Techniques for analysis and visualization of large-scale LIDAR scan data of urban environments

Dr. Patrick Flynn and Alexandri Zavodny

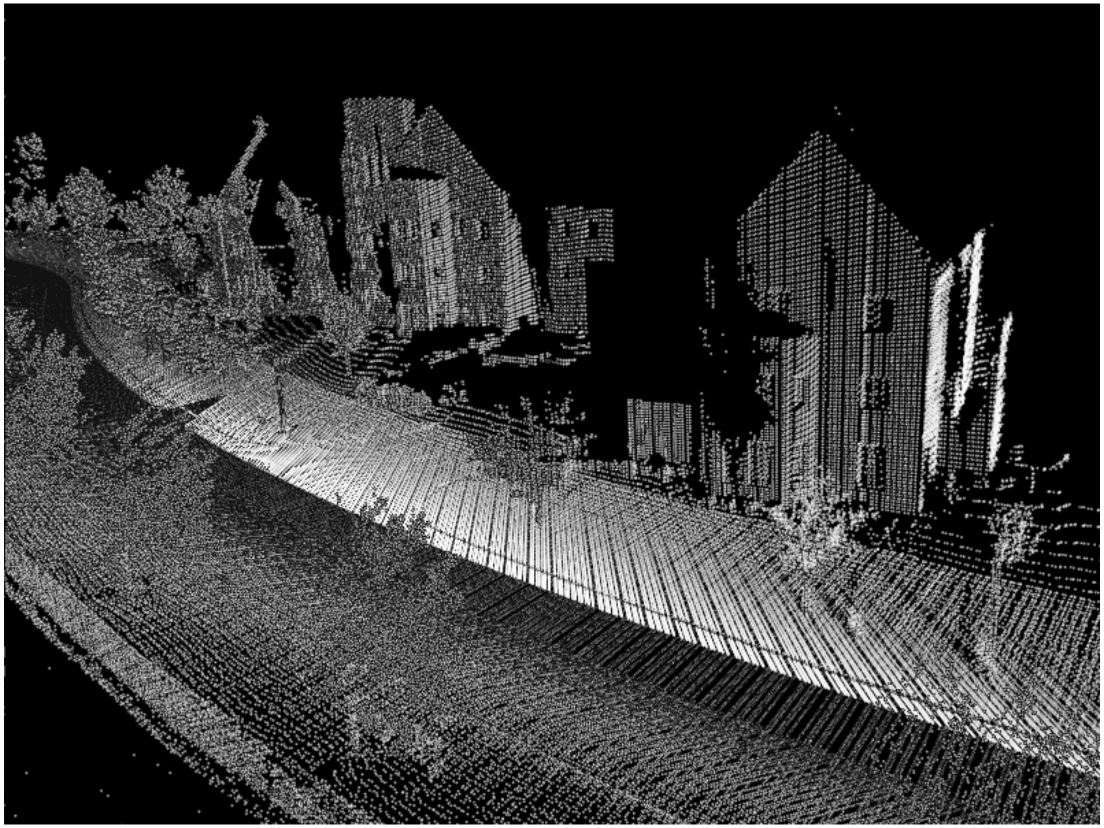
flynn@nd.edu and azavodny@nd.edu

University of Notre Dame

This research sponsored by a gift from the NAVTEQ Corporation.

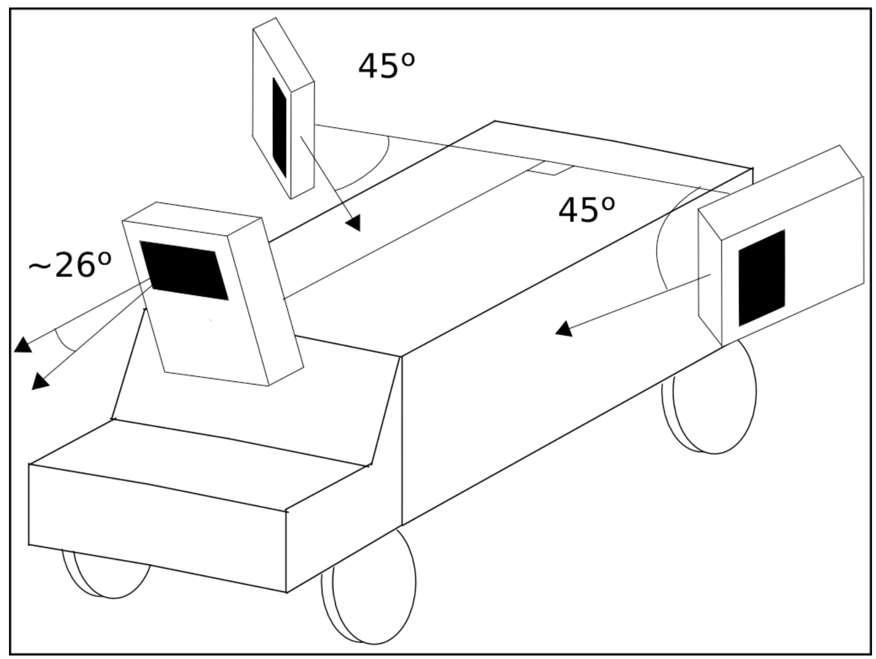
Introduction

- What is LIDAR?

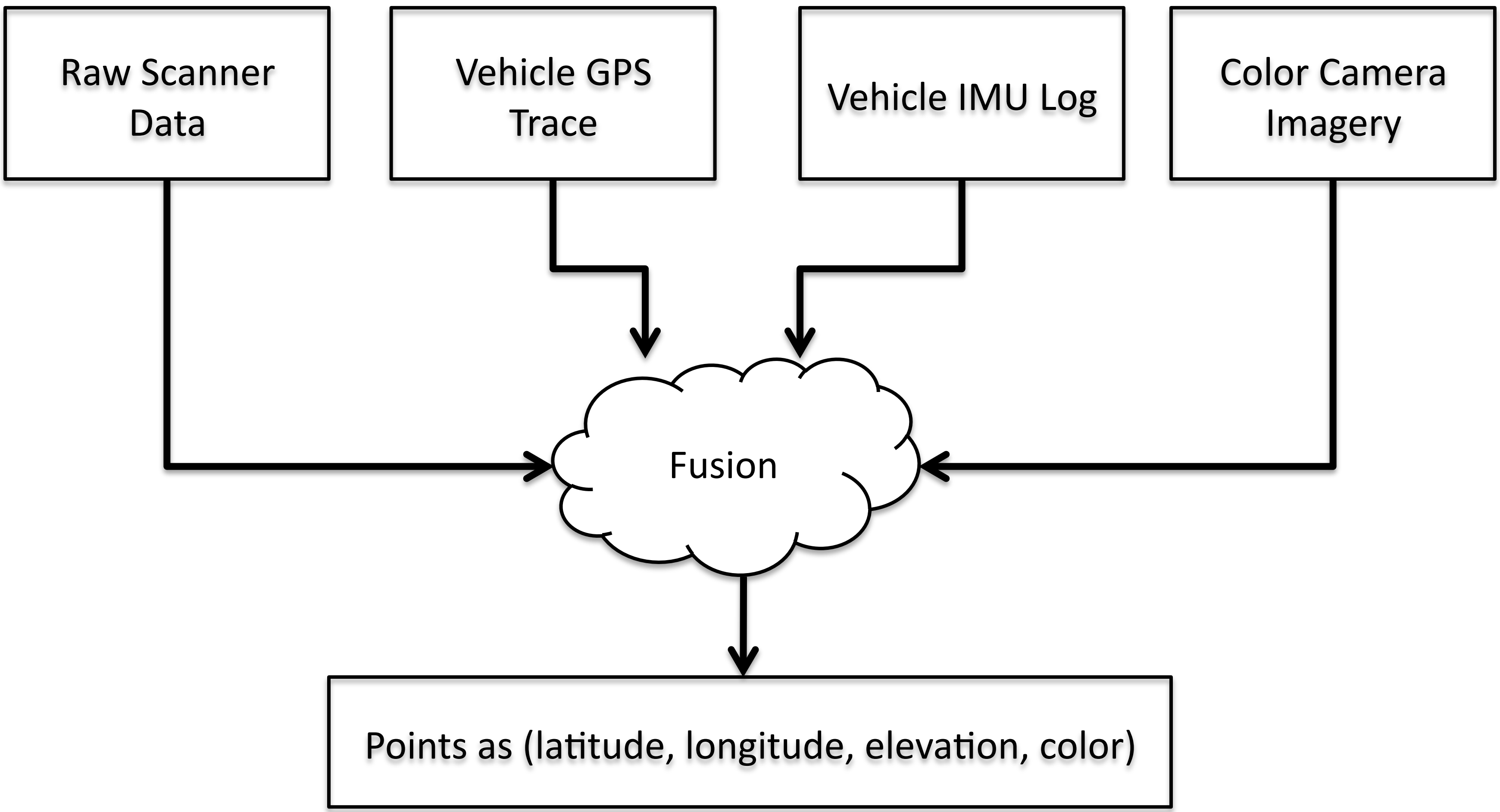


Acquisition System

- Typically attach multiple scanners to cover a larger range
- Track vehicle's location with GPS and IMUs
- Obtain color with multiple mounted cameras

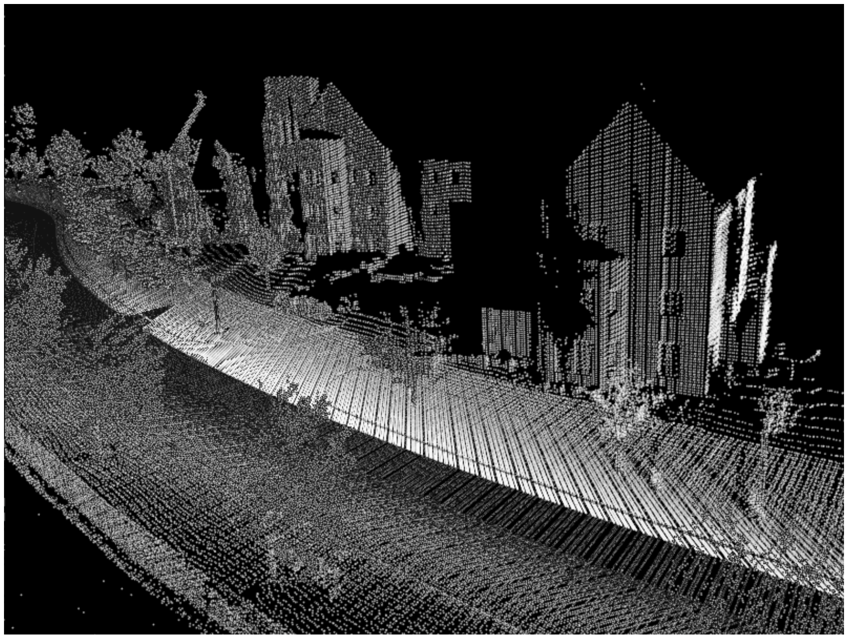


Data Fusion


Raw Scanner Data
Vehicle GPS Trace
Vehicle IMU Log
Color Camera Imagery
Fusion
Points as (latitude, longitude, elevation, color)

Data Acquisition

- Different scanners provide very different data:

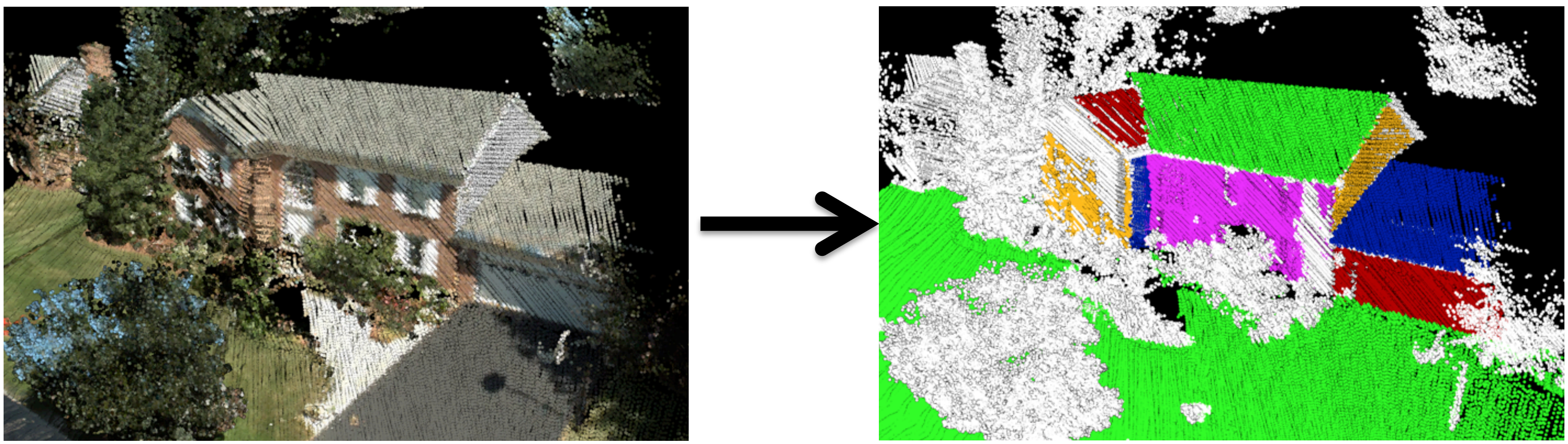


Acquisition Details

- SICK scanners:
  - High precision (10mm)
  - Low density (27,000 points/sec max. per scanner)
  - Total campus: 95,809,671 points (from all 3)
- Velodyne scanner:
  - Lower precision (2cm*)
  - Very high density (1.5 million points/sec)
  - Total campus: 3,157,484,707 points

Region Extraction

- Want to identify underlying surface models from point sampling



- Look for sets of points that lie in the same plane

Region Extraction – Related Work

- Popular research topic for digital archaeology[1-3], aerial land surveys[4-6], urban planning applications, and more[7,8].

- However, most papers rely on *structured* scan data
  - Stationary platforms[3,4,7] and single-laser mobile platforms[9] can exploit structure

Planar Region Extraction

- We utilize a **region-growing** approach
- Our approach consists of 4 main steps:
  - Computation of ‘planarity measures’
  - Region growing
  - Region combination
  - Region pruning

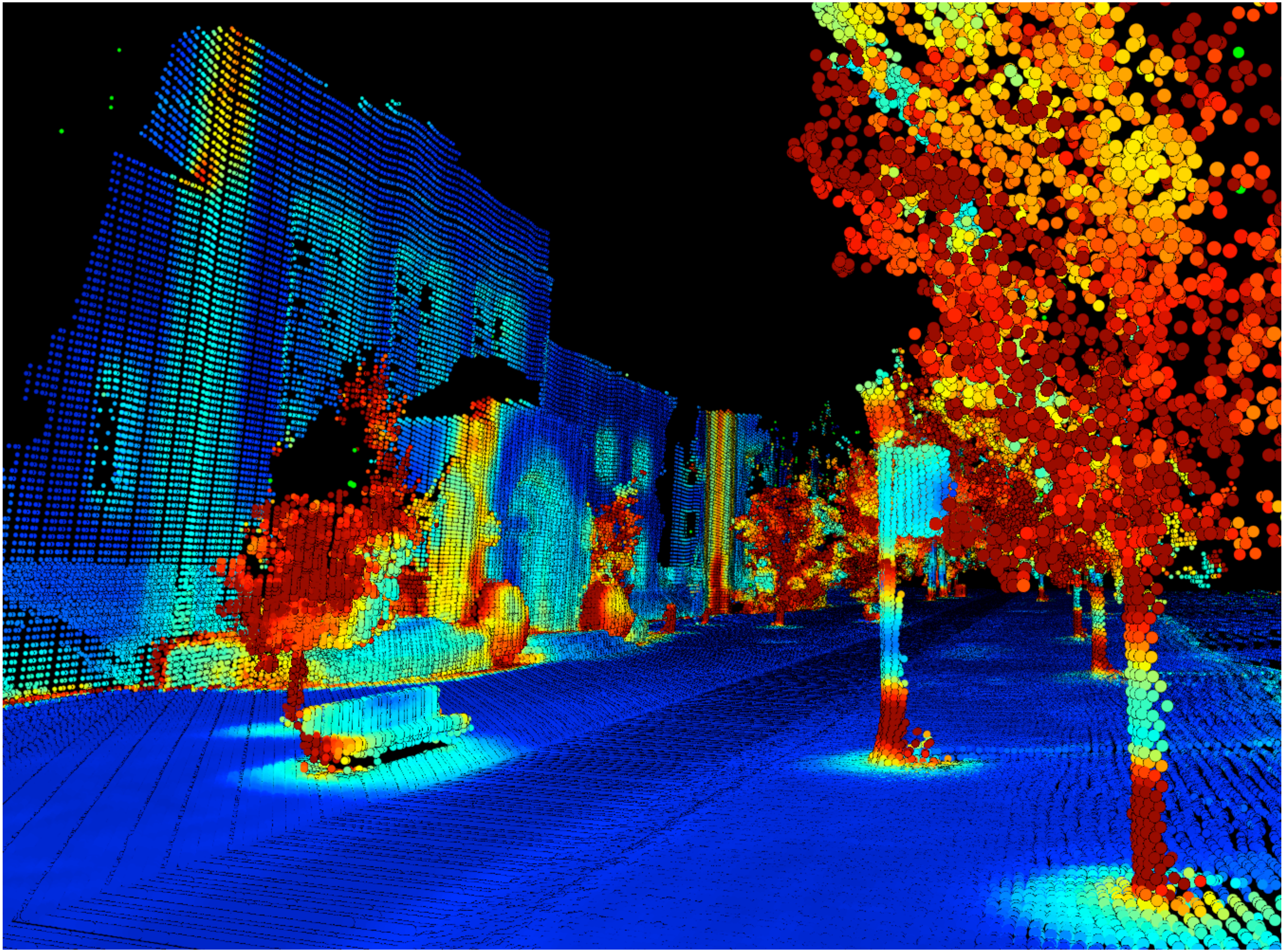
Planar Region Extraction: Planarity Measures

- For speed and accuracy, only grow regions from points with planar neighborhoods
- Establish 'planarity measures' for each point in two passes:
- Pass 1:
  - Compute surface normal approximation N_i for each point

Planar Region Extraction: Planarity Measures

- Pass 2:
  - Compute the variance of normals within the neighborhood

$$PM(\overrightarrow{P_s}) = \frac{\sum_{i}^{|N|} \angle(\overrightarrow{N_{avg}}, \overrightarrow{N_i})^2}{|N|}$$



Planar Region Extraction: Region Growing

- Given a seed point $\vec{P_s}$, fit a plane to the point's neighborhood
- Iteratively, add points near the edge of the region that fit this planar equation
- Re-fit plane and re-evaluate membership, but only allow plane normal to change slowly:

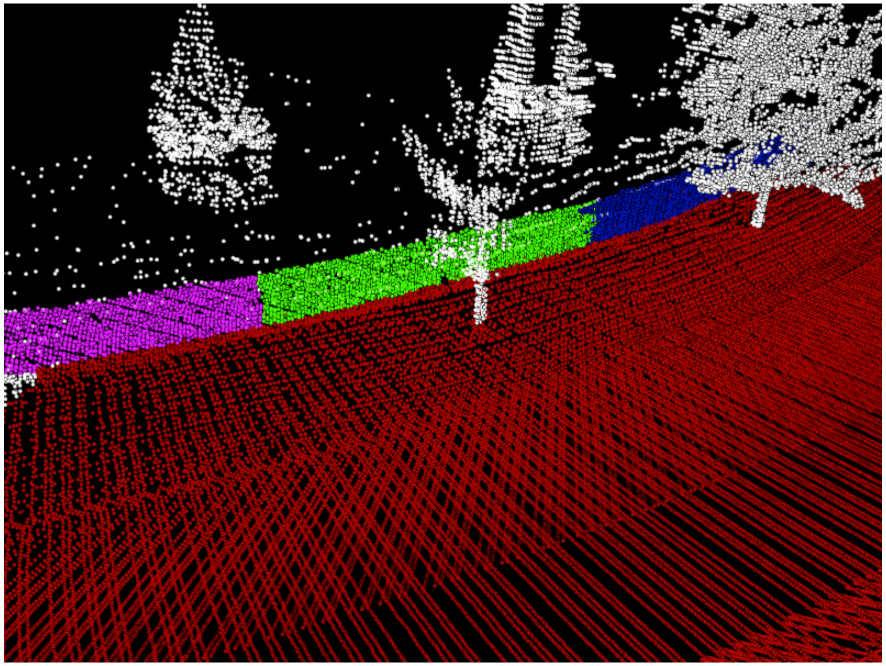
$$\vec{N} = \frac{1}{4}\vec{N}_{old} + \frac{3}{4}\vec{N}_{new}$$

Planar Region Extraction: Region Growing

- Globally:
  - Points are sorted based on planarity measure.
  - Do:
    - A region is grown from the most planar point.
    - Points found in this region are removed.
  - Until: the next-best planarity measure is above a threshold.

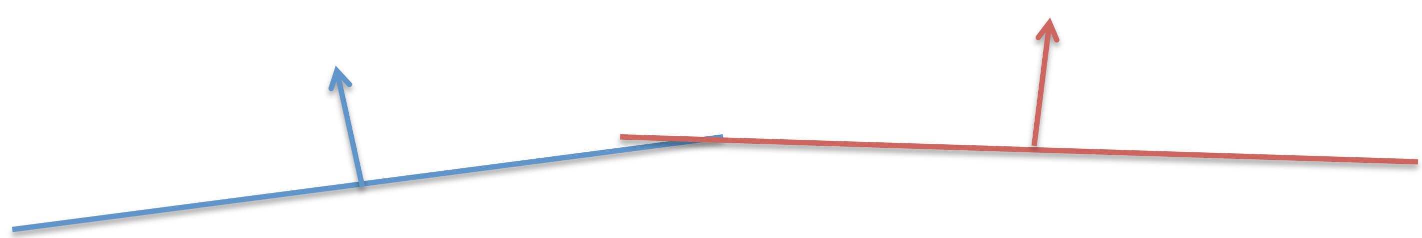
Planar Region Extraction: Region Combination

- Region growing will find all globally planar regions (ideally)
- Locally planar regions will be fragmented:



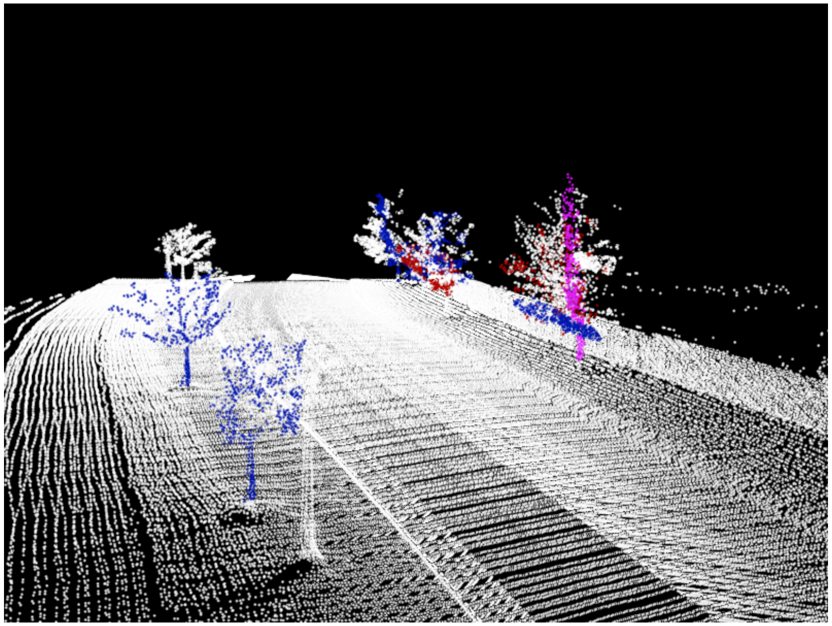
Planar Region Extraction: Region Combination

- Combine regions that:
  - Are close in space
  - Have similar planar normals



Planar Region Extraction: Region Pruning

- Noise regions often are identified:



- Prune regions based on average planarity measure of constituent points

Parallelization

- Region combination also works for globally planar regions, if fragmented

- This observation allows us to split the dataset into **spatially coherent pieces** and process these subsets separately

Results

- We established a hard accuracy metric based on a hand-created ground truth
- Per-region metrics:
  - Ground truth regions correctly identified (match)
  - Ground truth regions not identified (miss)
  - Identified regions that are not in ground truth (bogus)
  - For correctly matched regions, the percentage of points shared between the two (region accuracy)
- Per-point metrics:
  - True/false positives/negatives for region membership

Results

- Conducted experiments over a variety of subsets with varying thresholds and neighborhood definitions for SICK data

- Best parameters gave, on average:

Matched	**88.62%**
Missed	**7.23%**
Bogus	25.05%
Region Accuracy	**90.36%**

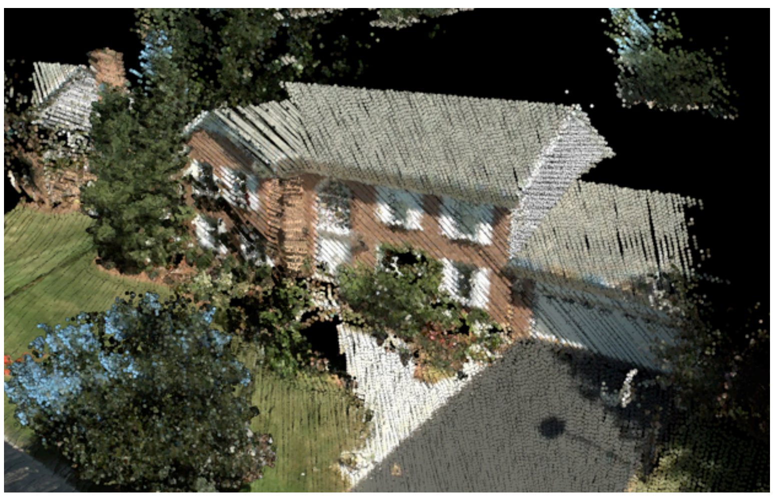
True Positives	**99.63%**
True Negatives	87.44%
False Positives	12.56%
False Negatives	**0.37%**

Region Extraction

- Questions?

Visualization: Mesh Generation

- Points are just samples; what is the underlying model?



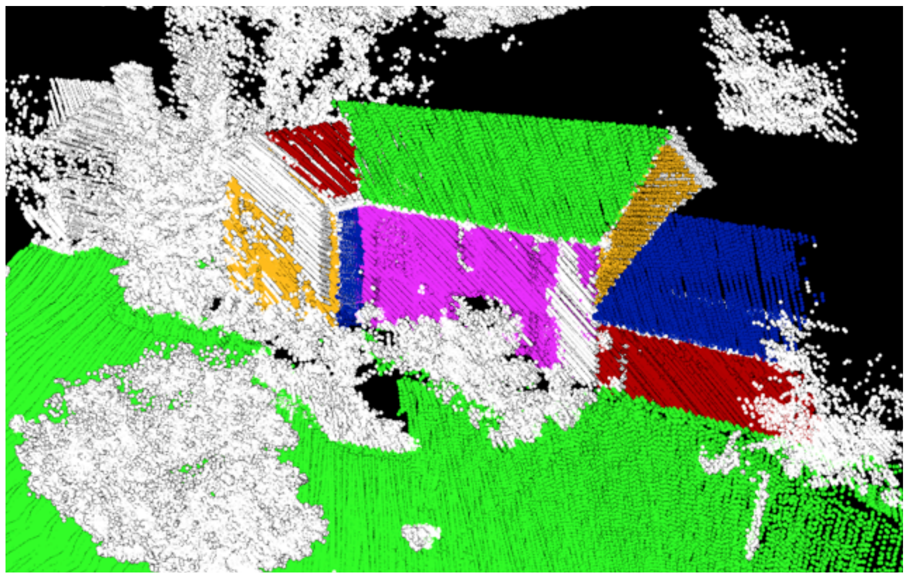
Mesh Generation – Related Work

- Solid representations are preferable in many applications, including visualization[1], object modeling[7,8], urban planning[10], and more.

- Most work in the area exploits structure from scan data, which is lost on scalable platforms.

Mesh Generation

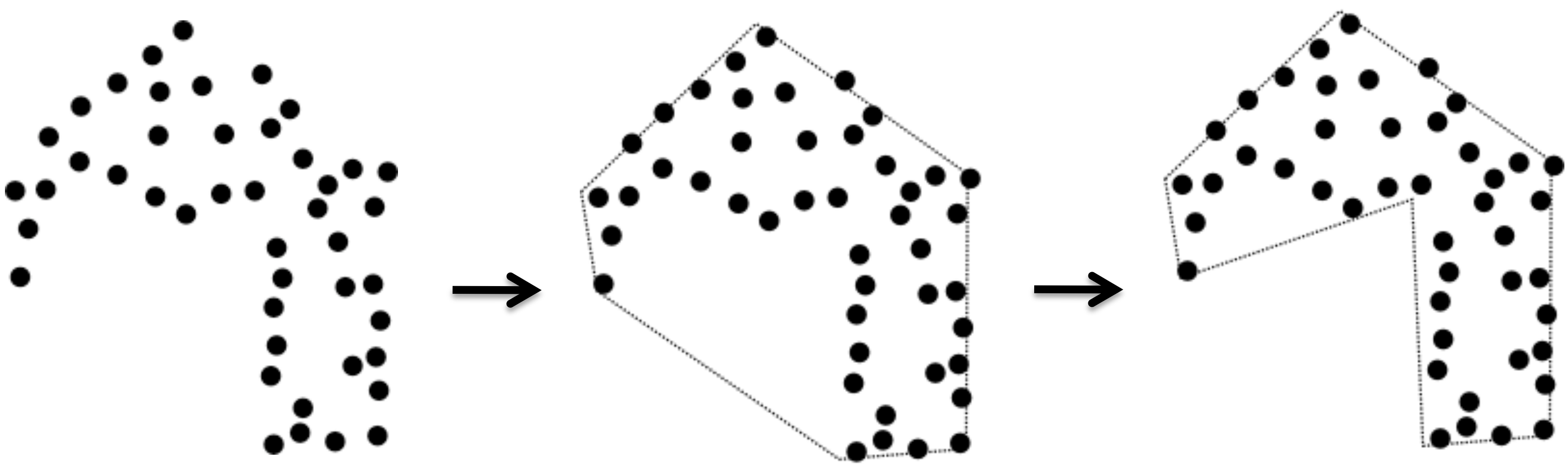
- Our approach uses extracted regions as input:



- And each region can be triangulated in 2D

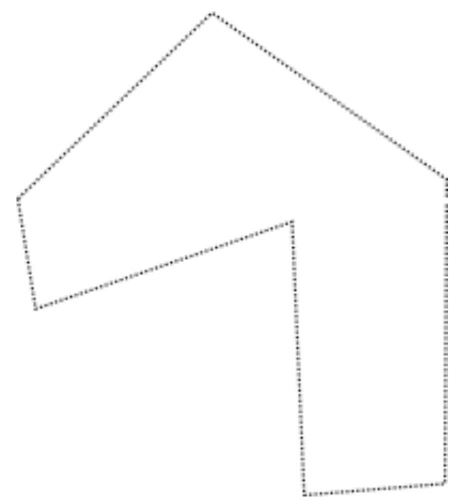
Mesh Generation

- In 2D, outline extraction is done via a modified convex hull approach



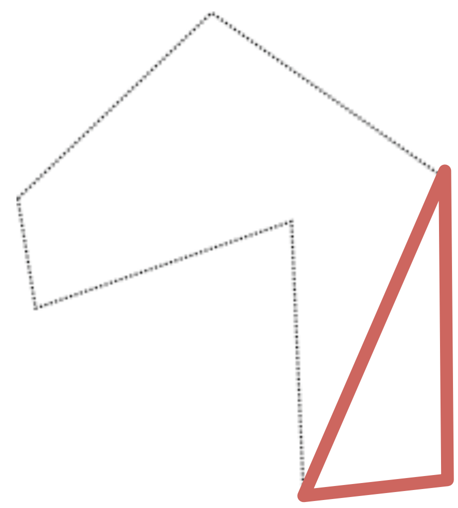
Mesh Generation

- Once we have an ordered outline for a region, we triangulate using an ear-cutting method [TODO]



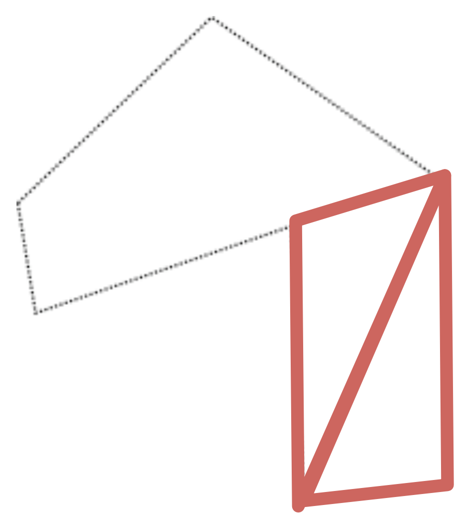
Mesh Generation

- Once we have an ordered outline for a region, we triangulate using an ear-cutting method [TODO]



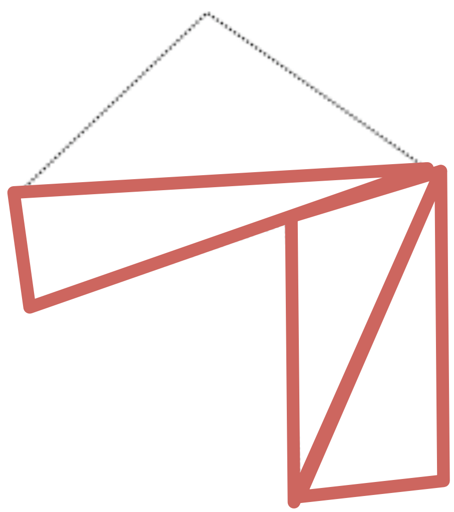
Mesh Generation

- Once we have an ordered outline for a region, we triangulate using an ear-cutting method [TODO]



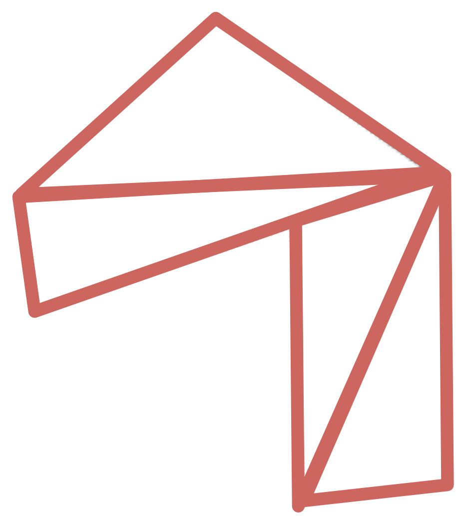
Mesh Generation

- Once we have an ordered outline for a region, we triangulate using an ear-cutting method [TODO]

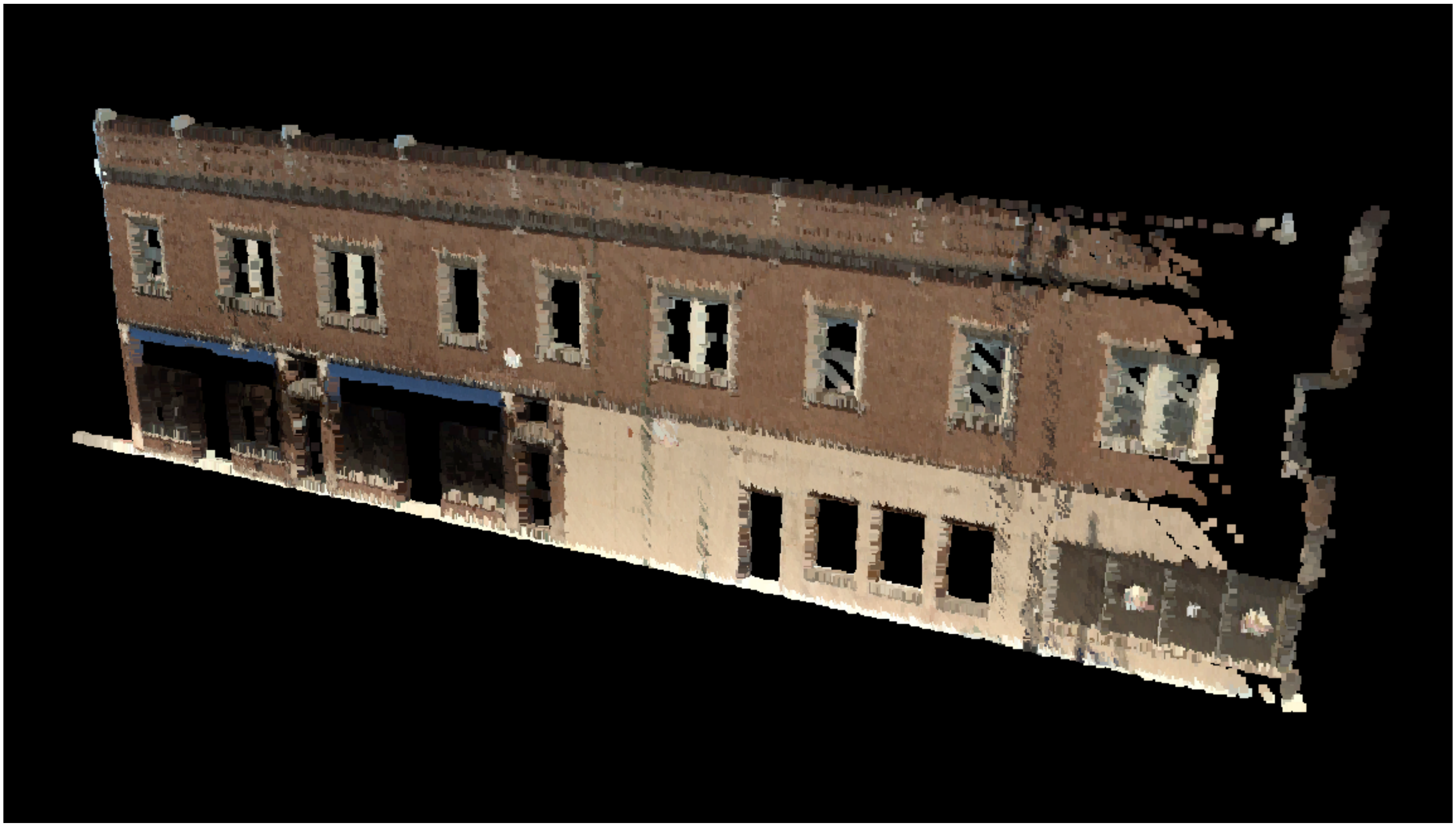


Mesh Generation

- Once we have an ordered outline for a region, we triangulate using an ear-cutting method [TODO]

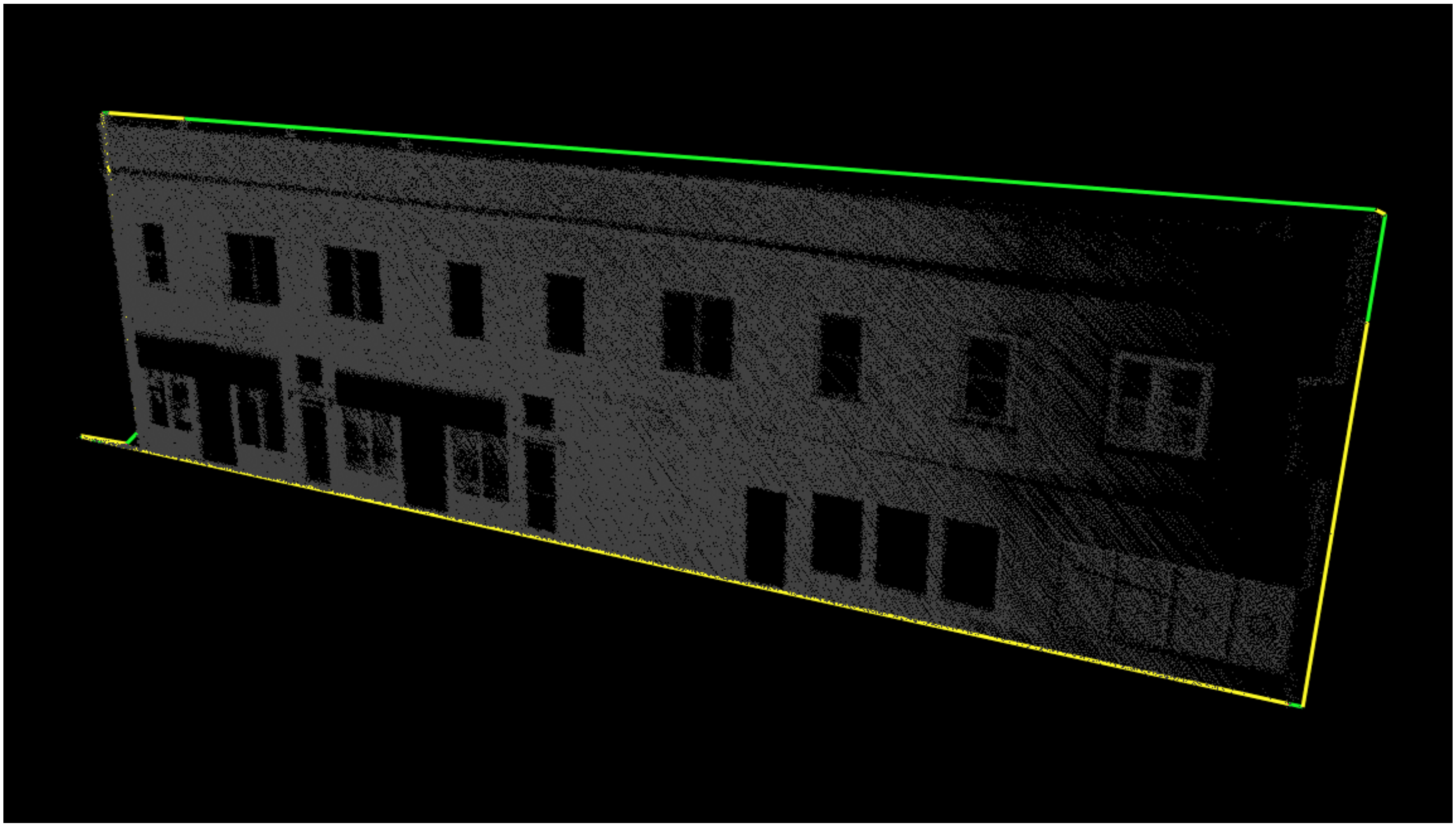


Mesh Generation: Single Region



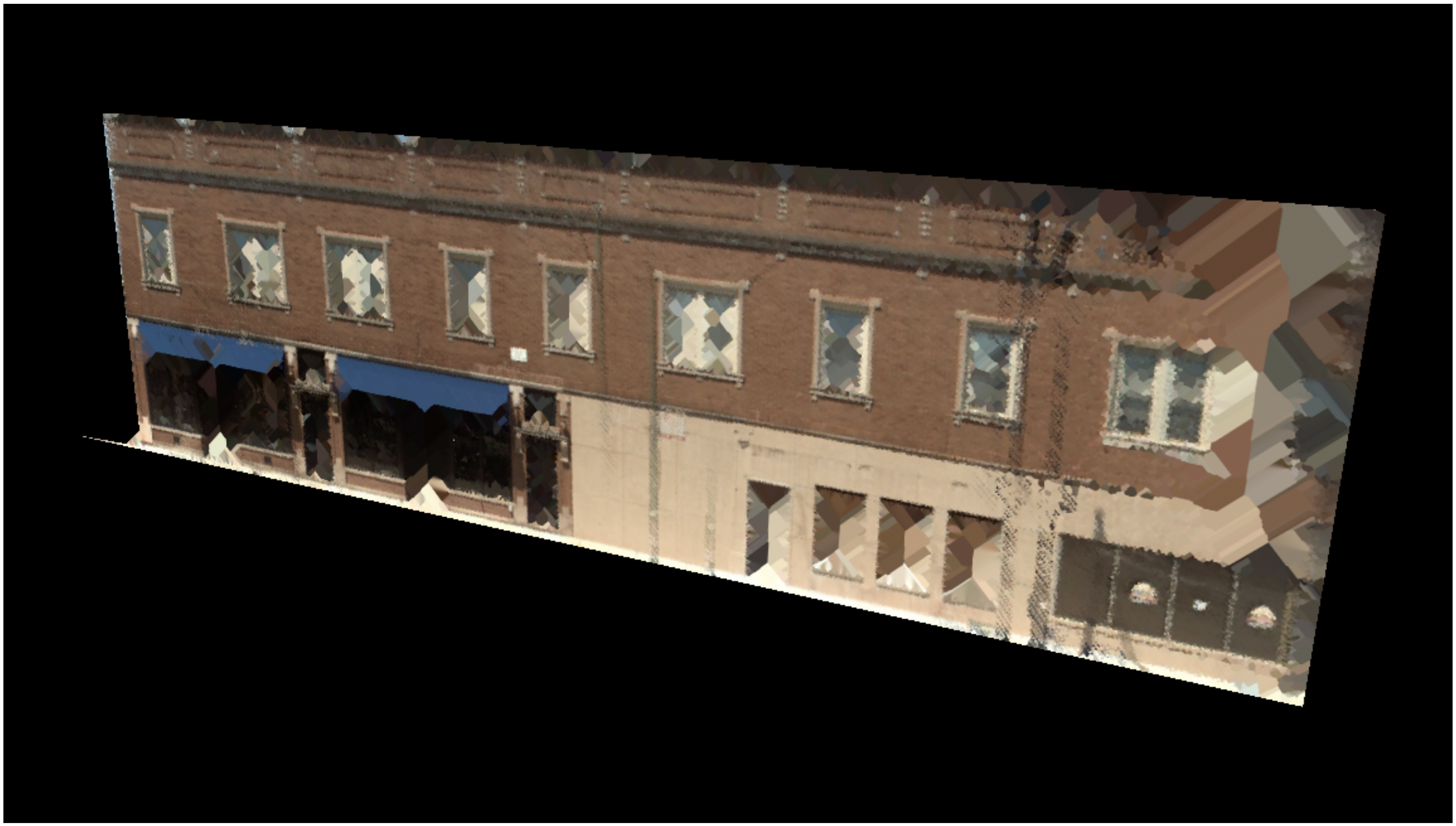
Points detected as belonging to a single region

Mesh Generation: Single Region



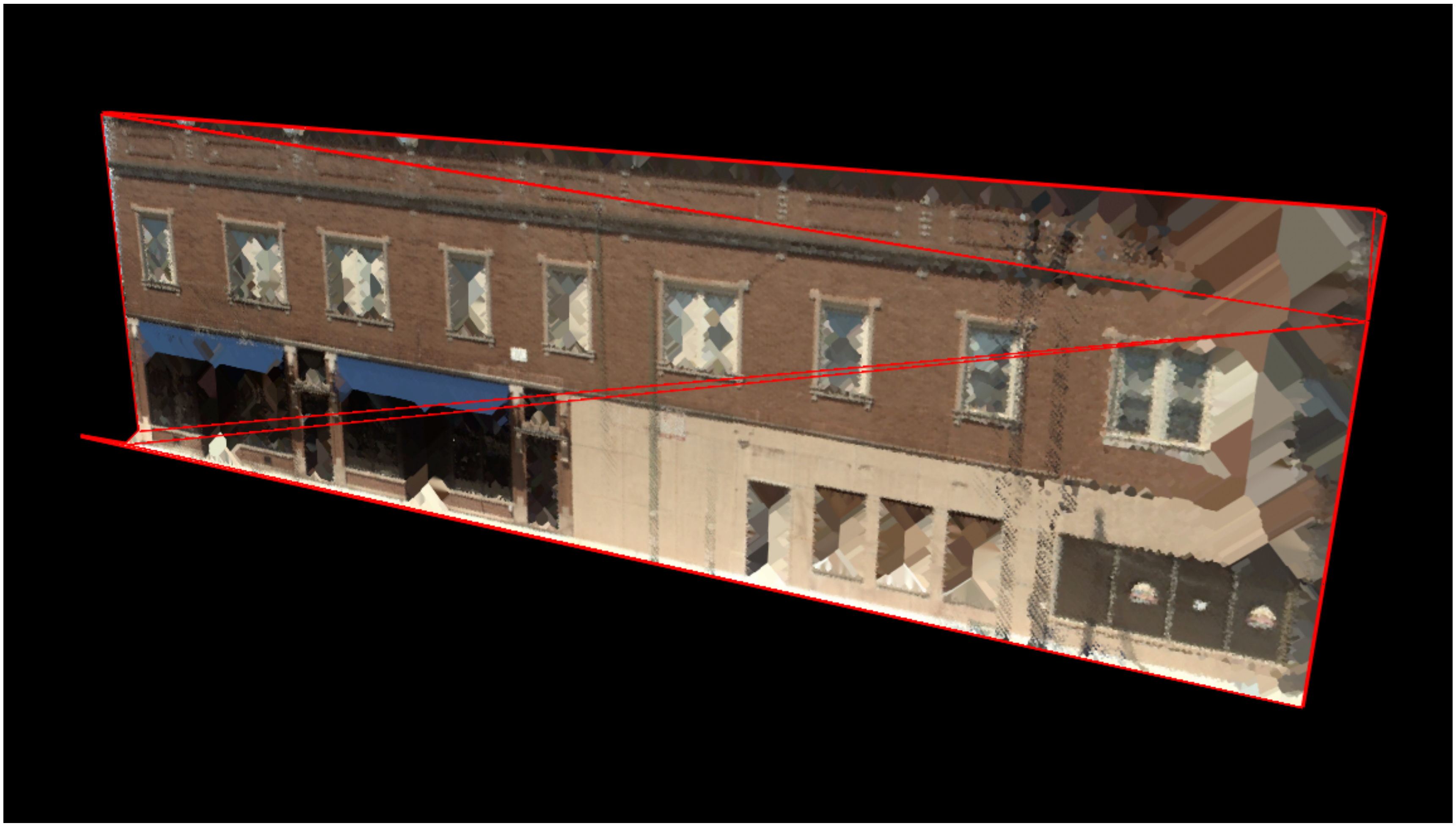
Simplified outline after outline extraction

Mesh Generation: Single Region



Triangulated region, with texture generated from nearby points

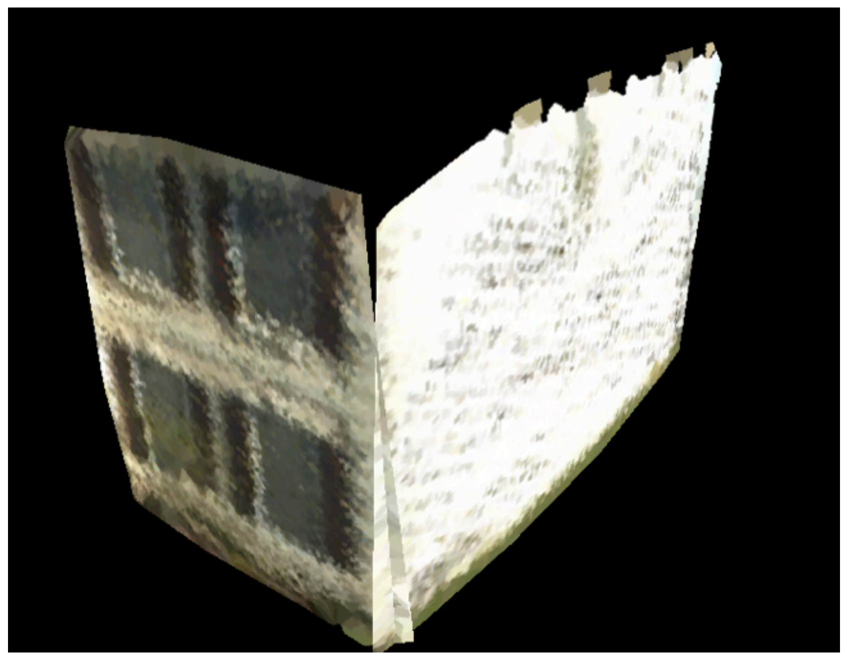
Mesh Generation: Single Region



Textured triangulation, with wireframe overlaid

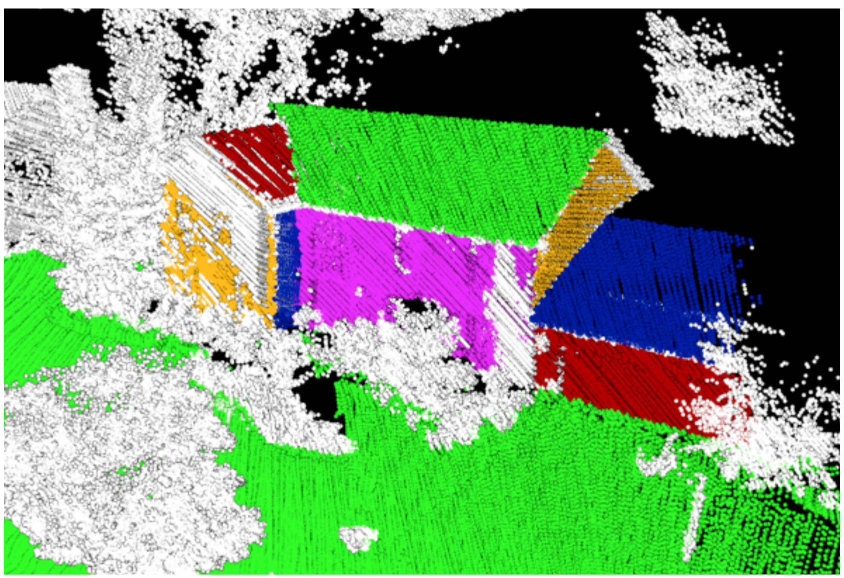
Mesh Generation

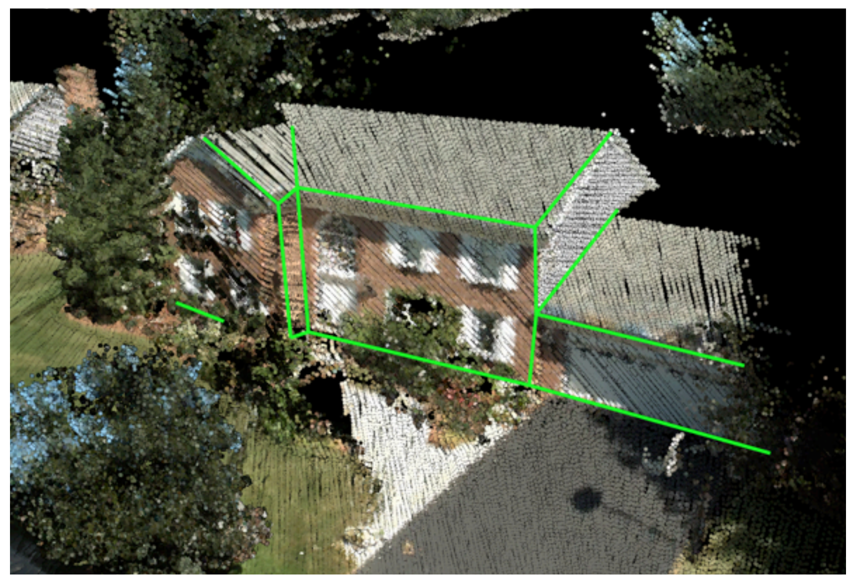
- Triangulating regions independently causes boundary issues:



Mesh Generation

- Global unification step identifies shared borders and corners among all regions
- Uses these guidelines to snap outlines

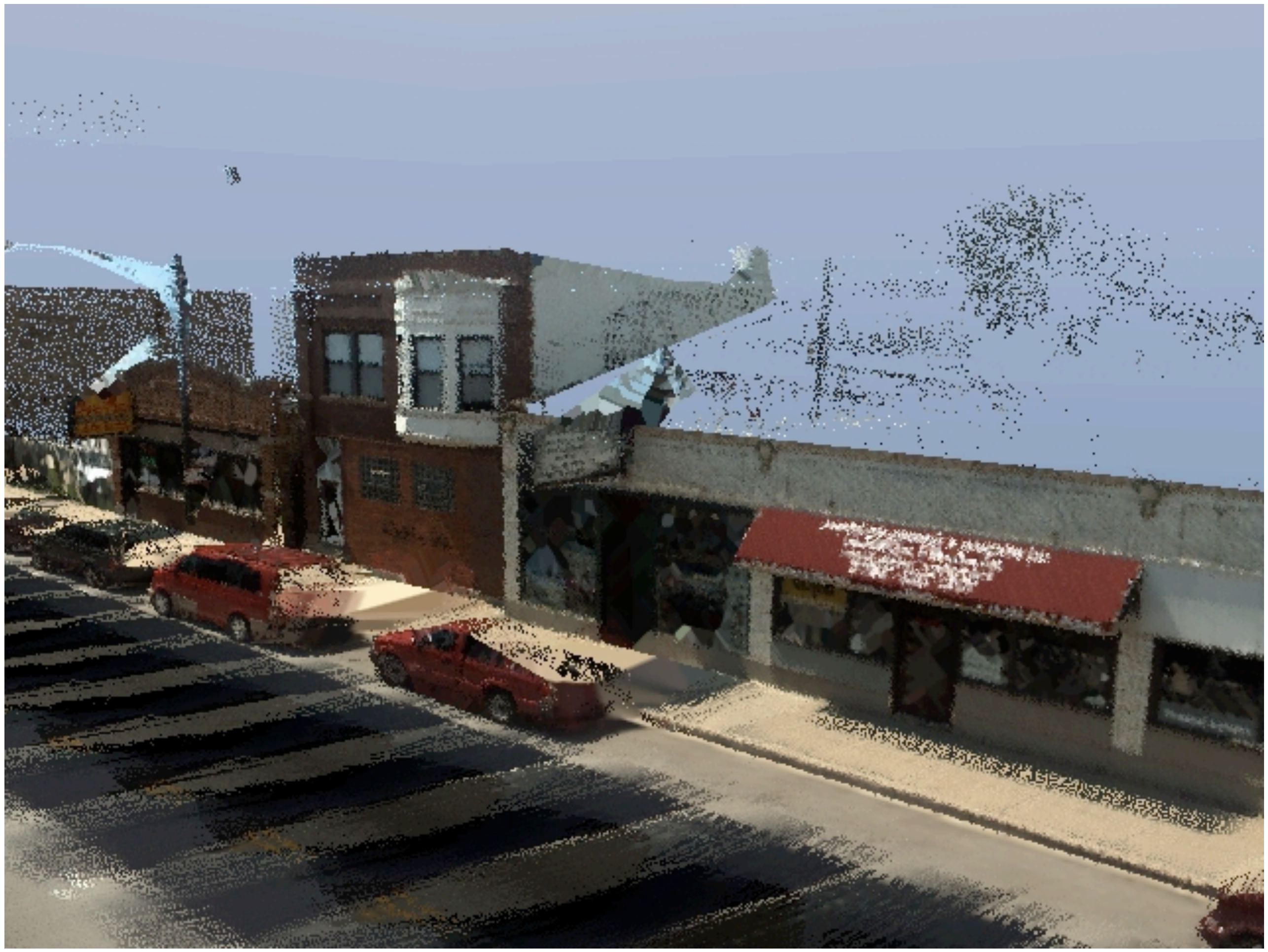




Mesh Generation

- Perform the stages in this order:
  - Global edge unification
  - Outline extraction and simplification
  - Surface triangulation
  - Texture generation

Mesh Generation



Future Work

- Improved mesh and texture generation using camera images instead of point colors
- Improved region segmentation using domain-specific knowledge
- Cross-time change detection using extracted regions

Thank you!

- Questions?

- *Special thanks to the NAVTEQ Corporation for their sponsorship of this research*

- Publications:

  A. Zavodny, P. Flynn, and X. Chen, *Region Extraction in Large-Scale Urban LIDAR Data*, 3D Digital Imaging and Modeling 2009 (3DIM'09)

  A. Zavodny, P. Flynn, and X. Chen, *Textured Mesh Generation of Extracted Regions from Urban Range-Scanned LIDAR Data* (submitted, 3DIMPVT '11)

References

- [1] P. Allen, S. Feiner, A. Troccoli, H. Benko, E. Ishak, and B. Smith. ***Seeing into the past: Creating a 3d modeling pipeline for archaeological visualization***. Proceedings of the International Symposium on 3D Data Processing Visualization and Transmission (3DPVT04), 0:751–758, 2004.
- [2] W. Boehler and A. Marbs. ***3d scanning and photogrammetry for heritage recording: A comparison***. In Procedings of the Twelfth International Conference on Geoinformatics, 2004.
- [3] G. Guidi, F. Remondino, M. Russo, F. Menna, and A. Rizzi. ***3d modeling of large and complex sites using multi-sensor integration and multi-resolution data***. In Proceedings of the Ninth International Symposium on Virtual Reality, Archaeology and Cultural Heritage (VAST08), pages 85–92, 2008.
- [4] G. Sithole and G. Vosselman. ***Automatic structure detection in a point cloud of an urban landscape***. In Proceedings of the Second GRSS/ISPRS Joint Workshop on Remote Sensing and Data Fusion over Urban Areas, pages 67–71, 2003.
- [5] Ahmed F. Elaksher and James S. Bethel. ***Reconstructing 3d buildings from lidar data***. In Proceedings of Photogrammetric Computer Vision (PCV02), ISPRS 02, 2002.
- [6] Gianfranco Forlani, Carla Nardinocchi, Marco Scaioni, and Primo Zingaretti. ***Complete classification of raw lidar data and 3d reconstruction of buildings***. Pattern Analysis and Applications, 2006.

References

- [7] Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, and Duane Fulk. ***The digital michelangelo project: 3D scanning of large statues***. In Proceedings of ACM SIGGRAPH 2000, pages 131–144, July 2000.
- [8] Jeremy Rowe and Anshuman Razdan. ***Digital library system: capture, analysis, query, and display 3d data***. In Proceedings of the Second ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL02), pages 382–382, New York, NY, USA, 2002. ACM.
- [9] H. Zhao and R. Shibasaki. ***Reconstructing a textured cad model of an urban environment using vehicle-borne laser range scanners and line cameras***. Machine Vision and Applications, volume 14, number 1:35 – 41, 2003.
- [10] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, and G. Taubin. ***The ball-pivoting algorithm for surface reconstruction***. volume 5, pages 349–359, 1999.