Plane detection in RGB-D images

GPU Programming in Computer Vision Praktikum

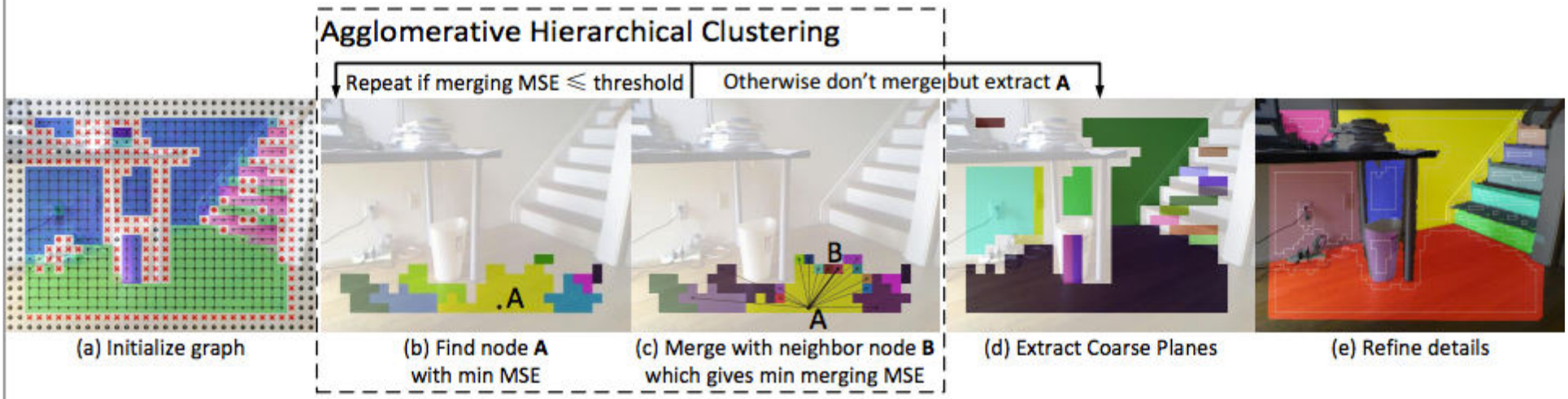
Johannes Klein, Amad Junaid, Patricia Vitoria Carrera
supervised by Lingni Ma

Agenda

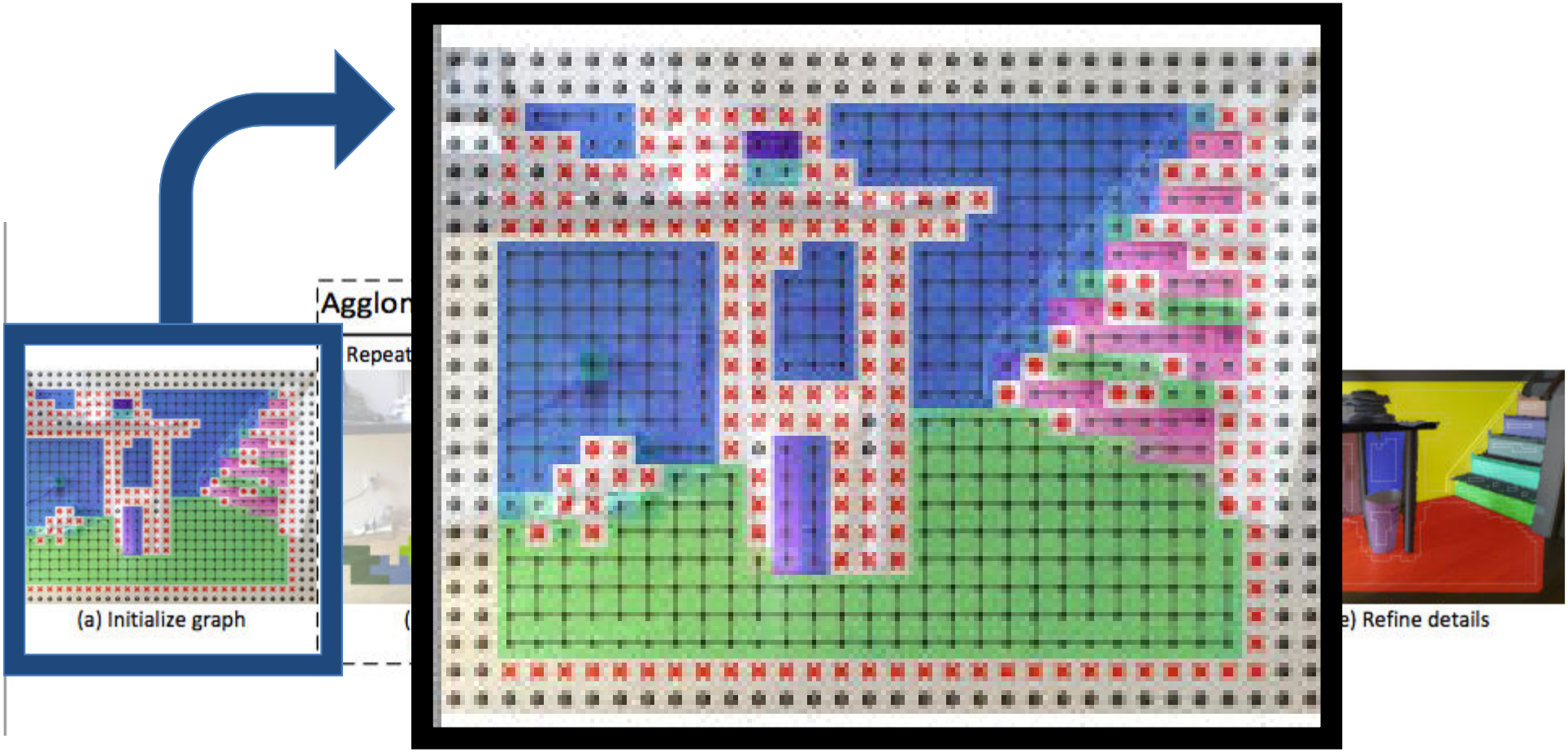
- Overview of the Algorithm
- Overview of the main tasks in the project
- Implementation details
- Final results
- What have we done to make the algorithm faster?
- Comparison of speed up

Overview of Algorithm

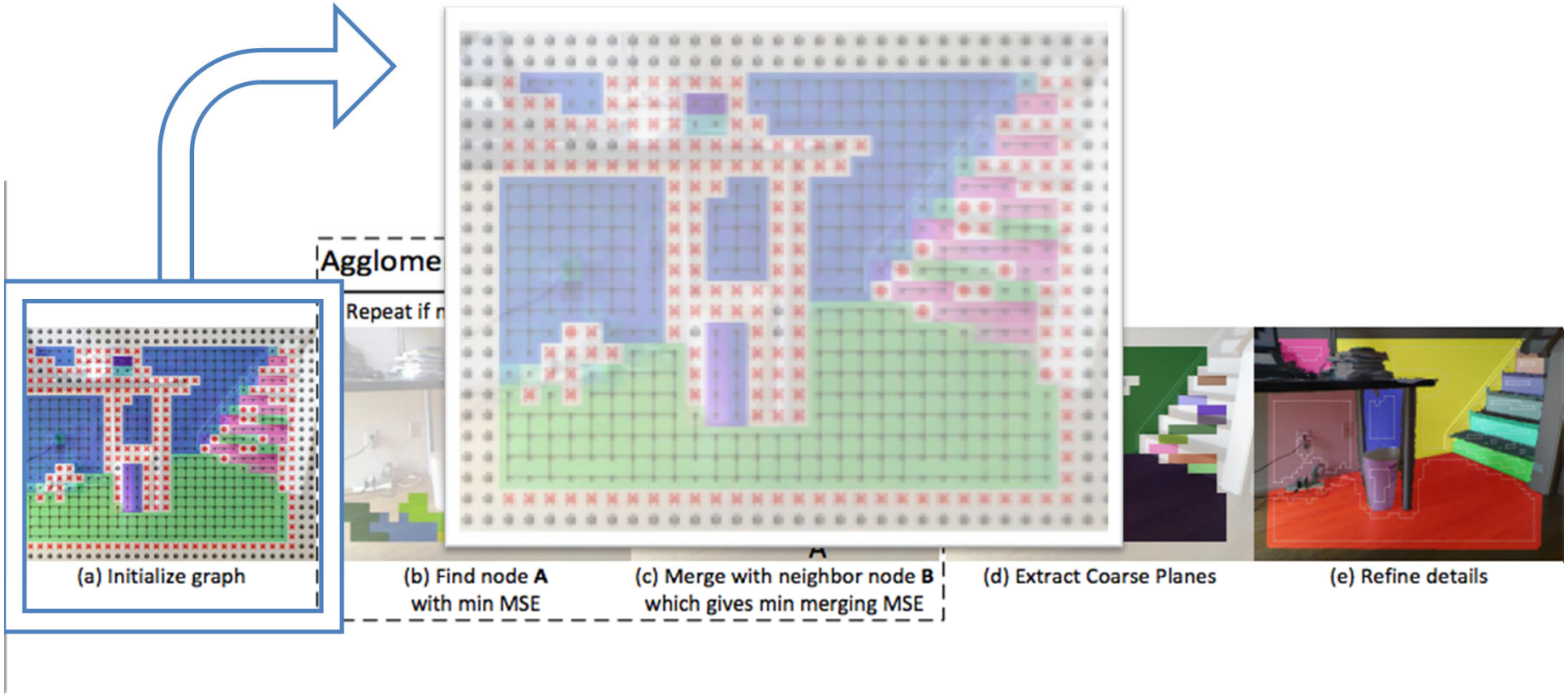
Fast Plane Extraction in Organized Point Clouds Using Agglomerative Hierarchical Clustering (Chen Feng, Yuichi Taguchi and Vineet R. Kamat)



Graph initialization



Overview of the tasks of the project



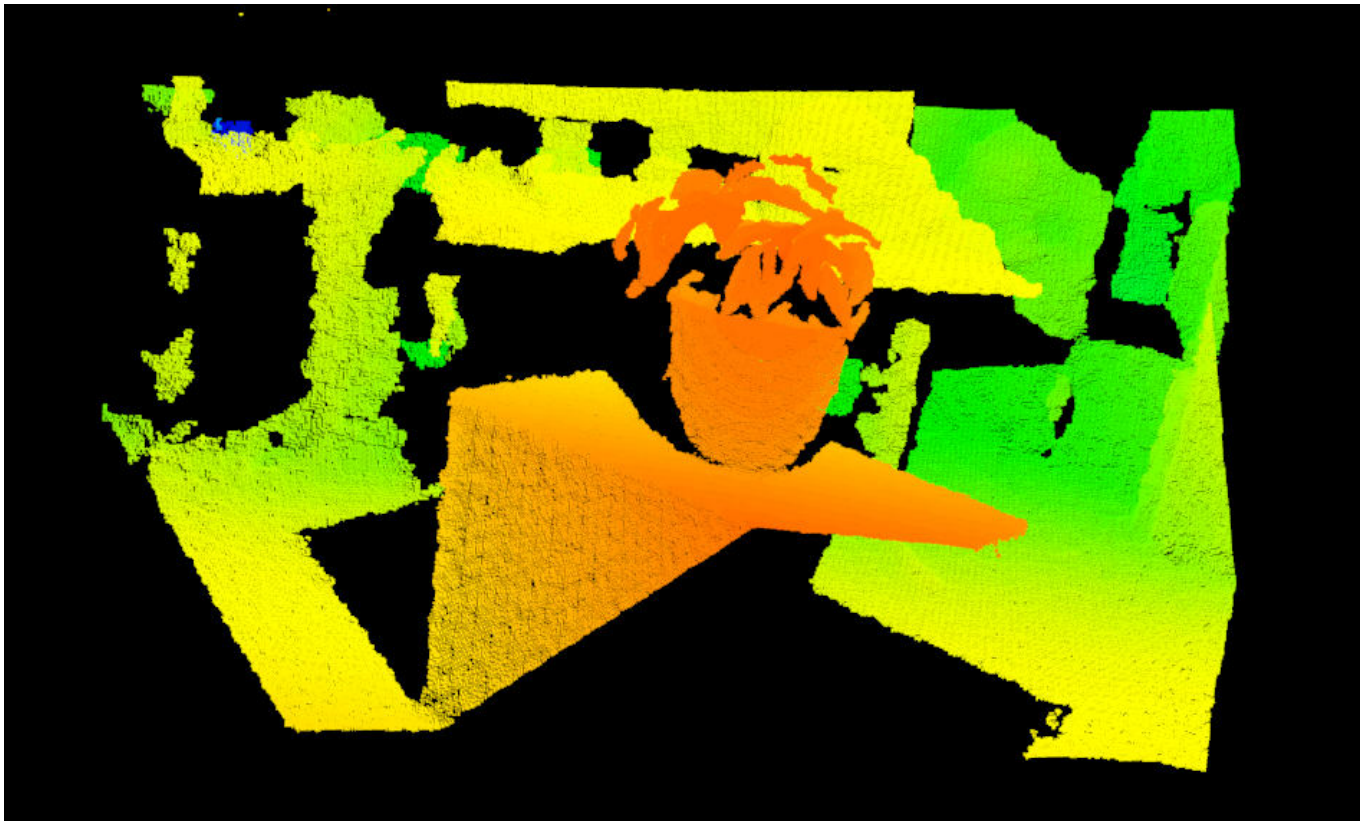
Convert Depth images into a point cloud → **Non-overlapping node initialization** → First check bad nodes: Missing data → Second check bad nodes: Depth discontinuities → Third check bad nodes: High MSE → Fourth check bad nodes: Edge rejection

Convert depth images to a point cloud

- Input: Depth Image
- Output: Point cloud

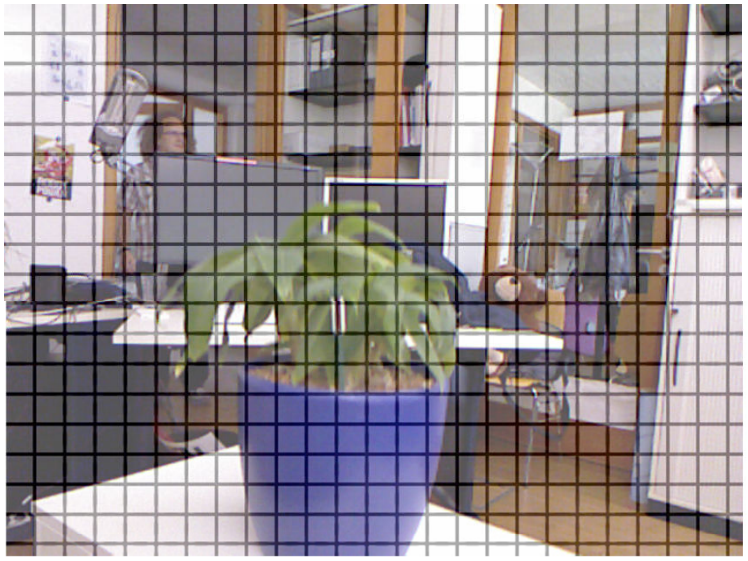
$$\begin{cases} x = \dfrac{u - o_x}{f_x} \times d \\ y = \dfrac{v - o_y}{f_y} \times d \\ z = d \end{cases}$$

Convert depth images to a point cloud



Convert depth images to a point cloud

- Divide the point cloud into a set of initial nodes of the size HxW in the image space.



Reject bad initial nodes

- Nodes containing Missing Data
- Nodes containing Depth Discontinuities
- Nodes Having High MSE

Convert Depth images into a point cloud

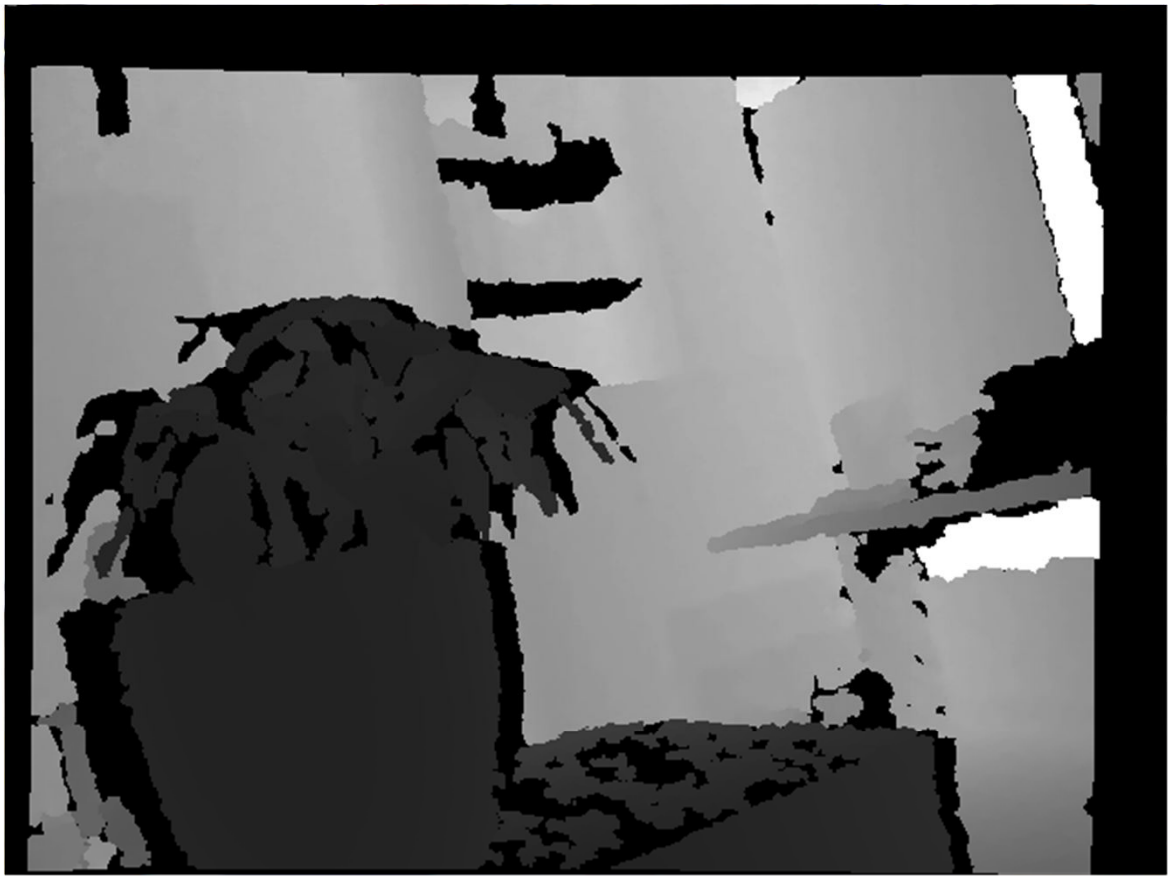
Non-overlapping node initialization

First check bad nodes: Missing data

Second check bad nodes: Depth discontinuities

Third check bad nodes: High MSE

Fourth check bad nodes: Edge rejection



Convert Depth images into a point cloud → **Non-overlapping node initialization** → First check bad nodes: Missing data → Second check bad nodes: Depth discontinuities → Third check bad nodes: High MSE → Fourth check bad nodes: Edge rejection

Edge Rejection

- Nodes at boundary between two planes.
- Rejected if the angle between every node plane is bigger than a threshold.

Implementation Details

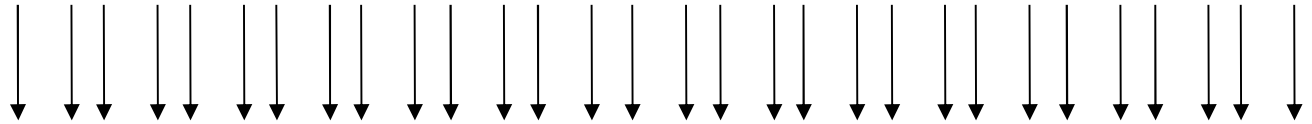
- Our convention is to use:
  - Shared memory block size as the node dimension that the user choose.
  - Then every thread corresponds to one point.
  - This limits us to use a maximum node size of 1024 as a block size.

Implementation Details

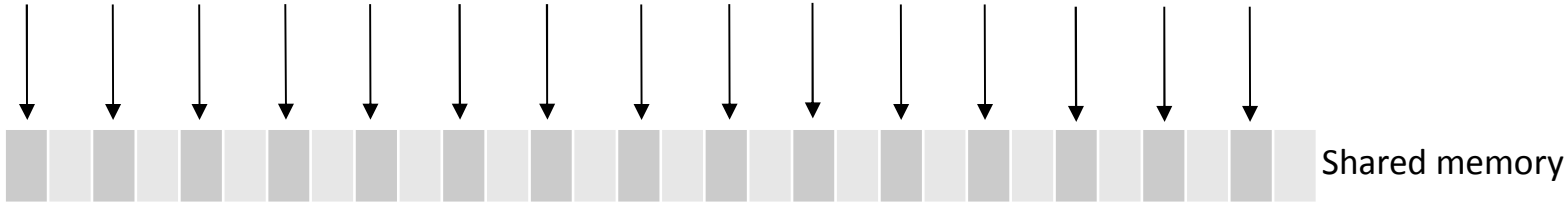
Node[0]	stores node data
Node[1]	stores node data
Node[2]	stores node data
...	
Node[n-1]	stores node data

Covariance Inplace Calculation

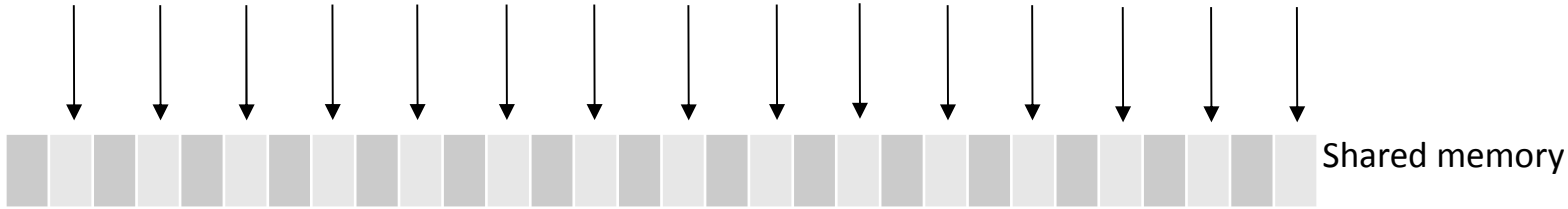
each thread processes a point



- select all **even** thread-Ids and process the covariance matrix
- save the covariance calculation in place



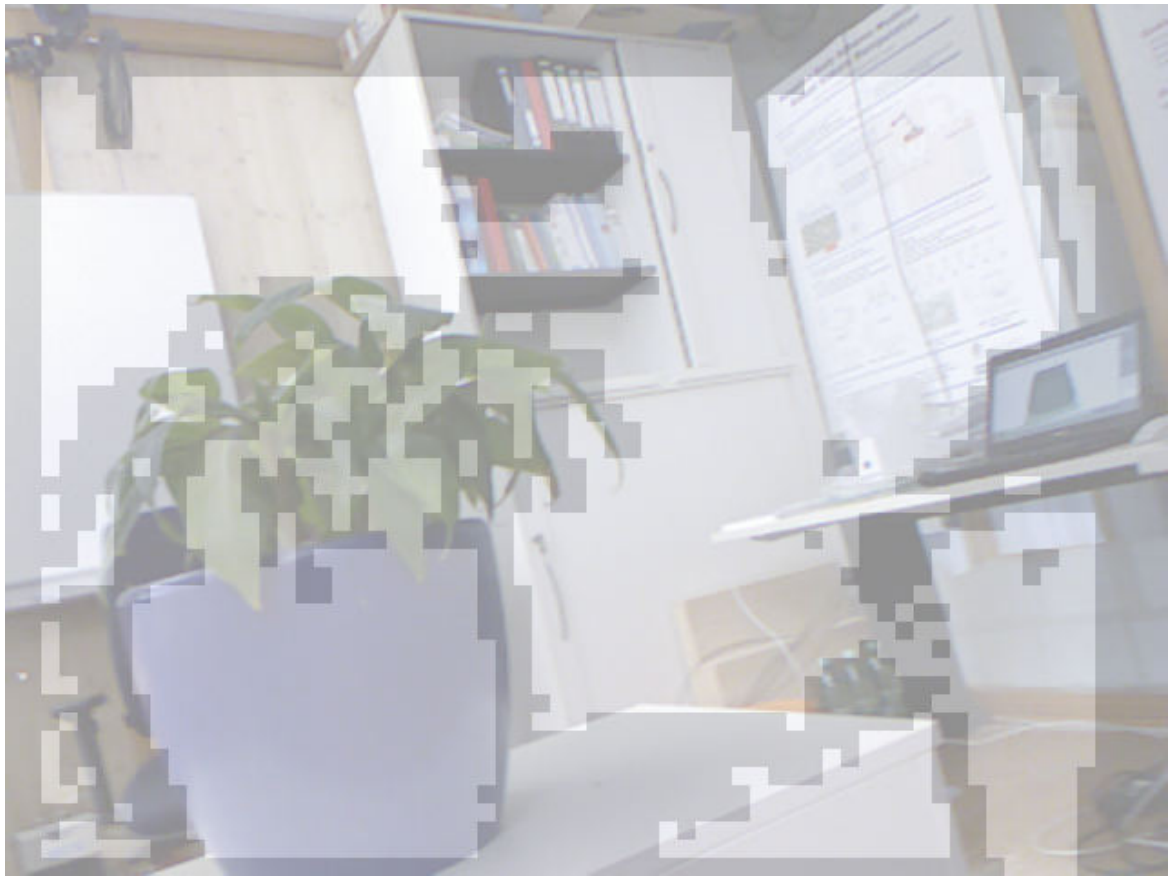
- select all **odd** thread-Ids and process the covariance matrix
- save the covariance calculation in place



Results: RGB Data

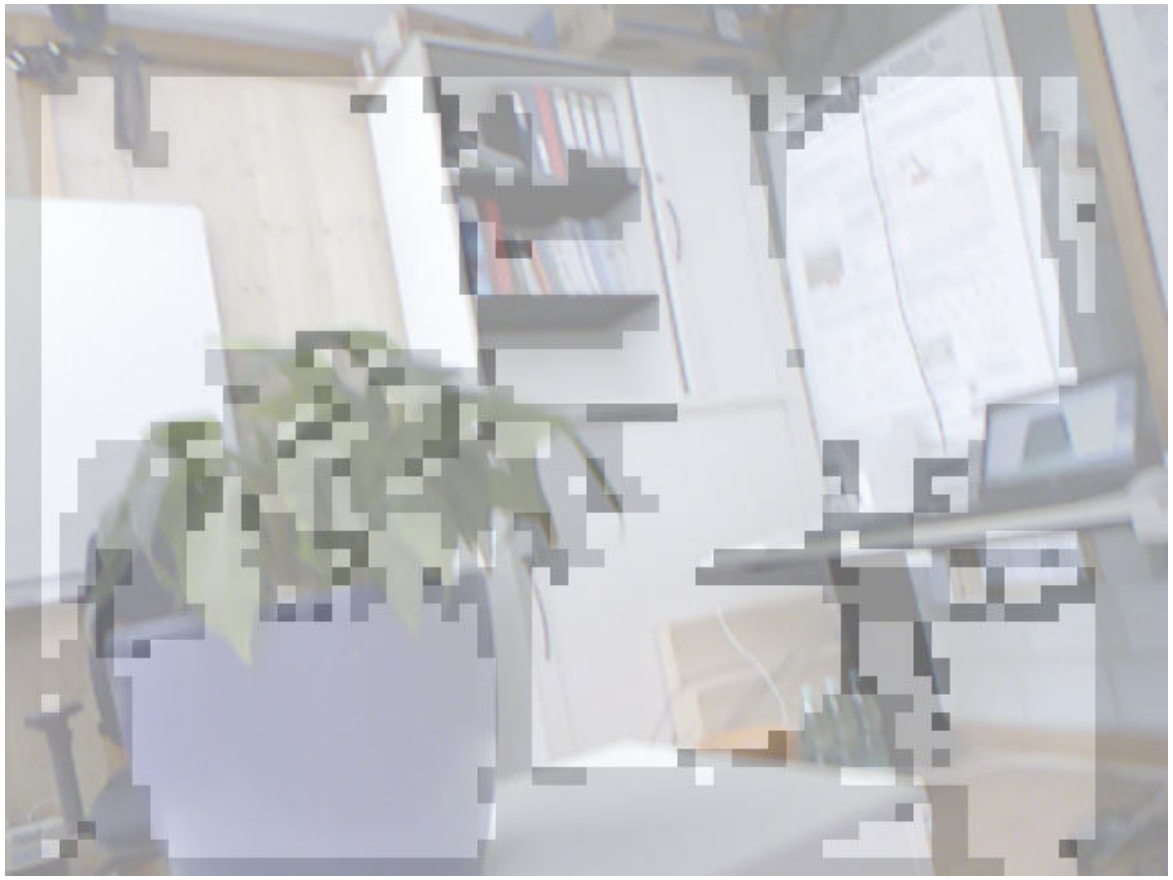
1st Check: Missing Data

1st Check: Missing Data



2nd Check: Depth Discontinuity

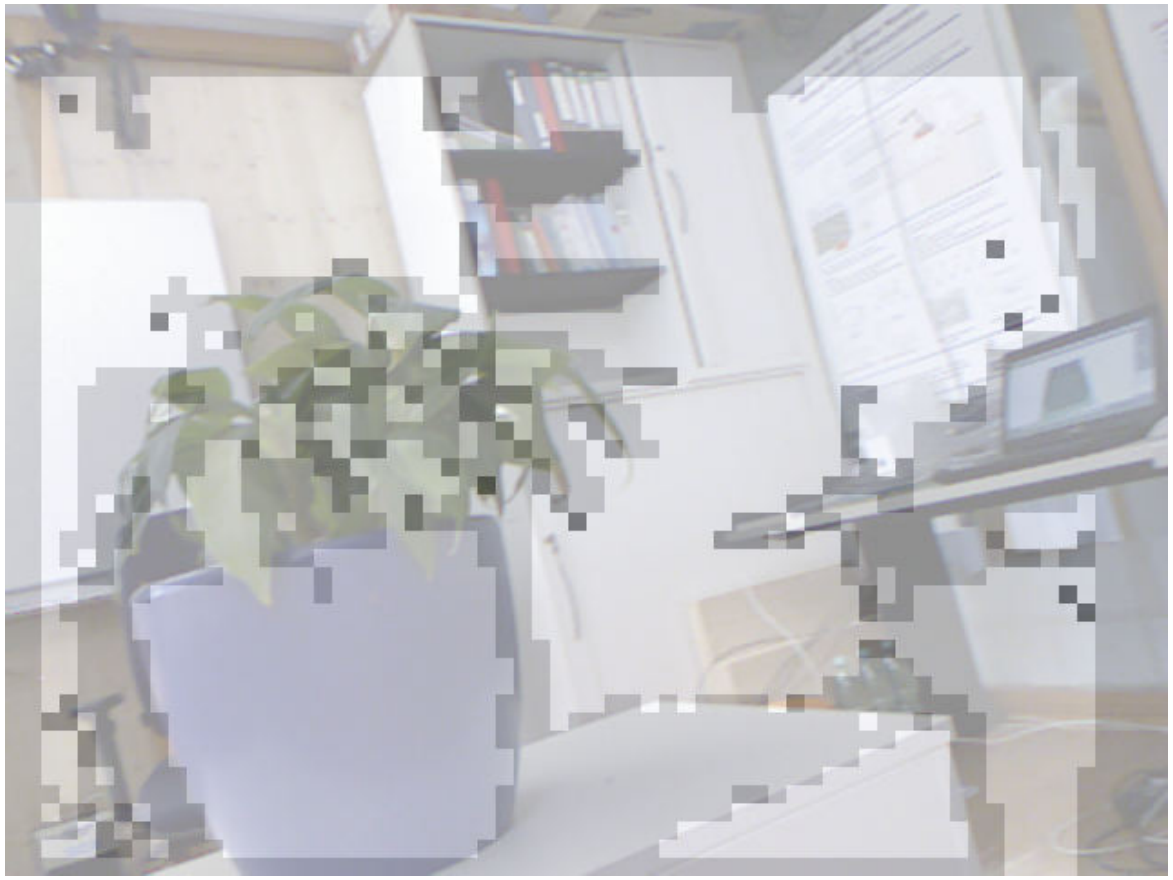
2nd Check: Depth Discontinuity



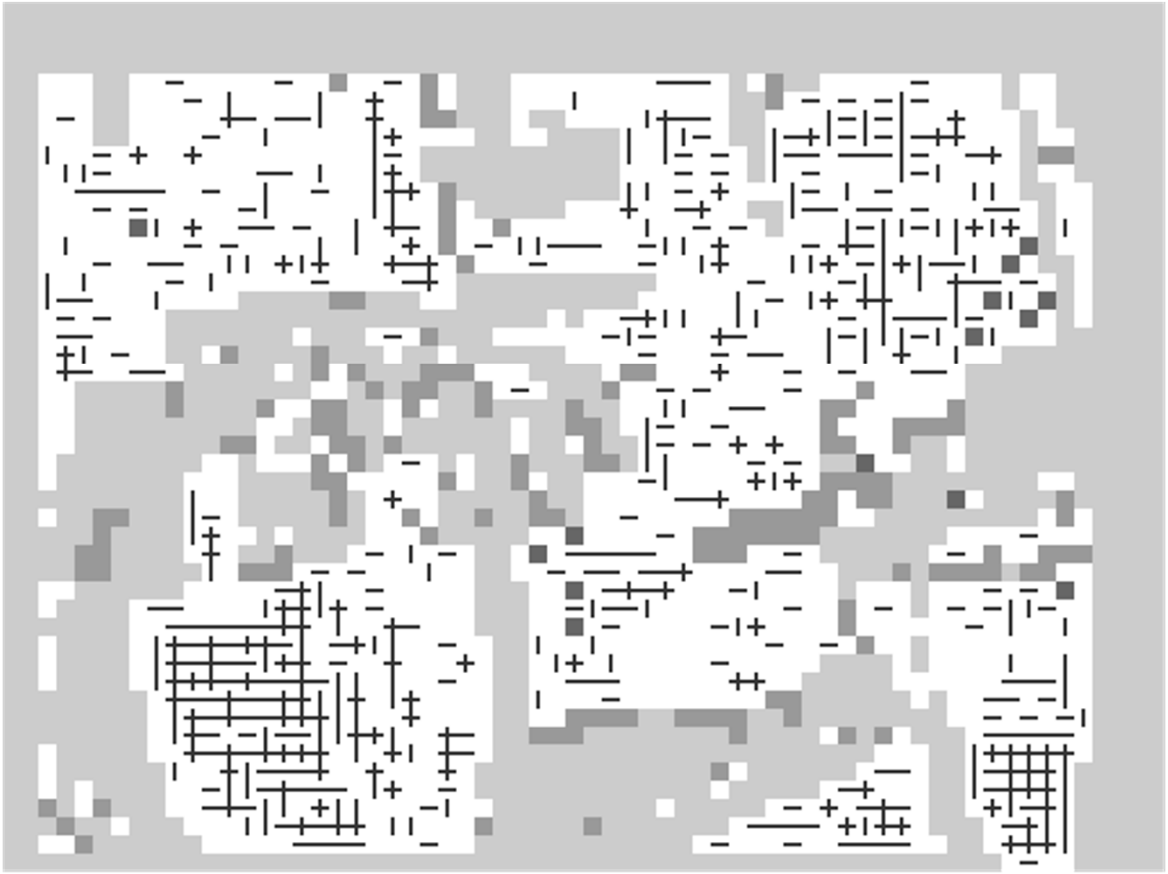
3rd Check: MSE Plane fitting

Convert Depth images into a point cloud
Non-overlapping node initialization
First check bad nodes: Missing data
Second check bad nodes: Depth discontinuities
Third check bad nodes: High MSE
Fourth check bad nodes: Edge rejection

3rd Check: MSE Plane fitting



4th Check: Edge Rejection

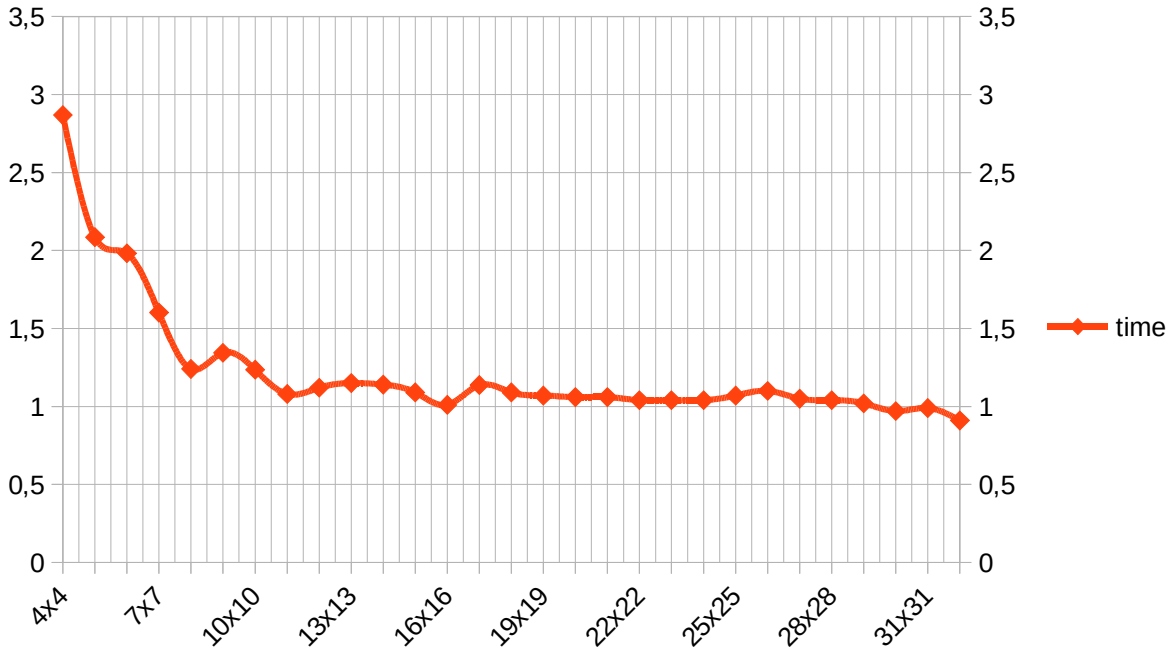


4th Check: Edge Rejection

Steps that improved the performance

- Reduce access to global memory using shared memory whenever it was possible.
- Using parallel reduction for array sumations.
- Intereseting Observations:
  - CULA library usage
  - Atomic adds thread blocking

Comparison of speed up (Per Frame)



CPU: 15 ms (20 x 20)
GPU: 1.13 ms

Demo

Thank you for your attention

Questions?