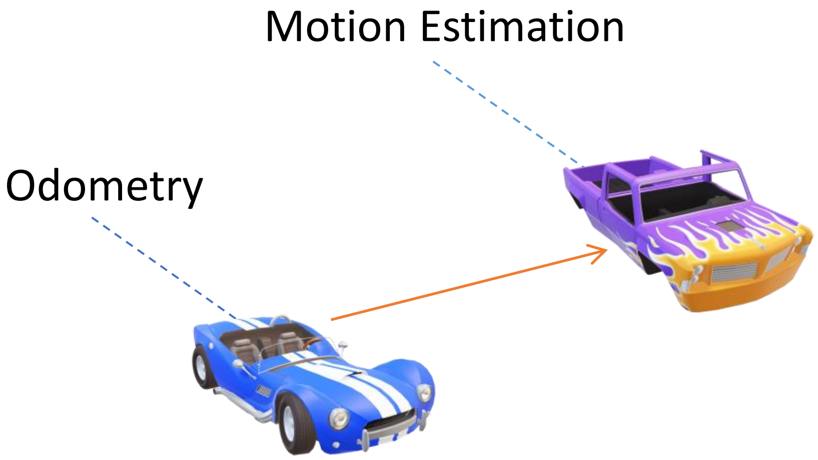
Instance Completion and Motion Estimation with Deep Shape Priors for Autonomous Driving



Team members

Panyawat (Ohm) Rattana

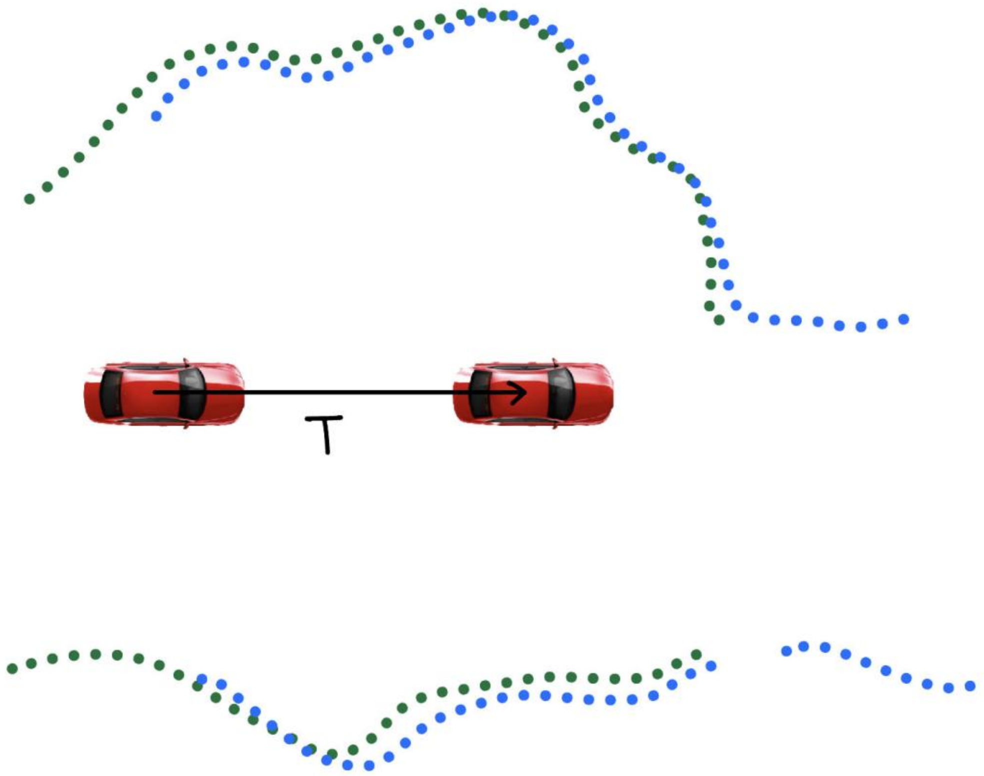
Shashank (Sai) Dammalapati

Supervisors

Xingguang (Starry) Zhong

Yue Pan

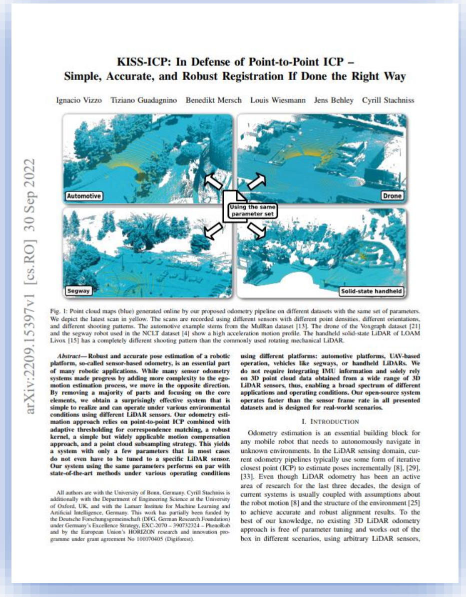
Lidar Odometry (KISS ICP)



Motion Estimation of Ego Car

"Keep it small and simple"

- Accurately compute a robot's pose.
- Point cloud alignment.
- A few parameters tuning.
- Assumption: The output of KISSICP is accurate.



arXiv:2209.15397v1 [cs.RO] 30 Sep 2022

KISS-ICP: In Defense of Point-to-Point ICP –
Simple, Accurate, and Robust Registration If Done the Right Way

Ignacio Vizzo Tiziano Guadagnino Benedikt Mersch Louis Wiesmann Jens Behley Cyrill Stachniss

Automotive
Drone
Using the same parameter set
Segway
Solid-state handheld

Fig. 1: Point cloud maps (blue) generated online 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 [13]. The drone of the Voxgraph dataset [21] and the segway robot used in the NCLT dataset [4] show a high acceleration motion profile. The handheld solid-state LiDAR of LOAM Livox [15] has a completely different shooting pattern than the commonly used rotating mechanical LiDAR.

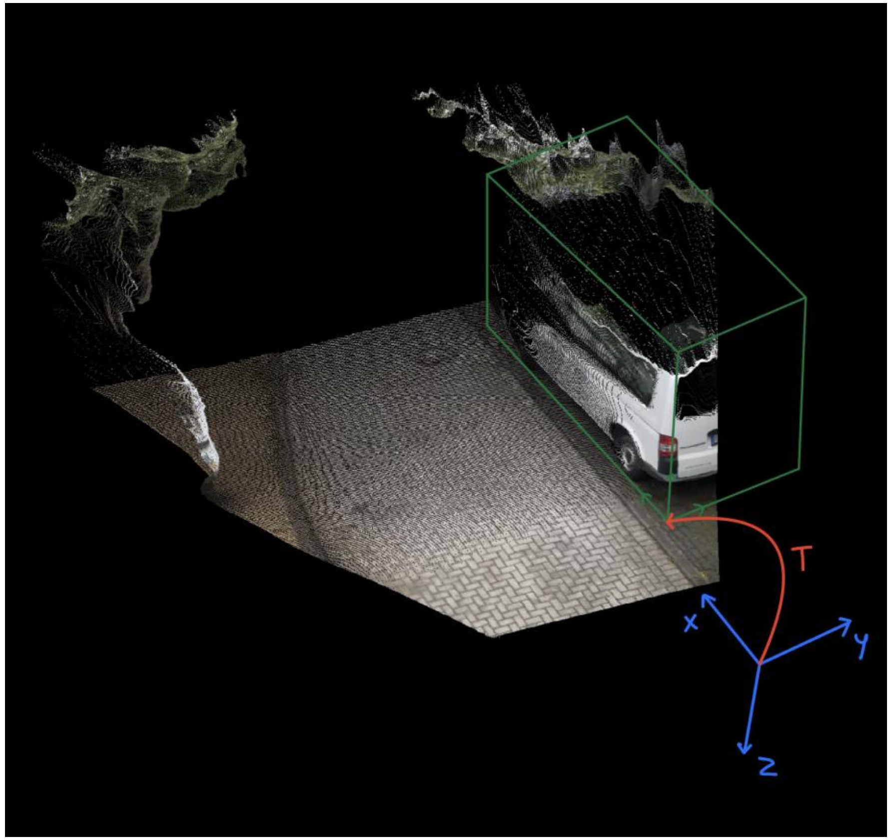
Abstract— **Robust and accurate pose estimation of a robotic platform, so-called sensor-based odometry, is an essential part of many robotic applications. While many sensor odometry systems made progress by adding more complexity to the ego-motion estimation process, we move in the opposite direction. By removing a majority of parts and focusing on the core elements, we obtain a surprisingly effective system that is simple to realize and can operate under various environmental conditions using different LiDAR sensors. Our odometry estimation approach relies on point-to-point ICP combined with adaptive thresholding for correspondence matching, a robust kernel, a simple but widely applicable motion compensation approach, and a point cloud subsampling strategy. This yields a system with only a few parameters that in most cases do not even have to be tuned to a specific LiDAR sensor. Our system using the same parameters performs on par with state-of-the-art methods under various operating conditions using different platforms: automotive platforms, UAV-based operation, vehicles like segways, or handheld LiDARs. We do not require integrating IMU information and solely rely on 3D point cloud data obtained from a wide range of 3D LiDAR sensors, thus, enabling a broad spectrum of different applications and operating conditions. Our open-source system operates faster than the sensor frame rate in all presented datasets and is designed for real-world scenarios.**

All authors are with the University of Bonn, Germany. Cyrill Stachniss is additionally with the Department of Engineering Science at the University of Oxford, UK, and with the Lamarr Institute for Machine Learning and Artificial Intelligence, Germany. This work has partially been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy, EXC-2070 – 390732324 – PhenoRob and by the European Union's HORIZON research and innovation programme under grant agreement No 101070405 (Digiforest).

I. INTRODUCTION

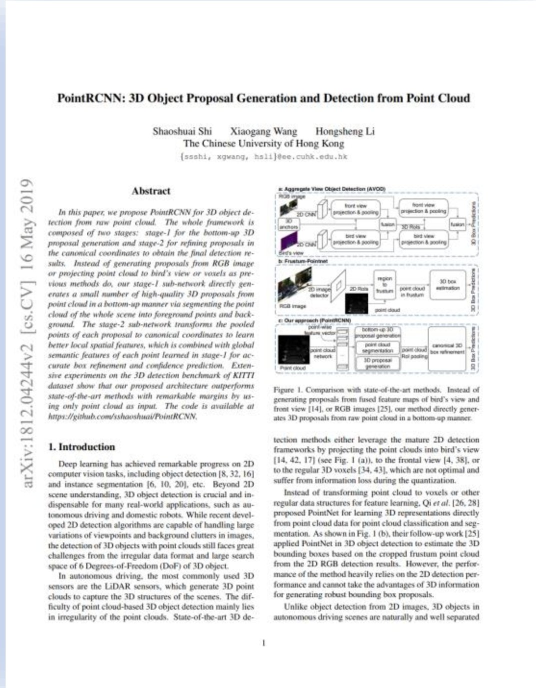
Odometry estimation is an essential building block for any mobile robot that needs to autonomously navigate in unknown environments. In the LiDAR sensing domain, current odometry pipelines typically use some form of iterative closest point (ICP) to estimate poses incrementally [8], [29], [33]. Even though LiDAR odometry has been an active area of research for the last three decades, the design of current systems is usually coupled with assumptions about the robot motion [8] and the structure of the environment [25] to achieve accurate and robust alignment results. To the best of our knowledge, no existing 3D LiDAR odometry approach is free of parameter tuning and works out of the box in different scenarios, using arbitrary LiDAR sensors,

Point RCNN



Pose Estimation of surrounding cars in 1 frame

- Idea is to generate bounding boxes for cars
- Point RCNN gives us an initial guess for the relative pose of the external cars relative to ego car.



arXiv:1812.04244v2 [cs.CV] 16 May 2019

PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud

Shaoshuai Shi Xiaogang Wang Hongsheng Li
The Chinese University of Hong Kong
{sshi, xgwang, hsli}@ee.cuhk.edu.hk

Abstract

In this paper, we propose PointRCNN for 3D object detection from raw point cloud. The whole framework is composed of two stages: stage-1 for the bottom-up 3D proposal generation and stage-2 for refining proposals in the canonical coordinates to obtain the final detection results. Instead of generating proposals from RGB image or projecting point cloud to bird's view or voxels as previous methods do, our stage-1 sub-network directly generates a small number of high-quality 3D proposals from point cloud in a bottom-up manner via segmenting the point cloud of the whole scene into foreground points and background. The stage-2 sub-network transforms the pooled points of each proposal to canonical coordinates to learn better local spatial features, which is combined with global semantic features of each point learned in stage-1 for accurate box refinement and confidence prediction. Extensive experiments on the 3D detection benchmark of KITTI dataset show that our proposed architecture outperforms state-of-the-art methods with remarkable margins by using only point cloud as input. The code is available at https://github.com/sshaoshuai/PointRCNN.

1. Introduction

Deep learning has achieved remarkable progress on 2D computer vision tasks, including object detection [8, 32, 16] and instance segmentation [6, 10, 20], etc. Beyond 2D scene understanding, 3D object detection is crucial and indispensable for many real-world applications, such as autonomous driving and domestic robots. While recent developed 2D detection algorithms are capable of handling large variations of viewpoints and background clutters in images, the detection of 3D objects with point clouds still faces great challenges from the irregular data format and large search space of 6 Degrees-of-Freedom (DoF) of 3D object.

In autonomous driving, the most commonly used 3D sensors are the LiDAR sensors, which generate 3D point clouds to capture the 3D structures of the scenes. The difficulty of point cloud-based 3D object detection mainly lies in irregularity of the point clouds. State-of-the-art 3D detection methods either leverage the mature 2D detection frameworks by projecting the point clouds into bird's view [14, 42, 17] (see Fig. 1 (a)), to the frontal view [4, 38], or to the regular 3D voxels [34, 43], which are not optimal and suffer from information loss during the quantization.

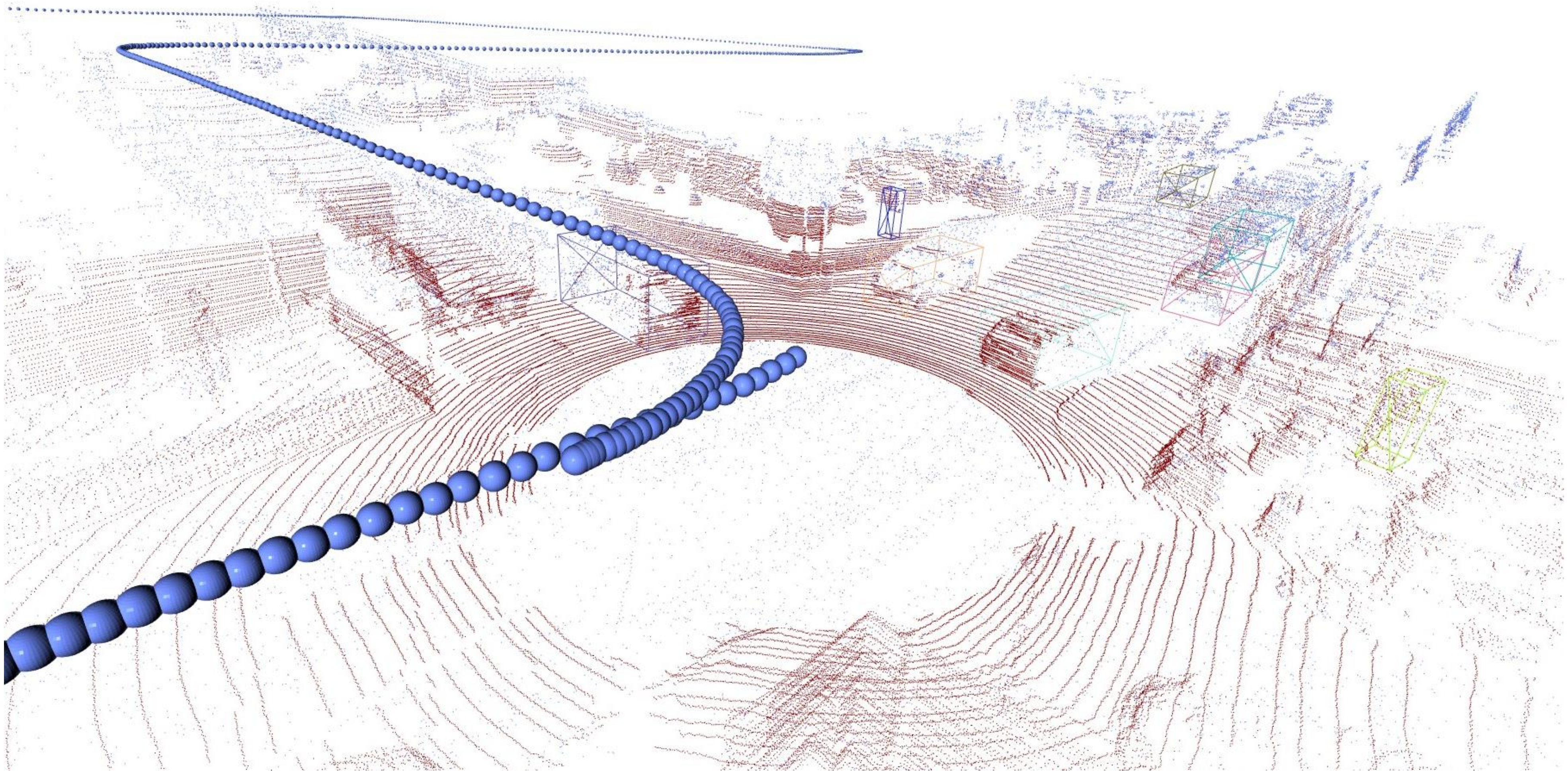
Figure 1. Comparison with state-of-the-art methods. Instead of generating proposals from fused feature maps of bird's view and front view [14], or RGB images [25], our method directly generates 3D proposals from raw point cloud in a bottom-up manner.

Instead of transforming point cloud to voxels or other regular data structures for feature learning, Qi *et al.* [26, 28] proposed PointNet for learning 3D representations directly from point cloud data for point cloud classification and segmentation. As shown in Fig. 1 (b), their follow-up work [25] applied PointNet in 3D object detection to estimate the 3D bounding boxes based on the cropped frustum point cloud from the 2D RGB detection results. However, the performance of the method heavily relies on the 2D detection performance and cannot take the advantages of 3D information for generating robust bounding box proposals.

Unlike object detection from 2D images, 3D objects in autonomous driving scenes are naturally and well separated

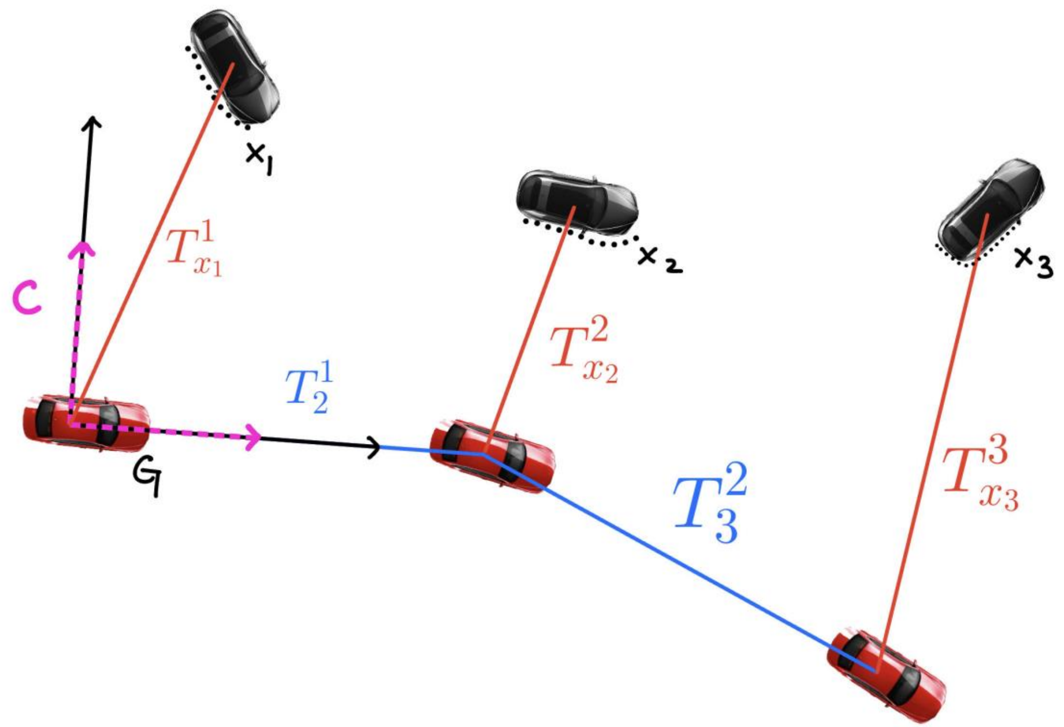
1

Lidar Odometry and 3D Bounding Box Detection



KISS ICP + Point RCNN

Workflow: Transformation



$$X_{1[C]} = T_{x_1}^{1} X_1$$

$$X_{2[C]} = T_2^1 T_{x_2}^{2} X_2$$

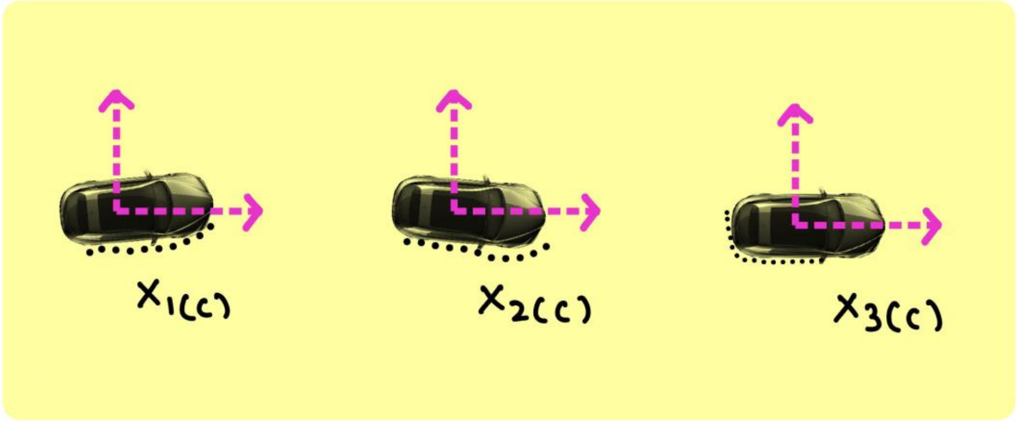
$$X_{3[C]} = T_2^1 T_3^2 T_{x_3}^{3} X_3$$

C – Canonical Frame

G – Global Frame

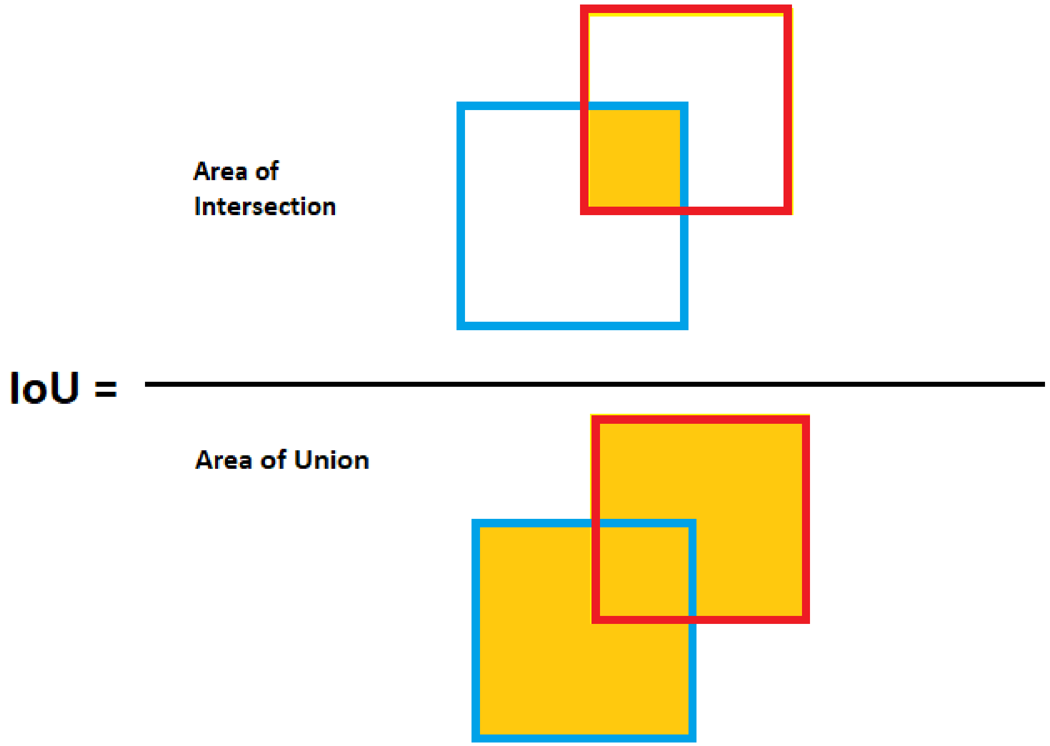
$T_{x_N}^{N}$ Transformation from Point RCNN

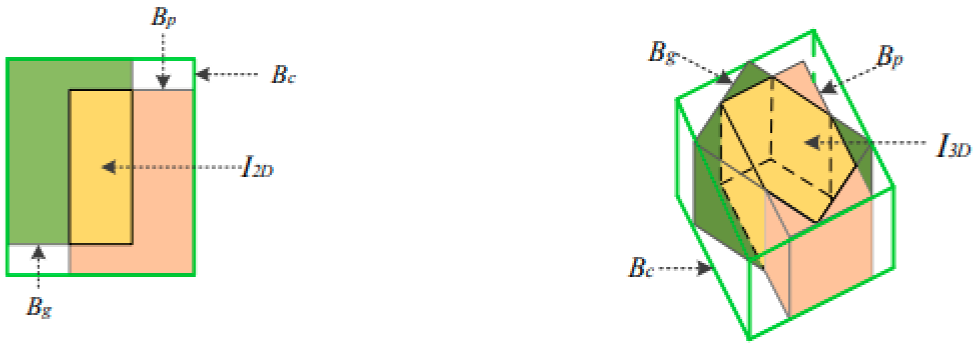
T_N^{N-1} Transformation from KISS ICP



Point Clouds in Canonical Space

IOU: Definition



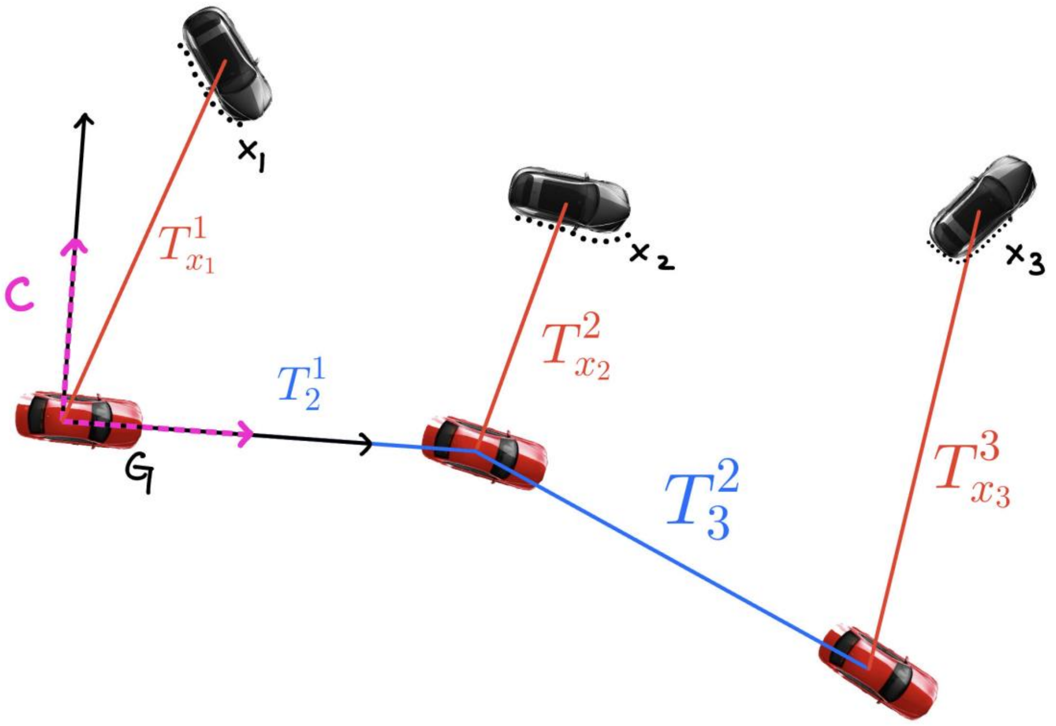


(a) 2D (b) 3D

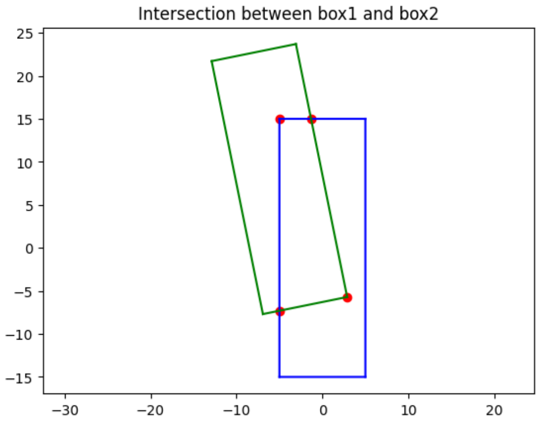
Source: https://debuggercafe.com/evaluation-metrics-for-object-detection/

Video

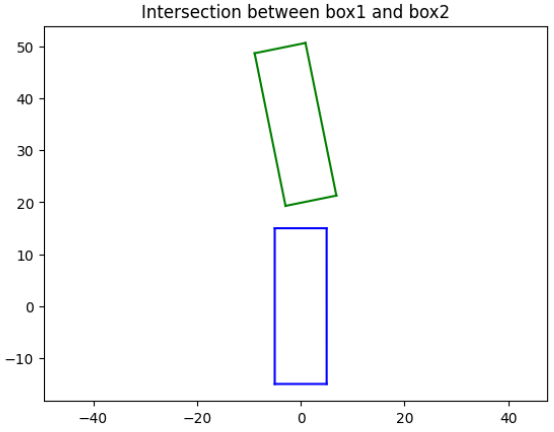
Tracking Multiple Objects



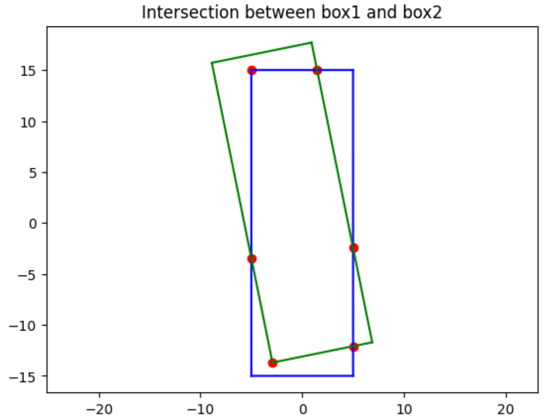
Example of tracking one car using lidar odometry and bounding box measurements



0.266 IOU

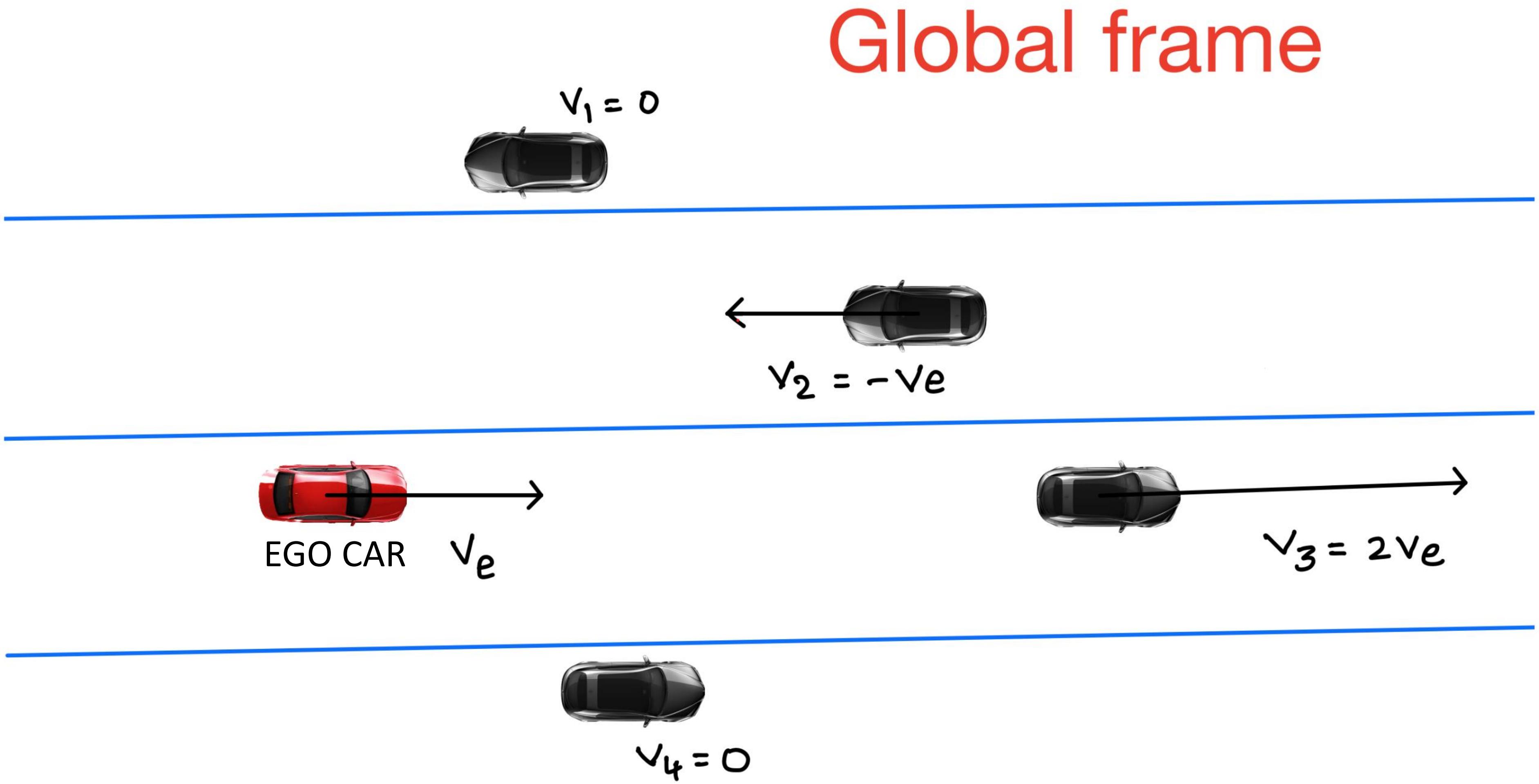


0 IOU

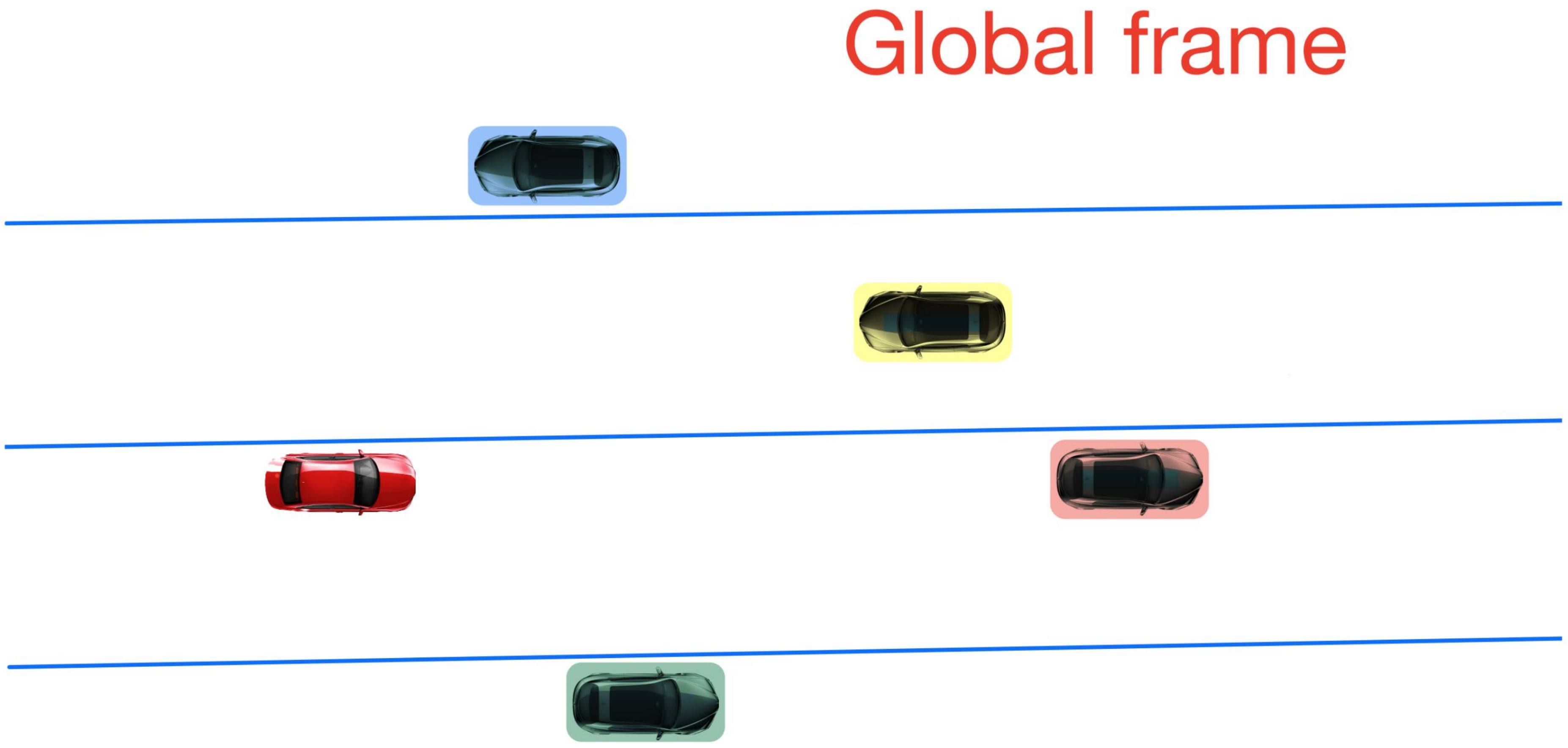


0.663 IOU

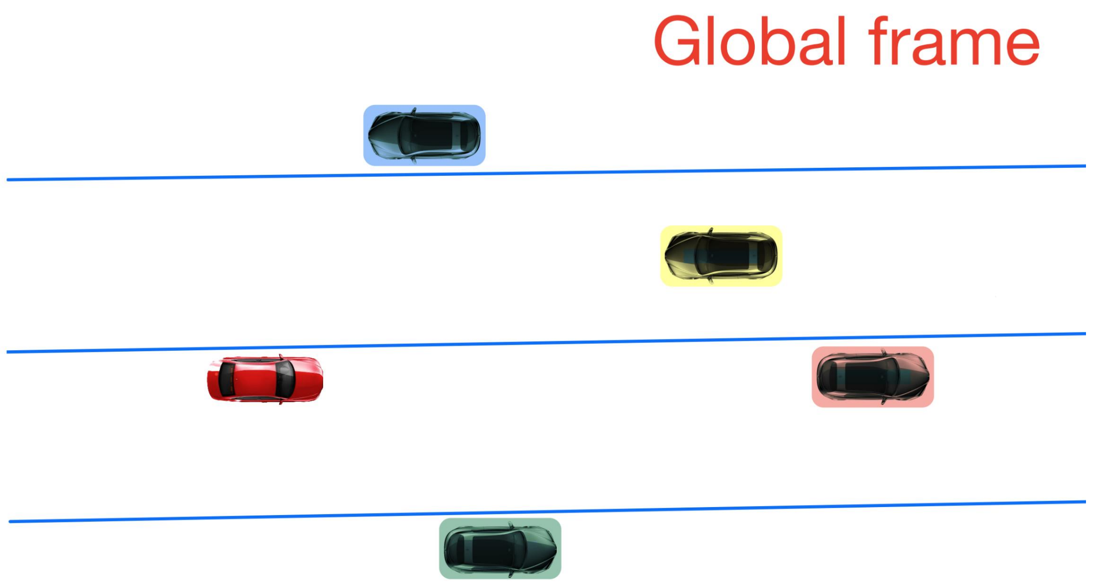
Tracking Multiple Objects

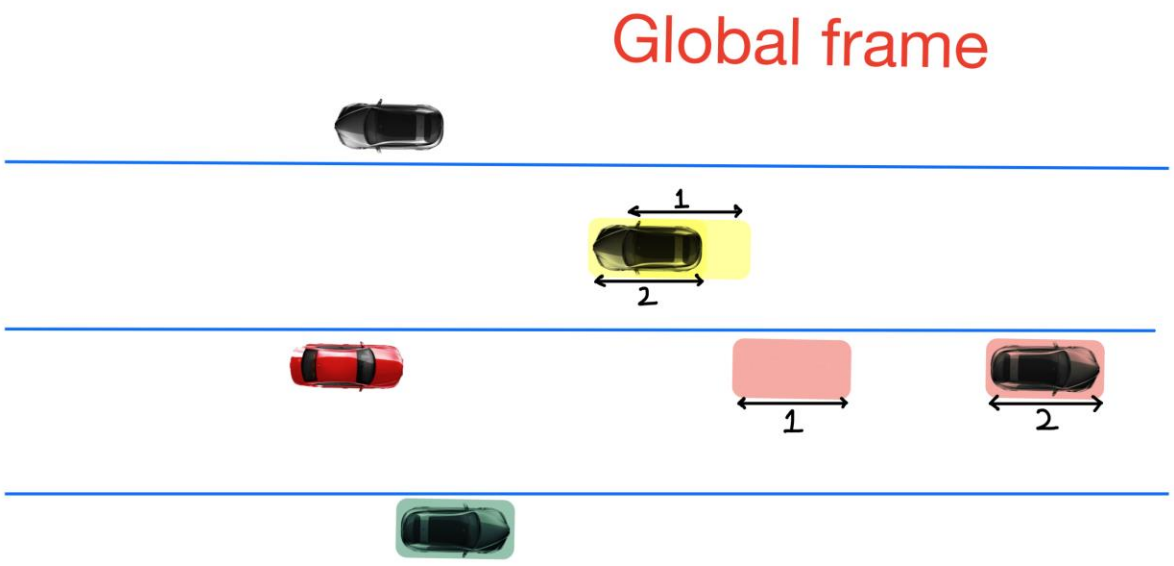


Prediction of Bounding Boxes



Basic IOU problems

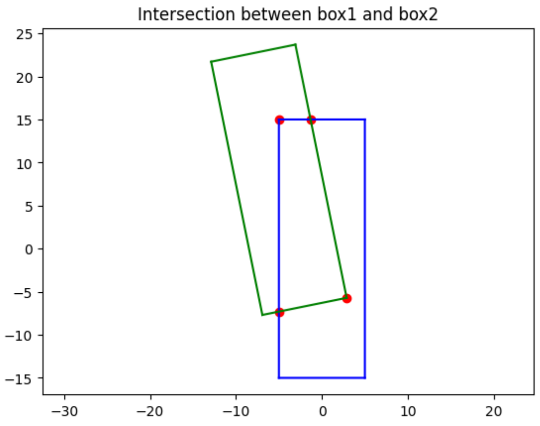




1. Sometimes PointRCNN detection fails to detect cars
2. Cars moving with high speeds may have 0 IOU, failing instance tracking

How can we improve instance tracking?

How can we improve instance tracking?



0.266 IOU

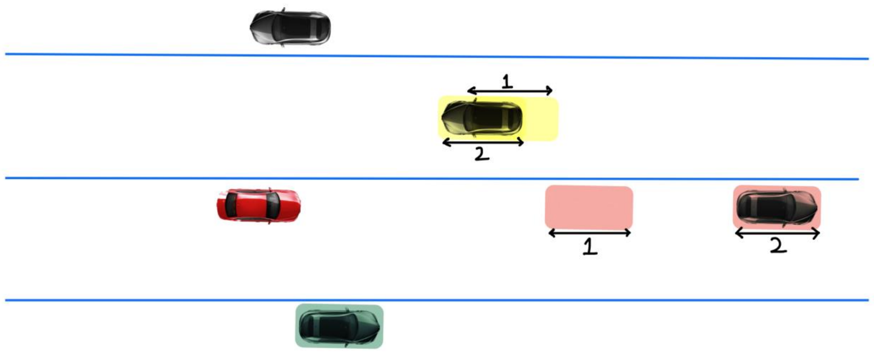
PyTorch3D IoU

Improving IoU calculation

Frame	Cars in Scene	Cars Detected	Buffer
1	1 2 3	1 2 3	1 2 3
2	1 2 3 4	1 3 4	1 3 4
3	2 3 4	2 3 4	2 3 4

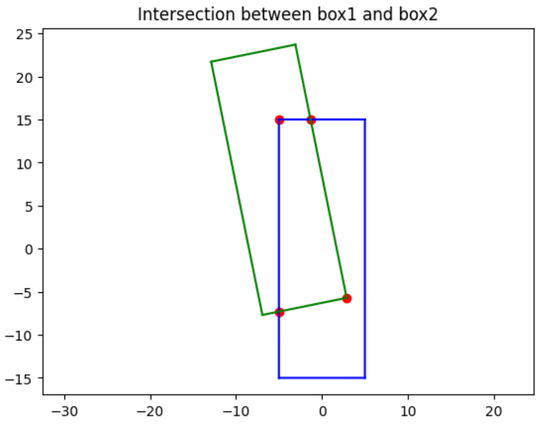
If (Buffer = Detections)

Global frame



More robust instance association using motion models

How can we improve instance tracking?



0.266 IOU

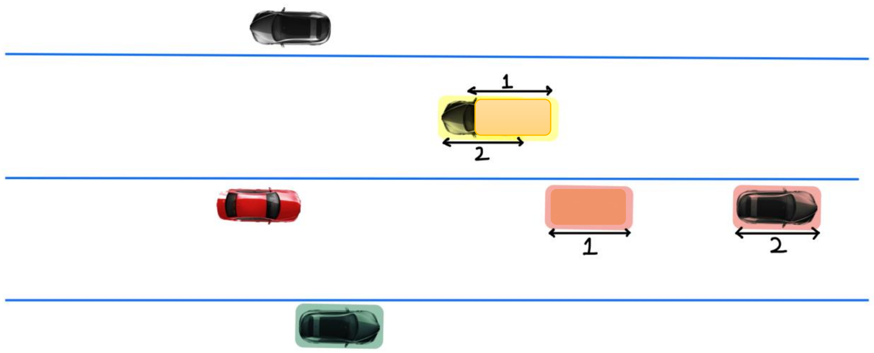
PyTorch3D IoU

Improving IoU calculation

Frame	Cars in Scene	Cars Detected	Buffer
1	1 2 3	1 2 3	1 2 3
2	1 2 3 4	1 3 4	1 3 4
3	2 3 4	2 3 4	2 3 4

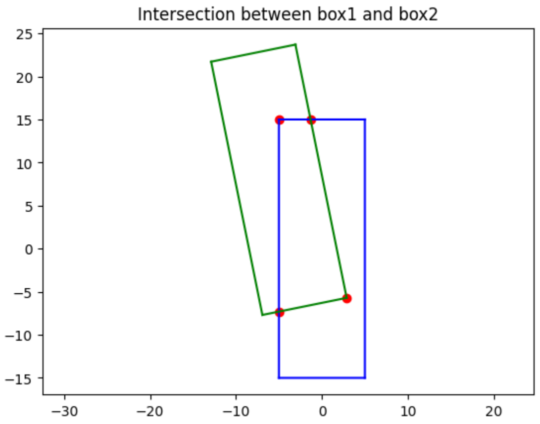
If (Buffer = Detections)

Global frame



More robust instance association track cars’ velocities

How can we improve instance tracking?



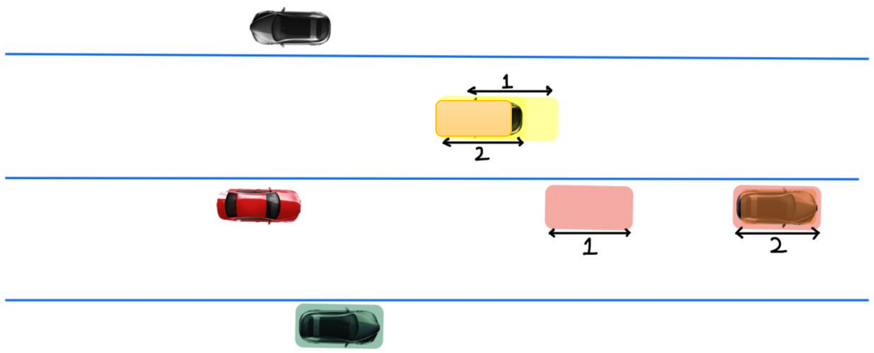
0.266 IOU

PyTorch3D IoU

Frame	Cars in Scene	Cars Detected	Buffer
1	1 2 3	1 2 3	1 2 3
2	1 2 3 4	1 3 4	1 3 4
3	2 3 4	2 3 4	2 3 4

If (Buffer = Detections)

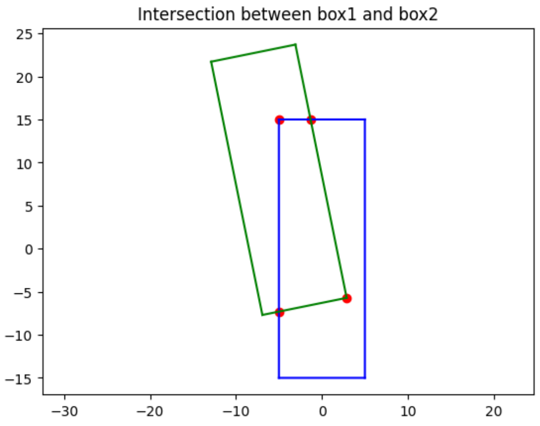
Global frame



Improving IoU calculation

More robust instance association track cars’ velocities

How can we improve instance tracking?



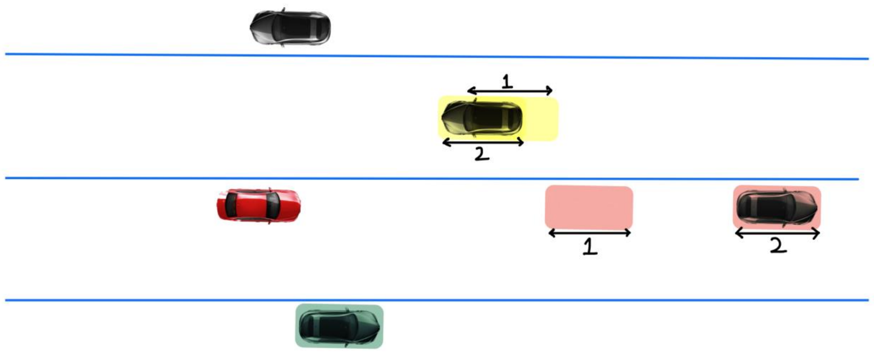
0.266 IOU

PyTorch3D IoU

Improving IoU calculation

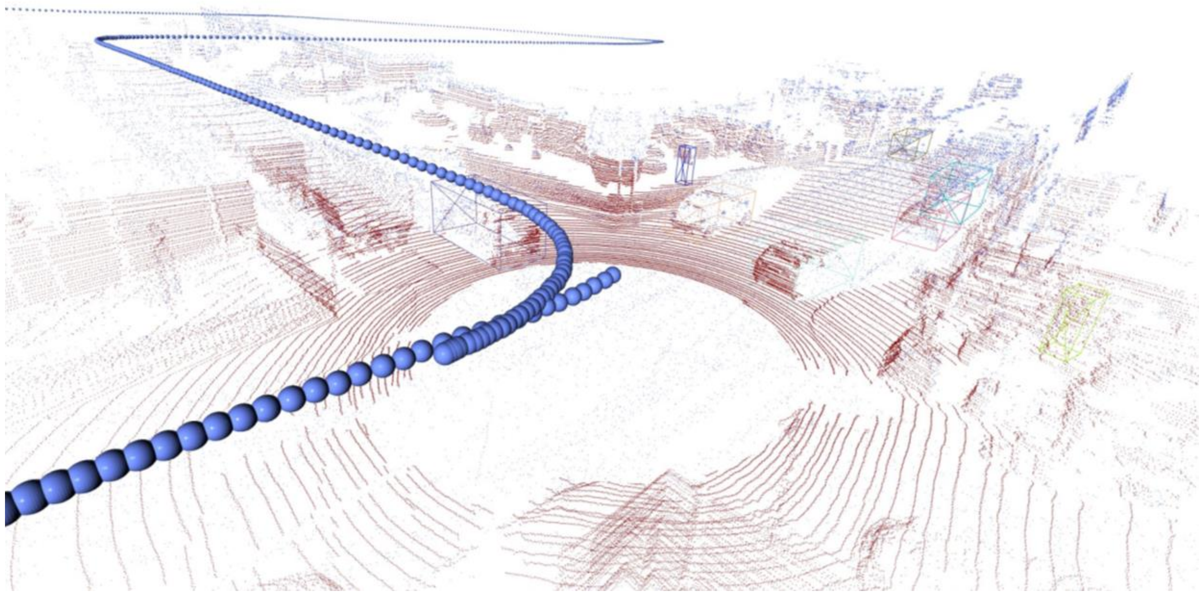
Frame	Cars in Scene	Cars Detected	Buffer	Motion Model
1	1 2 3	1 2 3	1 2 3	
2	1 2 3 4	1 3 4	1 3 4	2
3	2 3 4	2 3 4	2 3 4	1

Global frame



More robust instance association track cars' velocities

Tracking in Global and Sensor Frame

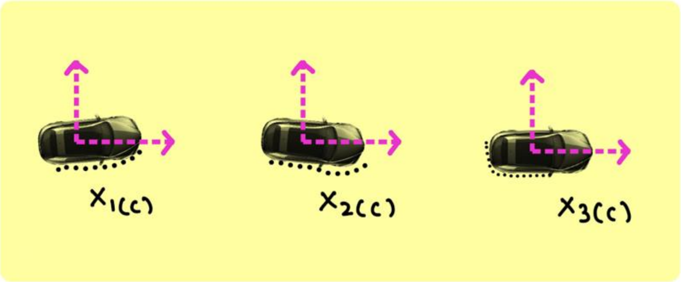


After proper instance association, we can use DeepSDF to jointly optimize shape and pose of the detected moving cars

Optimizing Latent Code and Pose Estimation

Optimize latent Code through back propagation

[Latent Code Vector].T



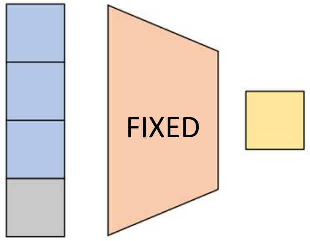
$$X_{1[C]} = T^1_{x_1} X_1$$

$$X_{2[C]} = T^1_2 T^2_{x_2} X_2$$

$$X_{3[C]} = T^1_2 T^2_3 T^3_{x_3} X_3$$

Code

(x,y,z)



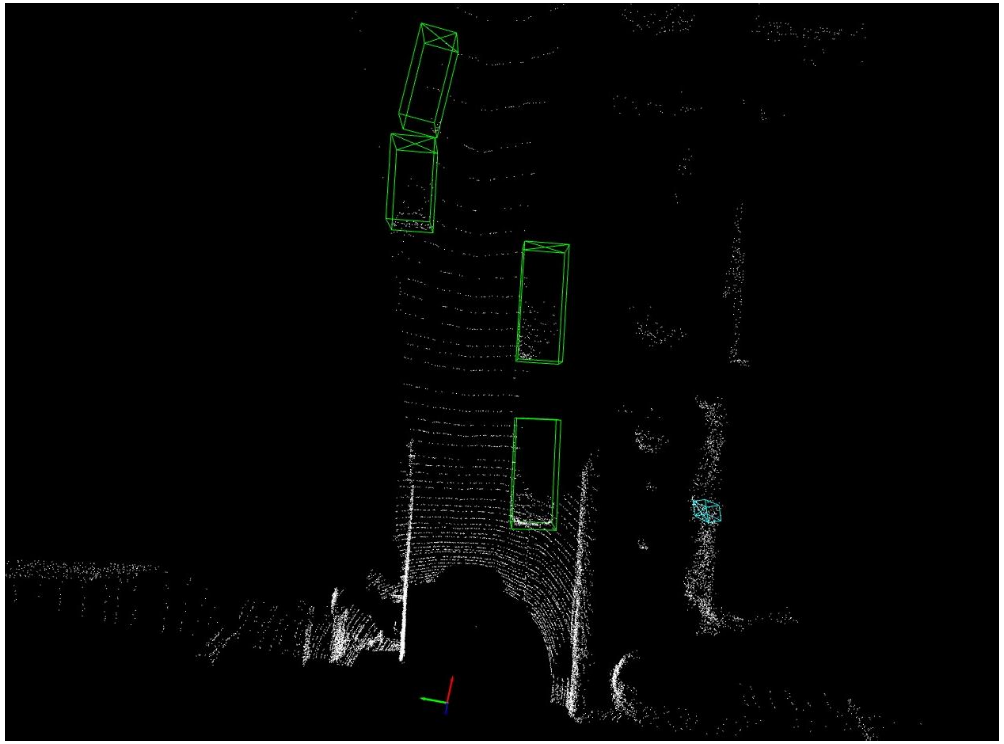
SDF

$$L = \lambda_s L_s + \lambda_R L_R$$

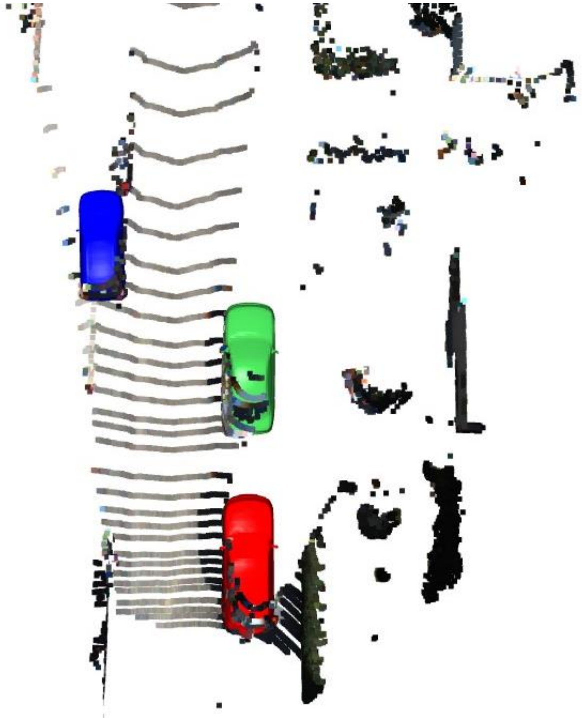
1. Surface Consistency
2. Latent Code Regularization

Optimize Pose transformations through back propagation

Optimizing Latent Code and Pose Estimation



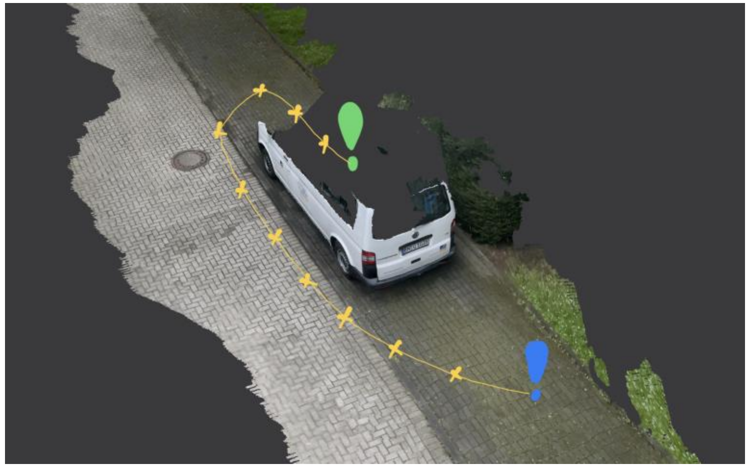
Lidar + Bounding Box



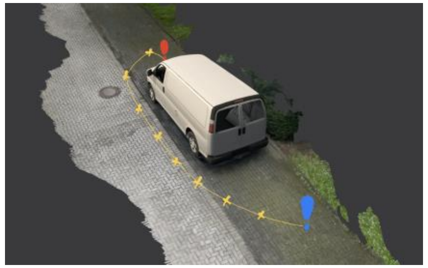
Stereo Image + Shape Completion

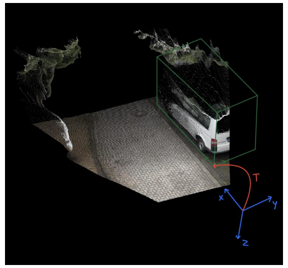
Note: To reconstruct this scene, it takes 0.2 seconds (5 fps)

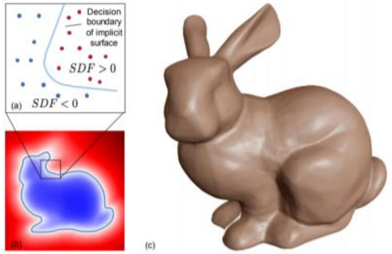
Overview of the project

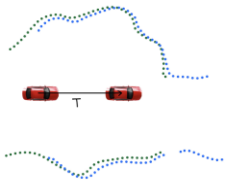


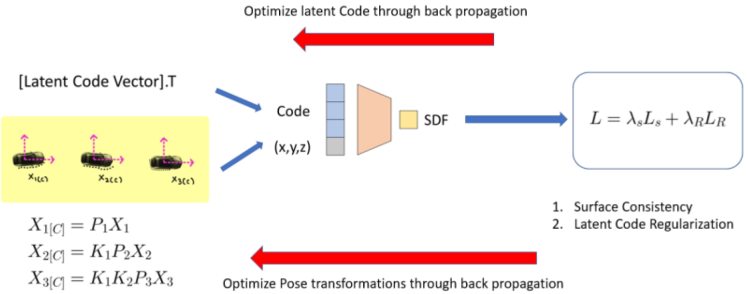
Challenges



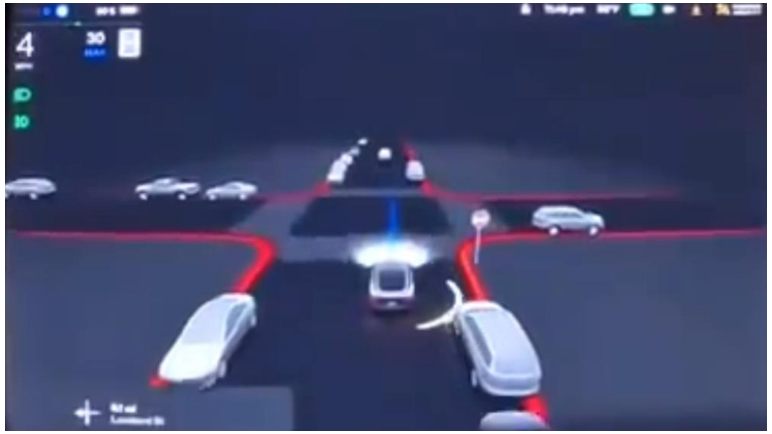






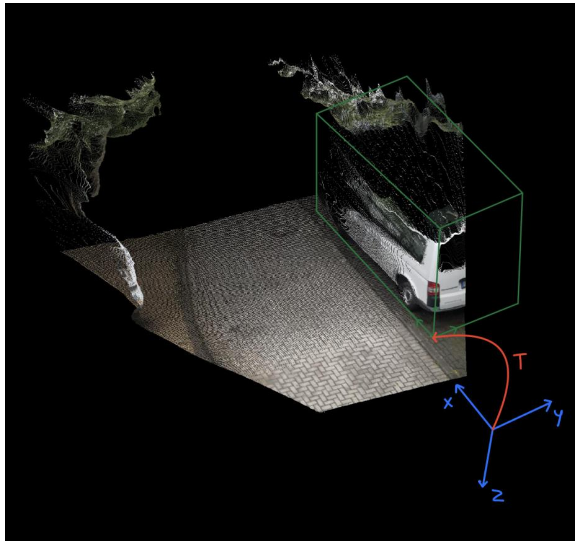


Method

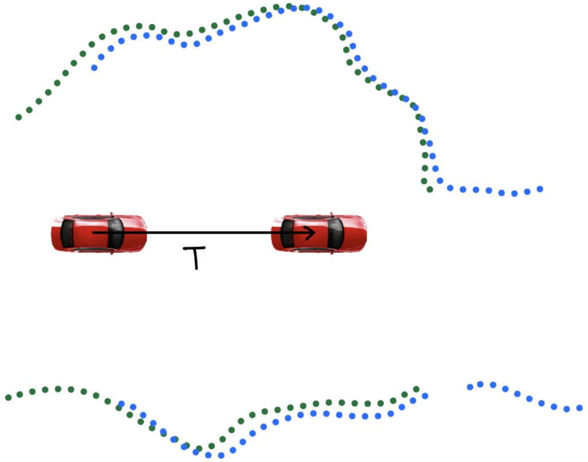


Final Output

Completed Tasks



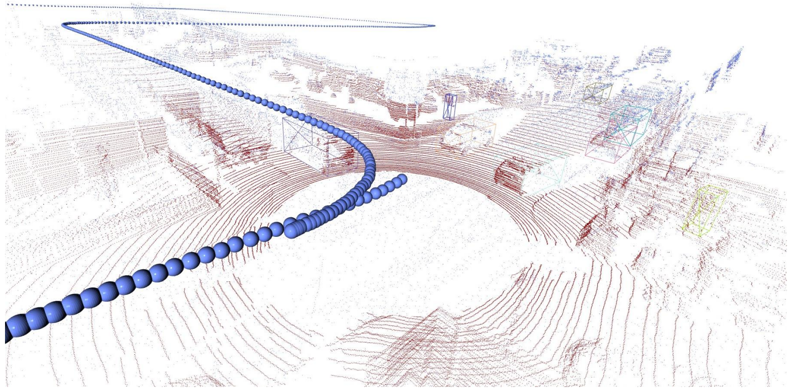
Bounding Box Prediction



Lidar Odometry

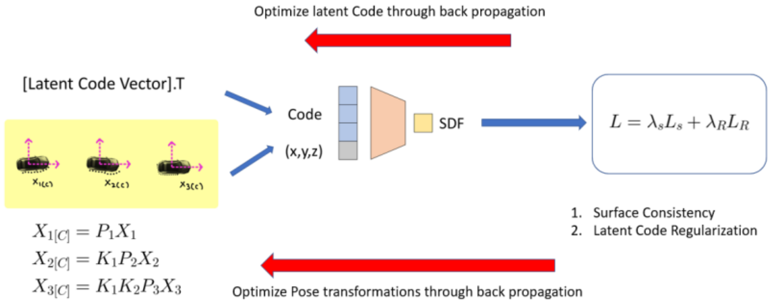
Instance Association and
Multi Object Tracking

Next Steps

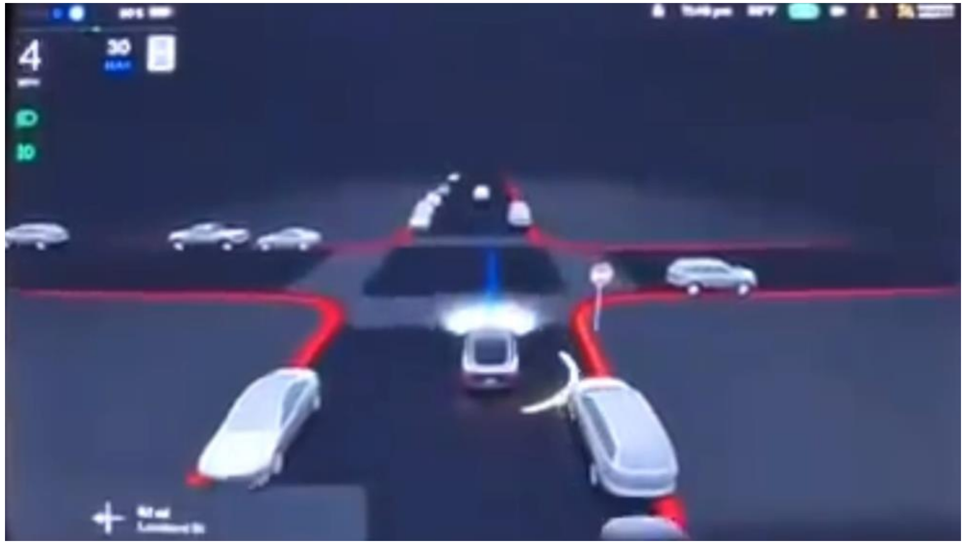


Pose and Point Cloud of the cars

Optimized Pose and Shape Code of the cars



Marching Cubes
+
Visualization



Thank You

Other Works using Shape Completion

Panoptic Mapping with Fruit Completion and Pose Estimation for Horticultural Robots

Yue Pan, Federico Magistri, Thomas Läbe, Elias Marks
Claus Smitt, Chris McCool, Jens Behley, Cyrill Stachniss

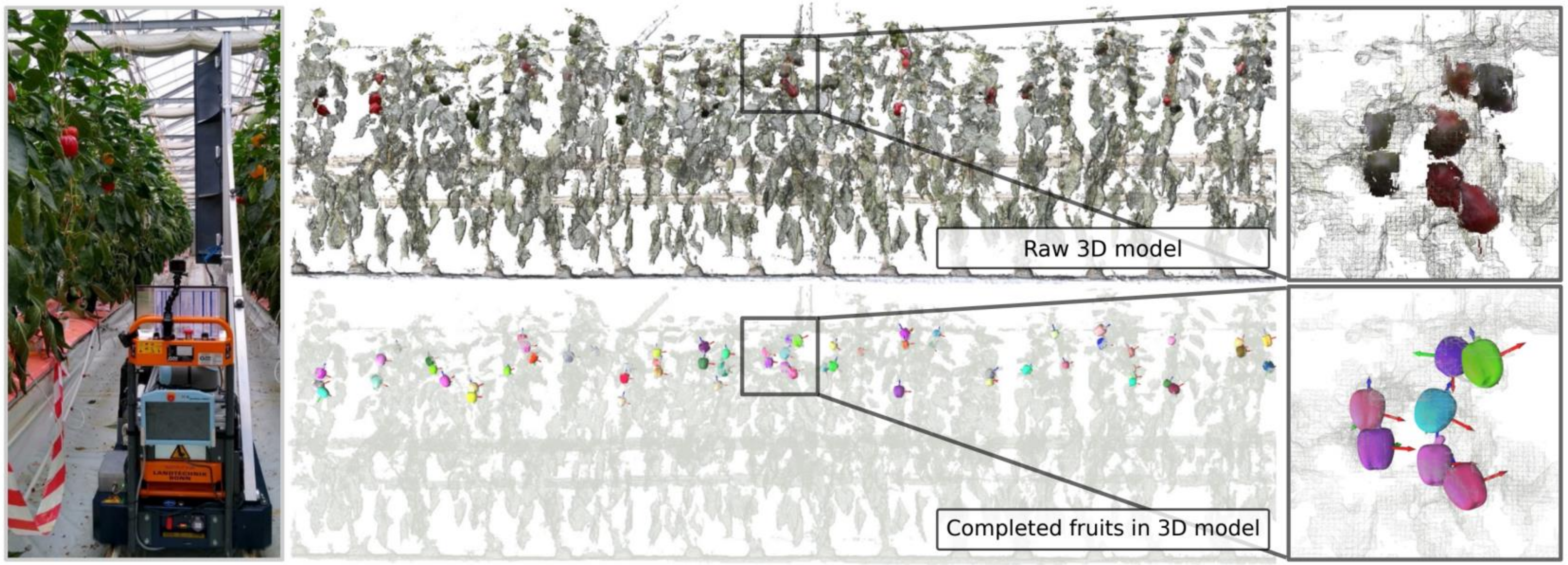
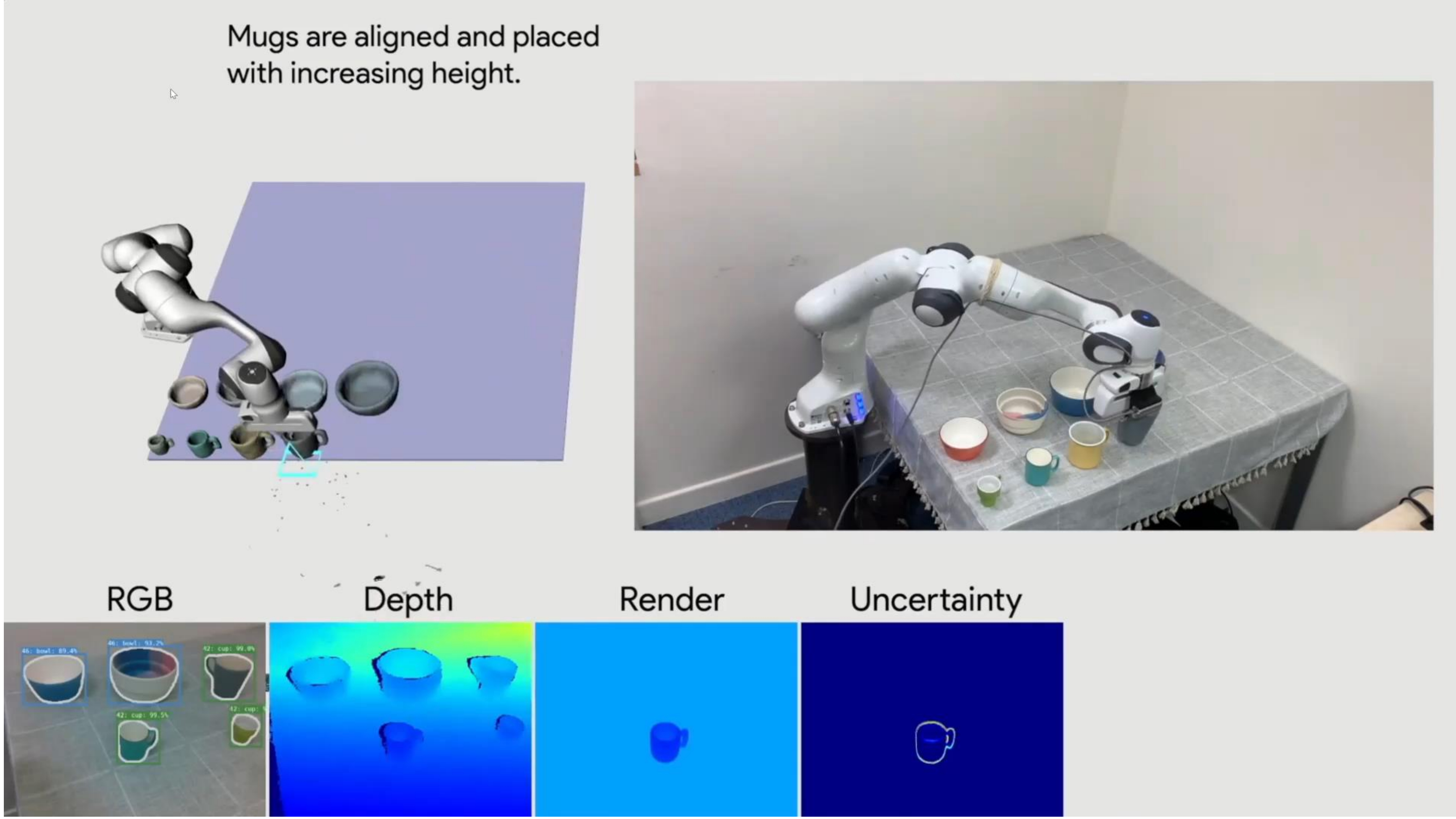


Fig. 1: Our method is able to build a multi-resolution panoptic map (top) of a challenging commercial glasshouse environment online using a mobile horticultural robot equipped with RGB-D cameras (left). Furthermore, our method manages to jointly estimate the complete shape and pose of each fruit in the map (bottom).

Node SLAM



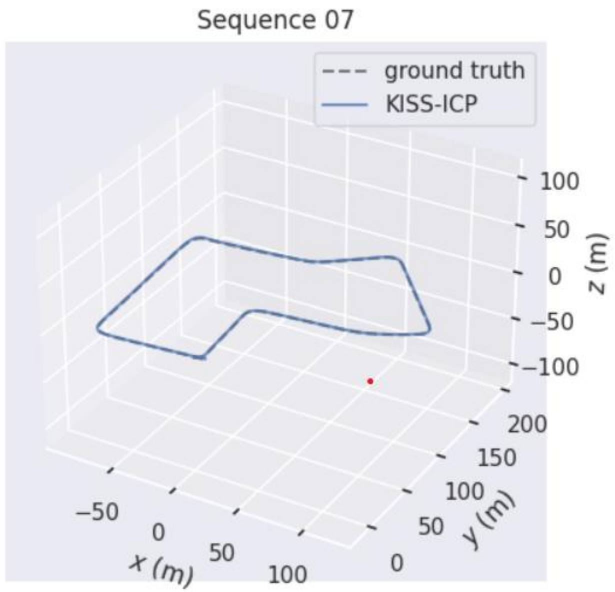
Source: Node SLAM

Metrics

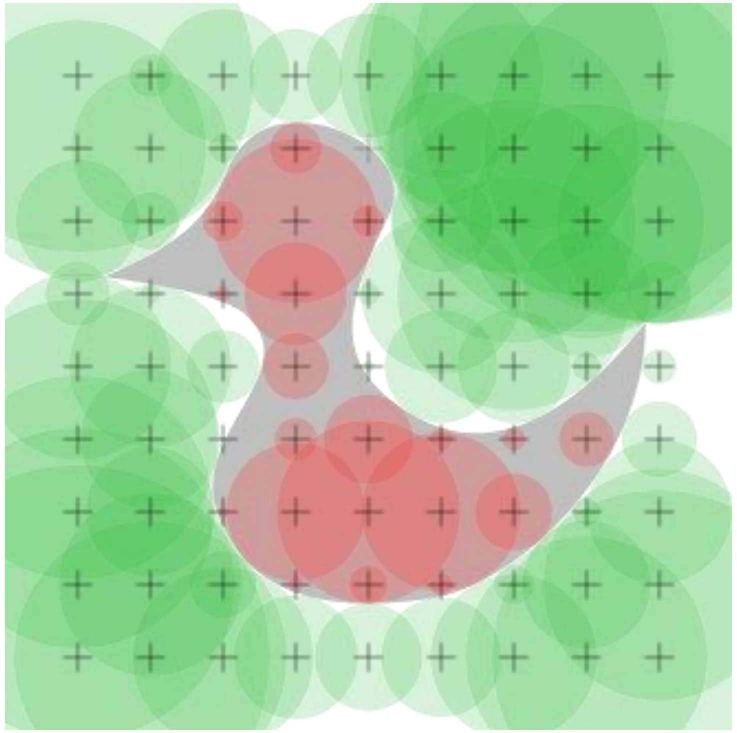
- PointRCNN - 8 fps
- Kiss ICP – 60 fps
- Shape Completion – 5 fps (avg)
- Kiss ICP + Bounding Box (Stored) – 30 fps
- DSP SLAM – 5 fps

```
Now evaluating sequence 07
  0%|
```

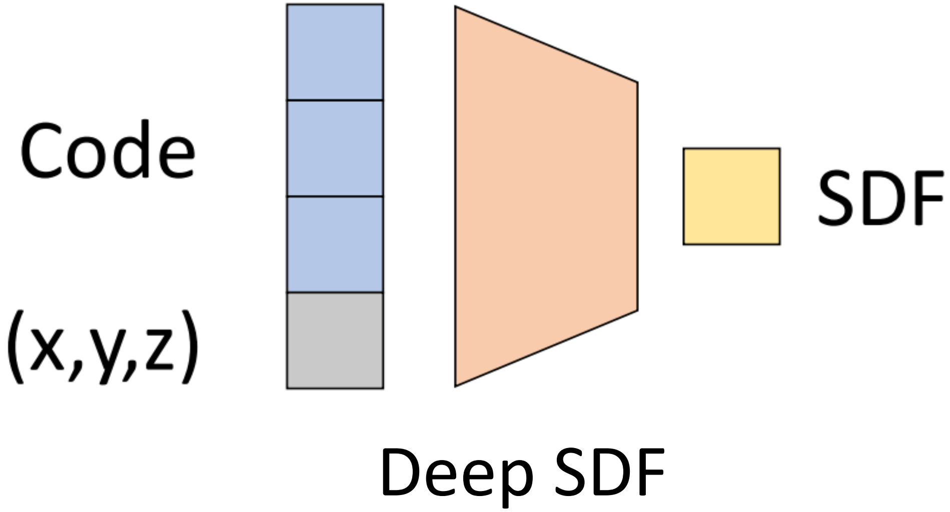
Metric	Value	Units
Average Translation Error	0.328	%
Average Rotational Error	0.164	deg/m
Absoulte Trajectory Error (ATE)	0.776	m
Absoulte Rotational Error (ARE)	0.007	rad
Average Frequency	69	Hz
Average Runtime	14	ms



Deep Shape Priors using Signed Distance Fields



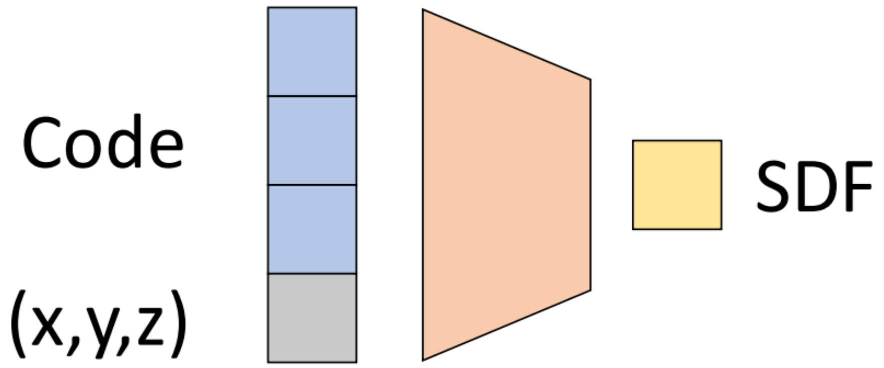
SDF $f_\theta(x, y, z) \approx SDF(x, y, z)$



DeepSDF can implicitly model a class of objects

Source (SDF): https://creativecommons.org/

Deep SDF

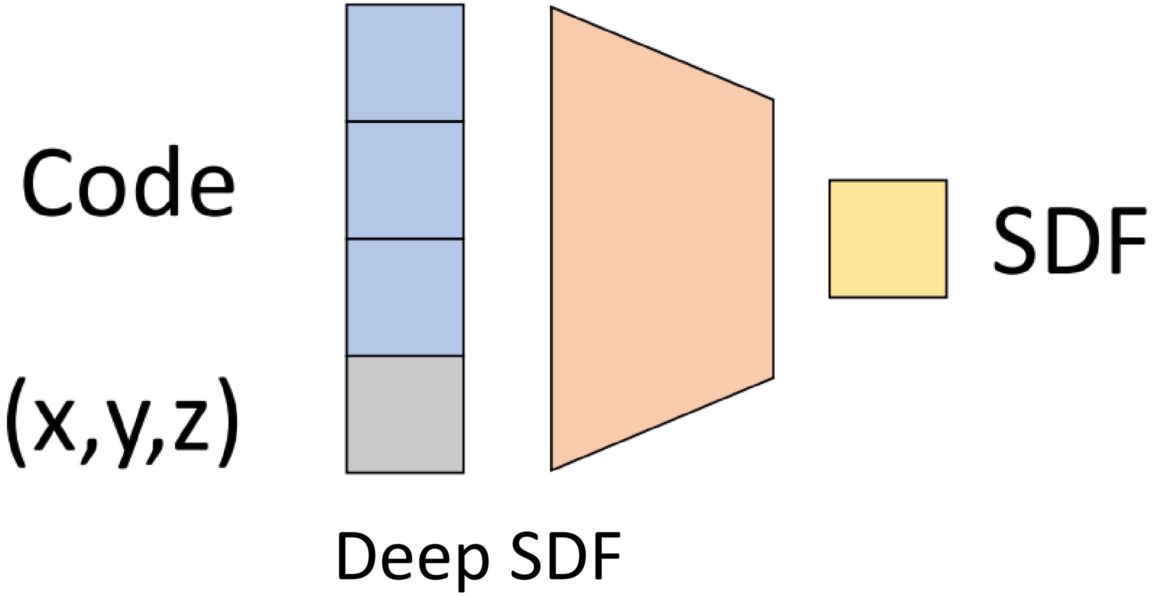


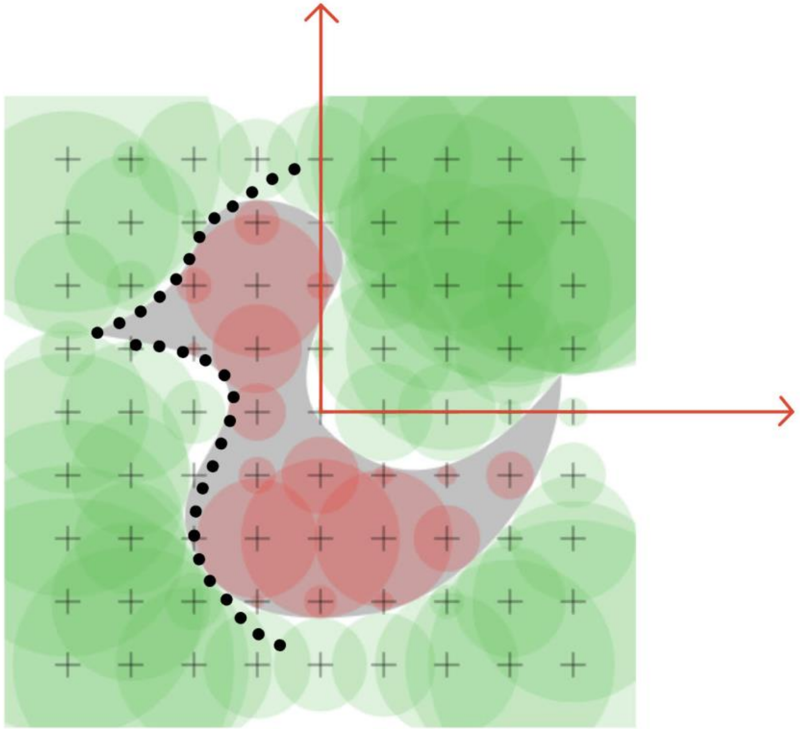
Deep SDF

A latent code collapses a general representation into in a single shape

Source: Deep SDF paper

0 SDF

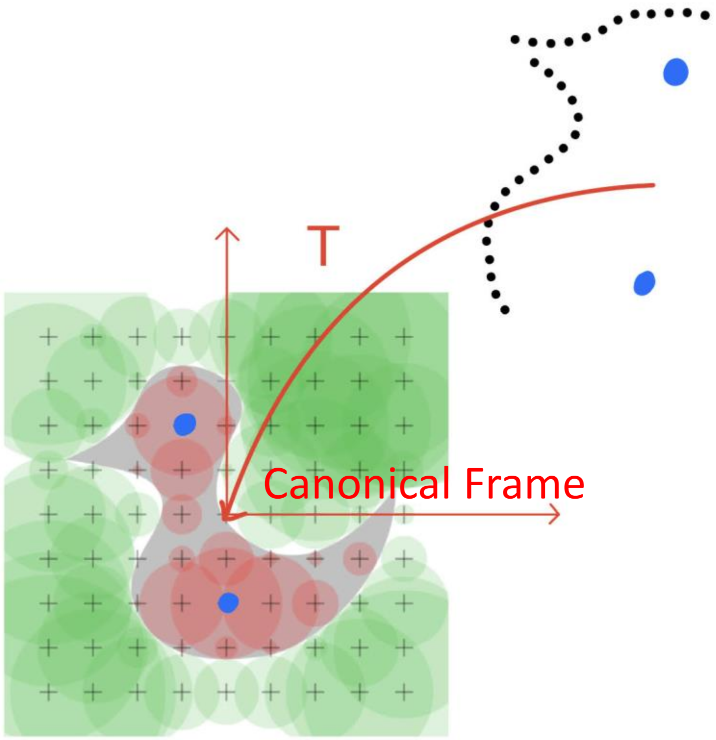


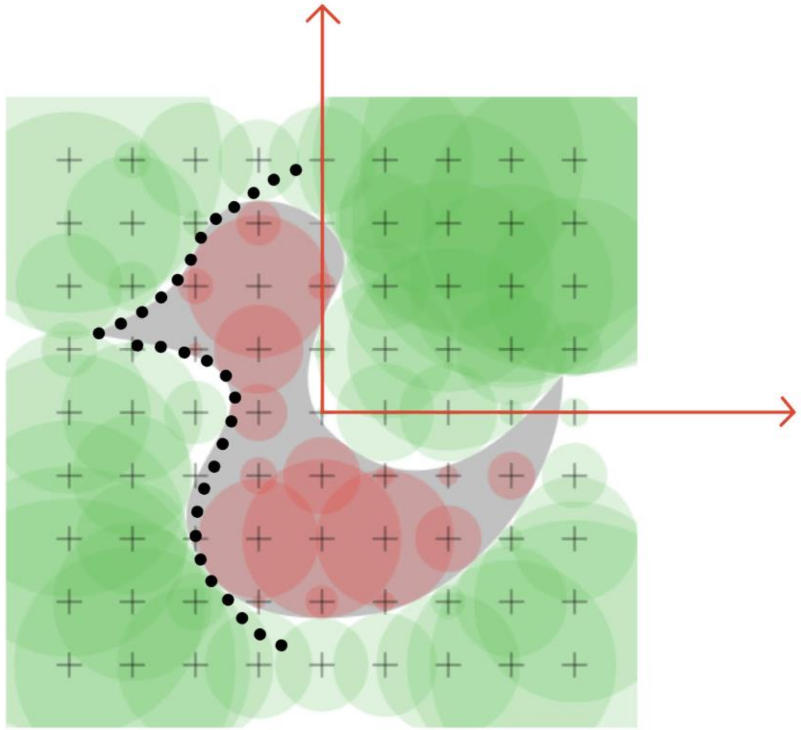


SDF value for a point on surface modelled by SDF is ZERO

Source (SDF): https://creativecommons.org/

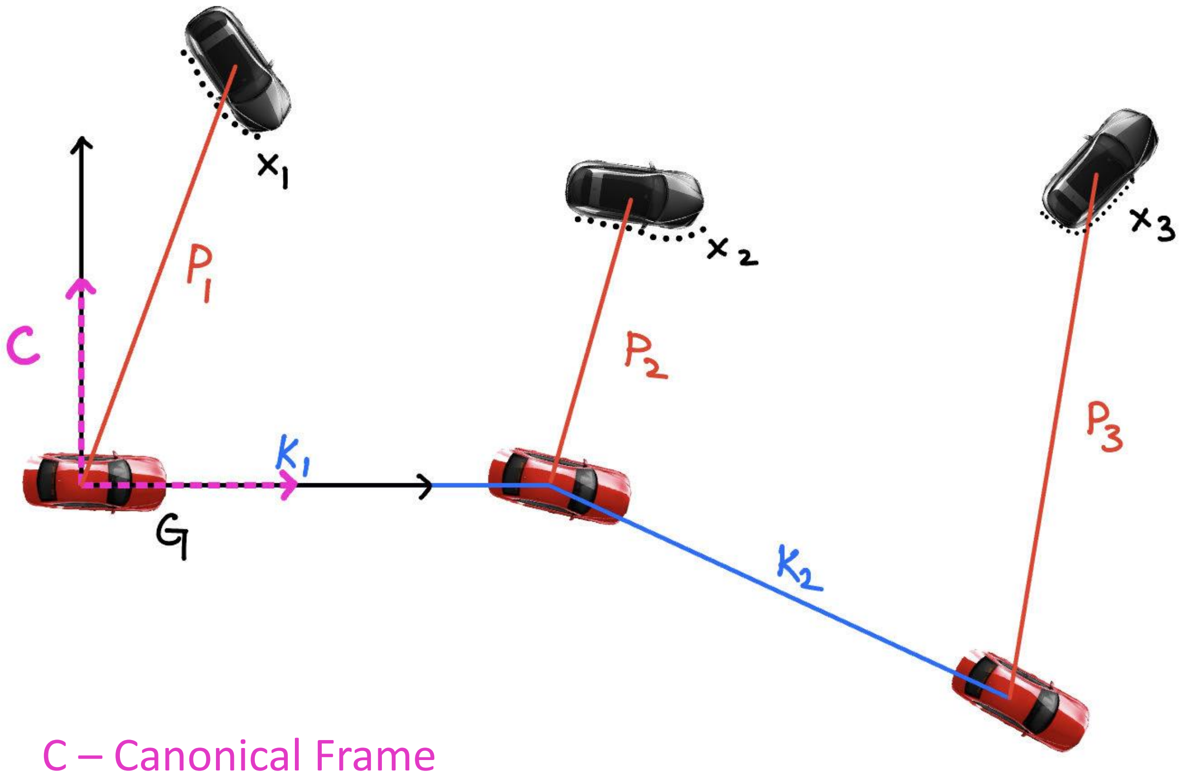
Canonical Frame





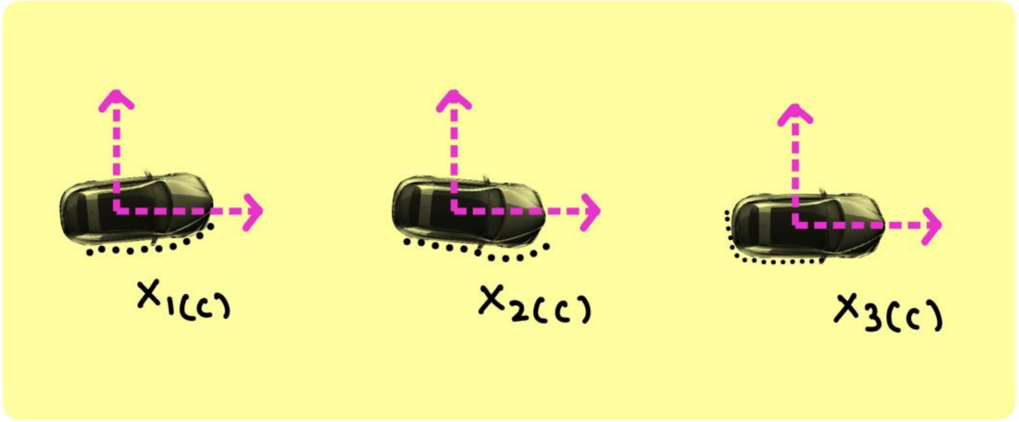
SDF presumes that the input is in Canonical Frame

Workflow: Transformation



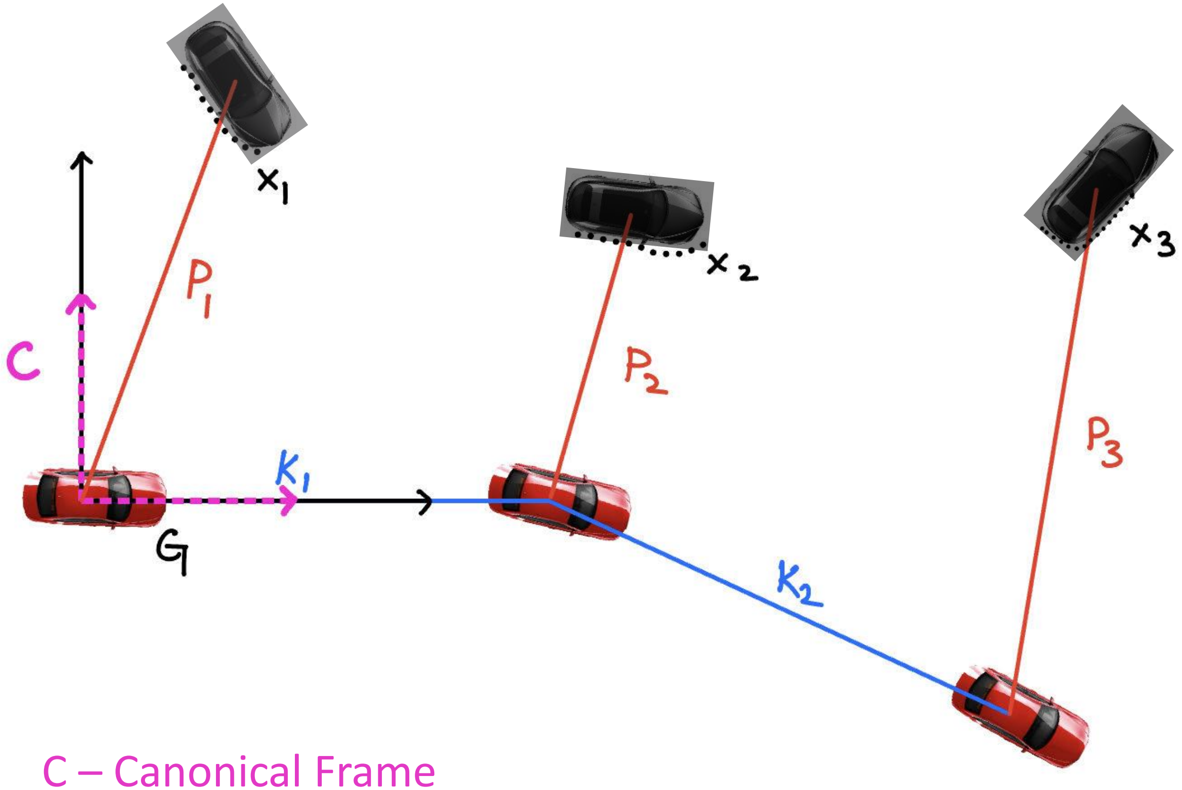
C – Canonical Frame
G – Global Frame
P – Transformation from Point RCNN
K – Transformation from KISS ICP

$$X_{1[C]} = P_1 X_1$$
$$X_{2[C]} = K_1 P_2 X_2$$
$$X_{3[C]} = K_1 K_2 P_3 X_3$$



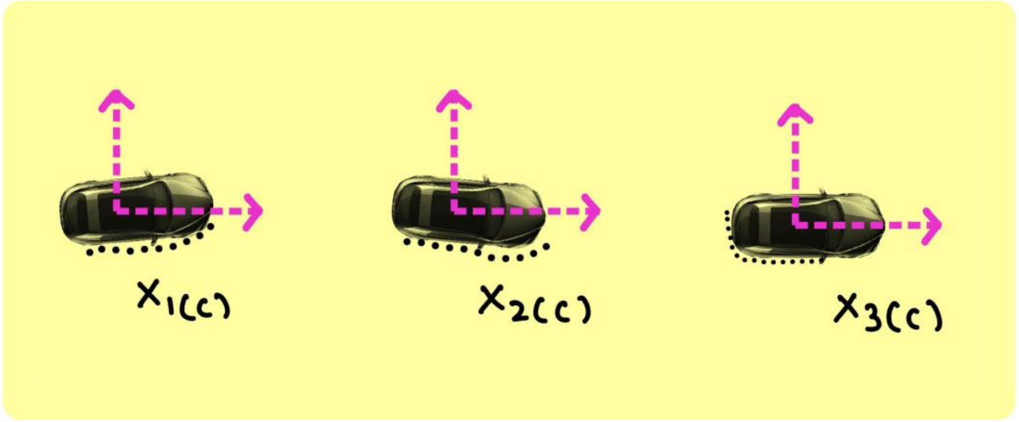
Canonical Space

Workflow: Transformation



$$X_{1[C]} = P_1 X_1$$
$$X_{2[C]} = K_1 P_2 X_2$$
$$X_{3[C]} = K_1 K_2 P_3 X_3$$

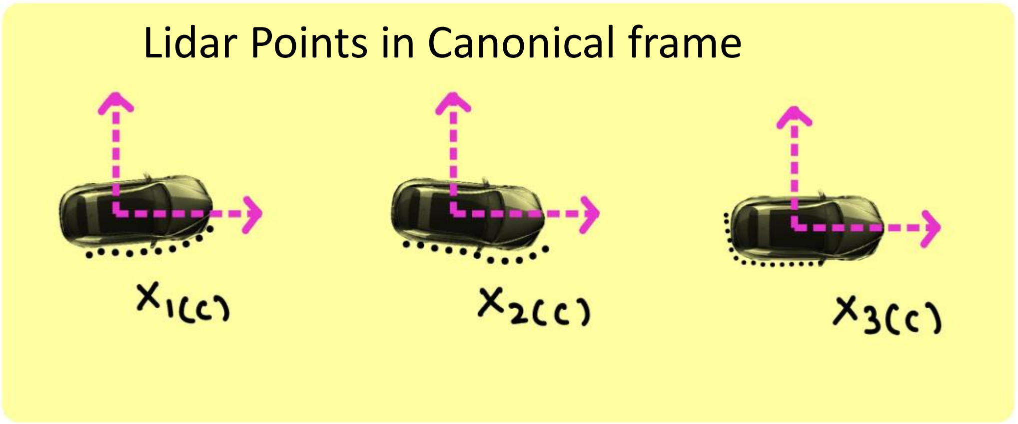
C – Canonical Frame
G – Global Frame
P – Transformation from Point RCNN
K – Transformation from KISS ICP

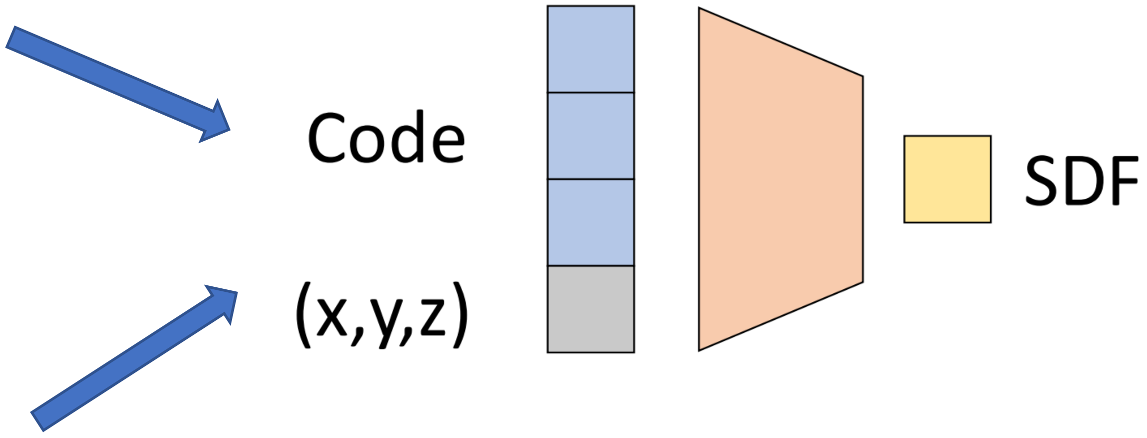


Canonical Space

Optimizing Latent Code and Pose Estimation

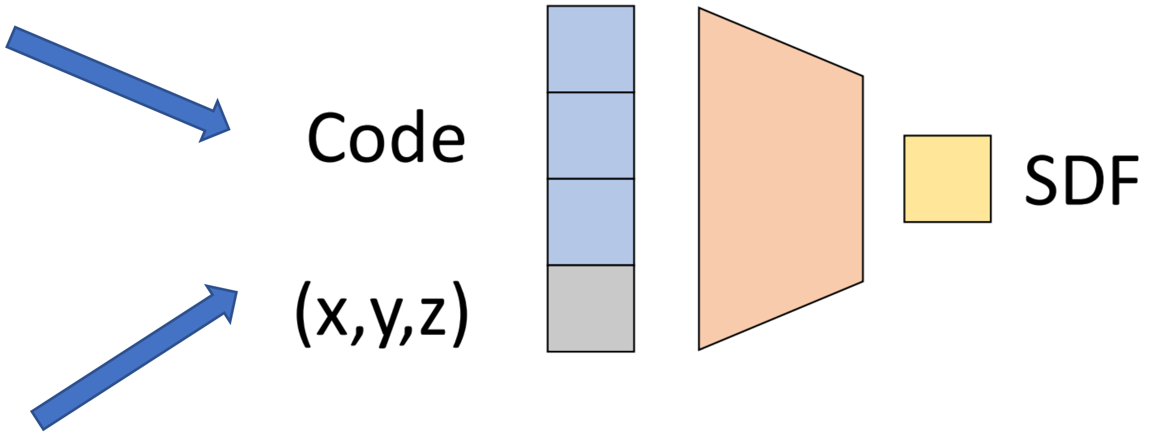
[Latent Code Vector].T



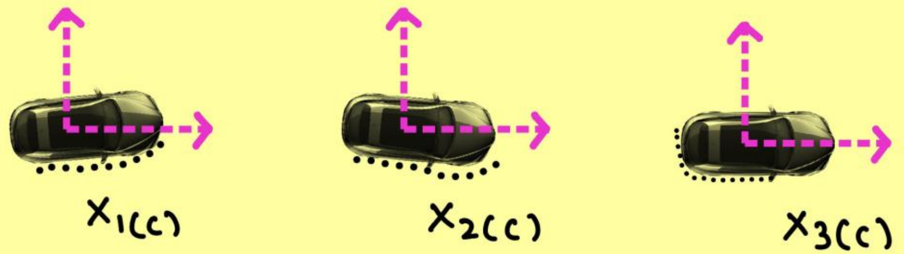


Optimizing Latent Code and Pose Estimation

[Latent Code Vector].T



Lidar Points in Canonical frame

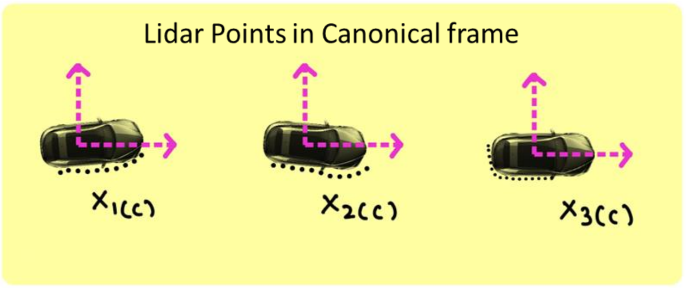


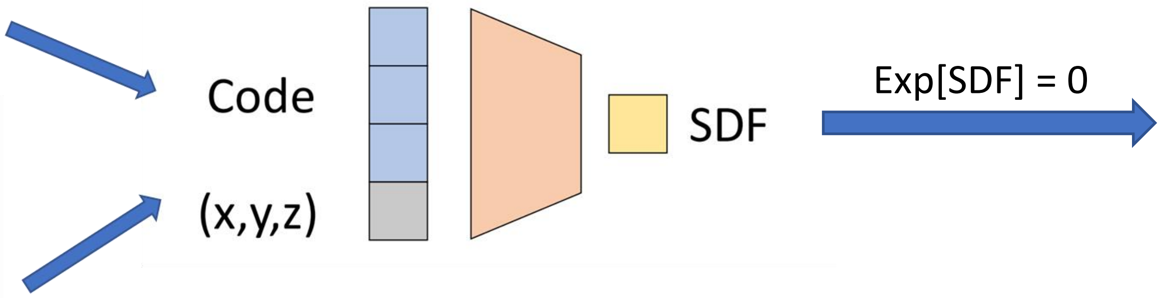
$$X_{1[C]} = P_1 X_1$$
$$X_{2[C]} = K_1 P_2 X_2$$
$$X_{3[C]} = K_1 K_2 P_3 X_3$$

Remember, points in canonical frame are a function of transformation predicted by PointRCNN

Optimizing Latent Code and Pose Estimation

[Latent Code Vector].T





$$L = \lambda_s L_s + \lambda_R L_R$$

$$X_{1[C]} = P_1 X_1$$

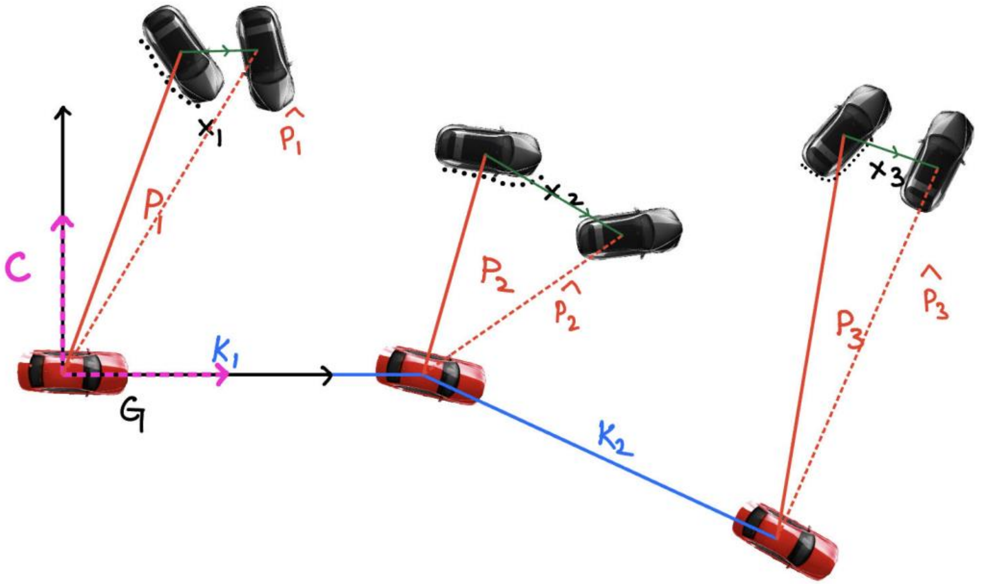
$$X_{2[C]} = K_1 P_2 X_2$$

$$X_{3[C]} = K_1 K_2 P_3 X_3$$

1. Surface Consistency
2. Latent Code Regularization

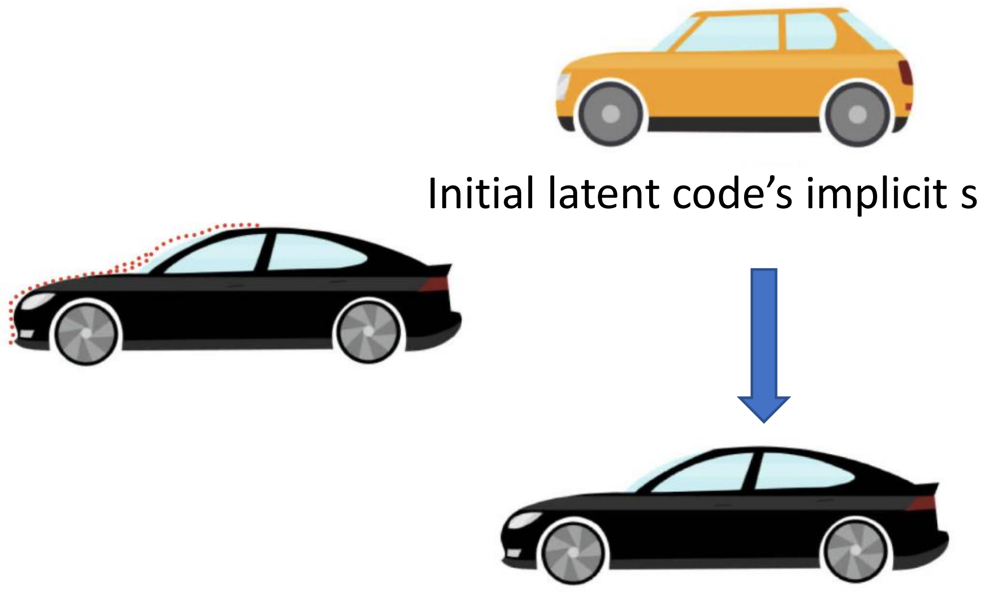
Optimizing Latent Code and Pose Estimation

Optimized Pose Estimation



Optimized poses of dynamic obstacles

Optimized latent Code



Initial latent code’s implicit surface

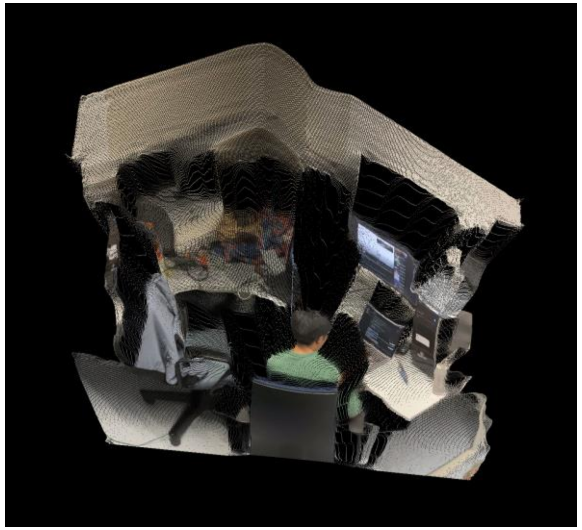
Optimized latent code’s implicit surface

DataSets and Evaluation

Dynamic Object Pose Estimation

3D Reconstruction

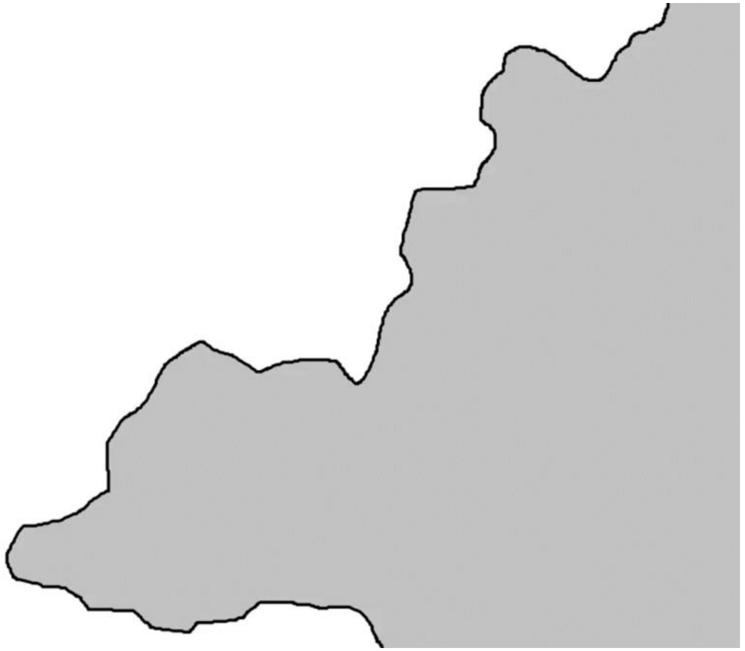
Multiple Images

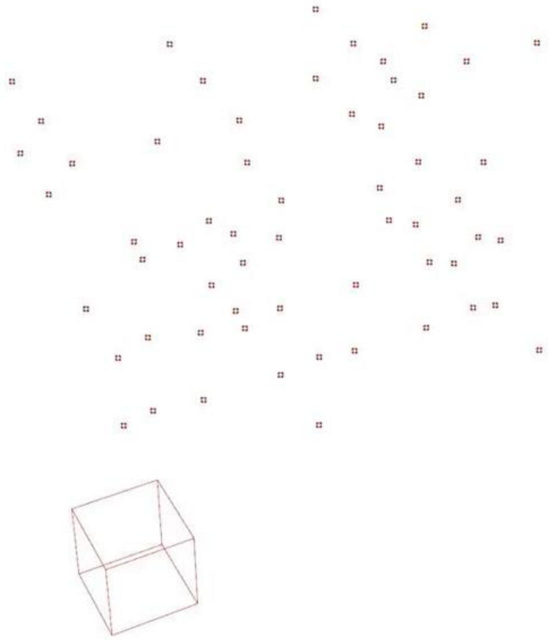


Partial Point Clouds

3D Scene

Read reasons for why shape completion is important from papers





Source (SDF): https://creativecommons.org/

M1

M1: Basic knowledge learning and paper reading (~4 weeks)

- Pytorch learning: finish [A1] [A2] [A3] Assignments of eecs498. (Only if you know nothing about Pytorch before)
- Read and understand the following papers:
  - Cluster-VO [code][paper][video]
  - DeepSDF [code][paper]
  - Nerf [paper]
  - DSP-SLAM [code][paper][video]
  - Weakly Supervised Learning of Rigid 3D Scene Flow [code][paper]
- Get familiar with the following datasets we will use:

1. KITTI
2. Nuscenes
3. Waymo

Source (SDF): https://creativecommons.org/

EECS 498.008 / 598.008

Deep Learning for Computer Vision

Winter 2022

Course Description

Computer Vision has become ubiquitous in our society, with applications in search, image understanding, apps, mapping, medicine, drones, and self-driving cars. Core to many of these applications are visual recognition tasks such as image classification and object detection. Recent developments in neural network approaches have greatly advanced the performance of these state-of-the-art visual recognition systems. This course is a deep dive into details of neural-network based deep learning methods for computer vision. During this course, students will learn to implement, train and debug their own neural networks and gain a detailed understanding of cutting-edge research in computer vision. We will cover learning algorithms, neural network architectures, and practical engineering tricks for training and fine-tuning networks for visual recognition tasks.

Source (SDF): https://creativecommons.org/

Future Mile Stones

M2: Code Reading and demo test (~2 weeks)

- Read the code of DSP-SLAM and test it in different datasets, especially in dynamic scenes, and observe failure situations.
- Read the instance association part of Cluster-VO.

M3: Build the main program framework (~4 weeks)

- Finish the data preprocessing part, including detection network and lidar odometry.
- Finish 2 in Methodology based on DSP-SLAM.
- Finish 3 in Methodology with the simplest method (bounding box overlap).

M4: Make some contribution (~6 weeks)

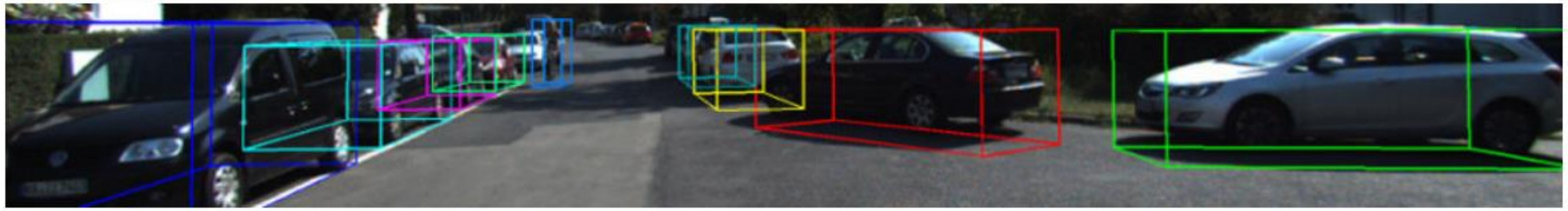
- Finish 4 in Methodology by designing a proper loss function.
- Explore more robust methods to achieve instance association.
- Explore the complete differentiable process from instance association to motion estimation.

M5: Final Result: Evaluation (~4 weeks)

- Use scene flow dataset and metric to evaluate motion estimation.
- Design another metric to evaluate instance association with panoptic segmentation dataset.

The KITTI Vision Benchmark Suite

3D Object Detection Evaluation 2017



The 3D object detection benchmark consists of 7481 training images and 7518 test images as well as the corresponding point clouds, comprising a total of 80.256 labeled objects. For evaluation, we compute precision-recall curves. To rank the methods we compute average precision. We require that all methods use the same parameter set for all test pairs. Our development kit provides details about the data format as well as MATLAB / C++ utility functions for reading and writing the label files.

- Download left color images of object data set (12 GB)
- Download right color images, if you want to use stereo information (12 GB)
- Download the 3 temporally preceding frames (left color) (36 GB)
- Download the 3 temporally preceding frames (right color) (36 GB)
- Download Velodyne point clouds, if you want to use laser information (29 GB)
- Download camera calibration matrices of object data set (16 MB)
- Download training labels of object data set (5 MB)
- Download object development kit (1 MB) (including 3D object detection and bird's eye view evaluation code)
- Download pre-trained LSVM baseline models (5 MB) used in Joint 3D Estimation of Objects and Scene Layout (NIPS 2011). These models are referred to as LSVM-MDPM-sv (supervised version) and LSVM-MDPM-us (unsupervised version) in the tables below.
- Download reference detections (L-SVM) for training and test set (800 MB)
- Qianli Liao (NYU) has put together code to convert from KITTI to PASCAL VOC file format (documentation included, requires Emacs).
- Karl Rosaen (U.Mich) has released code to convert between KITTI, KITTI tracking, Pascal VOC, Udacity, CrowdAI and AUTTI formats.
- Jonas Heylen (TRACE vzw) has released pixel accurate instance segmentations for all 7481 training images.
- We thank David Stutz and Bo Li for developing the 3D object detection benchmark.