Sensor Fusion and Calibration of Velodyne LiDAR and RGB Camera

Martin Veľas, Michal Španěl, et al.

Department of Computer Graphics and Multimedia,
Faculty of Information Technology,
Brno University of Technology

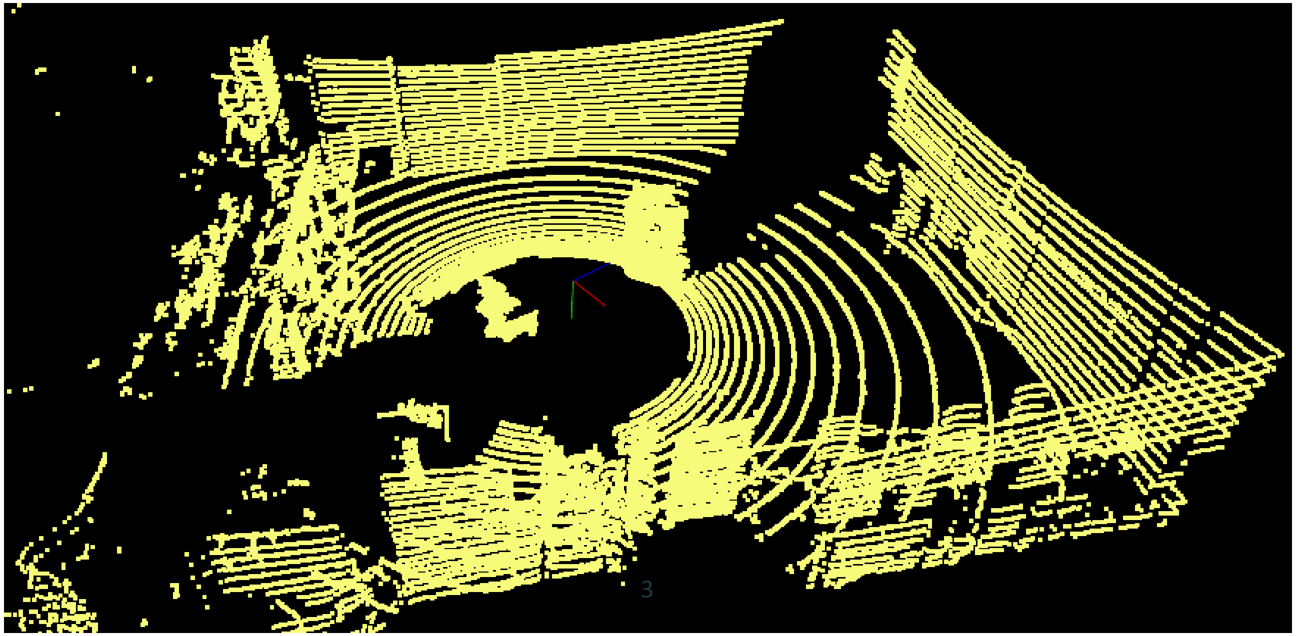
IT4Innovations national supercomputing center
BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY
ROBOFIT
EUROPEAN UNION
EUROPEAN REGIONAL DEVELOPMENT FUND
INVESTING IN YOUR FUTURE
2007-13
OP Research and Development for Innovation

What is Velodyne LiDAR?

- LiDAR = Light Detection and Ranging
- HDL-32: 32 lasers scan the area around the sensor
- Beam is rotating (10Hz)
- Field of view
  - Horizontal 360°
  - Vertical <-30°; +10°>
- 70m range, 2cm accuracy

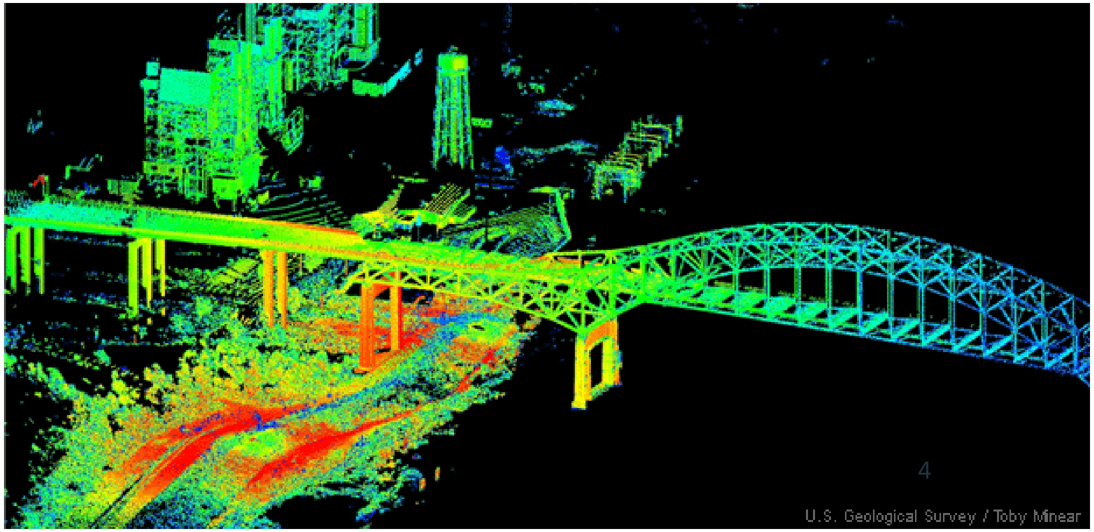
The data provided by Velodyne

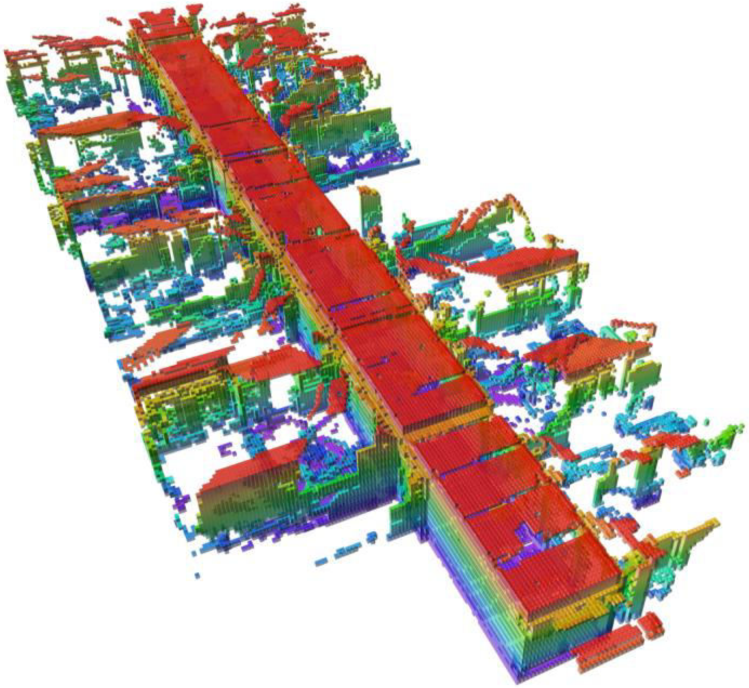
- Point cloud
- Velodyne captures 700k points per each scan (7M pt/s)
- Points are organized into the „rings“
  - ring = points captured by one lasser scanner/ray (32 rings)



Practical application (1/3)

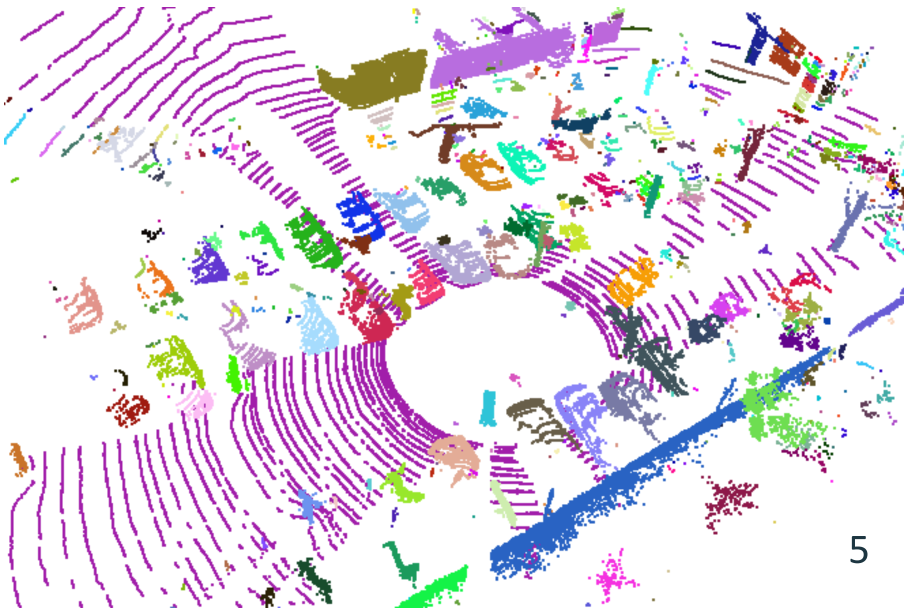
- Merging the point cloud (with proper alignment)
- 3D reconstruction (GPU acceleration)
- Creation of the environment map
  - Large data

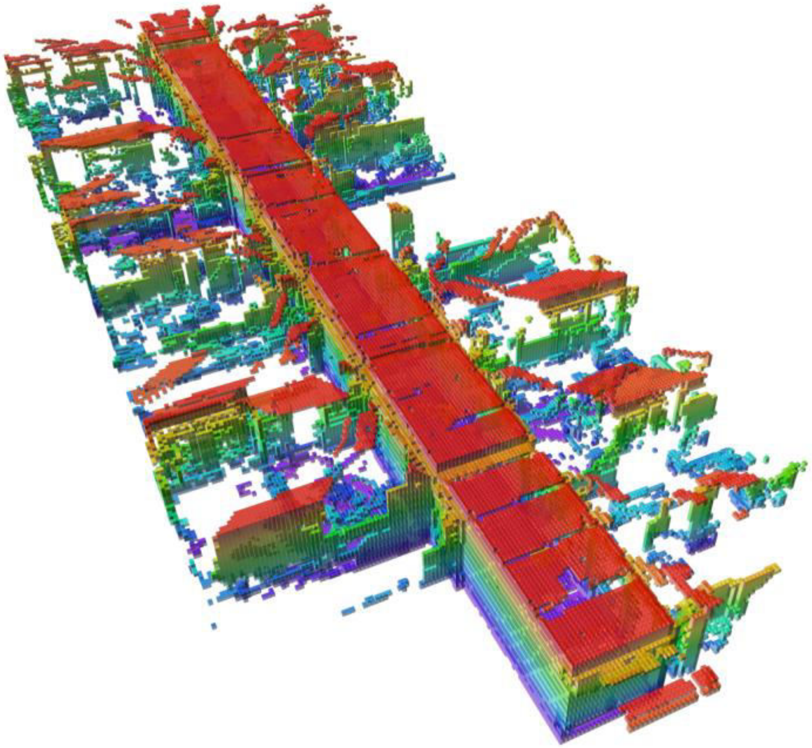




Practical application (2/3)

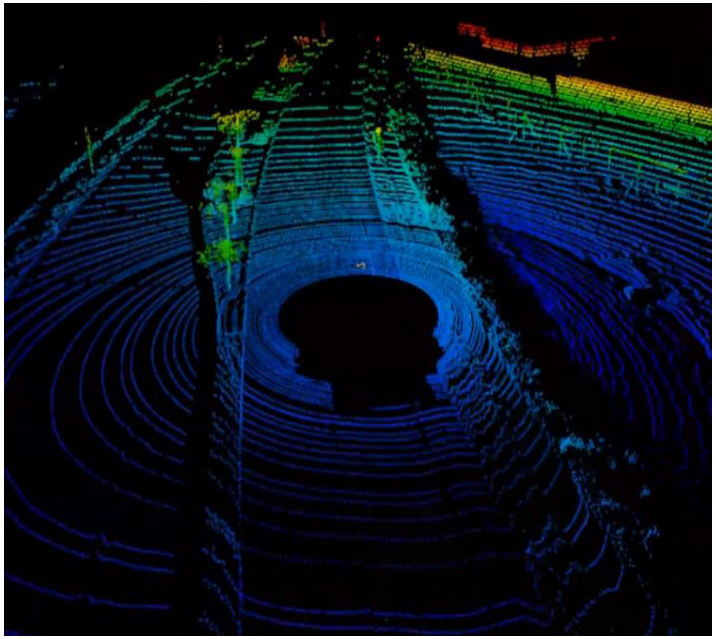
- Robotics ☺
- Navigation in 3D environment (using the map)
- Obstacle avoidance
- Object detection





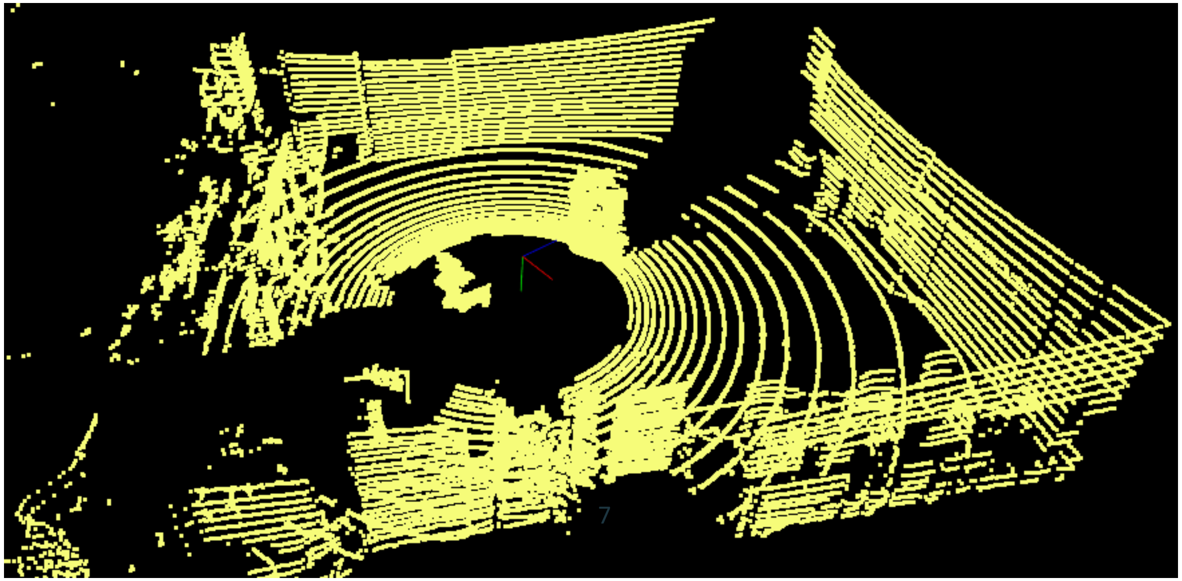
Practical application (3/3)

- **Google car**
  - Autonomous vehicle
  - Successful in **DARPA** Urban Challenge
  - Using Velodyne HDL-64
  - Navigation in the urban environment



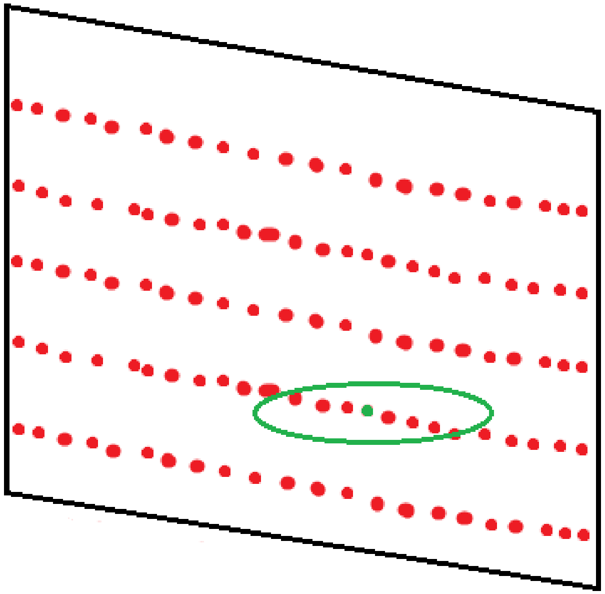
Common issues with Velodyne data

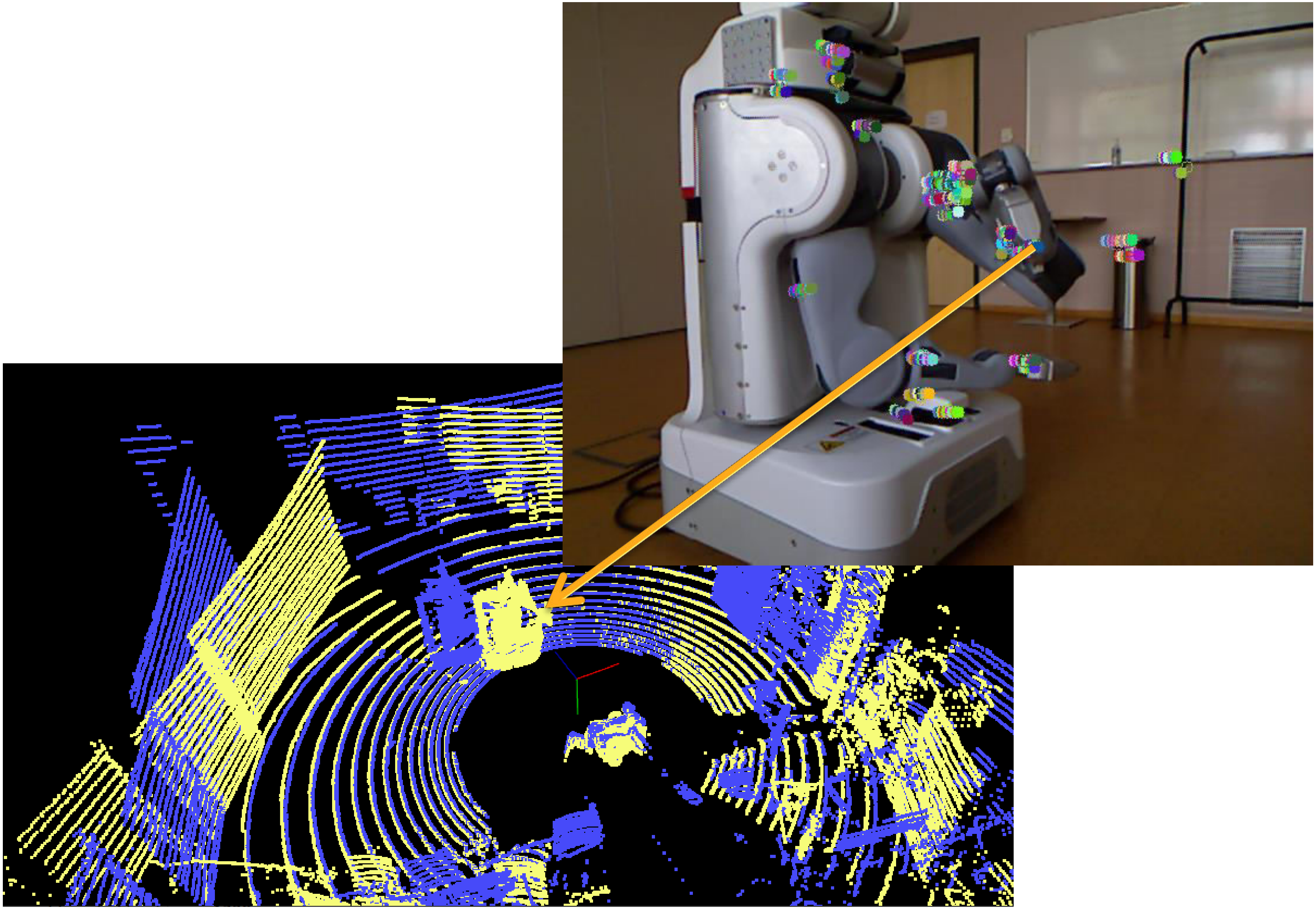
- Captured data are really large (7M pt/sec)
  - 20s record ~1GB
- Velodyne point cloud are very sparse
  - Large „gaps“ between the rings



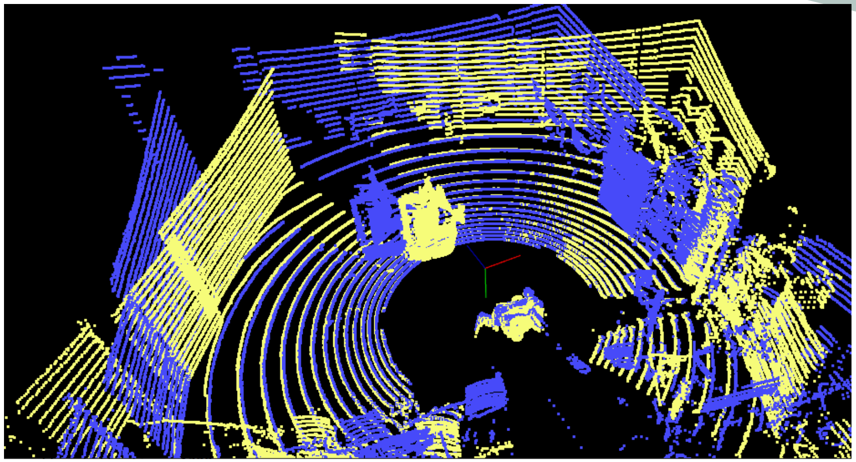
How to extract features?

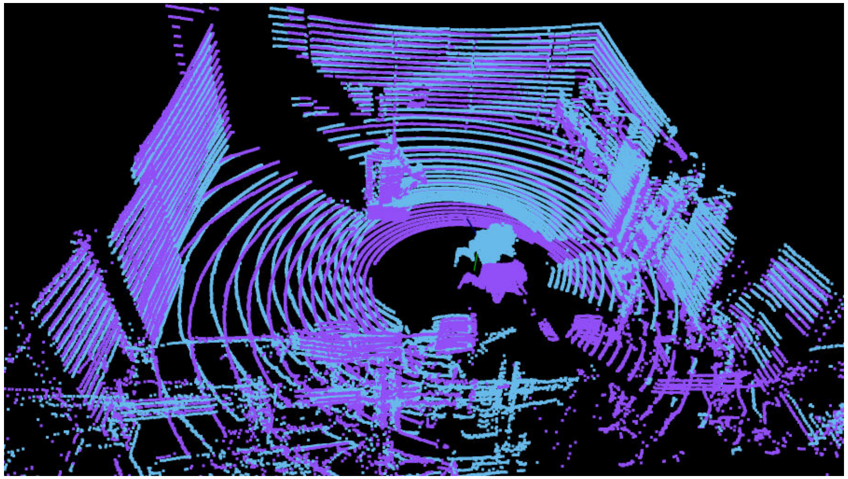
* We want to describe the point cloud by features
* 3D features:
  * Commonly based on the normals
  * How to compute the normal?
  * The neighbourhood of the point are the points of the ring
* Fusion with other data could help
  * Color, image features, ...





How to align the point clouds?

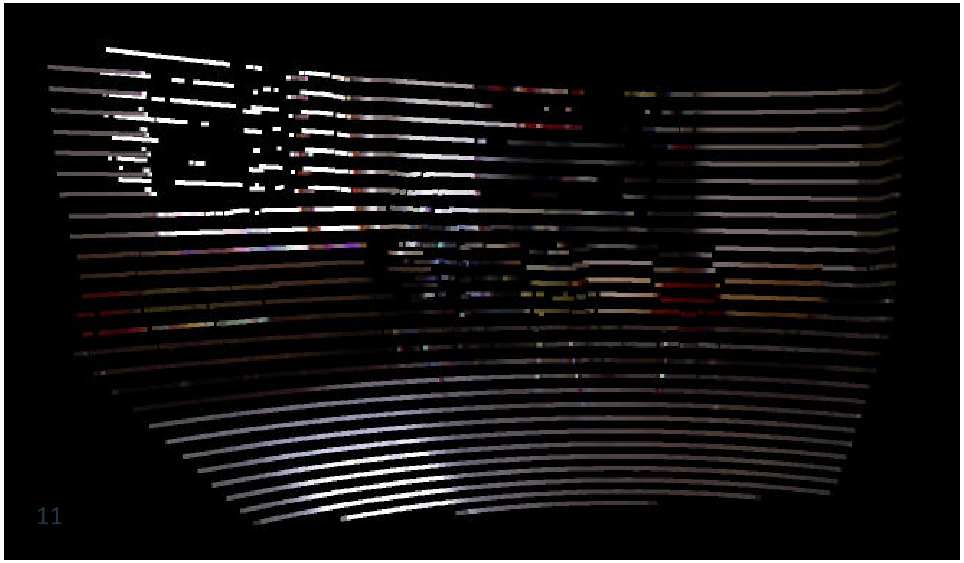




- We want to merge (register) multiple clouds
  - For 3D reconstruction & mapping
- Data are large – performance?
- Typical ICP approach fails
  - Rings will „fit“ to each other

How to calibrate Velodye with other sensors?

- Different modalities
- With other 3D scanner (Kinect, ToF camera, ...)
  - Dense point clouds of small area
  - Normals can be easily computed
- With **RGB camera**
  - Colored point clouds
  - Image feature for description

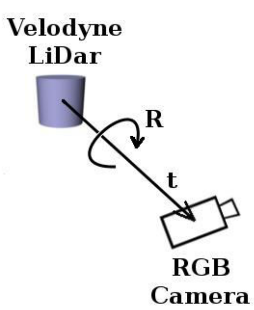


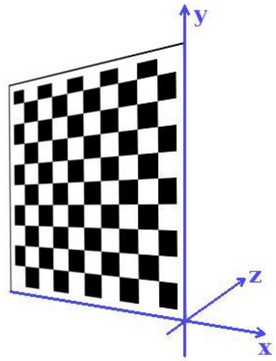
Camera-Velodyne calibration (1/2)

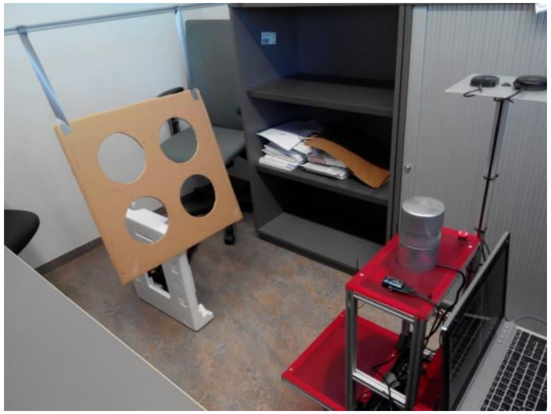
- Already solved:

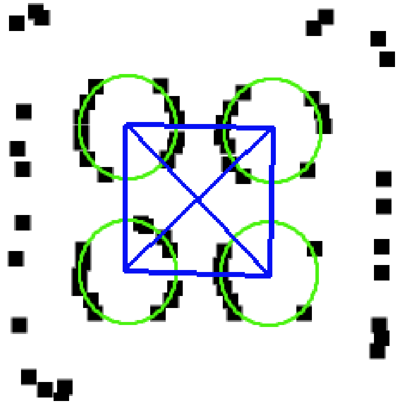
  Calibration of RGB Camera With Velodyne LiDAR

  M. Veľas, M. Španěl, Z. Materna, A. Herout
- Edges – easily detected in the image and point cloud
- Novel 3D marker and it‘s detection algorithm



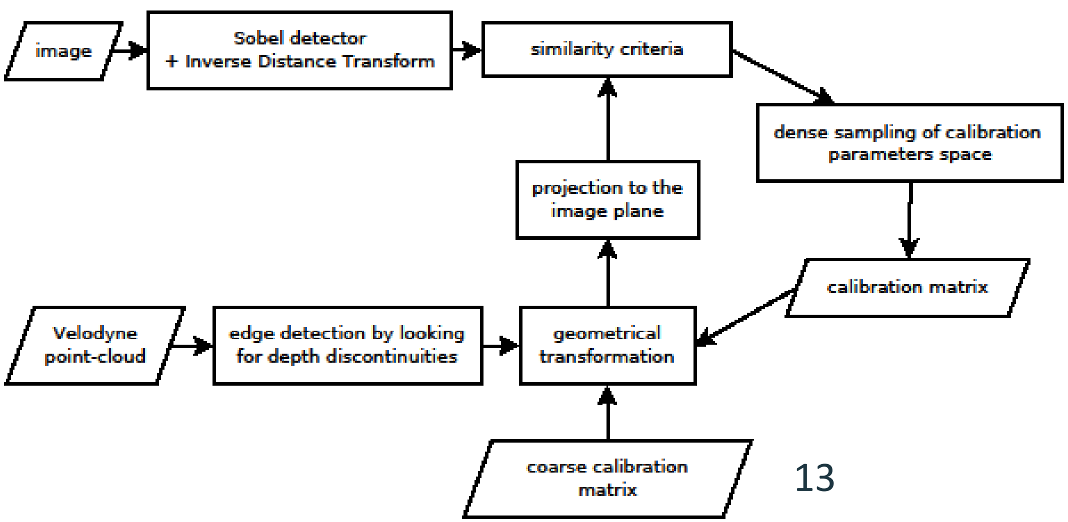


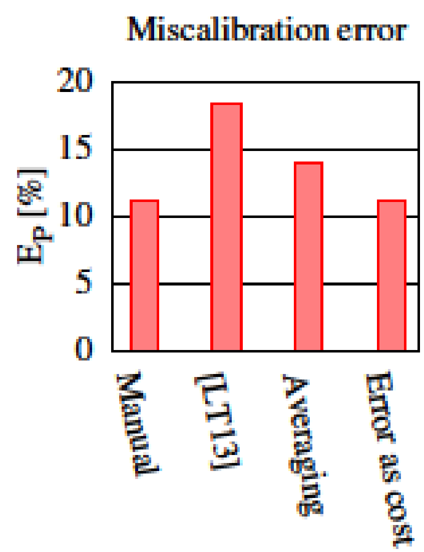




Camera-Velodyne calibration (2/2)

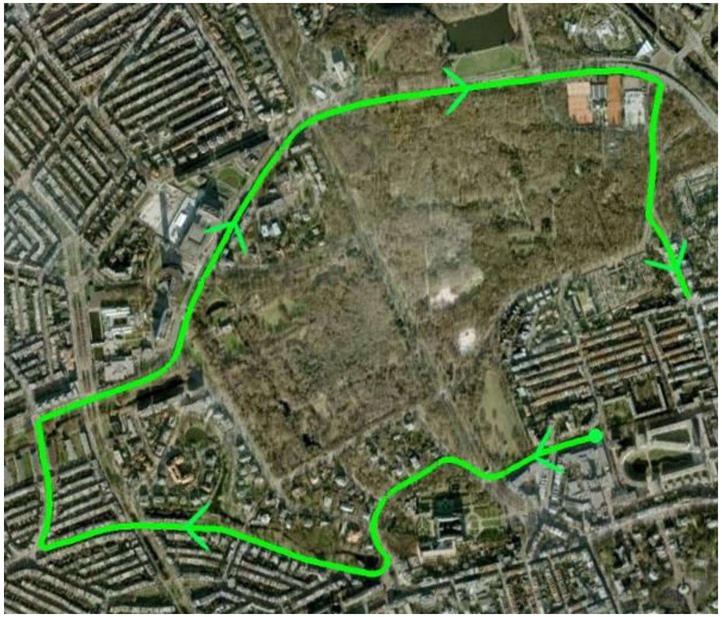
- 3D marker is used for the coarse calibration
- Refinement:
  - Grid search in small subspace of calibration parameters
  - 5% lower miscalibration error as [LT13]



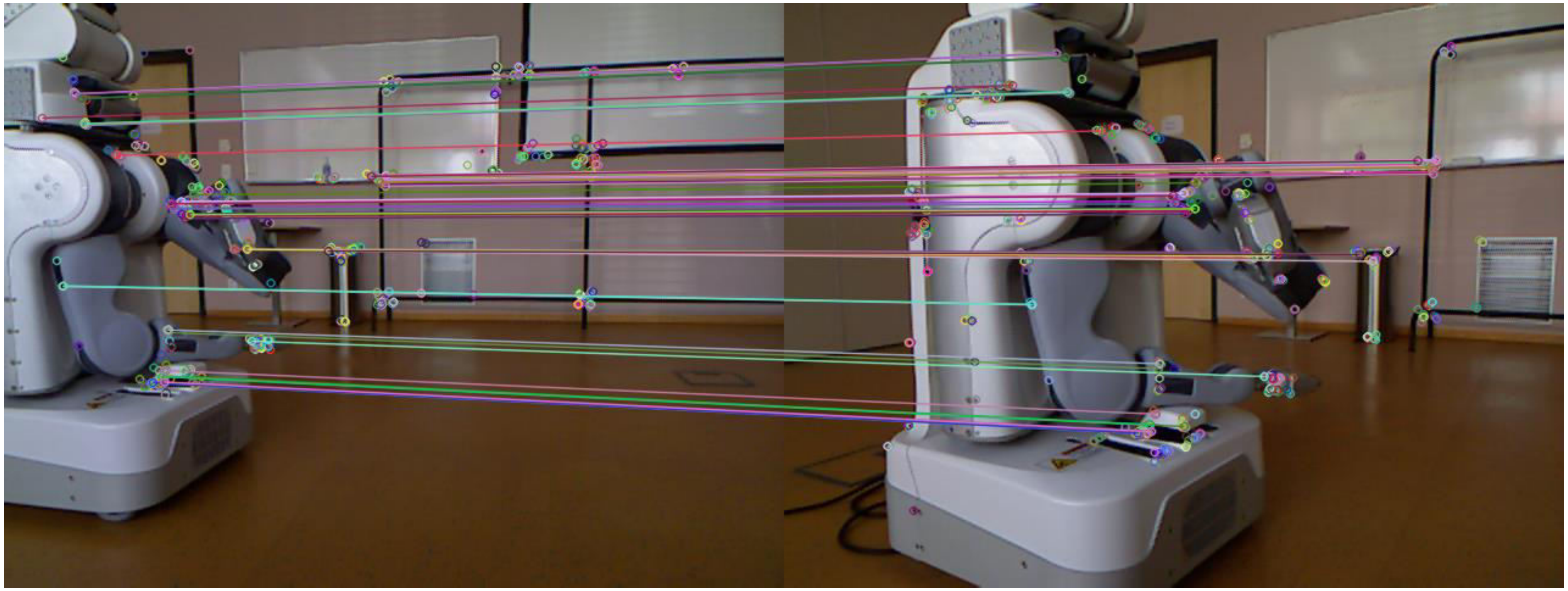


3D Visual odometry (1/5)

* We need to register (merge) multiple point clouds
* Visual odometry
  * How the robot was moving?
  * Pure:
    * Only camera images
    * 2D features
  * With Velodyne
    * Depth information

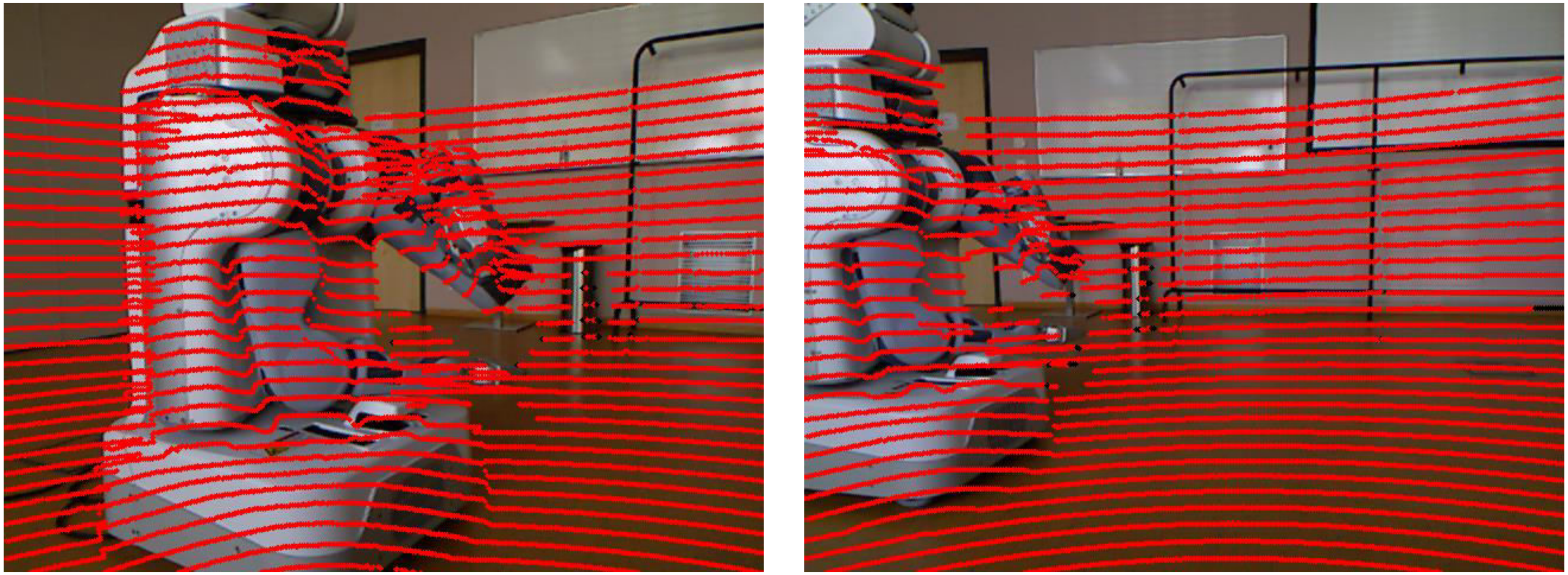


3D Visual odometry (2/5)



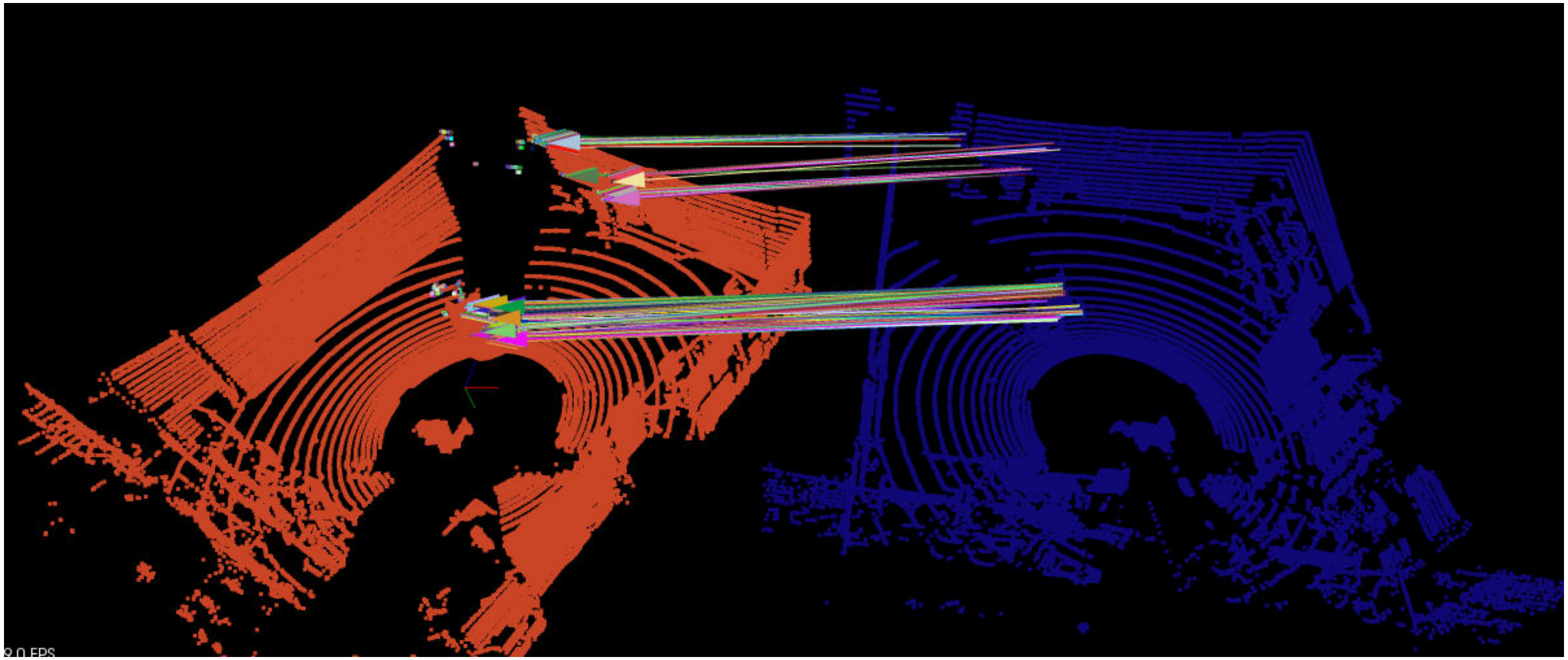
- Image features detection & matching
- RANSAC inliers of homography

3D Visual odometry (3/5)



- Projection of the point cloud on the image plane
- Assuming the calibration and the projection matrix are given

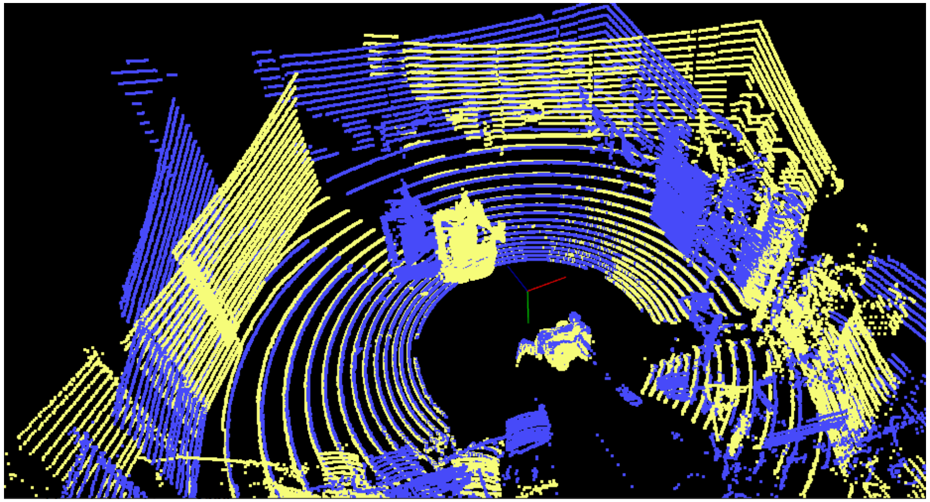
3D Visual odometry (4/5)

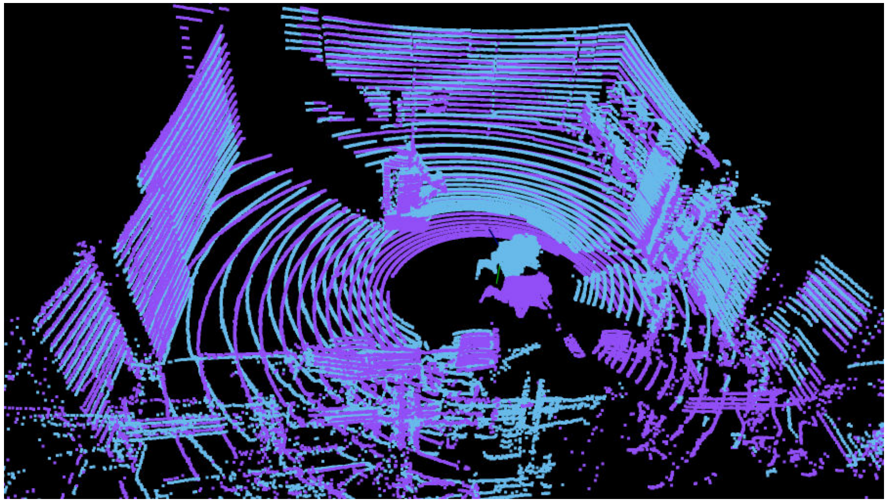


- 2D image correspondences assigned to the closest projected 3D point from Velodyne scan

3D Visual odometry (5/5)

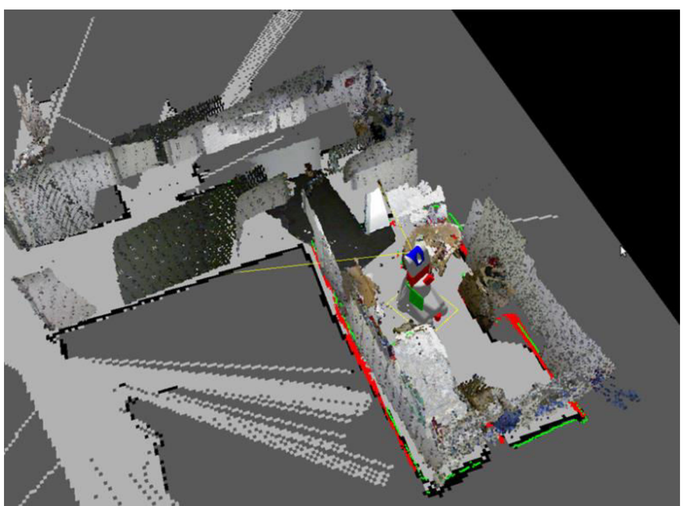
* From correspondences trajectory of the robot can be computed
  * EVD
* This trajectory can be used for point clouds registration

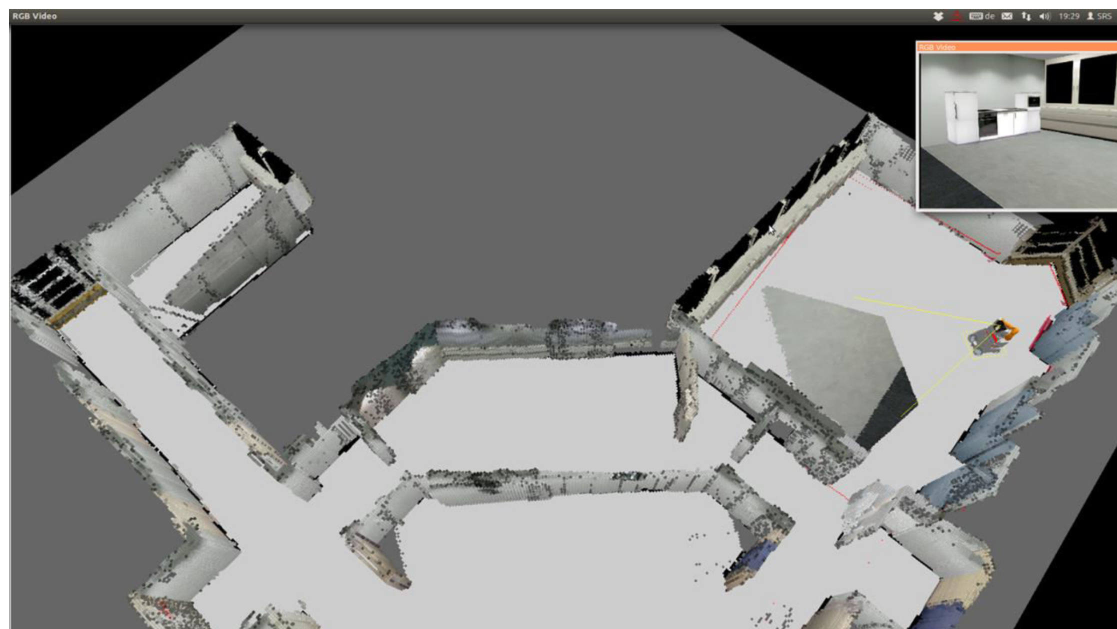




Building Large-Scale 3D Environment Maps

- We can:
  - Compute visual odometry
  - SLAM++
  - Building maps of small environments
- We start:
  - Building large maps

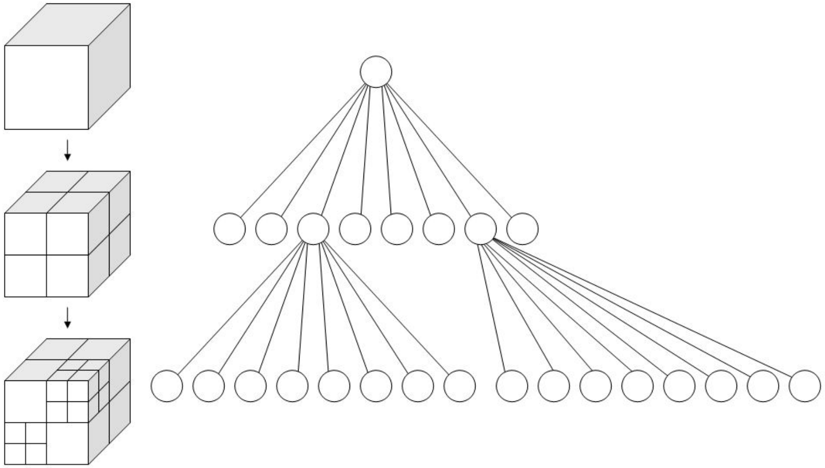




OctoMap

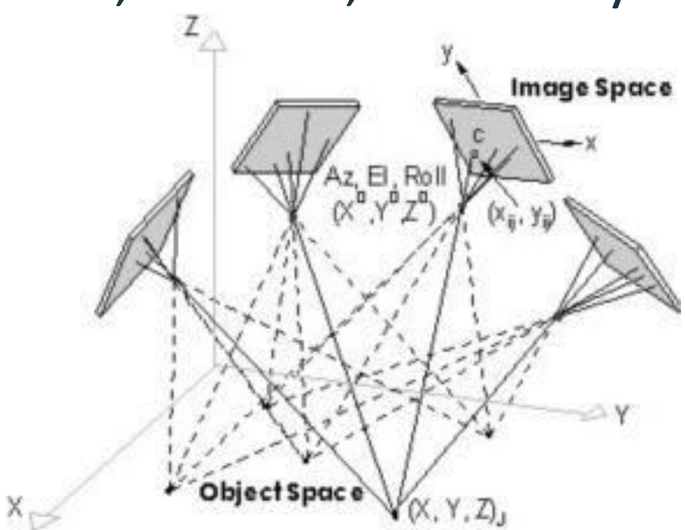
- Hierarchical scene representation
- Dealing with (very) large pointclouds
  - Each one individual pixel can not be storred
- OctoTree structure
  - Leaves = voxels (free/occupied)

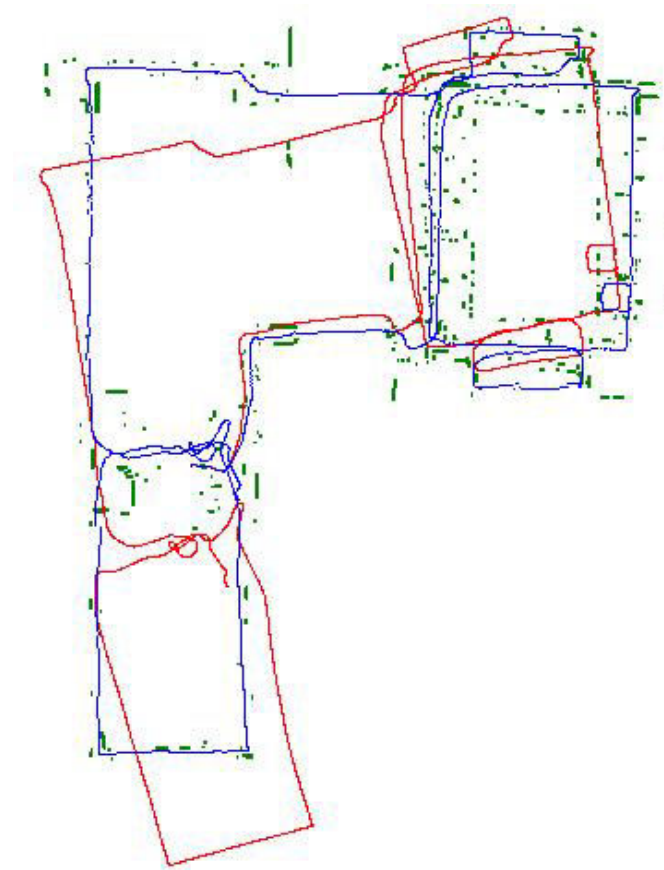
- Our improvements:
  - Better processing and visualisation
  - plugins



Block matrices operations

- Effecient operations with block matrices
  - Acclerated with GPUs
  - Possibly computed by the cluster
- SLAM++ and bundle adjustement
- V. Ila, L. Polok, M. Solony





Change Detection in Large-Scale 3D Maps

* What objects in the scene are moving?
* In OctoMap
  * State of the cell in the grid is not binary
  * Probability of being moving object
  * Updated with each observation
* **Anselm for the data processing** ☺

Robo@FIT