Curating Long-Term Vector Maps

Samer Nashed[1] and Joydeep Biswas[2]

Abstract— Autonomous service mobile robots need to consistently, accurately, and robustly localize in human environments despite changes to such environments over time. Episodic non-Markov Localization addresses the challenge of localization in such changing environments by classifying observations as arising from Long-Term, Short-Term, or Dynamic Features. However, in order to do so, EnML relies on an estimate of the Long-Term Vector Map (LTVM) that does not change over time. In this paper, we introduce a recursive algorithm to build and update the LTVM over time by reasoning about visibility constraints of objects observed over multiple robot deployments. We use a signed distance function (SDF) to filter out observations of short-term and dynamic features from multiple deployments of the robot. The remaining long-term observations are used to build a vector map by robust local linear regression. The uncertainty in the resulting LTVM is computed via Monte Carlo resampling the observations arising from long-term features. By combining occupancy-grid based SDF filtering of observations with continuous space regression of the filtered observations, our proposed approach builds, updates, and amends LTVMs over time, reasoning about all observations from all robot deployments in an environment. We present experimental results demonstrating the accuracy, robustness, and compact nature of the extracted LTVMs from several long-term robot datasets.

I. INTRODUCTION

Long-term autonomy is an essential capability for service mobile robots deployed in human environments. One challenging aspect of long-term autonomy in a human environment is robustness to changes in the environment.

Many approaches have been proposed to reason about a changing environment, including estimating the latest state of the environment [18], [13], estimating different environment configurations [2], [6], or modeling the dynamics of the environment [5], [16], [17], [1], [19]. However, in large environments, it may not be feasible to make the periodic observations required by these approaches. Therefore, we model observations in human environments as arising from three distinct types of features [4]: Dynamic Features (DFs) or moving objects such as people or carts, Short- Term Features (STFs) or movable objects such as tables or chairs, and Long-Term Features (LTFs) which persist over long periods of time, such as office walls or columns. Episodic non-Markov Localization (EnML) [4] simultaneously reasons about global localization information from LTFs, and local relative information from STFs. A key requirement to EnML is an estimate of the Long-Term Vector Map: the features in the environment that persist over time, represented in line segment or vector form (Fig. 1).

[1]Samer Nashed and [2]Joydeep Biswas are with the College of Information and Computer Sciences, University of Massachusetts, Amherst, MA 01003, USA. Email: {snashed, joydeepb}@cs.umass.edu
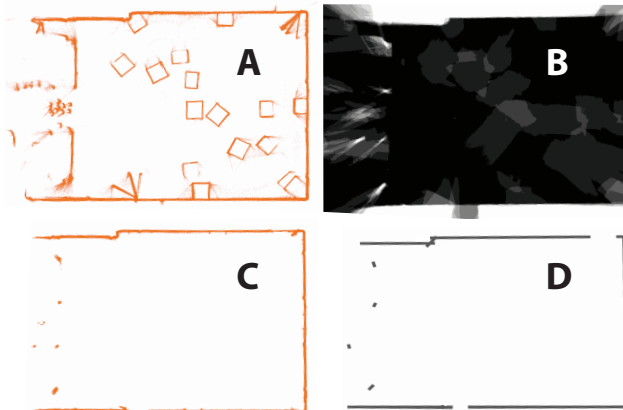
Fig. 1: Observations at different stages of the LTVM pipeline. In alphabetical order: raw data from all deployments, weights computed by the SDF, filtered data, final LTVM.

In this paper, we introduce an algorithm to build and update Long-Term Vector Maps indefinitely, utilizing observations from all deployments of all the robots in an environment. Our proposed algorithm filters out observations corresponding to DFs from a single deployment using a signed distance function (SDF) [9]. Merging the SDFs from multiple deployments then filters out the short-term features. Remaining observations correspond to LTFs, and are used to build a vector map via robust local linear regression. Uncertainty estimates of the resultant Long-Term Vector Map are calculated by a novel Monte Carlo uncertainty estimator. Our proposed approach thus consists of the following steps:

1) *Filter:* Use the most recent observations to compute an SDF and discard points based on weights and values given by the SDF (Section IV).
2) *Line Extraction:* Use greedy sequential local RANSAC [11] and non-linear least-squares fitting to extract line segments from the filtered observations (Section V).
3) *Estimate Feature Uncertainty:* Compute segment endpoint covariance estimates via Monte Carlo resampling of the observations (Section VI).
4) *Map Update:* Add, merge, and delete lines using a decoupled scatter matrix representation [3] (Section VII).

Our approach takes advantage of the robust filtering provided by the SDF while avoiding dependency on a discrete world representation and grid resolution by representing LTFs as line segments in $\mathbb{R}^2$. Evaluation of our approach is detailed in section VIII. Across all metrics examined in this study, we find vector maps constructed via SDF filtering comparable or favorable to occupancy grid based approaches.

II. RELATED WORK

The problem of long-term robotic mapping has been studied extensively, with most algorithms relying on one of two dominant representations: occupancy grids [10], [14] and geometric or polygonal maps [8], [20]. Recently, work towards metric map construction algorithms that are able to cope with dynamic and short-term features has accelerated.

Most approaches fall into one of four categories: dynamics on occupancy grids, latest state estimation, ensemble state estimation, and observation filters.

One common approach models map dynamics on an occupancy grid using techniques such as learning non-stationary object models [5] or modeling grid cells as Markov chains [16], [17]. Alternatively, motivated by the widely varying timescales at which certain aspects of an environment may change, some approaches seek to leverage these differences by maintaining information relating to multiple timescales within one or more occupancy grids [1], [19].

Other approaches estimate the latest state of the world, including dynamic and short-term features. Dynamic pose-graph SLAM [18] can be used in low-dynamic environments, and spectral analysis techniques [13] attempt to predict future environment states on arbitrary timescales.

Instead of estimating solely the latest state, some approaches estimate environment configurations based on an ensemble of recent states. Temporal methods such as recency weighted averaging [2] determine what past information is still relevant, and other techniques such as learning local grid map configurations [6] borrow more heavily from the dynamic occupancy grid approach.

Another approach filters out all observations corresponding to non-LTFs, resulting in a "blueprint" map. Previous algorithms have had some success filtering dynamic objects, specifically people [12], but have struggled to differentiate between STFs and LTFs. Furthermore, all of the methods mentioned above rely on an occupancy grid map representation, whereas our method produces a polygonal, vector map.

III. LONG-TERM VECTOR MAPPING

Long-term vector mapping runs iteratively over multiple robot deployments, operating on the union of all registered laser observations from the given deployment, aligned to the same frame. We call these unions composite scans and denote them $C = \cup_{i=1}^{N} S_i$, where S_i is a single laser scan. Composite scans are processed in batch after each deployment, and may be generated via Episodic non-Markov Localization [4] or a similar localization algorithm.

After each deployment, a short-term signed distance function (ST-SDF) given by a set of weights W' and values V' is computed over the area explored by the robot by considering the composite scan C. The ST-SDF is then used to update the long-term SDF (LT-SDF), given by W and V, which aggregates information over all previous deployments. The updated LT-SDF is denoted W^* and V^*, and is used to determine a filtered composite scan $C' \subset C$, containing observations corresponding exclusivley to LTFs. We call this process SDF-filtering. Note that after only one deployment,

it is not possible to distinguish STFs from LTFs. However, as the number of deployments increases, the number of observations corresponding to STFs in C' approaches zero.

The next step in our algorithm is feature (line) extraction. Line extraction does not rely on any persistent data, using only the filtered composite scan C' to extract a set of lines L'. Each $l'_i \in L'$ is defined by endpoints p_{i_1} and p_{i_2}, a scatter matrix S_i, a center of mass $p_{i_{cm}}$, and a mass M_i.

Uncertainties in the endpoints of each line segment are computed by analyzing a distribution of possible endpoints generated via Monte Carlo resampling the initial set of observations and subsequently refitting the samples. For a given line l_i the uncertainty estimation step takes as input the endpoints p_{i_1} and p_{i_2} and a set of inliers I_i, and produces covariance estimates Q_{i_1} and Q_{i_2} for these endpoints.

The long-term vector map is updated based on the newest set of extracted lines and the current SDF. Similar lines are merged into a single line, obsolete lines are deleted, and uncertainties are recomputed. Thus, this step takes the results from the most recent deployment, L', as well as the existing map given by L, W^*, and V^*, and outputs an updated map, L^*. Fig. 2 presents an overview of the algorithm as it operates on data from a single deployment.

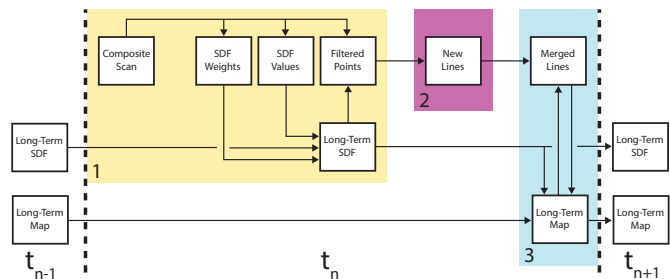

Fig. 2: Flow of information during processing of a single deployment, deployment n. Boxes 1, 2, and 3 correspond to SDF filtering, line finding and uncertainty estimation, and map updating, respectively.

IV. SDF-BASED FILTERING

Let C be a composite scan, where $c \in C$ is a single observation providing depth ρ and angle α with respect to the laser's frame and the robot pose $p = (x, y, \theta)$ at which ρ and α were recorded. That is, $c = [\rho, \alpha, p]$. The filtering problem is to determine $C' \subset C$ such that all $c' \in C'$ originate from LTFs and all $c \in C \setminus C'$ originate from STFs and DFs.

Determining C' is a three-step process. First, we construct a ST-SDF over the area observed by the robot during the deployment corresponding to C. Next, we update the LT-SDF based on the ST-SDF. Finally, we use the updated LT-SDF to decide which observations correspond to LTFs.

A. SDF Construction

SDFs operate over discretized space, so we create a grid of resolution q containing all $c \in C$. Each pixel in the SDF maintains two measurements, a value d_0 and a weight w_0. For every observation $c \in C$, all pixels that lie along the given laser ray update their respective values according to

$d_0 = \frac{w_0 d_0 + wd}{w_0 + w}$ and $w_0 = w_0 + w$, where d_0 and w_0 are the current distance and weight values, respectively, and d and w are the distance and weight values for the given reading c. d and w are given by

$$d(r) = \begin{cases} \delta & \text{if } r > \delta \\ r & \text{if } |r| \leq \delta \\ -\delta & \text{if } r < -\delta \end{cases}, \quad w(r) = \begin{cases} 1 & \text{if } |r| < \epsilon \\ e^G & \text{if } \epsilon \leq |r| \leq \delta \\ 0 & \text{if } |r| > \delta, \end{cases}$$
(1)

where $G = -\sigma(r - \epsilon)^2$ and r is the signed distance from the range reading to the pixel, with pixels beyond the range reading having $r < 0$ and those in front having $r > 0$. σ and ϵ are parameters that depend on the accuracy of the sensor. Pixels that are located along the ray but are more than δ beyond the detection point are not updated since we do not know whether or not they are occupied. Fig. 3 illustrates a single pixel update during SDF construction. Note that this process is parallelizable since weights and values for each pixel may be calculated independently. Thus, the SDF construction step, outlined in Algorithm 1, runs in time proportional to $|C|$.
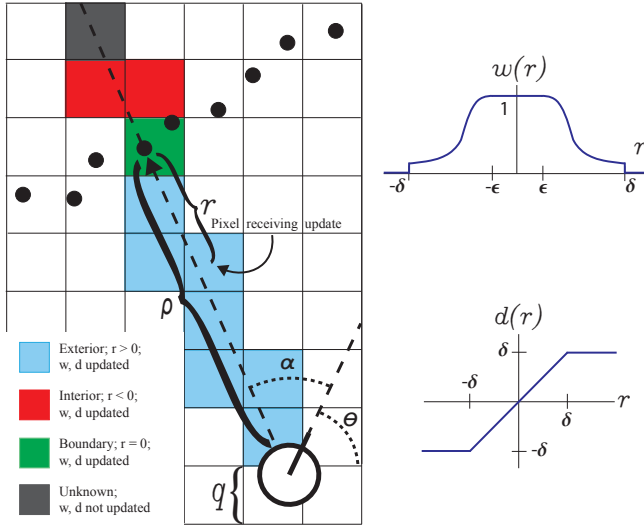

Fig. 3: SDF construction from a single laser ray. Pixels along the laser ray are updated if they are free, or if they are just beyond the obstacle. Over many ray casts, pixels may be marked as belonging to more than one category (boundary, interior, exterior) due to sensor noise. The SDF's main advantage is that it ignores erroneous readings.

The intuition for the weight and value functions comes from two observations. First, capping $|d(r)|$ by δ helps keep our SDF robust against anomalous readings. Second, $w(r)$ follows common laser noise models. Other choices for $d(r)$ and $w(r)$ may yield similar results; however, these choices have already been successfully adopted elsewhere [7].

B. SDF Update

Once the ST-SDF is calculated we normalize the weights:

$$w_{norm} = \begin{cases} 0 & \text{if } \frac{w}{w_{max}} \leq T_1, \\ 1 & \text{otherwise.} \end{cases}$$
(2)

Here, w_{max} is the maximum weight over all pixels and T_1 is a dynamic feature threshold. The LT-SDF is then updated as the weighted average of the ST-SDF and the LT-SDF, i.e. $W^* = \text{WEIGHTEDAVERAGE}(W, W')$. Pixel weights loosely map to our confidence about the associated value, and values are an estimate for how far a pixel is from the nearest surface.

C. SDF Filter

Given an up-to-date SDF, we determine C' using bicubic interpolation on the position of each observation c and the updated LT-SDF. The filtering criteria are $c' \in C'$ if BICUBICINTERPOLATION$(c, W^*) > T_2$ and BICUBICIN-TERPOLATION$(c, V^*) < T_d$, where T_2 is a STF threshold and T_d is a threshold that filters observations believed to be far from the surface of any object. Lines 11-15 in Algorithm 1 detail the filtering process. Thus, after running SDF FILTERING(C, W, V), we obtain a filtered composite scan C', used to find LTFs. Additionally, SDF FILTERING updates the LT-SDF needed for the map update step.

Algorithm 1 SDF FILTERING

1: **Input:** Raw composite scan C, long-term SDF weights W and values V
2: **Output:** Filtered composite scan C', updated SDF weights W^* and values V^*
3: $V' \leftarrow$ empty image
4: $W' \leftarrow$ empty image
5: **for all** range readings $c \in C$ **do**
6: $\quad V' \leftarrow$ VALUEUPDATE(W', c)
7: $\quad W' \leftarrow$ WEIGHTUPDATE(W', c)
8: $W' \leftarrow$ NORMALIZE(W')
9: $W^*, V^* \leftarrow$ UPDATESDF(V', W')
10: $C' \leftarrow \emptyset$
11: **for all** range readings $c \in C$ **do**
12: $\quad b_w \leftarrow$ BICUBICINTERPOLATION(W^*, c)
13: $\quad b_v \leftarrow$ BICUBICINTERPOLATION(V^*, c)
14: $\quad$ **if** $b_w > T_w$ **and** $b_v < T_v$ **then**
15: $\quad\quad C' \leftarrow C' \cup c$

V. LINE EXTRACTION

Given C', extracting lines $l_1 \dots l_n$ requires solving two problems. First, for each l_i, a set of observations $C_i \subset C'$ must belong to line l_i. Second, endpoints p_{i_1} and p_{i_2} defining line l_i must be found. We take a greedy approach, utilizing sequential local RANSAC to provide plausible initial estimates for line endpoints p_1 and p_2. Points whose distance to the line segment $\overline{p_1 p_2}$ is less than T_r, where T_r is proportional to the noise of the laser, are considered members of the inlier set I. Once a hypothetical model and set of inliers with center of mass p_{cm} have been suggested by RANSAC (lines 5-7 in Algorithm 2), we perform a non-linear least-squares optimization using the cost function

$$R = \frac{||p_{cm} - p_1|| + ||p_{cm} - p_2||}{|I|}$$

$$+ \begin{cases} ||p - p_2|| & \text{if } t < 0 \\ ||p - p_1|| & \text{if } t > 1 \\ ||p_1' + t(p_2' - p_1') - p|| & \text{otherwise} \end{cases}$$
(3)

The new endpoints p_1' and p_2' are then used to find a new set of inliers I'. $t = \frac{(p-p_1')\cdot(p_2'-p_1')}{||p_2'-p_1'||^2}$ is the projection of a point $p \in I$ onto the infinite line containing p_1' and p_2'. Iteration terminates when $||p_1 - p_1'|| + ||p_2 - p_2'|| < T_c$, where T_c is a convergence threshold.

This cost function has several desireable properties. First, when all points lie between the two endpoints, the orientation of the line will be identical to the least-squares solution. Second, when many points lie beyond the ends of the line segment, the endpoints are pulled outward, allowing the line to grow and the region in which we accept inliers to expand. Last, the $\frac{||p_{cm}-p_1||+||p_{cm}-p_2||}{|I|}$ term allows the line to shrink in the event that the non-linear least-squares solver overshoots the appropriate endpoint. Once a set of lines L' has been determined by running LINE EXTRACTION(C') we complete our analysis of a single deployment by estimating our uncertainty in feature locations.

Algorithm 2 LINE EXTRACTION

1: **Data:** Filtered composite scan C'
2: **Result:** Set of new lines L'
3: $L' \leftarrow \emptyset$
4: **while** C' **not** empty **do**
5:     Propose p_1, p_2 via RANSAC
6:     $I \leftarrow$ FINDINLIERS(p_1, p_2)
7:     $p_1', p_2' \leftarrow$ FITSEGMENT(I)
8:     **while** $||p_1' - p_1|| + ||p_2' - p_2|| > T_C$ **do**
9:         $I \leftarrow$ FINDINLIERS(p_1', p_2')
10:         $p_1, p_2 \leftarrow p_1', p_2'$
11:         $p_1', p_2' \leftarrow$ FITSEGMENT(I)
12:     $L' \leftarrow L' \cup \overline{p_1'p_2'}$
13:     $C' \leftarrow C' \setminus I$

VI. UNCERTAINTY ESTIMATION

Given a line l_i, with endpoints p_{i_1} and p_{i_2} and a set of inliers I_i, uncertainty estimation produces covariance estimates Q_{i_1} and Q_{i_2} for p_{i_1} and p_{i_2}, respectively. To estimate Q_{i_1} and Q_{i_2} we resample $c_i \sim I_i$ using the covariance Q_i^c of each range reading. Q_i^c is derived based on the sensor noise model in [15]. In world coordinates Q_i^c is given by

$$
\begin{aligned}
Q_i^c = &\frac{\rho^2\sigma_\alpha^2}{2} \left[\begin{array}{cc} 2\sin^2(\alpha+\theta) & -\sin(2(\alpha+\theta)) \\ -\sin(2(\alpha+\theta)) & 2\cos^2(\alpha+\theta) \end{array} \right] \\
&+ \frac{\sigma_\rho^2}{2} \left[\begin{array}{cc} 2\cos^2(\alpha+\theta) & \sin(2(\alpha+\theta)) \\ \sin(2(\alpha+\theta)) & 2\sin^2(\alpha+\theta) \end{array} \right],
\end{aligned}
\tag{4}
$$

where σ_ρ and σ_α are standard deviations for range and angle measurements for the sensor, respectively. Resampling k times, as shown in Fig. 4, produces a distribution of p_1 and p_2. We then construct a scatter matrix S from the set of hypothetical endpoints, and compute covariances Q_1 and Q_2 by using the SVD of S.

The Monte Carlo approach, detailed in Algorithm 3, is motivated by the following factors: 1) There is no closed-form solution to covariance for endpoints of a line segment. 2) A piece-wise cost function makes it difficult to calculate

the Jacobian reliably. 3) Resampling is easy since we already have I_i and can calculate Q_i^c.

Algorithm 3 FEATURE UNCERTAINTY ESTIMATION

1: **Input:** Set of new lines L', number of samples k
2: **Output:** Set of endpoint covariance estiamates Q_1', Q_2'
3: $Q_1' \leftarrow \emptyset, Q_2' \leftarrow \emptyset$
4: **for all** $l_i' \in L'$ **do**
5:     $P_1' \leftarrow \emptyset, P_2' \leftarrow \emptyset$
6:     $I_i \leftarrow$ inliers associated with l_i'
7:     **for** k iterations **do**
8:         $I_i' \leftarrow \emptyset$
9:         **for all** $c \in I_i$ **do**
10:             $c' \leftarrow$ SAMPLE(I_i, c, Q_i^c)
11:             $I_i' \leftarrow I_i' \cup c'$
12:         $p_1', p_2' \leftarrow$ FITSEGMENT(I_i')
13:         $P_1' \leftarrow P_1' \cup p_1', P_2' \leftarrow P_2' \cup p_2'$
14:     $Q_1' \leftarrow Q_1' \cup$ ESTIMATECOVARIANCE(P_1')
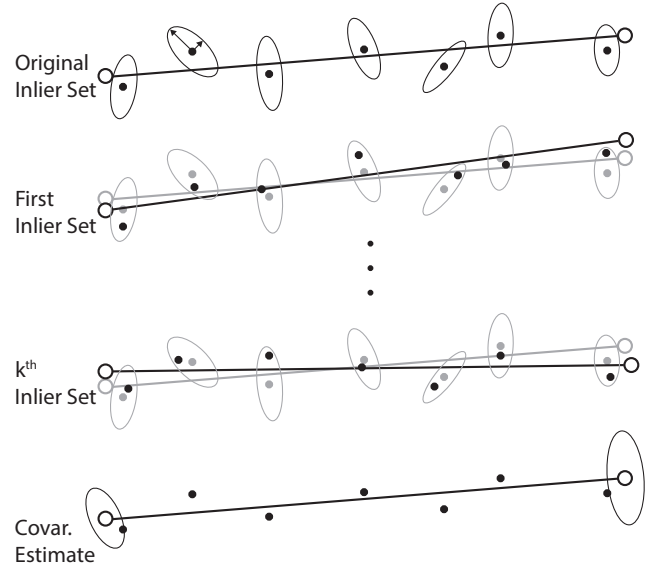15: $Q_2' \leftarrow Q_2' \cup$ ESTIMATECOVARIANCE(P_2')

Fig. 4: Monte Carlo uncertainty estimation of feature endpoints. Given an initial set of observations and their corresponding covariances represented by ellipses, we resample the observations and fit a line k times. The resulting distribution of endpoints is used to estimate endpoint covariance.

VII. MAP UPDATE

Given the current map L, the LT-SDF, and a set of new lines, L', where every $l_i \in L'$ is specified by a set of endpoints p_{i_1} and p_{i_2}, a set of covariance matrices Q_{i_1} and Q_{i_2}, a center of mass $p_{i_{cm}}$, a mass M_i, and a partial scatter matrix S_i, the update step produces an updated map L^*.

The map updates are divided into two steps outlined in Algorithm 4. First, we check if all current lines in the map are contained within high-weight regions. That is, we check that the weight w_{xy} of every pixel a given line passes through satisfies $w_{xy} \geq T_2$. If a line lies entirely within a high-weight region, it remains unchanged. Similarly, if a line lies

entirely outside all high-weight regions, it is removed. If only part of a line remains within a high-weight region, we can lower bound the mass of the remaining region by $M' = M\frac{||p'_1-p'_2||}{||p_1-p_2||}$, where p'_1 and p'_2 are the extreme points of the segment remaining in the high-weight region (line 11). We then resample M' points uniformly along the line between p'_1 and p'_2, adding Gaussian noise in the perpendicular direction with a standard deviation σ based on the sensor noise model. The points are then fit using the cost function given in (4). The sampling and fitting process is executed a fixed number of times (lines 12-16), and the distribution of fit results is then used to compute covariance estimates for the new line.

The second part of the update involves merging new lines, L', with lines from the current map, L. This process consists of first finding candidates for merging (lines 22-23) and then computing the updated parameters and covariances (lines 24-25). Note that the mapping from new lines to existing lines may be onto, but without loss of generality we consider merging lines pairwise. Because our parameterization uses endpoints, lines which ought to be merged may not have endpoints near one another. So, we project p'_{i_1} and p'_{i_2} from l'_i onto l_j, resulting in $p^{proj}_{j_1}$ and $p^{proj}_{j_2}$, respectively. l'_i and l_j are merged if they pass the chi-squared test:

$$\chi^2 = \Delta l_k^T (Q^{int}_{j_k} + Q'_{i_k})\Delta l_k < T_\chi, \quad k = 1, 2 \quad (5)$$

where $\Delta l_k = p'_{i_k} - p^{proj}_{j_k}$, and $Q^{int}_{j_k}$ is given by a linear interpolation of the covariance estimates for p_{j_1} and p_{j_2} determined by where p'_{i_k} is projected along l_j.

We would like our merged LTFs and their uncertainty measures to remain faithful to the entire observation history. However, storing every observation is infeasible. Instead, our method implicitly stores observation histories using decoupled scatter matrices [3], reducing the process of merging lines with potentially millions of supporting observations to a single matrix addition.

The orientation of the new line is found via eigenvalue decomposition of the associated scatter matrix, and new endpoints are found by projecting the endpoints from the original lines onto the new line and taking the extrema. Thus, after executing MAP UPDATE(L', L, W^*, V^*), we have a set of vectors L^* in $\mathbb{R}^2$ corresponding to LTFs.

Table I displays the major parameters and physical constants needed for long-term vector mapping.

TABLE I: Thresholds and physical constants

Name	Symbol	Domain	Our Value
DF Threshold	T_1	(0, 1)	0.2
STF Threshold	T_2	(0, 1)	0.95
Line Merge Criteria	T_{χ^2}	> 0	30
Sensor Noise Threshold	T_d	> 0	0.05 meters
RANSAC Inlier Criteria	T_r	> 0	0.12 meters
Line Fit Convergence	T_c	> 0	0.05 meters
SDF Max Value	δ	> 0	0.2 meters

VIII. RESULTS

To demonstrate the effectiveness of our algorithm we mapped 4 different environments, including standard data. Data was also collected from three separate environments, in addition to the MIT Reading Room: AMRL, Wall-occlusion,

---

Algorithm 4 MAP UPDATE

1: **Input:** Set of new lines L', set of long-term lines L, long-term SDF W, V
2: **Output:** Updated long-term lines L^*
3: **for all** $l_i \in L$ **do**
4:      trace along l_i
5:      **if** l_i exists entirely outside high-weight regions **then**
6:          $L \leftarrow L \setminus l_i$
7:      **if** l_i exists partially outside high-weight regions **then**
8:          $L \leftarrow L \setminus l_i$
9:          $P'_1, P'_2 \leftarrow \emptyset$
10:          **for all** remaining parts of l_i, ∂l_i **do**
11:              $p_1, p_2 \leftarrow$ GETENPOINTS(∂l_i)
12:              **for** k iterations **do**
13:                  $I \leftarrow$ REGENERATEINLIERS(p_1, p_2)
14:                  $p'_1, p'_2 \leftarrow$ FITSEGMENT(I)
15:                  $P'_1 \leftarrow P'_1 \cup p'_1$
16:                  $P'_2 \leftarrow P'_2 \cup p'_2$
17:              ESTIMATECOVARIANCE(P'_1, P'_2)
18:              $L \leftarrow L \cup \partial l_i$
19: $L^* \leftarrow \emptyset$
20: **for all** $l'_i \in L'$ **do**
21:      **for all** $l_j \in L$ **do**
22:          $\chi^2 \leftarrow \Delta l_k^T (Q^{int}_{j_k} + Q'_{i_k})\Delta l_k < T_\chi, \quad k = 1, 2$
23:          **if** $\chi^2 < T_{\chi^2}$ for $k = 1, 2$ **then**
24:              $l_j \leftarrow$ MERGE(l_j, l'_i)
25:              $L^* \leftarrow L^* \cup l_j$
26:          **else**
27:              $L^* \leftarrow L^* \cup l'_i$

---

and Hallway-occlusion, using a mobile robot platform and a Hokuyo UST-10LX laser range finder. The AMRL, Wall, and Hall datasets consist of 8, 5, and 5 deployments, respectively. MIT Reading Room contains 20 deployments. Each deployment contains hundreds of scans, corresponding to hundreds of thousands of observations. Datasets are intended to display a high amount of clutter and variability, typical of scenes mobile robots need to contend with.

The MIT and AMRL data sets are intended to test the accuracy of our algorithm over larger numbers of deployments. Both rooms contain multiple doors, walls of varying lengths, and achieve many different configurations, testing our ability to accurately identify LTF positions. The Wall- and Hallway-occlusion datasets are intended to measure our algorithm's robustness in environments where LTFs are heavily occluded.

Quantitatively we are concerned with accuracy and robustness, defining accuracy as pairwise feature agreement, where inter-feature distances are preserved, and feature-environment correspondance, where vectors in the map correspond to LTFs in the environment. Vectors should also be absent from areas where there are no long-term features such as doorways. Metric ground truth is established by either measuring wall lengths or hand-fitting the parts of the data we know correspond to LTFs. Robustness refers to a map's ability to deal with large numbers of STFs and lack of degradation over time.
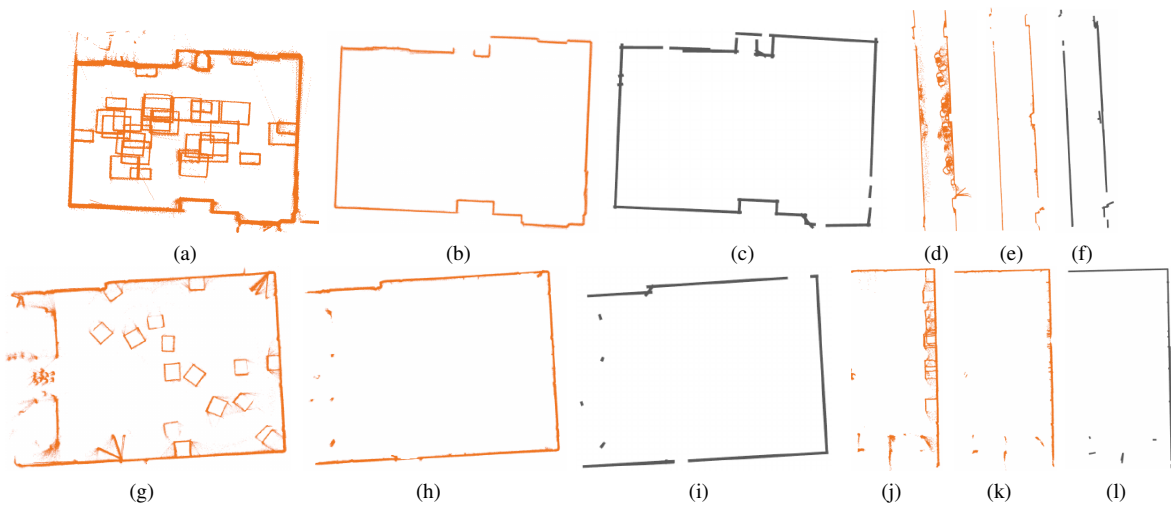
Fig. 5: Raw data, filtered data, and resultant maps for MIT (a-c), AMRL (d-f), Wall-occlusion (g-i) and Hallway-occlusion (j-l) datasets. Data shown in the left and center columns are aggregates of all deployments and are never actually stored while the algorithm is operating. The last column is the resultant LTVM which is stored in full, requiring only a few KB. Note the absence of STFs from the final maps, as well as the presence of doorways. In the MIT dataset, some doors were open only once over the entire deployment history. The Hallway-occlusion dataset demonstrates the algorithm's robustness to STFs, as it is able to distinguish the column in the hallway even as it is partially or completely occluded on every deployment.

Over all datasets, our approach correctly identifies all 7 doorways (4 in MIT, 2 in AMRL, 1 in Hallway), and correctly ignores all 73 STFs. Using the AMRL and MIT datasets, we find the average difference between pair-wise feature separation in the generated map versus ground truth to be on the order of 2cm. Our line extraction method yields MSE values in the order of 0.0001 meters, while marching squares yields a MSE of roughly 0.0003, about three times as large. Additionally, marching squares yields over 3000 features while LTVM maintains on the order of 30 LTFs. Furthermore, the maps do not degrade over the timescales tested in this paper, with no noticeable difference in average pair-wise feature disagreement between maps generated after one deployment and those considering all deployments. Fig. 5 displays qualitative results for all environments.

IX. CONCLUSION

In this paper, we introduced an SDF-based approach to filter laser scans over multiple deployments in order to build and maintain a long-term vector map. We experimentally showed the accuracy and robustness of the generated maps in a variety of different environments and further evaluated the effectiveness of long-term vector mapping compared to more standard, occupancy grid techniques. Currently this approach processes observations in batch post deployment, but modifying the algorithm to run in an online setting seems an appropriate next step.

X. ACKNOWLEDGEMENTS

REFERENCES

[1] D. Arbuckle, A. Howard, and M. Mataric. Temporal occupancy grids: a method for classifying the spatio-temporal properties of the environment. In *IROS, 2002*.

[2] P. Biber. Dynamic maps for long-term operation of mobile service robots. In *RSS*, 2005.

[3] J. Biswas and M. Veloso. Planar polygon extraction and merging from depth images. In *IROS*, 2012.

[4] J. Biswas and M. Veloso. Episodic non-markov localization: Reasoning about short-term and long-term features. In *ICRA*, 2014.

[5] R. Biswas, B. Limketkai, S. Sanner, and S. Thrun. Towards object mapping in non-stationary environments with mobile robots. In *IROS*, 2002.

[6] W. Burgard, C. Stachniss, and D. Hähnel. *Autonomous Navigation in Dynamic Environments*, pages 3–28. 2007.

[7] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *RSS*, 2013.

[8] R. Chatila and J. Laumond. Position referencing and consistent world modeling for mobile robots. In *Robotics and Automation.*, 1985.

[9] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *ACM*, 1996.

[10] A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 1989.

[11] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *ACM*, 1981.

[12] D. Hahnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *IROS*, 2002.

[13] T. Krajnik, J. P. Fentanes, G. Cielniak, C. Dondrup, and T. Duckett. Spectral analysis for long-term robotic mapping. In *ICRA*, 2014.

[14] H. P. Moravec and D. W. Cho. A Bayesean method for certainty grids. In *AAAI Spring Symposium on Robot Navigation*, 1989.

[15] S. T. Pfister, S. I. Roumeliotis, and J. W. Burdick. Weighted line fitting algorithms for mobile robot map building and efficient data representation. In *ICRA*, 2003.

[16] J. Saarinen, H. Andreasson, and A. J. Lilienthal. Independent markov chain occupancy grid maps for representation of dynamic environment. In *IROS, 2012*.

[17] G. D. Tipaldi, D. Meyer-Delius, and W. Burgard. Lifelong localization in changing environments. *IJRR*, 2013.

[18] A. Walcott-Bryant, M. Kaess, H. Johannsson, and J. J. Leonard. Dynamic pose graph slam: Long-term mapping in low dynamic environments. In *IROS*, 2012.

[19] D. F. Wolf and G. S. Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 2005.

[20] L. Zhang and B. K. Ghosh. Line segment based map building and localization using 2d laser rangefinder. In *ICRA*, 2000.