

Probabilistic Object Maps for Long-Term Robot Localization

Amanda Adkins¹ and Joydeep Biswas¹

Abstract—Robots deployed in settings such as warehouses and parking lots must cope with frequent and substantial changes when localizing in their environments. While many previous localization and mapping algorithms have explored methods of identifying and focusing on long-term features to handle change in such environments, we propose a different approach – can a robot understand the distribution of movable objects and relate it to observations of such objects to reason about global localization? In this paper, we present probabilistic object maps (POMs), which represent the distributions of movable objects using pose-likelihood sample pairs derived from prior trajectories through the environment and use a Gaussian process classifier to generate the likelihood of an object at a query pose. We also introduce POM-Localization, which uses an observation model based on POMs to perform inference on a factor graph for globally consistent long-term localization. We present empirical results showing that POM-Localization is indeed effective at producing globally consistent localization estimates in challenging real-world environments, and that POM-Localization improves trajectory estimates even when the POM is formed from partially incorrect data.

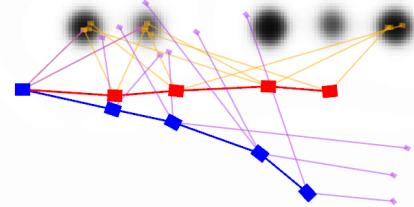
I. INTRODUCTION

Mobile robots deployed in real world environments with humans frequently encounter changes due to movable objects. Since localization algorithms that assume the world is static fare poorly in such dynamic environments, state-of-the-art long-term localization approaches explicitly model movable and moving objects in an attempt to improve robustness, but come with several limitations. Some of these approaches [1]–[5] attempt to discover which features persist over long time scales and either discard the remaining features or keep them only for short-term use. This reduces the information available for localization, particularly when such movable objects comprise a large portion of the scene, and can cause the localization estimate to drift over time. Other methods [6]–[8] impose assumptions about the configurations or movement patterns of objects that may not match the true dynamics of the environment.

In many environments, movable objects tend to follow patterns, with some areas more likely to contain objects than others. Consider a parking lot like the one in Fig. 1a: the scene contains many movable objects, and consequently, this environment would challenge many localization algorithms. There is no fixed set of configurations and the individual objects may not have consistent periodic movement. However, such objects do follow a distribution that describes where they are likely to occur and where they are less likely. This



(a) Parking lot at three different times.



(b) POM for the above parking configuration, with more frequently occupied spots resulting in higher map values. An initial trajectory is shown in blue with object detections in purple. An optimized trajectory from POM-Localization is shown in red. Corresponding object detections in orange line up with peaks in the POM.

Fig. 1: POM-Localization as applied to a parking lot.

holds true for other scenarios, such as pallets and boxes in warehouses or furniture in home and office environments.

We posit that localization algorithms can utilize the distribution of movable objects to improve robustness in real world environments. We propose a method for creating a probabilistic object map (POM) that models the likelihood of a movable object of a given semantic class occurring at a given pose. The POM is formed in a data-driven manner from object detections from past trajectories and uses Gaussian process classification to generate likelihoods for new query poses. We also introduce the POM-Localization algorithm, which uses POMs to inform localization. The goal of POM-Localization is to prevent significant drift in environments with a high density of movable objects, rather than to improve average-case performance. Given the current estimate for the robot’s trajectory and the object detections from that trajectory, POM-Localization calculates where objects would be in the global frame and adds a cost based on the likelihood of an object occurring at the pose, penalizing the trajectory when objects would occur in unlikely positions. As

¹Amanda Adkins and Joydeep Biswas are with the Department of Computer Science, The University of Texas at Austin, Austin, TX. Email: {aaadkins, joydeepb}@cs.utexas.edu

shown in Fig. 1b, this formulation is used to optimize the trajectory, resulting in a sequence of poses that best aligns with current object detections, odometry measurements, and the POMs. We also provide a method for incrementally updating the POM given a newly optimized trajectory with object detections.

We present experimental results on two different datasets to highlight the performance of POM-Localization in changing environments and to demonstrate the impact of our approach given limited knowledge of the object distribution represented in the POM.

II. RELATED WORK

A. Semantic SLAM and Localization

Similar to our method, semantic landmark SLAM and localization approaches rely on high-level semantic objects rather than low-level features. Many approaches, like [9], require the feature extractor to associate measurements to landmarks, which is the problem of data association. This can be challenging, particularly in dynamic environments, and mistakes can degrade results. Some methods try to improve robustness by shifting responsibility for data association to the optimization, allowing correspondence decisions to be updated as more information is obtained. Bowman, et al. [10] propose a semantic SLAM approach that integrates data association into the optimization using expectation maximization, while [11] and [12] use factor graphs with novel factors that handle data association. Like these approaches, our method uses high-level semantic objects and does not require the feature extractor to resolve data associations. However, our approach avoids the correspondence problem altogether by considering a distribution of objects rather than a discrete set of landmarks. These approaches also differ from ours in that they do not explicitly model changes in environments.

B. Localization and Mapping in Changing Environments

While many SLAM and localization approaches assume objects in the scene are static, some methods explicitly model moving and movable objects. Some techniques simply filter movable and/or moving objects from the data [1]. This improves robustness, but results in loss of information that could be valuable for localization. Others aim to understand which features persist over long time scales, with remaining features either discarded or kept only for short-term processing. [2] removes old data that conflict with more recent information. [3] probabilistically models feature persistence and [4] extends this approach to consider relationships between features. Episodic non-Markov Localization (EnML) [5] matches long-term features to a map and uses short-term features for relative corrections. Such methods require accurate data association to ensure the correct features are discarded and do not obtain global understanding from short-term features. In scenes with few long-term features, this can lead to drift in the localization estimate. Other approaches impose assumptions about the patterns of movable objects. Such assumptions include a limited range of possible configurations for regions of the space [6] or that object movement

conforms to some periodicity [7] or transition function [8]. For scenarios such as warehouses, parking lots, or areas with movable furniture, these assumptions may not appropriately model the real dynamics of the world, and consequently, localization performance could degrade.

C. Continuous Mapping

Appropriate map representations are critical to localization and SLAM. Occupancy grids are commonly used, but introduce errors from discretization. The fixed resolution of occupancy grids is also poorly suited to variable density data and continuous optimization techniques used in many modern localization approaches. Recent works have explored continuous map representations to avoid these shortcomings. Hilbert maps [13] model the occupancy of an environment by projecting data into a Hilbert space and using a logistic regression classifier to compute occupancy for a location, while Gaussian process occupancy maps [14] address the same problem using a Gaussian process classifier. We employ a similar technique for modeling object occurrences.

III. MAPPING AND LOCALIZATION WITH PROBABILISTIC OBJECT MAPS

We introduce the formulation for the POMs and the POM-Localization algorithm that uses these to inform robot localization in environments with movable objects. We also outline techniques to increase the speed of POM evaluation and POM-Localization.

Our approach requires a set of initial trajectories with object detections from which to bootstrap the POMs for each environment. These can be created from higher quality sensors, computationally intensive reference localization algorithms, or both. The POM-Localization algorithm also requires odometry estimates from either wheel encoders, inertial measurements, or visual odometry.

A. POM Evaluation using Gaussian Process Classification

The goal of a probabilistic object map is to estimate the likelihood that an object occurs at a given pose o^* . For this, we utilize a variant of Gaussian process classification (GPC) [15]. Our POM estimates the distribution $p(c^*=1|o^*)$, where $c^* \in [0, 1]$ is a class label indicating whether the pose is occupied, with $p(c^*=1|o^*)$ evaluating to 1 when there should always be an object at o^* and 0 when there is never an object at the pose. Let $o_{1:M}$ be the M sample inputs for which we have corresponding output values. GPC works by transforming the output of Gaussian process regression (GPR), denoted a^* , from the interval $(-\infty, \infty)$ to $[0, 1]$ using an activation function, such as the logistic function, $s(x) = \frac{1}{1+e^{-x}}$. Traditionally, GPC assumes ground truth training output values $t_{1:M} \in \{0, 1\}$ and requires approximations to map these to $a_{1:M} \in (-\infty, \infty)$ as used by the underlying GPR model. We instead assume that we directly obtain sample values $a_{1:M}$ that are used with GPR.

Using these quantities, $p(c^*=1|o^*)$ is given by GPC as

$$p(c^*=1|o^*) = \int p(c^*=1|a^*)p(a^*|a_{1:M}, o_{1:M}, o^*)da^*. \quad (1)$$

Based on the GPR formulation, $p(a^*|a_{1:M}, o_{1:M}, o^*)$ is a normal distribution having mean μ and variance σ^2 , with μ given by

$$\mu = \mu_0 + K_x^T K_D^{-1} (a - \mu_0), \quad (2)$$

where μ_0 is a prior mean on the interval $(-\infty, \infty)$, $a - \mu_0$ is a vector of values $a_{1:M}$ less μ_0 ,

$$K_D = \begin{bmatrix} k(o_1, o_1) & \dots & k(o_1, o_M) \\ \vdots & k(o_i, o_j) & \vdots \\ k(o_M, o_1) & \dots & k(o_M, o_M) \end{bmatrix}, \quad (3)$$

$$K_x = \begin{bmatrix} k(o_1, o^*) \\ \vdots \\ k(o_M, o^*) \end{bmatrix} \quad (4)$$

and $k(o_i, o_j)$ is a kernel function providing the similarity of o_i and o_j . Calculation of σ^2 is described in section III-B.

$p(c^*=1|a^*)$ is the logistic function, making $p(c^*=1|o^*)$ the convolution of a normal distribution and the logistic function. We adapt the approximation from [15] for such a convolution to incorporate the prior mean μ_0 , giving

$$p(c^*=1|o^*) \approx s \left(\mu_0 + \frac{\mu - \mu_0}{\sqrt{1 + \frac{\pi\sigma^2}{8}}} \right). \quad (5)$$

For POM evaluation, the kernel for computing μ is a scaled product of a radial basis function (RBF) kernel [15] on position and a periodic variant of an RBF kernel on orientation. A prior for the likelihood of an object is transformed from $[0, 1]$ by the logit function to obtain μ_0 .

B. Uncertainty Estimation in POMs

Traditionally, Gaussian process regression assumes that the output for a fixed input is drawn from a normal distribution. However, in our case, predicting the likelihood of an object occurrence is closer to predicting the bias of a coin, where we have a series of trials (past observations for poses) and we want to know the likelihood of an event (object occurrence at the given pose). For this reason, we are instead computing σ^2 using an approach derived from the desired properties of the variance: more sample inputs should result in lower variance and a set of samples that are closer to the query pose o^* should result in lower variance than the same number of more distant samples. Consequently, we approximate σ^2 using the inverse of an unnormalized kernel density estimator (KDE) [15], which satisfies these conditions. Using this, σ^2 is given by

$$\sigma^2(o^*) = \frac{1}{\text{KDE}(o^*)} = \frac{1}{\sum_{i=1}^M k_\sigma(o_i, o^*)}, \quad (6)$$

where k_σ is the same kernel function as in the calculation of μ with different parameters.

C. Building Probabilistic Object Maps

To evaluate the likelihood of an object occurring at a given pose using Gaussian process classification, we need a set of samples. Each sample is a pair of a pose o_i in the global frame and a value a_i in the range $(-\infty, \infty)$ representing the object likelihood at the pose based on past trajectories. When the POM is initially created, we use a set of highly accurate trajectories and their object detections. To generate the samples, we use two different sources to obtain sample poses in the local frame of each node from every past trajectory. The first source is the set of object detections for the node. To ensure that we understand both where objects are likely and unlikely, we also add *off-detection* sample poses drawn randomly from the free space around the robot. The next step is generating values for each sample pose. Given a set of object detection poses $\{s_{t_j}\}$ relative to the robot at time t_j and corresponding variances $\{\sigma_{t_j}^2\}$ for the object detections, the value \hat{a}_i for a sample pose \hat{o}_i relative to the robot can be obtained using the following formula:

$$\hat{a}_i = \max_{s_{t_j}} \mathcal{N}(\hat{o}_i | s_{t_j}, \sigma_{t_j}^2). \quad (7)$$

If there are no object detections, then \hat{a}_i is 0. We map $\{\hat{a}_i\}$ from $[0, \infty)$ to $(-\infty, \infty)$ using a series of transformations comprised of a squashing function to map values into the $[0, 1]$ range used for Gaussian process classification values $\{t_i\}$, a linear mapping to reduce the range to $[\epsilon, 1 - \epsilon]$, where ϵ is a small positive number, and a logit function to generate samples $\{a_i\}$. Mapping to $[\epsilon, 1 - \epsilon]$ prevents infinite values from the logit function from dominating POM evaluation. We finally use the localization estimates from past trajectories to transform the local frame sample poses $\{\hat{o}_i\}$, ultimately giving sample poses $\{o_i\}$ in the global frame and their corresponding values $\{a_i\}$ in the $(-\infty, \infty)$ range. In our approach, a separate POM is created for each movable object class. For example, in a campus setting, there would be a POM for parked cars and another for parked bicycles.

D. POM-Localization

To improve robot localization, we add a cost based on the likelihood generated by the POM for each detected object in the current trajectory. Let the belief over the current trajectory of poses $x_{1:n}$ be given by

$$\text{Bel}(x_{1:n}) = p(x_{1:n}|x_0, s_{1:n}, u_{1:n}) \quad (8)$$

$$\propto \prod_{i=1}^n \prod_{k=1}^{N_i} p(s_{i_k}|x_i) \prod_{j=0}^{n-1} p(x_{j+1}|x_j, u_{j+1}), \quad (9)$$

where N_i is the number of detections at pose x_i , s_{i_k} is the k th object detection relative to the robot at pose x_i , and u_i is the odometry measurement from x_{i-1} to x_i which is obtained from perceptual or wheel odometry and is normally distributed with covariance Σ_{odom} . s_{i_k} is composed of a relative pose r_{i_k} and classification variable c_{i_k} , and, as we do not consider negative information about objects, $c_{i_k}=1$ for all s_{i_k} . Next, $p(s_{i_k}|x_i) = p(r_{i_k}, c_{i_k}=1|x_i)$ can be written as

$$p(s_{i_k}|x_i) = p(r_{i_k}|x_i, c_{i_k}=1)p(c_{i_k}=1|x_i) \quad (10)$$

To use the object likelihood in the belief, we marginalize over the true object pose o_{i_k} in the global frame. We can express $p(r_{i_k}|x_i, c_{i_k}=1)$ in terms of o_{i_k} as follows

$$p(r_{i_k}|x_i, c_{i_k}=1) = \int_{o_{i_k}} p(r_{i_k}, o_{i_k}|x_i) p(o_{i_k}|x_i, c_{i_k}=1) do_{i_k} \quad (11)$$

$$= \int_{o_{i_k}} p(r_{i_k}, o_{i_k}|x_i) p(c_{i_k}=1|x_i, o_{i_k}) \frac{p(o_{i_k}|x_i)}{p(c_{i_k}=1|x_i)} do_{i_k}. \quad (12)$$

Since the existence of an object is independent of the robot's pose, i.e. $p(o_{i_k}|x_i) = p(o_{i_k})$, combining (10) and (12) yields

$$p(s_{i_k}|x_i) = \int_{o_{i_k}} p(r_{i_k}, o_{i_k}|x_i) p(c_{i_k}=1|o_{i_k}) p(o_{i_k}) do_{i_k}. \quad (13)$$

Substituting (13) into (9) gives an updated form for the belief. We frame the solution to the localization problem as finding the trajectory $x_{1:n}$ that minimizes the negative log likelihood given by

$$\begin{aligned} -\log(\text{Bel}(x_{1:n})) &\propto \frac{1}{2} \sum_{j=0}^{n-1} \|x_{j+1} \ominus (x_j \oplus u_{j+1})\|_{\Sigma_{\text{odom}}}^2 + \\ &\sum_{i=1}^n \sum_{k=1}^{N_i} -\log \int_{o_{i_k}} p(r_{i_k}|o_{i_k}, x_i) p(c_{i_k}=1|o_{i_k}) p(o_{i_k}) do_{i_k}. \end{aligned} \quad (14)$$

The integral above is intractable, so we estimate it by sampling. We represent the true object pose relative to the robot as \hat{o}_{i_k} and assume that the measurement r_{i_k} is normally distributed around \hat{o}_{i_k} . Since this is a normal distribution, $p(o_{i_k}|r_{i_k}, x_i)$ is equal to $p(r_{i_k}|o_{i_k}, x_i)$, allowing a sample $\hat{o}_{i_k s}$ to be drawn from $p(\hat{o}_{i_k s}|r_{i_k}, x_i)$. We thus approximate the integral by drawing N_s samples $\hat{o}_{i_k s}$ from $p(\hat{o}_{i_k s}|r_{i_k}, x_i)$, obtaining the corresponding global frame pose $o_{i_k s}$ from the sample using $o_{i_k s} = x_i \oplus \hat{o}_{i_k s}$, and then summing, giving

$$\begin{aligned} \int_{o_{i_k}} p(r_{i_k}|o_{i_k}, x_i) p(c_{i_k}=1|o_{i_k}) p(o_{i_k}) do_{i_k} &\approx \\ \frac{1}{N_s} \sum_{s=1}^{N_s} p(c_{i_k s}=1|o_{i_k s}) p(o_{i_k s}). \end{aligned} \quad (15)$$

We assume $p(o_{i_k})$ is uniform, so we can replace this by a normalization constant η , yielding

$$\frac{1}{N_s} \sum_{s=1}^{N_s} p(c_{i_k s}=1|o_{i_k s}) p(o_{i_k s}) \approx \frac{\eta}{N_s} \sum_{s=1}^{N_s} p(c_{i_k s}=1|o_{i_k s}). \quad (16)$$

Since the normalization constant is independent of the trajectory, we drop η . Combining (14), (15) and (16) gives

$$\begin{aligned} -\log(\text{Bel}(x_{1:n})) &\propto \frac{1}{2} \sum_{j=0}^{n-1} \|x_{j+1} \ominus (x_j \oplus u_{j+1})\|_{\Sigma_{\text{odom}}}^2 + \\ &\sum_{i=1}^n \sum_{k=1}^{N_i} -\log \left(\frac{1}{N_s} \sum_{s=1}^{N_s} p(c_{i_k s}=1|o_{i_k s}) \right). \end{aligned} \quad (17)$$

With this formulation and the POM to provide object likelihoods $p(c_{i_k s}=1|o_{i_k s})$, a nonlinear optimizer can be used to find the trajectory $x_{1:n}$ that best aligns with the movable object observations and odometry.

E. Updating the POM Based on New Trajectories

We can update the POM to incorporate information from a newly optimized trajectory to better capture the true distribution of objects in the environment. We generate new sample poses and values using the same process outlined in section III-C using the object detections $\{s_{i_k}\}$, robot fields of view and optimized poses $x_{1:n}$ from the new trajectory. These new samples are added to the existing samples for the POM for use in optimizing subsequent trajectories.

F. Optimization for Computational Efficiency

As Gaussian processes are computationally expensive, we employ a number of mechanisms to make the speed of this approach tractable. The first is limiting the samples used to compute μ and σ^2 in the POM evaluation to those within a radius of the query pose, as distant samples will have little influence on the POM output. We do this by storing the sample poses and values in a KD-tree [16] and constructing a POM at the beginning of each optimization cycle with only samples around the initial estimate for the object pose. Using a subset of the sample poses and outputs when evaluating the POM also reduces the computation time. Let r_s be the fraction of the samples to use and M be the full number of samples. We draw $N = r_s M$ random samples and adjust the POM evaluation by modifying (6) to sum instead over the N samples and compensate for subsampling by multiplying the full equation by r_s . We can also scale the computational power needed by modifying the number of samples N_s used to approximate the marginalization over object poses. Lastly, using only a subset of the object detections or optimizing over a window of only the most recent nodes in the trajectory will improve computation time.

IV. EXPERIMENTAL RESULTS

We present results from two sets of experiments¹ that evaluate POM-Localization's ability to 1) accurately estimate trajectories when knowledge of the distribution of movable objects is limited and the correctness of the POM is varied, and 2) ensure consistency in global localization over long time scales in environments with movable objects.

A. Accuracy Given Limited Object Distribution Knowledge

We aim to understand how quality of data used to form the POM impacts trajectory estimates when we assume low confidence in our map, which would occur when we have not collected sufficient past trajectories to converge to the true distribution. To do so, we measure accuracy using absolute trajectory error (ATE) on 10 sequences from the KITTI dataset [17] and compare against the output of LeGO-LOAM [18], a state-of-the-art lidar-inertial odometry and mapping algorithm. Though our approach supports 3D, we use a 2D projection for these experiments. We use the trajectory estimate of LeGO-LOAM for our odometry constraints. The object detections are derived from global poses of "car" instances in the SemanticKITTI dataset [19], which contains

¹The code for POM-Localization and our experiments is available at https://github.com/ut-amrl/pom_localization.

KITTI Sequence Number	00	02	03	04	05	06	07	08	09	10
LeGO-LOAM	27.12	1630.55	3.51	4.11	7.97	1.19	0.76	113.44	6.48	1.76
POM-Localization (0-100)	29.46	1630.55	3.51	4.11	7.97	0.80	0.76	113.44	6.48	1.76
POM-Localization (20-80)	22.93	1630.55	3.55	4.11	1.83	0.37	0.20	113.44	6.48	1.76
POM-Localization (50-50)	1.73	1322.19	1.39	4.06	1.95	0.08	0.03	106.86	0.80	0.33
POM-Localization (80-20)	1.50	1332.74	0.89	4.10	1.71	0.09	0.04	113.42	0.81	1.06
POM-Localization (100 ⁺ -0)	1.59	1333.48	0.82	4.04	0.18	0.04	0.04	97.58	0.23	0.17

Fig. 2: Absolute Trajectory Error (m) on KITTI Dataset Sequences

labels for lidar scan points in the KITTI benchmark. The object detections are created by transforming the global object pose into each of the frames in which the instance was observed.

To assess the impact of the correctness of the POM, we test five different configurations. In all cases, we generate the POM from a single simulated past trajectory, and thus have low confidence in our distribution to emulate the range of performance when bootstrapping the POM from one or few observed past trajectories. The initial step in creating the POM is generating poses where cars occurred in the past trajectory: $X\%$ of simulated car poses for the first four configurations are obtained by selecting poses from the current trajectory’s observed cars and adding a small amount of Gaussian noise (0.4 m), with the remaining $Y\%$ randomly placed in the environment. As X increases, the correctness of the POM increases. These are denoted “POM-Localization ($X-Y$)” in Fig. 2. In the last configuration, POM-Localization (100⁺-0), the POM is created from the same car poses that were used to generate the detections without any added noise, simulating perfect knowledge of the distribution and a deterministic environment in which movable objects are always at the same poses. In all cases, once the simulated car poses are selected, we generate the POM following the steps in section III-C, with the ground truth trajectory as the prior trajectory, the relative poses of nearby simulated cars with added noise as past object detections, and a fixed-radius region around the trajectory poses as the free space for obtaining off-detection samples.

Fig. 2 shows the ATE for LeGO-LOAM and our approach with the five configurations. When the POMs are completely misaligned with the observations as in POM-Localization (0-100), estimates are comparable to the LeGO-LOAM results, with all but one sequence having the same or lower error and the remaining estimate having only 8.6% higher ATE than the LeGO-LOAM result. As POM correctness increases, results generally improve, with the ATE of most trajectories generated using the POM created from perfect knowledge substantially lower than the ATE for LeGO-LOAM estimates. The only sequences for which our approach does not substantially improve upon LeGO-LOAM are 02, 04 and 08. In all of these sequences, there were trajectory segments that simultaneously had substantial odometry error and few or no object detections, and, as odometry is weighted highly relative to object detections, the drift was not corrected and persisted for the remainder of the trajectories.

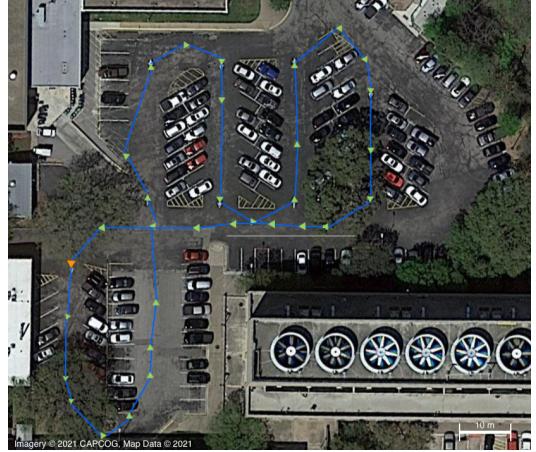


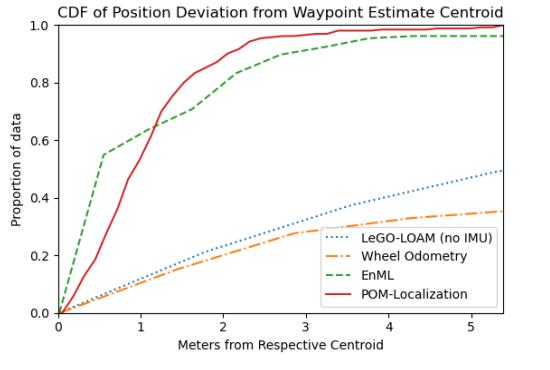
Fig. 3: Satellite view of Lot 53 on UT Austin campus where trajectories were collected. The trajectories started and ended at the orange marker and the approximate positions for the remaining 31 waypoints are designated in green.

B. Consistency Over Trajectories in Changing Environments

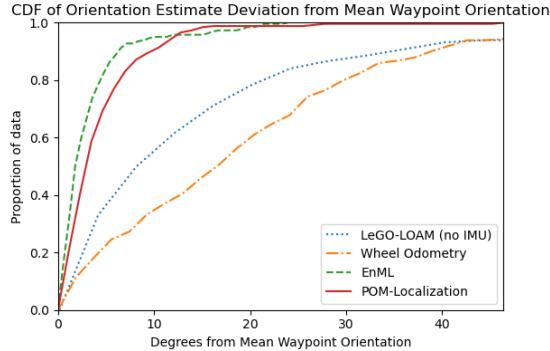
To understand our approach’s ability to produce consistent localization estimates over long time scales in changing environments, we collected trajectories over eight sessions in a UT Austin campus parking lot using a Clearpath Jackal with a Velodyne VLP-16 Lidar. In each trajectory, the robot visited 32 waypoints consistent across all trajectories, while the segments between the waypoints varied. Fig. 3 shows the parking lot with waypoint positions and connections.

Data used for these experiments were collected in 2D. Odometry constraints are obtained from wheel odometry and object detections are derived from human-provided annotations of point clouds. The POM is generated from a manually-created map of parking spots. We simulate five possible parking configurations by randomly selecting a subset of the parking spots to be occupied and placing cars according to a normal distribution around each selected spot. Samples to form the POM are created by simulating trajectories through each of the parking configurations, transforming the object poses to the frame of each node, and adding Gaussian noise to simulate noisy detection in past trajectories. Off-detection sample poses are drawn from within a radius around nodes in each simulated trajectory.

We assess performance by measuring consistency of waypoint estimates over the eight trajectories. Position estimate consistency is evaluated by calculating the centroid of all estimates for each waypoint across all trajectories and finding the distance of each estimate from the centroid. Similarly,



(a) CDF of position estimate deviation.

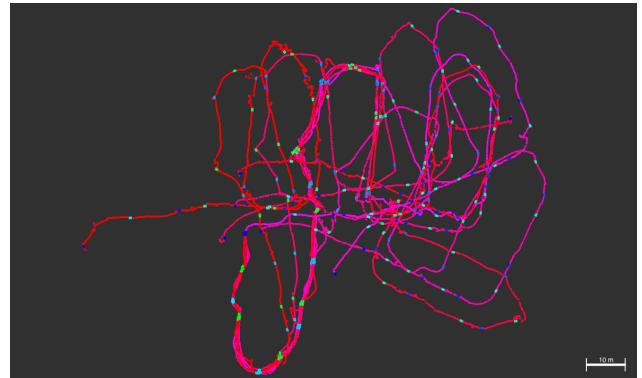


(b) CDF of orientation estimate deviation.

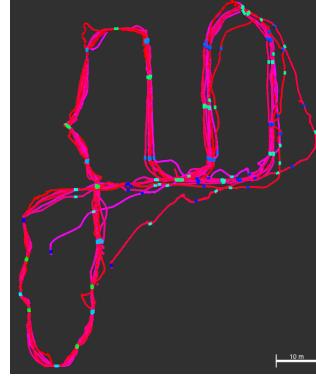
Fig. 4: Position and orientation consistency across approaches. An optimal algorithm would quickly rise to 1.

the orientation consistency is measured by computing the mean orientation estimate for each waypoint and the distance of each estimate from the corresponding mean orientation. We show the results of POM-Localization with estimates from wheel odometry, LeGO-LOAM [18] without use of an inertial measurement unit (IMU), and EnML [5]. It should be noted that the map and point clouds used by EnML are higher fidelity than the object detections and POM used by POM-Localization.

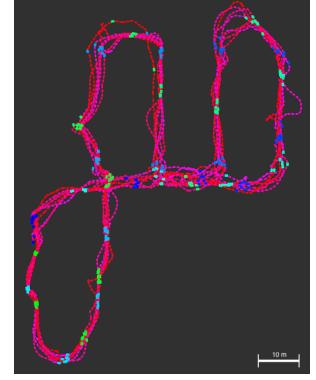
Fig. 4 shows the cumulative distribution functions (CDFs) for the position and orientation estimate consistency. For both position and orientation, while POM-Localization significantly outperforms raw wheel odometry and LeGO-LOAM, EnML has slightly better average-case performance. This is expected, given the difference in information sources for the two algorithms. However, POM-Localization has better or comparable worst-case performance: a greater portion of the orientation estimates are within 15 degrees of the mean for POM-Localization as compared to EnML and no position estimate from POM-Localization deviates more than 5.5 meters from its waypoint centroid. The trajectories and waypoints estimated by EnML, LeGO-LOAM, and POM-Localization are shown in Fig. 5. These plots support the conclusions drawn from Fig. 4; the estimates from LeGO-LOAM drift drastically after the initial segment of the trajectory, while POM-Localization results in slightly greater spread for the waypoint estimates than EnML, but unlike EnML, has no trajectory estimates that diverge significantly.



(a) LeGO-LOAM



(b) EnML



(c) POM-Localization

Fig. 5: Plot of trajectories through Lot 53 as estimated by the approaches with highlighted blue/green waypoints. Performance of an approach is good when all estimates for a given waypoint are colocated.

V. CONCLUSION AND FUTURE WORK

This paper introduces probabilistic object maps to model the distribution of movable objects in an environment. We also introduced POM-Localization to incorporate object detections and corresponding POMs to achieve globally consistent localization in changing environments.

There are a number of interesting areas for future exploration. Though our enhancements improved speed, investigation of other approximations may further reduce computation time. Future improvements could also include modifying our model to work with monocular visual detectors or integrating the observation factors of POM-Localization with a method that uses short-term features, thus gaining benefits of both approaches. Finally, POMs could be applied to other problems such as navigation to avoid likely occupied areas or finding objects in home environments.

ACKNOWLEDGMENT

This work has taken place in the Autonomous Mobile Robotics Laboratory (AMRL) at UT Austin. AMRL research is supported in part by NSF (CAREER-2046955, IIS-1954778, SHF-2006404), ARO (W911NF-19-2-0333), DARPA (HR001120C0031), Amazon, JP Morgan, and Northrop Grumman Mission Systems.

REFERENCES

- [1] S. Zhao, Z. Fang, H. Li, and S. Scherer, “A robust laser-inertial odometry and mapping method for large-scale highway environments,” in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1285–1292.
- [2] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard, “Dynamic pose graph SLAM: Long-term mapping in low dynamic environments,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 1871–1878.
- [3] D. M. Rosen, J. Mason, and J. J. Leonard, “Towards lifelong feature-based mapping in semi-static environments,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 1063–1070.
- [4] F. Nobre, C. Heckman, P. Ozog, R. W. Wolcott, and J. M. Walls, “Online probabilistic change detection in feature-based maps,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 3661–3668.
- [5] J. Biswas and M. M. Veloso, “Episodic non-markov localization,” *Robotics and Autonomous Systems*, vol. 87, pp. 162–176, 2017.
- [6] C. Stachniss and W. Burgard, “Mobile robot mapping and localization in non-static environments,” in *AAAI*, 2005, pp. 1324–1329.
- [7] T. Krajník, J. P. Fentanes, J. M. Santos, and T. Duckett, “FrEMEn: Frequency map enhancement for long-term mobile robot autonomy in changing environments,” *IEEE Transactions on Robotics*, vol. 33, no. 4, pp. 964–977, 2017.
- [8] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, “Lifelong localization in changing environments,” *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1662–1678, 2013.
- [9] S. Yang and S. Scherer, “CubeSLAM: Monocular 3-D object SLAM,” *IEEE Transactions on Robotics*, vol. 35, no. 4, pp. 925–938, 2019.
- [10] S. L. Bowman, N. Atanasov, K. Daniilidis, and G. J. Pappas, “Probabilistic data association for semantic SLAM,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 1722–1729.
- [11] K. Doherty, D. Fourie, and J. Leonard, “Multimodal semantic SLAM with probabilistic data association,” in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 2419–2425.
- [12] K. J. Doherty, D. P. Baxter, E. Schneeweiss, and J. J. Leonard, “Probabilistic data association via mixture models for robust semantic SLAM,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 1098–1104.
- [13] F. Ramos and L. Ott, “Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent,” *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016.
- [14] S. T. O’Callaghan and F. T. Ramos, “Gaussian process occupancy maps,” *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012.
- [15] C. Bishop, *Pattern Recognition and Machine Learning*. Springer, January 2006. [Online]. Available: <https://www.microsoft.com/en-us/research/publication/pattern-recognition-machine-learning/>
- [16] J. L. Bentley, “Multidimensional binary search trees used for associative searching,” *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.
- [17] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2012, pp. 3354–3361.
- [18] T. Shan and B. Englot, “LeGO-LOAM: Lightweight and ground-optimized lidar odometry and mapping on variable terrain,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.
- [19] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, “SemanticKITTI: A dataset for semantic scene understanding of lidar sequences,” in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019, pp. 9296–9306.