Perpetua: Multi-Hypothesis Persistence Modeling for Semi-Static Environments

Miguel Saavedra-Ruiz[1,2], Samer B. Nashed[1,2], Charlie Gauthier[1,2], Liam Paull[1,2]

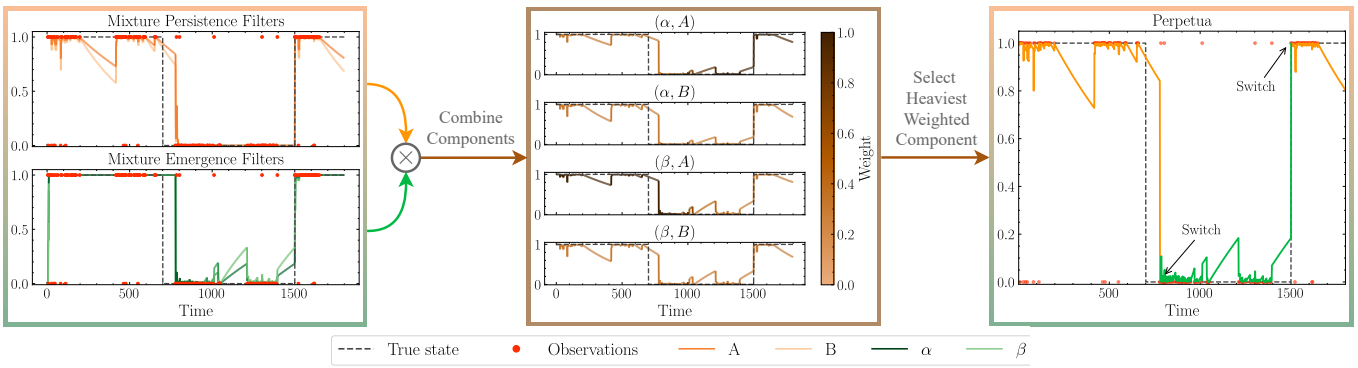

Fig. 1: ***Perpetua*** models the presence and absence, also known as feature persistence, of semi-static features through a combination of two mixtures models, one for disappearance (mixture of persistence filters) and one for reappearance (mixture of emergence filters), combined with a switching mechanism to toggle between them. In the above figure, the mixture of emergence and persistence filters (left panel) each have two components, which combine to make 4 possible outcomes (middle). Based on the available measurements, we select the most likely combination (right) to model the persistence of a semi-static feature.

***Abstract*—Many robotic systems require extended deployments in complex, dynamic environments. In such deployments, parts of the environment may change in between subsequent observations by the robot. Few robotic mapping or environment modeling algorithms are capable of representing dynamic features in a way that enables predicting their future state. Instead, most approaches opt to filter certain state observations, either by removing them or some form of weighted averaging. This paper introduces *Perpetua*, a method for modeling the dynamics of semi-static features. Perpetua is able to: incorporate prior knowledge about the dynamics of the feature if it exists, track multiple hypotheses, and adapt over time to enable predicting their future state. Specifically, we chain together mixtures of "persistence" and "emergence" filters to model the probability that features will disappear or reappear in a formal Bayesian framework. The approach is an efficient, scalable, general, and robust method for estimating the state of features in an environment, both in the present as well as at arbitrary future times. Through experiments on both simulated and real-world data, we find Perpetua yields better accuracy than similar approaches while also being online adaptable and robust to missing observations.**

I. Introduction

Effective robotic planning requires accurate state estimation of the robot's operating environment. However, achieving this for both present and future time steps, and for aspects of the environment not currently visible, is particularly challenging due to partial, noisy sensor data and the potential for complex, *semi-static feature dynamics* wherein features (e.g., points, lines, surfels, objects) may appear or disappear between observations. While recent research has shown the value of estimating the state of semi-static features, sometimes called persistence estimation, for tasks such as localization [1], [2], mapping [3], [4], navigation [5]–[7], and planning [8], our ability to estimate persistence in practice remains limited.

Many semi-static changes exhibit a time-dependent nature, such as an office door being open on weekdays and closed on weekends, allowing the possibility of learning persistence estimators that can capture the qualitative dynamics and predict future changes [5], [9]. In addition, ideal solutions should also be (1) robust to partial observations, (2) adaptable and updatable online, (3) capable of representing multiple dynamic modes (e.g., hourly versus daily changes), and (4) reliant on minimal prior knowledge.

Methods for estimating or predicting feature persistence fall on a spectrum, with filtering methods such as those based on the persistence filter introduced by Rosen *et al.* [10] on one end, and predictive methods, such as FreMen [5], on the other. Contemporary persistence filtering methods [2], [11] are typically highly adaptable and robust to observation noise, but have difficulty modeling feature *reappearance*, and often require accurate priors over model parameters to be effective. In contrast, predictive methods can model feature reappearance, learn dynamics from data (albeit offline), and capture multiple dynamic modes. However, they cannot adapt to new information online and are susceptible to noise in the training data.

This paper proposes *Perpetua* (Fig. 1), an efficient method for estimating the persistence of semi-static features that combines the benefits of both filtering and predictive approaches, enabling robustness to missing observations, online adaptation from data, future persistence prediction, characterization of multiple dynamical hypotheses, and reduced dependence on prior knowledge. To achieve this, we first develop a posterior inference framework that jointly estimates persistence and latent mixture components within a mixture of persistence

[1]Department of Computer Science and Operations Research, Université de Montréal. [2]Mila - Quebec AI Institute.

filters modeling multiple feature dynamics (§IV-A). Next, we derive an *emergence filter*, based on the persistence filter, to capture feature reappearance (§IV-B), and show how to combine mixtures of persistence and emergence filters via a state machine to track and predict the disappearance and reappearance of features arbitrarily far into the future (§IV-C). Finally, we derive an expectation-maximization-based approach for learning model parameters from noisy observations (§IV-D).

Empirically, Perpetua's persistence estimates are more accurate than baseline methods over a variety of prediction horizons while remaining robust to sensor noise and missing observations. Experiments are conducted on both simulated and real-world datasets exhibiting semi-static changes. Webpage at: `https://montrealrobotics.ca/perpetua`

II. Related Work

Modeling, tracking, and predicting feature persistence has been an important topic in robotics [12]–[14]. Generally, work in this area falls into one of four broad categories: (1) modeling via Markov chains, (2) using deep learning, (3) predictive models, and (4) filters. Markov chain methods represent presence and absence as states, and model dynamics as state transitions [15], [16]. While these approaches can learn parameters from data, they are limited to predicting future persistence at fixed intervals, rather than continuously.

On the other extreme, deep learning-based methods have shown an ability to estimate feature persistence, for example, by using object-graphs built from latent representations of point clouds [17], or training neural classifiers to predict position, state, and variability of objects on a 3D scene graph [18]. Thomas *et al.* [19] employ a kernel point convolution network to predict spatio-temporal occupancy over short time windows. However, except for Thomas *et al.*, these methods cannot predict future persistence and many require large amounts of labeled data for training.

Many approaches use either predictive models or filters. Predictive models often relax the assumption of known priors over feature dynamics, and one of the most well-known predictive models is FreMen, introduced by Krajník *et al.* [5], which models feature persistence using Fourier analysis enabling future persistence estimates at any time via an inverse Fourier transform. Later, FreMen was extended to jointly model persistence over space and time [20]. Other methods have proposed using Guassian processes over dynamic Hilbert maps for continuous future occupancy prediction [21], or employed autoregressive moving average models (ARMA) to estimate and predict persistence [9], [22]. Although these methods offer strong predictive capabilities, they lack online adaptability and rely on observation sequences for training.

Filtering methods, and in particular the persistence filter [10] and its derivatives (e.g. [11], [23]), have several important strengths, including robustness to noisy observations, real-time adaptation, and a strong theoretical foundation. While they can predict persistence at future time steps, these models generally become inaccurate quickly. Deng *et al.* [23] addressed this by introducing a long short-term exponential model that updates persistence estimates by prioritizing the latest observation, but this approach remains sensitive to observation noise. The main drawbacks of such methods are the reliance on accurate a priori model parameters, a single persistence hypothesis per feature, and the inability to handle feature reappearance.

Perpetua unifies key aspects of these methods: adaptability and noise resilience of online methods, and predictive power and robustness to priors of offline methods. We extend the persistence filter to a mixture model to track multiple dynamic modes, introduce the emergence filter to model reappearance, present a method for switching between filters, and derive an algorithm for learning model parameters directly from data.

III. Background and Preliminaries

A. Problem Definition

We consider an agent making repeated observations of an environment undergoing semi-static changes, where features appear and disappear over time, but where these transitions are not necessarily observable. Our goal is to model and predict the presence or absence of features, also known as feature persistence, at any time $t \in [0, \infty)$. Let $X_t \in \{0, 1\}$ represent the *persistence* of a feature at time t, where $X_t = 1$ indicates presence and $X_t = 0$ indicates absence. Given a sequence of noisy observations modeled as Boolean random variables $\{Y_{t_i}\}_{i=1}^N \subseteq \{0, 1\}^N$, sampled at times $\{t_i\}_{i=1}^N \in [t_0, \infty)$, the goal is to infer the belief over feature persistence. We make no assumption about the type of sensor used for observation.

B. The Persistence Filter

The *Persistence filter* [10], is a probabilistic model rooted in Bayesian survival analysis [24] that describes the survival time, or the length of time a feature exists before disappearing, of semi-static features. The persistence model is defined as

$$\begin{aligned} T &\sim p_T(\cdot), \\ X_t \mid T &= \begin{cases} 1, & t \leq T, \\ 0, & t > T, \end{cases} \\ Y_t \mid X_t &\sim p_{Y_t}(\cdot \mid X_t); \end{aligned} \tag{1}$$

where $p_T : [0, \infty] \to [0, \infty]$ is a probability density function that denotes the prior over survival time $T \in [0, \infty)$, and $p_{Y_t}(\cdot \mid X_t)$ is a conditional distribution that denotes the measurement model for some time $t \in [0, \infty)$. In this model, $X_t = 1$ is equivalent to $T \geq t$, and we will use them interchangeably. The measurement model is characterized by the probability of missed detections $P_M = p(Y_t = 0 | X_t = 1)$, and the probability of false alarm $P_F = p(Y_t = 1 | X_t = 0)$. Further, let the cumulative distribution function (CDF) of p_T be $F_T(t) \triangleq p(T \leq t) = \int_0^t p_T(\tau) d\tau$.

Given a sequence of noisy observations $\mathcal{Y}_{1:N} \triangleq \{y_{t_i}\}_{i=1}^N$ and the parameters of the measurement model $P_M, P_F \in [0, 1]$, Rosen *et al.* [10] derived a closed-form solution to compute the posterior probability $p(X_t = 1 \mid \mathcal{Y}_{1:N})$ at any *present* or *future* time $t \in [t_N, \infty)$

$$p(X_t = 1 \mid \mathcal{Y}_{1:N}) = \frac{p(\mathcal{Y}_{1:N} \mid T \geq t) p(T \geq t)}{p(\mathcal{Y}_{1:N})}. \tag{2}$$

The general form of the measurement model $p(\mathcal{Y}_{1:N} \mid T)$, which accounts for both cases $T \geq t$ and $T < t$, is given by

$$p(\mathcal{Y}_{1:N} \mid T) = \prod_{t_i \leq T} P_M^{1-y_{t_i}} (1-P_M)^{y_{t_i}} \prod_{t_i > T} P_F^{y_{t_i}} (1-P_F)^{1-y_{t_i}}. \tag{3}$$

Thus, the measurement model $p(\mathcal{Y}_{1:N} | \, T \geq t) \triangleq p(\mathcal{Y}_{1:N} | \, t_N)$ in (2) for $t_N > t_{N-1}, t_{N-2}, \ldots$ is computed recursively as

$$p(\mathcal{Y}_{1:N} \mid T \geq t) = \prod_{i=1}^{N} P_M^{1-y_{t_i}} (1-P_M)^{y_{t_i}}. \tag{4}$$

As (3) is right-continuous and constant on the interval $[t_i, t_{i+1})$, the evidence $p(\mathcal{Y}_{1:N})$ can be efficiently computed as a sum over disjoint time intervals for all $i \in \{0, 1, \ldots, N\}$

$$p(\mathcal{Y}_{1:N}) = \sum_{i=0}^{N} p(\mathcal{Y}_{1:N} \mid t_i)[F_T(t_{i+1}) - F_T(t_i)], \tag{5}$$

where $t_0 = 0$ and $t_{N+1} = \infty$. Rosen et al. [10] showed that (5) can be iteratively updated by decomposing $p(\mathcal{Y}_{1:N})$ as

$$p(\mathcal{Y}_{1:N}) = L(\mathcal{Y}_{1:N}) + p(\mathcal{Y}_{1:N} \mid t_N)[1 - F_T(t_N)], \tag{6}$$

with $L(\mathcal{Y}_{1:N}) \triangleq \sum_{i=0}^{N-1} p(\mathcal{Y}_{1:N} \mid t_i) \, [F_T(t_{i+1}) - F_T(t_i)]$ denoting the *lower partial sum* of the evidence, obtained by excluding the contribution of the N^{th} term in (5). In turn, it is easy to iteratively update $L(\mathcal{Y}_{1:N})$ as

$$\begin{aligned} L(\mathcal{Y}_{1:N}) = & P_F^{y_{t_N}} (1-P_F)^{1-y_{t_N}} (L(\mathcal{Y}_{1:N-1}) + \\ & p(\mathcal{Y}_{1:N-1} \mid t_{N-1})[F_T(t_N) - F_T(t_{N-1})]). \end{aligned} \tag{7}$$

Combining (4), (6), and (7), and noting that the prior follows from $p(T \geq t) = 1 - F_T(t)$, the persistence filter can be iteratively updated in constant time as new observations arrive. Despite its advantages, the persistence filter and its extensions are constrained by three key limitations: (i) there is only a single persistence hypothesis, restricting cases where the feature may exhibit multiple plausible dynamics; (ii) once a feature vanishes, it is considered permanently gone, even when it may reappear, and (iii) the parameters of the prior are known a priori and not learned from data. In the next section, we present *Perpetua*, a method that adapts and extends the persistence filter in order to address these limitations.

IV. Perpetua

The key idea of *Perpetua* is to estimate a belief over feature persistence of each feature in the environment using a pair of mixture models: a mixture of *persistence* and *emergence* filters. These models capture the time-dependent state transition probability of appearance $0 \to 1$ (emergence) and disappearance $1 \to 0$ (persistence), and are used in combination within a state machine that transitions based on the belief of these filters. At a high level, there are three important subroutines within Perpetua: (1) the mixture models, (2) the Perpetua state machine, and (3) parameter learning. When combined, these subroutines produce a system that can update its belief about the state online and predict arbitrarily far into the future in a principled manner.

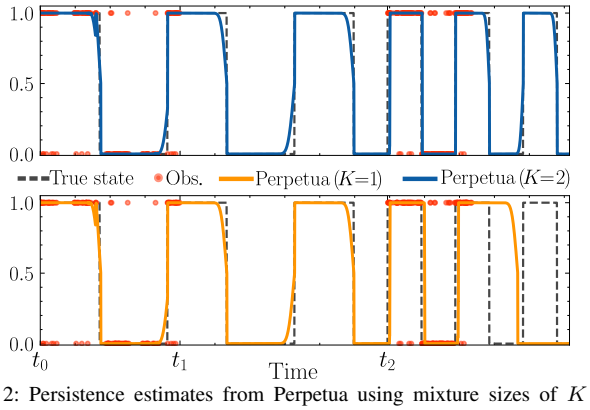

Fig. 2: Persistence estimates from Perpetua using mixture sizes of $K = 2$ (top) and $K = 1$ (bottom). From t_0 to t_1, persistence estimates adapt to the dynamics given the current noisy observations. Between t_1 and t_2, Perpetua uses the component with the largest posterior weight to *predict* feature persistence in the absence of observations. At t_2, observations resume. Perpetua ($K = 2$) adapts its persistence estimate correctly, switching to the mixture component that best explains the data, while Perpetua ($K = 1$) cannot.

A. Mixture of Persistence Filters

Handling multiple persistence hypotheses for the survival time T is crucial, as the persistence X_t may have complex dynamics, or a lack of observations may create multiple equally plausible estimates for T. This section extends the persistence filter to a mixture model, computing the posterior not only over survival time but also over latent mixture components. This allows selecting the most likely component based on available measurements of a given feature.

We extend the first line in (1) with an additional latent variable $C \sim \text{Categorical}(\pi)$, which models the mixture components, where $C \in \{1, 2, \ldots, K\}$, $\sum_{k=1}^{K} \pi_k = 1$, and $p(C = k) = \pi_k$. The mixture of persistence filters is

$$T \mid C = k \sim p_{T_k}(\cdot \mid C = k), \tag{8}$$

with key differences from (1) being the categorical prior over mixture components C, and a conditional prior $p_{T_k}(\cdot \mid C = k)$ for each mixture component. The rest of the model is identical as in (1).

To simplify the expressions in this section, we adopt the following notation: $X_t = 1 \to X_t^1$ and $C = k \to C_k$. We aim to compute the *joint posterior* $p(X_t^1, C_k \mid \mathcal{Y}_{1:N})$ of this mixture model at $t \in [t_N, \infty)$. Applying Bayes' rule, we have

$$p(X_t^1, C_k \mid \mathcal{Y}_{1:N}) = \frac{p(X_T^1 \mid C_k, \mathcal{Y}_{1:N}) p(C_k, \mathcal{Y}_{1:N})}{p(\mathcal{Y}_{1:N})}, \tag{9}$$

where $p(C_k, \mathcal{Y}_{1:N})$ is the *joint evidence* and the *conditional posterior* $p(X_t^1 \mid C_k, \mathcal{Y}_{1:N})$ is the probability that $X_t = 1$, according to component k and given observations $\mathcal{Y}_{1:N}$. Here, each $p(X_t^1 \mid C_k, \mathcal{Y}_{1:N})$ represents a persistence filter. Applying Bayes' rule to decompose the conditional posterior, we have

$$\begin{aligned} p(X_t^1 \mid C_k, \mathcal{Y}_{1:N}) &= \frac{p(\mathcal{Y}_{1:N} \mid X_t^1) p(X_t^1 \mid C_k) p(C_k)}{\int_0^\infty p(\mathcal{Y}_{1:N} \mid \tau) p(\tau \mid C_k) p(C_k) d\tau} \\ &= \frac{p(\mathcal{Y}_{1:N} \mid X_t^1) p(X_t^1 \mid C_k)}{p(\mathcal{Y}_{1:N} \mid C_k)}. \end{aligned} \tag{10}$$

The denominator of (10), $p(\mathcal{Y}_{1:N} \mid C_k)$, is the *conditional evidence*, which can be computed using (5). Specifically, we can iteratively update the conditional evidence $p(\mathcal{Y}_{1:N} \mid C_k)$ by first updating (7) as

$$L(\mathcal{Y}_{1:N} \mid C_k) = P_F^{y_{t_N}}(1-P_F)^{1-y_{t_N}}\big(L(\mathcal{Y}_{1:N-1} \mid C_k) + p(\mathcal{Y}_{1:N-1} \mid t_{N-1})A_N\big), \tag{11}$$

and plugging in $L(\mathcal{Y}_{1:N} \mid C_k)$ and $F_{T_k}(t_N) \triangleq \int_0^{t_N} p_{T_k}(\tau)d\tau$ into (6). Both $A_N \triangleq F_{T_k}(t_N) - F_{T_k}(t_{N-1})$, and $p(\mathcal{Y}_{1:N} \mid C_k)$ are computed for all mixture components. Note that the likelihood $p(\mathcal{Y}_{1:N}|t_N)$ is recursively updated the same way as in (4) and the conditional prior can be computed as $p(X_t^1 \mid C_k) = 1 - F_{T_k}(t)$, for all $k \in \{1, 2, \ldots, K\}$. Once the conditional evidence $p(\mathcal{Y}_{1:N} \mid C_k)$ is computed, the joint $p(C_k, \mathcal{Y}_{1:N})$ and marginal evidence $p(\mathcal{Y}_{1:N})$ can be efficiently obtained as

$$\text{Joint} \quad \rightarrow \quad p(C_k, \mathcal{Y}_{1:N}) = p(\mathcal{Y}_{1:N} \mid C_k)p(C_k), \tag{12}$$

$$\text{Marginal} \quad \rightarrow \quad p(\mathcal{Y}_{1:N}) = \sum_{k=1}^{K} p(\mathcal{Y}_{1:N}, C_k). \tag{13}$$

Putting together (4), (6), and (11)-(13), we have a recursive Bayesian estimator for the conditional $p(X_t^1 \mid C_k, \mathcal{Y}_{1:N})$ and joint posterior $p(X_t^1, C_k \mid \mathcal{Y}_{1:N})$ for $t \in [t_N, \infty)$.

By marginalizing the joint posterior $p(X_t^1 \mid \mathcal{Y}_{1:N}) = \sum_{k=1}^{K} p(X_t^1, C_k \mid \mathcal{Y}_{1:N})$, we recover the original persistence filter, where the effects of all mixture components are aggregated. However, due to potential destructive interference between mixture components, we instead use the component with the largest weight. We do this using the posterior weights $p(C_k \mid \mathcal{Y}_{1:N}) = \frac{p(C_k, \mathcal{Y}_{1:N})}{p(\mathcal{Y}_{1:N})}$, derived from the joint and marginal evidence. Therefore, the probability of the component with the largest posterior weight at time $t \in [t_N, \infty)$ is

$$p(X_t^1 \mid C_{k^*}, \mathcal{Y}_{1:N}) \quad \text{s.t.} \quad k^* = \arg\max_k p(C_k \mid \mathcal{Y}_{1:N}). \tag{14}$$

The procedure for estimating X_t using a mixture of persistence filters is outlined in Alg. 1. While mixtures of persistence filters handle multiple persistence hypotheses, once the filter state changes to "absence," it becomes difficult to revert to "presence" even with new observations. In the following section, we will address this problem by deriving a related model, called a mixture of emergence filters, that allows us to model the dynamics of feature reappearance.

B. Mixture of Emergence Filters

Instead of modeling feature disappearance, the mixture of emergence filters models feature reappearance. The derivation of this model follows similar steps as the mixture of persistence filters with some key differences. We start by updating the conditional $X_t \mid T^E$ as

$$X_t \mid T^E = \begin{cases} 0, & t \leq T^E \\ 1, & t > T^E; \end{cases} \tag{15}$$

here, T^E is the emergence time, representing the time it takes for a feature to reappear after disappearing. This formulation

Algorithm 1 The Mixture of Persistence Filters

Input: Observation model (P_M, P_F), CDFs $F_{T_k}(\cdot)$, prior mixture weights $p(C)$, and observations $\{y_{t_i}\}$
Output: Probability $p(X_t^1 \mid C_{k^*}, \mathcal{Y}_{1:N})$ for $t \in [t_N, \infty)$.

for $k \in \{1, \ldots, K\}$ **do**
  $t_0 \leftarrow 0$; $N \leftarrow 0$; $L(\mathcal{Y}_0 \mid C_k) \leftarrow 0$
  $p(\mathcal{Y}_0) \leftarrow 1$; $p(\mathcal{Y}_0 \mid t_0) \leftarrow 1$; $p(\mathcal{Y}_0 \mid C_k) \leftarrow 1$
end for
while $\exists$ new observation $y_{t_{N+1}}$ and $\forall k \in \{1, \ldots, K\}$ **do**
Update:
  Compute partial evidence $L(\mathcal{Y}_{1:N+1} \mid C_k)$ via (11).
  Compute likelihood $p(\mathcal{Y}_{1:N+1} \mid t_{N+1})$ via (4).
  Compute conditional evidence $p(\mathcal{Y}_{1:N+1}|C_k)$ via (6).
  Compute joint evidence $p(C_k, \mathcal{Y}_{1:N})$ via (12).
  Compute marginal evidence $p(\mathcal{Y}_{1:N})$ via (13).
  $N \leftarrow (N+1)$
Predict: (For any time $t \in [t_N, \infty)$)
  Compute probability $p(X_t^1 \mid C_{k^*}, \mathcal{Y}_{1:N})$ via (14).
end while

is the opposite of the one in (1), making the emergence filter the *complement* of the persistence filter.

Using the derivation from §IV-A, we can easily compute the marginal posterior for the feature's absence ($X_t = 0$) with the mixture of emergence filters: $p^E(X_T^0 \mid C_k, \mathcal{Y}_{1:N})$. The posterior for the feature's presence is obtained by taking the complement: $p^E(X_T^1 \mid C_k, \mathcal{Y}_{1:N}) = 1 - p^E(X_T^0 \mid C_k, \mathcal{Y}_{1:N})$. To ease notation, we will drop the "E" superscript. Therefore, to compute $p(X_T^0 \mid C_k, \mathcal{Y}_{1:N})$, the equations for the likelihood (4), and lower partial sum (11) must be updated as:

$$p(\mathcal{Y}_{1:N} \mid t_N) = \prod_{i=1}^{N} P_F^{y_{t_i}}(1-P_F)^{1-y_{t_i}} \tag{16}$$

$$L(\mathcal{Y}_{1:N} \mid C_k) = P_M^{1-y_{t_N}}(1-P_M)^{y_{t_N}}\big(L(\mathcal{Y}_{1:N-1} \mid C_k) + p(\mathcal{Y}_{1:N-1} \mid t_{N-1})B_N\big), \tag{17}$$

where $B_N = F_{T_k^E}(t_N) - F_{T_k^E}(t_{N-1})$ and $F_{T_k^E}$ is the CDF of the k^{th} mixture component of the emergence model. The conditional evidence, $p(\mathcal{Y}_{1:N} \mid C_k)$, is recovered by plugging $F_{T_k^E}$ and (16), (17) into (6). These equations can be incorporated into Alg. 1 alongside the cumulative density functions $F_{T_k^E}$ and emergence mixture prior $p^E(C)$ to recursively compute the Bayesian posterior of the emergence model.

The combination of the persistence and emergence mixture models allows us to model both types of state transitions, and operating with mixture models relaxes the assumption of a single dynamical mode. However, we still need to integrate both mixture models to produce a single, coherent estimate for the persistence X_t, and do so in a manner that maintains high-quality estimates in the presence of noisy or missing observations. In the next section, we introduce a state machine for performing this integration.

C. The Perpetua State Machine

The purpose of the state machine is to control inference on X_t by switching between the persistence and emergence models based on the probability of their heaviest weighted component. When a feature is first observed at time t_a, the state machine is initialized in the persistence state. This instantiates a mixture of persistence filters for a given feature, which starts updating its belief given observations. While in the persistence state, only the persistence mixture is used to estimate X_t. If the probability $p(X_t^1 \mid C_{k^*}, \mathcal{Y}_{a:b})$, of the persistence model drops below a threshold δ_{low} at time t_b, Perpetua transitions to the emergence state and instantiates a mixture of emergence filters. The emergence model runs until $p^E(X_t^1 \mid C_{k^*}, \mathcal{Y}_{b:c})$ exceeds a threshold δ_{high} at time t_c, at which point the state machine transitions back to the persistence state.

Upon *re-entry* into emergence or persistence states, the respective model is reset, and prior mixture weights updated as

$$p^{\text{new}}(C) = \epsilon p(C) + (1-\epsilon)p(C \mid \mathcal{Y}_{p:q}), \quad \epsilon \in [0,1], \tag{18}$$

with $\mathcal{Y}_{p:q}$ denoting the observations taken while the mixture was last active, $p(C)$ the *initial* mixture weights, and $p^{\text{new}}(C)$ the updated weights used to initialize the model after a reset.

The motivation for (18) is two-fold. First, every reset destroys information from previous observations. To alleviate this loss of information, we re-use the posterior weights of the previous model during initialization. Second, incorporating $p(C)$ into (18) prevents mode collapse while maintaining the ability to quickly adapt to new modes in the data (see Fig. 2).

Each time a new observation is made, we must assess whether the state of Perpetua has changed between the previous time t_{i-1} and the current time t_i. To do this, we determine the state change by monitoring the probability of the component with the largest posterior weight of the persistence and emergence mixtures from t_{i-1} to t_i. During this "simulation", Perpetua can switch between the emergence and persistence states multiple times, depending on the temporal distance between t_{i-1} and t_i, as illustrated in Fig. 2.

Perpetua not only predicts when a feature will disappear or reappear, but also tracks and predicts the state of features with multiple potential dynamics. By combining components from the emergence and persistence models, we can estimate multiple hypotheses by pairing different mixture components. This work uses the heaviest weighted component of each mixture model to improve persistence estimates. Fig. 1 illustrates how Perpetua estimates the persistence of a feature and all the possible outcomes that can be obtained by combining the mixture components of the persistence and emergence models.

D. Parameter Learning

One of the strengths of Perpetua is its ability to estimate the parameters of the distributions over T^E/T and C from noisy data. To simplify notation, we will represent C as a one-hot vector with K binary random variables $C = [C_1, C_2, \ldots, C_K]^T$ instead of $C \in \{1, 2, \ldots, K\}$. The element $C_k = 1$ if and only if the result belongs to class k, and zero otherwise. Without loss of generality, we will demonstrate our derivations for the persistence model but the same steps can be followed to obtain the learning equations of the emergence model. It is worth noting that Dang *et al.* [23] mentioned performing parameter learning over the parameters of T, but we were unable to find details on its derivation.

Mixture of Exponential Priors. Assume p_{T_k} is an exponential distribution: $p_{T_k}(T \mid C_k = 1) \triangleq \lambda_k \exp(-\lambda_k T)$. Since the mixture of persistence filters contains two hidden variables (C and T), we use the expectation-maximization algorithm [25] to estimate the parameters $\Theta \triangleq \{(\lambda_k, \pi_k)\}_{k=1}^K$.

Given an observation sequence $\mathcal{Y}_{1:S}$ with $S >> N$, where a feature may transition multiple times between presence and absence, we assume that these transition times are identifiable. Therefore, we partition our data into M disjoint sets of the form $\mathcal{Y} \triangleq \{\mathcal{Y}_{N_j:N_j^{\text{Last}}}\}_{j=1}^M$, where N_j is the index of the first observation in the j^{th} set, N_j^{Last} the index of the last observation, and $N_1 = 1$. Denoting $\mathcal{T} = \{T_j\}_{j=1}^M$ and $\mathcal{C} = \{c_{jk}\}_{j=1,k=1}^{M,K}$, the *complete-data likelihood* can be written as

$$p(\mathcal{Y}, \mathcal{T}, \mathcal{C}; \Theta) = \prod_{j=1}^{M} p(\mathcal{Y}_{N_j:N_j^{\text{Last}}} \mid T_j) p(T_j \mid C_j) p(C_j), \tag{19}$$

with $p(\mathcal{Y}_{N_j:N_j^{\text{Last}}} \mid T_j)$ defined by (3), $p(C_j) \triangleq \prod_{k=1}^K \pi_k^{c_{jk}}$, and $p(T_j \mid C_j) = \prod_{k=1}^K [\lambda_k \exp(-\lambda_k T_j)]^{c_{jk}}$. Taking the logarithm of (19) and then the expectation with respect to $q^{[u+1]}(\mathcal{T}, \mathcal{C}) \triangleq p(\mathcal{T}, \mathcal{C} \mid \mathcal{Y}; \Theta^{[u]})$, where u is the current EM iteration, gives the maximization objective: $\mathcal{L}(q, \Theta) \triangleq \mathbb{E}_q[\log p(\mathcal{Y}, \mathcal{T}, \mathcal{C}; \Theta)]$, defined as

$$\mathcal{L}(q, \Theta) \propto \sum_{j=1}^{M} \sum_{k=1}^{K} \Big[\mathbb{E}_q[c_{jk}] (\log \lambda_k + \log \pi_k) - \lambda_k \mathbb{E}_q[c_{jk} T_j] \Big], \tag{20}$$

where we dropped the terms that do not depend on Θ.

E-Step: In this step, we fix the parameters $\Theta^{[u]}$ and compute the distribution $q^{[u+1]}$. Although q can be intractable we show it allows for a closed-form solution. From the definition of $q^{[u+1]}(T_j, c_{jk}) \triangleq p(T_j, c_{jk} \mid \mathcal{Y}_{N_j:N_j^{\text{Last}}})$, we have

$$q^{[u+1]}(T_j, c_{jk}) = \frac{p(\mathcal{Y}_{N_j:N_j^{\text{Last}}} \mid T_j) p(T_j \mid c_{jk}) p(c_{jk})}{\int_0^\infty \sum_{l=1}^K p(\mathcal{Y}_{N_j:N_j^{\text{Last}}} \mid \tau) p(\tau \mid c_{jl}) p(c_{jl}) d\tau}, \tag{21}$$

$$q^{[u+1]}(c_{jk}) = p(c_{jk} \mid \mathcal{Y}_{N_j:N_j^{\text{Last}}}). \tag{22}$$

Note that (22) are the *posterior weights* derived in §IV-A. We define $\phi_{jk}^{[u+1]} \triangleq \mathbb{E}_q[c_{jk}] = q^{[u+1]}(c_{jk} = 1)$, and $\psi_{jk}^{[u+1]} \triangleq \mathbb{E}_q[c_{jk} T_j]$. Hereafter, we omit the u superscript to improve readability. By using the same decomposition as in (5), and subtracting the observation time, $t_{N_{j-1}^{\text{Last}}}$, to all $t \in \{t_{N_j}, \ldots t_{N_j}^{\text{Last}}\}$ in the j^{th} set, we can compute ψ_{jk} as

$$\psi_{jk} = \frac{p(c_{jk} = 1)}{p(\mathcal{Y}_{N_j:N_j^{\text{Last}}})} \sum_i p(Y_{N_j:N_j^{\text{Last}}} \mid t_i) \int_{t_i}^{t_{i+1}} p(\tau \mid c_{jk} = 1) \tau d\tau, \tag{23}$$

with $i \in \{0\} \cup \{N_j, \ldots, N_j^{\text{Last}}\}$ and $t_{N_0^{\text{Last}}} = 0$. Following (5), we set $t_{N_j^{\text{Last}}+1} = \infty$. For an exponential distribution, it can be

shown that $\rho_{ijk} \triangleq \int_{t_i}^{t_{i+1}} p(\tau \mid c_{jk} = 1)\tau d_\tau$ (with j denoting the observation set) has the closed-form

$$\rho_{ijk} = (t_i + \frac{1}{\lambda_k}) \exp(-\lambda_k t_i) - (t_{i+1} + \frac{1}{\lambda_k}) \exp(-\lambda_k t_{i+1}).$$

M-Step: In this step, we fix the variational distribution $q^{[u+1]}$ and optimize the parameters $\Theta^{[u]}$ by maximizing the objective in (20). This leads to the parameters updates

$$\lambda_k^{[u+1]} = \frac{\sum_{j=1}^{M} \phi_{jk}}{\sum_{j=1}^{M} \psi_{jk}} \quad \text{and} \quad \pi_k^{[u+1]} = \frac{\sum_{j=1}^{M} \phi_{jk}}{\sum_{j=1}^{M} \sum_{l=1}^{K} \phi_{jl}}. \quad (24)$$

Therefore, to optimize $\Theta^{[u]}$, we first perform the E-step using (22) and (23) while keeping the parameters fixed. Then, with $q^{[u+1]}$ updated, we optimize $\Theta^{[u+1]}$ using (24). This process is repeated for U iterations or until convergence[1].

a) Model Selection: To determine the optimal number of components while balancing model complexity, we use the Akaike information criterion (AIC) [26]:

$$AIC = 2p - 2\ln(\hat{L}), \quad (25)$$

with $\hat{L} \triangleq p_{\text{Best}}(\mathcal{Y}_{1:S})$ denoting the highest evidence achieved by the model, and p the number of model parameters. The AIC balances goodness of fit and model simplicity by penalizing excessive parameters, helping us to prevent models that overfit the train set.

V. Results

We demonstrate that our method, Perpetua, exhibits improved adaptation capabilities compared to baselines while maintaining predictive performance comparable to methods specifically designed to predict persistence. Evaluation is performed in one simulated and one real-world environment with two datasets. We evaluate our method with three metrics: the mean absolute error $\text{MAE}(f, g) = \frac{1}{(t_1 - t_0)} \int_{t_0}^{t_1} |f(t) - g(t)|\, dt$, where f is the persistence estimate and g the ground truth; balanced accuracy $\text{B-Acc} = \frac{1}{2}\left(\frac{\text{TP}}{\text{TP+FN}} + \frac{\text{TN}}{\text{TN+FP}}\right)$, which accounts for class imbalance by equally weighting true positives and true negatives; and F1 score $\text{F1} = \frac{2\text{TP}}{2\text{TP+FP+FN}}$, to evaluate prediction reliability by balancing precision and recall. Here, TP, FP, TN and FN denote true/false positives, and true/false negatives, respectively. We threshold persistence estimates at 0.5 to compute B-Acc and F1 score.

Throughout evaluation, we use the following four baselines:

- **FreMen** [5]. Following the authors, we fit the model using 1000 Fourier coefficients but determine the optimal number for prediction via a held-out validation set. The model is periodically re-fit in all experiments.
- **ARMA** [9], [22]. Following the authors' procedure, we determine the model order by minimizing the AIC. Since the method was not available online, we implemented it ourselves. The model is also periodically re-fit.
- **PF** [10]. We extend the persistence filter with our mixture formulation, using a mixture of exponential distributions. The model parameters are learned from data following the procedure outlined in §IV-D.
- **PF LSTE** [23]. The persistence filter with long short-term exponential memory. For consistency, we extend this model analogously to the standard PF. Since no implementation was available, we implemented it ourselves.

[1]We present the derivation of the learning equations for a mixture model with log-normal priors here https://montrealrobotics.ca/perpetua.

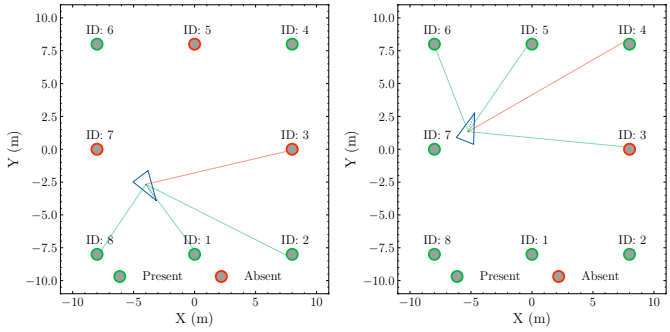

Fig. 3: Robot collecting data in the simulated room. (left) train; (right) test.

We set $\delta_{\text{low}} = 0.05$, $\delta_{\text{high}} = 0.95$ and $\epsilon = 0.1$ for all experiments. Perpetua is evaluated using two mixture distributions: exponential (Exp) and log-normal (Log-N). We use the `ruptures` library [27] during training to detect change points in the data (see §IV-D). The number of mixture components is determined by computing the AIC (25) with $k \in \{1, \ldots, 5\}$. We train our models for up to 250 iterations with uninformed initialization. For our method and persistence filter-based baselines, P_M and P_F are set based on dataset statistics.

A. Simulation in Room Environment

The first evaluation, conducted in simulation, compares the ability of Perpetua and the baselines to estimate persistence. In this experiment, a robot navigates a room composed of eight landmarks where four are static and four are semi-static (see Fig. 3). All semi-static landmarks have different appearance and disappearance times, ranging from 1min to 20min, sampled from a mixture of log-normal distributions. Landmarks can have up to three distinct appearance and disappearance times.

During data collection, the robot navigates counter-clockwise for 12 hours, while during testing the robot moves clockwise for 3 hours. To avoid measuring at regular intervals, the robot's speed is varied between $0.5\frac{m}{s}$ and $2.5\frac{m}{s}$ every five minutes. Both training and test sets have observation noise $P_M = P_F = 0.1$. Ground-truth is computed every 0.5 seconds, including intervals where features are not being observed.

Since all methods based on the persistence filter can leverage noisy test-time observations, we use the following evaluation procedure: Given a time t in the test set, we allow the model to process all data up until t and then query the model to estimate persistence at time $t + \Delta t$. We call Δt the prediction time. For methods requiring re-fitting, such as FreMen and ARMA, we follow the same procedure but re-fit the model after consuming data up to time t. While in some cases re-fitting may not be possible in real-time, we chose this approach to ensure a fair comparison.

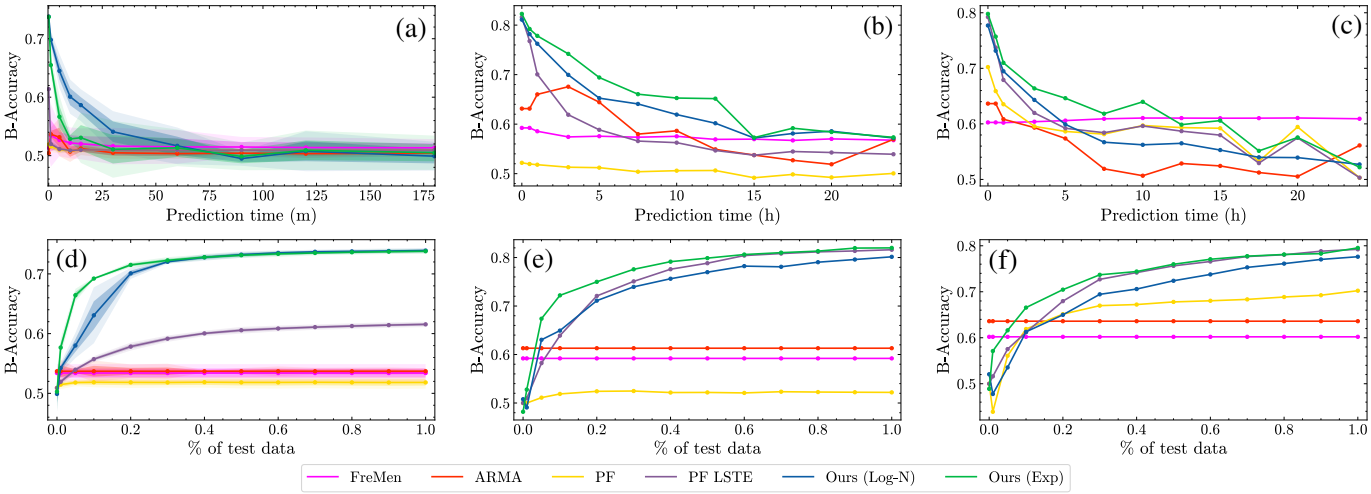

Fig. 4: Prediction (a-c) and data sparsity ablation (d-f) in the simulated room (a,d) and parking lot datasets, UFPR04: (b,e), UFPR05: (c,f). The top row shows the balanced accuracy of all methods when predicting feature persistence for a prediction time (Δt) in the future. The bottom row presents an ablation study on data sparsity, evaluating filter-based methods when only a random subset of the noisy test data is available. These results highlight the robustness of filtering methods, and Perpetua in particular, to limited data. The best predictive versions of FreMen and ARMA are included for reference.

TABLE I: Results in Room Environment. All metrics are averaged over five random seeds. Shaded rows show ground-truth-informed Perpetua.

Method	MAE ↓	B-Acc ↑	F1 ↑
FreMen [5]	0.244 ± 0.015	0.534 ± 0.027	0.700 ± 0.055
ARMA [9]	0.232 ± 0.003	0.537 ± 0.006	0.691 ± 0.012
PF [10]	0.125 ± 0.007	0.519 ± 0.005	0.569 ± 0.016
PF LSTE [23]	0.110 ± 0.002	0.614 ± 0.004	0.806 ± 0.007
Ours (Exp)	0.015 ± 0.001	0.737 ± 0.002	0.981 ± 0.003
Ours (Log-N)	$\mathbf{0.010 \pm 0.001}$	$\mathbf{0.738 \pm 0.003}$	$\mathbf{0.983 \pm 0.004}$
Ours GT (Exp)	0.015 ± 0.001	0.737 ± 0.002	0.981 ± 0.003
Ours GT (Log-N)	0.011 ± 0.001	0.736 ± 0.001	0.980 ± 0.002

Table I (see also Fig. 4 (a)) presents the results for all methods in the simulated room experiment. The results presented correspond to prediction times $\Delta t \in [0, 3\text{hr}]$ for which each model achieved the highest B-Acc. Note that maximizing B-Acc may not simultaneously achieve the lowest MAE or highest F1. In general, Perpetua outperforms the baselines across all metrics. The two bottom rows of Table I show Perpetua's performance when given the ground truth parameters governing the dynamics, demonstrating that the parameters Perpetua learns in the uninformed case are highly accurate.

We also evaluate the robustness of the filter-based methods to data sparsity. Here, persistence estimators receive only a random subset of noisy observations from the test set, and prediction time $\Delta t = 0$. As shown in Fig. 4 (d), Perpetua outperforms the baselines using only 5% of the test observations, and starts to plateau at 20%. We present the best FreMen and ARMA models from the previous experiment for reference.

B. Parking Lot Dataset

This experiment tests persistence estimators when learning model parameters from real-world data where priors are unavailable. For this, we use the parking lot dataset from Almeida *et al.* [28], which consists of 12,427 images of parking lots captured under varying environmental conditions. These images were taken in the parking lots of the Federal University of Parana (UFPR), with observations recorded every five minutes for more than 30 days. For this evaluation, we focus on the UFPR04 and UFPR05 subsets, which represent different views of the same parking lot captured from the fourth and fifth floors of the UFPR building. The UFPR04 set contains 28 parking spaces (tracked features), while UFPR05 has 45.

Fig. 5: YOLOv11 applied to UFPR04 (left) and UFPR05 (right).

To match real-world conditions, we augment the dataset using an off-the-shelf object detector, YOLOv11 [29], to obtain observations. Example detections are shown in Fig. 5, where some vehicles are not detected, introducing noise into the observations. YOLOv11 achieves an accuracy of 85.5% on UFPR04 and 79.9% on UFPR05. We follow the same evaluation procedure described in §V-A and use the last 30% of data as the test set.

The results, shown in Fig. 4 (b-c), demonstrate that Perpetua outperforms all methods in the UFPR04 set across all prediction times. In the UFPR05 set, Perpetua (Exp) outperforms the baselines for prediction time $\Delta t \leq 10$hr, beyond which fully predictive methods such as FreMen and ARMA surpass all filter-based methods, with Perpetua remaining the most accurate filter-based approach. These results suggest that fitting a mixture of exponential distributions may be easier than a mixture of log-normals. Table II presents the most accurate models for each method over all prediction times.

In Fig. 4 (e-f), we present the results of the data sparsity ablation. Perpetua (Exp) outperforms the baselines using only

TABLE II: Results in Parking Lot dataset. UFPR04 has 28 parking spaces (tracked features) and UFPR05 contains 45.

Method	UFPR04			UFPR05		
	MAE ↓	B-Acc ↑	F1 ↑	MAE ↓	B-Acc ↑	F1 ↑
FreMen [5]	0.389	0.592	0.470	0.388	0.602	0.560
ARMA [9]	0.409	0.613	0.451	0.396	0.636	0.536
PF [10]	0.539	0.512	0.067	0.421	0.702	0.528
PF LSTE [23]	0.208	0.813	0.733	**0.285**	0.792	0.738
Ours (Exp)	**0.201**	**0.823**	**0.745**	0.359	**0.798**	0.785
Ours (Log-N)	0.228	0.811	0.735	0.359	0.777	**0.794**

10% of noisy test-time data. Although PF LSTE is similarly robust to data sparsity, its overall prediction performance (MAE, B-Acc, F1) is weaker than Perpetua. A snapshot of the persistence estimates in the UFPR04 set is shown in Fig. 6.

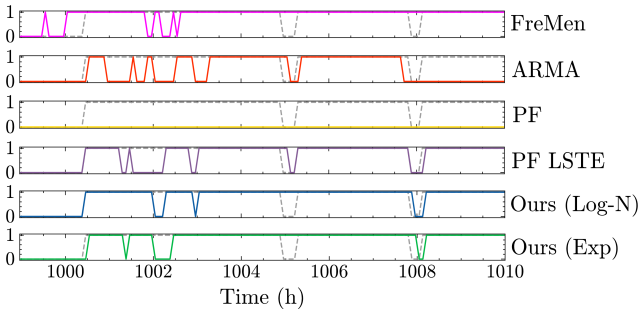

Fig. 6: Example results obtained by our method and baselines when estimating the persistence of a feature in the UFPR04 data set. Here, we threshold the persistence estimates of all methods at 0.5.

Implementation and hardware details. We implement Perpetua using JAX [30], on a Ryzen 1950X CPU with 32GB of RAM. Training a mixture model with exponential priors on 12,500 data points takes 0.058 seconds per iteration, and 0.170 seconds for the log-normal prior.

VI. Conclusion

This paper presents Perpetua, a method for estimating feature persistence that models and predicts semi-static dynamics with robust online adaptation capabilities. We show Perpetua's superior adaptation and predictive capabilities on both real and simulated data, and present additional results on robustness to missing observations. In the future, we plan to explore updating the persistence priors of our mixture models dynamically using large vision-language models. Finally, we intend to further evaluate Perpetua in complex sequential reasoning domains within long-horizon planning tasks.

References

[1] A. Adkins, T. Chen, and J. Biswas, "Probabilistic object maps for long-term robot localization," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2022.

[2] Z. Hashemifar and K. Dantu, "Practical persistence reasoning in visual slam," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2020.

[3] L. Schmid, M. Abate, Y. Chang, and L. Carlone, "Khronos: A unified approach for spatio-temporal metric-semantic slam in dynamic environments," in *Proc. of Robotics: Science and Systems (RSS)*, 2024.

[4] S. Nashed and J. Biswas, "Curating long-term vector maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2016.

[5] 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 Trans. on Robot.*, 2017.

[6] J. Qian, S. Zhou, N. J. Ren, V. Chatrath, and A. P. Schoellig, "Closing the perception-action loop for semantically safe navigation in semi-static environments," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2024.

[7] L. Nardi and C. Stachniss, "Long-term robot navigation in indoor environments estimating patterns in traversability changes," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2020.

[8] T. Krajník *et al.*, "Chronorobotics: Representing the structure of time for service robots," in *Proceedings of the International Symposium on Computer Science and Intelligent Control*, 2021.

[9] L. Wang, W. Chen, and J. Wang, "Long-term localization with time series map prediction for mobile robots in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2020.

[10] D. Rosen, J. Mason, and J. Leonard, "Towards lifelong feature-based mapping in semi-static environments," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2016.

[11] F. Nobre, C. Heckman, P. Ozog, R. W. Wolcott, and J. M. Walls, "Online probabilistic change detection in feature-based maps," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2018.

[12] P. Biber and T. Duckett, "Dynamic maps for long-term operation of mobile service robots," in *Robotics: Science and Systems*, 2005.

[13] C. Cadena *et al.*, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. on Robot.*, 2016.

[14] L. Schmid *et al.*, "Panoptic multi-TSDFs: A flexible representation for online multi-resolution volumetric mapping and long-term dynamic scene consistency," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2022.

[15] J. Saarinen, H. Andreasson, and A. J. Lilienthal, "Independent markov chain occupancy grid maps for representation of dynamic environment," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst.*, 2012.

[16] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard, "Lifelong localization in changing environments," *The International Journal of Robotics Research*, 2013.

[17] J. Fu, Y. Du, K. Singh, J. B. Tenenbaum, and J. J. Leonard, "NeuSE: Neural SE(3)-equivariant embedding for consistent spatial understanding with objects," in *Robotics: Science and Systems (RSS)*, 2023.

[18] S. Looper, J. Rodriguez-Puigvert, R. Siegwart, C. Cadena, and L. Schmid, "3D VSG: Long-term semantic scene change prediction through 3d variable scene graphs," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2023.

[19] H. Thomas, J. Zhang, and T. D. Barfoot, "The foreseeable future: Self-supervised learning to predict dynamic scenes for indoor navigation," *IEEE Trans. on Robot.*, 2023.

[20] T. Krajník *et al.*, "Warped hypertime representations for long-term autonomy of mobile robots," *IEEE Robot. and Automation Lett.*, 2019.

[21] V. Guizilini, R. Senanayake, and F. Ramos, "Dynamic hilbert maps: Real-time occupancy predictions in changing environments," in *Proc. IEEE Int. Conf. Robot. and Automation*, 2019.

[22] Y. Wang, Y. Fan, J. Wang, and W. Chen, "Long-term navigation for autonomous robots based on spatio-temporal map prediction," *Robotics and Autonomous Systems*, 2024.

[23] T. Deng, H. Xie, J. Wang, and W. Chen, "Long-term visual simultaneous localization and mapping: Using a bayesian persistence filter-based global map prediction," *IEEE Robotics & Automation Magazine*, 2023.

[24] J. Ibrahim, M. H. Chen, and D. Sinha, *Bayesian Survival Analysis*. Springer, 2005.

[25] K. P. Murphy, *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.

[26] H. Akaike, "A new look at the statistical model identification," in *Selected Papers of Hirotugu Akaike*. Springer, 1974.

[27] C. Truong, L. Oudre, and N. Vayatis, "Selective review of offline change point detection methods," *Signal Processing*, 2020.

[28] P. Almeida, L. S. Oliveira, E. Silva Jr, A. Britto Jr, and A. Koerich, "Pklot – a robust dataset for parking lot classification," *Expert Systems with Applications*, 2015.

[29] G. Jocher and J. Qiu, "Ultralytics yolo11," 2024. [Online]. Available: https://github.com/ultralytics/ultralytics

[30] J. Bradbury *et al.*, "JAX: composable transformations of Python+NumPy programs," 2018. [Online]. Available: http://github.com/jax-ml/jax