Local_INN: Implicit Map Representation and Localization with Invertible Neural Network

Zirui Zang, Hongrui Zheng, Johannes Betz, Rahul Mangharam

All authors are with the University of Pennsylvania, Department of Electrical and Systems Engineering, 19104, Philadelphia, PA, USA. Emails: {zzang, hongruiz, joebetz, rahulm}@seas.upenn.edu

Penn Engineering

About our lab at UPenn

xlab for safe autonomous systems

Rahul Mangharam
PI

Hongrui Zhang (Billy)
Multi-agent Interaction, Game-theoretic Learning

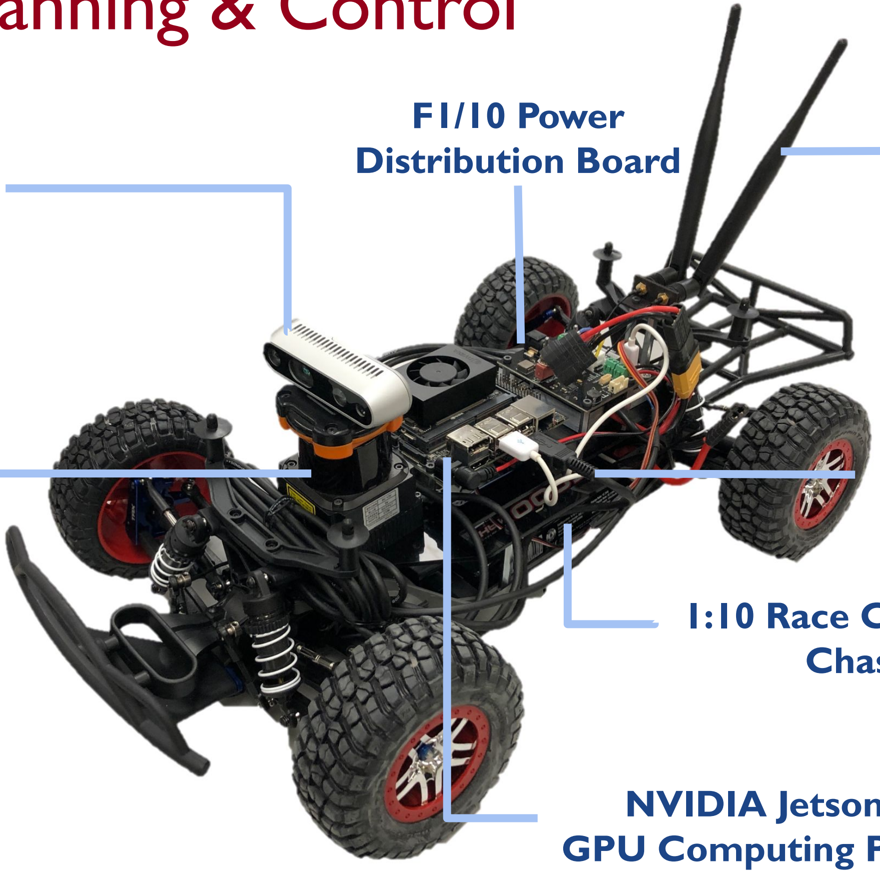
Nandan Tumu
Motion Prediction

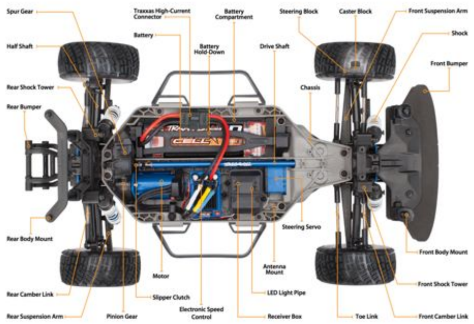
Zirui Zang
Localization, Perception

Ahmad Amine
Model Predictive Control

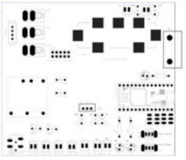
Penn Engineering

F1tenth: Low-Cost Open-Source Platform for Perception, Planning & Control





Chassis Design



F1tenth Powerboard

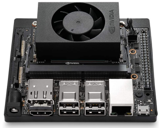
2D Lidar

Stereo Camera

Mono Camera

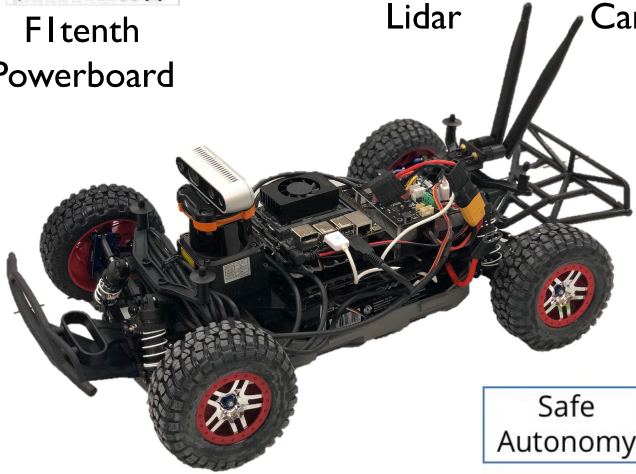
VESC Controller

Wifi



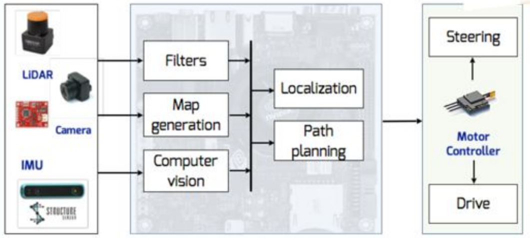
NVIDIA Jetson NX GPU Compute Platform

System Integration



Software Architecture

ROS2



Perception Planning Control

GPU Acceleration

NVIDIA CUDA

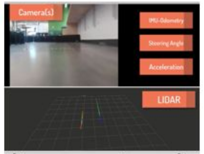
Research Enabled

Safe Autonomy	Secure Autonomy	Coordinated Autonomy	Efficient Autonomy

Lane Keeping Assist

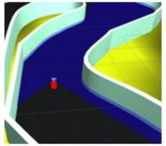
Model-Predictive Control

AV Data Collection



Simultaneous Localization And Mapping (SLAM)

F1/10 Simulator



DNN Racing

Penn Engineering F1 TENTH

F1tenth Course

A. Robotics and ROS basics
  a. Sensors and Mechanics
  b. Simulator
  c. Safe AV control basics

B. Navigation
  a. Reactive planning
  b. Mapping and Localization
  c. Pure Pursuit planning
  d. AV Ethics

C. AV Race Planning
  a. Raceline optimization
  b. RRT Planner
  c. Model Predictive Control

D. Vision and Learning
  a. Detection and Pose Estimation
  b. RL

E. Hands-on Project
  a. Drive till you MHz!

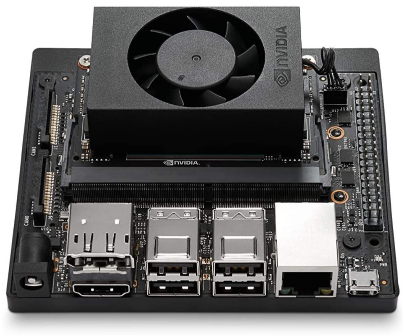
Lecture	Lecture Topic	Tutorial	Assignments Out
	Module A: Introduction to ROS, F110 & the Simulator		
1	Introduction to Autonomous Driving: Perception, Planning Control	T1: Intro to Docker & ROS 2	Lab 1: ROS 2 (individual)
2	Automatic Emergency Braking	T2: Intro to F1Tenth Sim	Lab 2: Automatic Emergency Braking
3	Rigid Body Transforms	T3: ROS2 and tf2	
	Module B: Reactive Methods & *RACE!*		
4	Wall Following	T4: Race car hands-on	Lab 3: Wall Following
5	Follow the Gap: Obstacle Avoidance		Lab 4: Follow the Gap
6	Vehicle State and Dynamics		
7	Scan matching		Lab 5. Scan Matching
8	Race Preparation		
9	**Race 1: Reactive**		
	Module C: Mapping & Localization		
10	Localization: Particle Filter	T5: Running Particle Filter	
11	SLAM: Cartographer	T6: Cartographer/HectorSLAM	Lab 6: Carto & Pure Pursuit
12	Pure Pursuit		
	Module D: Planning & Control		
13	Global Planning: Maps, A*, Djikstra		
14	Local Planning: RRT, Spline Based Planner		Lab 7: Motion Planning
	SPRING BREAK		
15	Path Tracking: MPC		
	Module E: Vision		
16	Classical Perception and Vision - Lane Detection & Optical Flow		Lab 8: Perception & Vision
17	ML Perception and Vision - Object Detection and Tracking		Lab 9: Ethics for AVs
18	Race Prep		
19	**Race 2: With Map**		
20	Final Project Overviews		Final Project Proposal
21	Ethics		
	Module F: Special Topics		
22	Raceline Optimization: Minimum Curvature, Minimum Time, Tunercar		Final Project
23	Theme 1: Autonomous Racing in the Real World		
24	Theme 2: AV Perception		
25	Theme 3: AV Middleware, OS		
	Flex		
	Module G: F1TENTH Grand Prix!!		
26	Project Demos		
27	Race Prep		
28	**Final Race Day**		
29	Final Project Documentation Submission Deadline		

10th F1Tenth Autonomous Racing Competition At ICRA 2022 in Philadelphia

F1tenth Race at ICRA2022

Current Racing Stack

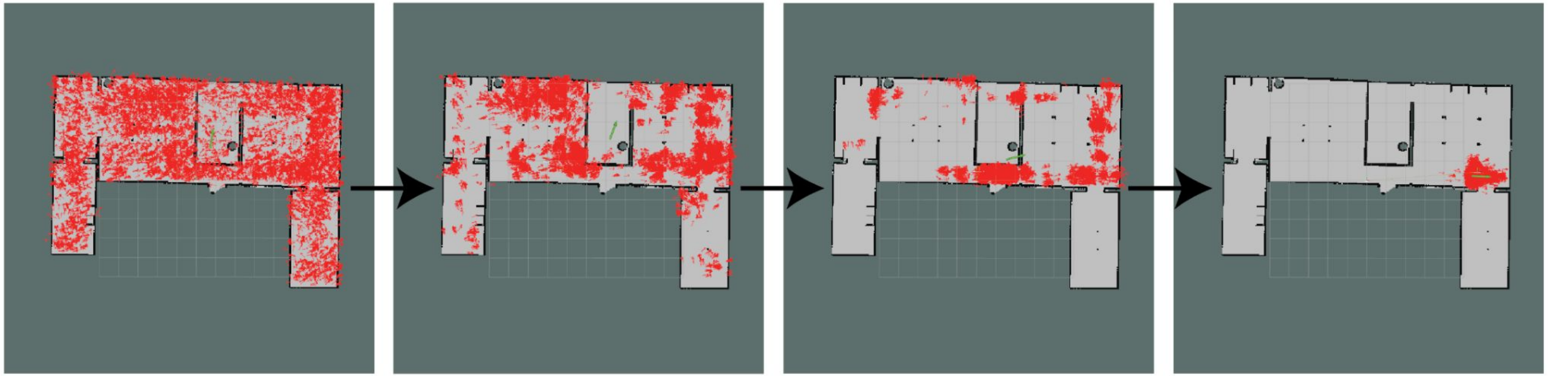
- CPU:
  - Planner: Raceline, Graph planner, etc.
  - Control: Pure Pursuit, MPC, etc.
- GPU
  - Localization: Particle Filter, VLAM, etc.
  - FULL



Indoor Localization with Map

- Commonly used methods --- Particle Filter
  - Need to simulate lidar for each particle
  - High computation and long latency
  - Only possible with 2D lidars

Particle filter localization process.

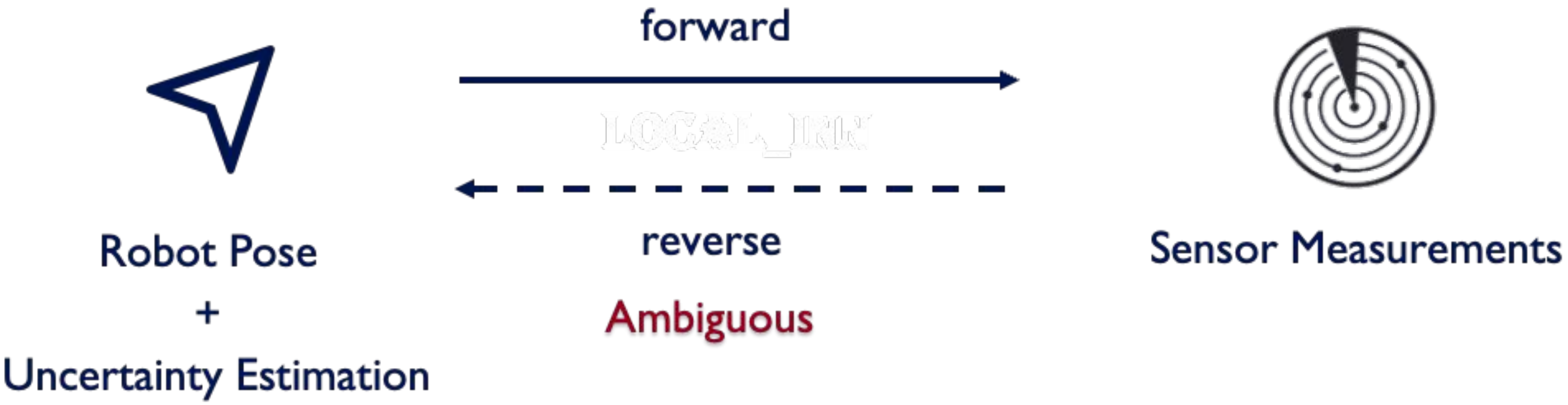


https://classes.cs.uchicago.edu/archive/2022/spring/20600-1/particle_filter_project.html

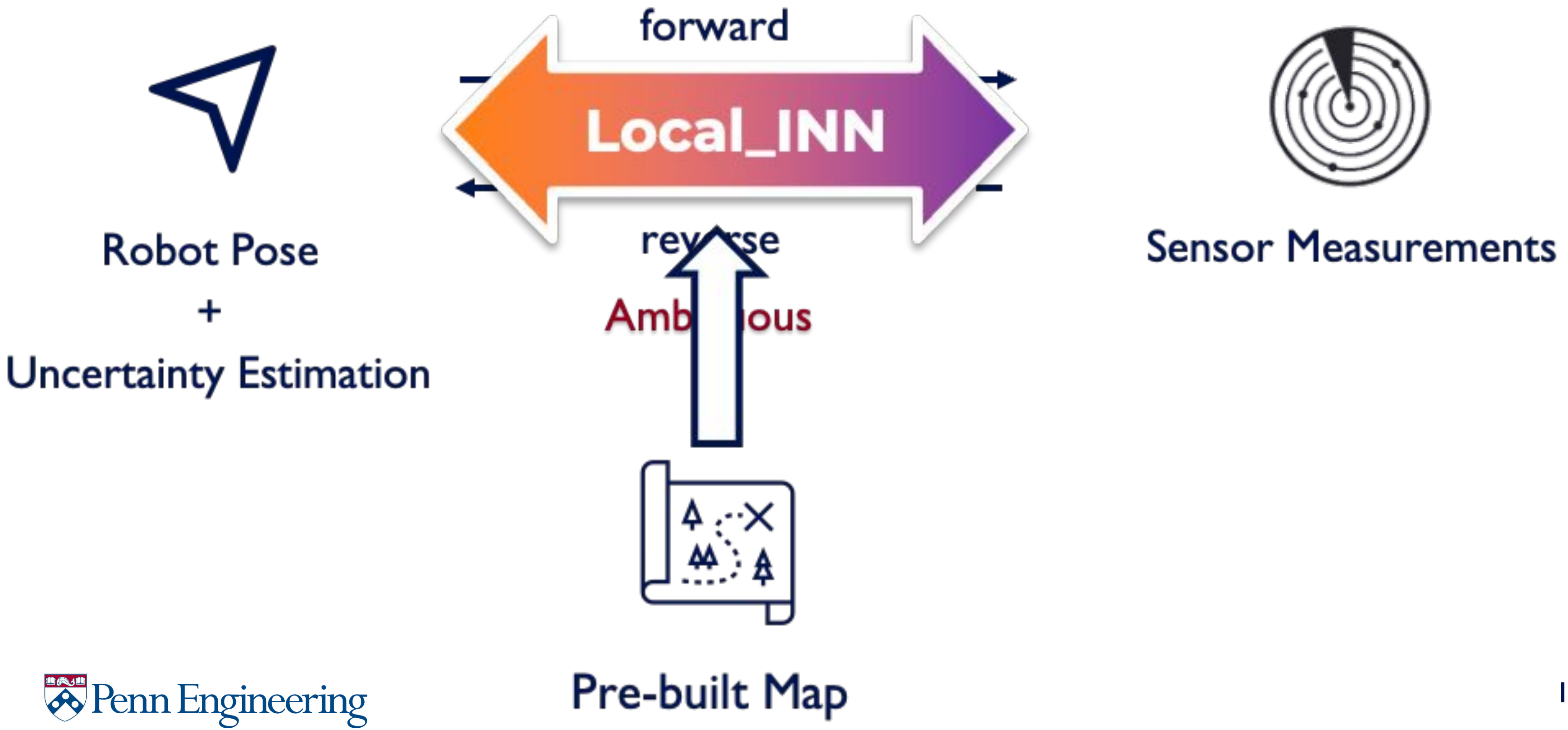
Indoor Localization with Map

- Commonly used methods --- Particle Filter
  - Need to simulate lidar for each particle
  - High computation and long latency
  - Only possible with 2D lidars
- Localization with Invertible Neural Networks
  - Small NN-based method: Cheap, Fast and Low latency.
  - Accurate localization and Expandable to 3D Lidar
  - Fast Converge in Global Localization.

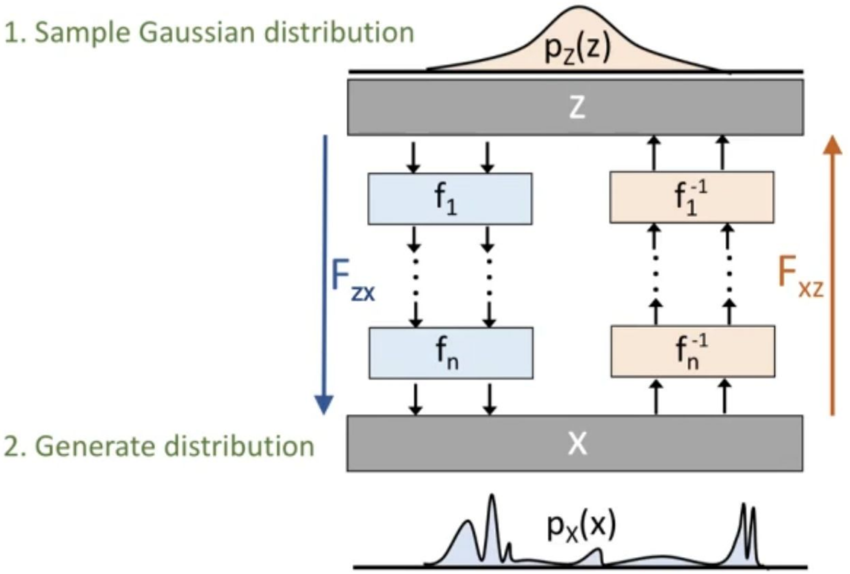
Localization as an Ambiguous Inverse Problem



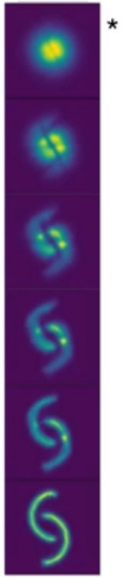
Localization as an Ambiguous Inverse Problem

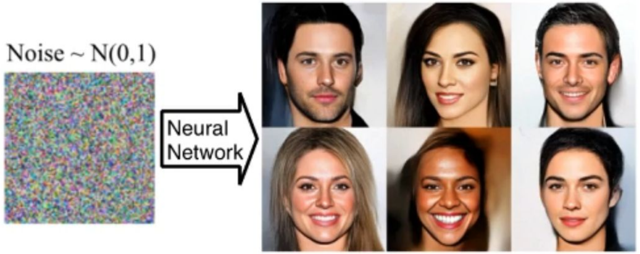


Invertible neural networks(INN) - Normalizing Flow



Example 1





Glow: Kingma, Dhariwal, *NeurIPS* 2018

Example 2

Invertible Structure

Dinh, Laurent, Jascha Sohl-Dickstein, and Samy Bengio. "Density estimation using real nvp." arXiv preprint arXiv:1605.08803 (2016).

Coupling Layer

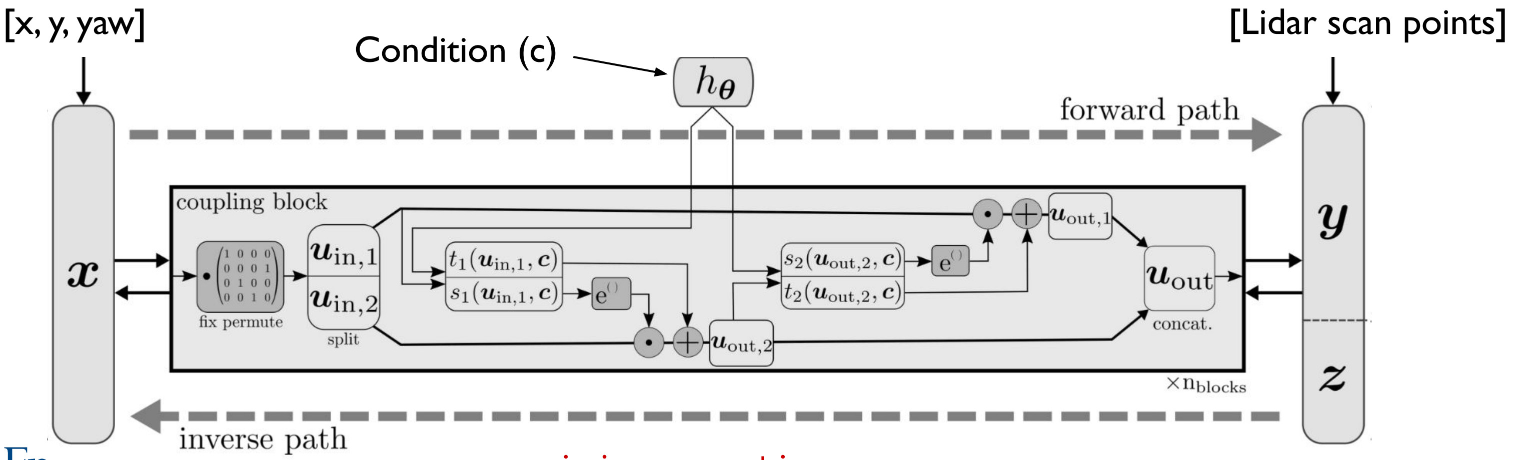
s() and t() are fully-connected layers with ReLU

- Forward:

$$\mathbf{v}_1 = \mathbf{u}_1 \odot \exp(s_2(\mathbf{u}_2)) + t_2(\mathbf{u}_2), \qquad \mathbf{v}_2 = \mathbf{u}_2 \odot \exp(s_1(\mathbf{v}_1)) + t_1(\mathbf{v}_1).$$

- Reverse:

$$\mathbf{u}_2 = (\mathbf{v}_2 - t_1(\mathbf{v}_1)) \odot \exp(-s_1(\mathbf{v}_1)), \qquad \mathbf{u}_1 = (\mathbf{v}_1 - t_2(\mathbf{u}_2)) \odot \exp(-s_2(\mathbf{u}_2)).$$



Penn Engineering

u_in is u, u_out is v

Localization as an Inverse Problem

X: robot pose x

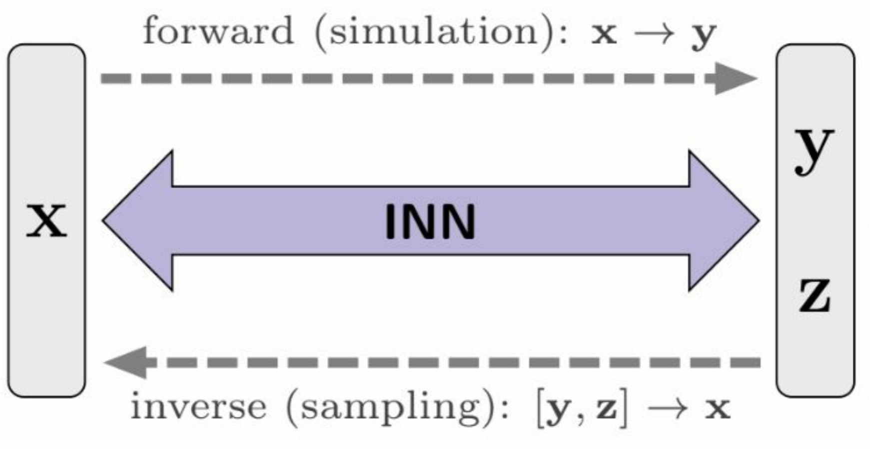
Condition: previous robot pose x'

Y: lidar scans s

Z: latent variables z ~N(0, 1)

Forward: x | x' ➝ s, z

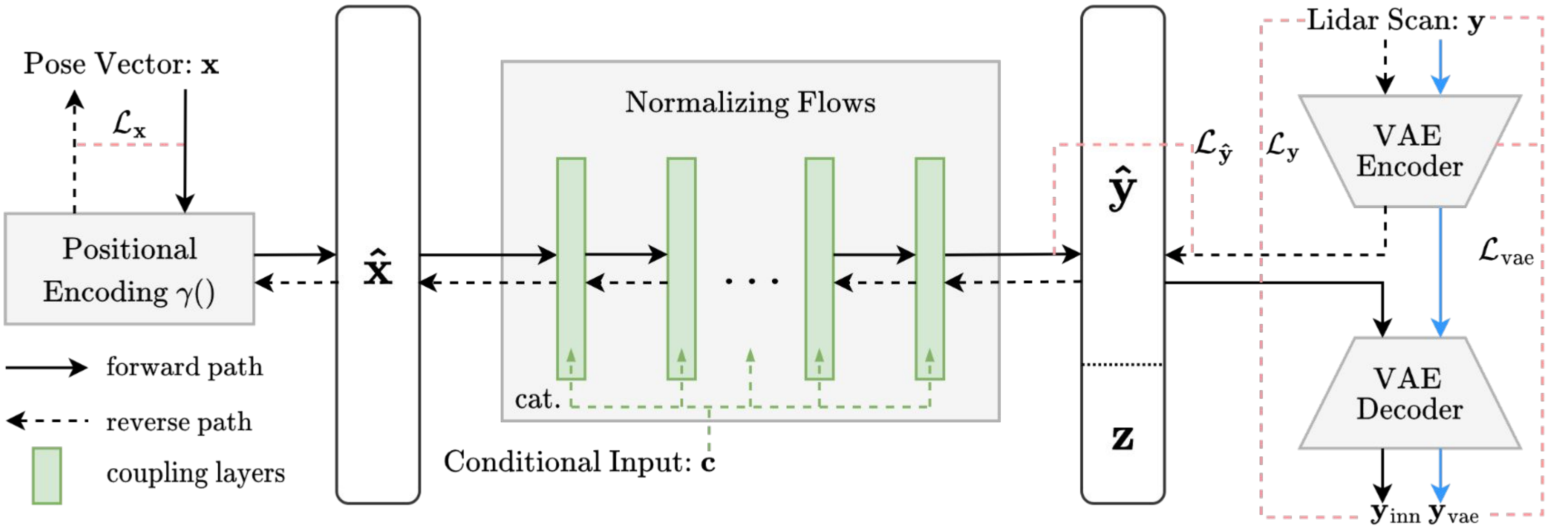
Reverse: s, z | x' ➝ x



Invertible Neural Network

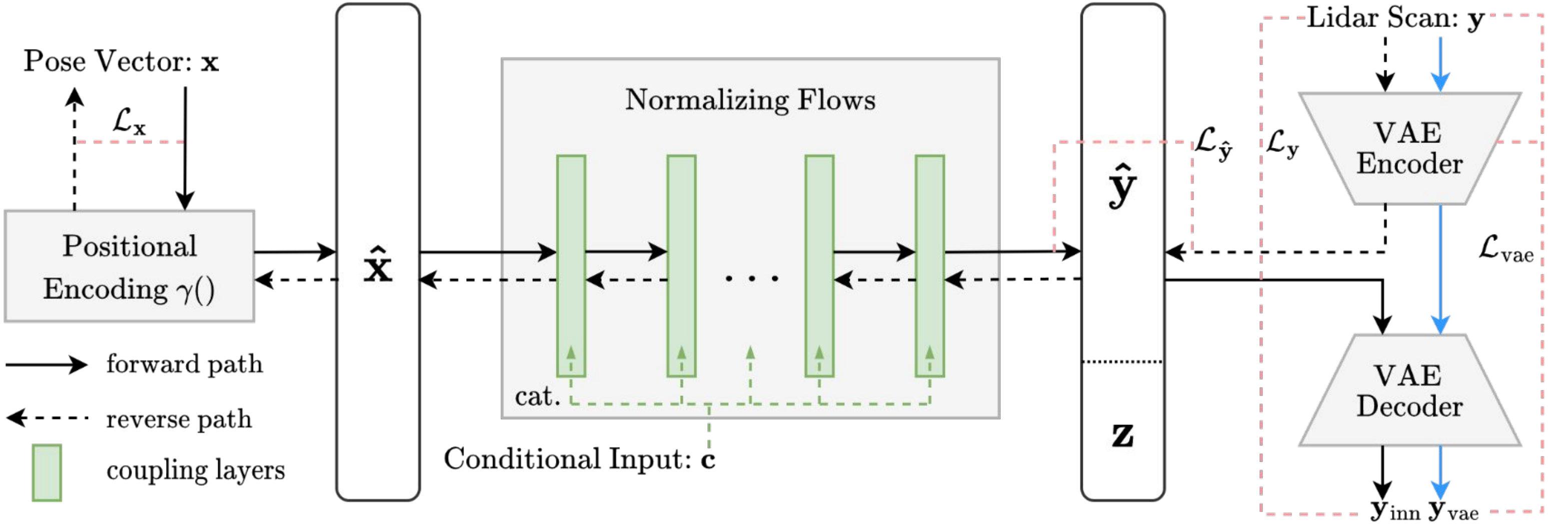
Ardizzone, Lynton, et al. "Analyzing inverse problems with invertible neural networks." arXiv preprint arXiv:1808.04730 (2018).

Structure



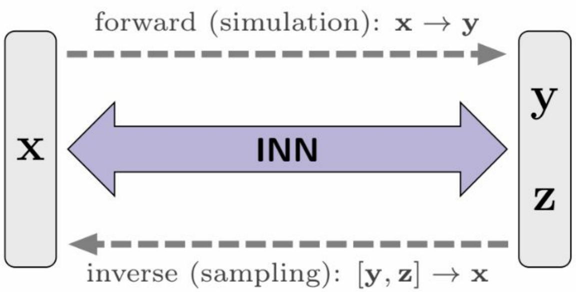
Positional Encoding:
$$\gamma(p) = \left(\sin(2^0 \pi p), \cos(2^0 \pi p), \dots, \sin(2^{L-1} \pi p), \cos(2^{L-1} \pi p) \right).$$

Structure



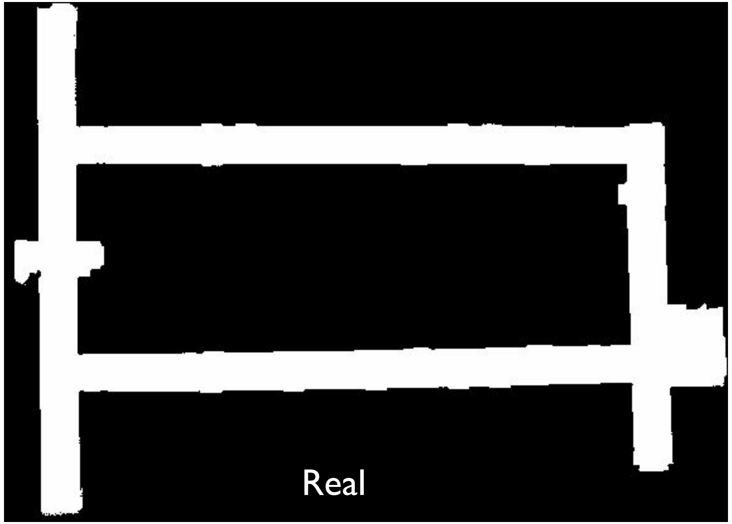
Conditional Input: $\mathbf{c} = \frac{\lceil N \mathbf{x}^{\mathrm{pre}} \rfloor}{N}, \hat{\mathbf{c}} = h_{\mathrm{cond}}(\gamma(\mathbf{c}))$

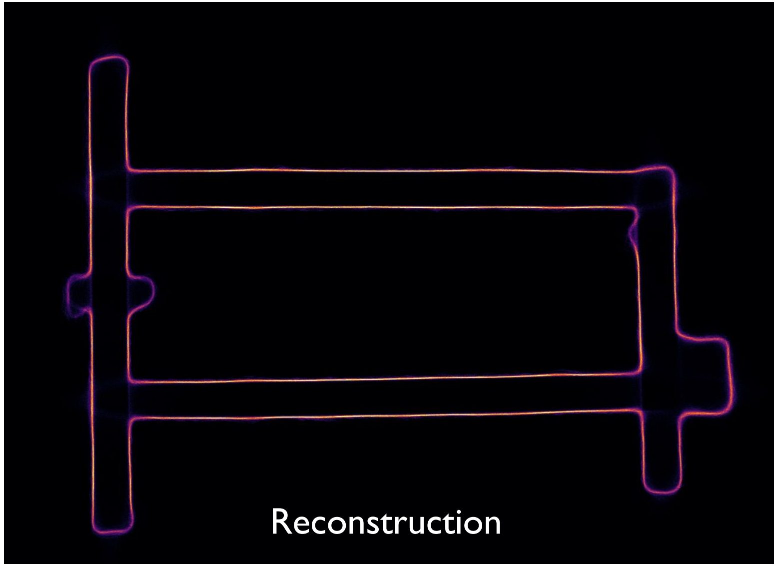
Map Reconstruction



- Random sampling in the state space.
- Convert output lidar scans to the map coordinates.

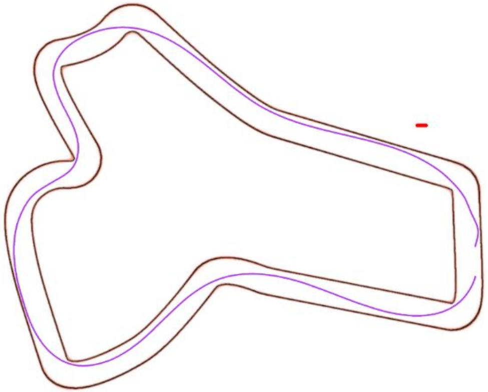
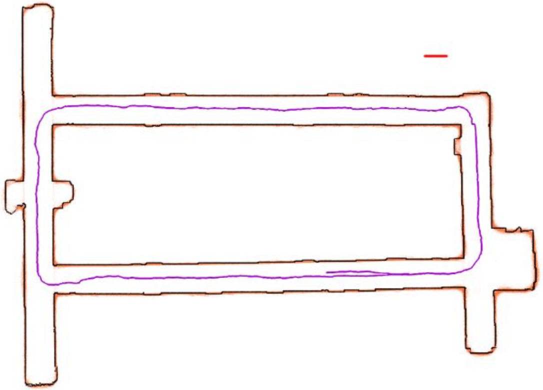
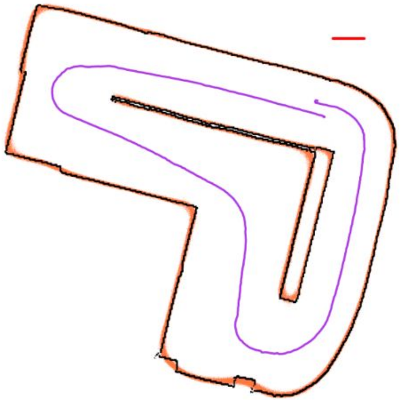
X - Robot state
Y - Lidar scans
Z - INN Latent





Indoor Hallway Map

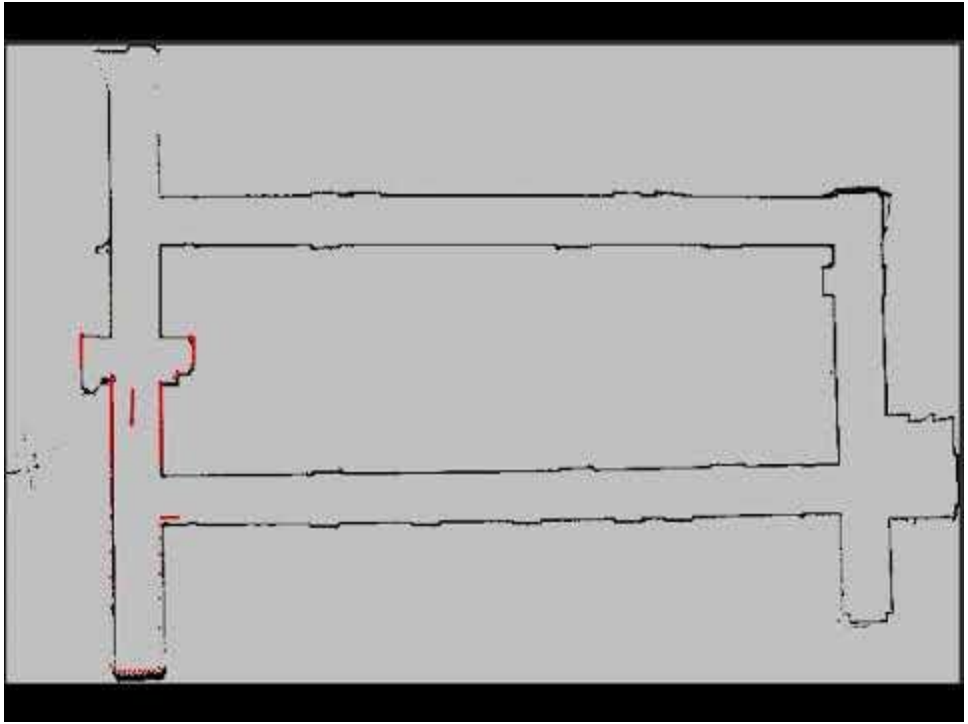
2D Experiments

	Race Track (Simulation)	Hallway (Real)	Outdoor (Real)
Original Map Reconstruction Test Trajectory			

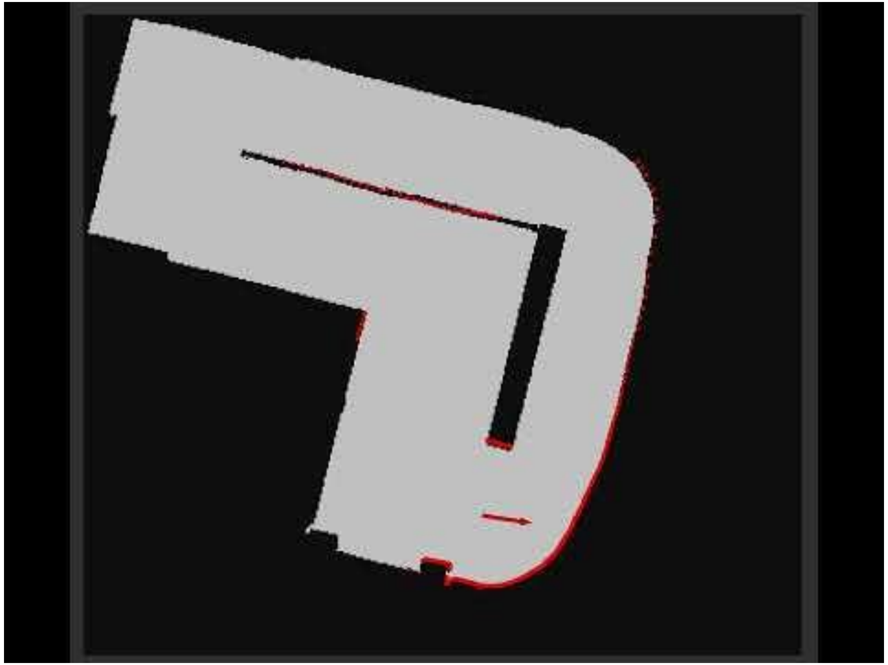
2D Experiments

F1tenth Car

2D Experiments

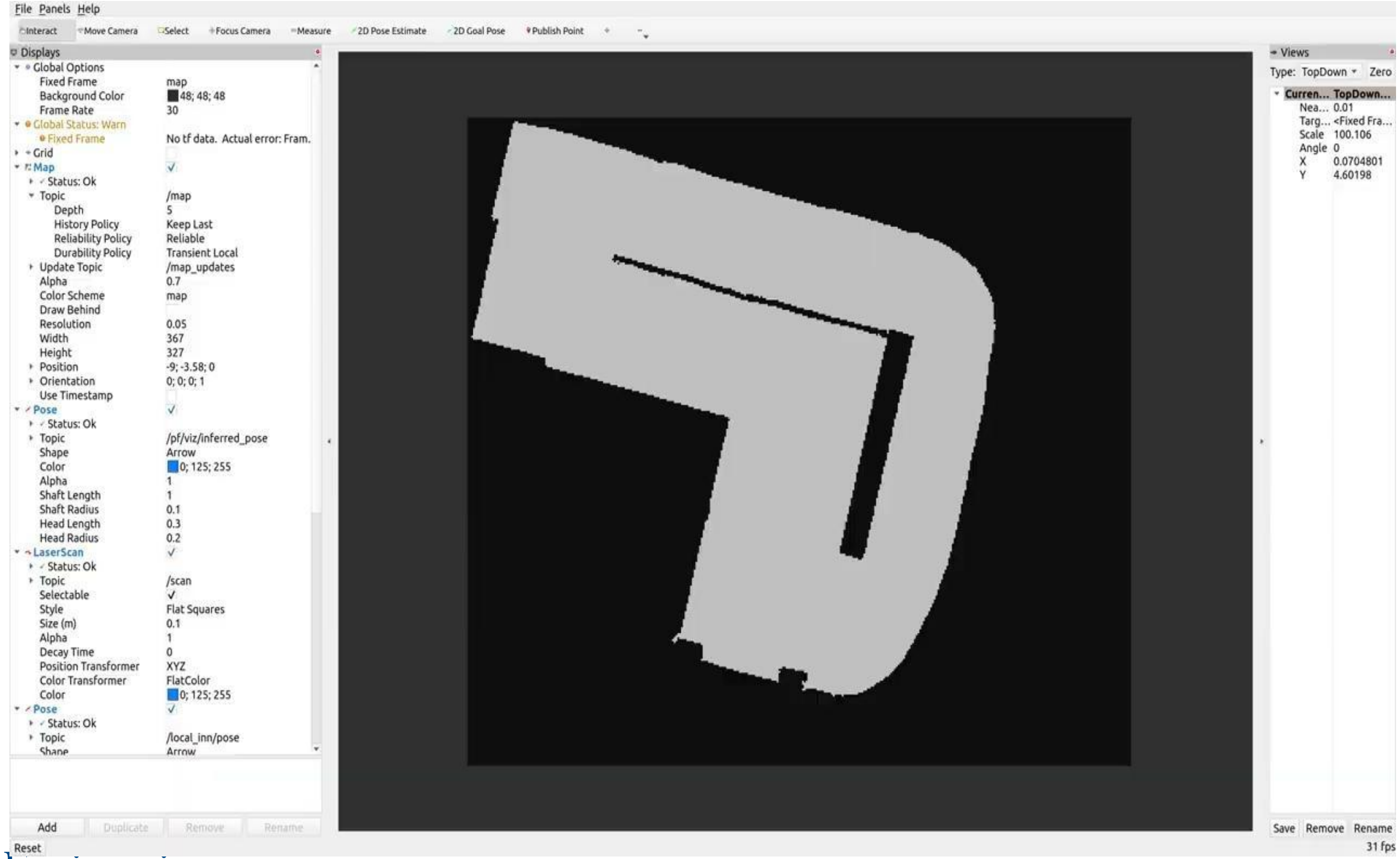


Hallway



Outdoor

2D Experiments (Local_INN vs. Particle filter)



Real-world Localization on the F1tenth Car

2D Experiments

TABLE I
MAP RECONSTRUCTION AND LOCALIZATION ERRORS WITH 2D LIDAR

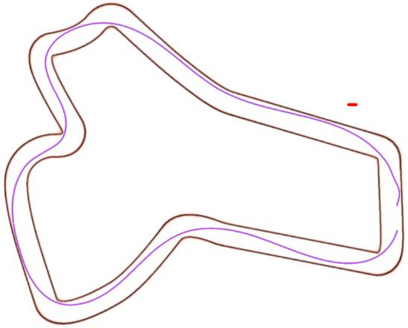
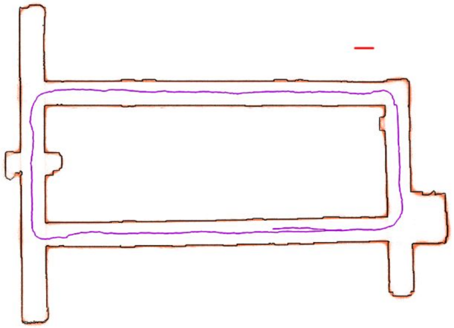
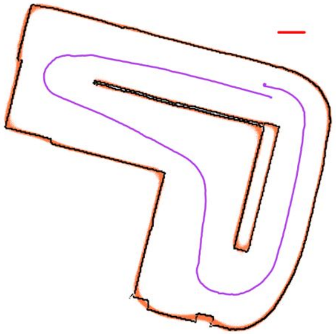
	Race Track (Simulation)		Hallway (Real)		Outdoor (Real)	
Original Map Reconstruction Test Trajectory						
	xy(m)	$\theta(^\circ)$	xy(m)	$\theta(^\circ)$	xy(m)	$\theta(^\circ)$
Online PF (1m/s)	$0.045 \pm \mathbf{0.058}$	0.400 ± 0.512	$\mathbf{0.039} \pm \mathbf{0.066}$	$\mathbf{0.482} \pm 0.808$	$\mathbf{0.013} \pm \mathbf{0.018}$	$\mathbf{0.358} \pm \mathbf{0.456}$
Local_INN (1m/s)	0.050 ± 0.102	0.201 ± 0.532	0.196 ± 0.433	$0.528 \pm \mathbf{0.792}$	0.034 ± 0.047	0.924 ± 1.130
↑ + EKF	$\mathbf{0.039} \pm 0.077$	0.182 ± 0.464	0.093 ± 0.139	0.536 ± 0.797	0.034 ± 0.047	0.917 ± 1.129
↑ + TensorRT	0.039 ± 0.076	$\mathbf{0.177} \pm \mathbf{0.443}$	0.104 ± 0.159	0.547 ± 0.802	0.033 ± 0.046	0.930 ± 1.142
Online PF (5m/s)	0.139 ± 0.168	1.463 ± 2.107	$\mathbf{0.071} \pm \mathbf{0.117}$	0.943 ± 1.738	0.033 ± 0.047	0.940 ± 1.371
Local_INN+EKF (5m/s)	$\mathbf{0.034} \pm \mathbf{0.056}$	$\mathbf{0.133} \pm \mathbf{0.284}$	0.100 ± 0.147	$\mathbf{0.565} \pm \mathbf{0.900}$	$\mathbf{0.032} \pm \mathbf{0.046}$	$\mathbf{0.915} \pm \mathbf{1.130}$

TABLE II
RUNTIME COMPARISONS ON NVIDIA JETSON NX

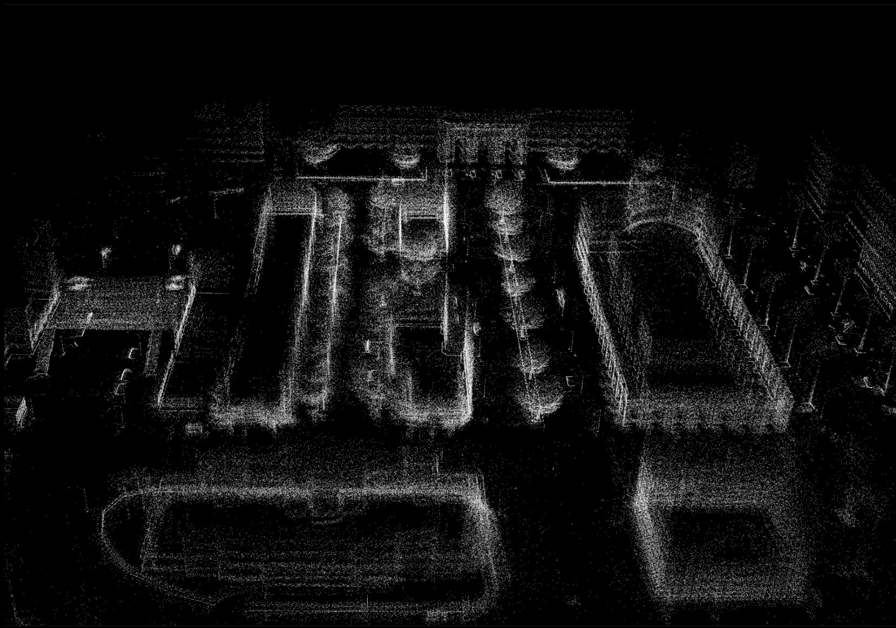
Online PF	45 Hz
Local_INN (Pytorch)	48 Hz
Local_INN+TensorRT	**270** Hz

3D Experiments

TABLE III
COMPARISON OF LOCALIZATION RMS ERRORS WITH 3D LIDAR

Methods (xy[m], θ[°])	CARLA	Mulran	Apollo
Local_INN in-session	$0.27, 0.12$	$0.29, 0.24$	$0.50, 0.26$
Local_INN out-session	$-, -$	$1.41, 1.00$	$1.22, 0.53$
Chen et al.	$0.48, 3.87$	$0.83, 3.14$	$0.57, 3.40$
RaLL (Yin et al.)	$-, -$	$1.27, 1.50$	$-, -$

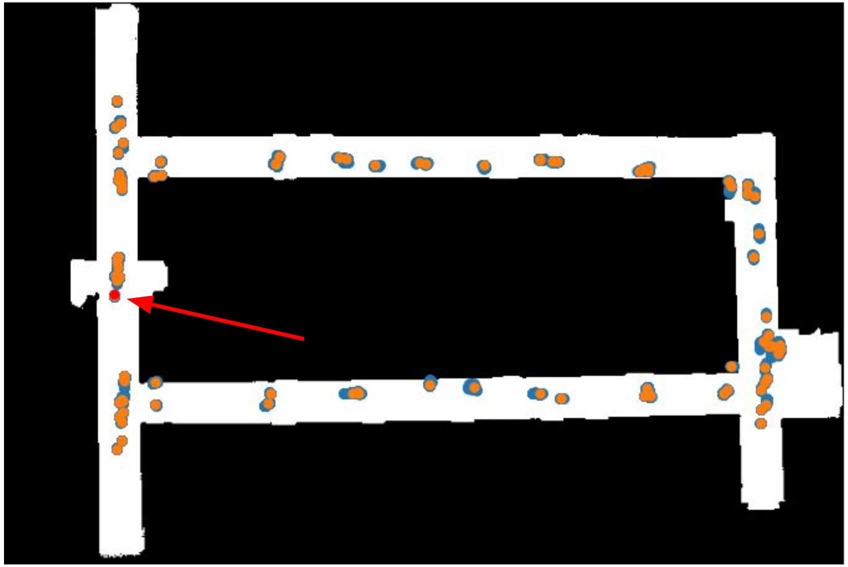
3D Experiments - Carla (simulation)

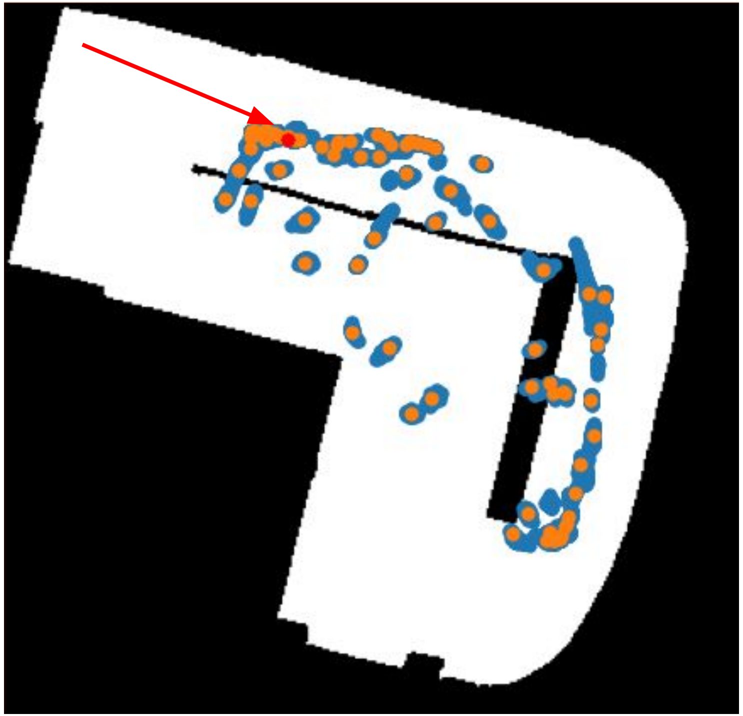


Reference

3D Experiments - Mulran (real)

Global Localization





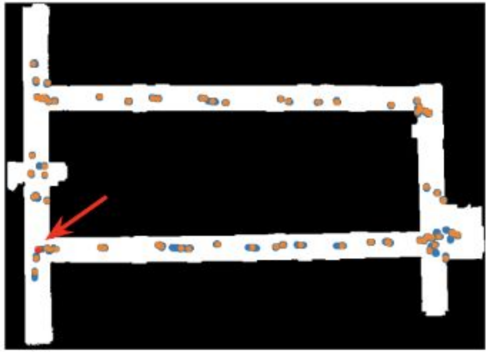
Blue: Inferred Pose Distribution
Orange: Inferred Pose Mean (Tracking)
Red: Correct Pose
Cyan: Converged Pose

Fast converge

Global Localization with Local_INN

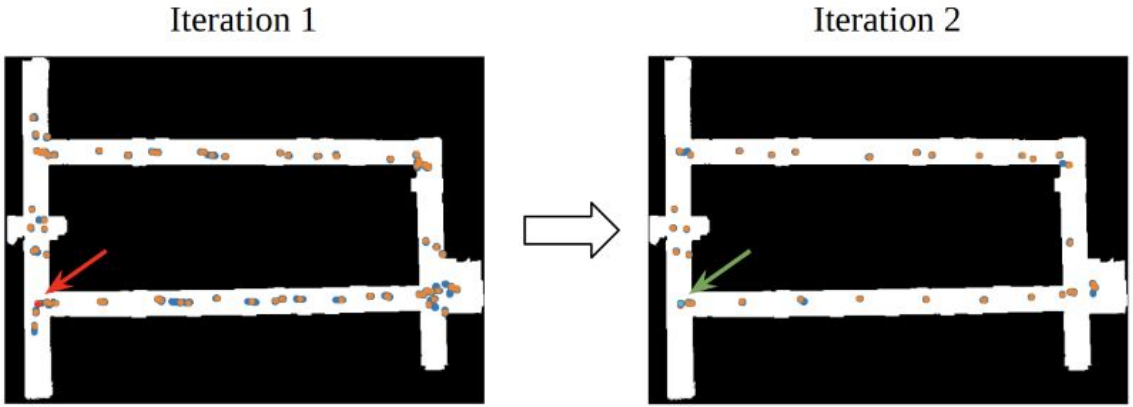
- During the global localization, we have multiple starting pose hypotheses, and we use incoming sensor measurements to narrow them down.

Iteration 1



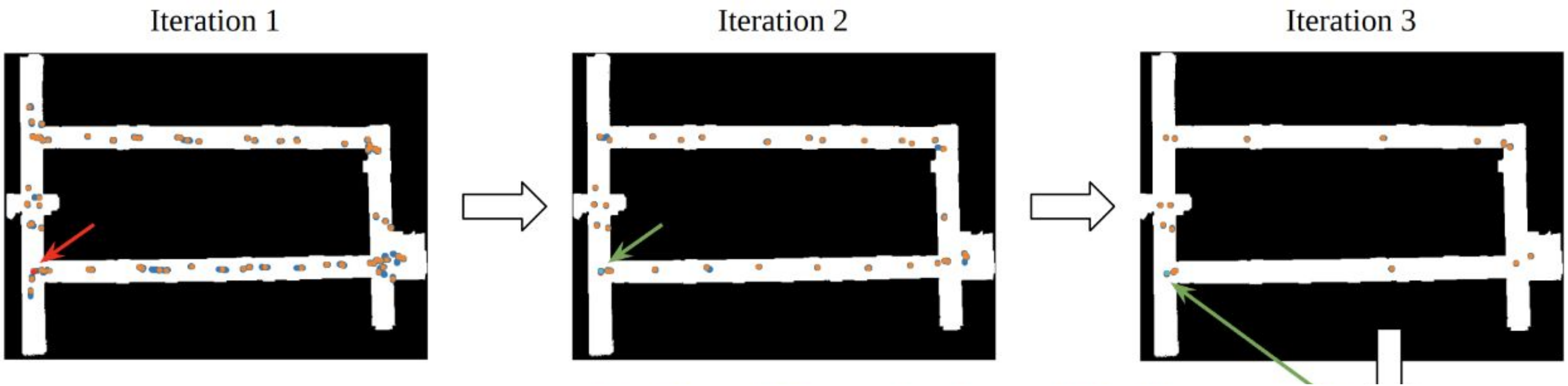
Global Localization with Local_INN

- During the global localization, we have multiple starting pose hypotheses, and we use incoming sensor measurements to narrow them down.



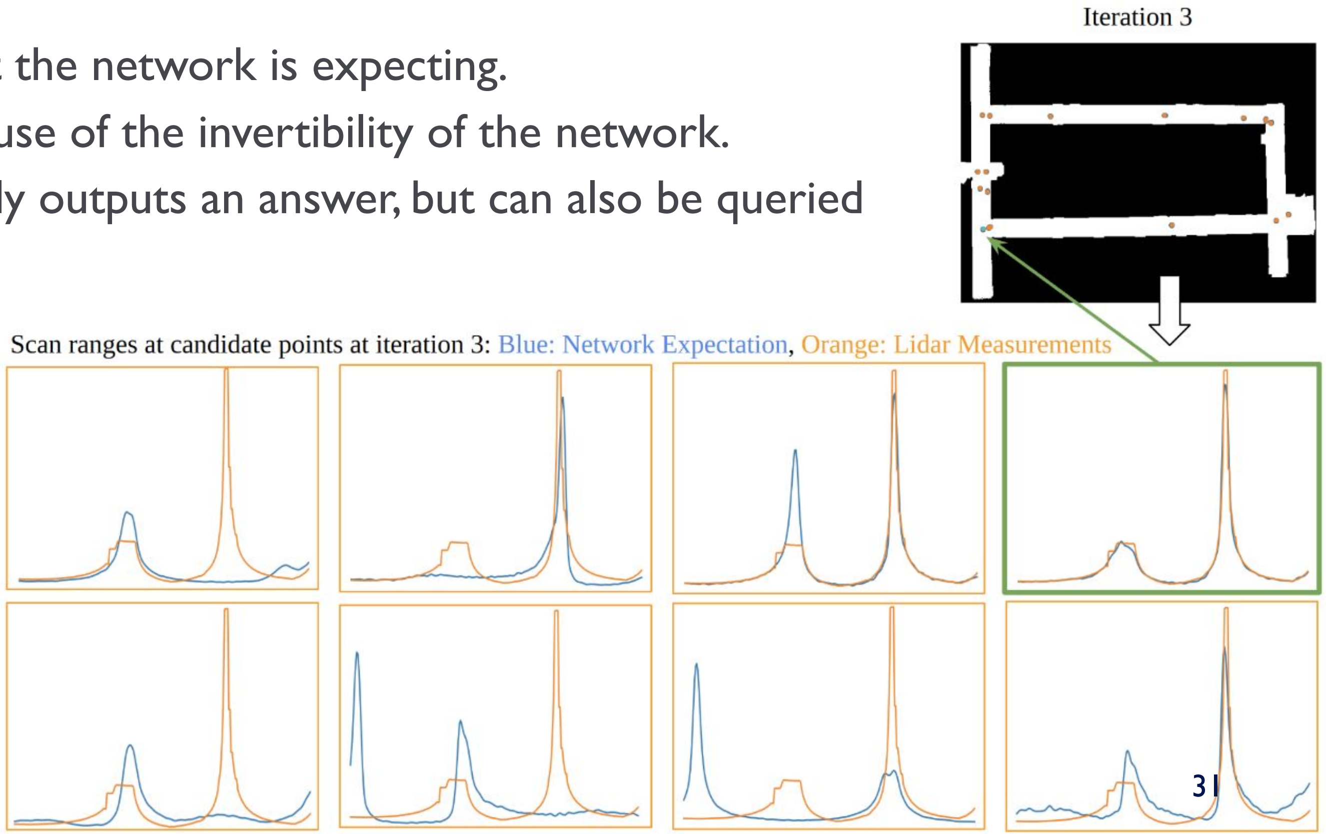
Global Localization with Local_INN

- How do we know which one is correct?



Global Localization with Local_INN

- We can look at what the network is expecting.
- This is possible because of the invertibility of the network.
- The network not only outputs an answer, but can also be queried for a reason.



Penn Engineering

Global Localization with Local_INN

Algorithm 1 Local_INN Global Localization

$n \leftarrow N, m_i \leftarrow M, w_i \leftarrow 1/M$ **for** $i = 1 \dots n$
$\mathcal{X}^{\text{rand}} \leftarrow$ random_sample$(\mathcal{S}, n)$
$\mathcal{C}_0 \leftarrow$ convert_to_cond_inputs$(\mathcal{X}^{\text{rand}})$
while *new LiDAR scan* $\mathbf{y}_{t+1}$ *coming* **do**
  for $i = 1 \dots n_t$ **do**
    $\mathbf{x}_{t+1,i} \leftarrow$ Local_INN_reverse$(\mathbf{y}_{t+1}, \mathbf{c}_i, m_i)$
    $\mathcal{X}_{t+1}$.append$(\mathbf{x}_{t+1,i})$
    $\mathbf{y}_{\text{inn},i} \leftarrow$ Local_INN_forward$(\mathbf{x}_{t+1,i}, \mathbf{c}_i)$
    $w_i \leftarrow 1/\|\mathbf{y}_{\text{inn},i} - \mathbf{y}_{t+1}\|_1$
  end for
  $\mathcal{C}_{t+1} \leftarrow$ convert_to_cond_inputs$(\mathcal{X}_{t+1})$
  $n_{t+1} \leftarrow |\mathcal{C}_{t+1}|$
  for $i = 1 \dots n_{t+1}$ **do**
    $m_i \leftarrow$ normalized$(w_i) n_{t+1} M$
  end for
end while

GLOBAL LOCALIZATION SUCCESS RATES IN DIFFERENT ENVIRONMENTS AT ITERATION 10

Map	Converged	Tracking	Δ_{xy}, Δ_θ
Race Track	79.5%	99.5%	0.075, 0.274
Hallway	66.4%	91.1%	0.258, 0.538
Outdoor	98.5%	100%	0.049, 0.911
Mulran	93.5%	95.0%	0.884, 0.454
Apollo	82.5%	83.0%	1.569, 0.122

Benefits of Local_INN

- Small NN-based method:
  - Cheap, Fast and Low latency.
- Accurate localization:
  - Comparable to particle filter at low speed; Higher precision than particle filter at high speed.
- Expandable to 3D Lidar:
  - No map file needed.
- Fast Converge in Global Localization.

Can we use it with images?

Can we do pose regression with INN?

The Pose Regression Problem (image -> 6DoF pose)

- PoseNet:
  - CNN + average pooling + linear
- After PoseNet:
  - different architectures,
  - better optimization methods, etc.
- Recently, with NeRF
  - More than 50% improvement: LENS, DFNet, etc.

What can NeRF do?

It can render photo-realistic images by interpolating between input frames.

Input Image → Train a NeRF Model → Render Any Angle!

What can NeRF do?

nerfstudio

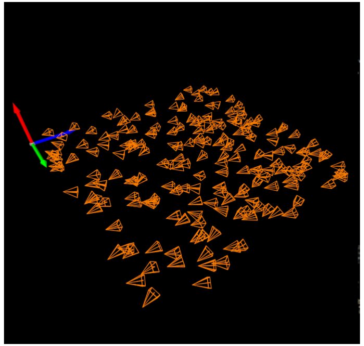
Penn Engineering

Pose Regression as An Inverse Problem

Trained NeRF

Guided Sampling

Rendering



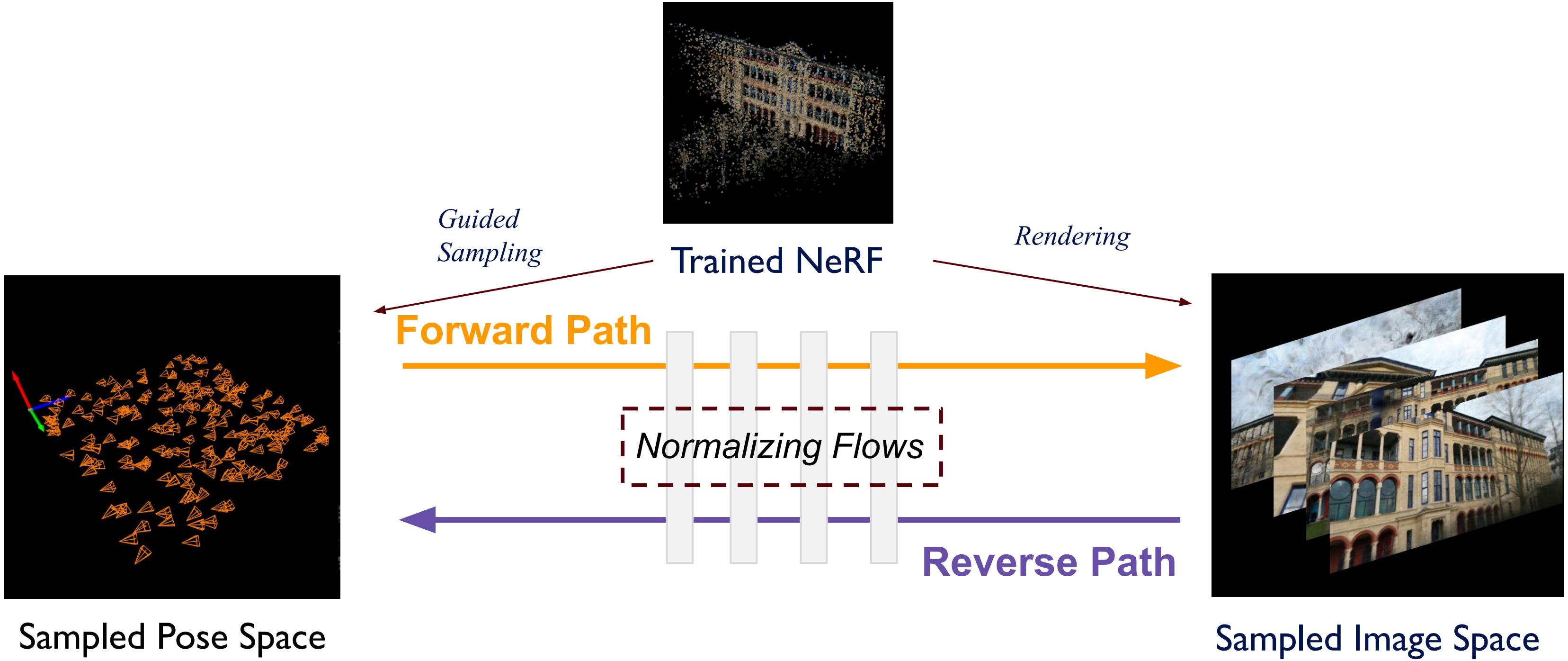
Sampled Pose Space

Forward Path

Reverse Path

Sampled Image Space

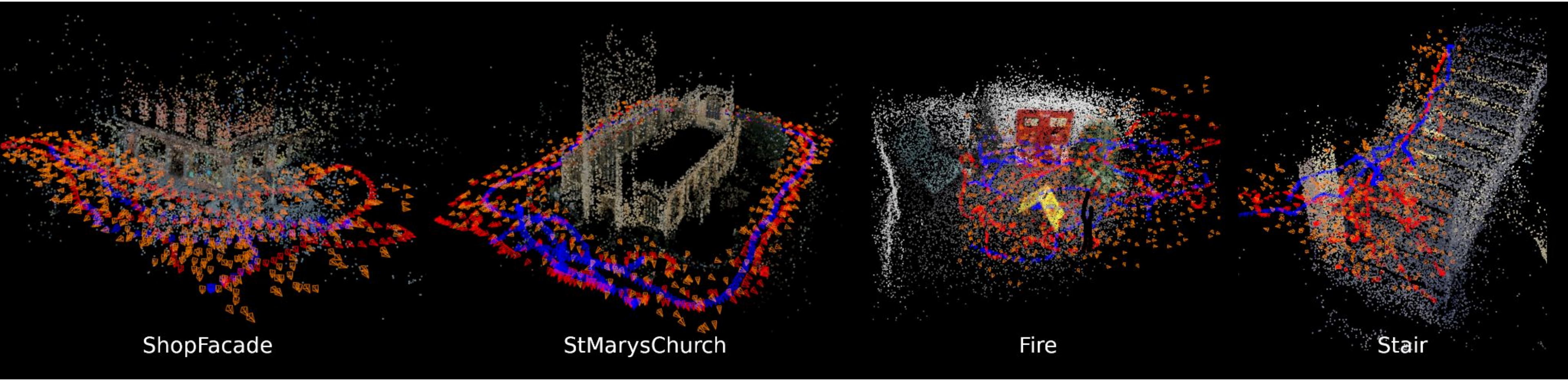
Pose Regression as An Inverse Problem



Data Preparation

Complete in 1 hour

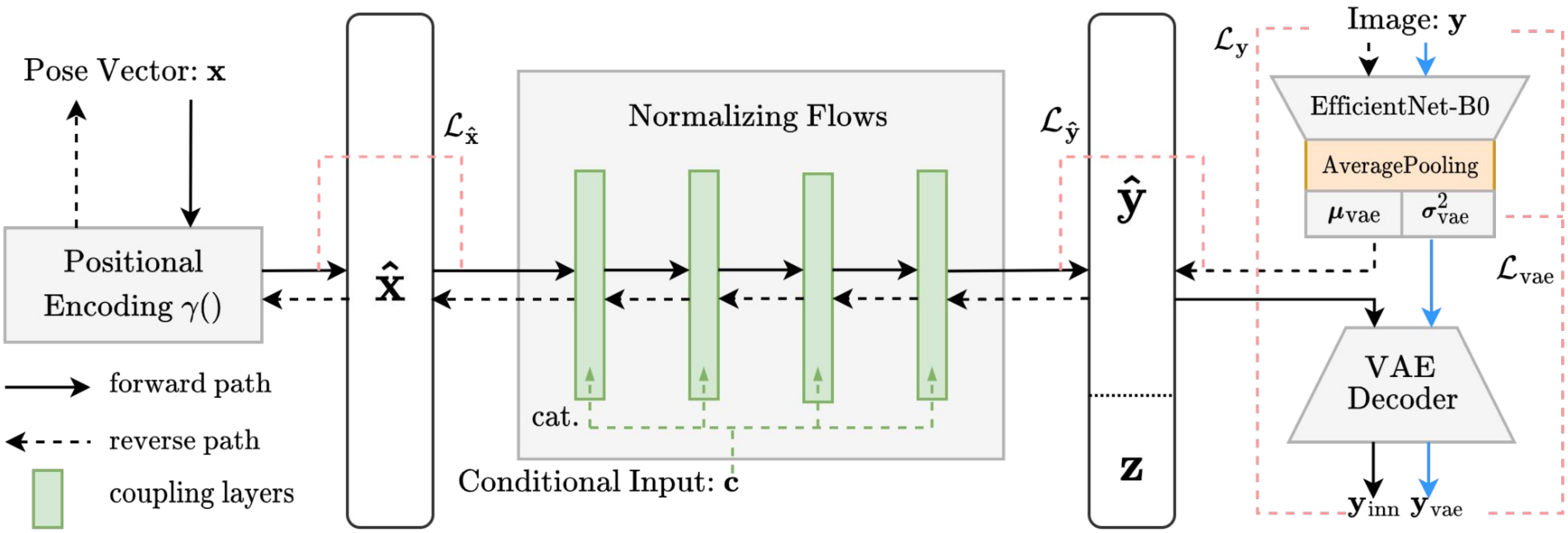
- Train a nerfacto with the training images.
- Output point cloud from NeRF model and sample 50k camera poses.
- Render 50k 160x90 images with NeRF.



Small pyramids represent training poses, testing poses, and sampled poses.

Penn Engineering

Structure



Pose Vector: $\mathbf{x}$
Positional Encoding $\gamma()$
$\hat{\mathbf{x}}$
$\mathcal{L}_{\hat{\mathbf{x}}}$
Normalizing Flows
cat.
Conditional Input: $\mathbf{c}$
$\mathcal{L}_{\hat{\mathbf{y}}}$
$\hat{\mathbf{y}}$
$\mathbf{z}$
$\mathcal{L}_{\mathbf{y}}$
Image: $\mathbf{y}$
EfficientNet-B0
AveragePooling
μ_{vae}
σ^2_{vae}
$\mathcal{L}_{\text{vae}}$
VAE Decoder
$\mathbf{y}_{\text{inn}}$
$\mathbf{y}_{\text{vae}}$
forward path
reverse path
coupling layers

Results

Table 1. Comparison of Median Errors of Camera Regression with Other State-of-the-Art Methods

(xy[m], θ[°])	Active Search[20]	SPPNet[19]	LENS[15]	DFNet$_{dm}$[5]	Pose_INN
Kings	0.42, 0.60	0.74, 1.00	**0.33**, **0.50**	0.43, 0.87	0.52, 0.75
Hospital	0.44, 1.00	2.18, 3.90	**0.44**, 0.90	0.46, 0.87	0.47, **0.83**
Shop	0.12, 0.40	0.59, 2.50	0.27, 1.60	**0.16**, **0.59**	0.25, 1.04
Church	0.19, 0.54	1.44, 3.35	0.53, 1.60	0.50, **1.49**	**0.47**, 1.58
Average	0.29, 0.63	1.24, 2.74	0.39, 1.25	**0.39**, **0.96**	0.43, 1.06
Chess	0.04, 2.00	0.12, 4.40	**0.03**, **1.30**	0.04, 1.48	0.05, 1.60
Fire	0.03, 1.50	0.22, 8.90	0.10, 3.70	**0.04**, **2.16**	0.10, 2.61
Heads	0.02, 1.50	0.11, 8.30	0.07, 5.80	**0.03**, **1.82**	0.06, 4.20
Office	0.09, 3.60	0.16, 5.00	**0.07**, **1.90**	0.07, 2.01	0.09, 3.16
Pumpkin	0.08, 3.10	0.21, 4.90	**0.08**, 2.20	0.09, 2.26	0.09, **1.86**
Kitchen	0.07, 3.40	0.21, 4.80	**0.09**, **2.20**	0.09, 2.42	0.10, 2.35
Stairs	0.03, 2.20	0.22, 7.20	0.14, 3.60	**0.14**, 3.31	0.18, **2.80**
Average	0.05, 2.50	0.18, 6.20	0.08, 3.00	**0.07**, **2.21**	0.09, 2.65

Table 2. Data Generation Strategy Comparison (Error Data from 7Scene)

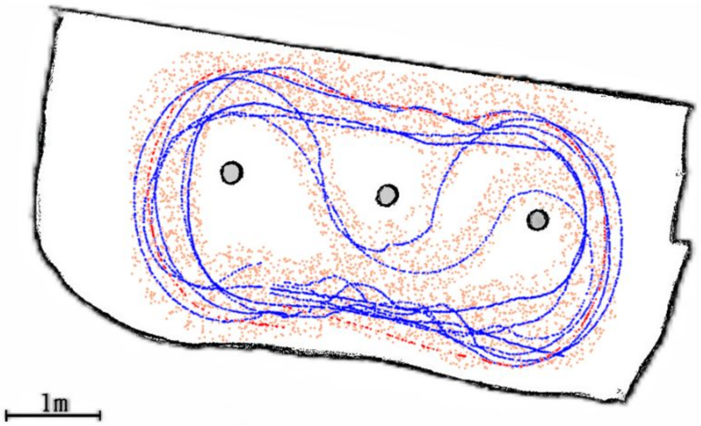
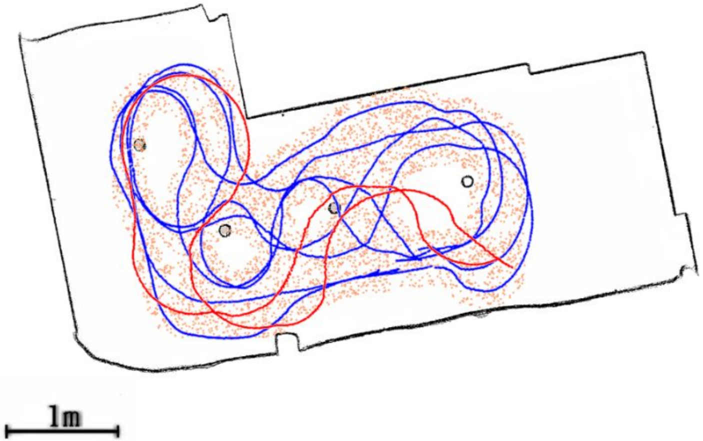
Model	Backbone Top-1 Acc.	Pose Error (m/°)	Synthetic Resolution	Rendering Cost	Generation Mode
LENS(EB3)	81.1%	0.08/3.00	High	Expensive	Offline
DFNet(EB0)	76.3%	0.08/3.47	Low	Cheap	Online
Pose_INN(EB0)	76.3%	0.09/2.65	Low	Cheap	Offline

Penn Engin

Real-world 2D Localization

154Hz on Jetson Xavier NX board

Table 3. Median Localization Errors with 2D LiDAR vs. Camera

	Indoor	Outdoor
train trajectory test trajectory sampled points		
	(xy[m], θ[°])	(xy[m], θ[°])
Online PF	**0.01**, 0.23	**0.02**, **0.36**
Pose_INN	0.02, 0.31	0.12, 0.72
Pose_INN + EKF	0.02, **0.22**	0.10, 0.65

Uncertainty Estimation

Table 4. Output Filtering with Uncertainty Estimation

(xy[m], θ[°])	Raw Mean Error	With Filtering
Kings	0.93, 1.02	**0.58**, **0.96**
Hospital	0.87, 1.14	**0.64**, **0.97**
Shop	0.59, 5.00	**0.20**, **1.04**
Church	0.81, 2.43	**0.52**, **1.21**
Average	0.84, 2.55	**0.68**, **1.87**

Thank you!