POV-SLAM: Supplementary Material

Contents

1 Derivation of the Measurement ELBO (E-Step)

In this section, we provide a more detailed derivation of the evidence lower bound (ELBO) for the Gaussian-Uniform measurement likelihood of potentially-changing objects, presented in Section IV-A of the main paper. Consider the following generative process for a landmark point, $\mathbf{l}^W$, from an object, $\mathbf{O}$. At frame T and EM iteration n, we obtain the object's updated Beta consistency model, $\text{Beta}(\alpha, \beta)$, according to the works of Qian *et al.*, with respect to the previous $(n-1)$ EM iteration's object pose estimate, $\mathbf{T}^{OW}$, robot pose estimate, $\mathbf{T}^{CW}$, and the current frame point depth measurement, $\mathbf{d}_T^C$. This Beta distribution characterizes the object's confidence score distribution. The subscripts in the variables are dropped for clarity. The object's true binary consistency, $\pi \in \{0, 1\}$, can be considered as a sample from a Bernoulli distribution parametrized by the probability v, where v is sampled from the Beta consistency model:

$$\begin{aligned} v &\sim \text{Beta}(\alpha, \beta) \\ \pi &\sim \text{Bernoulli}(v) \end{aligned} \tag{1}$$

Assuming that unchanged objects will follow a zero-mean, isotropic Gaussian measurement model and moved objects can be anywhere, we define the measurement residual to be the point-wise difference:

$$\mathbf{e}_T = \mathbf{T}^{CW-1}\mathbf{d}_T^C - \mathbf{l}^W, \tag{2}$$

We can then write a Gaussian-Uniform measurement model weighted by the sampled object consistency, π:

$$p(\mathbf{e}_T \mid \mathbf{T}^{CW}, \mathbf{l}^W, \pi) = \mathcal{N}(\mathbf{e}_T \mid \mathbf{0}, \sigma^2\mathbf{I})^{\pi}\mathcal{U}(\|\mathbf{e}_T\|_2 \mid 0, e_{\max})^{1-\pi} \tag{3}$$

The measurement model is now conditioned on two dependent latent variables, $\boldsymbol{\omega} = \{\pi, v\}$, and thus is difficult to maximize directly. However, we can apply the Mean-field Approximation and assume the two latent variables, $\boldsymbol{\omega}$, are fully independent, $p(\pi, v) \simeq q(\pi, v) = q(\pi)q(v)$. We can then write a factorized lower bound, $\mathcal{L}$, for the joint log likelihood, $\log p(\mathbf{e}_T \mid \mathbf{T}^{CW}, \mathbf{l}^W, \alpha, \beta)$:

$$\begin{aligned} &\mathcal{L}(\boldsymbol{\omega}, \mathbf{T}^{CW}, \mathbf{l}^W) = \mathbb{E}_{q(\boldsymbol{\omega})}\left[\log \frac{p(\mathbf{e}_T, \boldsymbol{\omega} \mid \mathbf{T}^{CW}, \mathbf{l}^W, \alpha, \beta)}{q(\boldsymbol{\omega})}\right] \\ &= \int q(\boldsymbol{\omega}) \log \frac{p(\mathbf{e}_T, \boldsymbol{\omega} \mid \mathbf{T}^{CW}, \mathbf{l}^W, \alpha, \beta)}{q(\boldsymbol{\omega})} d\boldsymbol{\omega} \\ &= \int q(\omega_j) \log \frac{\exp(\langle \log p(\mathbf{e}_T, \boldsymbol{\omega} \mid \mathbf{T}^{CW}, \mathbf{l}^W, \alpha, \beta)\rangle_{k\neq j})}{q(\omega_j)} d\omega_j \\ &\quad - \sum_{k\neq j} \int q(\omega_j) \log q(\omega_j) d\omega_j \\ &= -KL(q(\omega_j))||\tilde{p}_{k\neq j}) + \mathcal{H}(\omega_{k\neq j}) + const \end{aligned} \tag{4}$$

Here, $\langle\cdot\rangle_k$ is the expectation over the latent variable, ω_k. Since KL-divergence is non-negative, the ELBO is maximized when the first term $KL(\cdot||\cdot) = 0$. Thus, we want to find $q(\omega_j) = \frac{1}{Z}\tilde{p}_{k\neq j} = \frac{1}{Z}\exp(\langle p(\mathbf{e}_T, \boldsymbol{\omega} \mid \mathbf{T}^{CW}, \mathbf{l}^W, \alpha, \beta)\rangle_{k\neq j})$ for all $j \in \{0, 1\}$. Note that the full log joint likelihood in Eq. 4 is:

$$\begin{aligned} &\log p(\mathbf{e}_T, \boldsymbol{\omega} = \{\pi, v\} \mid \mathbf{T}^{CW}, \mathbf{l}^W, \alpha, \beta) \\ &\quad = \log p(\mathbf{e}_T \mid \mathbf{T}^{CW}, \mathbf{l}^W, \pi) + \log p(\pi \mid v) + \log p(v \mid \alpha, \beta) \\ &\quad = \log \mathcal{N}(\mathbf{e}_T \mid \mathbf{0}, \sigma^2\mathbf{I})^{\pi}\mathcal{U}(\|\mathbf{e}_T\|_2 \mid 0, e_{\max})^{1-\pi} \\ &\qquad + \pi \log v + (1-\pi)\log(1-v) \\ &\qquad + \log \text{Beta}(v \mid \alpha, \beta) \\ &\quad = \pi[\log v + \log \mathcal{N}(\mathbf{e}_T \mid \mathbf{0}, \sigma^2\mathbf{I})] \\ &\qquad + (1-\pi)[\log(1-v) + \log \mathcal{U}(\|\mathbf{e}_T\|_2 \mid 0, e_{\max})] \\ &\qquad + \log \text{Beta}(v \mid \alpha, \beta) \end{aligned} \tag{5}$$

Under the Mean-field Approximation, the optimal distribution, $q(\cdot)$, which maximizes the ELBO is:

$$\log q(\omega_j) = \mathbb{E}_{\omega_k, k\neq j}[\log p(\mathbf{e}_T, \boldsymbol{\omega} = \{\pi, v\} \mid \mathbf{T}^{CW}, \mathbf{l}^W, \alpha, \beta)] \tag{6}$$

Therefore, we can find $q(\pi)$ and $q(v)$ as follows:

$$\begin{aligned}\log q(\pi) =&\mathbb{E}_v[\log p(\mathbf{e}_T, \boldsymbol{\omega} \mid \mathbf{T}^{CW}, \mathbf{l}^W, \alpha, \beta)] \\ =&\pi\big[\mathbb{E}[\log v] + \log \mathcal{N}(\mathbf{e}_T \mid \mathbf{0}, \sigma^2\mathbf{I})\big] \\ &+ (1-\pi)\big[\mathbb{E}[\log(1-v)] + \log \mathcal{U}(\|\mathbf{e}_T\|_2 \mid 0, e_{\max})\big] \\ &+ \underbrace{\mathbb{E}[\log \mathrm{Beta}(v \mid \alpha, \beta)]}_{const}\end{aligned} \tag{7}$$

$$\begin{aligned}\log q(v) =&\mathbb{E}_\pi[\log p(\mathbf{e}_T, \boldsymbol{\omega} \mid \mathbf{T}^{CW}, \mathbf{l}^W, \alpha, \beta)] \\ =&\mathbb{E}[\pi]\big[\log v + \log \mathcal{N}(\mathbf{e}_T \mid \mathbf{0}, \sigma^2\mathbf{I})\big] \\ &+ \mathbb{E}[1-\pi]\big[\log(1-v) + \log \mathcal{U}(\|\mathbf{e}_T\|_2 \mid 0, e_{\max})\big] \\ &+ \log \mathrm{Beta}(v \mid \alpha, \beta) \\ =&\mathbb{E}[\pi] \log v + \mathbb{E}[1-\pi] \log(1-v) + \log \mathrm{Beta}(v \mid \alpha, \beta) + const\end{aligned} \tag{8}$$

Note that we can compute the expectations for v using the property of Beta distributions:

$$\begin{aligned}\mathbb{E}[\log v] &= \psi(\alpha) - \psi(\alpha+\beta) \\ \mathbb{E}[\log(1-v)] &= \psi(\beta) - \psi(\alpha+\beta)\end{aligned} \tag{9}$$

where $\psi(\cdot)$ is the digamma function. Moreover, we have:

$$\begin{aligned}\mathbb{E}[\pi] = q(\pi=1) &\propto \exp\{\mathbb{E}[\log v] + \log \mathcal{N}(\mathbf{e}_T \mid \mathbf{0}, \sigma^2\mathbf{I})\} := \rho_{\text{in}} \\ \mathbb{E}[1-\pi] = q(\pi=0) &\propto \exp\{\mathbb{E}[\log(1-v)] + \log \mathcal{U}(\|\mathbf{e}_T\|_2 \mid 0, e_{\max})\} := \rho_{\text{out}}\end{aligned} \tag{10}$$

Normalizing the expressions, we obtain the expectations for the binary object consistency, π:

$$\begin{aligned}\eta &:= \frac{1}{\rho_{\text{in}} + \rho_{\text{out}}} \\ \mathbb{E}[\pi] &= \eta \exp\{\mathbb{E}[\log v] + \log \mathcal{N}(\mathbf{e}_T \mid \mathbf{0}, \sigma^2\mathbf{I})\} \\ \mathbb{E}[1-\pi] &= \eta \exp\{\mathbb{E}[\log(1-v)] + \log \mathcal{U}(\|\mathbf{e}_T\|_2 \mid 0, e_{\max})\}\end{aligned} \tag{11}$$

An interesting observation is that the approximation $q(v)$ is also a Beta distribution:

$$\begin{aligned}\log q(v) &= \mathbb{E}[\pi] \log v + \mathbb{E}[1-\pi] \log(1-v) + \log \mathrm{Beta}(v \mid \alpha, \beta) + const \\ &= \mathbb{E}[\pi] \log v + \mathbb{E}[1-\pi] \log(1-v) + (\alpha-1)\log v + (\beta-1)\log(1-v) + const \\ &= (\alpha + \mathbb{E}[\pi] - 1)\log v + (\beta + \mathbb{E}[1-\pi] - 1)\log(1-v) + const \\ &= \log \mathrm{Beta}(v \mid \underbrace{\alpha + \mathbb{E}[\pi]}_{\tilde{\alpha}}, \underbrace{\beta + \mathbb{E}[1-\pi]}_{\tilde{\beta}})\end{aligned} \tag{12}$$

We can now compute the ELBO for the joint likelihood:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\omega}, \mathbf{T}^{CW}, \mathbf{l}^W) &= \mathbb{E}_v \mathbb{E}_\pi \log \frac{p(\mathbf{e}_T, v, \pi \mid \mathbf{T}^{CW}, \mathbf{l}^W, \alpha, \beta)}{q(v, \pi)} \\ &= \mathbb{E}_v \mathbb{E}_\pi \log p(\mathbf{e}_T \mid \mathbf{T}^{CW}, \mathbf{l}^W, \pi) p(\pi \mid v) p(v \mid \alpha, \beta) - \mathbb{E}_v \mathbb{E}_\pi \log q(z) q(\pi) \\ &= \underbrace{\mathbb{E}_\pi \log p(\mathbf{e}_T \mid \mathbf{T}^{CW}, \mathbf{l}^W, \pi)}_{\text{only care about this in M-step}} + \mathbb{E}_v \mathbb{E}_\pi \log p(\pi \mid v) p(v \mid \alpha, \beta) + \mathcal{H}(q(v)) + \mathcal{H}(q(\pi)) \\ &= \mathbb{E}_\pi \log p(\mathbf{e}_T \mid \mathbf{T}^{CW}, \mathbf{l}^W, \pi) + const \\ &= \mathbb{E}_\pi \log \mathcal{N}(\mathbf{e}_T \mid \mathbf{0}, \sigma^2\mathbf{I})^\pi \mathcal{U}(\|\mathbf{e}_T\|_2 \mid 0, e_{\max})^{1-\pi} + const \\ &= \mathbb{E}[\pi] \log \mathcal{N}(\mathbf{e}_T \mid \mathbf{0}, \sigma^2\mathbf{I}) + \mathbb{E}[1-\pi] \log \mathcal{U}(\|\mathbf{e}_T\|_2 \mid 0, e_{\max}) + const\end{aligned} \tag{13}$$

As discussed in the main paper, this new cost has a similar log Gaussian-log Uniform mixture form comparing to the single-value estimate (Equation 6 of main paper), but with different mixture weights, $\mathbb{E}[\pi]$ and $\mathbb{E}[1-\pi]$ from Eq. 11. The new weights incorporate both the full information of the Beta consistency model and the likelihood of the current measurement under the two scenarios, as opposed to just the expectation of the Beta object consistency model, $\mathbb{E}[v]$.

2 Algorithm Summary

Algorithm (1) outlines how the proposed POV-SLAM pipeline processes a new frame, as discussed in Section III and Section IV of the main paper. The algorithm consists of the following key steps:

2.1 Inputs to Algorithm

The input to the algorithm consists of a new RGB-D frame $\mathbf{F}_t$ with associated semantic segmentation, the current active object library $\mathcal{O}$, and the current estimated robot trajectory $\mathcal{T}^{CW}$ over the optimization window. We note that for each object $\mathbf{O}_i$ in $\mathcal{O}$, a Gaussian-Beta state model $p(l_i, v_i)$ is maintained to store its current change and consistency estimates.

which is independent from its state model propagated over the EM iterations. The advantage of int

2.2 Data Pre-processing

The input semantic RGB-D frame $\mathbf{F}_t$ is first segmented into 3D clusters based on semantics and geometry using the *extractObservations()* helper function. Then, the observations are associated to the existing objects in $\mathcal{O}$ using a greedy Hungarian matching algorithm in the *objectLevelAssociation()* helper function. More details of the object detection and association process are available in Qian *et al.* 2022.

Then, we leverage the original ORB-SLAM3's RGB-D front-end to obtain a visual odometry measurement $\mathbf{T}^C_{t-1,t}$, which is used as a prior to initialize the factor graph. In practice, we find that this improves the stability of the optimization.

Finally, for each observed object in the current frame, we initialize a new Gaussian-Beta state model $p_{EM}(l, v)$ to *propagate* (according to Qian *et al.* 2022) over the EM iterations. This is independent from the object's internal state model $p(l, v)$, which is only updated once at the end of the EM iterations using the final robot pose. There are two advantages of maintaining two state models for each object. First, the system can respond to changes faster when revisiting an old location as $p(l, v)$ might have been saturated from past frames. Second, the system will be more stable as inaccurate robot and object state estimates will not lead to immediate divergence since $p(l, v)$ is only propagated once per frame instead of in every EM iteration.

2.3 Variational EM Execution

Using the processed input data, we run the Variational EM algorithm presented in the main paper. In the E-step of each iteration, we first propagate the current frame state model $p_{EM}(l, v)$ for each observed object based on the new robot pose estimate from the last iteration to obtain the new consistency model $\text{Beta}(\alpha, \beta)$. Then we construct the factor graph according to Eq. 17 and Eq. 18 in the main paper. In the M-step, we solve the factor graph to obtain the updated robot and object states.

2.4 Output Processing

Based on the result of the final EM iteration, we can categorize the observed objects as *unchanged* (Gaussian mode active, $m = 1$) or *moved* (uniform mode active, $m = 1$). For the *unchanged* objects, their internal states $p(l, v)$ are propagated based on the final robot pose, and the new depth measurements are fused into their TSDF models. For the *moved* objects, their states are propagated using a large penalty, and their consistency scores $\mathbb{E}[v]$ are checked against a pruning a threshold. All objects fall below the threshold are removed and possibly reconstructed in future frames.

Moreover, observed objects not associated to any existing objects are spawned as new objects, and objects in the camera field of view but not observed are penalized. All reliable objects are used to construct the up-to-date map.

Algorithm 1 Processing of One Frame in POV-SLAM

Require: New RGB-D frame $\mathbf{F}_t$, current object library $\mathcal{O}$, robot trajectory estimate $\mathcal{T}^{CW}$

▷ Extract 3D Detections
$\mathcal{Y}_t \leftarrow$ extractObservations($\mathbf{F}_t$)
$\{(\mathbf{O}_i, \mathbf{Y}_{t,j})\} \leftarrow$ objectLevelAssociation($\mathcal{Y}_t, \mathcal{O}$)
▷ Estimate a visual odometry measurement using ORB-SLAM3 RGB-D
$\mathbf{T}^C_{t-1,t} \leftarrow$ ORB-SLAM3($\mathbf{F}_t$)
▷ Initialize for EM
for each `pair` in $\{(\mathbf{O}_i, \mathbf{Y}_{t,j})\}$ **do**
  Initialize EM state model for $\mathbf{O}_i$: $p_{EM}(l_i, v_i) \leftarrow$ Gaussian$(0, \sigma\mathbf{I})$Beta$(1, 1)$
end for
▷ Run Variational EM
for $n = 1$ `to` *max_iter* **do**
  for each `pair` in $\{(\mathbf{O}_i, \mathbf{Y}_{t,j})\}$ **do**
    $p_{EM}(l_i, v_i) \leftarrow$ propagateState$(\mathbf{Y}_{t,j}, \mathbf{O}_i, \mathbf{T}^{CW})$ using work of Qian *et al.* 2022
    Compute approx. ELBO: $\tilde{\mathcal{L}}(v, \pi, \mathbf{T}^{CW}, \mathbf{l}^W_i)$ according to Equation (17) in main paper
  end for
  Construct factor graph and solve according to Equation (18) in main paper
  $\mathcal{O}, \mathcal{T}^{CW} \leftarrow \operatorname{argmax} \log p(\mathcal{O}, \mathcal{T}^{CW}, \mathcal{Y}_t)$
end for
▷ Update observed objects
for each `pair` in $\{(\mathbf{O}_i, \mathbf{Y}_{t,j})\}$ **do**
  if decision($\mathbf{O}_i$) == `unchanged` **then**
    $p(l_i, v_i) \leftarrow$ propagateState$(\mathbf{Y}_{t,j}, \mathbf{O}_i, \mathbf{T}^{CW})$
    $\mathbf{O}_i \leftarrow$ integrate$(\mathbf{Y}_{t,j})$
  else if decision($\mathbf{O}_i$) == `moved` **then**
    $p(l_i, v_i) \leftarrow$ propagateState$(large_error, \mathbf{O}_i)$
    Discard $\mathbf{Y}_{t,j}$
    if consistencyExpectation($\mathbf{O}_i$) $\leqslant \theta_{\text{consist}}$ **then**
      $\mathcal{O} \leftarrow$ prune($\mathbf{O}_i$)
    end if
  end if
end for
for $\mathbf{Y}_{t,j}$ in $\mathcal{Y}_t$ **do**
  if notAssociated($\mathbf{Y}_{t,j}$) **then**
    $\mathcal{O} \leftarrow$ newObject($\mathbf{Y}_{t,j}$)
  end if
end for
for $\mathbf{O}_i$ in $\mathcal{O}$ **do**
  if expectedNotObserved($\mathbf{O}_i$) **then**
    $p(l_i, v_i) \leftarrow$ propagateState$(large_error, \mathbf{O}_i)$
  end if
end for
return $\mathcal{O}$, $\mathcal{T}^{CW}$

3 2D Simulation and Ablation Studies

In this section, we examine more example scenarios using the 2D simulation. We start with the system evolution of the scenario discussed in the main paper. We then show a scenario with dynamic objects, as well as a typical failure case where the system converges to an incorrect but more likely configuration, due to object symmetry. We also conduct ablation studies to illustrate the importance of certain design decisions.

3.1 System Evolution of Scenario in Main Paper

The main paper focuses on the per-iteration behavior of the EM algorithm at a single frame. Here, we take a look at how the system behaves across frames as the robot navigates in the simulator. Figure 1 shows the ground truth and estimated system state over the first four frames. In Frame 1, all objects are spawned with a consistency expectation of 0.5. After the first round of EM iterations, the system is able to successfully distinguish the static objects from the moved ones. Object consistency scores are also updated accordingly. In the third frame, the system decides that sufficient evidence has been observed, and all moved objects are relocalized to their correct locations. From then on, all object measurements are accepted in succeeding frames. Figure 2 illustrates the evolution of the object consistency expectation over frames. Note that the moved objects experience an initial drop in consistency (rejection phase), but their scores eventually increase after relocalization (acceptance phase).

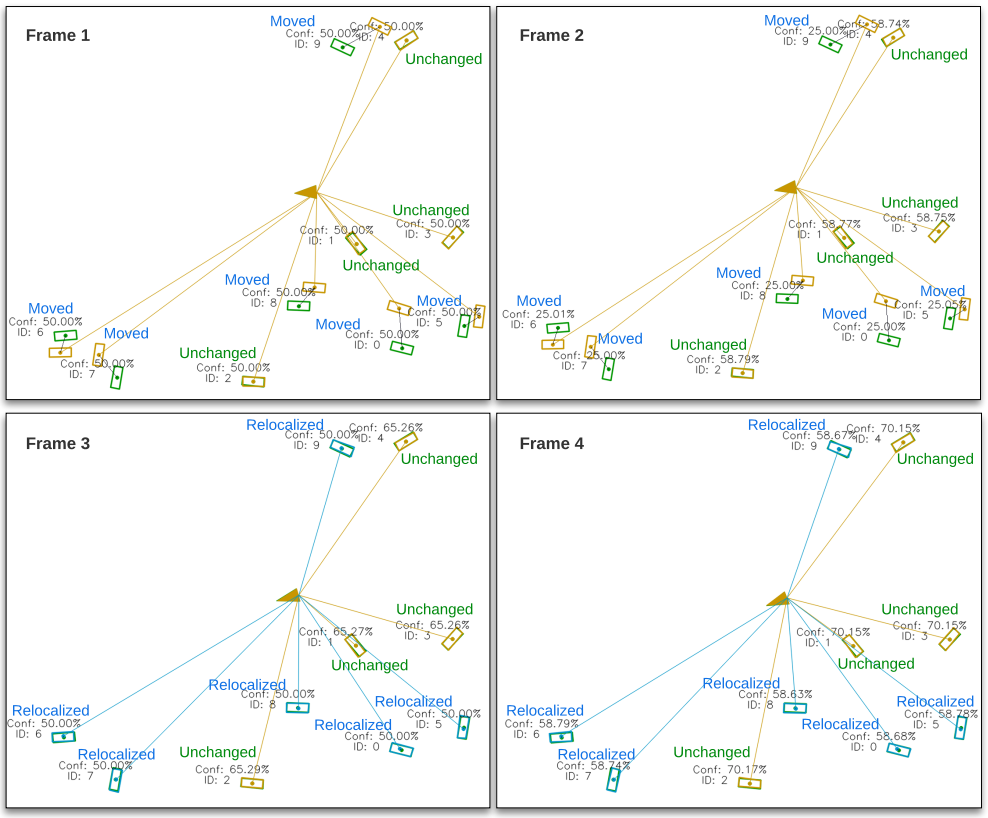


Figure 1: The setup of the 2D semi-static simulation with six moved objects and four unchanged objects over the first four frames of optimization. **Rectangle**: object, **triangle**: robot, **green**: ground truth, **yellow**: estimate.

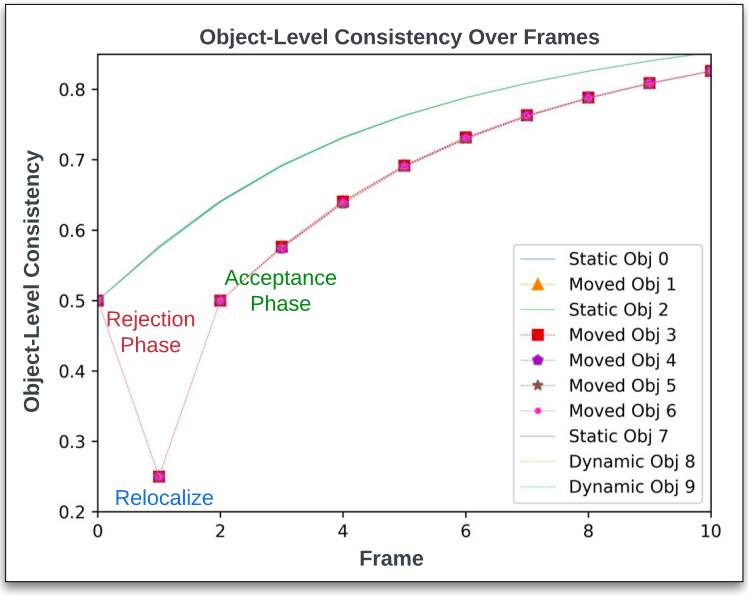


Figure 2: Object consistency expectation over frames. Moved objects are first rejected and relocalized in the first 3 frames. All objects are accepted henceforth.

3.2 Dynamic Object Handling

Figure 3 illustrates a scenario where our system tracks both semi-static changes and highly dynamic objects in a unified framework. The scenario contains two unchanged static objects, three moved static objects, and six dynamic objects. All dynamic objects, as well as the robot, move upwards in the scene at a speed of 1 m/s. All dynamic object estimates are initialized with a zero-velocity initial state and propagated separately using a linear Kalman filter. The system solves a temporal factor graph shown in Figure 3 of the main paper to optimize robot and all object states together at every EM iteration. As in the previous section, the three moved objects are successfully identified and relocalized in the first three frames. Figure 4a illustrates the estimation errors of the object centroids and Figure 4b illustrates the velocity errors. We can see that the system is able to track the dynamic objects with high accuracy even under noisy measurements and semi-static scene changes.

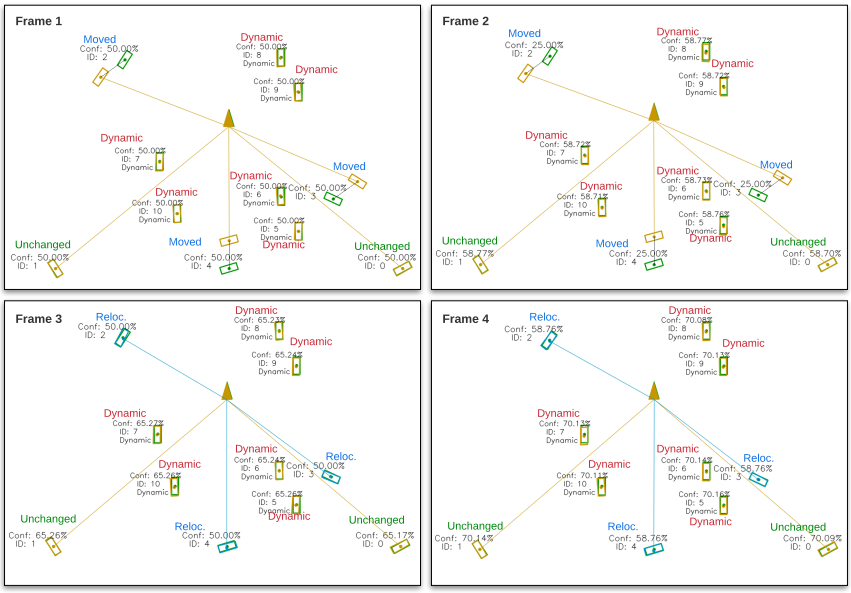
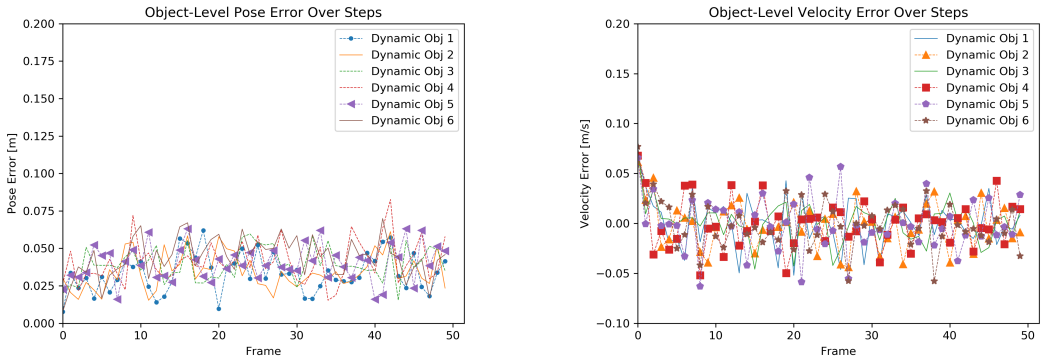


Figure 3: A scenario with unchanged static, moved static, and dynamic objects. The robot and dynamic objects all move upwards in the scene at 1 m/s. The system is robust enough to relocalize the moved objects and track all dynamic objects in a unified framework. **Rectangle**: object, **triangle**: robot, **green**: ground truth, **yellow**: estimate.



(a) Pose error of the dynamic objects over frames, measured at the object centroids.

(b) Velocity error of the dynamic objects over frames.

Figure 4: Pose error and velocity estimates over frames for dynamic objects moving at 1 m/s.

3.3 Adversarial Scenario: Symmetric Scene Change

This section illustrates a typical failure case of the system where several objects change in the same way, creating ambiguous solutions. Figure 5 illustrates a scenario with four unchanged objects and six moved objects. However, all moved objects are shifted in the up-right direction by the same amount. Since there are more moved objects than unchanged ones, the system actually converges to the incorrect state where it believes the unchanged objects have moved. Without prior robot pose information, however, the converged state does exhibit higher measurement likelihood, and thus it is not possible to distinguish the correct solution from a probabilistic point of view. Figure 6a shows the robot pose error and Figure 6b shows the object consistency estimates over the EM iterations at the first frame. In practice, we can use additional sensors, such as wheel odometry and an inertial measurement unit (IMU), to regularize the solution.

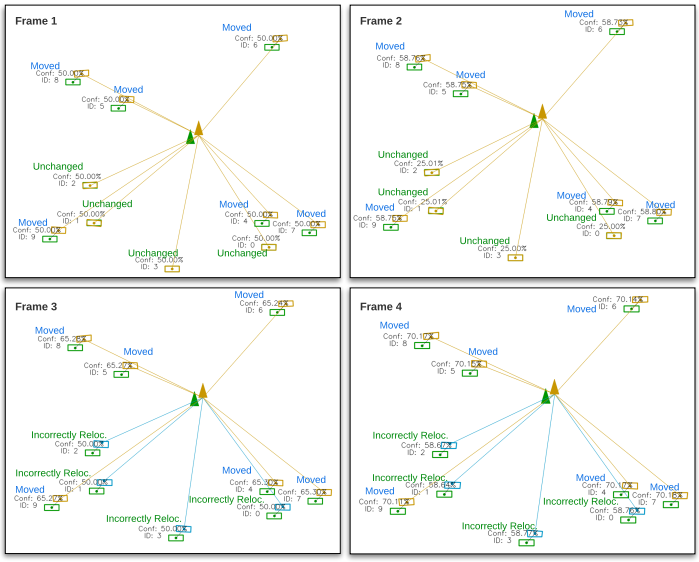
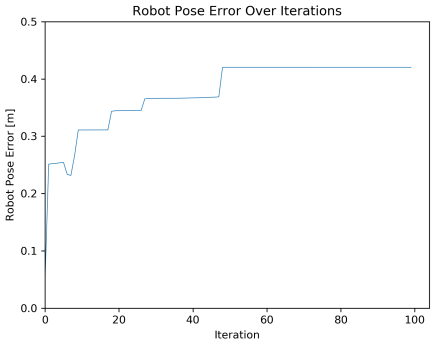
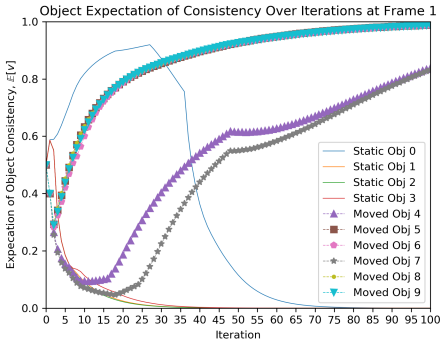


Figure 5: An adversarial scenario where a dominating number of objects move in the same direction. The system is not able to converge to the correct solution, as the adversarial solution exhibits higher measurement likelihood. As can be seen, the supposedly unchanged objects are incorrectly relocalized against the moved ones. **Rectangle**: object, **triangle**: robot, **green**: ground truth, **yellow**: estimate.



(a) Robot pose error over the EM iterations at Frame 1. The system converges to an incorrect, but more likely solution due to symmetry.



(b) Object consistency expectations over the EM iterations at Frame 1. Moved objects are classified as unchanged while the actual unchanged objects are classified as moved, due to symmetry.

Figure 6: Robot pose error and object consistency estimates in an adversarial scenario.

3.4 Adversarial Scenario: Fully Dynamic Scene

Another adversarial scenario where the proposed method can fail is when all objects in the scene are moving. Figure 7 illustrates a scenario with ten objects all moving in random directions, at random speeds. As we can see, the system fails to converge to the correct configuration and many objects are reconstructed at incorrect locations. Ultimately, our optimization-based approach only converges to a minimum-cost state but there is no guarantee to the correctness as no static "anchor" is available in a fully dynamic scene. In practice, additional sensors such as IMUs and wheel odometry can be used to alleviate this issue.

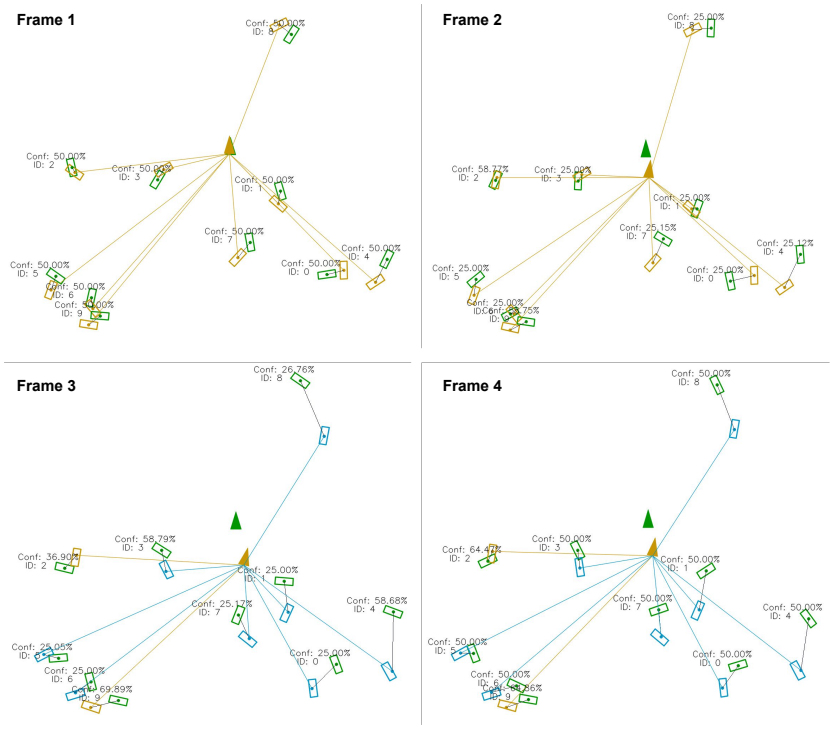
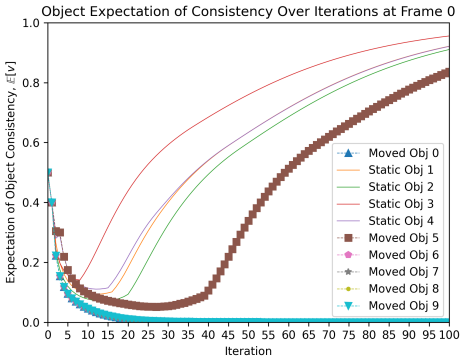


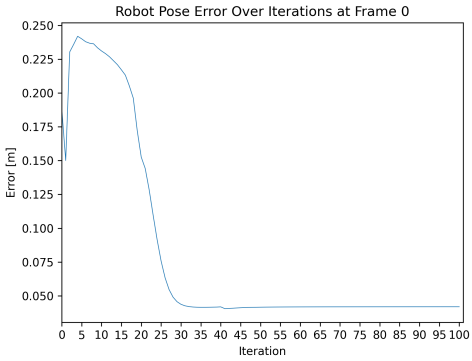
Figure 7: A scenario with fully dynamic objects. The objects all rotate and move at random velocities. The system fails as no static "anchor" can be used to correctly localize the robot. **Rectangle**: object, **triangle**: robot, **green**: ground truth, **yellow**: estimate.

3.5 Abalation Study: Full ELBO versus Max-Mixture Approximation

The full evidence lower bound (ELBO) overestimates the consistency of objects when there are not sufficient measurements, due to the use of the mean-field approximation. This section illustrates the importance of using max-mixture to guide the optimization. The following results are obtained from the scenario shown in the main paper. We compare Figure 8 and Figure 9. As can be seen in the plots, one of the moved objects is misclassified as unchanged when we use the full ELBO as the cost function, causing the robot pose to fall into a local minima. On the other hand, with the max-mixture approximation, all objects are classified correctly and the robot converges to the true state in just six iterations. This emphasizes the need to use the max-mixture, allowing the system to rely on objects which have higher measurement likelihoods and rejecting uncertain objects in the optimization. We should note that the decisions are revised at every gradient step so objects rejected previously could be accepted again when we have a more accurate robot pose estimate.

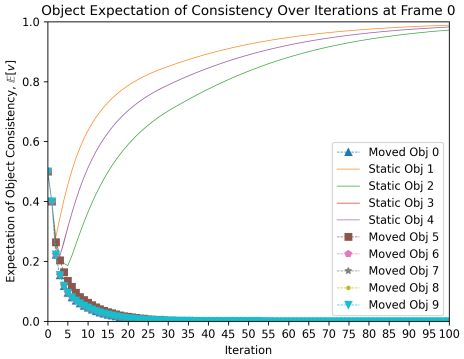


(a) Evolution of the object consistency expectations $\mathbb{E}(v)$ over the EM iterations at Frame 0. Without the max-mixture approximation, one moved object is misclassified due to ambiguity.

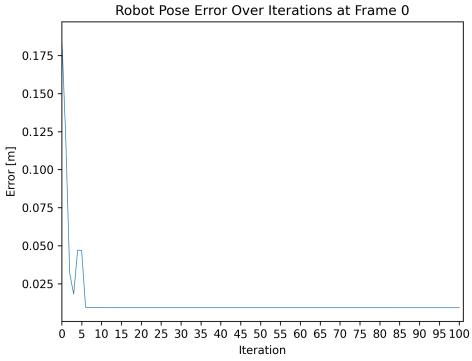


(b) Evolution of the robot pose error over the EM iterations at Frame 0. Without the max-mixture approximation, the system converges to a local minimum.

Figure 8: System using full ELBO cost.



(a) With max-mixture, all objects are classified correctly.

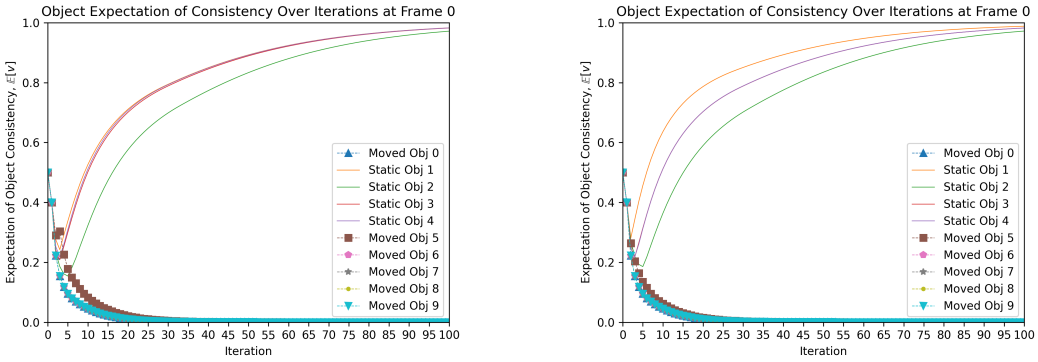


(b) With max-mixture, the system converges to the correct configuration and at a much faster rate.

Figure 9: System using max-mixture to guide optimization.

3.6 Ablation Study: ELBO versus Single Point Measurement Model

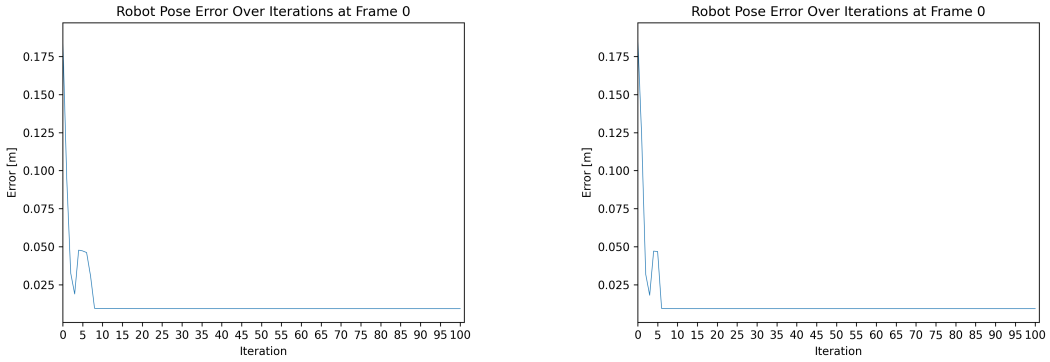
As discussed in the main paper, the mixture model in the derived ELBO is weighted by $\mathbb{E}[\pi]$, whereas the naive, single point measurement model (Equation 6 in the paper) is weighted by $\mathbb{E}[v]$. The following results are obtained from the scenario shown in the main paper. We first compare Figure 10a to Figure 10b. While in both cases the objects are classified correctly, the consistency expectation, $\mathbb{E}[v]$, converges slightly faster when using ELBO. Moreover, in Figure 11a and Figure 11b it can be seen that the robot pose converges to the correct solution in just six iterations instead of eight when using ELBO. The difference is more obvious when we use the full mixture models as the cost function. Comparing Figure 12a and Figure 8a, two objects are misclassifed when using the full naive measurement model whereas just one object got misclassified when using the full ELBO. Finally, comparing Figure 12b and Figure 8b, the robot pose converges much faster when using the full ELBO, although both converge to a local minimum by the end. Therefore, the ELBO improves the convergence behaviour of the system, as it provides a more accurate estimate for the underlying multimodal measurement likelihood.



(a) With the naive measurement model, all objects are classified correctly.

(b) With ELBO, all objects are classified correctly once again, but the consistency expectations converge faster.

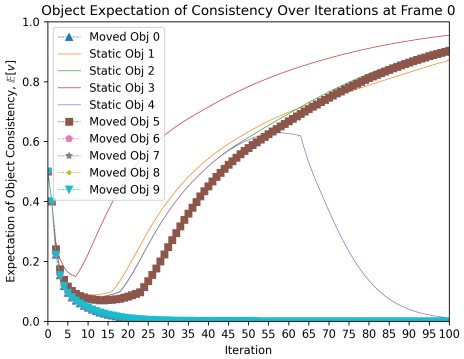
Figure 10: Object consistency, $\mathbb{E}[v]$, evolution over 100 EM iterations at Frame 0 using the naive single point measurement model and the derived ELBO. The max-mixture approximation is used in both cases.



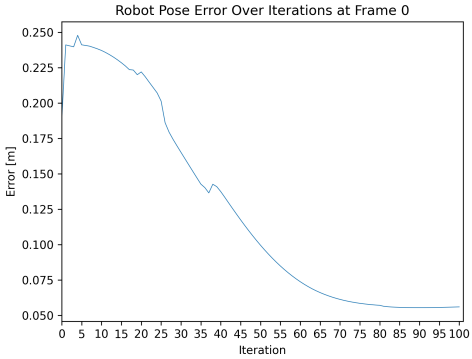
(a) With the naive measurement model, the system converges to the correct configuration in 8 iterations.

(b) With ELBO, the system converges to the correct configuration in 6 iterations.

Figure 11: Robot pose error over 100 EM iterations at Frame 0 using the naive single point measurement model and the derived ELBO. The max-mixture approximation is used in both cases.



(a) When using the full naive mixture measurement model, two objects are misclassified.

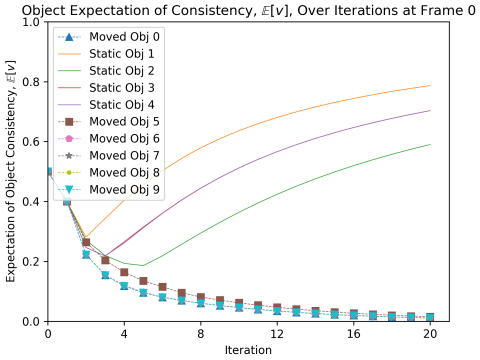


(b) When using the full naive mixture measurement model, the robot pose converges to a local minimum.

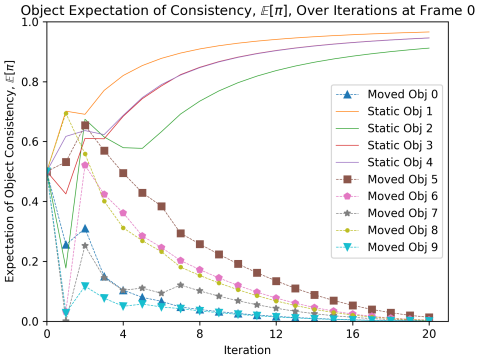
Figure 12: System convergence behavior over 100 EM iterations at Frame 0 using the naive single point measurement model and the derived ELBO. The max-mixture approximation is not used here.

3.7 Ablation Study: Mode Weight for Max-Mixture Approximation

Recall, in Equation 16 of the main paper, the max-mixture approximation compares the weighted measurement likelihood under the unchanged and moved scenarios. In this subsection, we study which weight should be used in the comparison. The following results are obtained from the scenario shown in the main paper. We begin two optimizations, one using $\mathbb{E}[\pi]$, the variational estimate of whether an object has changed, and one using $\mathbb{E}[v]$, the expectation of the Beta distribution, in the max-mixture approximation. Looking at Figure 13, $\mathbb{E}[v]$ is much smoother when compared to $\mathbb{E}[\pi]$. This is because $\mathbb{E}[v]$ is derived from the Bayesian update rule and is guaranteed to be smooth. On the other hand, $\mathbb{E}[\pi]$ tends to be overconfident in the object consistency due to the mean-field approximation, thus producing noisy estimates, especially in the first few iterations. To ensure a smooth gradient, we use $\mathbb{E}[v]$ as the weight when selecting which mode to use in optimization.



(a) Using $\mathbb{E}[v]$, the object-level confidence is much smoother at the beginning of the optimization.



(b) Using $\mathbb{E}[\pi]$, although the estimates converge to the true states faster, they are very noisy at the beginning when the Beta consistency models have high uncertainty.

Figure 13: Object-level evolution when using $\mathbb{E}[v]$ and $\mathbb{E}[\pi]$ in the optimization, over the first 20 EM iterations at Frame 0.

4 List of Parameters

We list the parameters used in this work. Symbols, descriptions and values are provided.

Table 1: Parameters used in the POV-SLAM framework.

Parameter	Description	BoxSim	Real-World Warehouse Dataset
Object Detection and Mapping Parameters	Refer to Qian *et. al* for details		
ν	voxel size	10 cm	10 cm
s	stationarity class	0: -, 1: box	0: pillar, roof, shelf, 1: box, pallet
λ_1	association weight for position difference	1	1
λ_2	association weight for orientation difference	1	1
λ_3	association s for semantic consistency	1000	1000
λ_{diff}	scaling factor for geometric change estimation	1.6	1.8
θ_{dist}	maximum centroid-distance for association	1.0 m	3 m
θ_{sim}	percent of outliers from ICP during association	10%	10%
θ_{cutoff}	cutoff cost for feasible association	3.6	3.6
θ_{vis}	threshold of points within camera FOV to label an unassociated object as *unobserved*	20%	30%
θ_{stat}	stationarity threshold for object pruning	0.4	0.3
θ_{depth}	cutoff threshold for depth information	8 m	10 m
v	initial stationarity expectation	semi-static: 0.67, dynamic: 0.67	semi-static: 0.67, dynamic: 0.67
v_{max}	max stationarity expectation	0.9999	0.9999
μ	initial geometric change expectation	0 cm	0 cm
σ	initial geometric change standard deviation	0.5 m	0.5 m
Δ_{max}	maximum geometric change cutoff	10 m	10 m
τ	measurement standard deviation	8 cm	15 cm
k	weights for stationarity class measurements	outlier measurements, dynamic-class: 3 inlier measurements, dynamic-class: 0 outlier measurements, static-class: 0 inlier measurements, static-class: 3	outlier measurements, dynamic-class: 3 inlier measurements, dynamic-class: 0 outlier measurements, static-class: 0 inlier measurements, static-class: 3
POV-SLAM Parameters			
T	factor graph window size	5	8
N	maximum EM iterations	20	15
H	ORB-SLAM3 steps between V-EM update	1	6
σ_{rigid}	factor graph rigidity stddev	0.01	0.1
σ_{prior}	factor graph pose prior stddev	0.1	0.1
$\sigma_{\text{key-pt}}$	factor graph measurement stddev	0.05	0.15
σ_{pose}	factor graph odometry stddev	0.15	0.2
e_{max}	maximum association distance	10 m	10 m
v	initial consistency expectation for EM	0.5	0.5

5 Dataset Information

In this section, we provide additional information on the real-world, semi-static, long-term warehouse dataset to be released with this paper.

5.1 Robot and Sensors

The dataset was collected on a mobile platform, remotely driven by a human operator at walking speed. Sensor measurements were recorded from two Microsoft Azure RGB-D cameras, an Ouster 128-beam LiDAR, and three inertial measurement units (IMU), all rigidly mounted on the platform. Figure 14 shows the robot platform with the mounted sensors, and Table 2 lists the specifications of the sensors.

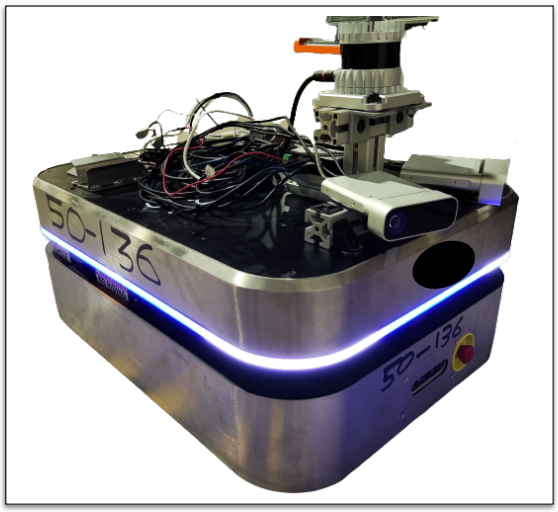


Figure 14: The robot platform used to collect the dataset, with the mounted sensors.

Table 2: Sensor Data and Specifications

Sensor	Data	FPS	Resolution	FOV	Format
Azure Kinect	colour	10	1280x720	H:90 V:59	RGB image
Azure Kinect	depth	10	1280x720	H:90 V:59	16-bit depth (mm)
Azure Kinect	semantics	10	1280x720	H:90 V:59	RGB and Class ID mask
Azure Kinect	IMU (accel & gyro)	200	-	-	text file timestamp a_x a_y a_z ω_x ω_y ω_z
Ouster OS1-128 3D LiDAR	point cloud	10	H:2048 V:128	H:360° V:45°	PCD point cloud file
Ouster OS1-128 3D LiDAR	pose	10	-	-	text file timestamp p_x p_y p_z q_x q_y q_z q_w

5.2 Sensor Calibration and Synchronization

We calibrate the intrinsics of the Microsoft Azure cameras using the Matlab Camera Calibration Toolbox, and the extrinsics are obtained by aligning the depth data to the Ouster Lidar. It is assumed that the calibrations remain intact for all trajectories. The two stereo cameras are triggered synchronously with a cable. The Ouster 3D LiDAR is time-stamped with respect to the internal oscillator instead of using the ROS arrival time. Both Azures and the Ouster are time-synchronized to the host computer via precision time protocol (PTP).

5.3 Dataset Scenarios

The dataset provides 20 trajectories in three scenarios. Each trajectory contains the robot traversing through a specific portion of the warehouse environment, starting and finishing around the origin of the provided ground truth map. A high level overview of the scenarios and trajectories is listed in Table 3. Examples of dynamic objects include other robots, people, and forklifts. Semi-static objects include boxes and pallets that shift over the course of the four months. The warehouse blueprint can be seen in Figure 15. The data was collected on June 15, June 23, and October 12, 2022.

Table 3: Dataset Trajectory Breakdown by Scenarios

Route	Number of Trajectories	Description
Aisle - CW	5	Moving through the aisle area clockwise.
Aisle - CCW	5	Moving through the aisle area counter-clockwise.
Hallway Straight - CW	2	Moving through the straight hallway area clockwise.
Hallway Straight - CCW	3	Moving through the straight hallway area counter-clockwise.
Hallway Full - CW	4	Moving through the hallway, aisle, and receiving area clockwise.
Hallway Full - CCW	1	Moving through the hallway, aisle, and receiving area counter-clockwise.

Table 4: Dataset Trajectory Breakdown by Date

Date	Number of Trajectories
Jun. 15, 2022	7
Jun. 23, 2022	7
Oct. 12, 2022	6

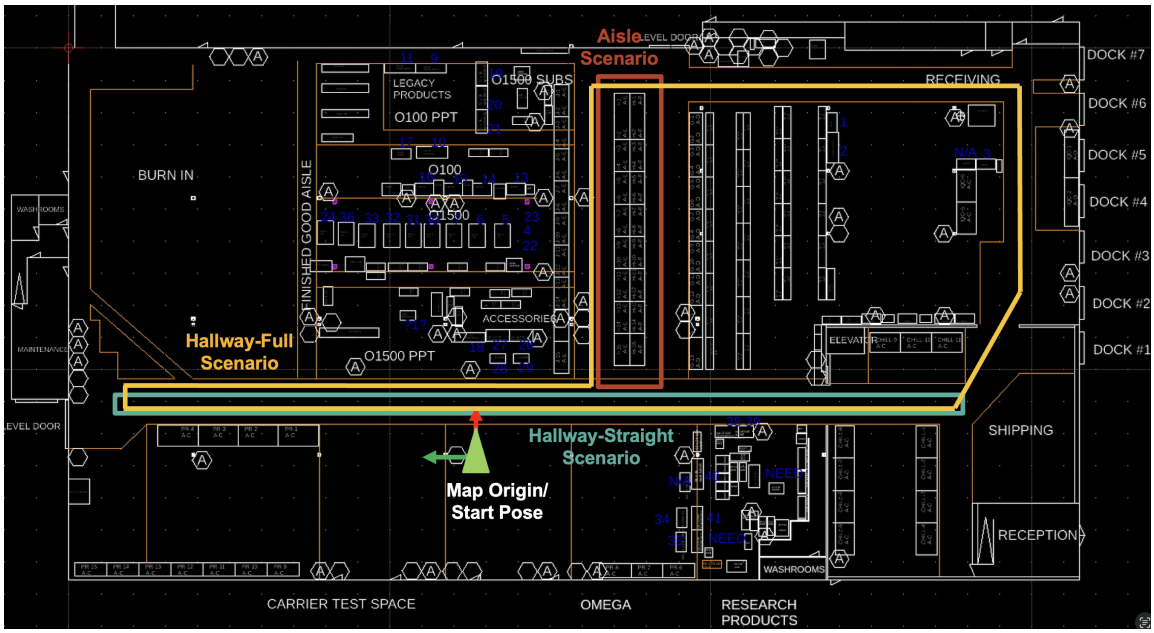


Figure 15: A blueprint of the warehouse in which the dataset was collected. The trajectories all start at the map origin.

5.4 Ground Truth Information

To facilitate with benchmarking, we also provide a high-precision map of the warehouse and ground truth robot pose for each trajectory. A Leica MS60 multistation, shown in Figure 16, was used to obtain

high-density 3D scans of the warehouse, which were later stitched together using the Leica Geosystems Cyclone software to form the complete ground truth map. To obtain the ground truth robot poses, we first run a Lidar SLAM algorithm using the onboard 3D Ouster Lidar and then fine tune the poses against the ground truth map.

Figure 16: An image of the Leica MS60 multistation that was used to obtain a ground truth map of the warehouse, as seen in Figure 17.

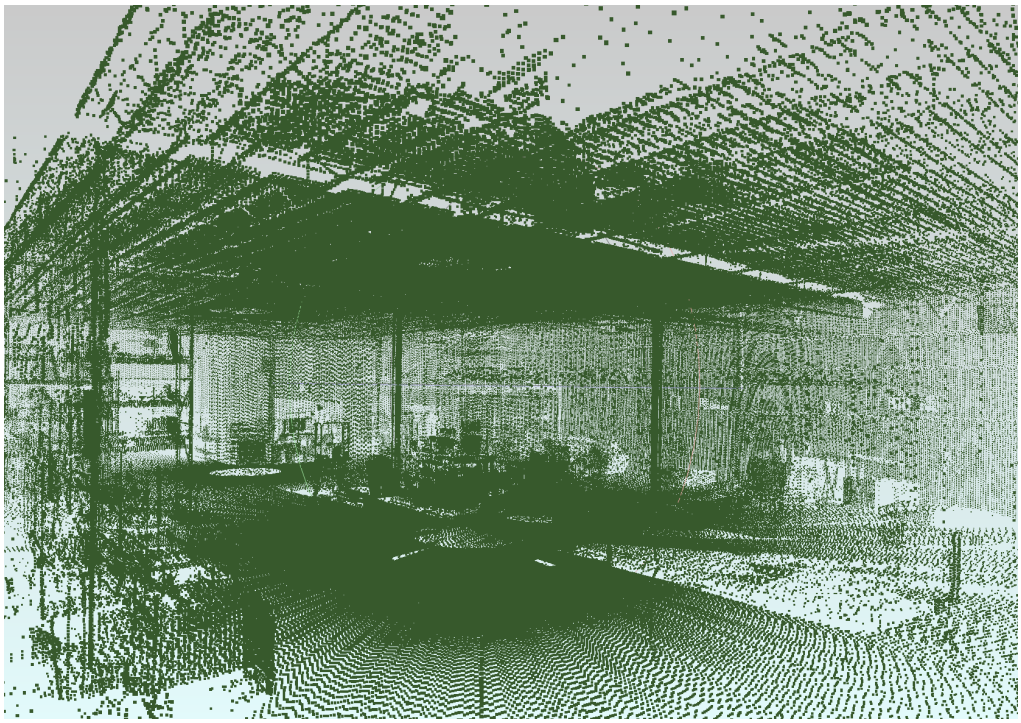


Figure 17: A sample view of the ground truth 3D map taken with the Leica MS60 multistation.