The Semantic PHD Filter for Multi-class Target Tracking: From Theory to Practice

Jun Chen, Zhanteng Xie, and Philip Dames*

Temple University, Philadelphia, PA 19122, USA

Abstract

In order for a mobile robot to be able to effectively operate in complex, dynamic environments it must be capable of understanding both where and what the objects around them are. In this paper we introduce the semantic probability hypothesis density (SPHD) filter, which allows robots to simultaneously track multiple classes of targets despite measurement uncertainty, including false positive detections, false negative detections, measurement noise, and target misclassification. The SPHD filter is capable of incorporating a different motion model for each type of target and of functioning in situations where the number of targets is unknown and time-varying. To demonstrate the efficacy of the SPHD filter, we conduct both simulated and hardware tests with multiple target types containing both static and dynamic targets. We show that the SPHD filter allows effective tracking of multiple classes of targets even with detection error to some level, and performs better than a collection of PHD filters running in parallel, one for each target class. We also provide a detailed methodology that practitioners can use to fit the probabilistic sensor models necessary to run the SPHD filter.

Keywords: multiple target tracking, semantic tracking, PHD filter, robot learning

1. Introduction

Multi-target tracking is a fundamental problem in robotics, wherein a robot simultaneously estimates the states of a potentially large number of individual objects. These objects can be either static or dynamic, and may leave or enter the search space over time. In addition to tracking the kinematic or dynamic state of each target, as is standard, it is important for robots to be capable of tracking the semantic state, *i.e.,* the type of object. For example, a last-mile delivery robot must be able to recognize various building facilities such as gates, doorways, yards, etc. and localize them as it is moving around a neighborhood in order to decide a safe place to drop a package. Achieving this requires a robot to be equipped with a sensor capable of measuring both geometric information and the semantic label, such as an RGB-D camera. While real-time machine vision algorithms are reaching high levels of object classification accuracy [1, 2, 3, 4], there may still be significant uncertainty in the semantic label and it is often at the expense of high false positive rates.

Multi-target tracking (MTT) algorithms were originally developed to track dynamic objects. The MTT task is challenging due to the difficulty of solving the data association problem (*i.e.,* matching measurements to targets), which is further exacerbated by the possibility of false positive or false negative detections. In computer vision communities, tracking-by-detection techniques have been investigated in recent decades to track multiple objects from a sequence of video frames [5, 6]. However, such methods only work for detecting objects using RGB cameras, and there exist limitations in estimating object states such as three-dimensional pose. Stone et al. [7] discuss in their book a number of probabilistic, multi-target tracking approaches, including the multiple hypothesis tracker (MHT) [8], joint probabilistic data association (JPDA) [9], and the probability hypothesis density (PHD) filter [10]. All of these approaches simultaneously solve the data association and tracking problems in different ways. The MHT makes hard associations and maintains a tree over the history of these associations. This results in unique tracks for each target. However, the number of branches in the tree grows quickly, requiring aggressive pruning algorithms that can lead to suboptimal performance. The JPDA makes soft associations and uses multiple measurements to update each target at each time step, which does not scale well with the number of targets. Finally, the

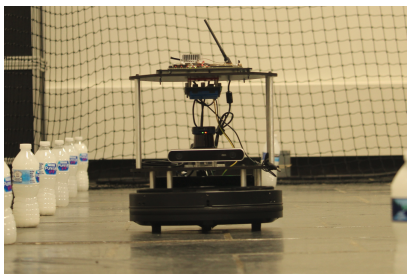


Figure 1: Turtlebot 2 with a depth camera and lidar.

*Corresponding author

Email address: {jchen, zhanteng.xie, pdames}@temple.edu (Jun Chen, Zhanteng Xie, and Philip Dames)

PHD does not require any explicit choice for data association. As a result, the PHD does not actually distinguish between individual objects but rather represents the spatial density of objects. We argue that while it does not work for all tasks that require multiple target tracking, this is sufficient for a great amount of tasks, where it is only important to know what and where objects are but not to distinguish between objects of the same type. For example, navigating through an office environment does not require the robot to know *which* chair it is passing by, only that *a* chair is nearby. However, currently none of these existing trackers are able to utilize semantic measurements to track multiple types.

A number of methods have been provided for labeled tracking with the PHD filter, *i.e.,* uniquely identifying each individual object. Lin et al. [11] proposed a track labeling method by extracting peaks from the estimated PHD and correlating these over time. Vo et al. [12, 13] proposed a multi-target tracking filter using multi-object conjugate priors constructed by labeled RFSs, which is the first RFS-based multi-target filter that produces track-valued estimate in a principled manner. While these methods successfully solved the data association problem and thus realized multi-target labeled tracking, the huge computational load for each target to match its label may not be necessary in many scenarios where targets are classified by labels and labeling within a class is not important, and may decrease the real-time performance, making it challenging to apply in many real-world scenarios.

The PHD filter has been used in other contexts within robotics in the past. Mullane et al. [14] proposed an integrated Bayesian frame-work for SLAM in the general case of uncertain feature number and data association. This approach was then extended to track two types of objects, one static and one dynamic [15, 16], however this lacks the ability to differentiate between different types of static and/or dynamic objects. Dames and Kumar [17] enabled a decentralized team of robot to autonomously explore an environment to detect and localize an unknown number of targets using the PHD filter. Dames [18] later introduced a distributed algorithm for multiple robots to search and track multiple targets in a coordinated manner using the PHD filter.

Note in this paper we consider the task of mapping as a subset of multi-target tracking wherein all targets are stationary. Most existing approaches to robot localization and mapping collect low-level geometric features such as points, lines, and planes [19]. However, by doing this robots only have the ability to plan paths and navigate through a geometrical map, which is different from comprehending the environment the way a human does and navigating from place to place [20]. Recently, other methods for semantic localization (using the same mathematical underpinnings as the PHD filter) [21, 22] and semantic SLAM [23, 24, 25, 26, 27] have been proposed. However, these works assume that all objects in the map are stationary, which is often not the case in complex, real-world environments.

This paper extends the results of our previous conference paper [28], which first introduced the semantic probability hypothesis density (SPHD) filter and provided proof-of-concept experiments in a simulated environment. The key contributions of this work over [28] are: 1) an extended discussion of the SPHD filter, 2) hardware validation in three distinct scenarios, and 3) a detailed guide to experimentally deriving the probabilistic sensor models used within the SPHD filter. The SPHD filter can be easily generalized to a multi-robot setting using the approaches from the authors' past work [17, 18].

2. The Semantic PHD (SPHD) Filter

A robot is tasked with exploring an environment $E \subset R^n$ which, at time t, contains a set of targets $X(t) = \{x_1(t), \ldots, x_n(t)\}$.[1] This set X encodes both the number of targets (*i.e.,* the cardinality of the set $|X(t)|$) and the state of each target (*i.e.,* the elements of the set $x_i(t)$). The key to defining the SPHD filter is to augment the dynamic state of each individual target (*i.e.,* the standard state in tracking algorithms) with a discrete class label, *e.g.,* $x \in \mathcal{X} = R^2 \times C$, where $C = \{c_1, ..., c_k\}$ is a set of discrete class labels. We differentiate these two parts of the single-target state space as the metric part $x^m \in \mathcal{X}^m$ (*e.g.,* $\mathcal{X}^m = R^2$) and the semantic part $x^s \in \mathcal{X}^s = C$, so $\mathcal{X} = \mathcal{X}^m \times \mathcal{X}^s$. Note that within the environment there are multiple targets of multiple classes and there may be multiple targets of each class, so not every target is uniquely identifiable. The number of targets of each class of targets is unknown and may change over time due to the motion of targets into and out of the environment. The number of target classes is not constrained so long as the robot is equipped with a classification algorithm to detect each class. For example, instead of simply using the class "person," as we do in our experiments, a robot could distinguish between people in different states, *e.g.,* "person sitting," "person standing," "person walking," and others.

The pose of the robot at time t is given by $q(t) \in SO(n)$. At each time step, the robot collects a set of local measurements, $Z(t) = \{z_1(t), \ldots, z_m(t)\}$. The number of measurements (*i.e.,* $|Z(t)|$) changes over time due to false positive and false negative detections as well as motion of the robot and targets causing targets to enter and leave the sensor field of view (FoV). Like the target state space, the measurement space also contains a metric and semantic component, $\mathcal{Z} = \mathcal{Z}^m \times \mathcal{Z}^s$. For example, running an object detection algorithm on an RGB image would yield a set of detected objects, each of which includes bearing ($z^m \in \mathcal{Z}^m$) and class label ($z^s \in \mathcal{Z}^s = C$).

[1] For compactness of notation, in the remainder of the paper we will drop the dependence on time except where necessary.

2.1. Random Finite Sets

The sets X and Z from above are realizations of random finite sets (RFSs). An RFS is a set containing a random number of random elements, *e.g.,* each of the n elements x_i in the set $X = \{x_1, \ldots, x_n\}$ is a vector indicating the state of a single target. See [29] for a more thorough treatment of the mathematics presented in this section. In deriving the PHD filter, Mahler [10] assumes that: 1) the clutter and true measurement RFSs are independent and 2) the clutter, target, and birth RFSs are Poisson. The first assumption is standard for target localization tasks. The second assumption is a result of assuming that the number of points in each finite region is independent if the regions do not overlap [30]. A Poisson RFS is one that has independently and identically distributed (i.i.d.) elements and where the number of elements follows a Poisson distribution. The likelihood of such an RFS X is

$$p(X) = e^{-\lambda} \prod_{x \in X} v(x), \tag{1}$$

where $v(\cdot)$ is the *Probability Hypothesis Density* (PHD), $\lambda = \int_E v(x)\,dx$, and $p(X = \varnothing) = e^{-\lambda}$. The PHD is a density function over the state space of the targets, with the unique property that the integral of the PHD over a region $S \subseteq \mathcal{X}$ is the expected cardinality of an RFS X in that region. Note that by introducing this semantic component to the target state, the cardinality of objects both overall and specific to one (or more) classes can be retrieved by conditioning on x^s. The PHD is also the first statistical moment of a distribution over RFSs. Note that it is *not* a probability density function, but it may be turned into one by normalizing by the expected cardinality,

$$p(x) = \lambda^{-1} v(x). \tag{2}$$

2.2. SPHD Models

The (S)PHD filter recursively updates the PHD using models of target motion and the measurement sets collected by the robots. Targets may move about within the environment, may appear in the environment, or may disappear from the environment. Each of these phenomena is explained by a separate target model.

- The **target motion model**, $f(x \mid \xi)$, describes the probability of a target transitioning from an initial state ξ to a new state x. While this may, in theory, allow targets to transition between different classes (*e.g.,* sitting person, standing person, and walking person could be different classes), we do not test this possibility in this paper. Instead, in our experiments we assume there is a collection of class-dependent metric motion models, $f(x^m \mid \xi^m, \xi^s = c), \forall c \in C$, and that the class cannot change over time.

- The **birth model**, $b(x)$, is a PHD that describes both the number and states (including classes) of the new targets entering the environment. For many situations the birth PHD will only be non-zero near the boundaries of the environment, where new targets can enter the area of interest, and only for dynamic objects.

- The **survival probability**, $p_s(x)$, models the survival (and conversely the disappearance) of a target with state x.

The birth and survival models also typically take the form of a collection of class dependent models, *i.e.,* the birth and survival process is different for each class type.

Each robot is equipped with a sensor to detect targets. This sensor may experience false negative detections, return noisy measurements to true targets, or receive false positive detections. Each of these phenomena is covered by a different sensor model.

- The **detection model**, $p_d(x \mid q)$, of a robot with state q detecting a target with state x characterizes the true (and false negative) detections. Note that the probability of detection is identically zero for all x outside the sensor field of view (FoV). In theory the detection likelihood could be different for each class, but in the experiments within this paper we assume that it is independent of class, *i.e.,* $p_d(x \mid q) = p_d(x^m \mid q)$.

- The **observation model**, $g(z \mid x, q)$, returns a measurement z for a target with state x that is detected by a robot with state q. Like the target state space, the measurement also contains two separate parts: the metric part z^m and the semantic part $z^s \in C$. We assume that these two parts are independent conditioned on the target state, so that the observation model becomes:

$$g(z \mid x, q) = g^s(z^s \mid x^s) g^m(z^m \mid x^m, q). \tag{3}$$

  An example of a metric part could be the range and bearing to a target, equivalent to the measurement models in standard non-semantic mapping and tracking tasks. The class part is represented by a confusion matrix which describes the probability of detecting class z^s conditioned on the true class x^s. This takes the form of a confusion matrix where each row matrix represents the instances of the true class while each column represents the instances of the measured class. For example, the entry in row 2 column 4 represent the probability of measuring class 4 given that the true target is of class 2. Mathematically, this is a right stochastic matrix.

- The **false positive** (or clutter) measurements are modeled by the clutter PHD, $\gamma(z \mid q)$, which describes both the number and locations (in measurement space) of the clutter measurements. As with the detection model, in this paper we assume that this is independent of the class, though nothing about

the theory of the SPHD filter requires this to be the case.

These three target models and three sensor models are all necessary to utilize the (S)PHD filter In practice, the user can either specify the models based on experience/intuition or learn models in a data-driven manner, as we have done numerous times in the past [17, 31, 32]. In this paper, we present a detailed example of how to learn these models for a given scenario, including the steps taken to set up the experiment and collect and analyze the data. We have found that obtaining accurate detection and clutter models is essential to obtaining a correct target estimate. In particular, if these models do not accurately reflect the true behavior of the sensor then often the PHD will contain the correct number of peaks but the weight in each peak will not be close to 1. As a result, in practice we have found that counting the number of peaks in the final PHD to be a more reliable estimate of the target number than integrating the PHD.

2.3. SPHD Prediction and Update Steps

Using these target and sensor models from above, the SPHD filter prediction and update equations are:

$$\bar{v}(x,t) = b(x) + \int_E f(x \mid \xi) p_s(\xi) v(\xi, t-1)\, d\xi \tag{4}$$

$$v(x,t) = \big(1 - p_d(x \mid q)\big)\bar{v}(x,t) + \sum_{z \in Z} \frac{\psi_{z,q}(x)\bar{v}(x,t)}{\eta_z\left(\bar{v}(x,t)\right)} \tag{5}$$

$$\eta_z(v) = \gamma(z \mid q) + \int_E \psi_{z,q}(x) v(x)\, dx \tag{6}$$

$$\psi_{z,q}(x) = g(z \mid x, q) p_d(x \mid q), \tag{7}$$

where $\psi_{z,q}(x)$ is the probability of a sensor at q receiving measurement z from a target with state x and η_z is a normalization term. Note that these take the equivalent form to those of the standard PHD filter, except that in our case both the target state space and measurement space include a discrete class label from the set C. The SPHD filter recursively applies (4) and (5) to track the first order statistical moment of RFS for each target.

The addition of the discrete label space offers advantages beyond simply providing a mechanism to track the type of object. Due to the mathematical form of the PHD, the standard PHD filter does not perform well when targets are densely clustered. When a group of targets are close (compared to the sensor noise), all of the targets would appear as one combined peak in the density function rather than being separate discrete peaks. However, the SPHD filter provides a way to separate targets out based on the class label. For example, a person seated on a chair next to a desk in front of a computer could show up as 4 distinct targets with separate class labels in the SPHD filter instead of a single peak of size 4 in the standard PHD filter. Adding in the semantic information will help because it provides a way to separate out targets of different types.

2.4. The Parallel PHD Filters Method

As a point of comparison, we will test the SPHD filter against the standard PHD filter since none of the existing methods provide a combined static/dynamic semantic target solution. In the latter case, we will have multiple PHD filters running in parallel, one for each target class. Each of these separate PHD filters will use the target models for their respective classes. The measured class will be used to funnel the measurements to their respective PHD filters, which will use class-agnostic sensor models. In particular, the observation model g of the standard PHD filter only contains the metric portion g^m. This implies that the filters completely trust the observed class, which is a reasonable assumption when the confusion matrix is close to the identity. Most of the semantic mapping work makes the same assumption.

3. Simulations

We demonstrate the results by a series of simulations using ROS Kinetic running on Ubuntu 16.04. The source code for our SPHD filter implementatio is available at `https://github.com/TempleRAIL/PHD-Filter`. For simplicity, in each simulation we use only one robot to search for a small number of (static and/or dynamic) targets. The robot is a differential drive robot with a maximum linear velocity of 0.4 m/s and a maximum angular velocity of 1.2 rad/s. The PHD is represented by a uniform grid of particles [33] with a resolution of 0.2 m. The initial weight of each particle is identical, meaning that the targets are uniformly likely to be appear in the environment, and the initial number of targets is set to 1.

3.1. OSPA Error

We measure the error between the estimated target set and the true target set using the Optimal SubPattern Assignment (OSPA) metric [34]. The error between two sets X, Y, where $|X| = m \leq |Y| = n$ without loss of generality, is

$$d(X,Y) = \left(\frac{1}{m} \min_{\pi \in \Pi_n} \sum_{i=1}^{m} d_\delta(x_i, y_{\pi(i)})^p + \delta^p (n-m) \right)^{1/p}, \tag{8}$$

where δ is a cutoff distance, $d_\delta(x,y) = \min(\delta, \|x-y\|)$, and Π_n is the set of all permutations of the set $\{1, 2, \ldots, n\}$. OSPA finds the lowest cost assignment, where elements $x \in X$ and $y \in Y$ can be matched only if they are within distance δ of each other. This can be efficiently computed using the Hungarian algorithm [35, 36]. We use $\delta = 10$ m and $p = 1$.

The OSPA error describes the average error in the target positions with a maximum per target error of δ. Given that we use $\delta = 10$ m, when a target is found there is typically a drop in the OSPA of around $10/n$ (if there are n targets), indicating that the error for that target went

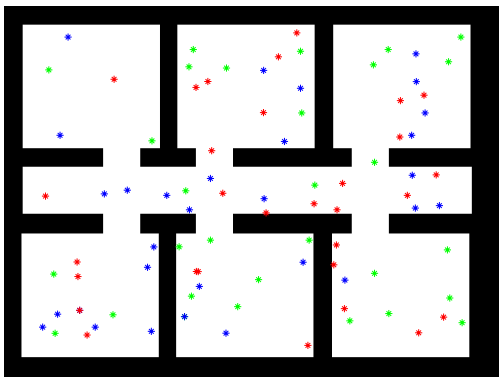


Figure 2: The 40×30 m environment for stationary target search. Markers show an example of target distribution of 3 classes: person (red), chair (green) and table (blue).

from 10 m to around 0 m. Therefore, an OSPA near 0 implies that all targets have been precisely tracked and there are no spurious targets in the estimate.

To extract the estimated target set from our PHD, we take advantage of the grid structure of the PHD. We use a convolution operation to identify the local maxima over a 5×5 grid of particles (1×1 m area) for each type. We then discard any local maxima that do not have a sufficiently high weight. The true target set comes from our a priori knowledge, but is unknown to the robot during operation.

3.2. Stationary Targets

Our first scenario will test the SPHD filter's ability to track multiple types of stationary targets in the simulated office environment shown in Fig. 2. This is representative of an environment where a service robot needs to track multiple classes of targets in order to deliver packages, interact with people, etc. The robot navigates through the environment via a sequence of predefined waypoints, traversing this route twice during each trial (for a total approximate travel time of 1900 s). The set of waypoints ensure that the robot see the entire environment twice to observe the differences between first seeing an object and re-observing it later.

3.2.1. Target Models

There are three classes of targets: person, chair, and table, with 30 targets of each class. All targets are randomly distributed in the environment. Both chairs and table are static, meaning the target motion model is the identity map, the survival probability is unity, and the birth PHD is zero. This was true for both the ground truth motion of the targets and the models used by the robots in the (S)PHD prediction equation (4).

While in general people may enter or leave the environment and may move about within the environment, in this first test all people remain stationary. Thus the ground truth target models are the same as for chairs and tables. However, the models used within the (S)PHD filters are different. The motion model is a truncated Gaussian random walk with spherical covariance matrix with standard deviation 0.01 m per time step (0.1 s). The probability of survival is

$$p_s(x) = \begin{cases} \|x - \partial E\| & \|x - \partial E\| \leq 1\,\text{m} \\ 1 & \text{else} \end{cases} \tag{9}$$

where ∂E is the boundary of the environment. This model says that people within 1 m of the boundary of the environment have a probability of disappearance that is proportional to the distance from the boundary, while all people within the center of the environment always persist from one time step to the next. The birth model is uniform, $b(x) = 1.0 \cdot 10^{-4}$, meaning the location of a new person is uniform and the robot expects 0.12 new people per time step (calculated as b times the area of the environment). When analyzing the results we will examine the effects of the mismatch in the true target models and those used in the SPHD filter.

Table 1: Confusion matrices for different trials.

True \ Observed	Person	Chair	Table
Person	0.9	0.05	0.05
Chair	0.1	0.8	0.1
Table	0.15	0.15	0.7

(a) Confusion matrix 1

True \ Observed	Person	Chair	Table
Person	0.8	0.15	0.05
Chair	0.2	0.7	0.1
Table	0.2	0.25	0.55

(b) Confusion matrix 2

True \ Observed	Person	Chair	Table
Person	0.7	0.15	0.15
Chair	0.15	0.6	0.25
Table	0.25	0.25	0.5

(c) Confusion matrix 3

True \ Observed	Person	Chair
Person	0.9	0.1
Chair	0.1	0.9

(d) Confusion matrix 4

3.2.2. Sensor Models

We assume that the robot carries an RGB-D camera with a forward-facing 120° field of view (FoV) and 5 m

maximum detection range.[2] This sensor returns the range and bearing to each detected target (the metric part z^m) and a class label for each target (the semantic part, z^s). The detection model and clutter of the sensor is shown as

$$p_d(x \mid q) = \begin{cases} 1 - 0.02\|x^m - q\| & x \text{ in FoV} \\ 0 & \text{else} \end{cases} \tag{10}$$

$$\gamma(z \mid q) = 1.5 \cdot 10^{-3} \tag{11}$$

The total expected number of clutter detections per measurement set, found by integrating the clutter PHD over the sensor FoV, is $\int \gamma(z \mid q)\, dz = 0.04$. We assume that the range-bearing (metric) measurement model $g^m(z^m \mid x^m, q)$ follows a multivariate Gaussian distribution with mean $\mu(x, q)$ (the position of the target in the robot's sensor frame) and diagonal covariance Σ (so that the noise of range and bearing measurements are independent). The standard deviation of the range and bearing noise are 0.02 m and 2.0 degrees respectively. We will test several different confusion matrices, $g^s(z^s \mid x^s)$, for these three classes, which are given in Table 1.

To simulate the sensor data in these trials, we directly use these sensor models along with the ground truth data about the robot and targets. To generate a measurement from a true target we start from the relative position of that target with respect to the robot. Using the relative position, we can calculate the detection probability, p_d. We then draw a uniform random number r in the interval $[0, 1]$ and if $r < p_d$ then the target was detected and we can use the measurement model g to generate the measurement that the robot will receive (*i.e.,* draw a vector from the multivariate Gaussian distribution g^m to get z^m and use the true object class along with the confusion matrix to randomly sample z^s). If $r \geq p_d$ then the target is not detected (*i.e.,* there was a false negative). To generate false positive detections we first draw a Poisson random number from a distribution with parameter $\int \gamma(z \mid q)\, dz = 0.04$ to get the number of false positive measurements in the measurement set. Then, for each false positive, we draw the specific measurement using the clutter model γ (in our case, this is uniformly at random from within the sensor field of view).

3.2.3. Results

As we previously mentioned, we compare the SPHD filter, which simultaneously tracks all classes, to a set of parallel PHD filters method, which each track a single target type. For each method we use three different confusion matrices (CM), shown in Tables 2a–2c. We conduct 5 trials for each configuration (SPHD vs. parallel PHD and each confusion matrix). Each trial has a different target distributions. However, the target distributions are the same across different configurations, so, for example, trial 1 using the SPHD filter with CM1 has the same target configuration as trial 1 using parallel PHD filters with CM3. Figure 3 show the average OSPA errors for each class over all 5 trials, and Extension 1 (Table 2) shows a typical simulation run.

Extension	Video Description
1	Simulation of static targets
2	Simulation of mixed static and dynamic targets
3	Hardware test of mixed static and dynamic targets
4	Hardware test of small confusing targets

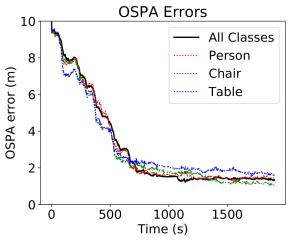
Table 2: Multimedia extensions show videos of experiments using both simulations and hardware.

Figure 3a shows the results of using the SPHD filter with confusion matrix 1 (Table 2a), which has the highest classification accuracy. This is expected as CM1 is the closest to the identity matrix. We see that the OSPA errors decrease in the first half of time since the robot keeps exploring new area in the environment. In the second half of time the robot passes through the environment for the second time, during which time the OSPA error fluctuates slightly due to the appearance of new clutter/missed detections, the correction of previous clutter/missed detections, and classification errors. We can see that all of the classes have a similar OSPA error throughout and that each of these is approximately the same as the class-agnostic OSPA error (All Classes label), which uses only metric information. This indicates that the tracking performance is not limited by classification error, but rather by other phenomena, such as clutter/missed detections or sensor noise. This is despite the fact that each class has only a 70–90% chance of being correctly identified on a per-frame basis.

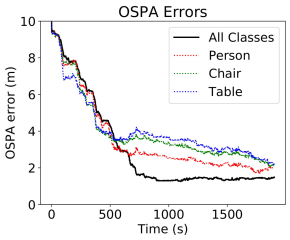
Figure 3d shows the results of the same scenario using the parallel PHD filters. We can see that the OSPA error follows a similar trend, decreasing steadily during the first pass and then leveling out after that. However, the final OSPA error is significantly higher than with the SPHD. From these results we can see that the parallel PHD filters have a much more difficult time dealing with misclassifications.

Figures 3b–3f show the results with the other confusion matrices (Tables 2b and Table 2c), which have significantly higher rates of misclassification. We can see that the class-agnostic OSPA is very similar between all three SPHD tests. This indicates that all of the differences between the class-dependent OSPA lines are likely due to the differences in the confusion matrix. We can see that the SPHD filter follows a similar trend during the first 500 s in every case. After this, we see that the confusion matrix with a higher chance of misclassification has a higher OSPA, a very intuitive result. Despite this, the OSPA continues to steadily, if slowly, decrease (eventually reach-

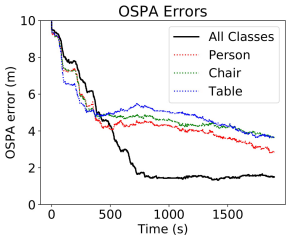
[2] This is the maximum FoV. When working in a known occupancy map, we use ray tracing to determine the true FoV to account for occlusions caused by the static environment. Alternatively, the polygon defined by the lidar detections can be used to limit the FoV to handle occlusions from other obstacles (*e.g.,* people).



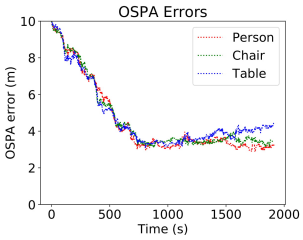
(a) SPHD Filter with CM1



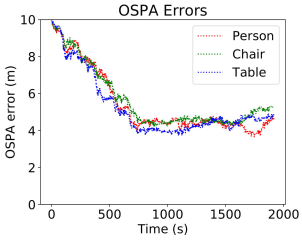
(b) SPHD Filter with CM2



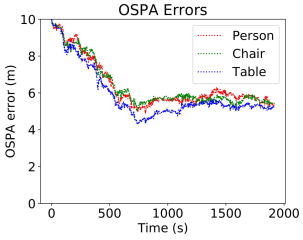
(c) SPHD Filter with CM3



(d) Multiple PHD Filters with CM1



(e) Multiple PHD Filters with CM2



(f) Multiple PHD Filters with CM3

Figure 3: Results of the two methods for stationary target tracking with different confusion matrices (CM) to classify targets. Each figure shows the average OSPA errors over 5 runs of tracking each of the three classes: person, chair and table. We also plot the class-agnostic OSPA error in the case of the SPHD filter. This is not available for the case of multiple PHD filters since there is no single PHD filter for all targets.

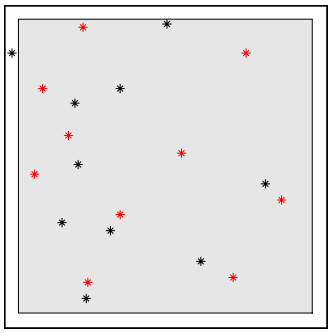


Figure 4: Example of the 20×20 m environment (gray square) with markers showing the distribution of 2 classes: people (black) and chairs (red). Note that people may move anywhere within the 22×22 m area (white square), allowing them to enter and leave the observed environment.

ing the class-agnostic levels in Figure 3b). This indicates that the SPHD filter is able to perform well even with high error rates, provided that it receives sufficient data. On the other hand, the parallel PHD filters do not show this trend. Instead, the OSPA error simply levels out and does not increase or decrease by a significant amount after about 750 s. This indicates that by explicitly including the target class within the state, the SPHD filter is able to benefit from re-observations of the same targets to correct past mistakes. Finally, the OSPA error in the case of the parallel PHD filters fluctuates more wildly than in the case of the SPHD filter. This is likely due to the SPHD filter's superior ability to handle uncertainty in the class of targets.

3.3. Moving Targets

We also want to test the SPHD filter's ability to track a combination of static and mobile targets. In this case, the robot monitors an open 20×20 m environment that originally contains 10 people and 10 chairs. The robot is placed statically in the middle of the environment with a sensor FoV that covers the whole environment. We make this choice because the focus of this work is to demonstrate the capabilities of the SPHD filter, not to develop a control strategy for target search and tracking. Active sensing will be left as future work, perhaps using some of the authors' previous work on target tracking controllers [17, 18].

3.3.1. Target Models

Just like the last test, chairs are stationary in both their ground truth motion and in their SPHD filter motion model. The robot uses the same motion model for people in the SPHD filter as in the static target case. However, instead of standing still, people are continuously moving at 0.3 m/s towards random waypoints, uniformly sampled from a 22×22 m area, shown in Fig. 4. This leaves 1 m outside of the environment for each boundary so that people may occasionally leave or enter the robot's area. When a person reaches their destination, they select a new waypoint and repeat this process. Note that again there is a

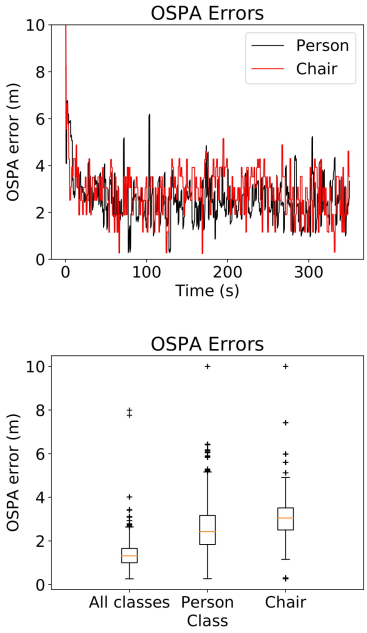


Figure 5: Results of the SPHD filter tracking both dynamic and static targets. Figure 5a shows the OSPA errors of person and chair class over 350 s. Figure 5b shows boxplots of the average OSPA errors of both classes as well as all targets.

discrepancy between the true and modeled motion of the people. In particular, the true velocity of the people is 3 standard deviations from the mean, making this a very challenging tracking task.

3.3.2. Sensor Models

The detection probability is 0.99 in the entire environment. The clutter model, γ, and the metric observation model, g^m, are identical to those from the static target case. Table 2d shows the confusion matrix assumption of the sensor classifying these four classes in this environment.

3.3.3. Results

Extension 2 in Table 2 shows a typical simulation run, with the corresponding OSPA error in Figure 5a. During the 350 s of searching, both static and dynamic targets are well tracked most of time. Compared with the stationary target tracking test using CM1 (Table 2a), where the probability of a person being classified correctly is also 0.9, the OSPA error fluctuates more and the overall OSPA error is a little higher. This is not surprising given that the targets are now moving and there is a larger difference between the true and assumed motion model for the people.

Figure 5b shows boxplots of the average OSPA error over a run, providing a clearer view of the steady-state behavior. We can see that the chair class has a lower standard deviation compared to the person class, which is not surprising given that the true and assumed motion models for the chair match. Also, the median OSPA error is lower

for the person that it is for the chair class, though not significantly. Both of these medians are higher (by about 1) than the class-agnostic OSPA. Given the definition of the OSPA error and the fact that there are 10 targets of each type, this means that the SPHD filter is misclassifying one target of each type.

3.4. Computation time

We conducted our simulations on a workstation equipped with a 3.7 GHz Intel Xeon E3-1240 v6 and 16 GB of RAM and we implemented the SPHD filter in C++ using ROS libraries. In our trials, each recursion of both the PHD filter and the SPHD filter took approximately 5–10 ms per class, depending on the number of particles within the sensor field of view, the number of measurements received, and also on the other processes concurrently running on the computer. We did not see any significant difference between the time per class using the parallel PHD filters versus the single SPHD filter.

Extrapolating from these results, we could expect real-time operation in these scenarios using sensors that receive data at 30–50 Hz. This is at or above the frame rate of most image-based sensors. However, in practice, most processors available onboard mobile robots are less capable than the Xeon that we used. To address this, the PHD update step is highly parallelizable, so one could use multi-threading or GPU-based computation to significantly decrease run time. Finally, deploying our system in hardware requires the use of an image-based classification algorithm. These tend to run more slowly than the SPHD filter updates, and thus we do not expect the SPHD filter to be the computational bottleneck in the perception and estimation pipeline.

4. Hardware Validation

We conduct semantic mapping tests using hardware to validate our proposed state estimation algorithms. We choose to use a Kobuki Turtlebot 2, shown in Fig 1, which has a maximum velocity 0.5 m/s and is equipped with a Jetson TX2 embedded computer, a Hokuyo UTM-30LX lidar, and a Stereolabs ZED stereo camera. The Hokuyo lidar has a maximum range of 30 m, a FoV of 270°, and an angular resolution of 0.25°. The ZED camera has a minimum detection range of 0.5 m, a maximum range 10 m, and a FoV of 90°. The robot localizes itself using the `amcl` ROS package, which requires a pre-built map of the environment (that does not include the targets) and the lidar.

4.1. Measurement Extraction

The ZED stereo camera is used for object detection. First, the raw RGB image data from the ZED camera is passed into the YOLOv3 detection algorithm [1] to find the object classes and labeled bounding boxes for each object. With the bounding boxes of each object, we then extract the pixel coordinates of a small square area (*i.e.,* 25 pixel points) around the center pixel point of this object. Next, we use the pixel coordinates to extract the position measurements of these 25 pixel points from the 3-D point cloud data and average the positions to get the resulting position estimate.

There are three things should be noted. First, there are two primary sources of clutter detections, misclassified objects and multiple detections (*e.g.,* the reflection of an object on glass or the polished floor), which causes multiple simultaneous bounding boxes for a single object. Second, if the object is too close to or too far away from the robot then may be no data associated with it in the point cloud. So even if an object is detected by YOLOv3 algorithm, we cannot always generate a full measurement of it. Third, the performance of the image processing pipeline depends on the level of illumination in the environment. In our trials we train and test a model in the same ambient conditions. For long term operation, where the robot may experience multiple illumination levels, one solution would be to use the procedure outlined below in Sec. 4.3.1 to learn several different sensor models for different ambient conditions. Then the robot can use the observed illumination levels to determine which model to use at any given instant. This would not affect the theory behind the SPHD filter, but would require additional engineering work.

4.2. Semantic Mapping With Large Objects

We first test semantic mapping using the SPHD filter in the lobby of the Temple University College of Engineering building, shown in Fig 6a. We performed two test cases in this environments. In both of these cases we use the default YOLOv3-tiny model trained from Microsoft COCO dataset [37] to detect objects. The robot navigates through a series of user-defined way points using the standard ROS navigation stack [38].

4.2.1. Stationary Targets

The first test case has three classes of static objects: chairs (single sofas), tables, and umbrellas. We select these classes due to their large sizes and distinct shapes, making them reliable to detect and easy to identify. We use class-agnostic sensor models

$$p_d(x \mid q, \theta) = \begin{cases} 0.9 & 3\,\text{m} \le r(x,q) \le 5\,\text{m} \\ & \text{and } |b(x,q) - \theta| \le 45^\circ \\ 0 & \text{else} \end{cases}, \quad (12)$$

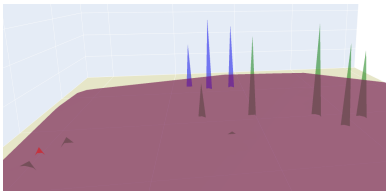
$$g(r, b \mid x, q, \theta) = \mathcal{N}(r, b \mid x, 0.1 I_2), \quad (13)$$

$$c(z \mid q, \theta) = 0.01. \quad (14)$$

We also assume that the class recognition is always correct, which is equivalent to the confusion matrix being the identity matrix. This assumption was backed up in our trials, where we did not experience any misclassification.

The robot drove through the loop within the area marked in green Fig 6a, which took approximately 1 minute. Figure 6b shows the resulting PHD estimate. We see that all

(a) Lobby–Ground Truth



(b) Lobby–PHD

Figure 6: Fig 6a: Ground truth map in the Temple University College of Engineering building lobby map. There are three classes of objects: dining tables (light gray rectangles), chairs (single sofas) and umbrellas. Fig 6b: The resulting semantic map, with the PHD for dining tables (red), chairs (green), and umbrellas (blue).

eight chairs and three umbrellas have been localized and correctly identified, as indicated by the green and blue PHD peaks. However, only one out of three tables was localized, since the other two were occluded by chairs and could not be recognized by YOLOv3. Both object locations and classes are correct compared with the ground truth, indicating a successful semantic mapping of the environment based on objects it can find.

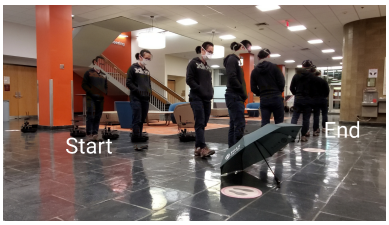
4.2.2. Mixed Stationary and Dynamics Targets

The second test has two target classes: umbrellas and people. In the test, the person walks through an L-shaped path (at approximately 0.2 m/s), shown in Fig. 7a, with the umbrella placed at the turning point. The robot is manually controlled to follow the person as they walk from the start location to the end location.

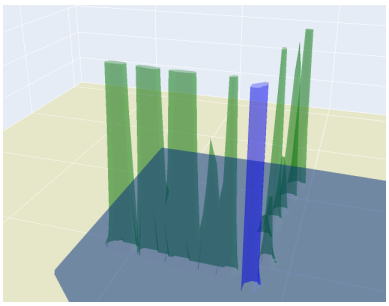
Extension 3 in Table 2 shows a video of the trial, with the cumulative PHD over the time interval given in Figure 7b. We see that as the person moves, the sequence of (green) peaks is consistent with the true trajectory of the person. We can also see that the umbrella is correctly localized and that both classes are consistently classified, as shown by peaks with separate colors. This test demonstrates that the SPHD filter is able to track both static and moving targets while distinguishing their class labels.

4.3. Semantic Mapping With Small Confusing Objects

The final hardware test stresses the system's ability to distinguish between target classes. There are four classes of objects, shown in Fig 8, each a distinct brand of 500 ml water bottle. The bottles are all of very similar size, shape, and coloration, making distinguishing between them very



(a) Person and Umbrella–Ground Truth



(b) Person and Umbrella–PHD

Figure 7: Ground truth of a moving person from a start point to an end point and a stationary umbrella and the result of semantic mapping. The person is walking through an L-shape trajectory and the robot is following him, keeping him inside its field of view at all time, as shown in Fig 7a. In Fig 7b, the cumulative PHD of the person over time is shown as green peaks and the blue peak shows the cumulative PHD of the umbrella.

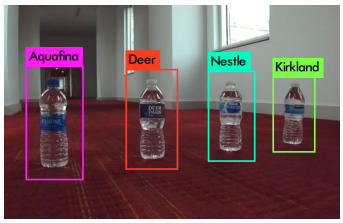


Figure 8: Four brands of water bottles.

challenging. In other words, we expect the confusion matrix to be far from the identity matrix. Note: to improve the ability of pose measurement using point cloud, we dissolve condensed milk in all bottled waters to make them opaque.

4.3.1. Sensor Characterization

In our previous tests, the sensor models were all based on heuristics and rough observation. In these trials, we can no longer do this, especially given that the target classes are easily confused with one another. Instead we will demonstrate how to experimentally determine the sensor models, providing a template for other researchers who wish to use the SPHD filter.

Experimental Setup. In general, the sensor characterization procedure only requires one dataset with known target classes and positions to train the object detector, and develop the sensor models necessary to utilize the PHD filter. However, in order to facilitate the acquisition of a

diversified detection dataset, we collect two datasets in this experiment: one called detection dataset to train the object detector, and one called measurement-model dataset to learn the observation, detection, and clutter models.

To collect our detection dataset, the robot drove around the lobby and lab environment in the College of Engineering building at Temple University for 30 min, collecting around 5,000 RGB images of 48 water bottles (12 of each brand) at random positions. We then manually label these images using the LabelImg tool[3] with the Pascal VOC dataset format. Figure 8 shows an example of a labeled image. With this detection dataset, we train a custom YOLOv3-tiny model using the pre-trained YOLOv3-tiny model to bootstrap the learning procedure.

To collect our measurement-model dataset, we drove the robot (running our YOLOv3 algorithm) around the lobby environment for 20 min to collect measurements of 12 targets (3 of each brand) at known positions. We again manually label the dataset using the same procedure to get the ground truth data. The resulting dataset contained 2725 measurement sets with a total of 3492 individual measurements (*i.e.,* YOLOv3 detection bounding boxes with accompanying point cloud data). We also have to determine the measurement-to-target association in order to label each measurement as coming from a target or clutter and to count the false negative detections. Note, while we do have the ground truth positions of all water bottles, the robot has some (small) uncertainty in its estimated pose from `amcl`. Therefore, the sensor models (specifically the covariance in the metric observation model) will encapsulate both these small errors in robot localization as well as the errors in the relative range/bearing from the robot to the target. To account for both of these sources of uncertainty, we use a gating procedure to associate measurements, wherein a measurement is associated with a target if it is within the sensor field of view and within three standard deviations of a target (using the estimated robot pose as if it were ground truth). In this gating procedure, we also account for the fact that the SPHD filter assumes at most one measurement per target by associating only the closest measurement to a target (*e.g.,* in the case of multiple bounding boxes per target). If no measurement is associated to a target, the target is labeled as a false negative detection. We use this to derive the three sensor models for the SPHD filter.

Observation Model. As we discussed in the definition of the SPHD filter, the sensor observation model (3) is composed of separate semantic and metric components. The semantic part is modeled by a confusion matrix, which we can directly calculate from our detection dataset using the ground truth and estimated labels. Table 3 shows the resulting confusion matrix. We see that Aquafina is the most easily recognized, which makes sense given that it has the most distinct shape of the four brands. We also see that there are some significant sources of ambiguity, with the Nestle brand only being correctly classified 71.6% of the time. Note that in practice we avoid having 0 entries in the confusion matrix by adding a small number to each entry in the confusion matrix (0.001) and then normalizing each row to ensure that it remains a right stochastic matrix.

True \ Observed	Aquafina	Deer	Kirkland	Nestle
Aquafina	0.993	0	0	0.007
Deer	0	0.940	0.042	0.018
Kirkland	0	0	0.850	0.150
Nestle	0	0.252	0.032	0.716

Table 3: Confusion matrix for bottles.

The metric component of our observation model is the range and bearing of each detected target. We assume that these range and bearing measurements are corrupted by zero-mean Gaussian noise with covariance Σ, which is independent of the robot pose, target class, and the range and bearing to the target. Mathematically, this is:

$$g^m(z^m \mid x^m; q) = \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} e^{-\frac{1}{2}(z^m-\mu)^T\Sigma^{-1}(z^m-\mu)} \quad (15)$$

where $\mu = \left[r(x,q), b(x,q)\right]^T$, $r(x,q)$ and $b(x,q)$ are the target range and bearing in the local sensor frame, $\Sigma = \text{diag}(\sigma_r^2, \sigma_b^2)$, and σ_r and σ_b are measurement noise parameters.

To find the covariance values, we use our measurement-model dataset. As we stated above, this covariance value is used in the gating procedure to determine the number of correct associations. Since no accurate ground-truth localization data is available in the experimental environment, we fit the noise parameter Σ by performing a grid search over σ_r (0 to 0.50 m) and σ_b values (0 to 10.00°). For each point, we compute both the fraction of measurements (Frac) considered true detections (using the gating procedure) and the sum-of-squares error (SSE) between the measurements and ground truth data. We seek to find the parameter values that maximizes the number of true detections (*i.e.,* higher Frac) while minimizing the SSE between them.

Figure 9 shows the resulting values for our dataset. We see that the SSE data mostly depends upon the bearing noise, increasing in a roughly linear fashion as a function of σ_b. Therefore, minimizing σ_b will minimize the SSE. On the other hand, the Frac data sharply increases for small σ_r and σ_b values, reaching a plateau near $\sigma_r = 0.25$ m and $\sigma_b = 3.00°$. Therefore, we select this point as our optimal measurement noise. We also will use the associations resulting from this gate to fit the detection and clutter models. Using these values, the measurement dataset contains 910 true detections and 1581 false negative detections.

[3] https://github.com/tzutalin/labelImg

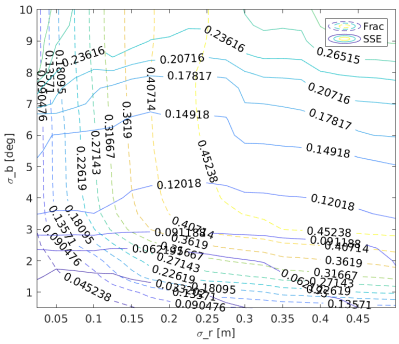
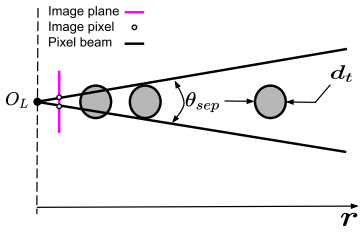
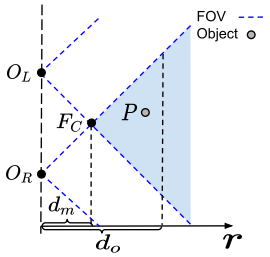


Figure 9: The grid search contour figure where the Frac and SSE as a function of the measurement noise parameters σ_r and σ_b.



(a) Pixel imaging



(b) Binocular depth estimation

Figure 10: Geometric pictograms of the stereo camera detection model, where O_L/O_R are the left/right aperture centers, F_C is the intersection point of binocular filed of view, and r is the range. (a) Pixel imaging pictogram, where d_t is the diameter of the target, θ_{sep} is the angular separation between adjacent pixel beams. (b) Binocular depth estimation pictogram, where the light blue shaded area staring from F_C is the binocular field cone, d_m is the minimum working distance for the depth estimation, and d_o is optimal working distance for the depth estimation.

Detection Model. As previously noted, one valid detection includes not only the object class but also its position. In the other word, the detection model depends on the object class detection from raw RGB image data and the object position estimation from 3-D point cloud data. Therefore, our detection model must account for the physics of both pixel imaging and binocular depth estimation, as Fig. 10 shows.

From the physics of a stereo camera, we can see that a suitable range r between the camera and the object is important to the object detection and that when r is too far, pixel imaging limits the object class detection performance since there are not enough pixels to accurately recognize objects. Geometrically, each pixel beam in an image intersects a target that is within $\frac{d_t}{2}$ of the beam, where d_t is the target diameter, while the arc length between two beams emitted from two adjacent pixels (separated by angle θ_{sep}) of a image sensor at a range r is $r\theta_{\text{sep}}$. This yields the width (in pixels) of an object at distance r of $\frac{d_t}{r\theta_{\text{sep}}}$. We assume that the detection probability at high ranges is limited by this value, and that the two quantities are proportional.

From the principle of binocular depth estimation, only the targets within the binocular field cone can have binocular disparity and depth estimate. The minimum working distance to the binocular field cone is d_m, while the optimal working distance which offers the best depth accuracy is d_o. When r is too close, binocular depth estimation limits the object position estimation performance. We assume that the detection probability is independent of the bearing and that it is proportional to the relative distance from the optimal value, given by $\frac{r-d_m}{d_o}$.

These two phenomena result in the following detection model:

$$p_d(x;q) = \max\left(\min\left(p_{d,b}\frac{r-d_m}{d_o}, p_{d,p}\frac{d_t}{r\theta_{sep}}\right), 0\right) \times \mathbb{1}(b \in [b_{\min}, b_{\max}])\, \mathbb{1}(r \in [r_{\min}, r_{\max}]) \quad (16)$$

where $\mathbb{1}(\cdot)$ is an indicator function, $r = r(x,q)$ and $b = b(x,q)$ are the target range and bearing in the local sensor frame and $p_{d,b}$, $p_{d,p}$ are the proportionality constants (for the binocular and pixel imaging phenomena, respectively) to be fit. To get a reliable detection probability, the target range and bearing are limited to fall within $[r_{\min}, r_{\max}]$ and $[b_{\min}, b_{\max}]$ respectively. For our sensor and target configuration, $d_m = 0.5\,\text{m}$, $d_o = 1\,\text{m}$, $d_t = 6.67\,\text{cm}$, $\theta_{sep} = 0.07°$, $b_{\max} = -b_{\min} = 45°$, and $r_{\min} = 0.50\,\text{m}$, $r_{\max} = 1.50\,\text{m}$.

To find the optimal parameter values, we use the associated data from the previous step. We then bin this labeled data as a function of the true range to the target (in 0.2 m increments) computing the probability of detecting a target within each range bin using the true and false negative detection data. Similar to modeling the observation model, we use a grid search algorithm to search over $p_{d,b}$ (0 to 1 in steps of 0.01) and $p_{d,p}$ (0 to 0.1 in steps of

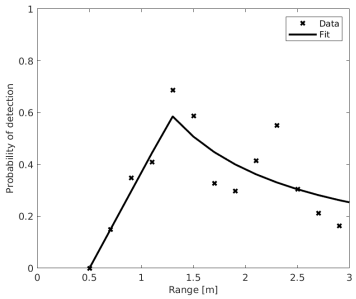


Figure 11: Best fit detection model to 910 true detections, and 1581 false negative detections, from 2725 measurement sets.

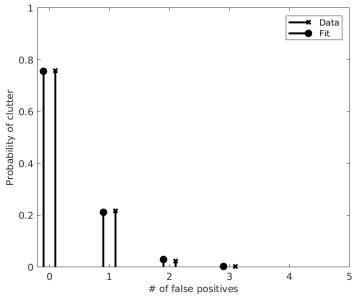


Figure 12: Best fit model for the clutter cardinality, using 734 clutter measurements from 2725 measurement sets.

0.001). For each parameter configuration, we compute the sum-of-squares error between the data and the parameterized model, with the best fit model parameters being

$$p_{d,b} = 0.74, \qquad p_{d,p} = 0.014.$$

Figure 11 shows the resulting model against the data.

Clutter Model. Based on the underlying assumptions of the SPHD filter, the clutter cardinality is assumed to follow a Poisson distribution with mean μ (where $\mu = \int c(z)\,dz$). Therefore, to find μ we can fit a Poisson distribution to the empirical distribution of the number of clutter measurements per measurement set $Z(t)$. Figure 12 shows the resulting cardinality model, where the optimal $\mu = 0.28$, meaning there are an average of 0.28 clutter detections per measurement set.

As we noted when describing the measurement extraction procedure, clutter (*i.e.,* false positive) measurements arise due to clutter items (*e.g.,* trash cans and reflective glass) and redundant detections. To avoid over fitting our clutter model to a specific environment, we assume that clutter detections are uniformly distributed with the sensor's field of view and that they are independent of the robot pose:

$$c(z) = \left[\frac{\mu}{\frac{1}{2}(b_{\max} - b_{\min})(r^2_{\max} - r^2_{\min})}\right] \times \mathbb{1}(b(x,q) \in [b_{\min}, b_{\max}])\ \mathbb{1}(r(x,q) \in [r_{\min}, r_{\max}]) \tag{17}$$

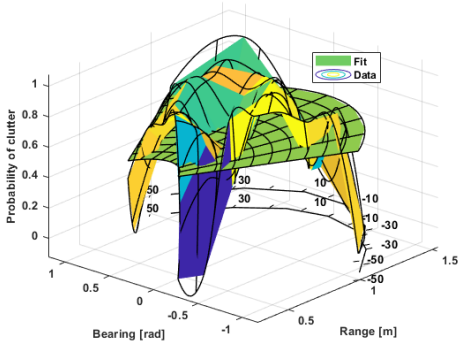


Figure 13: Best fit clutter probability density function, using 734 clutter measurements from 2725 measurement sets.

where $\mathbb{1}(\cdot)$ is an indicator function, and μ is the expected number of clutter measurements per scan. Figure 13 shows the best fit model against the empirical distribution of clutter (*i.e.,* the histogram of clutter detections). We see that the model is close except near the very corners of the field of view.

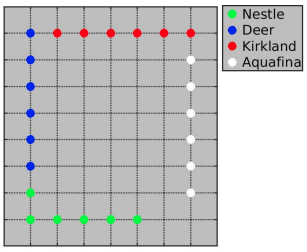
4.3.2. Results

While we used both the lobby and lab environments to collect the measurement dataset, we use only the lab environment, shown in Fig. 14a, to test the resulting SPHD filter models. The robot starts in the lower right corner and traverses one clockwise loop within the rectangular environment. Figure 14b shows the ground truth class labels, where there are 6 bottles of each class.

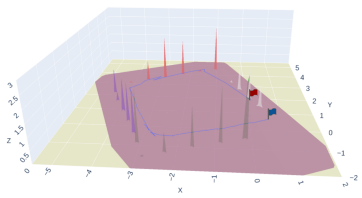
Extension 4 in Table 2 shows a video of the trial, including the first-person view from the robot and the estimated SPHD. Figure 14c shows the robot's trajectory (the flags and blue curve) and the final PHD for each object class. While most of the peaks are apparent to see, a few of them are too small to view (indicating that the cardinality count is low). Even though the peaks are small, they are still present and we can more easily see them if we run our peak extraction method. Figure 14d shows the locations of all extracted peaks, where we can see that all bottles have been correctly classified and the estimated locations coincide with the true locations shown in Fig 14b. There is only one clutter target (of class Kirkland) in the upper right corner, but the cardinality of this is near zero. If the robot had completed multiple circuits this would have likely disappeared.

Figure 15 demonstrates the process of object state estimation as the robot drives in the environment at four points along the robot's trajectory. At each point, we show the robot camera view with current object detections and the PHD. We can see that multiple bottles are misclassified within each frame, with the class errors rates coinciding closely with the confusion matrix in Table 3. Despite this, the prominent peaks within the PHD are correct, with the height of each peak tending to grow towards 1 as the

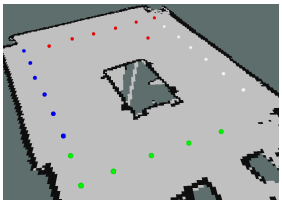
(a) Test 2 Environment



(b) True States of Bottles



(c) Final PHD



(d) Estimated States of Bottles

Figure 14: True and estimated states of bottles. Figure 14a and Figure 14b show the environment setup and the true positions and brands of bottles respectively. Each grid in Fig 14b is $0.7\,\mathrm{m} \times 0.7\,\mathrm{m}$. Figure 14c shows robot's trajectory, marked by the blue line with the start point (blue flag) and the end point (red flag), and the final PHD of different classes of bottles, marked by colored surfaces in 3D environment. Figure 14d shows a Rviz screenshot of estimated positions and brands of bottles in the environment extracted from the PHD peaks. In all above figures, green, blue, red and white correspond to Nestle, Deer, Kirkland and Aquafina respectively.

target was observed multiple times. We also saw many false positive targets appear during the process, such as the red peak in Fig 15d, due to the noisy point cloud measurement. However, the resulting PHD only had a single clutter target with weight near 0.

While we only demonstrate a single run, this performance was typical, with correct classification and localization of the true targets and the occasional clutter target. One challenge that we will address in future work is the temporal variance in our sensor model, specifically due to changes in illumination causing variations in performance. For example, the error rate of bottle classification increases as the environment gets darker. Similarly, the point cloud pose measurement accuracy also varies with illumination levels. Despite this, the SPHD filter still generally performed well. We expect that training the classification and sensor models using a wider range of data will alleviate, but not completely eliminate, this issue. Another direction of future work is to learn (or adapt) the sensor model parameters online.

5. Conclusion

In this paper we propose the semantic PHD filter algorithm, a RFS-based multi-target tracking algorithm which uses both metric and semantic information to simultaneously track multiple classes of targets. Mathematically, the key to defining the SPHD filter is to augment both the target state space and the measurement space with a discrete set of class labels. The various target and sensor models within the standard PHD filter framework utilize this additional label state to differentiate between target types. Some models, like the target motion model, are defined separately for each individual class while others, like the observation model, must contain a single model for all target types. Using these models, the SPHD filter can then iteratively propagate the PHD and handle uncertainty, such as the possibility of target misclassification, in a theoretically principled manner.

To demonstrate the efficacy of the SPHD filter, we conduct a series of experiments in both simulated and real environments and using a mixture of static and dynamic targets. In our simulated tests, we found that the SPHD filter outperforms a system that utilizes multiple standard PHD filters (one for each class) in parallel. In particular, the SPHD filter demonstrates an ability to recover from prior misclassifications, even when the probability of correct classification is barely over 50%. In our hardware tests, we demonstrated the ability of the SPHD filter to work in real-world settings, including in settings where the correct classification rate for an object is barely 70%. We also provided a template for how to carefully characterize the sensor models used within the SPHD filter, from data collection and labeling to find optimal sensor models. While the details of this procedure were specific to our sensor and targets, the same general process can be used in any other setting with the SPHD (or PHD) filter. Directions for future work include using multiple robots to search for and track multi-class targets in a coordinated manner and applying active control algorithms to enable one or more robots to search for and track dynamic targets.

6. Acknowledgement

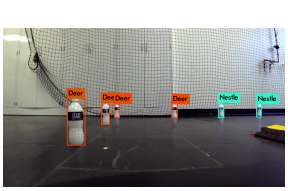
This work was supported by NSF Grant IIS-1830419; and the Amazon Research Awards program. The authors would like to thank Zhijia Chen from Temple University for assistance with the data processing work. The authors would also like to thank Olivia Chaves, Brandon Lutz, and Sacksit Vilayhong for their assistance in labeling the data used to train the water bottle detector.

References

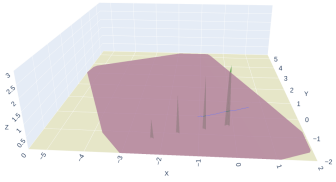
[1] J. Redmon, A. Farhadi, Yolov3: An incremental improvement, arXiv preprint arXiv:1804.02767.

[2] H. Zender, O. M. Mozos, P. Jensfelt, G.-J. Kruijff, W. Burgard, Conceptual spatial representations for indoor mobile robots, Robotics and Autonomous Systems 56 (6) (2008) 493–502.

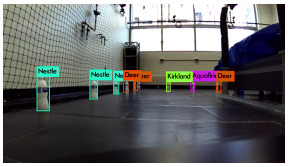
[3] A. Pronobis, P. Jensfelt, Large-scale semantic mapping and reasoning with heterogeneous modalities, in: 2012 IEEE International Conference on Robotics and Automation, IEEE, 2012, pp. 3515–3522.



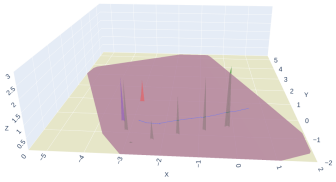
(a) Robot View at Position 1



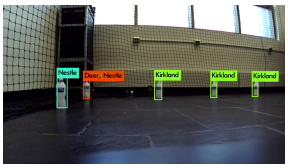
(b) PHD at Position 1



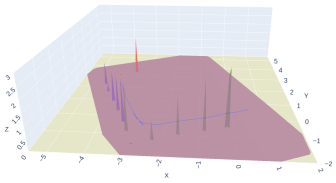
(c) Robot View at Position 2



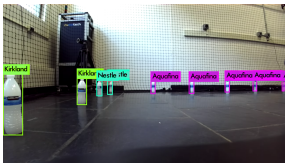
(d) PHD at Position 2



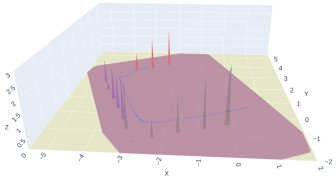
(e) Robot View at Position 3



(f) PHD at Position 3



(g) Robot View at Position 4



(h) PHD at Position 4

Figure 15: Robot view and PHD during process. In PHD figures, green, blue, red and white correspond to Nestle, Deer, Kirkland and Aquafina respectively.

[4] A. Nüchter, J. Hertzberg, Towards semantic maps for mobile robots, Robotics and Autonomous Systems 56 (11) (2008) 915–926.

[5] W. Choi, Near-online multi-target tracking with aggregated local flow descriptor, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 3029–3037.

[6] Z. Wang, L. Zheng, Y. Liu, S. Wang, Towards real-time multi-object tracking, arXiv preprint arXiv:1909.12605 2 (3) (2019) 4.

[7] L. D. Stone, R. L. Streit, T. L. Corwin, K. L. Bell, Bayesian multiple target tracking, Artech House, 2013.

[8] S. S. Blackman, Multiple hypothesis tracking for multiple target tracking, IEEE Aerospace and Electronics Systems Magazine 19 (1) (2004) 5–18.

[9] T. Fortmann, Y. Bar-Shalom, M. Scheffe, Sonar tracking of multiple targets using joint probabilistic data association, IEEE Journal of Oceanic Engineering 8 (3) (1983) 173–184.

[10] R. Mahler, Multitarget Bayes filtering via first-order multitarget moments, IEEE Transactions on Aerospace and Electronic Systems 39 (4) (2003) 1152–1178.

[11] L. Lin, Y. Bar-Shalom, T. Kirubarajan, Track labeling and PHD filter for multitarget tracking, IEEE Transactions on Aerospace and Electronic Systems 42 (3) (2006) 778–795.

[12] B.-T. Vo, B.-N. Vo, Labeled random finite sets and multi-object conjugate priors, IEEE Transactions on Signal Processing 61 (13) (2013) 3460–3475.

[13] B.-N. Vo, B.-T. Vo, D. Phung, Labeled random finite sets and the Bayes multi-target tracking filter, IEEE Transactions on Signal Processing 62 (24) (2014) 6554–6567.

[14] J. Mullane, B.-N. Vo, M. D. Adams, B.-T. Vo, A random-finite-set approach to Bayesian SLAM, IEEE Transactions on Robotics 27 (2) (2011) 268–282.

[15] D. Moratuwage, D. Wang, A. Rao, N. Senarathne, H. Wang, RFS collaborative multivehicle SLAM: SLAM in dynamic high-clutter environments, IEEE Robotics & Automation Magazine 21 (2) (2014) 53–59.

[16] C. S. Lee, D. E. Clark, J. Salvi, SLAM with dynamic targets via single-cluster PHD filtering, IEEE Journal of Selected Topics in Signal Processing 7 (3) (2013) 543–552.

[17] P. Dames, V. Kumar, Autonomous localization of an unknown number of targets without data association using teams of mobile sensors, IEEE Transactions on Automation Science and Engineering 12 (3) (2015) 850–864.

[18] P. M. Dames, Distributed multi-target search and tracking using the PHD filter, Autonomous Robots`doi:10.1007/s10514-019-09840-9`.

[19] S. Thrun, J. J. Leonard, Simultaneous localization and mapping, Springer Handbook of Robotics (2008) 871–889.

[20] I. Kostavelis, A. Gasteratos, Semantic mapping for mobile robotics tasks: A survey, Robotics and Autonomous Systems 66 (2015) 86–103.

[21] N. Atanasov, M. Zhu, K. Daniilidis, G. J. Pappas, Semantic localization via the matrix permanent, in: Robotics: Science and Systems, Vol. 2, 2014, pp. 1–10.

[22] A. Asgharivaskasi, N. Atanasov, Active bayesian multi-class mapping from range and semantic segmentation observation, arXiv preprint arXiv:2101.01831.

[23] C. Bahlmann, Y. Zhu, D. Comaniciu, T. Köhler, M. Pellkofer, Method for combining boosted classifiers for efficient multi-class object detection, uS Patent 7,769,228 (Aug. 3 2010).

[24] R. F. Salas-Moreno, R. A. Newcombe, H. Strasdat, P. H. Kelly, A. J. Davison, SLAM++: Simultaneous localisation and mapping at the level of objects, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 1352–1359.

[25] S. Y. Bao, S. Savarese, Semantic structure from motion, in: CVPR 2011, IEEE, 2011, pp. 2025–2032.

[26] D. Gálvez-López, M. Salas, J. D. Tardós, J. Montiel, Real-time monocular object SLAM, Robotics and Autonomous Systems 75 (2016) 435–449.

[27] S. L. Bowman, N. Atanasov, K. Daniilidis, G. J. Pappas, Probabilistic data association for semantic SLAM, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 1722–1729.

[28] J. Chen, P. Dames, Multi-class target tracking using the semantic PHD filter, in: 2019 International Symposium Robotics Research (ISRR), 2019, pp. 1–16.

[29] R. Mahler, Statistical multisource-multitarget information fusion, Vol. 685, Artech House Boston, 2007.

[30] D. J. Daley, D. Vere-Jones, An introduction to the theory of point processes, Vol. 1, Springer, 2003.

[31] P. Dames, V. Kumar, Experimental characterization of a bearing-only sensor for use with the PHD filter, arXiv preprint arXiv:1502.04661.

[32] P. Dames, P. Tokekar, V. Kumar, Detecting, localizing, and tracking an unknown number of moving targets using a team of mobile robots, The International Journal of Robotics Research 36 (13-14) (2017) 1540–1553. `doi:10.1177/0278364917709507`.

[33] B.-N. Vo, S. Singh, A. Doucet, et al., Sequential Monte Carlo implementation of the PHD filter for multi-target tracking, in: Proc. Int'l Conf. on Information Fusion, 2003, pp. 792–799.

[34] D. Schuhmacher, B.-T. Vo, B.-N. Vo, A consistent metric for performance evaluation of multi-object filters, IEEE Transactions on Signal Processing 56 (8) (2008) 3447–3457.

[35] H. W. Kuhn, The Hungarian method for the assignment problem, Naval Research Logistics Quarterly 2 (1-2) (1955) 83–97.

[36] J. Munkres, Algorithms for the assignment and transportation problems, Journal of the Society for Industrial and Applied Mathematics 5 (1) (1957) 32–38.

[37] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C. L. Zitnick, Microsoft COCO: Common objects in context, in: European Conference on Computer Vision, Springer, 2014, pp. 740–755.

[38] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, K. Konolige, The office marathon: Robust navigation in an indoor office environment, in: 2010 IEEE International Conference on Robotics and Automation, IEEE, 2010, pp. 300–307.