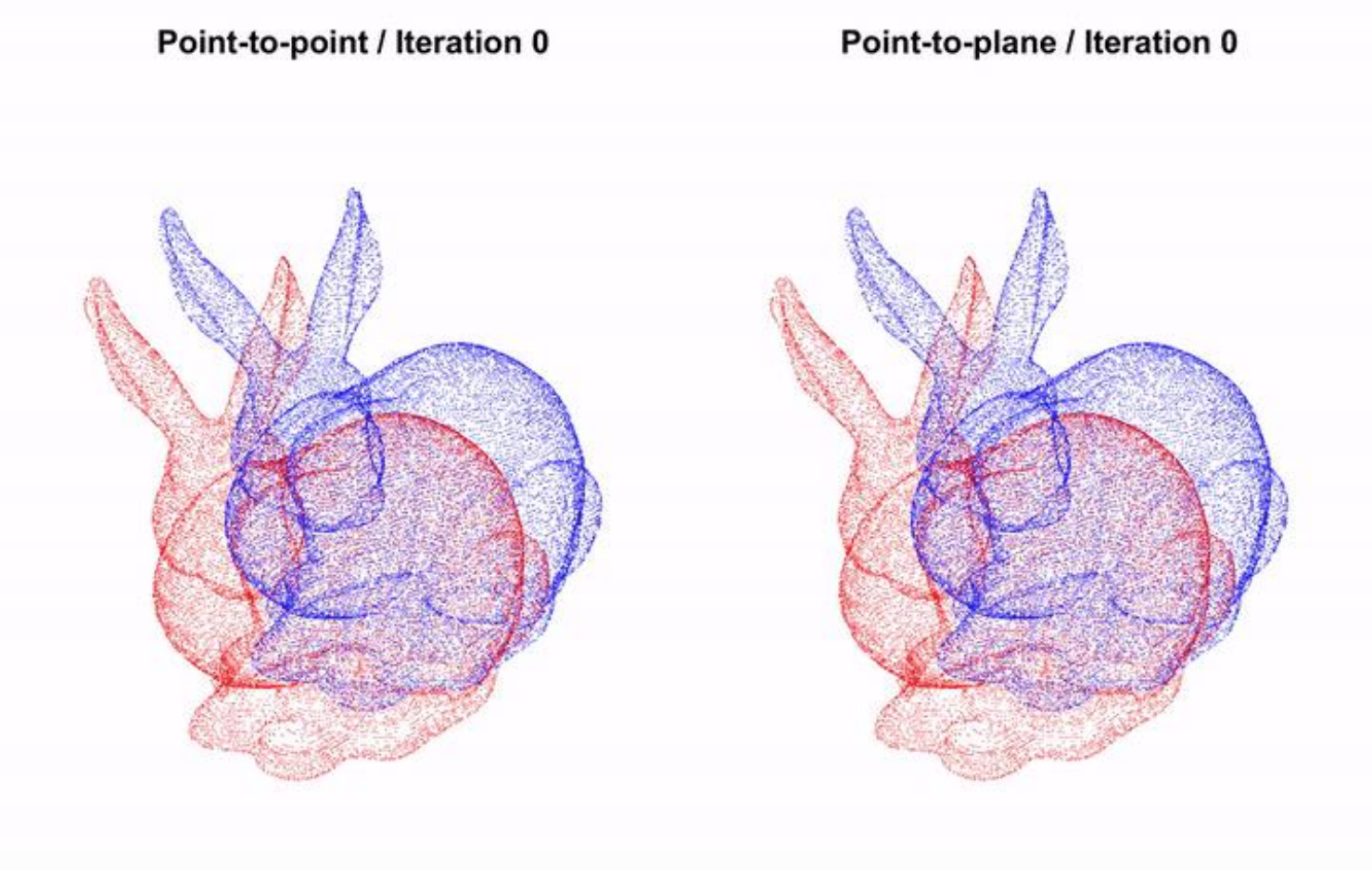


Image from Hao Li.

Yang Chen, and Gerard Medioni. *Object Modeling by Registration of Multiple Range Images*. International Journal of Image and Vision Computing, 10(3), pp. 145–155, 1992.

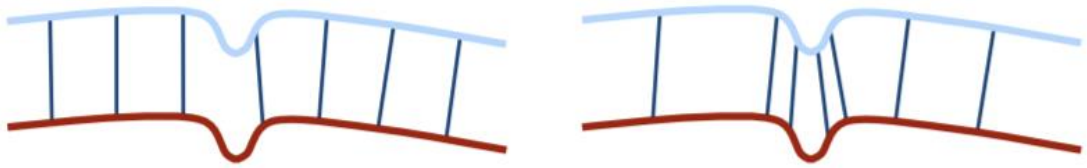
ICP variants - Point-to-Point vs Point-to-Plane ICP



From Philip Glira - https://www.youtube.com/watch?v=LcghboLgTiA

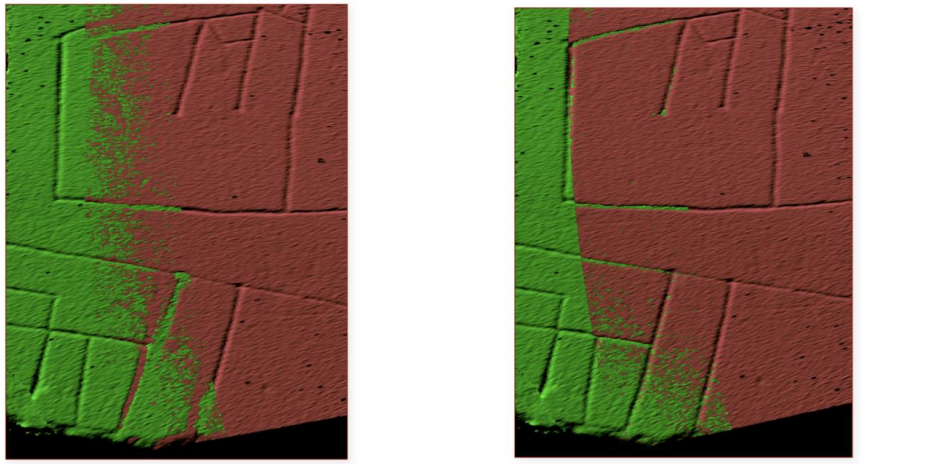
ICP variants – Sampling Source Points

- 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

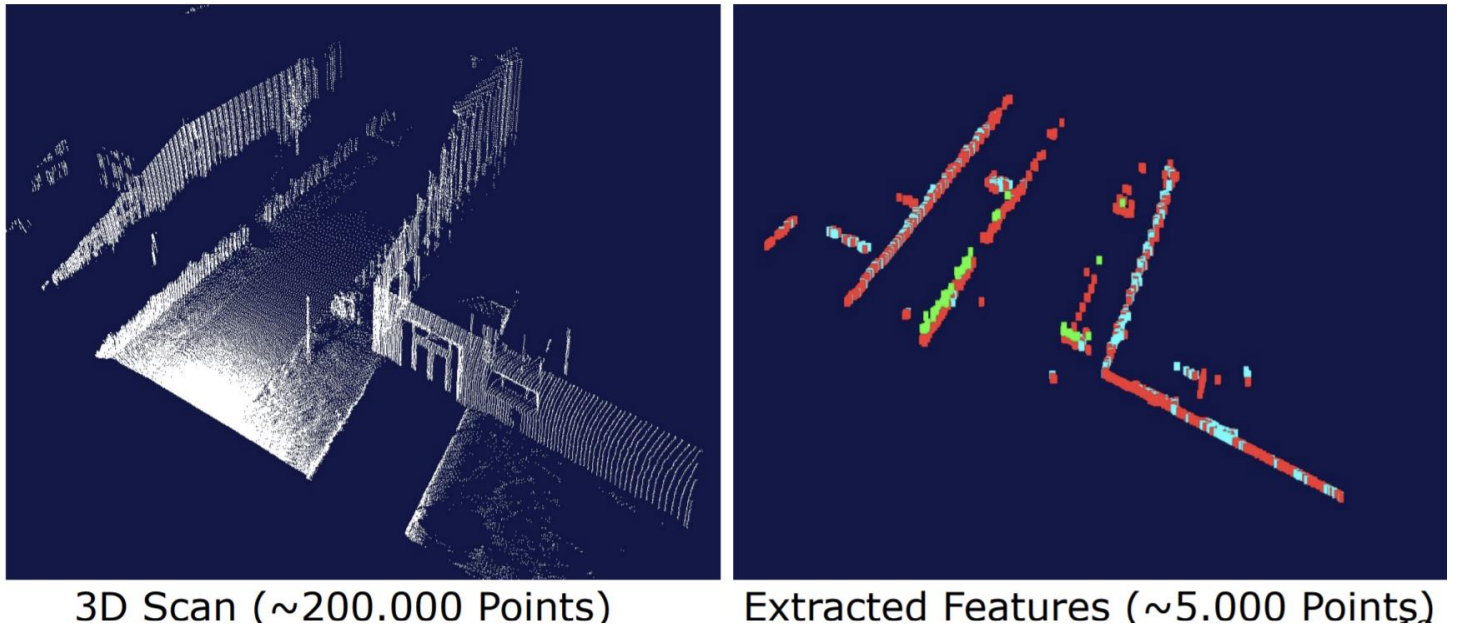


Image from Burgard, et al.

ICP variants - Change Data Association

- Beyond closest point search

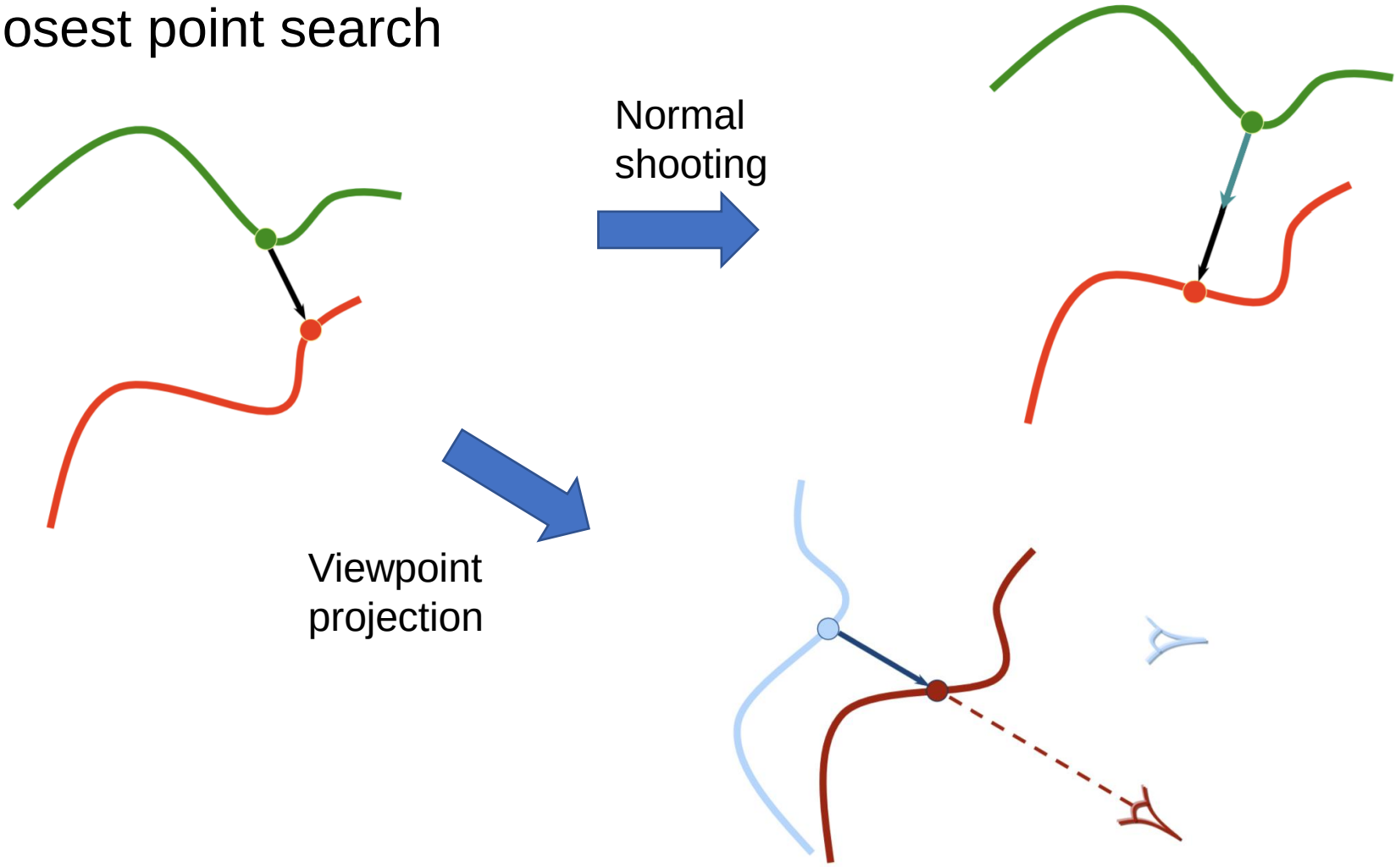


Image from Burgard, et al.

ICP for Degenerated Cases

- Solution: use color-based ICP

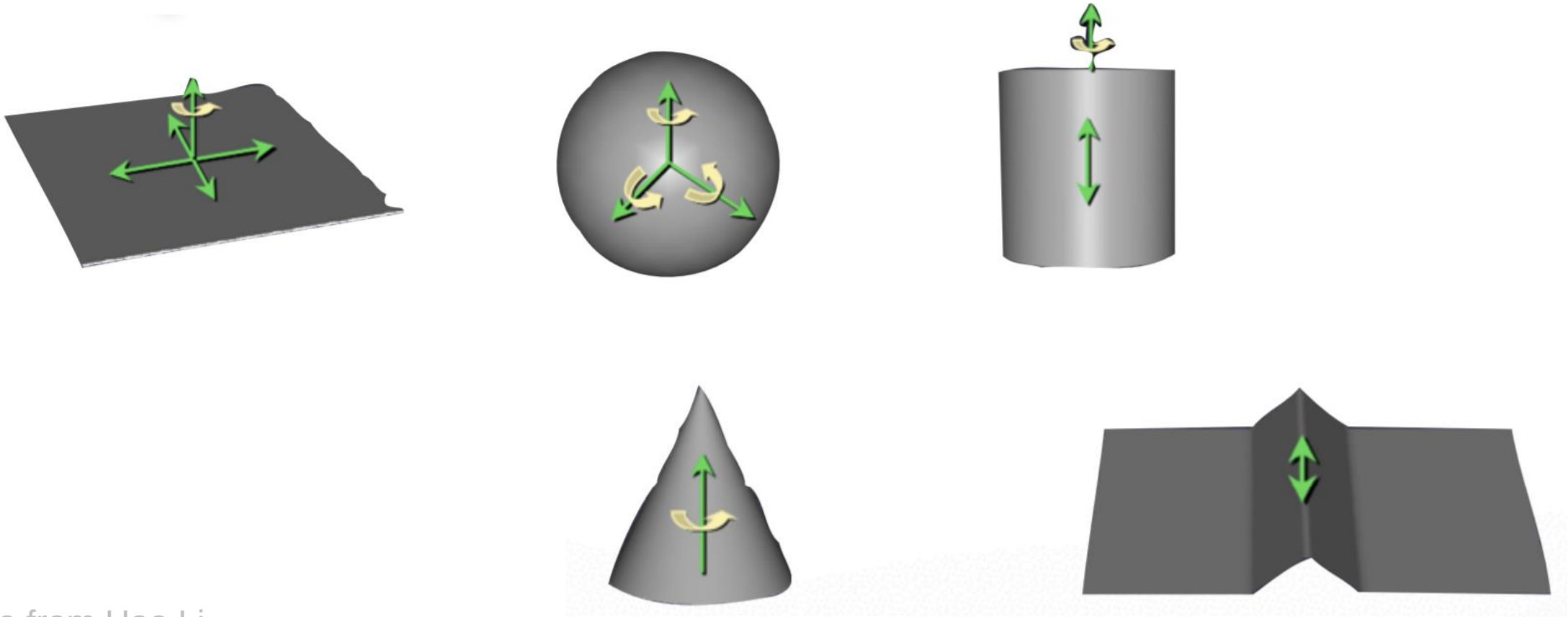


Image from Hao Li.

Recent Work with Deep Learning

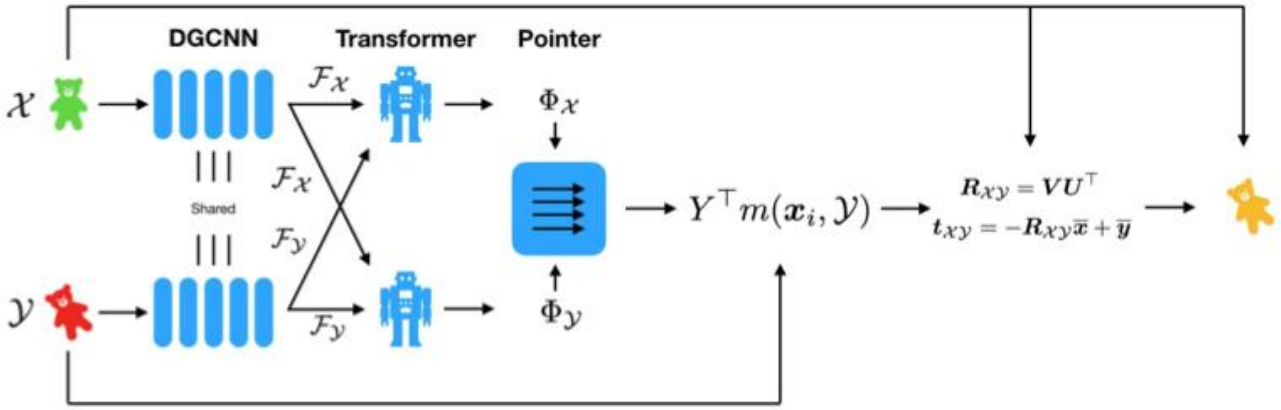


Figure 2. Network architecture for DCP, including the Transformer module for DCP-v2.

Model	MSE(R)	RMSE(R)	MAE(R)	MSE(t)	RMSE(t)	MAE(t)
ICP	892.601135	29.876431	23.626110	0.086005	0.293266	0.251916
Go-ICP [55]	192.258636	13.865736	2.914169	0.000491	0.022154	0.006219
FGR [61]	97.002747	9.848997	**1.445460**	0.000182	0.013503	0.002231
PointNetLK [18]	306.323975	17.502113	5.280545	0.000784	0.028007	0.007203
DCP-v1 (ours)	19.201385	4.381938	2.680408	**0.000025**	**0.004950**	**0.003597**
DCP-v2 (ours)	**9.923701**	**3.150191**	2.007210	**0.000025**	0.005039	0.003703

Table 2. ModelNet40: Test on unseen categories

Notes

- **Deep Closest Point**. Outperforms conventional ICP significantly in alignment.
- Uses Deep Learning to
  - Find good correspondences between point clouds – common failure of ICP is bad (far) initializing correspondences
  - Then regress to find a good mapping between the points
  - Finally solve for the optimal translation and rotation via differentiable learned SVD.

Recent Work ~~with Deep Learning~~

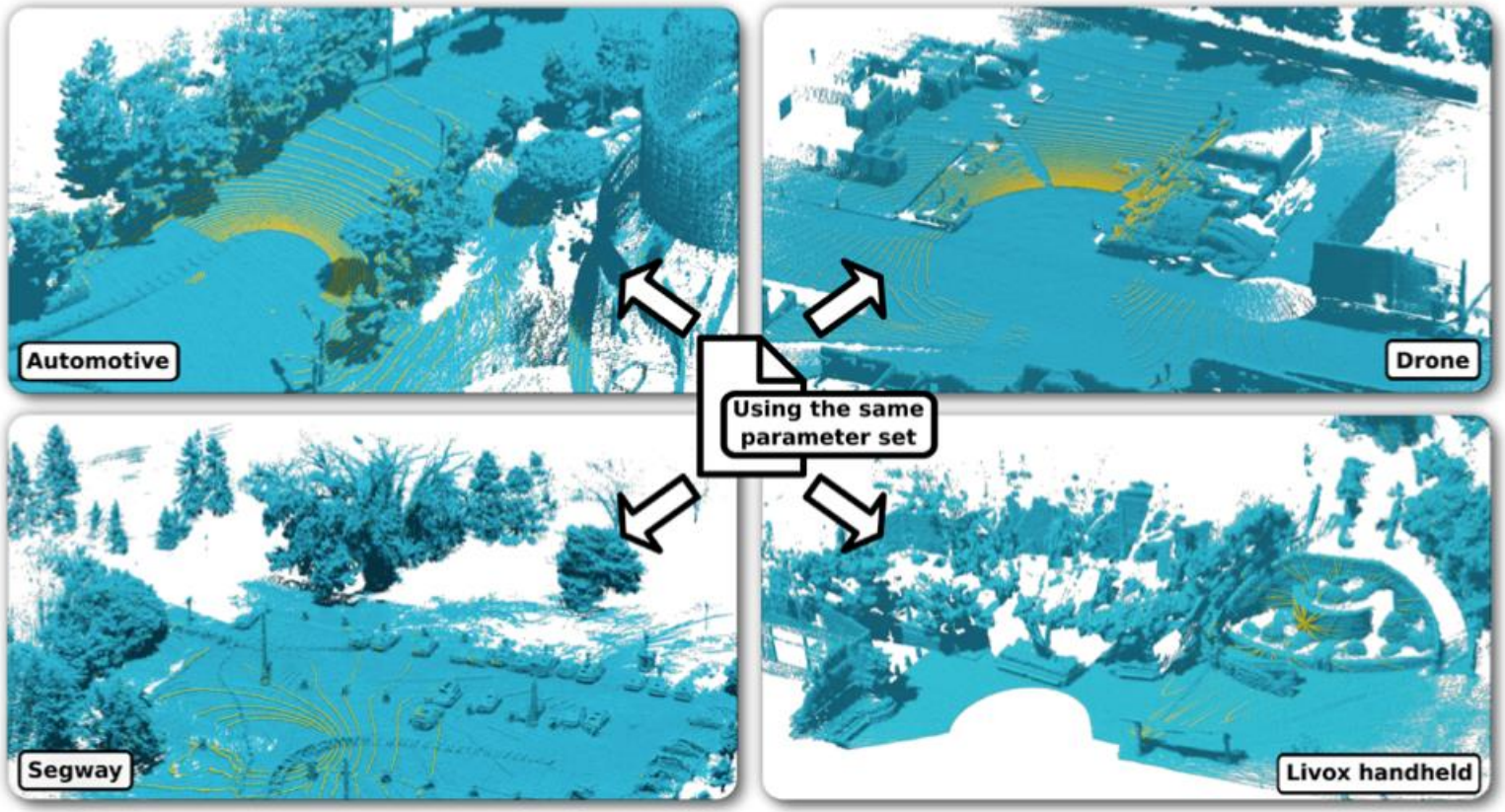


Fig. 1: Point cloud maps (blue) generated by our proposed odometry pipeline on different datasets with the same set of parameters. We depict the latest scan in yellow. The scans are recorded using different sensors with different point densities, different orientations, and different shooting patterns. The automotive example stems from the MulRan dataset [15]. The drone of the Voxgraph dataset [23] and the segway robot used in the NCLT dataset [5] show a high acceleration motion profile. The handheld Livox LiDAR [17] has a completely different shooting pattern than the commonly used rotating mechanical LiDAR.

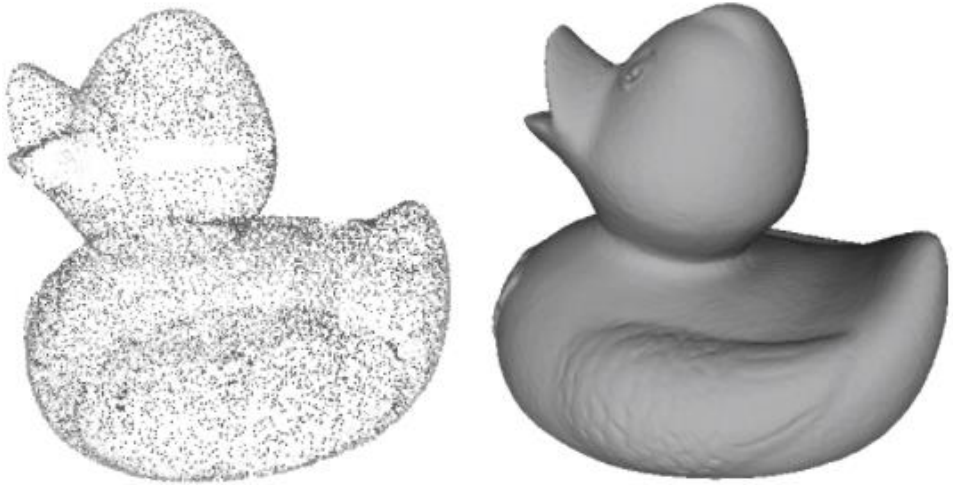
Notes

- **KISS-ICP** – non-DL based (fast!)
- Uses point-to-point ICP as a basis – no learning, no pose graph optimization, no loop closure.
- Shows that a well-tuned and well-defined point-to-point ICP can outperform other advanced (non-DL) ICP.
- Interestingly shows to perform across platforms with minimal (or no) tuning.

3D Data Representations

Point Cloud/Mesh

- ✓ Raw format/Efficiency
- • Explicit representation
- × Unorganized/Unordered



https://elmoatazbill.users.greyc.fr/point_cloud/reconstruction.png

Voxel

- • Implicit representation
- × Resolution/Scalability
- × Discretization artifact

https://www.planetminecraft.com/project/giant-snowman-1638162/

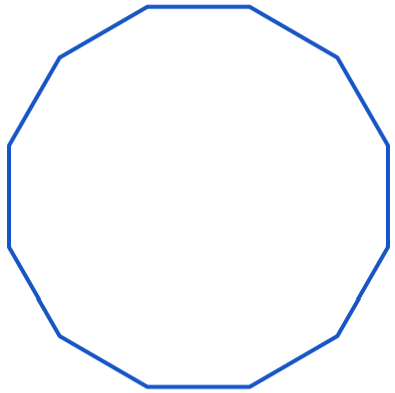
Primitives

- ✓ 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, …



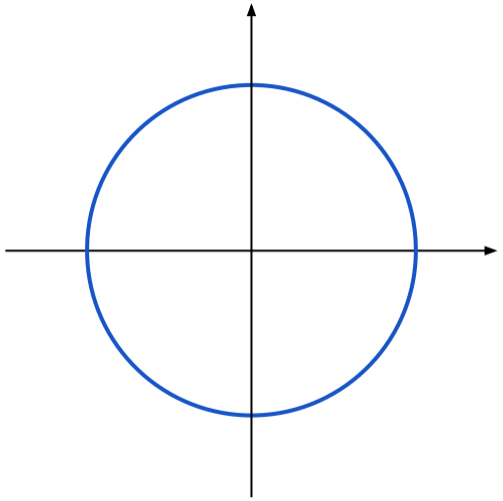
Vertices: [(x0, y0, z0), (x1, y1, z1), …, (xn, yn, zn)]

Indices: [(i0, i1), (i2, i3), …, (in-1, in)]

Slides modified from Tanner Schmidt

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**

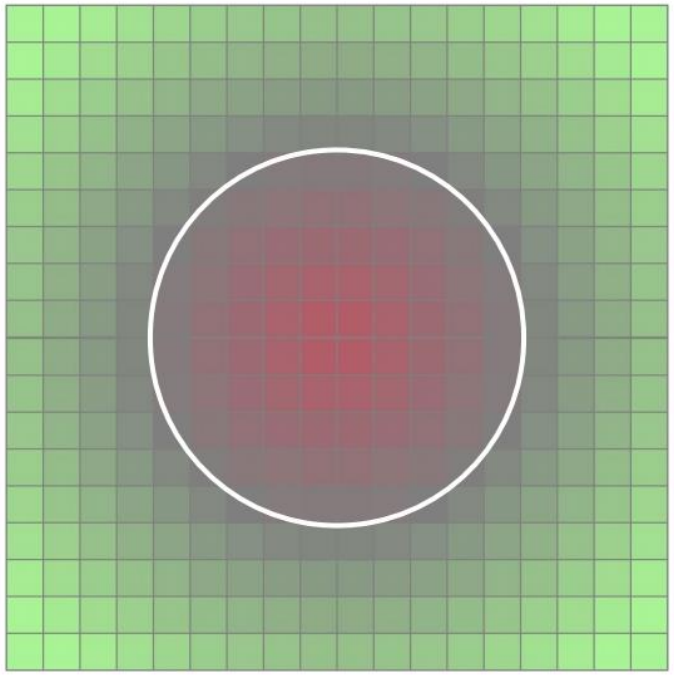


$$f(x, y) = x^2 + y^2 - r^2$$

Slides modified from Tanner Schmidt

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)**



Example

SDF of a circle centered at origin, with radius of 1 from before, we'll have the following SDF and sample values

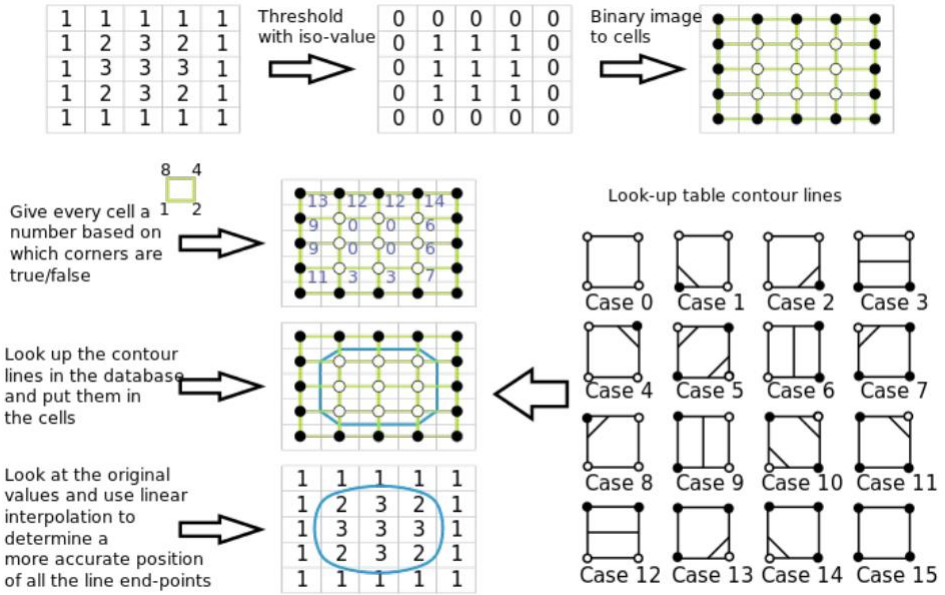
$$SDF = \sqrt{x^2 + y^x} - 1$$

$$f(1,0) = 0$$
$$f(0,2) = 3$$
$$f(0,0) = -1$$
$$f(0.5,0) = -0.5$$

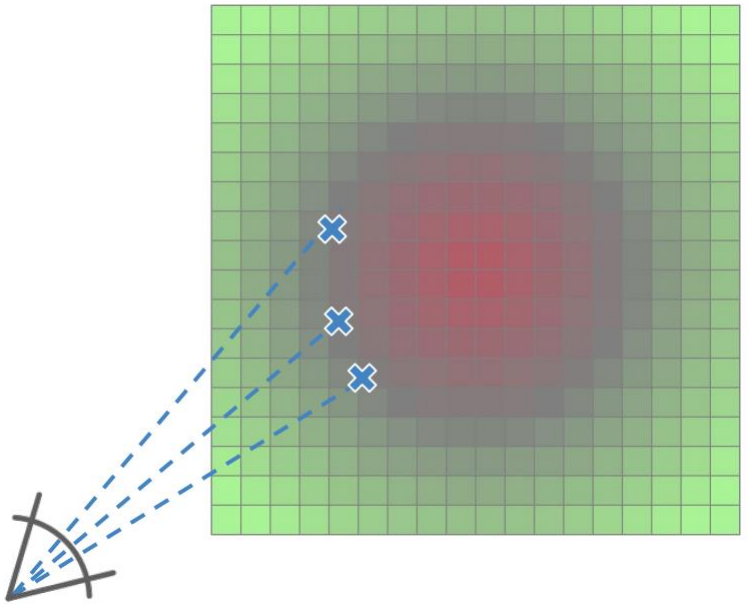
Slides modified from Tanner Schmidt

Implicit-to-Explicit Conversions

Marching Squares (2D)



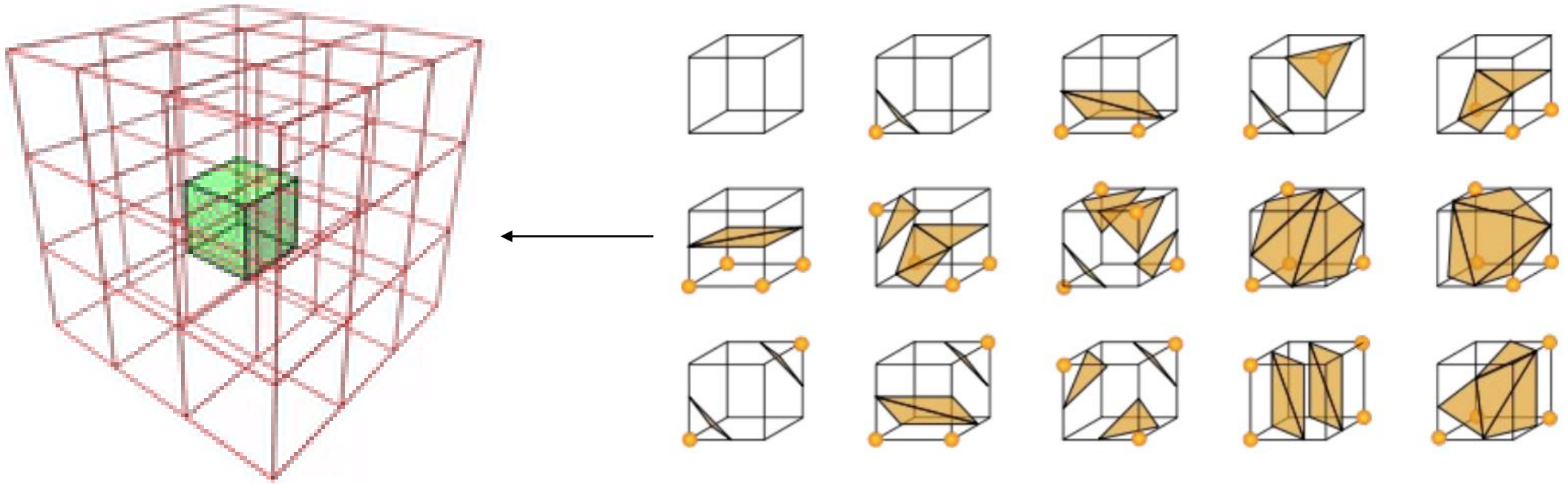
Raycasting for partial conversion



Slides modified from Tanner Schmidt

Implicit-to-Explicit Conversions

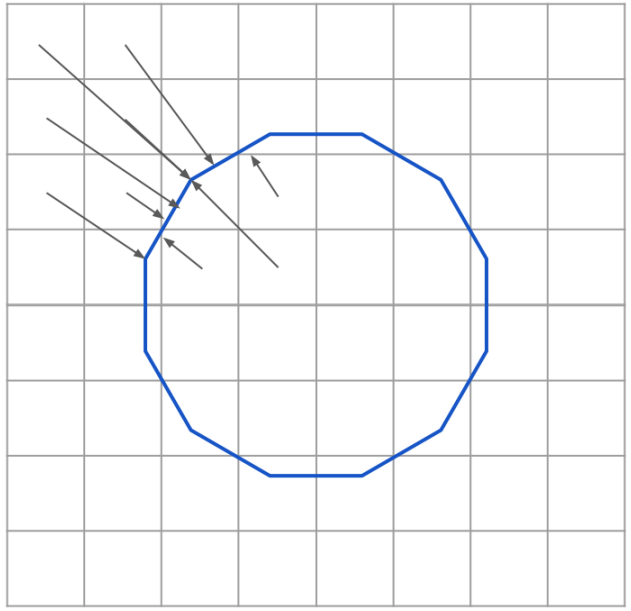
Surface Reconstruction via Marching Cubes (3D)



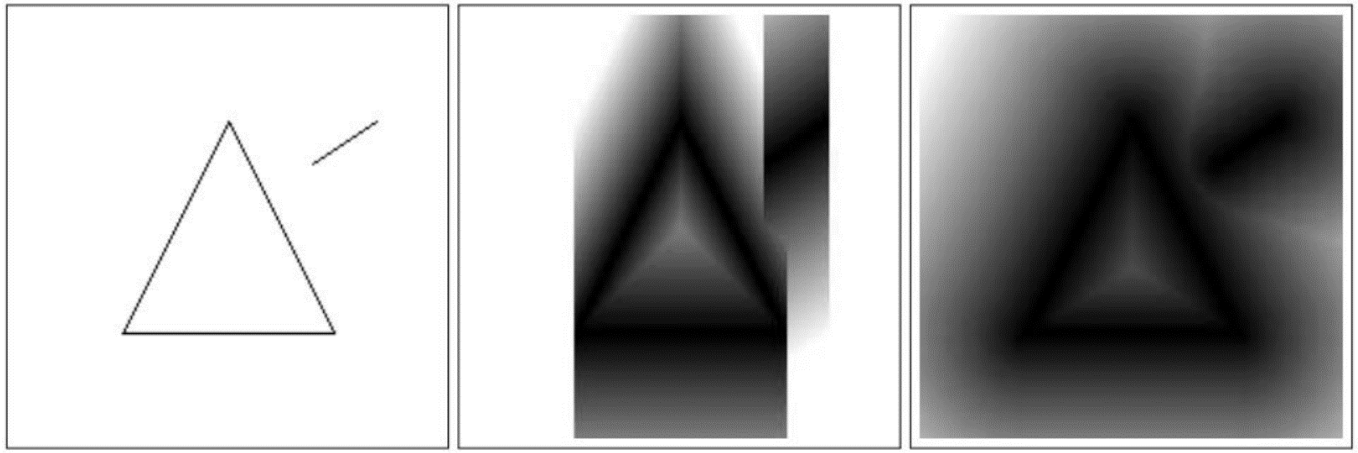
From Algorithms Visualized - https://www.youtube.com/watch?app=desktop&v=B_xk71YopsA and Tanner Schmidt

Explicit-to-Implicit Conversions

By finding closest points to the surface



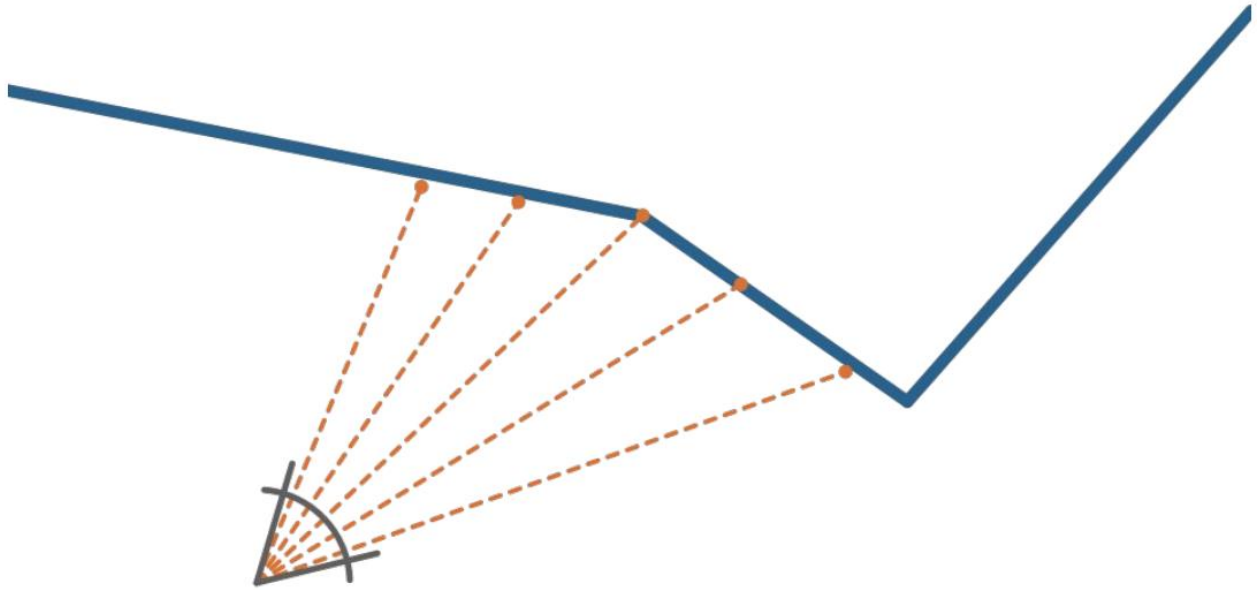
Distance Transform (2D)



Slides modified from Tanner Schmidt

Projective SDF and Truncated SDF (TSDF)

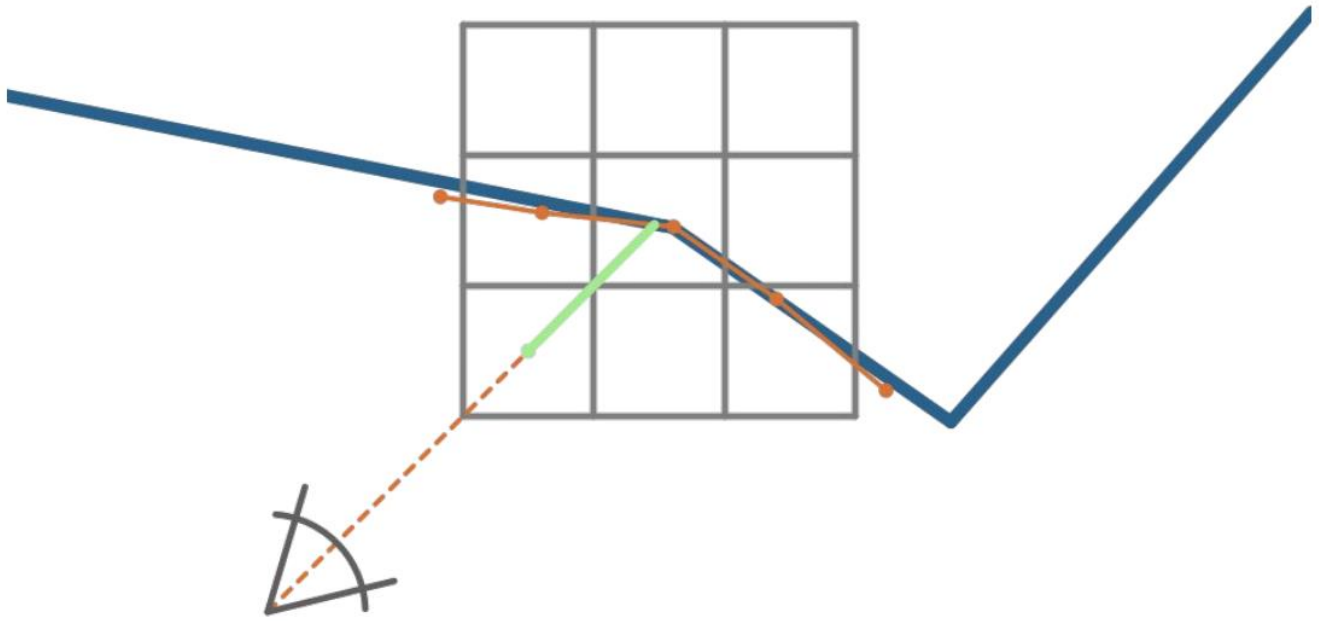
- Computing SDF requires a closed surface
- **What if I have only partial observation of the surface?**



Slides modified from Tanner Schmidt

Projective SDF and Truncated SDF (TSDF)

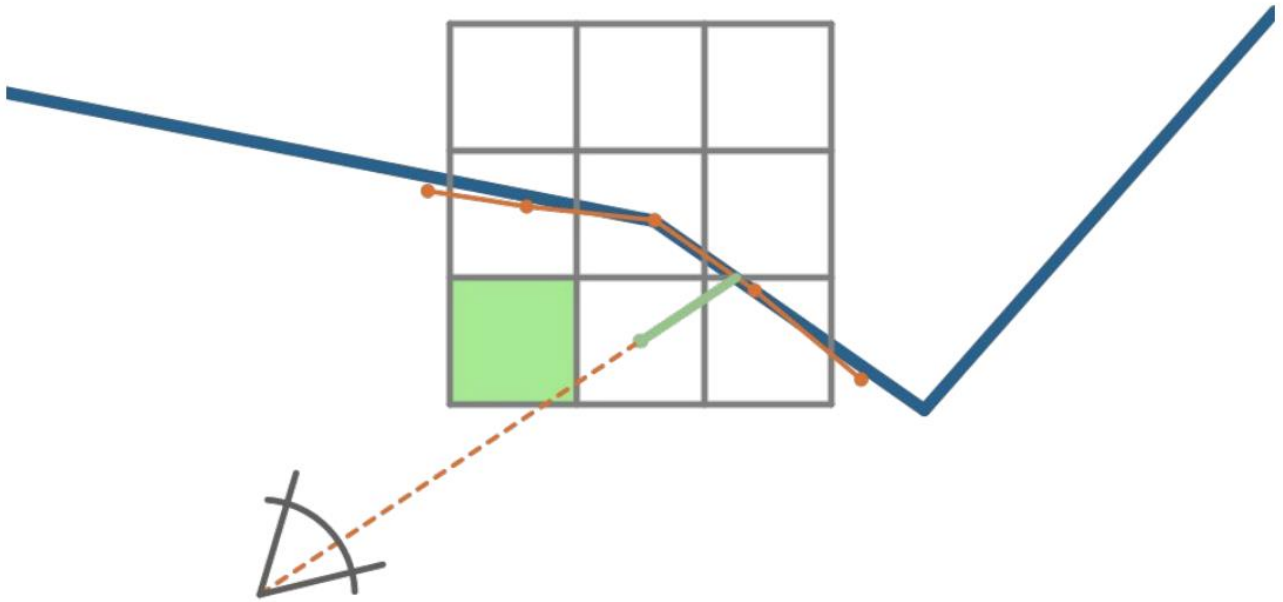
- What if I have only partial observation of the surface?
- We can define projective SDF:



Slides modified from Tanner Schmidt

Projective SDF and Truncated SDF (TSDF)

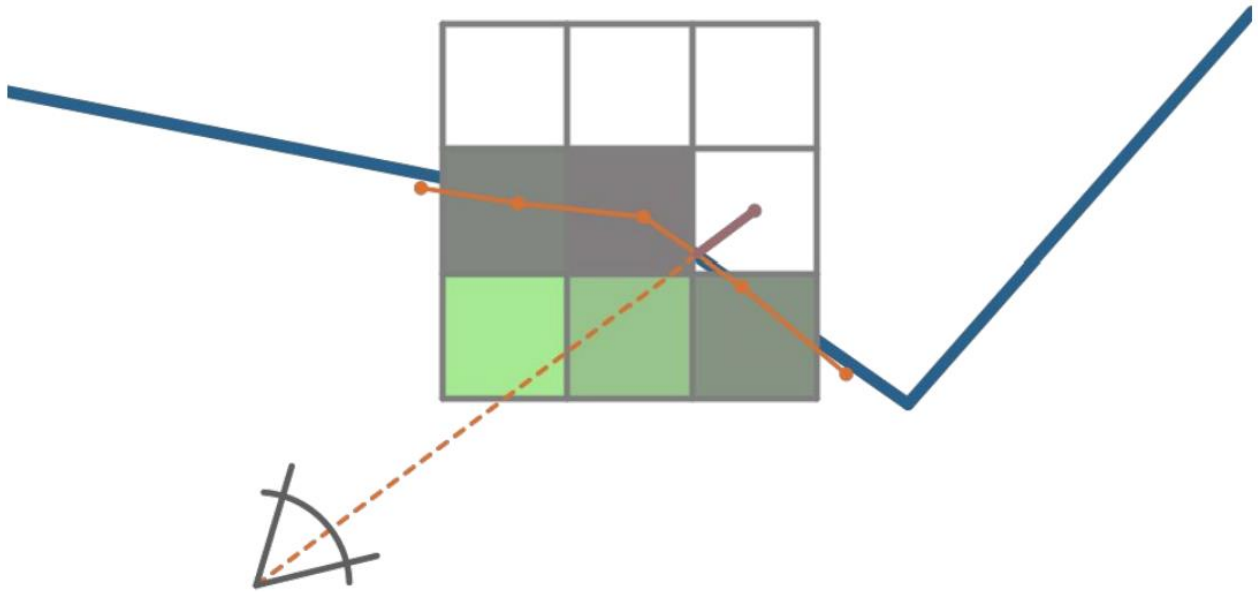
- What if I have only partial observation of the surface?
- We can define projective SDF:



Slides modified from Tanner Schmidt

Projective SDF and Truncated SDF (TSDF)

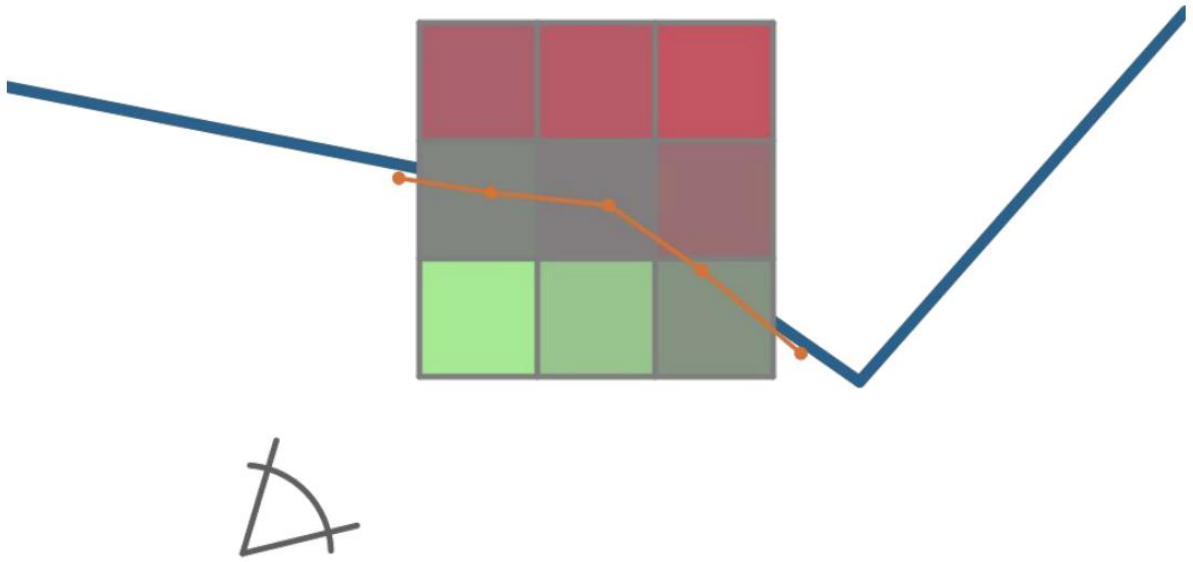
- What if I have only partial observation of the surface?
- We can define projective SDF:



Slides modified from Tanner Schmidt

Projective SDF and Truncated SDF (TSDF)

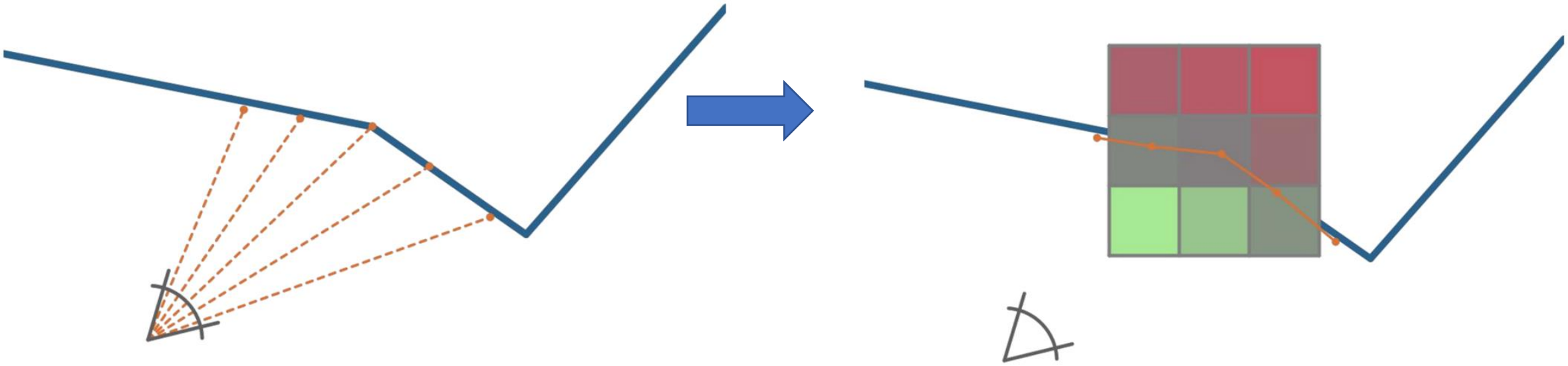
- What if I have only partial observation of the surface?
- Since we only compute the distance near both sides of the surface, we call this projective SDF as a **Truncated SDF**, or **TSDF**.



Slides modified from Tanner Schmidt

Projective SDF and Truncated SDF (TSDF)

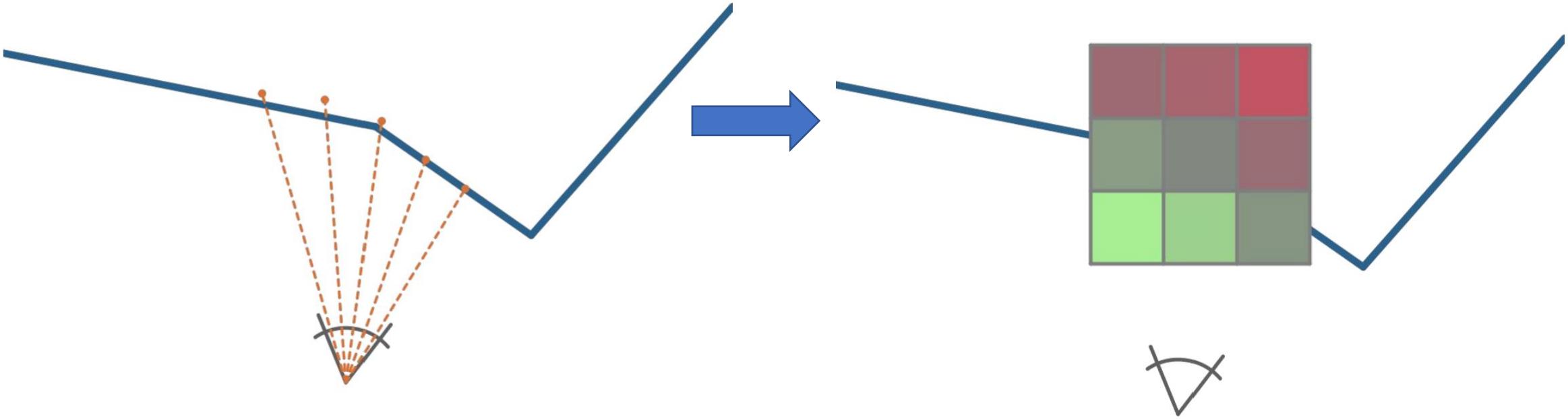
- TSDF is view dependent



Slides modified from Tanner Schmidt

Projective SDF and Truncated SDF (TSDF)

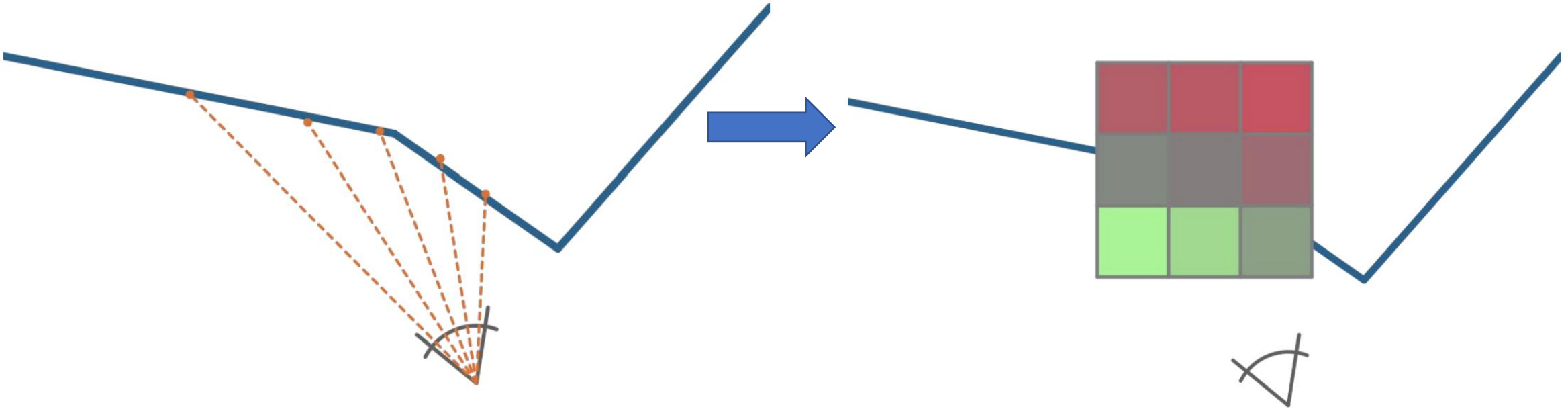
- TSDF is view dependent



Slides modified from Tanner Schmidt

Projective SDF and Truncated SDF (TSDF)

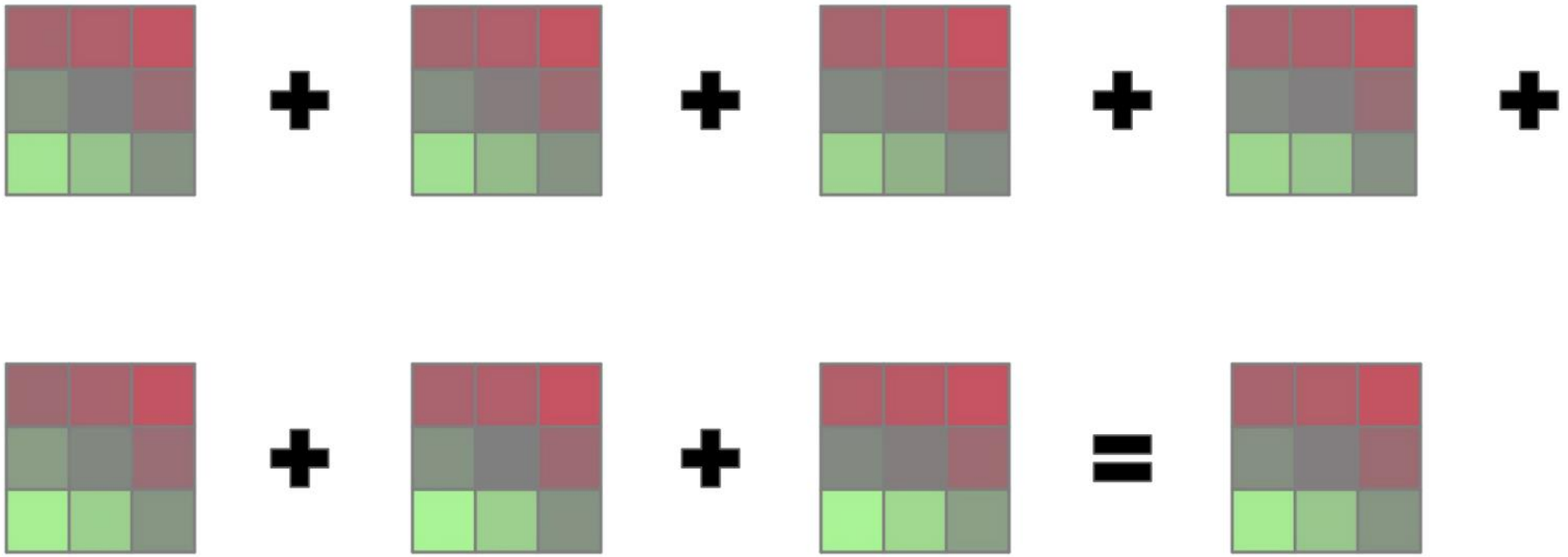
- TSDF is view dependent



Slides modified from Tanner Schmidt

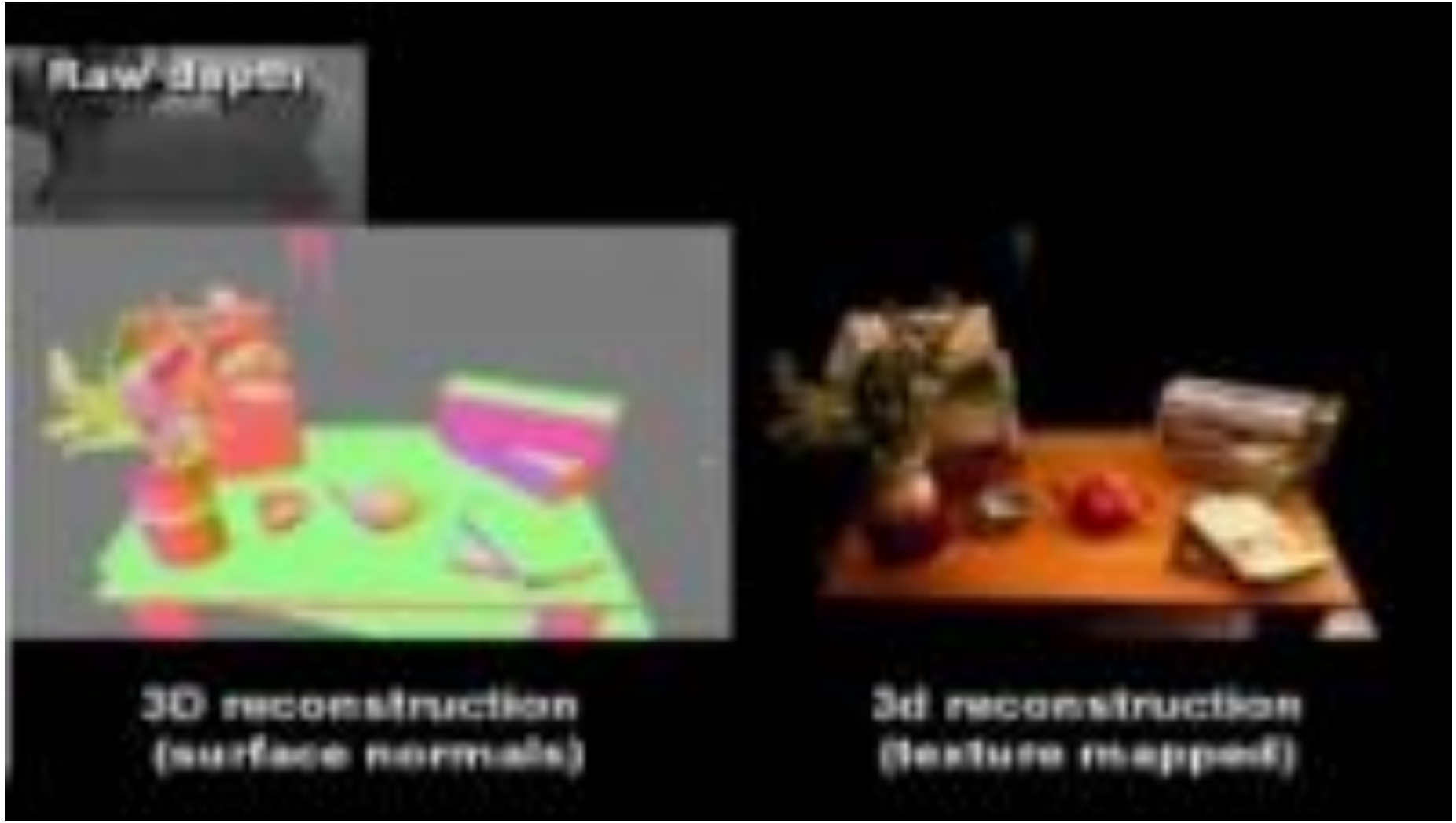
Projective SDF and Truncated SDF (TSDF)

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



Slides modified from Tanner Schmidt

Application: KinectFusion

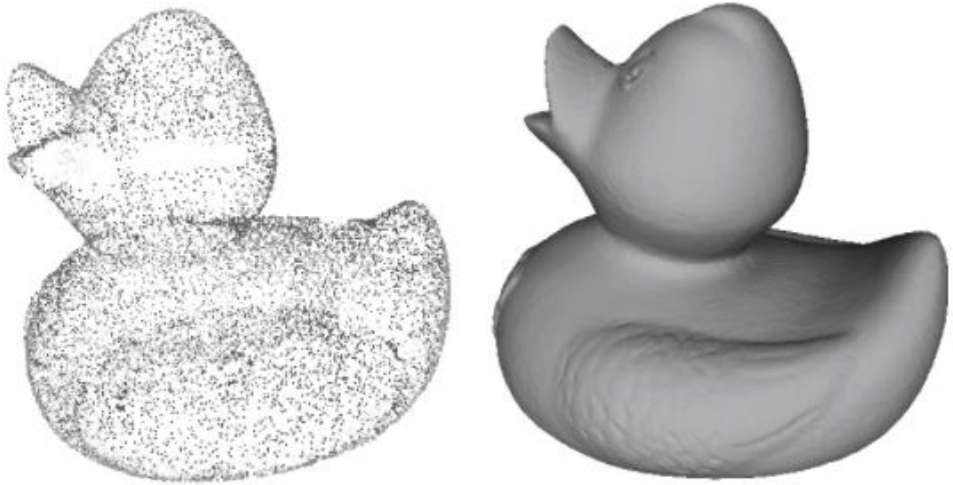


https://youtu.be/KOUSSlKUJ-A

3D Data Representations

Point Cloud/Mesh

- ✓ Raw format/Efficiency
- • Explicit representation
- × Unorganized/Unordered



https://elmoatazbill.users.greyc.fr/point_cloud/reconstruction.png

Voxel

- • Implicit representation
- × Resolution/Scalability
- × Discretization artifact

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[1)], Yuichi Taguchi[2)], and Vineet R. Kamat[1)]

1) University of Michigan, USA

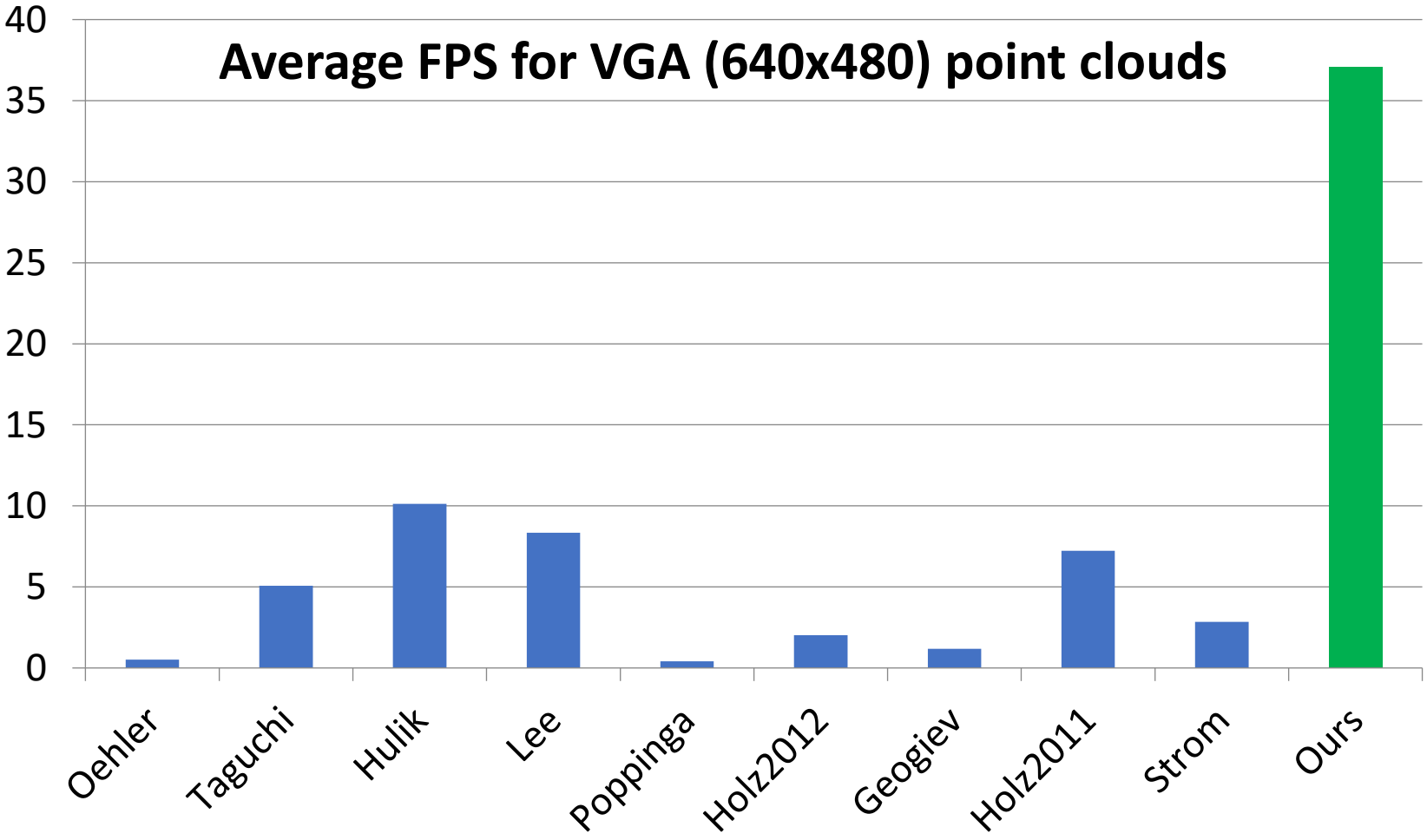
2) Mitsubishi Electric Research Labs, USA

June 4, 2014

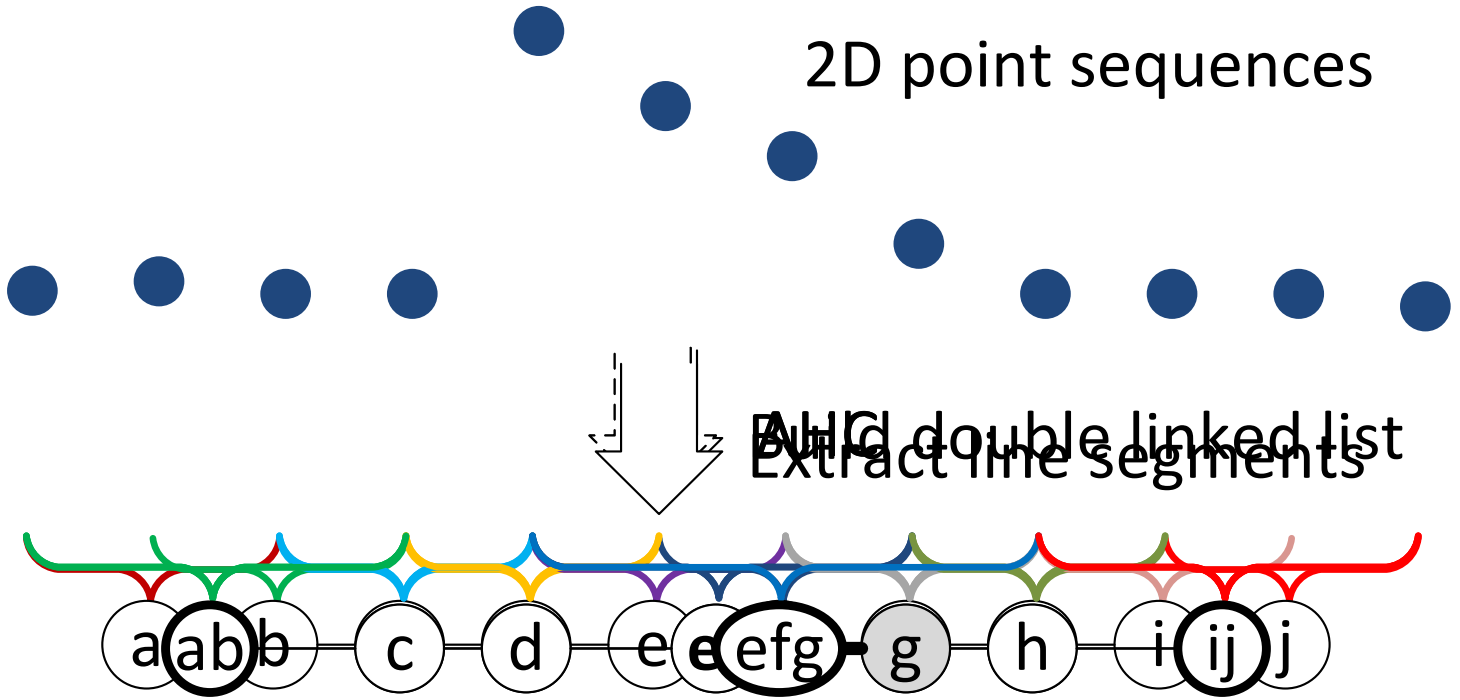
- Real-time plane extraction is crucial to various applications in robotics, computer vision, and 3D modeling:
  - Table-top object manipulation
  - Landmarks for SLAM
  - Compact and semantic scene modeling
- We present an efficient and reliable fast plane extraction algorithm applicable to organized 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)

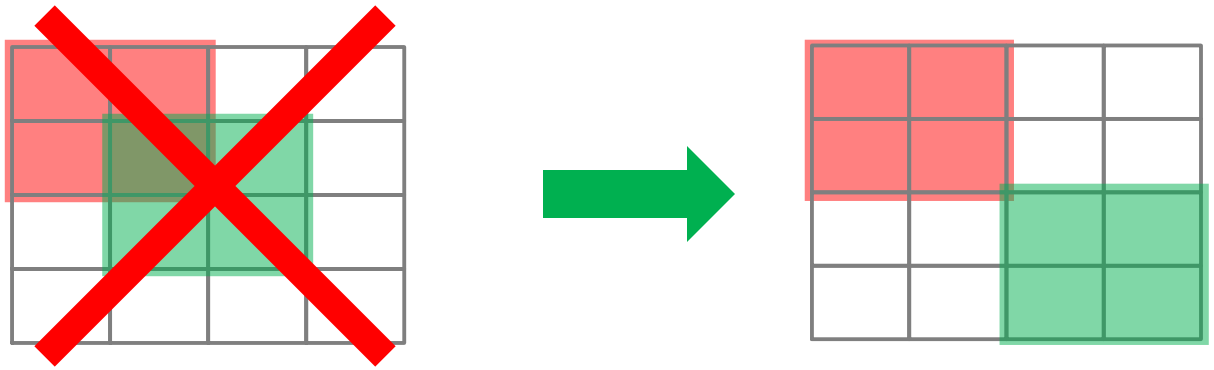
- Previous Work



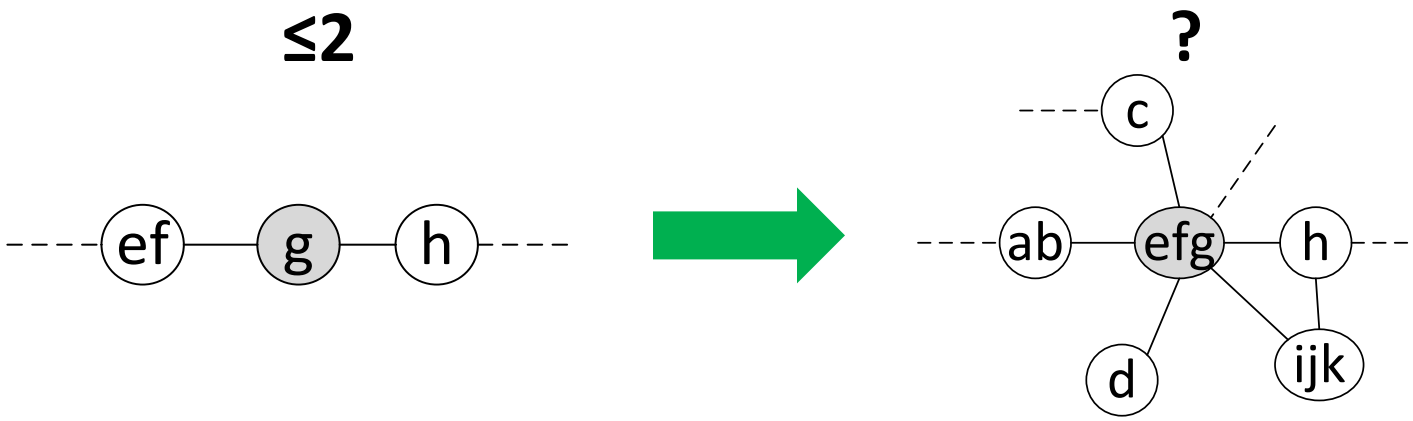
- Analogy to Line Regression (Nguyen et al. 2005; April Robotics Toolkit, 2010)
  - Exploit the neighborhood information given by the order of points



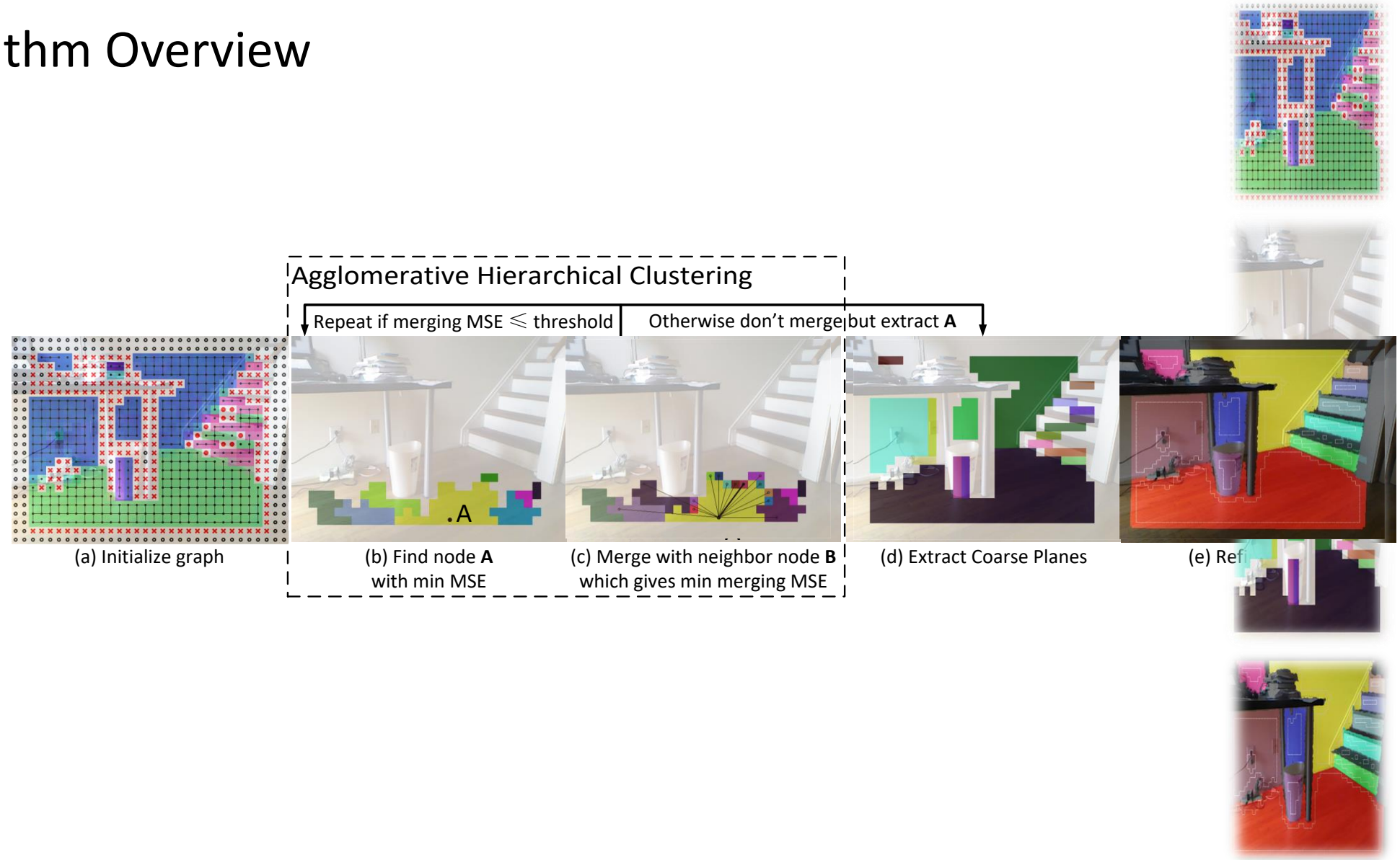
- Non-trivial Generalization to 3D
  - None-overlapping nodes



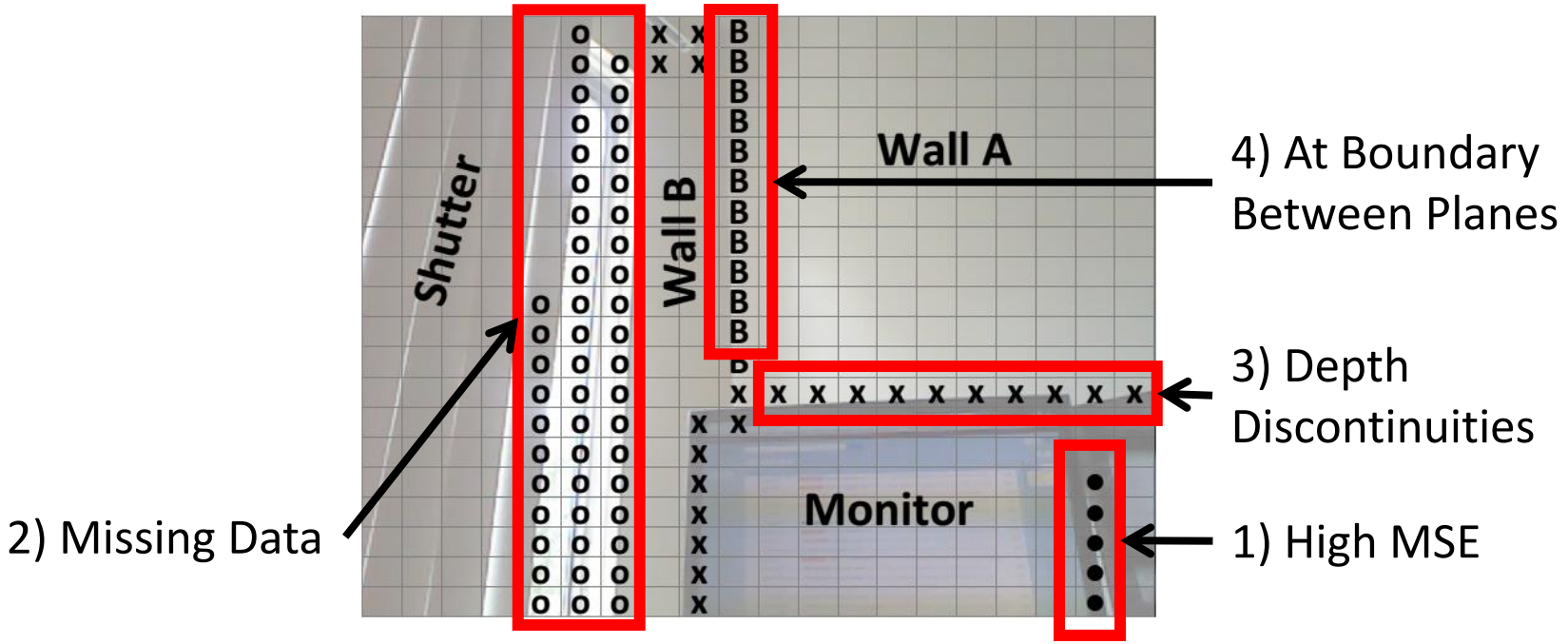
  - Number of merging attempts

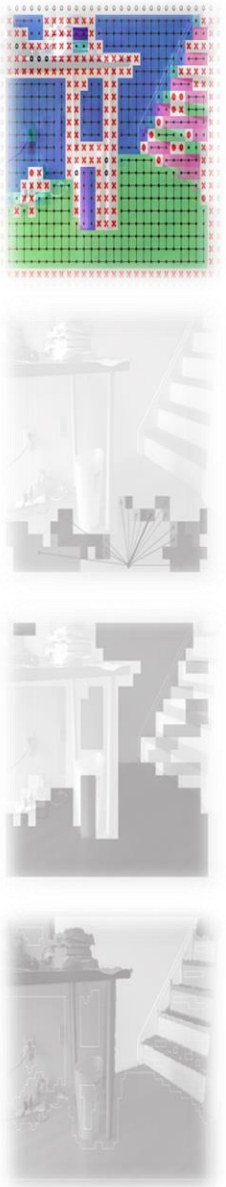


- Algorithm Overview



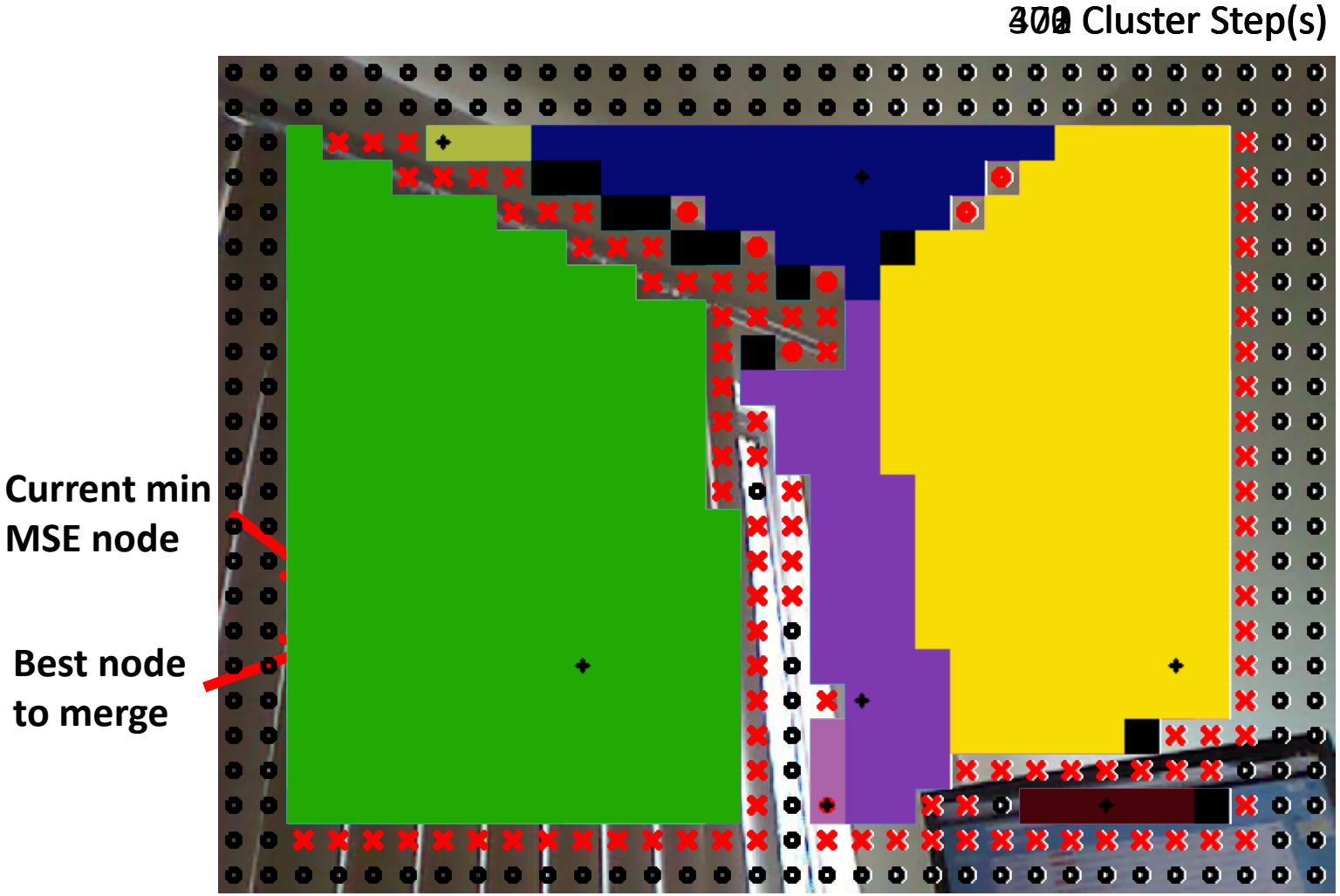
- Graph Initialization
  - Non-overlapping node initialization
  - Rejecting “bad” nodes

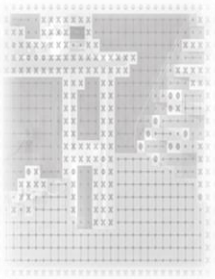




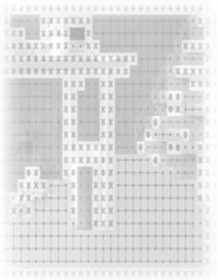
- Good! Avoid per-point normal estimation

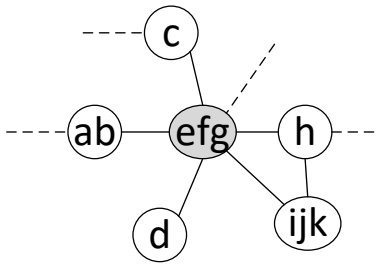
- Agglomerative Hierarchical Clustering

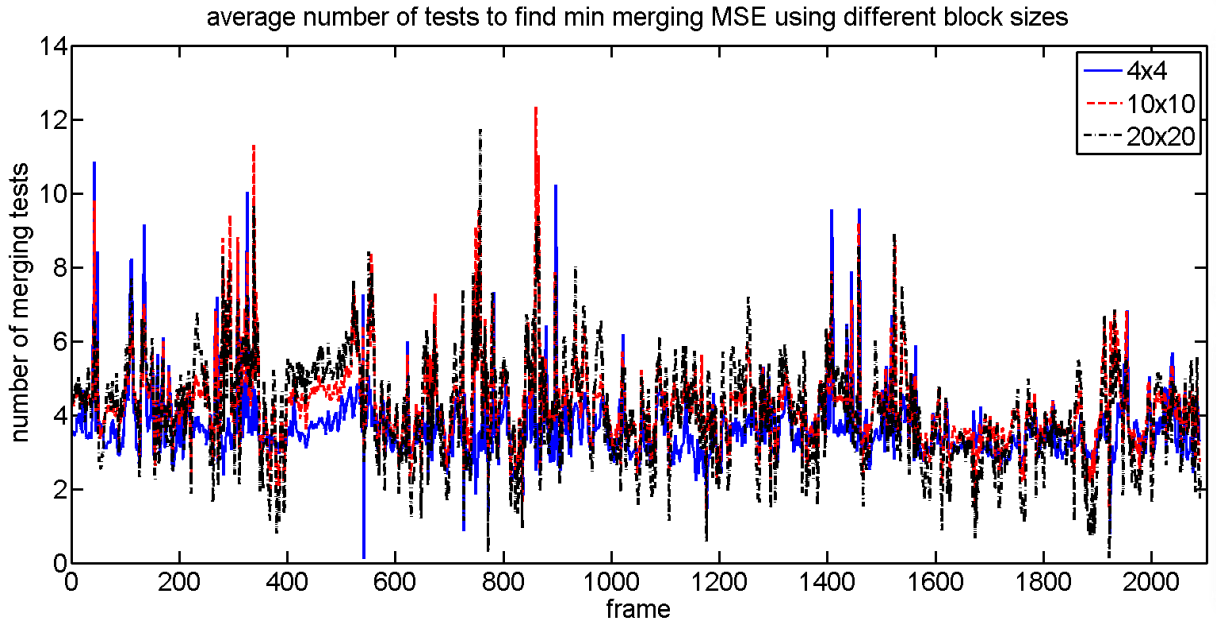




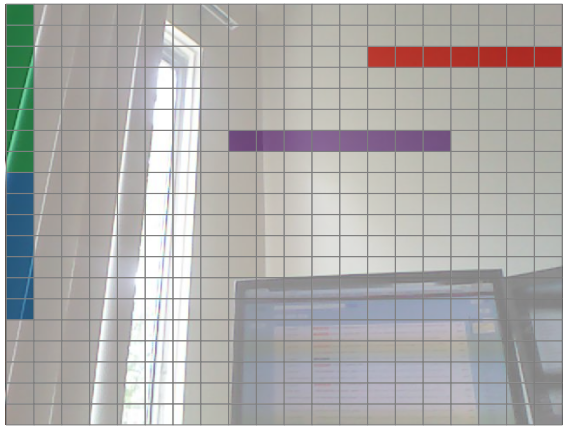

- Average Number of Merging Attempts
  - Small irrespective of initial number of nodes
  - Planar graph! Average node degree < 6
  - Merging is empirically a constant-time operation
    - O(nlogn), only arise from maintaining the min-heap

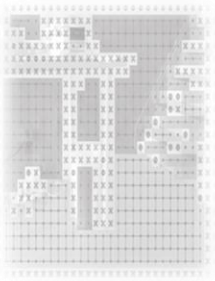




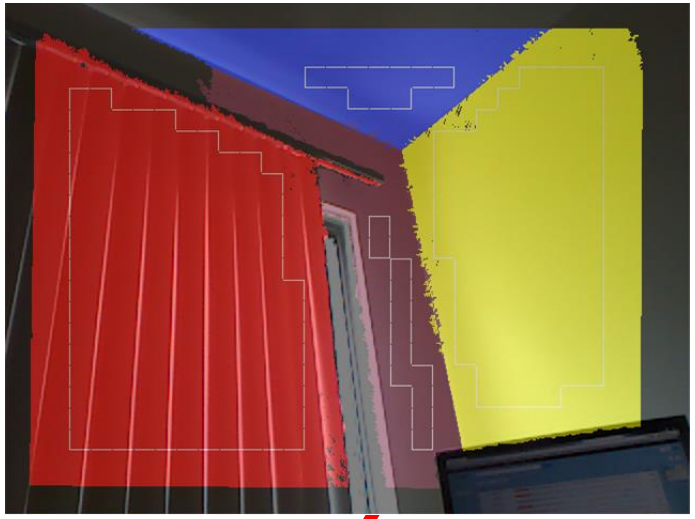


- Implementation Details
  - Disjoint set
  - Min-heap
  - Second-order statistics
  - Depth discontinuity/MSE threshold
    (Holzer et al. IROS 2012; Khoshelham & Elberink, 2012)
  - Avoid strip-like initial node shape

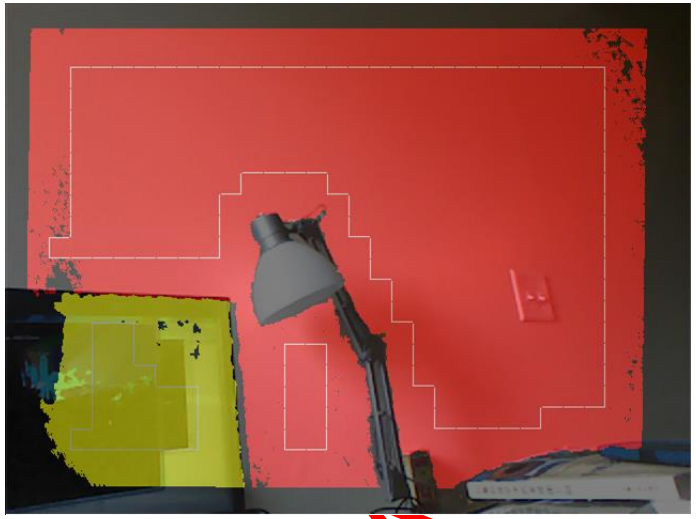




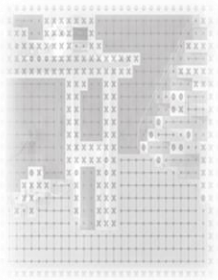
- Segmentation Refinement
  - Artifacts



Sawtooth

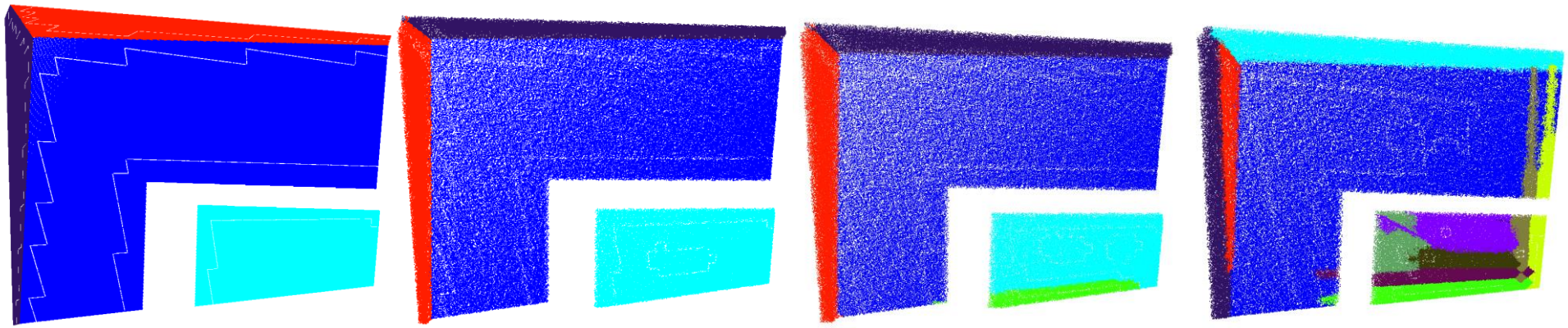


Over-segmentation



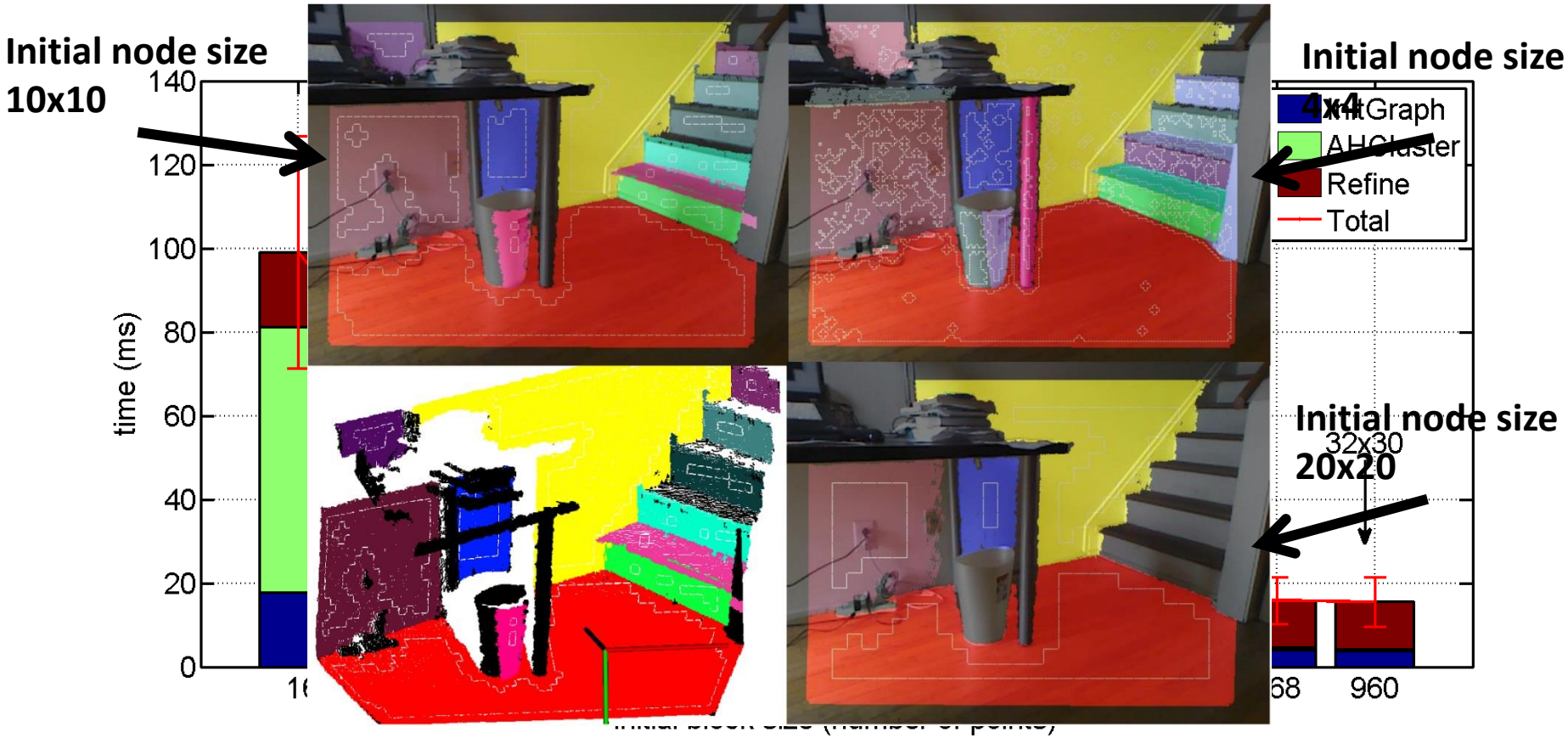
- Pixel-wise region-grow refinement
  - Only check boundary blocks and points

- Simulated Data
  - Robustness to uniformly distributed depth noise (Georgiev et al., IROS 2011)
  - Noise magnitude E = 0, 10, …, 200mm
  - Ground truth depth ranges from 1396mm to 3704mm

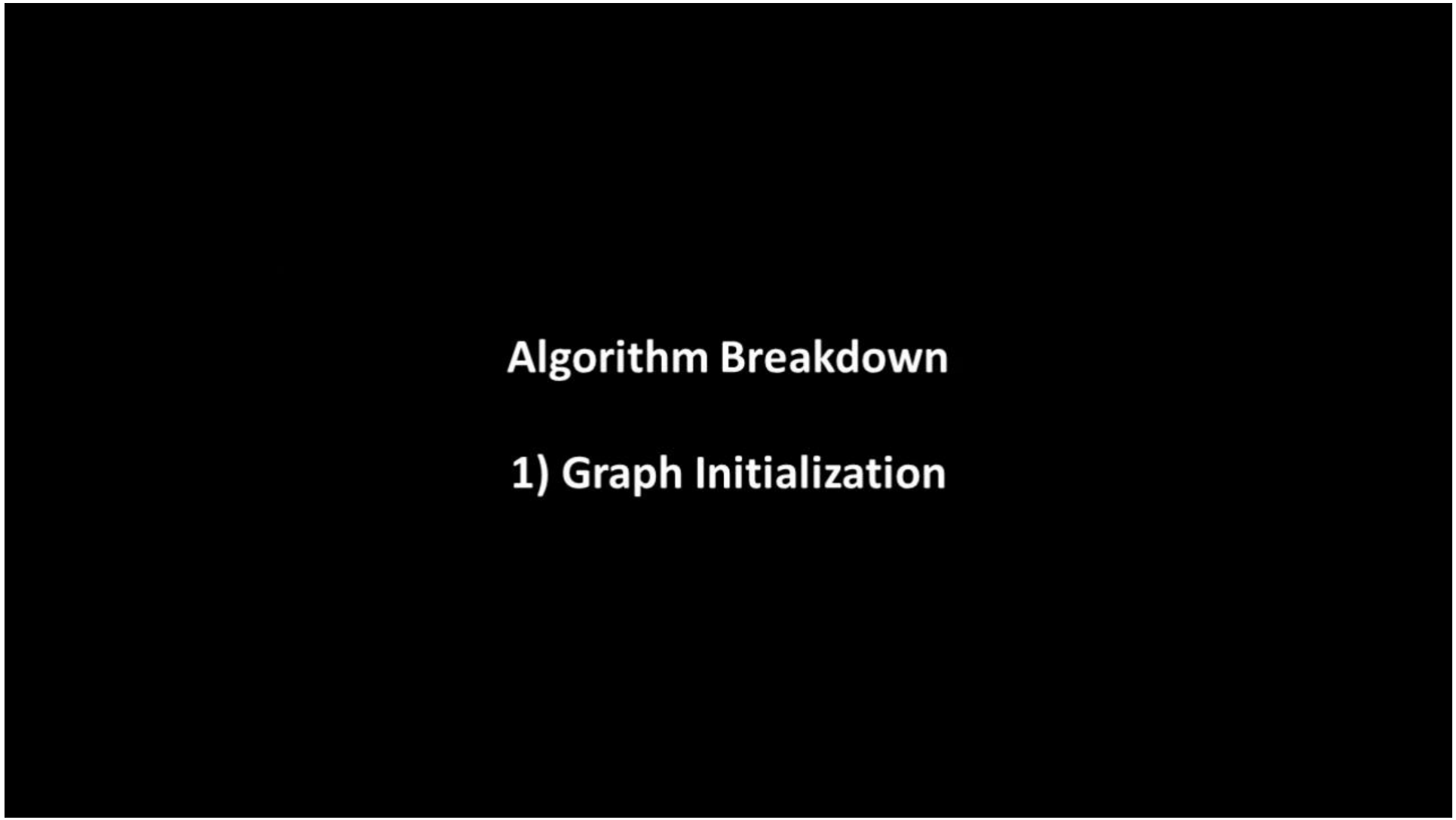


E=0 E=100 E=150 E=200

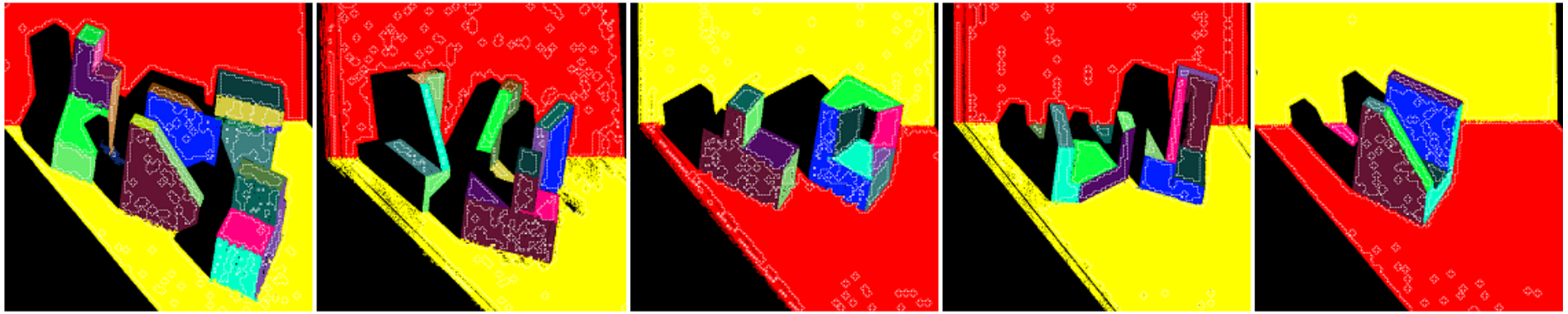
- Real-World Kinect Data
  - 2102 frames of an indoor scene
  - 640 × 480 pixel/frame



- Real-World Kinect Data

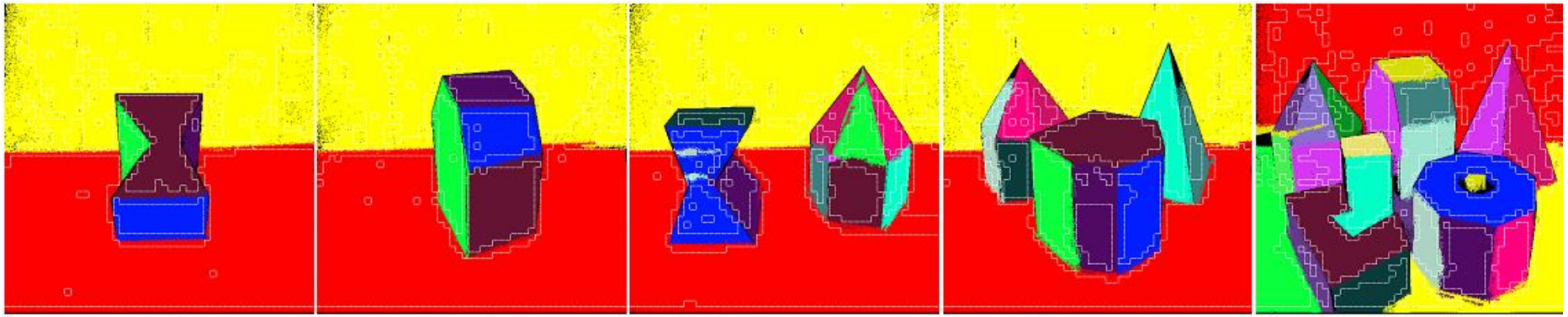


- SegComp Datasets (Hoover et al. PAMI 1996)
  - ABW-TEST



(a) 21/27, 1 under, 4 misses, 3 noises (b) 18/19, 1 miss (c) 14/17, 3 misses (d) 15/17, 2 misses (e) 10/10

  - PERCEPTRON-TEST



(a) 6/6 (b) 6/6 (c) 10/11, 1 over, 1 noise (d) 12/13, 1 miss, 1 noise (e) 20/30, 1 over, 1 under, 7 misses, 3 noises

- 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
Ours	**15.2**	**12.8 (84.2%)**	**1.7**	**0.1**	**0.0**	**2.4**	**0.7**
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
Ours	**14.6**	**8.9 (60.9%)**	**2.4**	**0.2**	**0.2**	**5.1**	**2.1**

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/
  - https://demuc.de/tutorials/cvpr2017/