MGF-Localization: State Estimation and Propagation using Moment Generating Functions

Boxiang Fu[1†], Joshua Pen[2†], Ricky Yuan[3†], and Tianzhi Li[4†]

***Abstract*— The *de facto* method for nonlinear state estimation has long been the Extended Kalman Filter. However, one significant drawback of the method is that the underlying state distribution is represented using a Gaussian distribution. Such unimodal distributions fail to estimate the underlying state when there are symmetries in the map geometry. In this paper, we introduce state localization using a moment generating function to represent the underlying state. Such a state representation does not require the assumption of any specific underlying distribution function. As long as the distribution function is continuous over state space, the state evolves when new motion and sensor information is obtained.**

I. INTRODUCTION

Accurate state estimation is a fundamental task in robotics and autonomous systems, particularly for localization. The Extended Kalman Filter (EKF) has long been the standard approach to nonlinear state estimation due to its simplicity and computational efficiency. EKF represents the uncertainty of the state using a single Gaussian distribution, maintaining only one peak in its belief. However, this single-peak assumption limits the EKF's performance in environments with symmetric or ambiguous map structures, where multiple robot poses may equally well explain the sensor measurements.

The particle filter algorithm, however, represents the belief distribution with a set of weighted samples (particles). Particle filters naturally handle multiple possible positions simultaneously and avoid converging too quickly to a single position until more sensor information resolves ambiguity. Despite their advantages, particle filters are computationally intensive, particularly in higher-dimensional state spaces, and can suffer from particle depletion, where important hypotheses are lost due to insufficient particle coverage.

In this work, we propose MGF-Localization, a new approach to state estimation using Moment Generating Functions (MGFs). Unlike EKF, our approach does not assume any specific probability distribution form. By tracking a set of moments (such as means and variances), MGF-Localization can represent multiple possible robot positions without relying on particles. This approach provides a balance between accurately representing uncertainty and maintaining reasonable computational efficiency.

[1]boxiangf@cs.cmu.edu
[2]jpen@cs.cmu.edu
[3]rickyy@andrew.cmu.edu
[4]tianzhil@cs.cmu.edu
[†]Robotics Institute, Carnegie Mellon University

This project presents the mathematical foundation of MGF-based localization, derives moment update rules for both scalar and vector states, and outlines an implementable algorithm. The MGF framework opens new possibilities for robust localization under complex, non-Gaussian scenarios.

II. RELATED WORK

Nonlinear state estimation is central to robot localization. The Extended Kalman Filter (EKF) [7] has long been the *de facto* standard due to its efficiency, but its assumption of a unimodal Gaussian belief limits its performance in environments with perceptual aliasing or geometric symmetries.

To address this, the Unscented Kalman Filter (UKF) [8] improves nonlinear handling but retains the unimodal constraint. Particle filters [4, 5] allow multi-modal belief representations and are widely used in Monte Carlo Localization (MCL), though they can be computationally expensive and degrade in high-dimensional spaces.

Several approaches seek richer belief representations for localization. Gaussian Mixture Models (GMMs) [12] and nonparametric belief propagation [11] allow for multi-modality, but often require additional mechanisms for component selection or pruning. Other techniques leverage kernel methods [13] or Fourier and Hilbert-space embeddings [10] to implicitly model distributions, offering strong theoretical guarantees but often requiring careful tuning and substantial computation.

More recently, non-Gaussian belief representations have been proposed to better capture the complexities of robot localization. For instance, normalizing flows have been employed to learn expressive belief distributions [3], and neural implicit representations have been used to encode spatial uncertainty [9]. However, such methods typically rely on learned components and may require extensive training data and offline computation.

In contrast, our method models the state using a moment generating function (MGF), which offers a compact and expressive continuous representation of the belief without assuming a specific distributional form. By propagating moments through time and incorporating measurement updates directly in the MGF space, our approach naturally accommodates symmetric and multi-modal posteriors, making it especially well-suited for ambiguous state estimation tasks. To our knowledge, this is the first application of MGFs for probabilistic state localization in robotics.

III. MATHEMATICAL THEORY

A. Moment Generating Functions

Let $\mathbf{X} \in \mathbb{R}^n$ be the random variable denoting the state estimate. We define the following objects:

Definition. The moment generating function (MGF) of a continuous real-valued random variable $\mathbf{X}$ is

$$M_{\mathbf{X}}(\mathbf{s}) = \mathbb{E}[e^{\mathbf{s}^T\mathbf{X}}] = \int_{-\infty}^{\infty} e^{\mathbf{s}^T\mathbf{x}} f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}$$

where $f_{\mathbf{X}}(\mathbf{x})$ is the probability density function of $\mathbf{X}$. The MGF is defined everywhere where the expectation is finite.

$\mathbf{s}$ is defined as an auxiliary variable and has the same dimensionality as $\mathbf{X}$.

Definition. The *n*'th order moment of a continuous real-valued random variable $\mathbf{X}$ is

$$\mathbb{E}[\mathbf{X}^n] = \int_{-\infty}^{\infty} \mathbf{x}^n f_{\mathbf{X}}(\mathbf{x}) d\mathbf{x}$$

where $f_{\mathbf{X}}(\mathbf{x})$ is the probability density function of $\mathbf{X}$ and is defined everywhere where the expectation is finite.

Despite the different mathematical objects, the underlying information represented by the different manifestations are the same – they all represent the random variable $\mathbf{X}$. Heuristically, the following information are equivalent and unique:

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

Their uniqueness is guaranteed by the uniqueness theorem [6]:

Theorem. *(Uniqueness theorem of MGFs)*
The distribution of $\mathbf{X}$ is determined uniquely by the function $M_{\mathbf{X}}(\mathbf{s})$. That is, if $\mathbf{Y}$ is any random vector whose MGF is the same as $M_{\mathbf{X}}(\mathbf{s})$, then $\mathbf{Y}$ has the same distribution as $\mathbf{X}$.

A very useful property of MGFs is that the function is a generator for the moments of $\mathbf{X}$. In other words, the derivatives of $M_{\mathbf{X}}(\mathbf{s})$ evaluated at $\mathbf{s} = 0$ gives the higher-order moments of $\mathbf{X}$. For an univariate continuous state variable $\mathbf{X}$, the following is true:

Lemma. *(Evaluation property of univariate MGFs)*
Let $M_{\mathbf{X}}(\mathbf{s}) = \mathbb{E}[e^{\mathbf{s}\mathbf{X}}]$ be the MGF of a continuous random variable $\mathbf{X} \in \mathbb{R}$. If $M_{\mathbf{X}}(\mathbf{s})$ is finite in an epsilon-neighborhood about $\mathbf{s} = 0$, then

$$\mathbb{E}[\mathbf{X}^k] = M_{\mathbf{X}}^{(k)}\Big|_{\mathbf{s}=0} \tag{1}$$

For multivariate continuous state variables, the result is similar:

Theorem. *(Evaluation property of multivariate MGFs)*
Let $M_{\mathbf{X}}(\mathbf{s}) = \mathbb{E}[e^{\mathbf{s}^T\mathbf{X}}]$ be the MGF of a continuous random variable $\mathbf{X} \in \mathbb{R}^n$. If $M_{\mathbf{X}}(\mathbf{s})$ is finite in an epsilon-neighborhood about $\mathbf{s} = 0$, then

$$\mathbb{E}[X_1^{\alpha_1} X_2^{\alpha_2} \cdots X_n^{\alpha_n}] = \frac{\partial^{(\alpha_1+\cdots+\alpha_n)}}{\partial {s_1}^{\alpha_1} \cdots \partial {s_n}^{\alpha_n}} M_{\mathbf{X}}\Big|_{\mathbf{s}=0} \tag{2}$$

where X_i is the i'th index of the vector $\mathbf{X}$ and s_i is the i'th index of the vector $\mathbf{s}$.

If the function is analytic (i.e. permits a Taylor expansion), then we obtain the following useful property:

Theorem. If there exists an epsilon-neighborhood about $\mathbf{s} = 0$ for which the MGF is finite, then every moment of $\mathbf{X}$ is finite.

This is useful since we no longer have to worry about checking if higher moments of $\mathbf{X}$ diverge. We only need to show that the MGF is finite in an epsilon-neighborhood about $\mathbf{s} = 0$. This is almost always the case for sufficiently "well-behaved" functions used in probability theory.

Propagating the MGF has some unique advantages over using the PDF. It allows us to use an arbitrary continuous function to represent the state instead of assuming the Gaussian distribution. This can be done by noting the linearity properties of MGFs:

Theorem. *(Linearity for constant vector)*
If $\mathbf{Y} = A\mathbf{X} + \mathbf{b}$, where A and $\mathbf{b}$ are constants, then

$$M_{\mathbf{Y}}(\mathbf{s}) = e^{\mathbf{s}^T\mathbf{b}} M_{\mathbf{X}}(A\mathbf{s})$$

Theorem. *(Linearity for independent vectors)*
If $\mathbf{X}$ and $\mathbf{Y}$ are independent random variables, then

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

Both theorems can easily be proven using the definition of MGFs.

B. Model Assumptions

We assume the model can be decomposed into three separate steps: motion step, sensor step, and finally fusion step. We discuss each in turn:

1) Motion Model: We assume a linear motion model of the form

$$\tilde{\mathbf{X}}_t = A\mathbf{X}_{t-1} + B\mathbf{u}_{t-1} \tag{3}$$

where $\mathbf{u}_t$ is the stochastic control input random variable centered around the true control input. We do not require the random variables $\mathbf{X}$ and $\mathbf{u}$ to be of any specified distribution form. The only requirement is that the distribution is continuous and analytic over state space.

2) Sensor Model: The state estimation using the sensor measurement ($\mathbf{Z}_{t-1}$) permits any arbitrary continuous probability density function. The only restrictions are that $\mathbf{Z}_{t-1}$ is continuous & analytic over state space and that the domain of the distribution is the same as the state estimate $\mathbf{X}$. For example, if the sensor used is a laser rangefinder, we need to apply the following transform to the sensor readings

$$\{r, \beta\} \rightarrow \{x, y, \theta\}$$

so that the range and bearing is converted into coordinates plus orientation.

3) Fusion Model: After separate state estimates are obtained from the motion and sensor models, we combine them linearly:

$$\mathbf{X}_t = \alpha \mathbf{Z}_{t-1} + (1-\alpha)\tilde{\mathbf{X}}_t \tag{4}$$

where $\alpha \in \mathbb{R}_{[0,1]}$ is a hyper-parameter that can be tuned. A possible method could be based on the Akaike information criterion [2]. We also assume that $\mathbf{Z}_{t-1}$ is independent of $\tilde{\mathbf{X}}_t$. This allows us to apply the linearity properties of MGFs.

C. Derivation in MGF Space

We now derive the state estimate propagation in MGF space. Using the motion model (Equation 3) and fusion model (Equation 4), we apply the MGF transformation to both sides to obtain:

$$\begin{aligned} M_{\mathbf{X}_t}(\mathbf{s}) &= M_{\mathbf{Z}_{t-1}}(\alpha \mathbf{s}) M_{\tilde{\mathbf{X}}_t}((1-\alpha)A\mathbf{s}) \\ &= \underbrace{M_{\mathbf{u}_{t-1}}((1-\alpha)B\mathbf{s})}_{f} \underbrace{M_{\mathbf{Z}_{t-1}}(\alpha \mathbf{s})}_{g} \underbrace{M_{\mathbf{X}_{t-1}}((1-\alpha)A\mathbf{s})}_{h} \end{aligned} \tag{5}$$

Note that the posterior state representation of $\mathbf{X}_t$ in MGF space is a simple product of three MGFs corresponding to the control distribution (f), sensor measurement distribution (g), and prior state distribution (h).

D. Transformation to Moment Space (Scalar $\mathbf{X}$)

Using the evaluation property of MGFs, the abstract MGF functions can be transformed into moment space. This allows the state propagation to become more tractable in the form of a look-up table of the moments of $\mathbf{Z}_{t-1}$, $\mathbf{u}_{t-1}$, and $\mathbf{X}_{t-1}$. For an univariate (scalar) state estimate $\mathbf{X}$, we can apply the n'th derivative to both sides of Equation 5 and invoking Lemma 1 to obtain the following:

$$\begin{aligned} \mathbb{E}[\mathbf{X}_t^n] &= M_{\mathbf{X}_t}^{(n)}\Big|_{\mathbf{s}=0} \\ &= \frac{\partial^n}{\partial s^n}[f(s)g(s)h(s)]\Big|_{s=0} \\ &= \sum_{i=0}^{n}\sum_{j=0}^{n-i} \frac{n!}{i!j!(n-i-j)!}[f^{(i)}(s)g^{(j)}(s)h^{(i)}(s)]\Big|_{s=0} \\ &= \sum_{i=0}^{n}\sum_{j=0}^{n-i} \frac{n!}{i!j!(n-i-j)!} \times [(1-\alpha)B]^i \mathbb{E}[\mathbf{u}_{t-1}^i] \\ &\times \alpha^j \mathbb{E}[\mathbf{Z}_{t-1}^j] \times [(1-\alpha)A]^{n-i-j}\mathbb{E}[\mathbf{X}_{t-1}^{n-i-j}] \end{aligned} \tag{6}$$

Example. *(Recovering the mean)*
Let $n = 1$, we recover

$$\mathbb{E}[X_t] = \alpha \mathbb{E}[Z_{t-1}] + (1-\alpha)(A\mathbb{E}[X_{t-1}] + B\mathbb{E}[u_{t-1}])$$

Example. *(Recovering the variance)*
Let $n = 2$, we recover (for deterministic u_{t-1})

$$\begin{aligned} \mathbb{E}[X_t^2] &= \alpha^2(\mathbb{E}[Z_{t-1}^2] - \mathbb{E}[Z_{t-1}]^2) \\ &+(1-\alpha)^2A^2(\mathbb{E}[X_{t-1}^2] - \mathbb{E}[X_{t-1}]^2) \\ &+ [\alpha\mathbb{E}[Z_{t-1}] + (1-\alpha)(A\mathbb{E}[X_{t-1}] + Bu_{t-1})]^2 \\ &= var[X_t] + \mathbb{E}[X_t]^2 \end{aligned}$$

which is precisely the equations for the 1st and 2nd moments of a linear combination of random variables.

E. Transformation to Moment Space (Vector $\mathbf{X}$)

The derivation is valid for a multivariate state estimate $\mathbf{X} \in \mathbb{R}^d$. However, the dimensionality of the higher moments scales as a power of the moment. For the n'th moment of $\mathbf{X}$, the dimensionality of the moment tensor is $\mathbb{R}^{d^n}$.

Applying the n'th derivative tensor to both sides of Equation 5 and invoking Theorem 2, we obtain the following:

$$\begin{aligned} \mathbb{E}[\mathbf{X}_t^{\otimes n}] &= D^n(M_{\mathbf{X}_t})\Big|_{\mathbf{s}=0} \\ &= D^n(f(\mathbf{s})g(\mathbf{s})h(\mathbf{s}))\Big|_{\mathbf{s}=0} \\ &= \sum_{(i_1,\ldots,i_n)\in\{f,g,h\}^n} (A_{i_1} \otimes \cdots \otimes A_{i_n}) \underbrace{\mathbb{E}[V_{i_1} \otimes \cdots \otimes V_{i_n}]}_{\text{factorizes if independent}} \end{aligned} \tag{7}$$

where we define

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

with D^n being the n'th derivative tensor, $\otimes$ being the tensor (outer) product, and $(i_1, \ldots, i_n) \in \{f, g, h\}^n$ indicating the order n permutation group on $\{f, g, h\}$.

For the expectation term, the tensor product can be separated for independent variables. For example, if the permutation group is (f, f, f, g, h), then the expectation is separated into

$$\mathbb{E}[\mathbf{u}_{t-1}^{\otimes 3}] \otimes \mathbb{E}[\mathbf{Z}_{t-1}] \otimes \mathbb{E}[\mathbf{X}_{t-1}]$$

which corresponds to a tensor product between the third moment (unstandardized skewness) of $\mathbf{u}_{t-1}$ and the first moments (mean) of $\mathbf{Z}_{t-1}$ and $\mathbf{X}_{t-1}$.

Example. *(Recovering the mean)*
Let $n = 1$, we recover

$$\mathbb{E}[\mathbf{X}_t] = \alpha \mathbb{E}[\mathbf{Z}_{t-1}] + (1-\alpha)(A\mathbb{E}[\mathbf{X}_{t-1}] + B\mathbb{E}[\mathbf{u}_{t-1}])$$

which is precisely the equation for the 1st moment (mean) of a linear combination of random variables.

Example. *(Recovering the variance)*
Let $n = 2$ and work through the summation. There should be 9 terms in total. The exercise is quite trivial and left as an exercise for the reader.

Characterizing the state propagation using Equation 7 allows us to use look-up tables of moments of $\mathbf{Z}_{t-1}$, $\mathbf{u}_{t-1}$, and $\mathbf{X}_{t-1}$ to calculate the posterior state $\mathbf{X}_t$. This eliminates the need to calculate matrix inverses as is required by the Kalman filter.

Additionally, we are no longer restricted on only using Gaussian distributions as the state variables. As long as the states are continuous and analytic, the state propagation obtained from Equation 7 is valid.

Finally, note that from Lemma 1 and Theorem 2, the Taylor expansion of the MGF $M_{\mathbf{X}}(\mathbf{s})$ about $\mathbf{s} = 0$ recovers the expectations of $\mathbf{X}$. So for analytic MGFs that have a convergent Taylor series, we can approximate the actual MGF of $\mathbf{X}$ using only finitely many moments of $\mathbf{X}$ to any arbitrary precision. The precision estimate is bounded by Taylor's Remainder Theorem [1].

IV. ALGORITHM

The pseudo-code for the MGF-Localization algorithm is shown in Algorithm 1:

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

The GitHub repository for our implementation is available here:

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

V. IMPLEMENTATION

A. 1D Localization through Moment Generating Functions

We implemented the 1D case using the MGF localization Algorithm 1. The implementation used the symbolic representation of the MGF and was done using functional programming so that operators can be imposed on explicit functions. Obstacles were placed at five, ten, and fifteen meters. The robot moves at one meter per second for twenty meters and has a sensor range of eight meters only looking forward.

B. 2D Localization through Moment Generating Functions

We tested 2D MGF localization on a 2D corridor with multiple obstacles with 6 sensor measurements per timestep. The robot had a sensor range of 10 meters. The environment was designed to have four symmetrical corridors that the robot enters from the East corridor and then it goes through the South corridor into a room.

C. 1D Localization through Expected Value Updates of Moments

We implemented the 1D localization using explicit expected value updates derived from MGF formulations. In this scenario, the robot moves forward one meter per second along a straight path, with obstacle landmarks placed at fixed positions. At each timestep, the robot performs a control step followed by a sensor measurement, observing distances relative to the landmarks with additive Gaussian noise. Through successive iterations, our method integrates these measurements and control actions to estimate the robot's global position accurately. Note that we only calculate the first and the second moment of the MGF. That is, we set $N = 2$.

VI. EXPERIMENTS AND RESULTS

A. 1D Localization through Moment Generating Functions

The dataset used in this simulation is synthetically generated and represents the robot's true position over 20 time steps as it moves steadily to the right, advancing one unit per step. Three fixed obstacles are positioned at locations 5, 10, and 15. At each time step, the robot receives simulated sensor measurements: if obstacles are within a specified range (8 units ahead), the measurements are modeled as a mixture of Gaussians centered around those obstacle positions; otherwise, the measurements are modeled as a Gaussian centered at the robot's true location. This dynamic sensor data drives the MGF-based localization updates.

1D localization with MGF's has been shown to be successful, based on our results from Figure 1. The figure shows that the initial estimation of the robot's location starts with high standard deviations, but as more measurements are taken along the robot path, the standard deviation drops drastically and our localized pose matches the true pose.

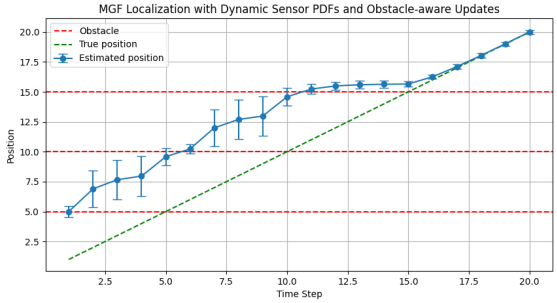


Fig. 1. 1D Localization through Moment Generating Functions

B. 2D Localization through Moment Generating Functions

The dataset for this simulation is synthetically generated and represents a robot navigating a structured environment composed of an east corridor, a south corridor, and a room. Landmarks are placed along the walls, sampled at regular intervals to create discrete points. The robot's true path is defined by a sequence of straight-line motions and turns, with slight random noise added to simulate motion uncertainty. At each time step, six noisy sensor readings estimate the robot's distance and angle to nearby landmarks, providing the observations that drive the MGF-based SLAM localization process.

2D Localization with MGF's ran into computational limitations. Due to the symbolic integration being done in Python and the large number of measurements, symbolic terms continued to grow until maxing out the simulation computer's memory. This is a limitation to of the current MGF implementation, which would need to be corrected for real world deployment.

C. 1D Localization through Expected Value Updates of Moments

For the dataset in our experiment, we manually construct data where obstacles are placed at position 5, 10, and 15, and the robot starts with position 0. In each timestep, the robot move forward by 1 unit control input.
As seen in Figure 2, our experiments demonstrated the effectiveness of using expected value updates of moments for 1D localization. Initially, the estimated position had a larger variance due to uncertainty from limited observations. As the robot proceeded and collected more measurements, the variance steadily decreased, resulting in highly accurate localization. The estimated trajectory closely matched the true positions over time, confirming the robustness and accuracy of our method under realistic conditions involving sensor noise and obstacle-aware updates. In conclusion, this method is able to model simple distribution. However, for the 2D or more dimension cases, real-time updates become more challenging since the terms that needs to be calculated are much more to model complex distributions.

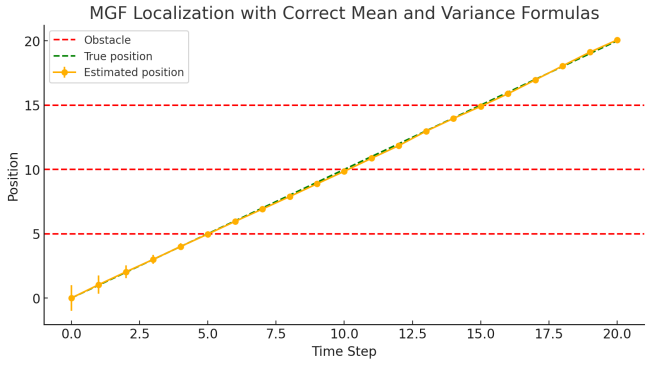


Fig. 2. 1D Localization through Expected Value Updates

VII. CONCLUSION & FUTURE WORK

Our work demonstrates that Moment Generating Functions offer a promising alternative to traditional Gaussian-based approaches for state estimation, particularly in scenarios with geometric symmetries or multi-modal uncertainty. The results from 1D experiments validate the method's accuracy and robustness, while also revealing its potential for further development.

However, the current symbolic computation implementation limits scalability to higher-dimensional spaces like 2D or 3D localization.

In future research, we can focus on:

1) Replacing symbolic integration with numerical or approximate techniques to reduce memory consumption.
2) Investigating stronger, more realistic motion models beyond the standard Gaussian assumption. Since the MGF representation does not require a fixed distribution form, it can naturally incorporate non-Gaussian uncertainties (e.g., heavy-tailed or multimodal noise), which often arise in real-world robot motion.
3) Benchmarking against standard SLAM algorithms in both simulated and physical environments to evaluate the scalability, accuracy, and robustness of the MGF-based approach.

VIII. REFERENCES

[1] Lior Bary-Soroker and Eli Leher. *On the remainder in the Taylor theorem*. 2008. arXiv: 0801.1271 [math.CA]. URL: https://arxiv.org/abs/0801.1271.

[2] Joseph E. Cavanaugh and Andrew A. Neath. "The Akaike information criterion: Background, derivation, properties, application, interpretation, and refinements". In: *WIREs Computational Statistics* 11.3 (2019), e1460. DOI: https://doi.org/10.1002/wics.1460.

[3] Harrison Delecki et al. "Deep normalizing flows for state estimation". In: *2023 26th International Conference on Information Fusion (FUSION)*. IEEE. 2023, pp. 1–6.

[4] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. "On sequential Monte Carlo sampling methods for Bayesian filtering". In: *Statistics and computing* 10 (2000), pp. 197–208.

[5] Dieter Fox. "Adapting the sample size in particle filters through KLD-sampling". In: *The international Journal of robotics research* 22.12 (2003), pp. 985–1003.

[6] Xi Geng. *The Theory of Moment Generating Functions*. Unpublished manuscript or lecture notes. n.d.

[7] Simon J Julier and Jeffrey K Uhlmann. "New extension of the Kalman filter to nonlinear systems". In: *Signal processing, sensor fusion, and target recognition VI*. Vol. 3068. Spie. 1997, pp. 182–193.

[8] Simon J Julier and Jeffrey K Uhlmann. "Unscented filtering and nonlinear estimation". In: *Proceedings of the IEEE* 92.3 (2004), pp. 401–422.

[9] Haofei Kuang et al. “Ir-mcl: Implicit representation-based online global localization”. In: *IEEE Robotics and Automation Letters* 8.3 (2023), pp. 1627–1634.

[10] Le Song et al. “Hilbert space embeddings of hidden Markov models”. In: (2010).

[11] Erik B Sudderth et al. “Nonparametric belief propagation”. In: *Communications of the ACM* 53.10 (2010), pp. 95–103.

[12] Rudolph Triebel, Patrick Pfaff, and Wolfram Burgard. “Multi-level surface maps for outdoor terrain mapping and loop closing”. In: *2006 IEEE/RSJ international conference on intelligent robots and systems*. IEEE. 2006, pp. 2276–2282.

[13] Nuo Xu et al. “GP-Localize: Persistent mobile robot localization using online sparse Gaussian process observation model”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 28. 1. 2014.

Elevation Mapping: Discrete Resolution Simultaneous Elevation Mapping and Localization

Boxiang Fu[1†], Joshua Pen[2†], Ricky Yuan[3†], and Tianzhi Li[4†]

***Abstract*— For autonomous mobile robots operating in a confined environment of established map dimensions, occupancy grid maps provide a simple and computationally-efficient method for mapping the environment. However, such maps restrictively represent each cell using a binary classifier as being either occupied or free. Furthermore, the algorithm requires localization as an input and cannot both map and localize simultaneously. We extend Moravec and Elfes' original occupancy grid algorithm by extending the classifier from a binary state (occupied/free) to a continuous state (elevation of cell). Furthermore, we estimate the camera pose using visual tracking so both localization and mapping can be simultaneously achieved. Finally, we implemented the algorithm in ROS (`rclcpp`) so that real-time performance can be attained.**

I. INTRODUCTION

Accurate and efficient mapping of real-world environments is critical for a wide range of robotic applications. Traditional mapping methods, such as occupancy grid maps, are widely used for their simplicity but are limited in their ability to represent continuous variations in terrain. In scenarios such as landscape reconstruction, terrain-aware path planning, or inspection tasks, robots need to understand not just whether a space is occupied, but how the terrain varies in elevation. Applications could include using UAVs for creating detailed maps of natural landscapes, supporting environmental monitoring, or assisting in search and rescue by providing real-time elevation maps of affected areas.

In this work, we propose a method that extends the conventional 2D occupancy grid map to the 2.5D elevation map, which is able to capture the vertical dimension of the environment. Our system uses a stereo camera called ZED 2i, which is paired with a depth sensor to accurately estimate 3D point clouds and the camera's 6-DoF pose for each frame. By filtering these point clouds to remove noise and outliers, we extract meaningful elevation data that can be used to construct the elevation map. Furthermore, the algorithm is computationally efficient and implemented in ROS2 with C++, making it suitable for real-time applications.

In summary, we developed a fast and accurate elevation mapping framework that combines dense 3D point cloud processing with real-time visual localization. This system enables autonomous robots to build detailed elevation maps of their surroundings while tracking their own pose.

[1]boxiangf@cs.cmu.edu
[2]jpen@cs.cmu.edu
[3]rickyy@andrew.cmu.edu
[4]tianzhil@cs.cmu.edu
[†]Robotics Institute, Carnegie Mellon University

II. RELATED WORK

Occupancy grid mapping was first introduced by Moravec and Elfes [3], offering a probabilistic framework for environment representation in mobile robotics. Their method discretizes space into a grid where each cell contains the probability of being occupied or free, providing a computationally efficient and robust mapping solution. This framework has been widely adopted in robotic systems due to its simplicity and compatibility with various sensor modalities.

However, the binary nature of traditional occupancy grids imposes a limitation when modeling complex or uneven environments. To address this, elevation maps have been proposed [4], extending occupancy grids by encoding each cell with height information. These maps are especially useful for ground robots navigating 3D terrains, allowing for obstacle detection and path planning in non-flat environments. Our work builds on this concept by integrating elevation as a continuous state, rather than a binary or probabilistic classification.

Simultaneous Localization and Mapping (SLAM) is another key component in autonomous navigation. Classic occupancy grid mapping assumes perfect localization input, which is often unrealistic. SLAM algorithms [2, 5] address this by jointly estimating both the map and the robot pose. Visual SLAM approaches, such as ORB-SLAM [6], utilize camera data for localization and mapping in real time. Inspired by these methods, our system employs visual tracking to estimate the camera pose, enabling real-time elevation mapping without relying on external localization.

Finally, our implementation in ROS using `rclcpp` aligns with a broader trend of modular, open-source robotic development frameworks [7]. This allows for seamless integration with other ROS-based tools and facilitates real-time performance, which is essential for deployment in practical robotics applications.

III. METHODOLOGY

A. Localization

In our system, localization is achieved by fusing visual and inertial data obtained from the ZED 2i stereo camera. The ZED 2i is equipped with an RGB stereo camera, an Inertial Measurement Unit (IMU), and GNSS; however, in our experiments, we utilize only the IMU and RGB images to estimate the camera's 6-DoF pose.

We employ a Visual-Inertial Odometry (VIO) approach provided by the ZED SDK, where visual data from the stereo camera provides depth and feature correspondence

between frames, while the IMU supplies acceleration and angular velocity measurements. The fusion of these data sources allows for more robust and accurate pose estimation, especially in conditions where visual tracking alone might degrade, such as in low-texture environments or during rapid motion.

By fusing stereo vision and IMU, we accurately estimate the camera's 6-DoF pose in real-time without GNSS, ensuring consistent elevation mapping and frame alignment.

B. Bayes Filter

Whenever a cell receives new elevation information, a Bayes filter function is called to fuse the original elevation with the new elevation information. This is done using a Kalman filter in one dimension. The mathematical derivation for the filter can be found in Reference [1], with a graphical depiction shown in Figure 1.

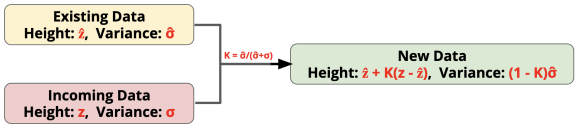


Fig. 1: Kalman Filter in one dimension

C. Fusion

The algorithm was implemented in ROS using `rclcpp` to attain real-time performance requirements. The entire cyberphysical architecture is depicted in Figure 2. The ZED 2i stereo camera operates at 30 Hz and relays its camera feed at this frequency. This feed is passed through the ZED ROS wrapper packages to enable compatibility with ROS.

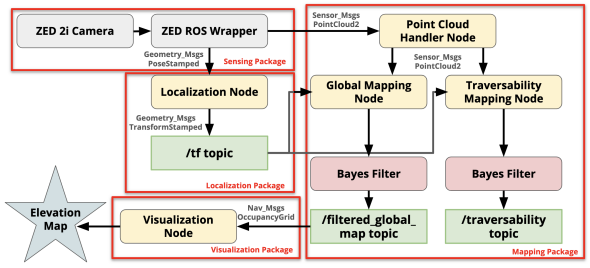


Fig. 2: Cyberphysical stack of elevation mapping algorithm

At each timestep, the wrapper calls both the `Localization` node and the `Point Cloud Handler` node. The `Localization` node parses the ROS wrapper information and publishes to the `/tf` topic on the extrinsic homogeneous transform between the map frame and the camera frame. The `Point Cloud Handler` node parses the raw point cloud from the ZED camera and filters it to remove any noise and/or outliers. The filtered point cloud is then re-published to be used.

Next, the `Global Mapping` node and the `Traversability Mapping` node subscribes to the filtered point cloud topic and the `/tf` topic. Using the extrinsic homogeneous transform between the map frame and the camera frame, the filtered point cloud is transformed from the camera frame to the map frame. For every grid cell in the map, we average all the point cloud elevation datapoints that fall inside the cell. This new averaged elevation is then sent to the Bayes filter function to be fused with the original elevation information. Once fusion is complete, the elevation map is published to the `/filtered_global_map` topic and the original binary occupancy map is published to the `/traversability` topic. The `Visualization` node subscribes to the `/filtered_global_map` topic and processes the elevation map for visualization in RViz.

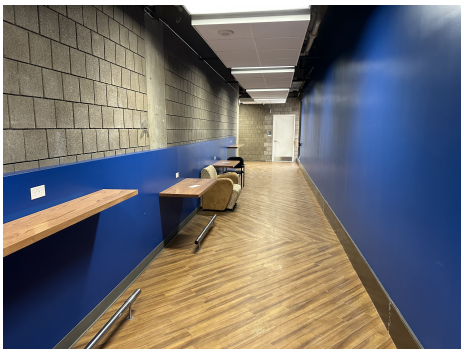


Fig. 3: Environment 1 – Straight hallway with tables and chairs along the left side

IV. CODE

The GitHub repository for our ROS implementation is available here:

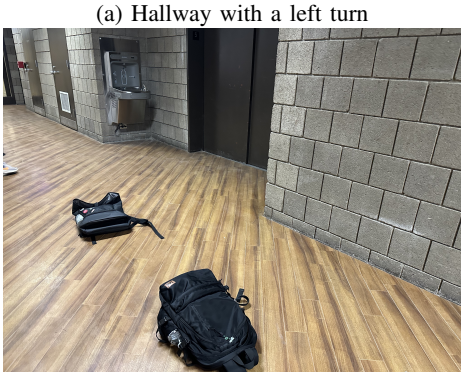
`https://github.com/CMU-SLAM25/Elevation-Mapping`

V. DATASET

To support the development and evaluation of our elevation mapping system, we collected a real-time dataset using an RGB-D camera at 30 frames per second. The dataset was gathered on the 5th floor of Wean Hall as well as Newell-Simon Hall 1305 and covers three distinct indoor environments representative of typical navigation scenarios.

- **Environment 1: Straight hallway**
  As shown in Figure 3, a long corridor with tables and chairs placed along the left side. This scene provides a structured layout with moderate clutter.
- **Environment 2 Hallway with a left turn**
  As shown in Figure 4, a hallway that includes a left turn, two bags placed on the floor, and an elevator on the right. This environment introduces fine-grained elements and occlusions for testing robustness.
- **Environment 3: Lecture hall with rows of tables and chairs**
  As shown in Figure 5, a lecture room containing multiple rows of tables and chairs. This environment provides a more complex and densely populated layout.

(a) Hallway with a left turn



(b) Hallway with two bags on the floor and an elevator on the right

Fig. 4: Environment 2 – Hallway with a left turn with two bags on the floor and an elevator on the right

Fig. 5: Environment 3 – Lecture hall with rows of tables and chairs

VI. EXPERIMENT

To evaluate the performance of the elevation mapping system, experiments were conducted in indoor environments at Carnegie Mellon University. The first environment was a straight hallway on the 5th floor of Wean Hall with tables and chairs placed along the left side, providing a structured scene with moderate static obstacles. The second environment was also on the 5th floor of Wean Hall and consisted of a hallway with a left turn, two bags placed on the floor, and an elevator on the right side, introducing localized elevation variations and occlusions. In addition, we conducted a third experiment in Room 1305 of Newell-Simon Hall, a lecture hall with rows of tables and chairs organized in a classroom format. During each trial, the robot traversed the environment while capturing synchronized stereo images and point cloud data using a ZED 2i stereo camera. Pose estimation and elevation mapping were performed in real time using the developed ROS framework. The resulting elevation maps were compared qualitatively to the known layouts to assess the system's ability to reconstruct both large-scale geometry and small object-level features.

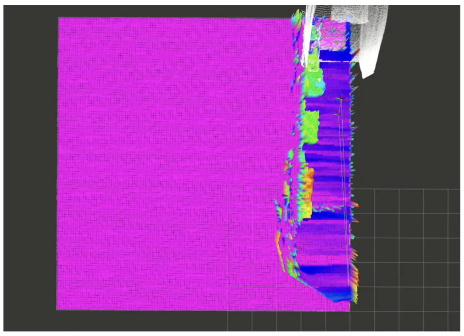


Fig. 6: The generated elevation map for Environment 1

VII. RESULTS

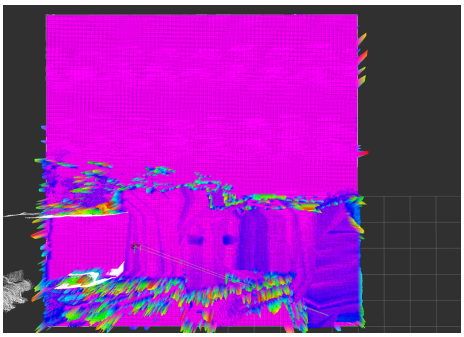
Our elevation mapping system successfully captured the structural features of both testing environments in Wean Hall. The generated elevation maps reflect the geometry and elevation changes in the scene with high fidelity, validating the effectiveness of our approach.

In **Environment 1** (the straight hallway), as shown in Figure 6, the map clearly identifies the structure of the straight hallway and the positions of the tables and chairs placed along the left side. These objects appear as distinct elevated regions relative to the flat hallway floor, demonstrating the map's ability to represent static obstacles with elevation differences.

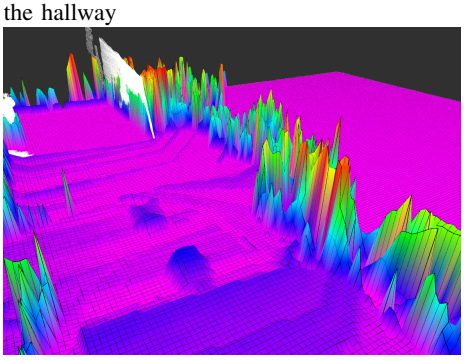
In **Environment 2** (the hallway with a left turn), as shown in Figure 7, the elevation map correctly captures the left turn in the hallway, enabling the layout of the hallway to be reconstructed. Additionally, two localized bumps detected in the elevation map correspond to the bags placed on the floor, while a recessed region in the wall aligns with the location of the elevator. These details confirm that the system is capable of mapping fine-grained features in indoor environments.

In contrast, results in **Environment 3** (the lecture hall), as shown in Figure 8, show significant limitations. The elevation map exhibits high noise levels and suffers from noticeable drift. The repetitive layout of rows of tables and chairs, combined with limited visual texture, likely contributed to degraded tracking performance and accumulated localization error. This highlights the challenges of operating in densely populated, visually ambiguous environments and suggests a need for improved robustness in pose estimation under such conditions.

Overall, the system performs well in structured and moderately cluttered spaces, but its performance degrades in highly cluttered environments, indicating important areas for future improvement.



(a) The elevation map captured the left turn in the hallway



(b) The elevation map captured two bags and the elevator on the right side

Fig. 7: The generated elevation map for Environment 2

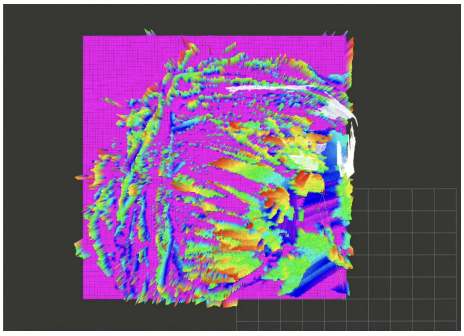


Fig. 8: The generated elevation map for Environment 3

VIII. CHALLENGES

One of the main challenges in our project was accurate localization. Initially, we used a RealSense camera as our sensor, obtaining point clouds from the camera's built-in processing of RGB and depth images. We manually computed surface normals from these point clouds and implemented the Iterative Closest Point (ICP) algorithm in C++ for pose estimation. However, the results were unstable, with frequent issues such as pose drift and flickering, causing the estimated camera pose to deviate significantly from the ground truth.

We suspected two possible reasons for the results. First, the quality of the point clouds was highly dependent on the visual richness of the environment. In low-feature areas, such as flat walls, floors, or smooth surfaces, the point cloud became ambiguous, and reduce the reliability of the ICP alignment. Second, our custom implementation of ICP may have suffered from suboptimal parameter settings, i.e. insufficient iterations or inappropriate distance thresholds.

To address these issues, we switched to using the ZED 2i stereo camera as mentioned, which offers more robust and accurate depth sensing. In addition to stereo depth, the ZED 2i also integrates an IMU, allowing us to leverage visual-inertial fusion for improved pose estimation. By combining stereo-based point cloud generation with high-frequency inertial data, we achieved more stable and accurate 6-DoF localization. In our experiments, the visual-inertial odometry (VIO) provided by ZED's SDK proved to be more reliable and stable than our initial ICP-based approach.

IX. FUTURE WORK

While the current elevation mapping system provides a strong foundation for terrain representation and navigation, several directions remain open for future improvement:

1) **Improving Robustness in Cluttered Environments** In cluttered or dynamic scenes, traditional elevation mapping approaches can struggle due to noisy or unstable feature matches during pose estimation. To address this, we propose incorporating semantic segmentation to identify and prioritize reliable, static structures such as ground, walls, and permanent obstacles. Additionally, filtering out unstable or transient features—such as vegetation or moving objects—during the pose estimation step can lead to more stable map updates and improved overall robustness.
2) **Extending to Full 3D Reconstruction** Standard elevation maps are inherently limited to a 2.5D representation, which cannot accurately model overhangs, bridges, or vertical structures. To overcome this, we aim to extend the elevation mapping framework to support full 3D reconstruction. This could involve integrating volumetric mapping techniques such as truncated signed distance fields (TSDFs) or surfel-based representations to capture the full geometric complexity of the environment.
3) **Integrating Dynamic Obstacle Detection** For real-world deployment, especially in environments shared with humans or moving vehicles, the system must be able to detect and react to dynamic obstacles. Future work will focus on integrating real-time dynamic obstacle detection into the mapping pipeline. This would enable the elevation map to support more advanced navigation tasks, such as trajectory replanning or collision avoidance, making the system suitable for practical autonomous navigation in dynamic, unstructured settings.

X. CONCLUSION

This project extends traditional occupancy grid mapping by incorporating continuous elevation estimation and simultaneous localization through visual-inertial odometry. The system was implemented in ROS and demonstrated the ability to reconstruct indoor environments with high structural fidelity in real time. Experiments conducted in

structured hallway environments showed that the system could accurately capture both the overall layout and smaller obstacles, such as tables, chairs, and bags. In more cluttered environments, such as a lecture hall in Newell-Simon Hall, the system faced challenges with mapping accuracy and localization stability due to dense object arrangements and noisy sensor measurements. These results demonstrate that the proposed elevation mapping system is effective in structured spaces and highlight its current limitations in highly cluttered indoor environments.

XI. REFERENCES

[1] Alex Becker. *Kalman Filter in One Dimension*. Accessed: 2025-04-20. 2025. URL: https://www.kalmanfilter.net/kalman1d.html.

[2] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous localization and mapping: part I". In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110.

[3] Alberto Elfes. "Using occupancy grids for mobile robot perception and navigation". In: *Computer* 22.6 (1989), pp. 46–57.

[4] Péter Fankhauser et al. "Robot-centric elevation mapping with uncertainty estimates". In: *Mobile Service Robotics*. World Scientific, 2014, pp. 433–440.

[5] Giorgio Grisetti, Cyrill Stachniss, and Wolfram Burgard. "Improved techniques for grid mapping with rao-blackwellized particle filters". In: *IEEE transactions on Robotics* 23.1 (2007), pp. 34–46.

[6] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. "ORB-SLAM: A versatile and accurate monocular SLAM system". In: *IEEE transactions on robotics* 31.5 (2015), pp. 1147–1163.

[7] Morgan Quigley et al. "ROS: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*. 2009.