Research Summary

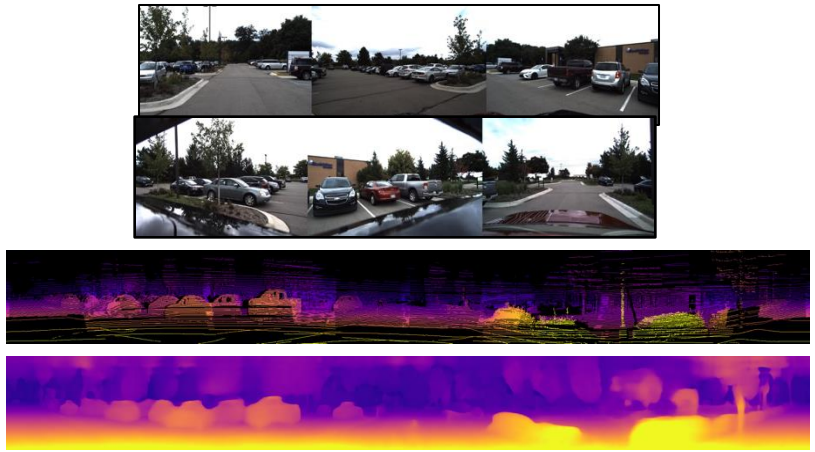
Hanyang University, Department of Automotive engineering

Chungwoo Lee

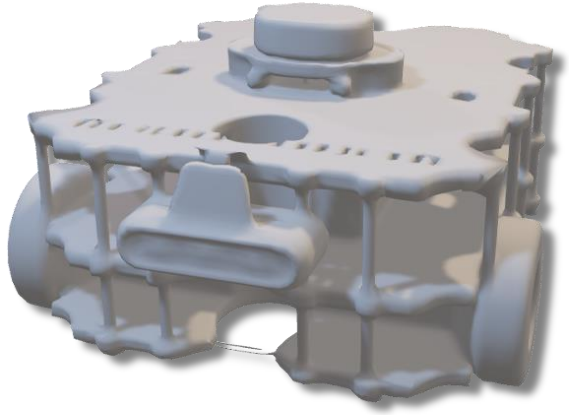
Contents

- **Autonomous PickingBot using LIDAR**

- **Variance-Integrated Panodepth for Cylindrical Depth Estimation**



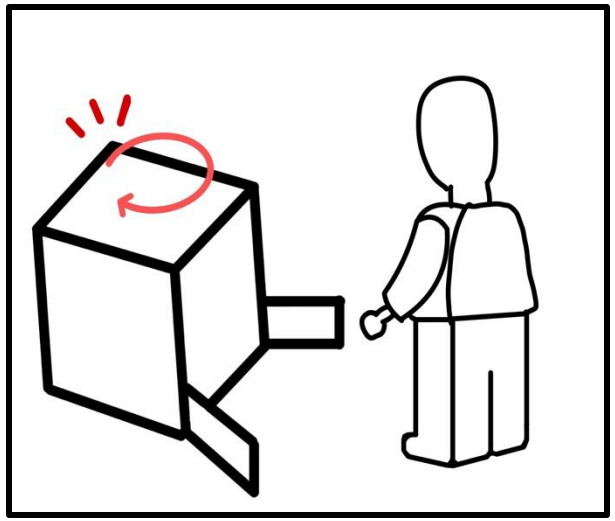
- **From Simulation to Reality: VLN on TurtleBot3**



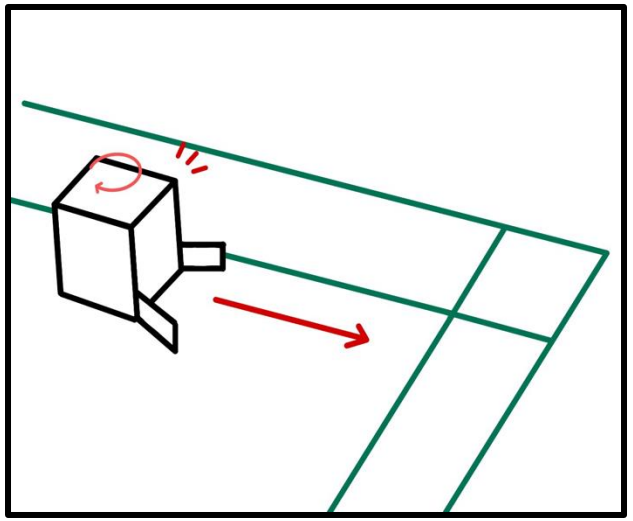
Autonomous PickingBot using LIDAR

Autonomous PickingBot using LIDAR

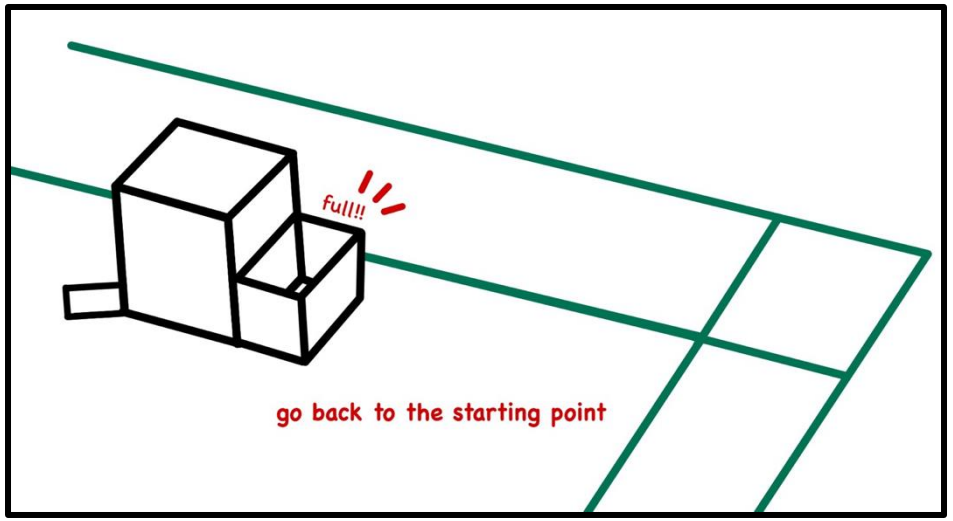
► **Core Features of the Shuttlecock Collecting Robot**



Obstacle Avoidance



Driving Along a Predefined Path



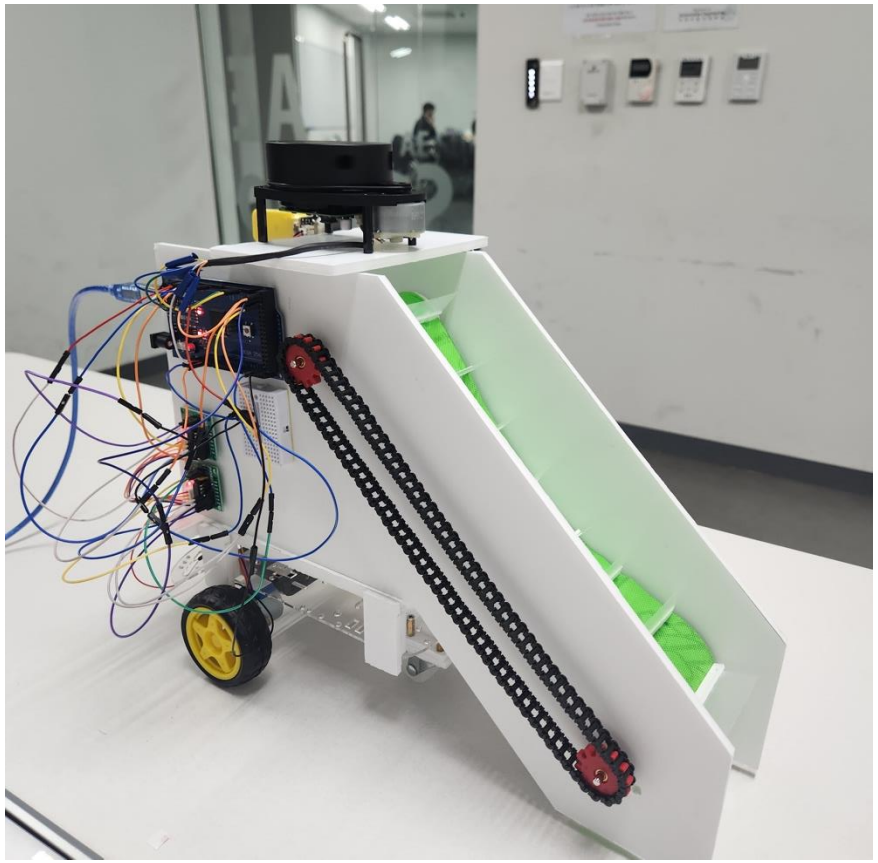
Returning to the Starting Point When the Shuttlecock Basket Is Full

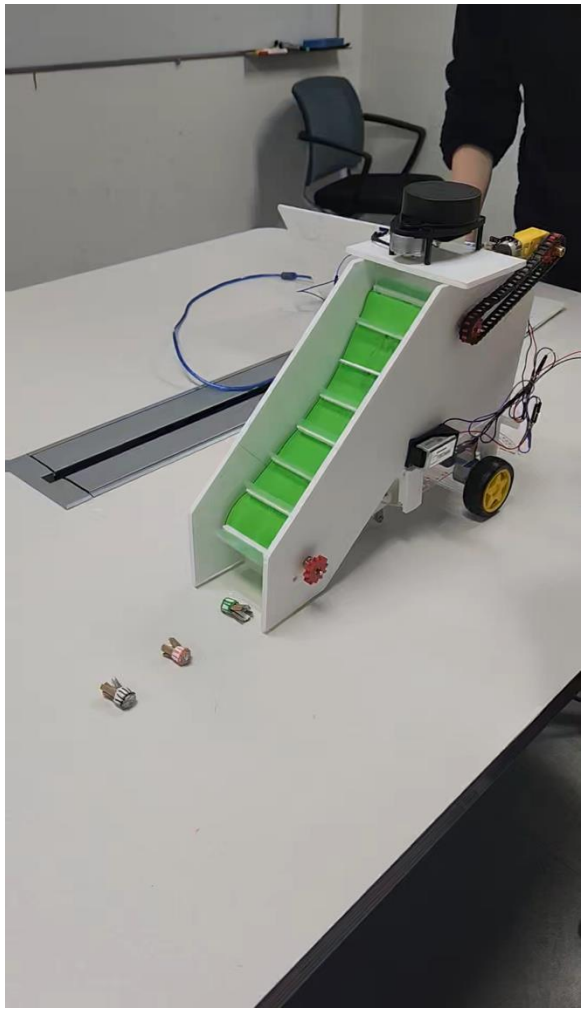
Introduction

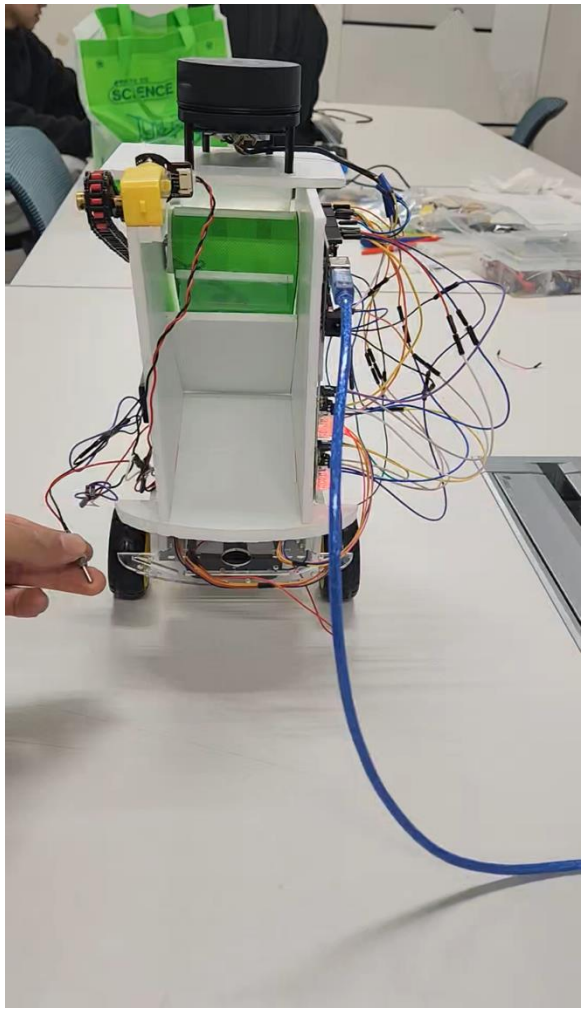
- **Project Background:**
  - **Accumulation of Shuttlecocks**
    During badminton practice, shuttlecocks often pile up in specific areas of the court.
  - **Manual Collection Challenges**
    The inconvenience of picking up shuttlecocks by hand was a major drawback, prompting the development of an autonomous robot.
- **Key Techniques:**
  - **Google Cartographer with 2D LIDAR**
    - Real-time localization and map generation
  - **LIDAR-Based Obstacle Detection**
    - Dynamically adjusts driving paths to avoid obstacles.

(By automating shuttlecock collection on the badminton court, the PickingBot significantly improves convenience. Future integration of computer vision aims to offer more precise detection and advanced autonomy.)

Autonomous PickingBot using LIDAR

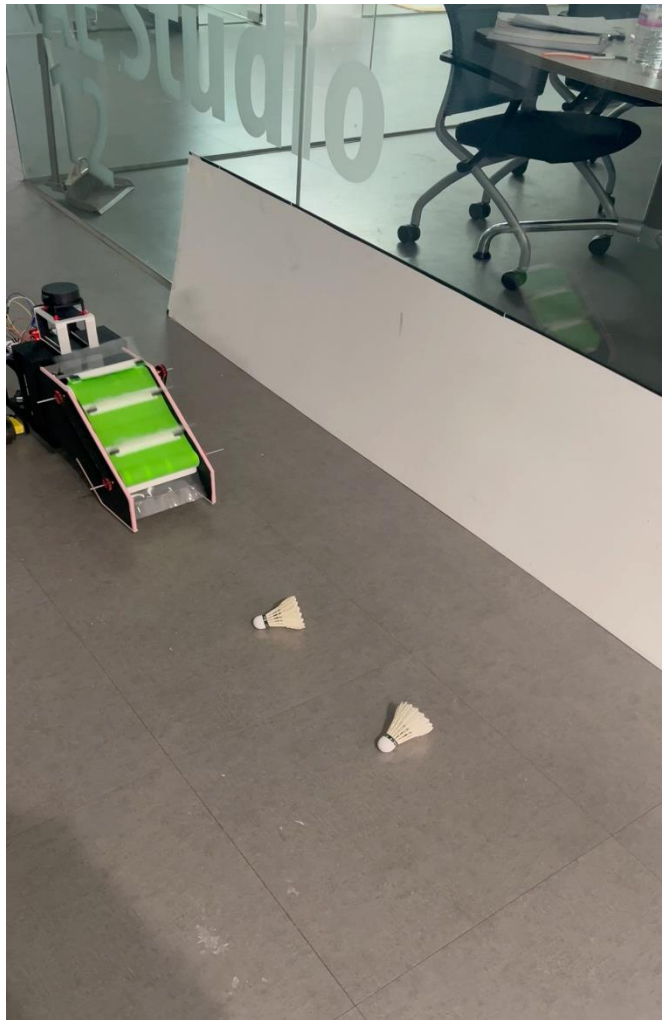






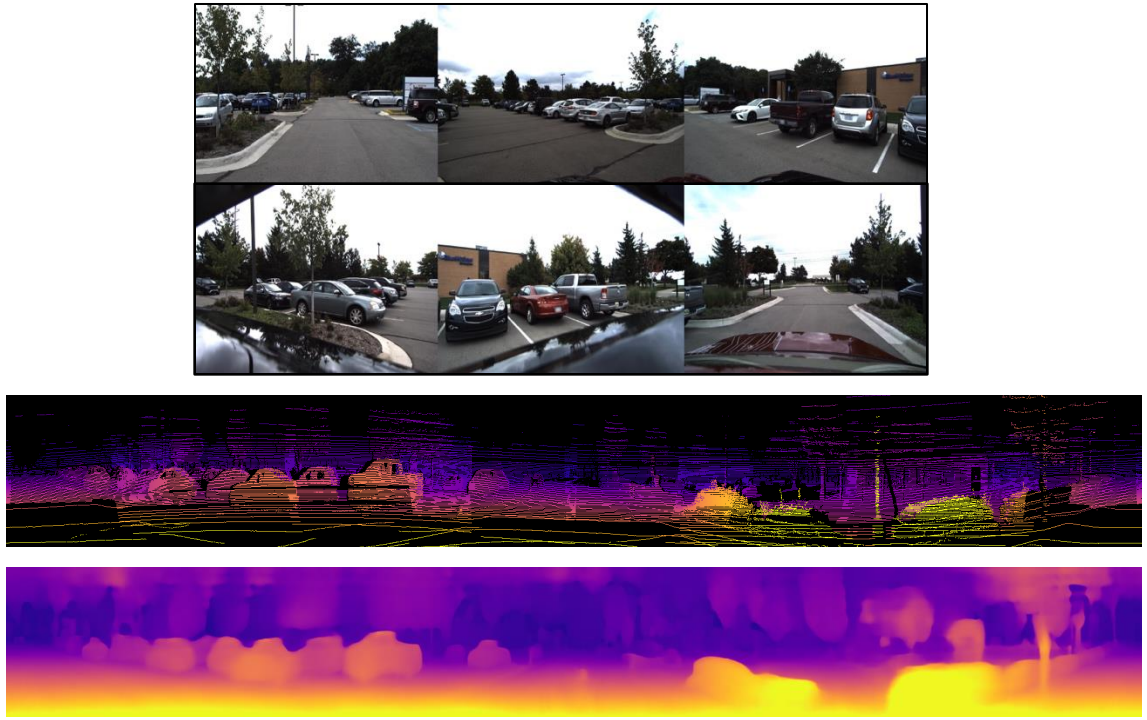
<Functional Prototype: Shuttlecock Collection Robot with LIDAR-Based Navigation>

Autonomous PickingBot using LIDAR



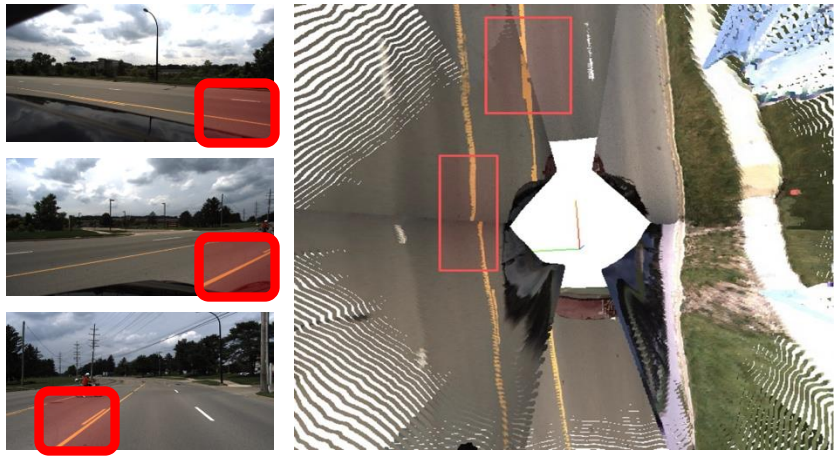
<Live Test: MintonBot Navigating and Collecting Shuttlecocks Using LIDAR>

Variance-Integrated Panodepth for Cylindrical Depth Estimation



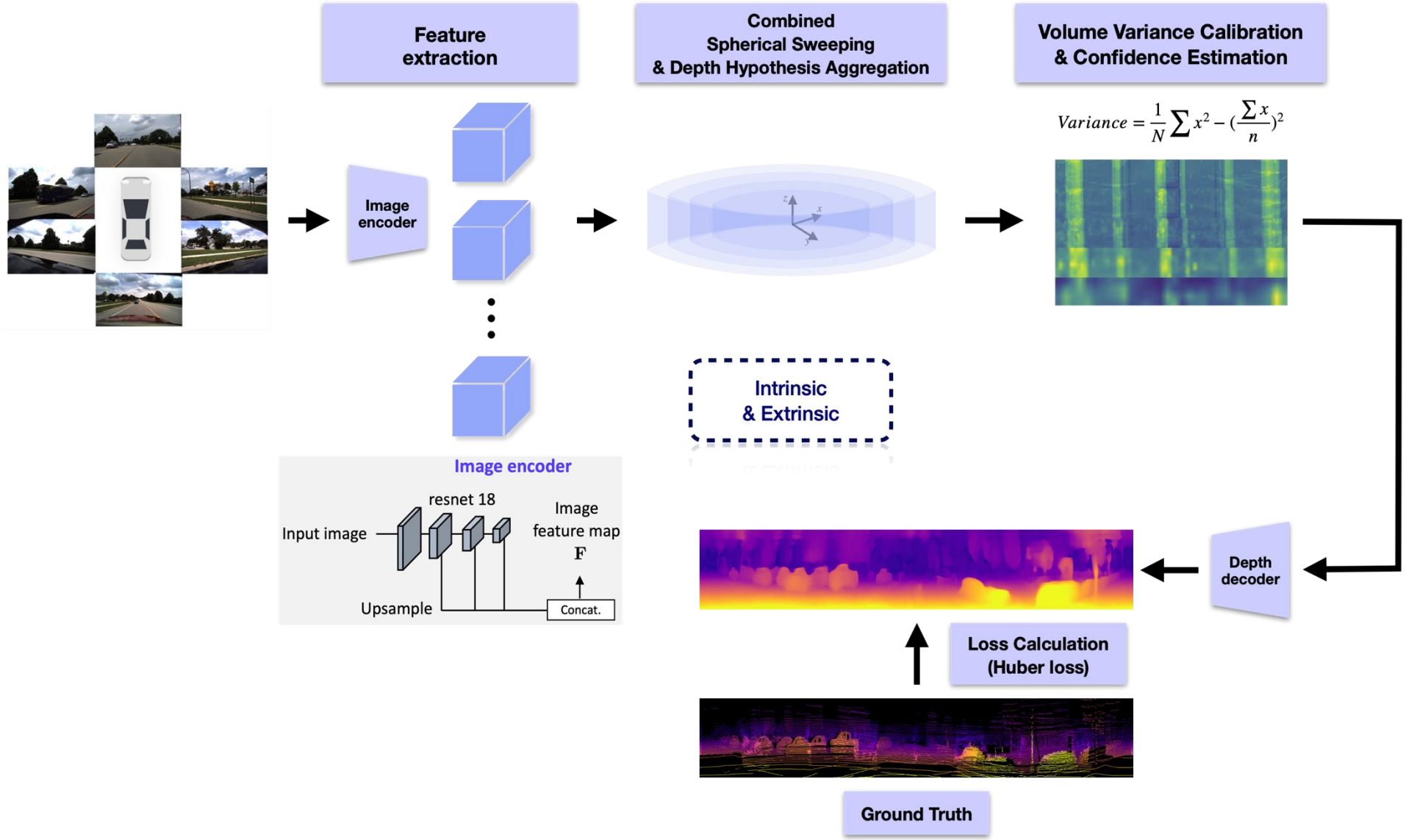
Variance-Integrated Panodepth for Cylindrical Depth Estimation

- **Limitations of Monocular Depth Estimation:**
  - Using only the front camera does not fully capture depth information of the surrounding, allowing only relative depth estimation.
- **Limitations of Surround View Monocular Depth Estimation**
  - Recently, studies have been proposed using surround-view cameras, but scale misalignment issues arise in the overlapping areas between cameras.
- **Variance-Integrated Panodepth**
  - To address the limitations of insufficient boundary information and scale misalignment, Panodepth is proposed using a cylindrical projection method.

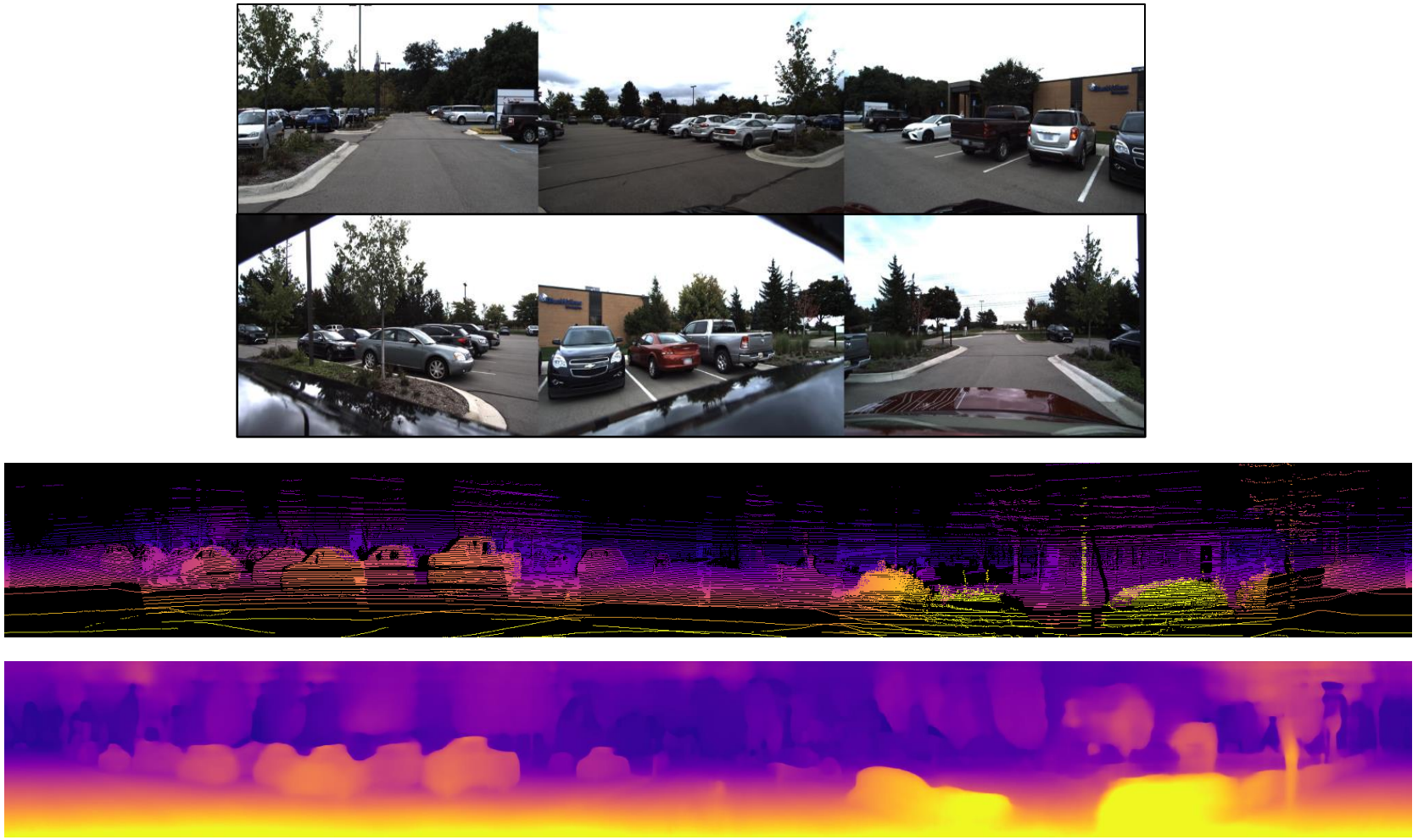


Scale misalignment on overlap areas

Variance-Integrated Panodepth for Cylindrical Depth Estimation

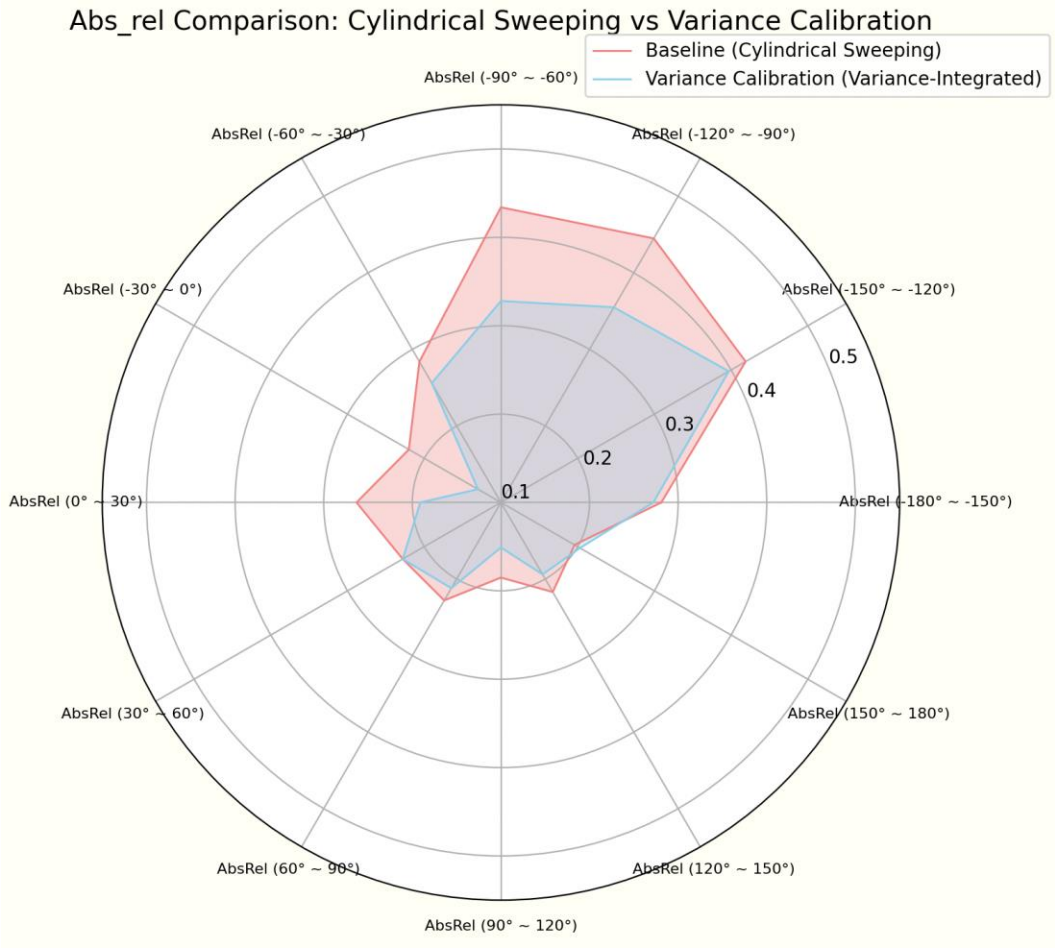


Variance-Integrated Panodepth for Cylindrical Depth Estimation



Ground Truth(GT) and estimated depth map results generated from 6 surround view images.

Variance-Integrated Panodepth for Cylindrical Depth Estimation



Angle-wise comparison of Abs_rel between Baseline (Cylindrical Sweeping) and Variance-Integrated Method

	AbsRel	SqrRel	RMSE	RMSElog	SiLog	A1	A2	A3
Cylindrical Sweeping	0.217	4.217	14.543	0.362	34.597	0.698	0.866	0.927
Variance Confidence	0.209	3.778	14.2	0.341	32.71	0.702	0.871	0.934

Performance comparison between the Baseline (Cylindrical Sweeping) model and the model with added Variance Confidence

- **Smoother and More Robust Results**
  - Integrating the Variance Confidence method into Cylindrical Sweeping leads to smoother, more reliable depth predictions.
- **Reduced Error Metrics**
  - The variance-integrated approach helps minimize errors and improve overall accuracy.
- **Enhanced Stability**
  - Improved consistency and stability, as demonstrated in the radial comparison.

Variance-Integrated Panodepth for Cylindrical Depth Estimation

Conclusions & Future Works

- **Improvement in consistency of Boundary Information and Consistency in Overlap Regions**
- **Model Lightweighting and Computational Efficiency Improvement**
  - By applying a method that divides the cylindrical coordinate system in half for projection, both lightweighting and computational efficiency can be enhanced.
- **Adaptive Depth Hypothesis Adjustment**
  - Research on adjusting the depth hypothesis on a pixel-wise basis, rather than at fixed intervals.
  - Placing denser hypotheses in areas close to the actual depth based on the predicted depth score.
  - Possibility of placing sparse depth hypotheses in distant locations.

From Simulation to Reality: VLN on TurtleBot3

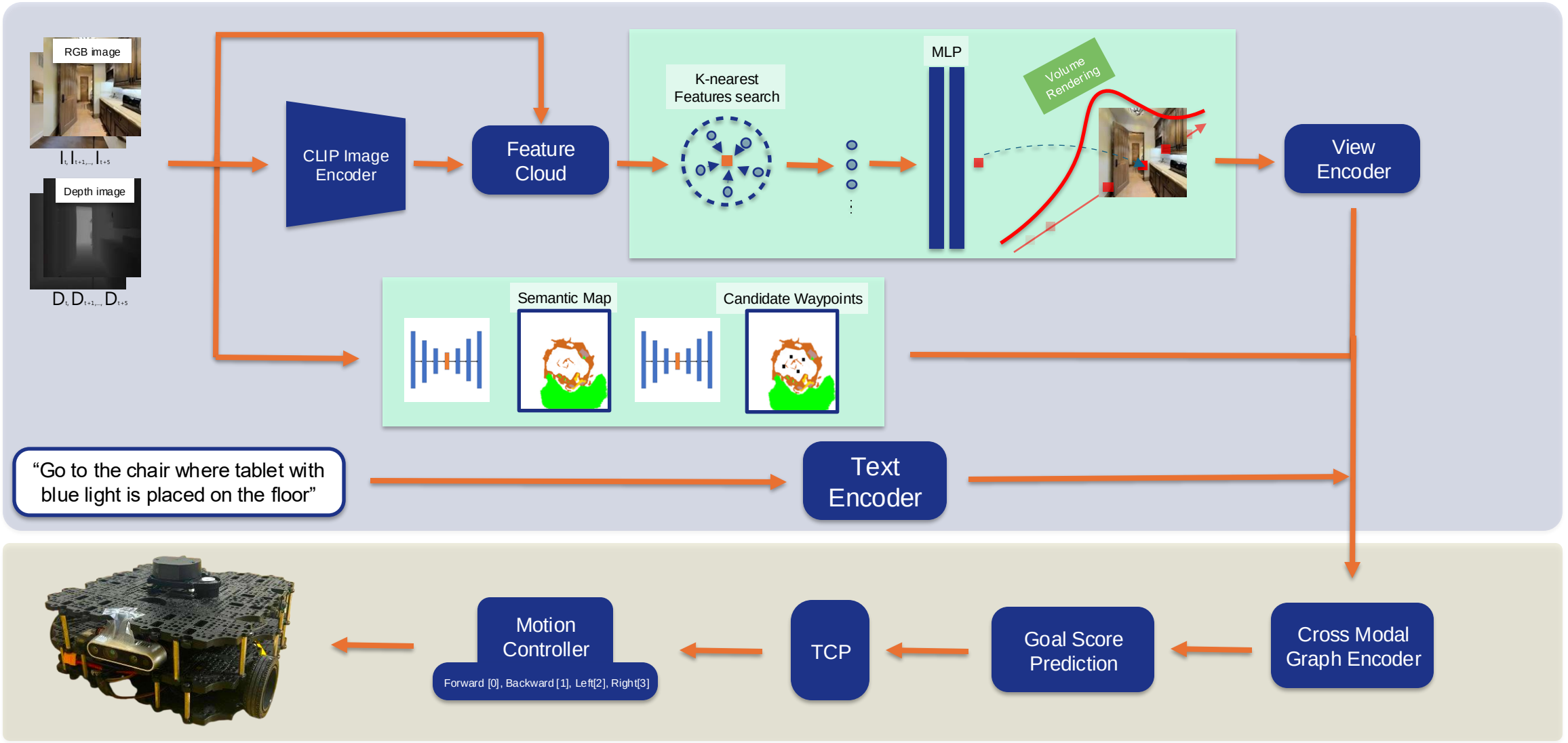
From Simulation to Reality: VLN on TurtleBot3

- **Simulation-Centric Evaluation**
  - Most VLN studies are conducted in synthetic, simulation-based environments, utilizing synthetic images.
  - Real-world evaluations are severely limited due to the absence of precise evaluation metrics.

- **Fundamental Issues with Simulation**
  - Simulation environments inherently differ from real-world conditions, presenting fundamental discrepancies.
  - Limitations of synthetic datasets:
    - Lack of environmental complexity.
    - Absence of sensor noise.
    - Simplified lighting conditions.
    - Failure to reflect real-world physical constraints.

- **Sensor Constraints**
  - Most robots rely on monocular cameras, which complicates the implementation of 360-degree panoramic cameras.
  - The limited field of view of monocular cameras leads to performance degradation in navigation tasks.

System Overview



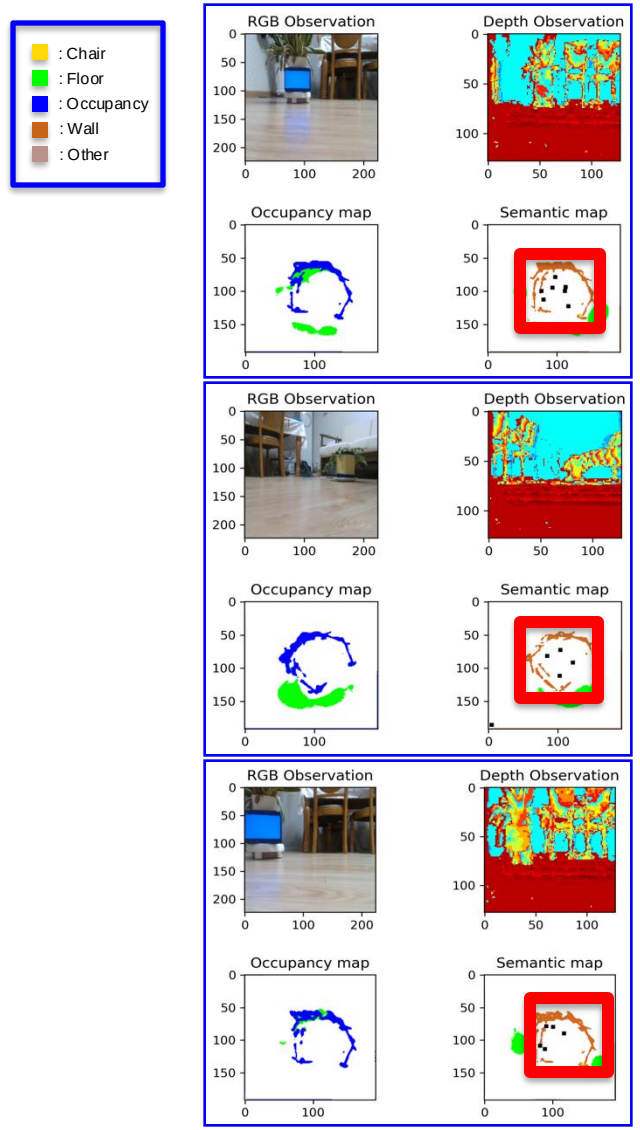
Experimental Result

- **Baseline Performance**
  - Success rate (SR): 43.3% (13/30)
  - The camera and LiDAR on the Locobot were mounted at the mid-thigh height of an adult male.
  - Viewpoints from the pre-trained VLNCE dataset clearly distinguished between the ceiling and the floor, aiding in navigation accuracy.

- **Proposed Model Performance**
  - Success rate (SR): 45% (18/40).
  - In experiments using the TurtleBot3 Waffle, the camera and LiDAR were mounted lower, closer to the ground.
  - The proposed model used past RGB-D data embeddings to improve the consistency of the traversable map.
  - Significant improvements in the consistency of the traversable map were visually confirmed during the experiments.

Result

- **Temporal Context Integration**
  - Use sequential RGB-D frames (past + current) instead of single-frame inputs.
  - Add Positional Encoding to each frame to capture temporal order effectively.
- **Feature Extraction Enhancement**
  - Process all frames through the HNR Model to generate a Temporal Feature Map that integrates spatial and temporal information.
- **Temporal Attention Mechanism**
  - Employ Attention Mechanism to dynamically weigh past frames based on their relevance to the current frame.
  - Ensure effective utilization of temporal dependencies.
- **Improvement Assessment**
  - Observe the consistency of the segment map during navigation
  - Verify that the map remains stable and accurate when the target point is temporarily out out view
  - Compare with the baseline model to ensure enhanced map continuity and better utilization of temporal dependencies



Experiment1

🎙️: Go to the chair near where the tablet emitting blue light is located.

» X10

Experiment2

🎙️: Go near the place where a tablet emitting blue light is placed near the plant.

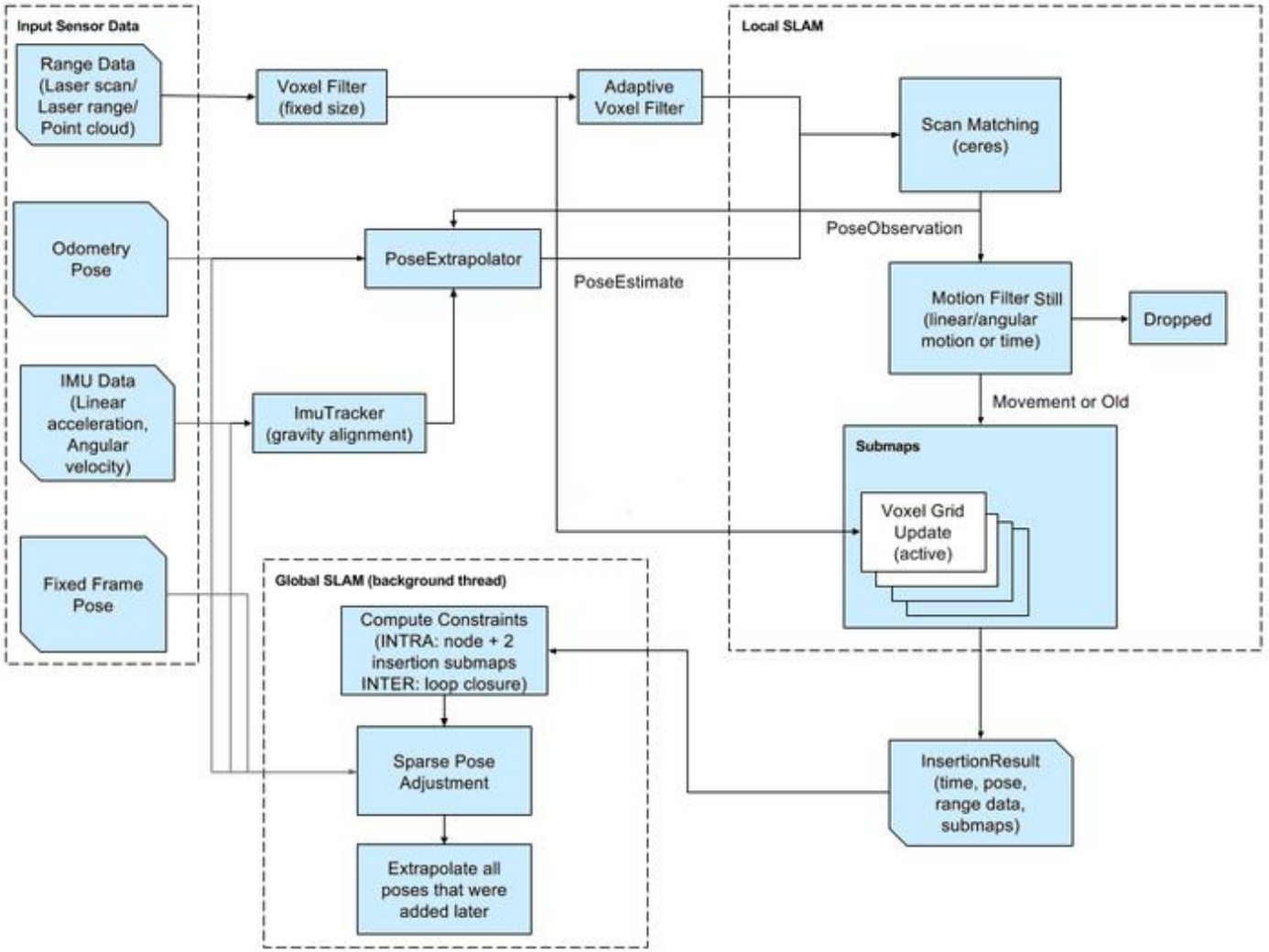
» X20

Thank you

Appendix:
Autonomous PickingBot using LIDAR

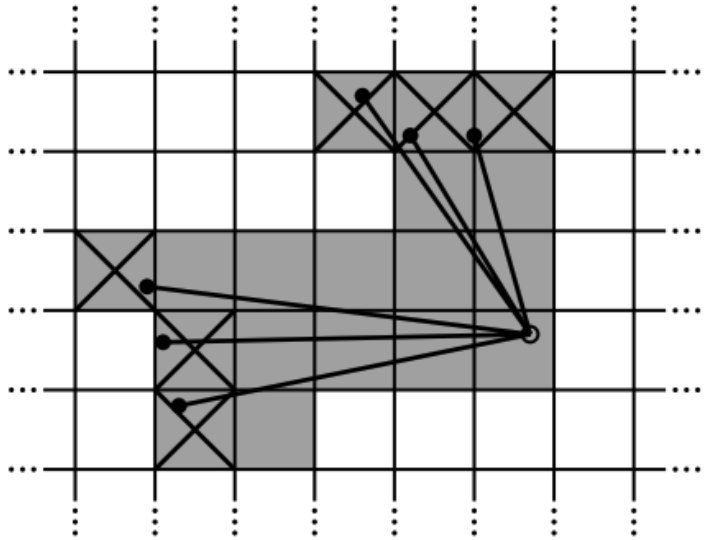
Autonomous PickingBot using LIDAR

- **Google Cartographer**



Autonomous PickingBot using LIDAR_Cartographer

- **Submaps in Probability Grids**
  - Submaps store their **range data** in a **probability grid.**
  - In 2D**, a signed distance field representation** is also possible.
- **Structure of Probability Grids**
  - The grid is a 2D table where each cell has a fixed size.
  - Each cell stores the ***odds*** of being obstructed.
- **Updating Odds (Hits & Misses)**
  - **"Hits"** correspond to measured **range data.**
  - **"Misses"** represent **the free space** between the sensor and measured points.
- **For every miss, we insert the grid point associated with each pixel that intersects one of the rays between the scan origin and each scan point, excluding grid points which are already in the hit set.**
- **If a grid point has yet to be observed it is assigned a value P_{hie} or P_{miss} depending on which set it is in.**



Define the submap as:

$$M : r\mathbb{Z} \times r\mathbb{Z} \rightarrow [p_{\min}, p_{\max}]$$

Then the map is updated according to the following operation:

$$\text{odds}(p) = \frac{p}{1-p}$$

$$M_{\text{new}}(x) = \text{clamp}\left(\text{odds}^{-1}\left(\text{odds}(M_{\text{old}}(x)) \cdot \text{odds}(p_{\text{hit}})\right)\right)$$

Autonomous PickingBot using LIDAR_Cartographer

$$\arg\min_{\xi} \sum_{k=1}^{K} \left(1 - M_{\text{smooth}}\left(T_{\xi} h_k\right)\right)^2$$

$\arg\min_{\xi}$: Find the pose in the local submap reference frame by minimizing the given function.

$\sum_{k=1}^{K}$: Sum over all scan points from $k = 1$ to K.

M_{smooth} : Bicubic interpolation of the probability grid.

T_{ξ} : Transform scan point from the scan frame to the local submap frame.

h_k : Coordinate of the scan return.

$\left(1 - M_{\text{smooth}}(T_{\xi} h_k)\right)^2$: Squared error representing the mismatch between scan points and the probability grid.

Autonomous PickingBot using LIDAR_Cartographer

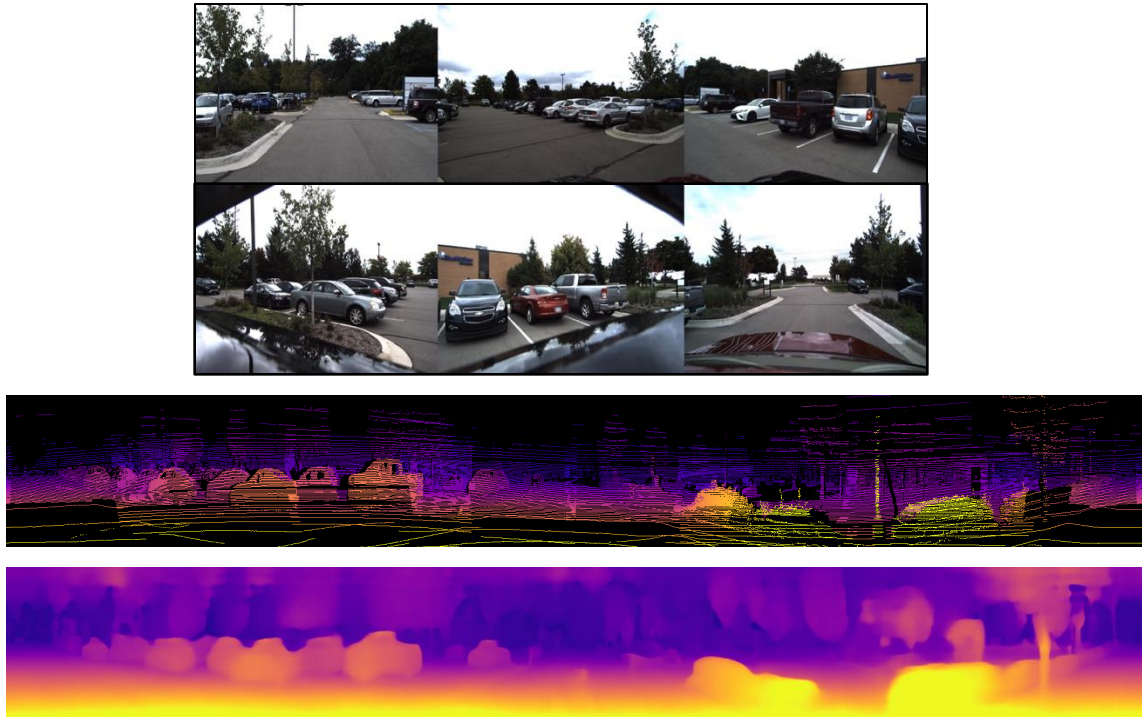
Algorithm 3 DFS branch and bound scan matcher for (BBS)

$best\_score \leftarrow score\_threshold$
Compute and memorize a score for each element in $\mathcal{C}_0$.
Initialize a stack $\mathcal{C}$ with $\mathcal{C}_0$ sorted by score, the maximum score at the top.
while $\mathcal{C}$ is not empty **do**
  Pop c from the stack $\mathcal{C}$.
  if $score(c) > best\_score$ **then**
    if c is a leaf node **then**
      $match \leftarrow \xi_c$
      $best\_score \leftarrow score(c)$
    else
      Branch: Split c into nodes $\mathcal{C}_c$.
      Compute and memorize a score for each element in $\mathcal{C}_c$.
      Push $\mathcal{C}_c$ onto the stack $\mathcal{C}$, sorted by score, the maximum score last.
    end if
  end if
end while
return $best\_score$ and $match$ when set.

- **Tree-structured Search Algorithm for Finding the Optimal Pose within a Given Submap**
  - **Node:** Candidate pose
  - **Height:** Higher resolution
  - Starts from the **low-resolution map** to infer the **pose** and gradually finds the **pose** in the **high-resolution map**.
  - Based on **DFS (Depth-First Search)** and **matching scores**

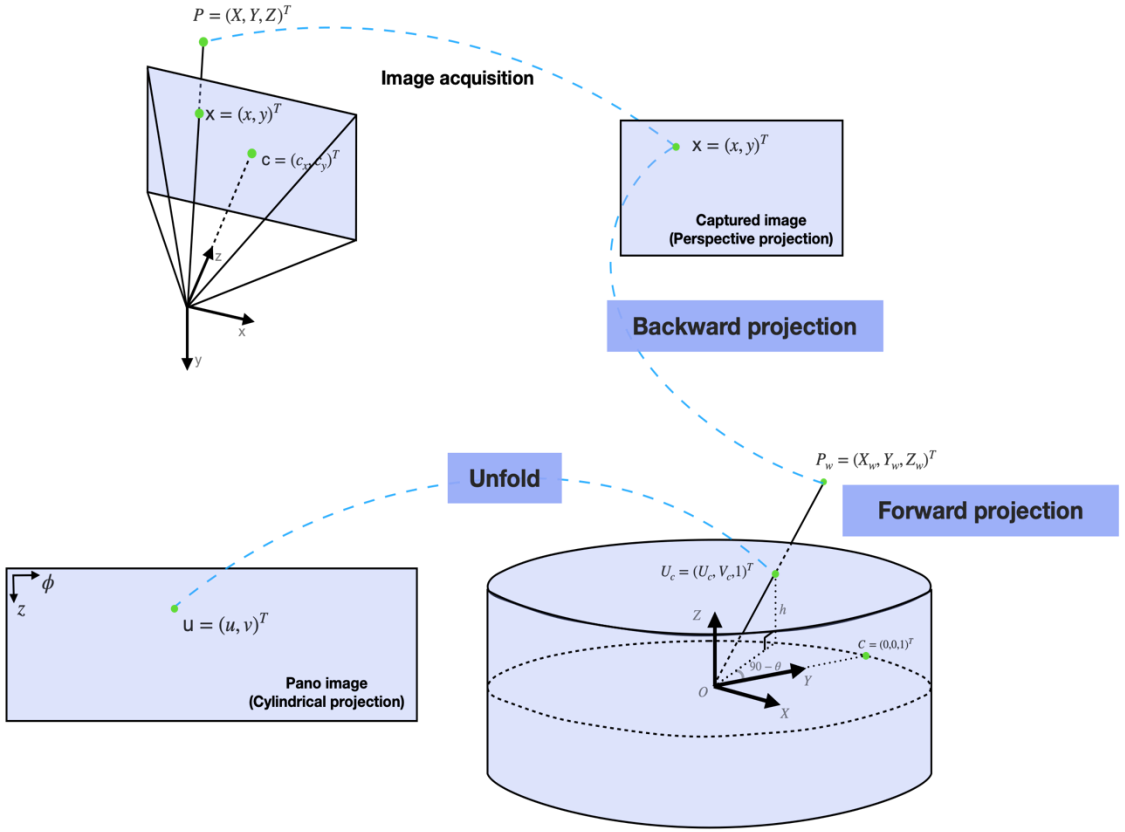
$$
\begin{aligned}
score(c) &= \sum_{k=1}^{K} \max_{j \in \overline{\overline{\mathcal{W}}}_c} M_{\text{nearest}}(T_{\xi_j} h_k) \\
&\geq \sum_{k=1}^{K} \max_{j \in \overline{\mathcal{W}}_c} M_{\text{nearest}}(T_{\xi_j} h_k) \\
&\geq \max_{j \in \overline{\mathcal{W}}_c} \sum_{k=1}^{K} M_{\text{nearest}}(T_{\xi_j} h_k).
\end{aligned}
$$

Appendix:
Variance-Integrated Panodepth for Cylindrical Depth Estimation



Variance-Integrated Panodepth for Cylindrical Depth Estimation

Backward projection & Forward projection



$$P_c \cong Z_c \cdot K^{-1} = Z_c \cdot (\frac{x - c_x}{f}, \frac{y - c_y}{f}, 1) = (X_c, Y_c, Z_c)^T, \quad P_w = R \cdot P_c + t$$

$$\theta_{cylinder} = arctan(\frac{Y_w}{X_w}), \quad z_{cylinder} = \frac{z_w}{\sqrt{x^2 + y^2}} \times \rho_r = Z_w$$

$$\phi = \frac{W_c(\theta - (\pi - \phi_{min}))}{\theta_{max} - \theta_{min}}, \quad z = \frac{H_c(z_{min} - z)}{z_{max} - z_{min}} \quad \theta_{min} = 0, \ \theta_{max} = 2\pi$$

Cylindrical Sweeping Method – *Process of converting images obtained from each camera into a cylindrical coordinate system*

Variance-Integrated Panodepth for Cylindrical Depth Estimation

Volume Variance Calibration & Confidence Estimation

$$variance = \frac{1}{N}\sum_{i=1}^{N} f_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} f\_i\right)^2$$

$$w = sigmoid\left(w_{net}(x)\right) \cdot \exp(-variance)$$

f_i: $Feature\ from\ each\ camera\ view$

N: $Number\ of\ depth\ hypothesis$

$Variance - based\ weighting;\ lower\ variance, higher\ weight$

Huber Loss

$$\mathrm{H}uberLoss = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(y_i - \hat{y}_i)^2, & if\ |y_i - \hat{y}_i| \le \delta \\ \frac{1}{n}\sum_{i=1}^{n} \delta(|y_i - \hat{y}_i| - \frac{\delta}{2}), & otherwise \end{cases}$$

Appendix:
From Simulation to Reality: VLN on TurtleBot3

Hadware configuration

- **TurtleBot3 Waffle Platform**
  - Equipped with Realsense D435i for depth sensing and RGB input

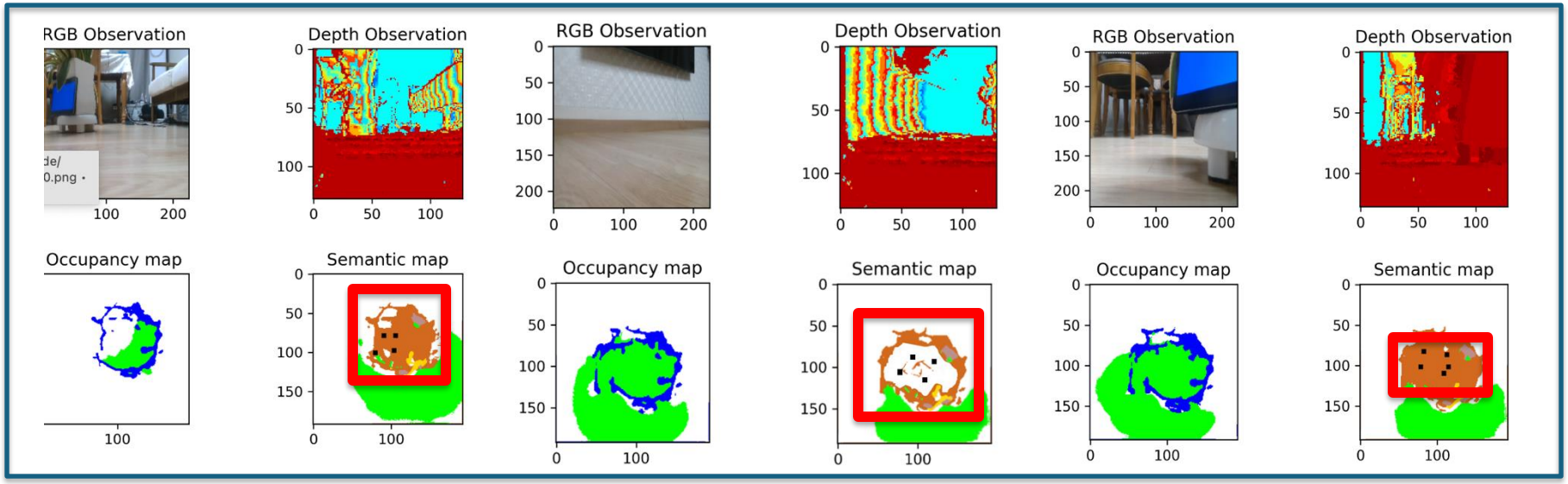
- **Realsense D435i**
  - Stereo depth camera with FOV: 87° x 58°.
  - Integrated RGB camera with FOV: 69° x 42°.
  - Sensing range: 0.1~10m.
  - Operates at ~30Hz.

- **Server PC: Intel NUC**
  - Compact mini-PC with Intel Core i7 processor.

- **Battery**
  - 3-cell battery powering the TurtleBot3 platform

Result

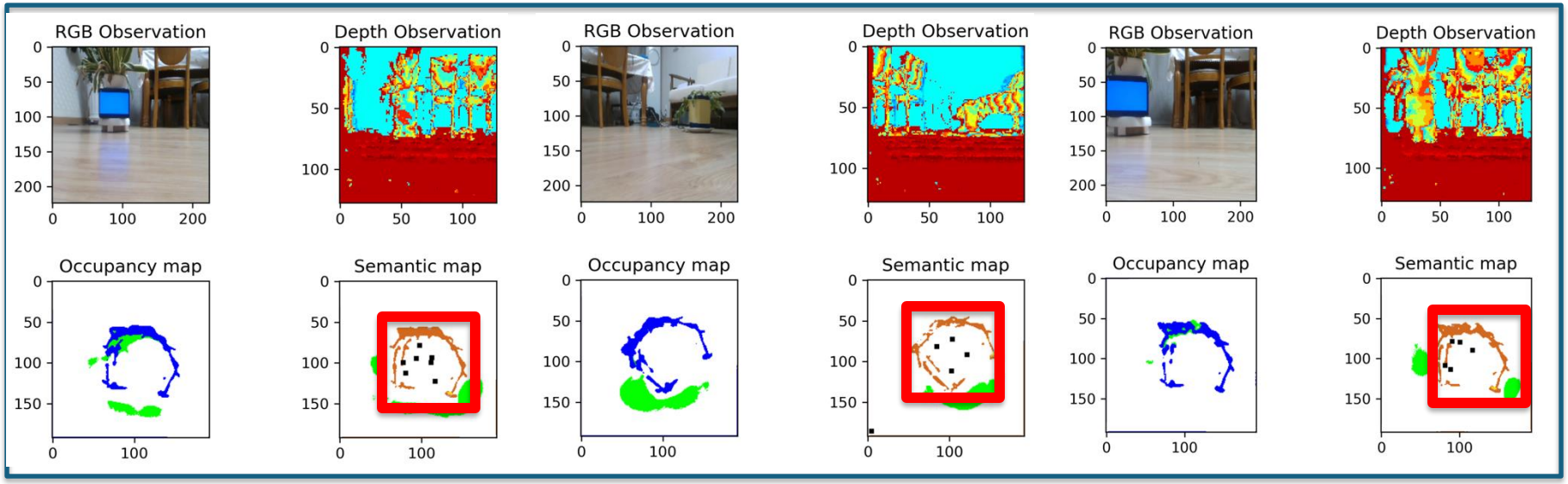
Base model



Base model

+

Temporal RGB-D Datas



Next Steps

- **Semantic segmentation Enhancement**
  - Enhance semantic segmentation by filling unseen areas in free space regions at the voxel level using diffusion models.
- **Dataset Alignment through Experimental Adjustments**
  - Build a taller structure to elevate cameras and LiDAR, ensuring that data collection aligns with the perspective and quality of the pre-trained dataset.
- **Enhancing Traversable Map Robustness Using Mamba**
  - Use Mamba, optimized for long-sequence processing, to enhance the robustness and consistency of traversable maps along extended navigation paths.
- **Addressing Simulation-to-Real Performance Drop**
  - Investigate and develop strategies to mitigate the performance drop when transitioning from simulated to real-world environments.