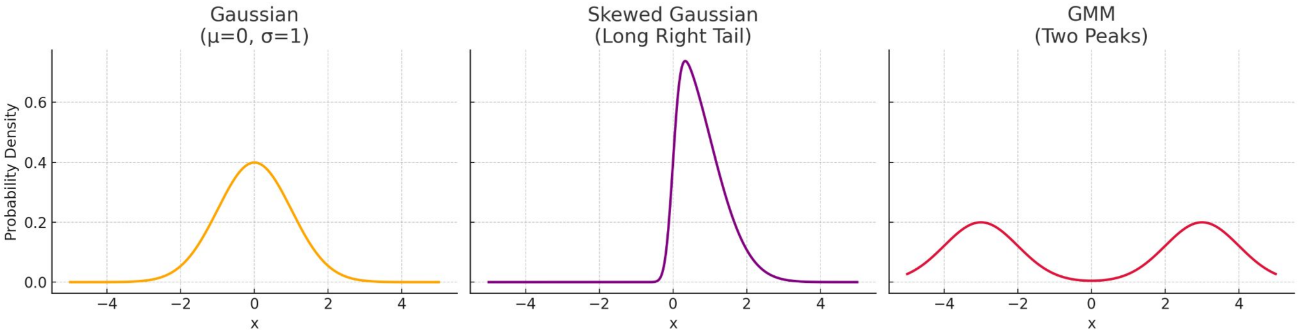
MGF Localization & Elevation Mapping

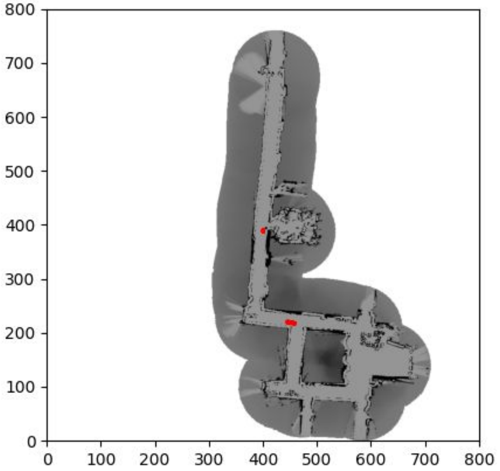
Ricky Yuan, Boxiang (William) Fu, Joshua Pen, and Tianzhi Li

Robot Localization and Mapping
Carnegie Mellon University

MGF Localization: Motivation

- **Gaussian State Assumption:**
  - MGF allows more complicated state distributions
- **Calculation Overhead:**
  - EKF / KF: Avoid calculating matrix inverses
  - Particle Filter: Avoid candidates with sub-optimal likelihood





MGF Localization: Methodology

Definition. (*Moment Generating Function*): $M_{\mathbf{X}}(\mathbf{s}) = \mathbb{E}[e^{\mathbf{s}^T\mathbf{X}}] = \int_{-\infty}^{\infty} e^{\mathbf{s}^T\mathbf{x}} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}$

Definition. (*n'th Order Moment*): $\mathbb{E}[\mathbf{X}^n] = \int_{-\infty}^{\infty} \mathbf{x}^n f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}$

Heuristic. (*Information Equivalency and Uniqueness*):

$$\{PDF\ of\ X\} = \{MGF\ of\ X\} = \{All\ Moments\ of\ X\}$$

Theorem. (*Evaluation Property of MGFs*): $\mathbb{E}[X_1^{\alpha_1} X_2^{\alpha_2} \cdots X_n^{\alpha_n}] = \frac{\partial^{(\alpha_1 + \cdots + \alpha_n)}}{\partial {s_1}^{\alpha_1} \cdots \partial {s_n}^{\alpha_n}} M_{\mathbf{X}} \Big|_{\mathbf{s}=0}$

Theorem. (*Linearity of MGFs*): If **X** and **Y** are independent random variables, then

$$M_{\mathbf{X}+\mathbf{Y}}(\mathbf{s}) = M_{\mathbf{X}}(\mathbf{s}) M_{\mathbf{Y}}(\mathbf{s})$$

MGF Localization: State Propagation

Motion Model: $\tilde{\mathbf{X}}_t = A\mathbf{X}_{t-1} + B\mathbf{u}_{t-1}$

Sensor Model: Permits any arbitrary continuous probability density function

Fusion Model: $\mathbf{X}_t = \alpha\mathbf{Z}_{t-1} + (1-\alpha)\tilde{\mathbf{X}}_t$

Where **X** is the state, **u** is the stochastic control input, **Z** is the sensor measurement, and **α** is a hyper-parameter that can be tuned

State Propagation (*Functional Representation*):

$$
\begin{aligned}
M_{\mathbf{X}_t}(\mathbf{s}) &= M_{\mathbf{Z}_{t-1}}(\alpha\mathbf{s}) M_{\tilde{\mathbf{X}}_t}((1-\alpha)A\mathbf{s}) \\
&= \underbrace{M_{\mathbf{u}_{t-1}}((1-\alpha)B\mathbf{s})}_{f} \underbrace{M_{\mathbf{Z}_{t-1}}(\alpha\mathbf{s})}_{g} \underbrace{M_{\mathbf{X}_{t-1}}((1-\alpha)A\mathbf{s})}_{h}
\end{aligned}
$$

MGF Localization: State Propagation

State Propagation (*Moment Representation for Scalar State **X***):

$$
\begin{aligned}
\mathbb{E}[\mathbf{X}_t^n] &= M_{\mathbf{X}_t}^{(n)}\Big|_{\mathbf{s}=0} \\
&= \frac{\partial^n}{\partial s^n}[f(s)g(s)h(s)]\Big|_{s=0} \\
&= \sum_{i=0}^{n}\sum_{j=0}^{n-i}\frac{n!}{i!j!(n-i-j)!}[f^{(i)}(s)g^{(i)}(s)h^{(i)}(s)]\Big|_{s=0} \\
&= \sum_{i=0}^{n}\sum_{j=0}^{n-i}\frac{n!}{i!j!(n-i-j)!}\times[(1-\alpha)B]^i\mathbb{E}[\mathbf{u}_{t-1}^i] \\
&\times \alpha^j\mathbb{E}[\mathbf{Z}_{t-1}^j]\times[(1-\alpha)A]^{n-i-j}\mathbb{E}[\mathbf{X}_{t-1}^{n-i-j}]
\end{aligned}
$$

MGF Localization: State Propagation

State Propagation (*Moment Representation for Vector State* ***X***):

$$
\begin{aligned}
\mathbb{E}[\mathbf{X}_t^{\otimes n}] &= D^n(M_{\mathbf{X}_t})\Big|_{\mathbf{s}=0} \\
&= D^n(f(\mathbf{s})g(\mathbf{s})h(\mathbf{s}))\Big|_{\mathbf{s}=0} \\
&= \sum_{(i_1,\ldots,i_n)\in\{f,g,h\}^n} \left(A_{i_1}\otimes\cdots\otimes A_{i_n}\right) \underbrace{\mathbb{E}\left[V_{i_1}\otimes\cdots\otimes V_{i_n}\right]}_{\text{factorizes if independent}}
\end{aligned}
$$

where we define

$$
\begin{aligned}
\{A_f, V_f\} &= \{(1-\alpha)B, \mathbf{u}_{t-1}\} \\
\{A_g, V_g\} &= \{\alpha, \mathbf{Z}_{t-1}\} \\
\{A_h, V_h\} &= \{(1-\alpha)A, \mathbf{X}_{t-1}\}
\end{aligned}
$$

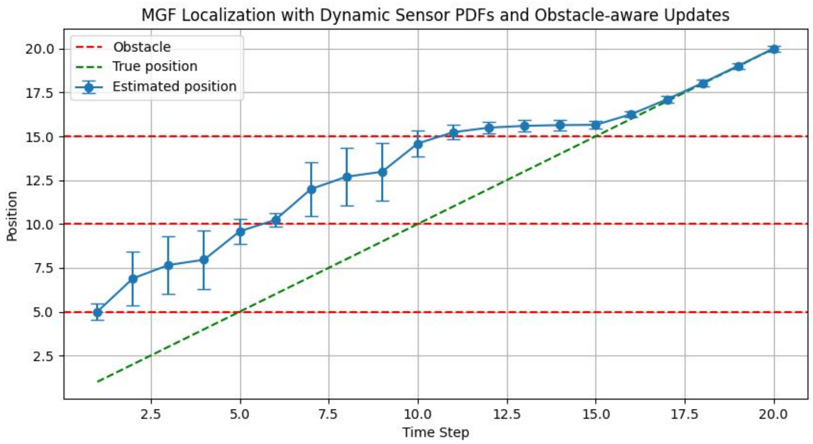
MGF Localization: Algorithm

Algorithm 1 MGF-Localization

Require: $N \in \mathbb{N}_{>0}$ ▷ Maximum order of moments
Require: $T \in \mathbb{N}_{>0}$ ▷ Termination timestep
Require: $\alpha \in \mathbb{R}_{[0,1]}$ ▷ Fusion hyper-parameter
Require: $A, B \in \mathbb{R}^{N \times N}$ ▷ Motion model matrix

$t = 1$
$\mathcal{X} = \{\mathbb{E}[\mathbf{X}_0], \cdots, \mathbb{E}[\mathbf{X}_0^{\otimes N}]\}$ ▷ Initial state moments
$\psi = \{\mathbb{E}[\mathbf{Z}_0], \cdots, \mathbb{E}[\mathbf{Z}_0^{\otimes N}]\}$ ▷ Initial sensor moments
$\Omega = \{\mathbb{E}[\mathbf{u}_0], \cdots, \mathbb{E}[\mathbf{u}_0^{\otimes N}]\}$ ▷ Initial control moments
while $t \leq T$ **do**
  $n = 1$
  $\mathcal{X}_t = \emptyset$
  while $n \leq N$ **do**
    $\mathbb{E}[\mathbf{X}_t^{\otimes n}] = \sum (A_{i_1} \otimes \cdots \otimes A_{i_n}) \mathbb{E}[V_{i_1} \otimes \cdots \otimes V_{i_n}]$
    summation over $(i_1, \ldots, i_n) \in \{f, g, h\}^n$,
    with $\{A_f, V_f\} = \{(1-\alpha)B, \mathbf{u}_{t-1}\}$,
    and $\{A_g, V_g\} = \{\alpha, \mathbf{Z}_{t-1}\}$,
    and $\{A_h, V_h\} = \{(1-\alpha)A, \mathbf{X}_{t-1}\}$
    $\mathcal{X}_t \leftarrow \mathbb{E}[\mathbf{X}_t^{\otimes n}]$
  end while
  $\mathcal{X} = \mathcal{X}_t$
  $\psi = \{\mathbb{E}[\mathbf{Z}_t], \cdots, \mathbb{E}[\mathbf{Z}_t^{\otimes N}]\}$ ▷ New sensor readings
  $\Omega = \{\mathbb{E}[\mathbf{u}_t], \cdots, \mathbb{E}[\mathbf{u}_t^{\otimes N}]\}$ ▷ New control commands
  $t = t + 1$
end while

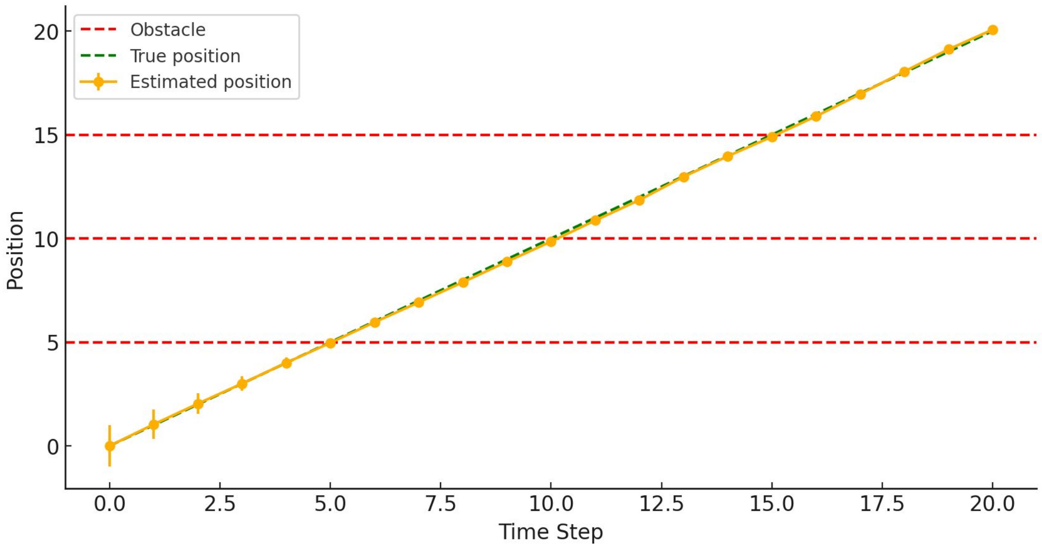
MGF Localization: Functional Representation



- ## 1D Localization Case
  - **High initial uncertainty:** Early position estimates have large standard deviations.
  - **Improved accuracy over time:** Standard deviation decreases with more measurements.
  - **Effective localization:** Estimated positions converge to true positions.
- ## 2D Localization Case
  - **Computational challenges:** Symbolic integration in Python leads to memory issues.
  - **Scalability limitations:** Current implementation unsuitable for real-world deployment.

MGF Localization: Moment Representation

- **Result**
  - Large initial uncertainty: Early position has large variance, but reduced over time
  - Accurate trajectory estimation
- **Challenges**
  - Real-time update of higher-order moments becomes non-trivial in 2D
- **Takeaways**
  - The MGF methods is useful when dealing simple distribution



MGF Localization: Future Work

- Replacing symbolic integration with **numerical or approximate techniques** to reduce memory consumption.
- Investigating **stronger non-Gaussian motion models** to incorporate real-world robot motion.
- **Benchmarking** against standard SLAM algorithms in both simulated and physical environments.

Elevation Mapping: Motivation

With some spare time and determination to submit a successful project, we pivoted to implementing another project

We **extended** the occupancy grid mapping algorithm (Moravec and Elfes, 1987) from a binary state (occupied/free) to a **continuous state (elevation of grid)**

Additionally, we also do not assume the pose is given to us. Instead, we **estimate the camera pose using visual tracking**

This extends the algorithm from only mapping to **simultaneous localization and mapping**

Finally, we implemented the algorithm in ROS so that it **works in real time**

Elevation Mapping: Localization Methodology

- **Sensor:** ZED 2i
- **Stereo Visual Odometry:**
    - Captures synchronized left/right images, detects & matches stable features, then solves a 6 DoF transform (PnP or reprojection-error minimization) at up to 100 Hz.
- **Inertial Fusion (VIO):**
    - Reads IMU accel/gyro at ~400 Hz and tightly fuses those deltas with the visual odometry via a filter (e.g. EKF or sliding window).
- **Pose Output:**
    - Emits a timestamped pose containing [X,Y,Z] + quaternion [x,y,z,w] in the world frame, along with a confidence score.

https://www.stereolabs.com/docs/positional-tracking

Elevation Mapping: Bayes Filter Methodology

Uses a **Kalman Filter** in one dimension

For each timestep, loop over all map cells

For each map cell, update:

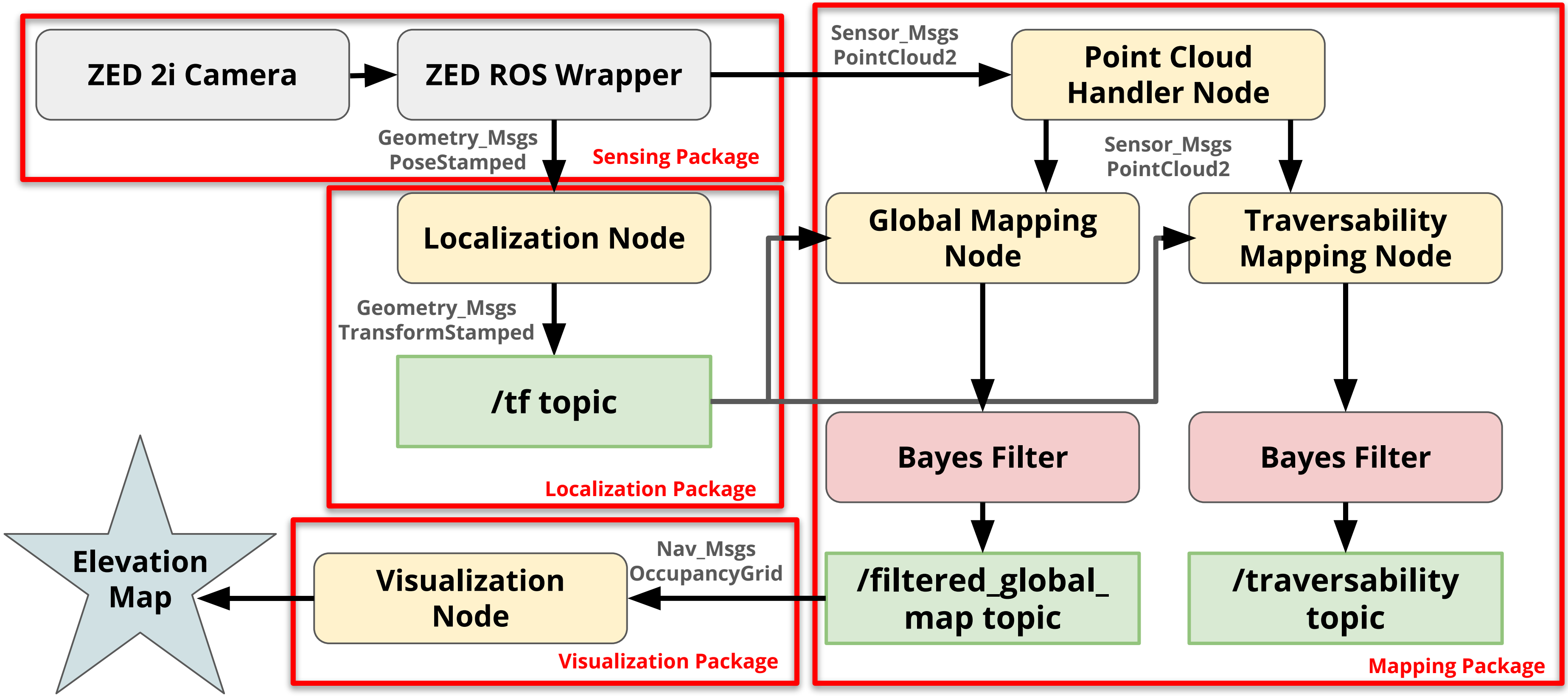
Existing Data
Height: $\hat{z}$, Variance: $\hat{\sigma}$

Incoming Data
Height: z, Variance: σ

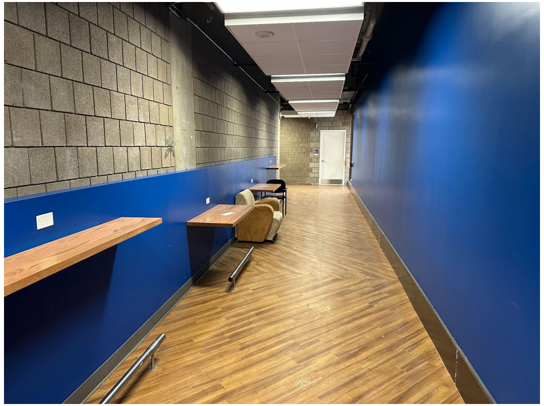
$K = \hat{\sigma}/(\hat{\sigma}+\sigma)$

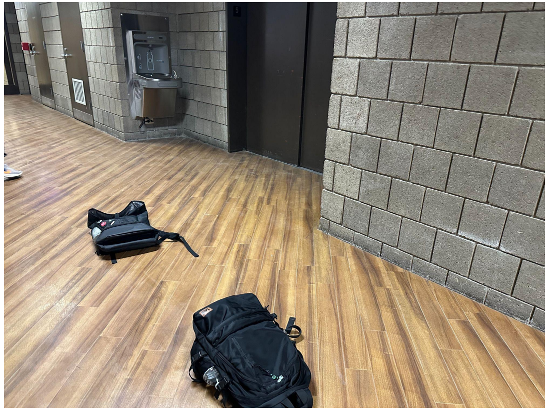
New Data
Height: $\hat{z} + K(z - \hat{z})$, Variance: $(1 - K)\hat{\sigma}$

Elevation Mapping: Fusion Methodology



Elevation Mapping: Experiment

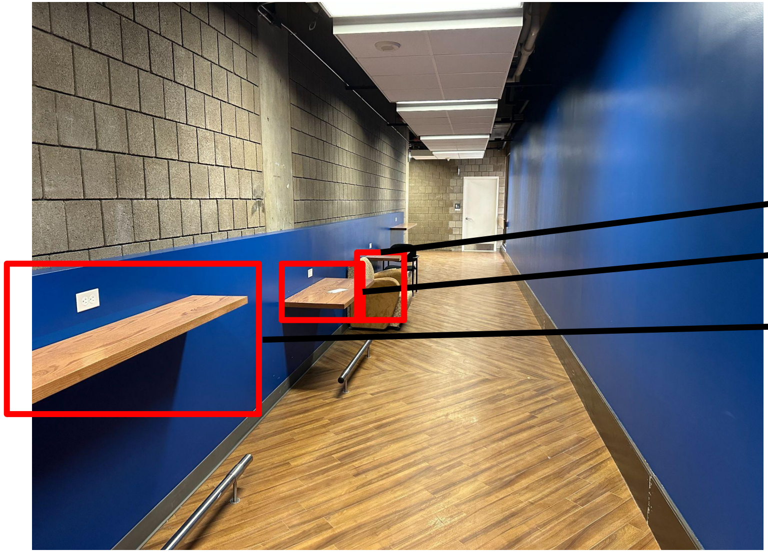


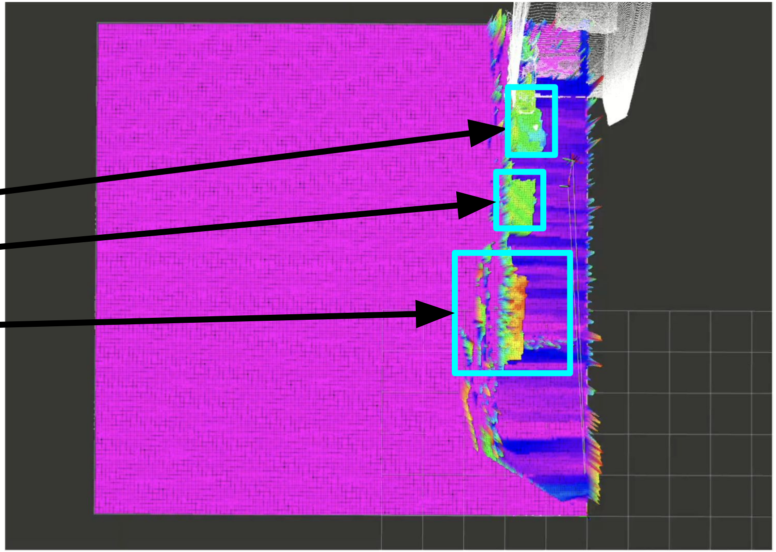


- **Purpose:** Demonstrate elevation mapping capabilities.
- **Experiment Setup:**
  - **Origin/Starting Location:** Bottom Right Corner of Map
  - **Location:** Wean Hall 5th Floor
  - **Environment 1:** Straight hallway with tables and chairs on the left side.
  - **Environment 2:** Hallway with a left turn with two bags on the floor and a elevator on the right.

Elevation Mapping: Results

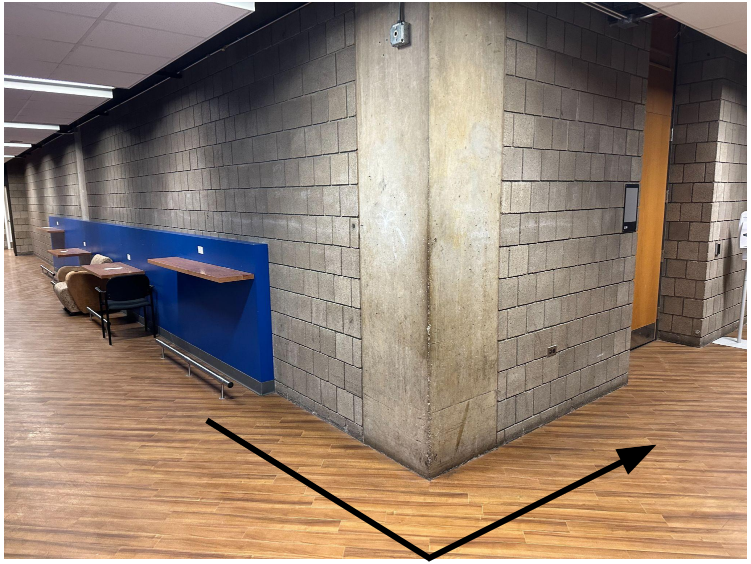
Environment 1: Straight hallway

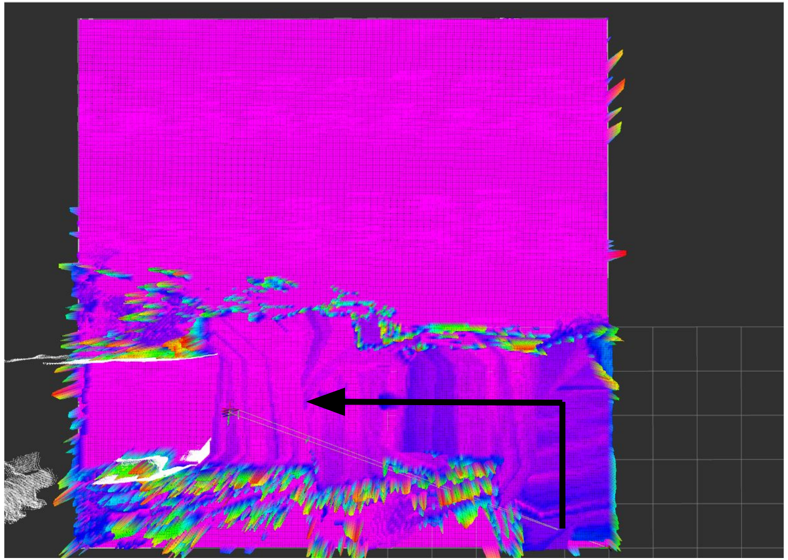




Elevation Mapping: Results

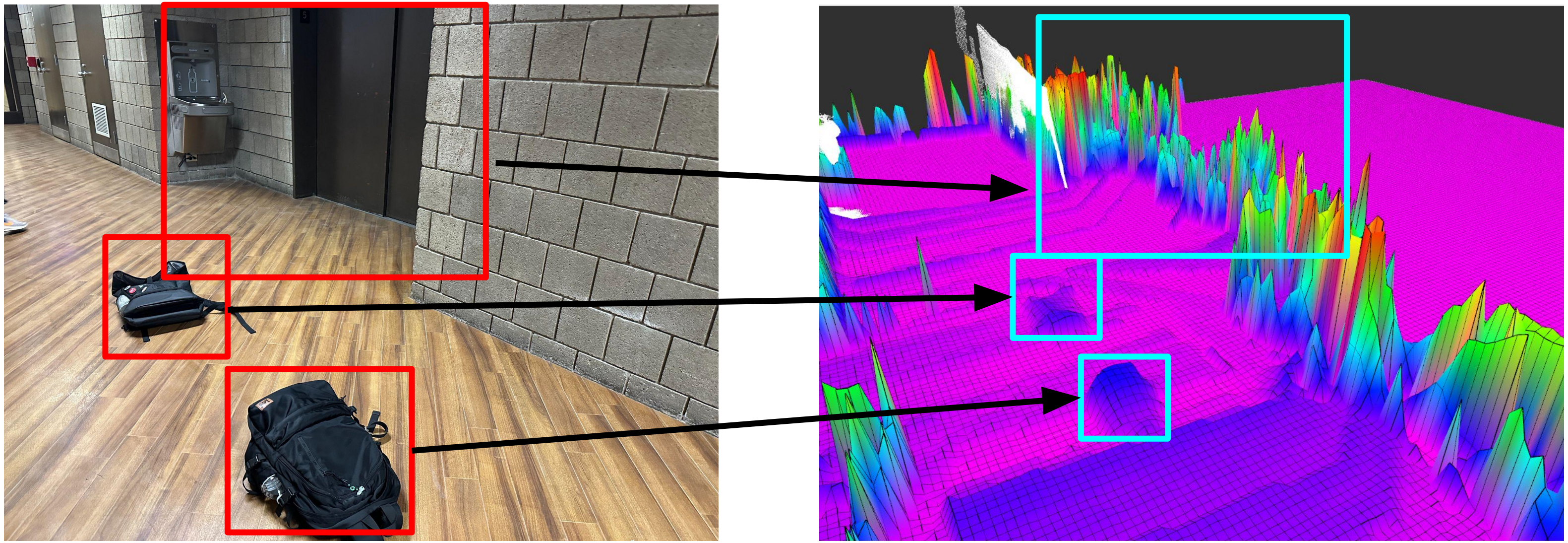
Environment 2: Hallway with a left turn





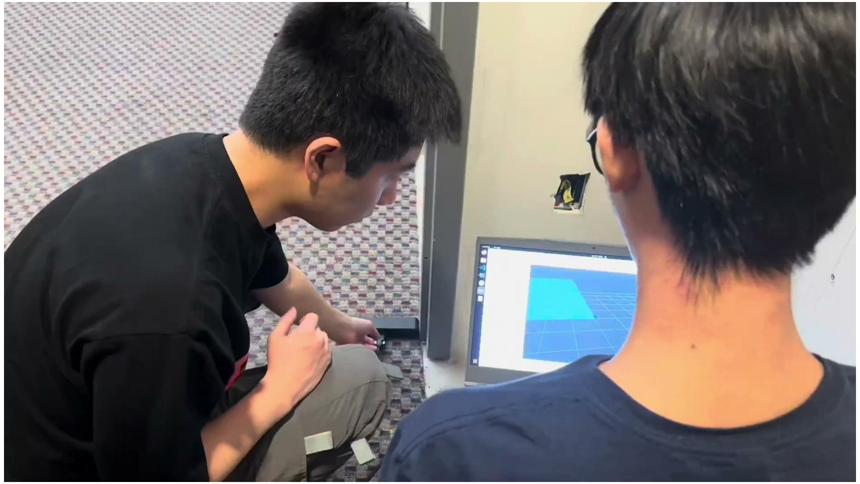
Elevation Mapping: Results

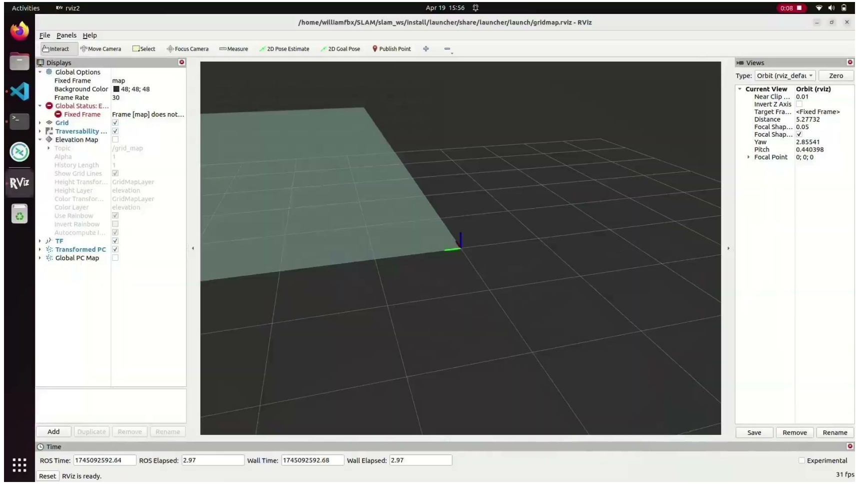
Environment 2: Hallway with a left turn



Elevation Mapping: Results

We also tried NSH 1305





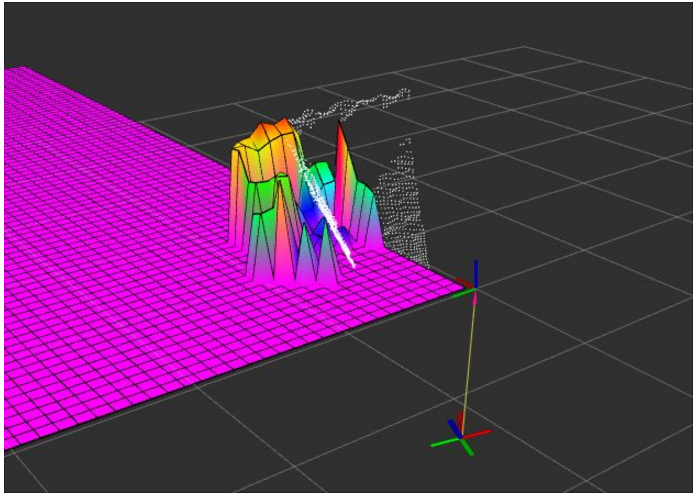
Elevation Mapping: Takeaways

Performed well in **structured, open environment** (e.g. hallways), where visual tracking and elevation mapping remains stable.

Struggled in **cluttered environments** (e.g. classrooms), due to occlusions and noisy measurements.

Elevation Mapping: Roadblocks (D435 + ICP)

- **Data Quality Issues**
  - Point Cloud of RealSense D435 is not as dense as ZED 2i
- **Overly Simple Scene Geometry**
  - Over-smooth surfaces cause drifts
- **Algorithmic Implementation Challenges**
  - Misaligned frames messed up localization
- **Parameter Tuning Limitations**
  - Might not converge or get stuck in local minima



Elevation Mapping: Future Work

- **Improve robustness in cluttered environments** by incorporating semantic segmentation or filtering unstable features during pose estimation
- **Extend elevation map to full 3D reconstruction** for handling overhangs and complex structures
- **Integrate dynamic obstacle detection** to enable use in real-world navigation tasks

Thank You!

MGF Localization: https://github.com/CMU-SLAM25/MGF-Localization

Elevation Mapping: https://github.com/CMU-SLAM25/Elevation-Mapping