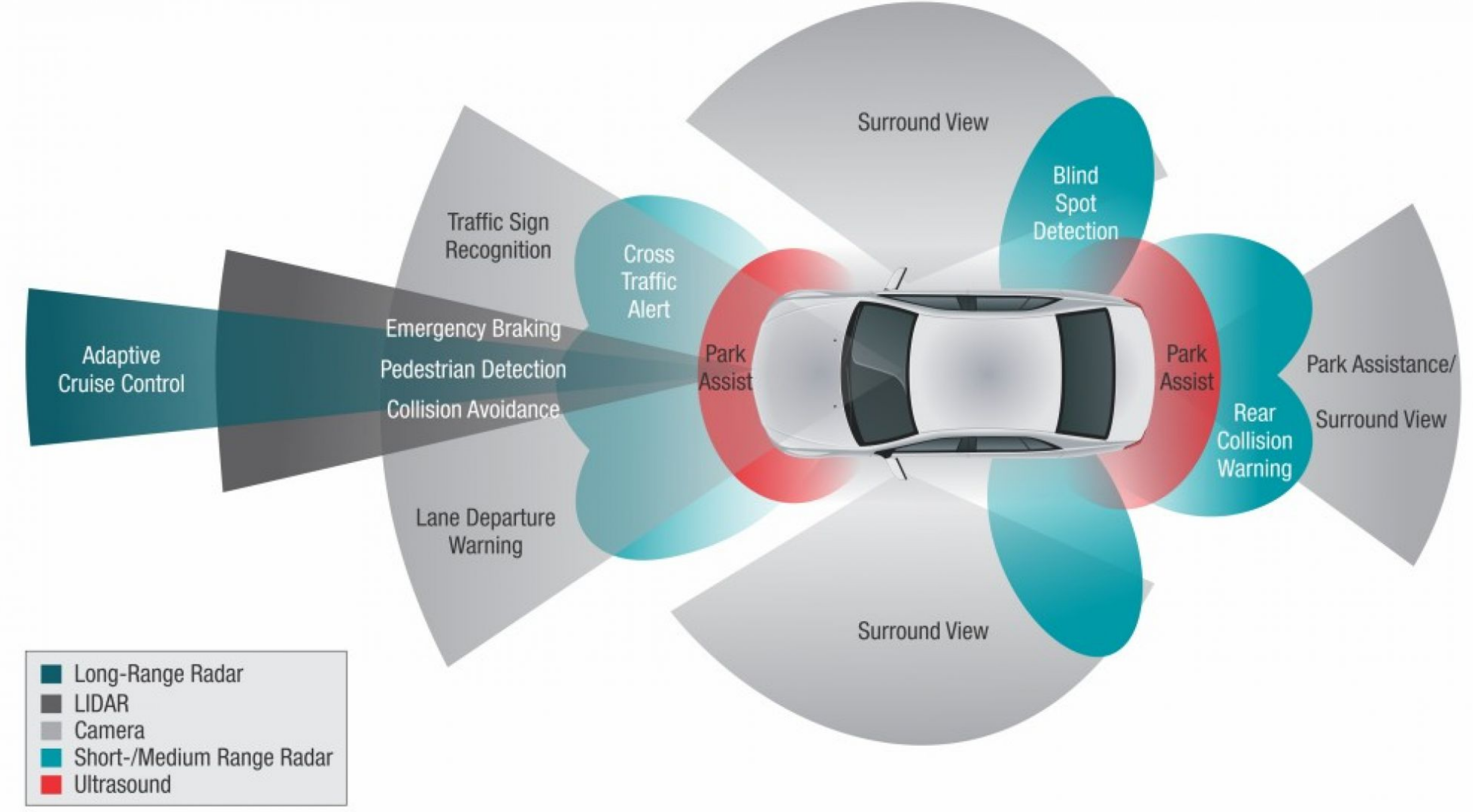
Lidar Odometry and Mapping

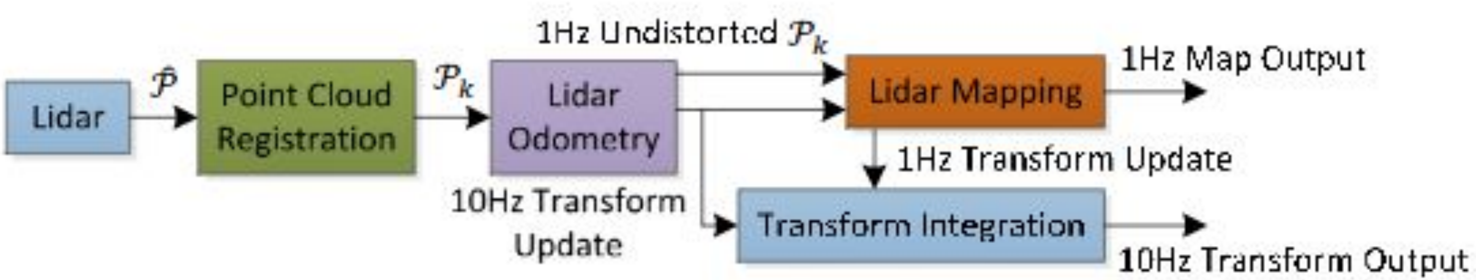
Team 18

Ali Abdallah
Alex Crean
Mohamad Farhat
Alex Groh
Steven Liu
Christopher Wernette


Surround View
Blind Spot Detection
Traffic Sign Recognition
Cross Traffic Alert
Emergency Braking
Adaptive Cruise Control
Pedestrian Detection
Collision Avoidance
Park Assist
Park Assist
Park Assistance/
Surround View
Rear Collision Warning
Lane Departure Warning
Surround View
Long-Range Radar
LIDAR
Camera
Short-/Medium Range Radar
Ultrasound


FRONT

LOAM Overview



- **Real Time, low-drift Odometry and Mapping**
- **Input** - LIDAR + IMU (optional)
- **Output -** 10Hz pose estimate and 1Hz Map
- **Author -** Ji Zhang and Sanjeev Singh (Carnegie Mellon University)

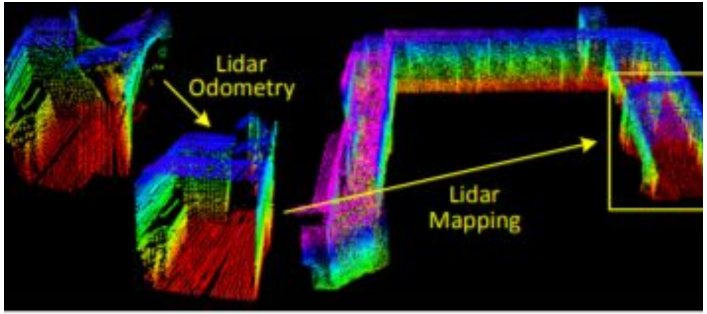


Figure 5: Zhang, J., Singh, S. (2014) LOAM Odometry and Mapping, https://www.ri.cmu.edu/pub_files/2014/7/Ji_LidarMapping_RSS2014_v8.pdf

Implementation

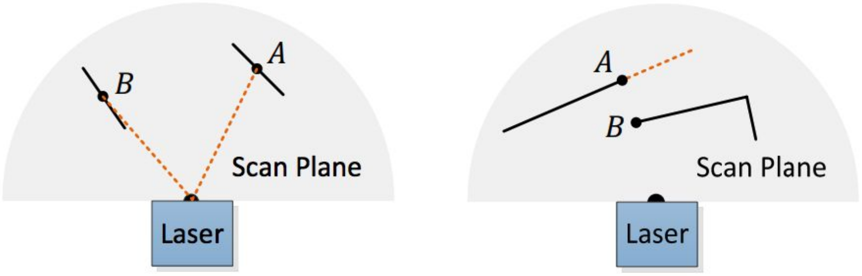
- ROS - Kinetic on Ubuntu 16.04
  - Core #0 - LIDAR Odometry @ 10Hz
  - Core #1 - LIDAR Mapping @ 1Hz
  - RVIZ for visualization
- Benchmark on KITTI odometry dataset
  - Velodyne HDL-64E
  - OXTS RT3003 GPS/IMU
- Replaying using ROS .bag files

LIDAR Feature Extraction

$$c = \frac{1}{|\mathcal{S}| \cdot ||\boldsymbol{X}^L_{(k,i)}||} \left|\left| \sum_{j \in \mathcal{S}, j \neq i} (\boldsymbol{X}^L_{(k,i)} - \boldsymbol{X}^L_{(k,j)}) \right|\right|.$$

Locations of each point

set of consecutive points of i returned by the laser scanner



(a) (b)

Feature Selection Process

1. Divide Point Cloud into Subregions
2. Calculate curvature in subregions
3. Features = sharpest edges and flatest planes
4. Reject planes parallel to beam and occluded edges

LIDAR Odometry

- Point to line correspondence
- Point to plane correspondence
- L-M iterative method to solve nonlinear LS problem

$$d_{\mathcal{E}} = \frac{\left|(\tilde{X}^L_{(k,i)} - \bar{X}^L_{(k-1,j)}) \times (\tilde{X}^L_{(k,i)} - \bar{X}^L_{(k-1,l)})\right|}{\left|\bar{X}^L_{(k-1,j)} - \bar{X}^L_{(k-1,l)}\right|}, \quad (2)$$

$$d_{\mathcal{H}} = \frac{\left|\begin{array}{c}(\tilde{X}^L_{(k,i)} - \bar{X}^L_{(k-1,j)}) \\ ((\bar{X}^L_{(k-1,j)} - \bar{X}^L_{(k-1,l)}) \times (\bar{X}^L_{(k-1,j)} - \bar{X}^L_{(k-1,m)}))\end{array}\right|}{\left|(\bar{X}^L_{(k-1,j)} - \bar{X}^L_{(k-1,l)}) \times (\bar{X}^L_{(k-1,j)} - \bar{X}^L_{(k-1,m)})\right|}. \quad (3)$$

Algorithm 1: Lidar Odometry

input : $\bar{\mathcal{P}}_k$, $\mathcal{P}_{k+1}$, T^L_{k+1} from the last recursion
output : $\bar{\mathcal{P}}_{k+1}$, newly computed T^L_{k+1}
begin
    if at the beginning of a sweep **then**
        $T^L_{k+1} \leftarrow \mathbf{0}$;
    end
    Detect edge points and planar points in $\mathcal{P}_{k+1}$, put the points in $\mathcal{E}_{k+1}$ and $\mathcal{H}_{k+1}$, respectively;
    for a number of iterations **do**
        for each edge point in $\mathcal{E}_{k+1}$ **do**
            Find an edge line as the correspondence, then compute point to line distance based on (9) and stack the equation to (11);
        end
        for each planar point in $\mathcal{H}_{k+1}$ **do**
            Find a planar patch as the correspondence, then compute point to plane distance based on (10) and stack the equation to (11);
        end
        Compute a bisquare weight for each row of (11);
        Update T^L_{k+1} for a nonlinear iteration based on (12);
        if the nonlinear optimization converges **then**
            Break;
        end
    end
    if at the end of a sweep **then**
        Reproject each point in $\mathcal{P}_{k+1}$ to t_{k+2} and form $\bar{\mathcal{P}}_{k+1}$;
        Return T^L_{k+1} and $\bar{\mathcal{P}}_{k+1}$;
    end
    else
        Return T^L_{k+1};
    end
end

Lidar Mapping

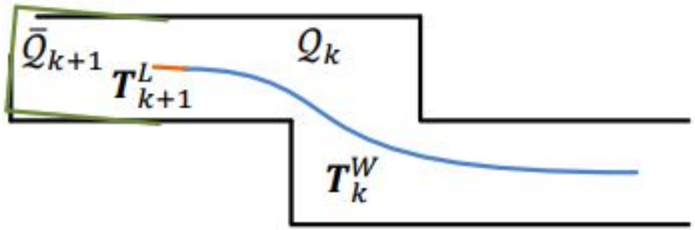
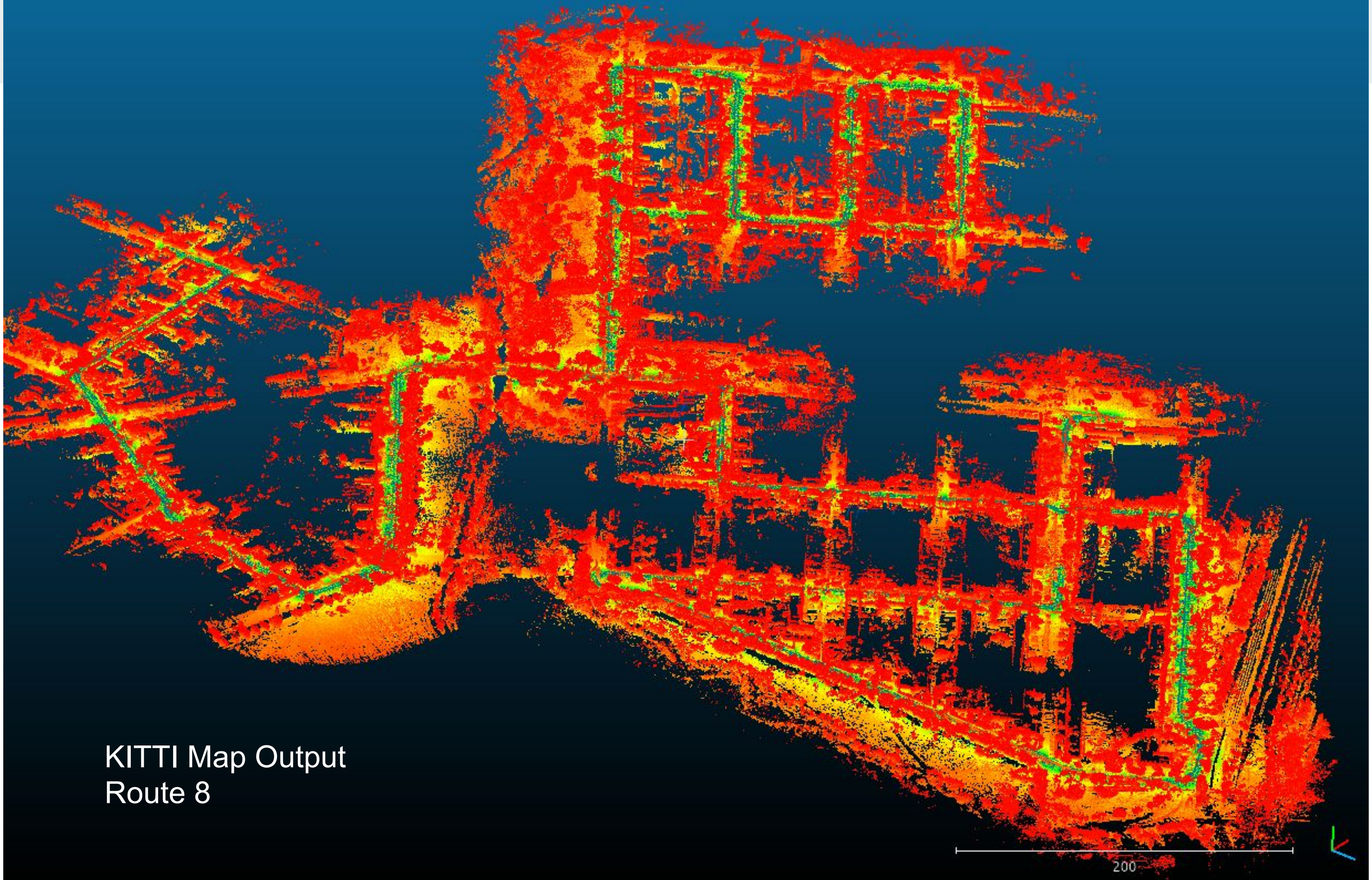
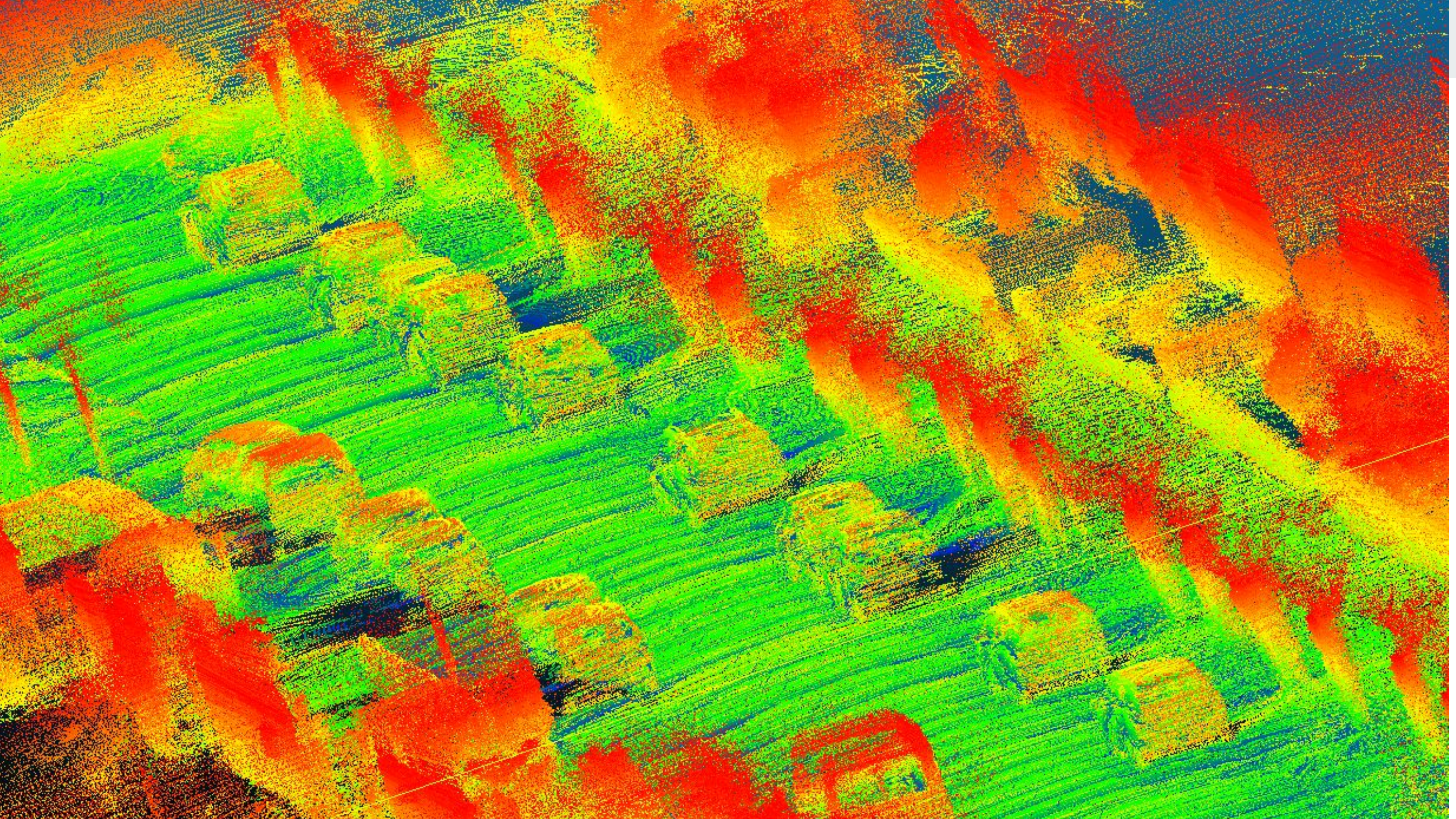
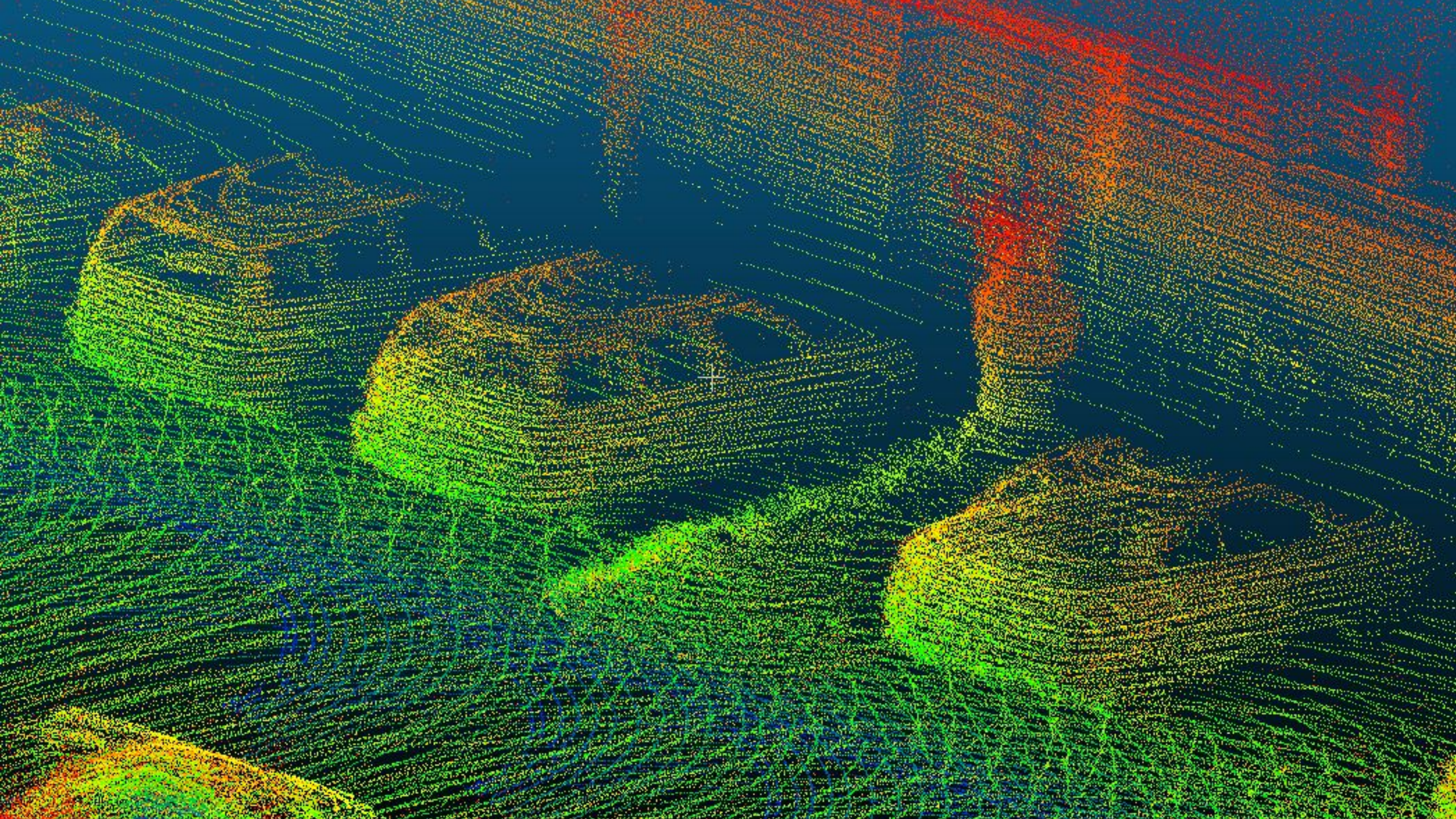


Fig. 8. Illustration of mapping process. The blue colored curve represents the lidar pose on the map, $\boldsymbol{T}_k^W$, generated by the mapping algorithm at sweep k. The orange color curve indicates the lidar motion during sweep $k+1$, $\boldsymbol{T}_{k+1}^L$, computed by the odometry algorithm. With $\boldsymbol{T}_k^W$ and $\boldsymbol{T}_{k+1}^L$, the undistorted point cloud published by the odometry algorithm is projected onto the map, denoted as $\bar{\mathcal{Q}}_{k+1}$ (the green colored line segments), and matched with the existing cloud on the map, $\mathcal{Q}_k$ (the black colored line segments).

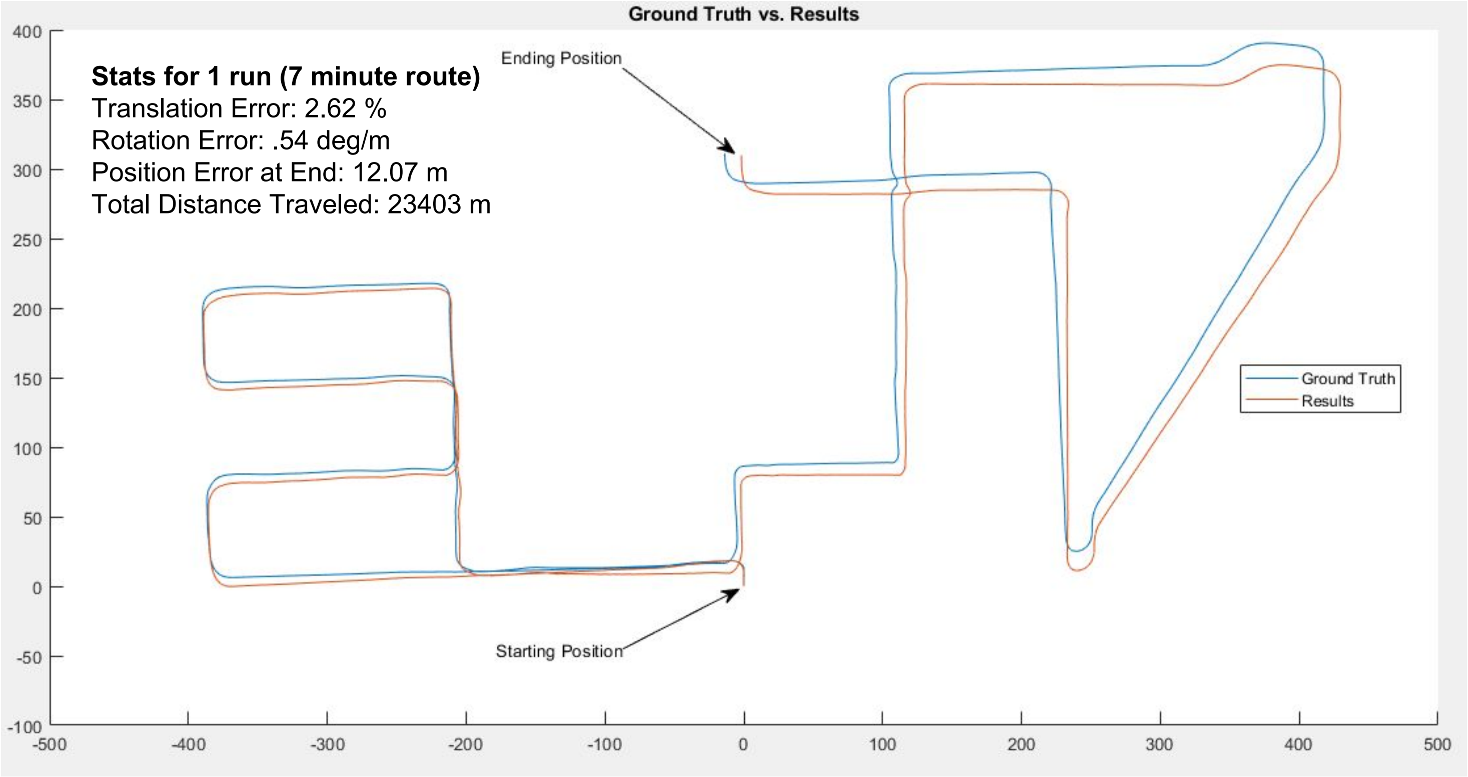


KITTI Map Output
Route 8





Validation: Comparing Odometry to Ground Truth



Challenges and Conclusions

- Replayed bag files below real time to get accurate odometry and mapping
  - Computational Requirements
    - One core for mapping, one core for odometry
    - RViz uses a core and causes odometry frame drop on dual core machine
- Logistics of large data files
  - 7 minute drive is 29GB
- Map building via lidar is highly accurate
  - Error in mapping from odometry
  - Still subject to drift/incorrect initialization
- Until vision solutions are more robust, lidar is necessary for autonomous vehicles

Next Steps

- VLOAM
    - Research grade IMU cost $250k
        - Production grade IMU sensors very noisy
    - 60hz camera odometry accurately captures high frequency motion
    - Use camera odometry as motion prior with LOAM
        - Camera + LIDAR + (noisy) IMU sensor fusion
- Loop Closure
- Publish final copy of code/paper to our Github and make public: https://github.com/stevenliu216/568-Final-Project

Q&A

Backup

ROS - RQT Graph

- ROS rqt_graph shows relationship between functions and outputs

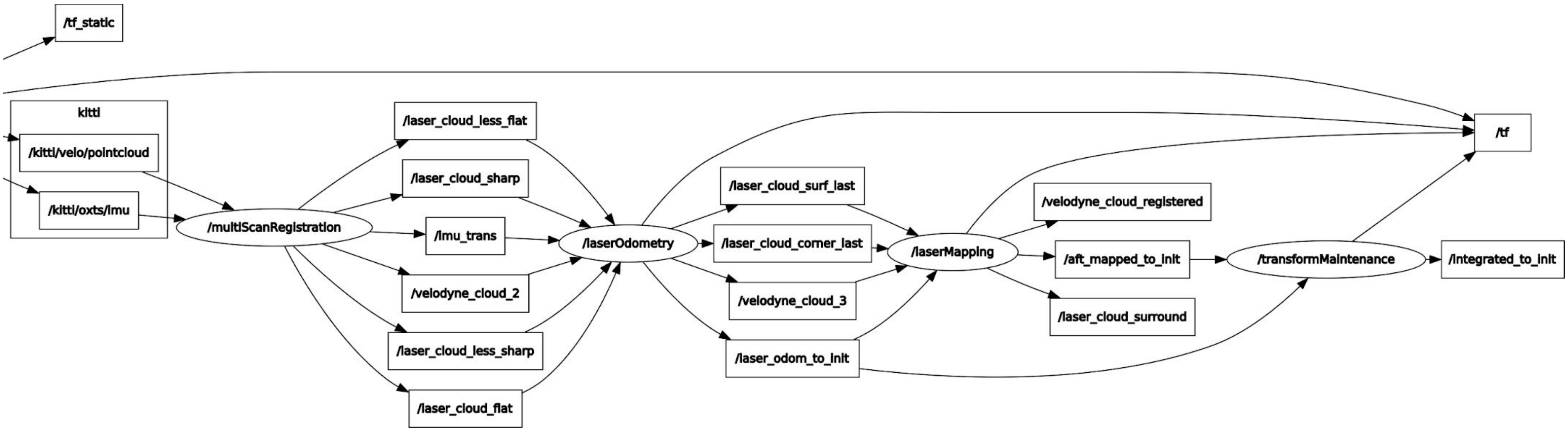


Figure 6: ROS Kinetic Rqt_graph Plot of LOAM System

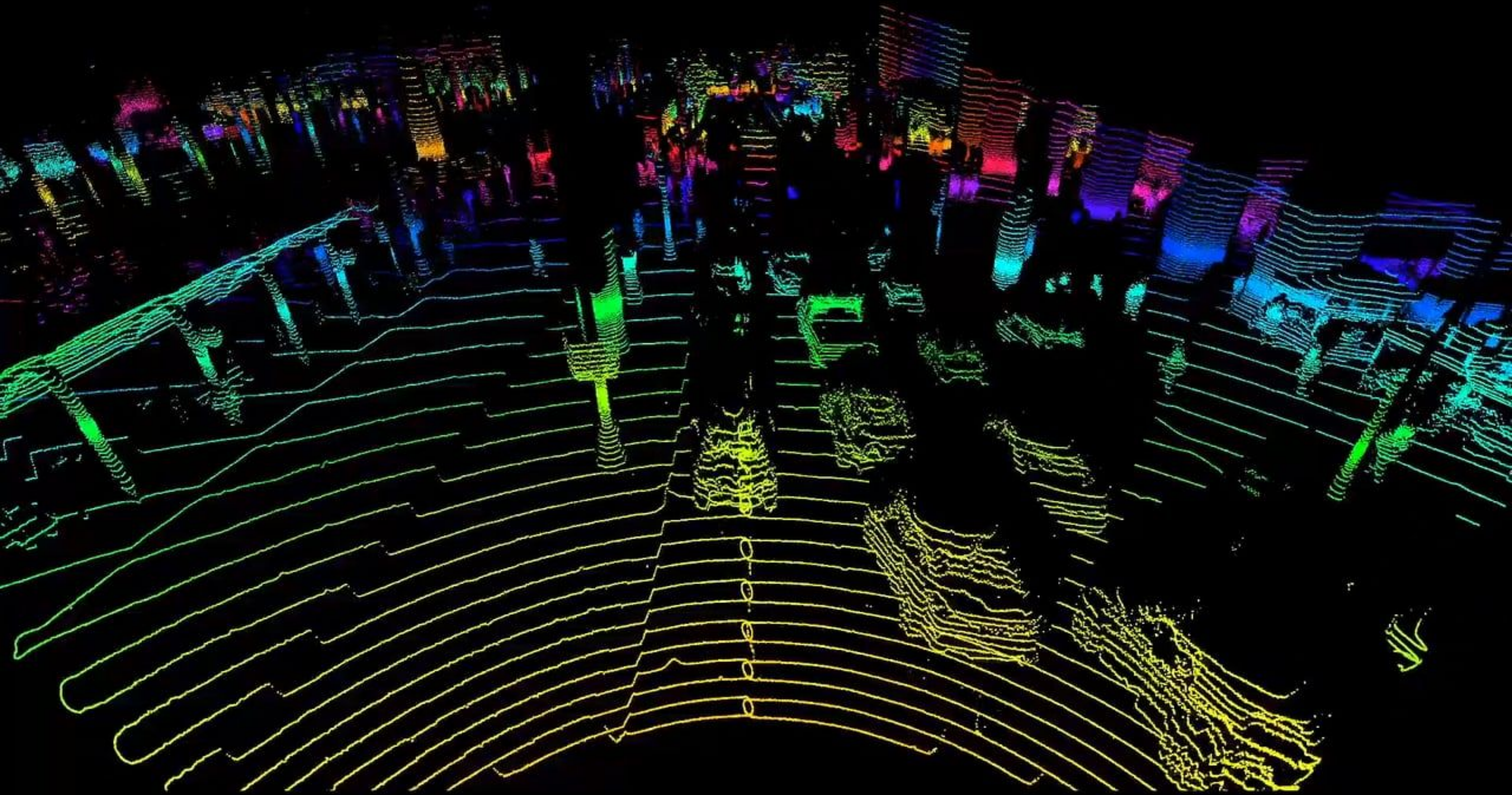
Talk about VLOAM + LOAM here?

Think it could be good to say we wanted to combine lidar and camera but we could not get the computer vision and lidar working at the same time in ros on the kitti data set

Standard AV sensor suite will include lidar, camera, radar, ultrasonics

Validation: Visualization of Map

- Runs in semi-real time (would run faster if not in a virtual machine, as well as if we all had graphics cards)
- Hard to benchmark mapping, as ground truth surface/point cloud maps do not exist



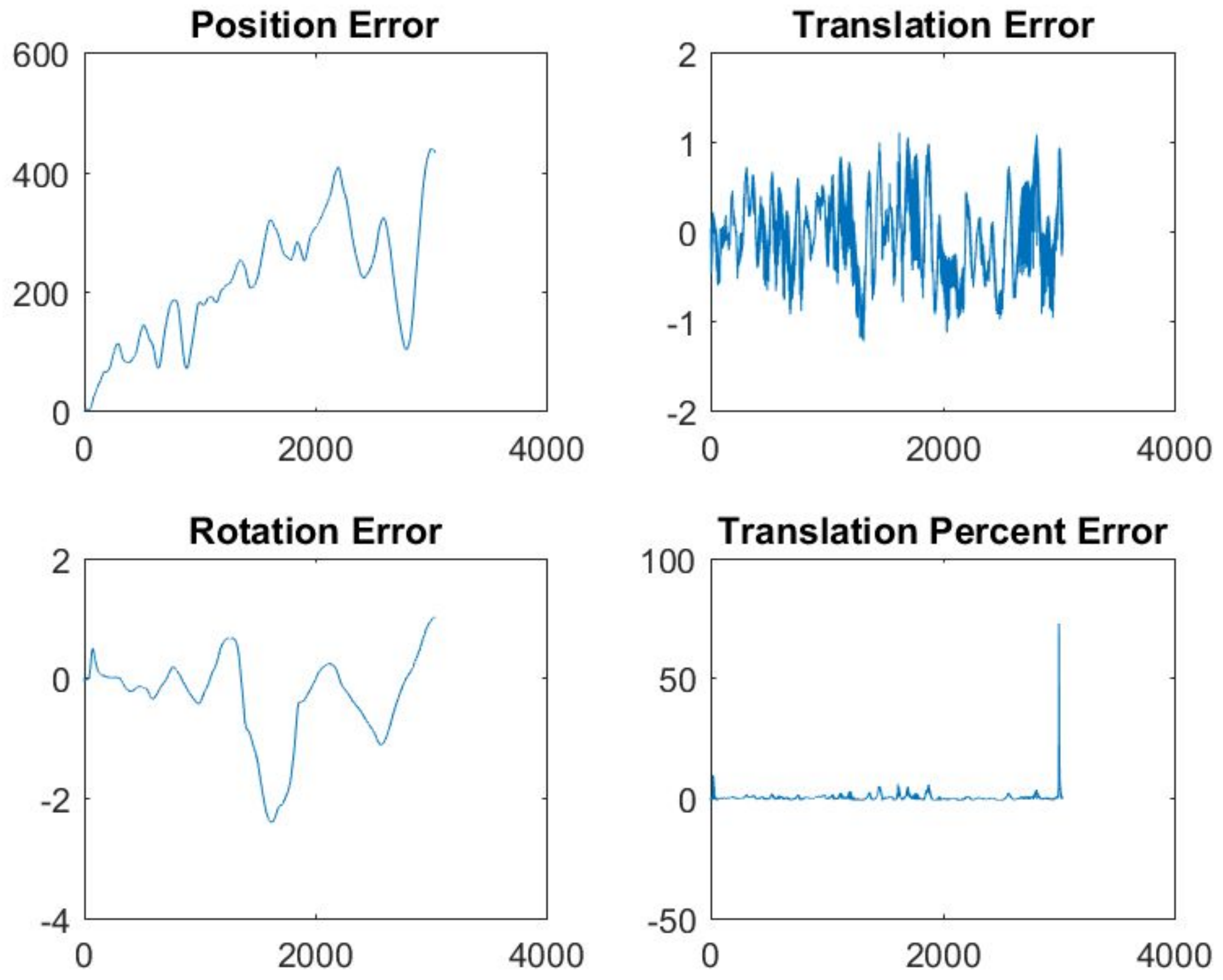
Lidar Odometry backup - sensors

Fig. 2. The 3D lidar used in this study consists of a Hokuyo laser scanner driven by a motor for rotational motion, and an encoder that measures the rotation angle. The laser scanner has a field of view of 180° with a resolution of 0.25°. The scan rate is 40 lines/sec. The motor is controlled to rotate from −90° to 90° with the horizontal orientation of the laser scanner as zero.

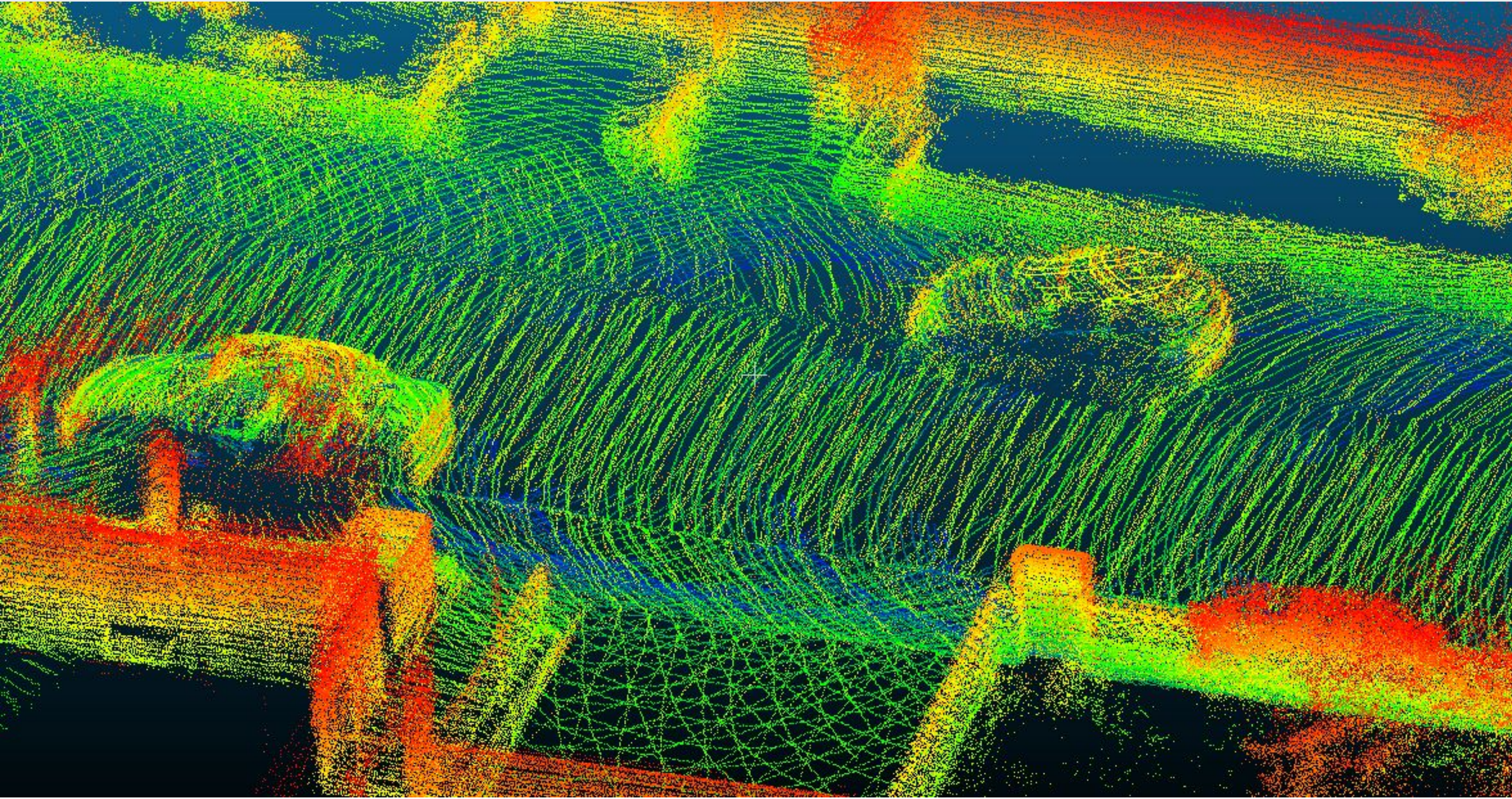
64 Channels, 120m range, 2.2 Million Points per Second, 360° Horizontal FOV, 26.9° Vertical FOV, 0.08° angular resolution (azimuth), ~0.4° Vertical Resolution

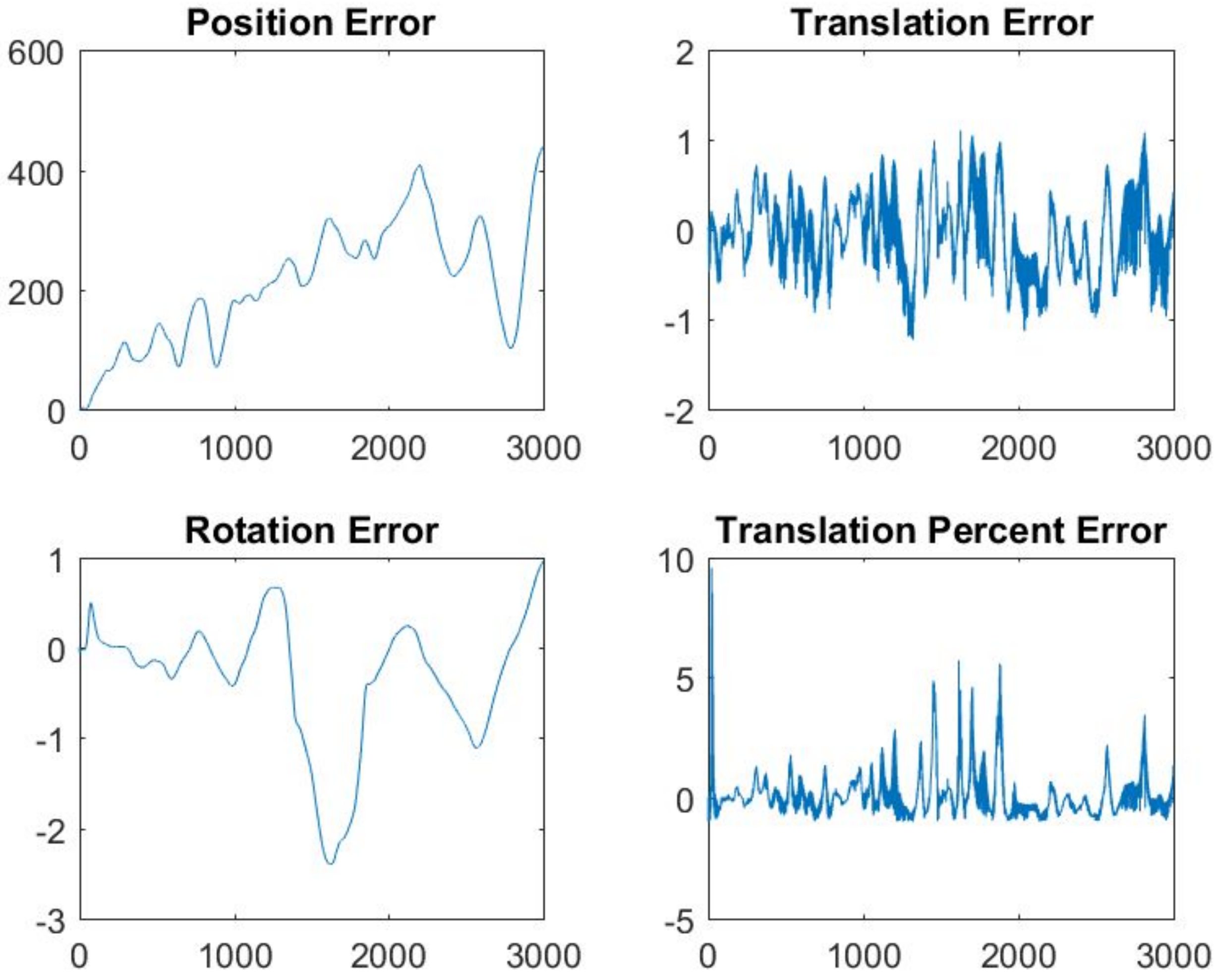
Lidar Odometry backup - optimization formulation

Lidar Mapping backup



Position Error
600
400
200
0
0
2000
4000
Translation Error
2
1
0
-1
-2
0
2000
4000
Rotation Error
2
0
-2
-4
0
2000
4000
Translation Percent Error
100
50
0
-50
0
2000
4000





Position Error
600
400
200
0
0
1000
2000
3000
Translation Error
2
1
0
-1
-2
0
1000
2000
3000
Rotation Error
1
0
-1
-2
-3
0
1000
2000
3000
Translation Percent Error
10
5
0
-5
0
1000
2000
3000