Appendix

A. Metric Upgrade for Learned Map

In the main body of the paper, we assumed that global position estimates of at least four landmarks were known. When these landmarks are known, we can recover all of the estimated landmark positions and robot locations.

In many cases, however, no global positions are known; the best we can hope to do is recover landmark and robot positions up to an orthogonal transform (translation, rotation, and reflection). It turns out that Eqs. (7) provide enough geometric constraints to perform this metric upgrade, as long as we have at least 8 landmarks and at least 8 time steps. The idea is to fit a quadratic surface to the rows of U or the columns of V, then change coordinates so that the surface becomes the functions given in (7).

To derive the metric upgrade, suppose that we start from an $N \times 4$ matrix U of learned landmark coordinates and an $4 \times N$ matrix V of learned robot coordinates from the algorithm of Sec. 3.1. We would like to transform the learned coordinates into two new matrices C and X such that

$$c_1 \approx 1 \tag{14}$$

$$c_4 \approx \frac{1}{2}c_2^2 + \frac{1}{2}c_3^2 \tag{15}$$

$$x_4 \approx 1 \tag{16}$$

$$x_1 \approx \frac{1}{2}x_2^2 + \frac{1}{2}x_3^2 \tag{17}$$

where c is a row of C and x is a column of X.

At a high level, we first fit a quadratic surface to the rows of U, then transform this surface so that it satisfies Eq. 14–15, and scale the surface so that it satisfies Eq. 16. Our surface will then automatically also satisfy Eq. 17, since X must be metrically correct if C is.

In more detail, we first (step i) linearly transform each row of U into approximately the form $(1, r_{i,1}, r_{i,2}, r_{i,3})$: we use linear regression to find a coefficient vector $a \in \mathbb{R}^4$ such that $Ua \approx \mathbf{1}$, then set $R = UQ$ where $Q \in \mathbb{R}^{4\times 3}$ is an orthonormal basis for the nullspace of a^T. After this step, our factorization is $(UT_1)(T_1^{-1}V)$, where $T_1 = (a\ Q)$.

Next (step ii) we fit an implicit quadratic surface to the rows of R by finding 9 coefficients b_{jk} (for $0 \le j \le k \le 3$) such that

$$\begin{aligned} 0 \ \approx\ & b_{00} + b_{01}r_{i,1} + b_{02}r_{i,2} + b_{03}r_{i,3} + b_{11}r_{i,1}^2 + \\ & b_{12}r_{i,1}r_{i,2} + b_{13}r_{i,1}r_{i,3} + b_{22}r_{i,2}^2 + b_{23}r_{i,2}r_{i,3} + b_{33}r_{i,3}^2 \end{aligned}$$

To do so, we form a matrix S that has the same number of rows as U but 9 columns. The elements of row i of S are $r_{i,j}r_{i,k}$ for $0 \le j \le k \le 3$ (in any fixed order; for conciseness, we take $r_{i,0} = 1$ for all i). Then we find a vector $b \in \mathbb{R}^9$ that is approximately in the nullspace of S^T by taking a singular value decomposition of S and selecting the right singular vector corresponding to the smallest singular value. Using this vector, we can define our quadratic as $0 \approx \frac{1}{2}r^\mathsf{T}Hr + \ell^\mathsf{T}r + b_{00}$, where r is a row of R, and the Hessian matrix H and linear part ℓ are given by:

$$H = \begin{pmatrix} \frac{1}{2}b_{11} & b_{12} & b_{13} \\ b_{21} & \frac{1}{2}b_{22} & b_{23} \\ b_{31} & b_{32} & \frac{1}{2}b_{33} \end{pmatrix} \qquad \ell = \begin{pmatrix} b_{01} \\ b_{02} \\ b_{03} \end{pmatrix}$$

Over the next few steps we will transform the coordinates in R to bring our quadratic into the form of Eq. 15: that is, one coordinate will be a quadratic function of the other two, there will be no linear or constant terms, and the quadratic part will be spherical with coefficient $\frac{1}{2}$.

We start (step iii) by transforming coordinates so that our quadratic has no cross-terms, i.e., so that its Hessian matrix is diagonal. Using a 3×3 singular value decomposition, we can factor $H = MH'M^\mathsf{T}$ so that M is orthonormal and H' is diagonal. If we set $R' = RM$ and $\ell' = M\ell$, and write r' for a row of R', we can equivalently write our quadratic as $0 = \frac{1}{2}(r')^\mathsf{T}H'r' + (\ell')^\mathsf{T}r' + b_{00}$, which has a diagonal Hessian as desired. After this step, our factorization is $(UT_1T_2)(T_2^{-1}T_1^{-1}V)$, where

$$T_2 = \begin{pmatrix} 1 & 0 \\ 0 & M \end{pmatrix}$$

Our next step (step iv) is to turn our implicit quadratic surface into an explicit quadratic function. For this purpose we pick one of the coordinates of R' and write it as a function of the other two. In order to do so, we must have zero as the corresponding diagonal element of the Hessian H'—else we cannot guarantee that we can solve for a unique value of the chosen coordinate. So, we will take the index j such that H'_{jj} is minimal, and set $H'_{jj} = 0$. Suppose that we pick the last coordinate, $j = 3$. (We can always reorder columns to make this true; SVD software will typically do so automatically.) Then our quadratic becomes

$$0 = \frac{1}{2}H'_{11}(r'_1)^2 + \frac{1}{2}H'_{22}(r'_2)^2 + \ell'_1r'_1 + \ell'_2r'_2 + \ell'_3r'_3 + b_{00}$$

$$r'_3 = -\frac{1}{\ell'_3}\left[\frac{1}{2}H'_{11}(r'_1)^2 + \frac{1}{2}H'_{22}(r'_2)^2 + \ell'_1r'_1 + \ell'_2r'_2 + b_{00}\right]$$

Now (step v) we can shift and rescale our coordinates one more time to get our quadratic in the desired form:

translate so that the linear and constant coefficients are 0, and rescale so that the quadratic coefficients are $\frac{1}{2}$. For the translation, we define new coordinates $r'' = r' + c$ for $c \in \mathbb{R}^3$, so that our quadratic becomes

$$r''_3 = c_3 - \frac{1}{\ell'_3}\left[\frac{1}{2}H'_{11}(r''_1 - c_1)^2 + \frac{1}{2}H'_{22}(r''_2 - c_2)^2 + \ell'_1(r''_1 - c_1) + \ell'_2(r''_2 - c_2) + b_{00}\right]$$

By expanding and matching coefficients, we know c must satisfy

$$0 = \frac{H'_{11}}{\ell'_3}c_1 - \frac{\ell'_1}{\ell'_3} \qquad \text{(coefficient of } r''_1\text{)}$$

$$0 = \frac{H'_{22}}{\ell'_3}c_2 - \frac{\ell'_2}{\ell'_3} \qquad \text{(coefficient of } r''_2\text{)}$$

$$0 = c_3 - \frac{H'_{11}}{2\ell'_3}c_1^2 - \frac{H'_{22}}{2\ell'_3}c_2^2 + \frac{\ell'_1}{\ell'_3}c_1 + \frac{\ell'_2}{\ell'_3}c_2 - b_{00}/\ell'_3 \qquad \text{(constant)}$$

The first two equations are linear in c_1 and c_2 (and don't contain c_3). So, we can solve directly for c_1 and c_2; then we can plug their values into the last equation to find c_3. For the scaling, the coefficient of r''_1 is now $-\frac{H'_{11}}{2\ell'_3}$, and that of r''_2 is now $-\frac{H'_{22}}{2\ell'_3}$. So, we can just scale these two coordinates separately to bring their coefficients to $\frac{1}{2}$.

After this step, our factorization is $U'V'$, where $U' = UT_1T_2T_3$ and $V' = T_3^{-1}T_2^{-1}T_1^{-1}V$, and

$$T_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ c_1 & -\frac{\ell'_3}{H'_{11}} & 0 & 0 \\ c_2 & 0 & -\frac{\ell'_3}{H'_{22}} & 0 \\ c_3 & 0 & 0 & 1 \end{pmatrix}$$

The left factor U' will now satisfy Eq. 14–15. We still have one last useful degree of freedom: if we set $C = U'T_4$, where

$$T_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 \\ 0 & 0 & \mu & 0 \\ 0 & 0 & 0 & \mu^2 \end{pmatrix}$$

for any $\mu \in \mathbb{R}$, then C will still satisfy Eq. 14–15. So (step vi), we will pick μ to satisfy Eq. 16: in particular, we set $\mu = \sqrt{\text{mean}(V'_{4,:})}$, so that when we set $X = T_4^{-1}V'$, the last row of X will have mean 1.

If we have 7 learned coordinates in U as in Sec. 3.2, we need to find a subspace of 4 coordinates in order to perform metric upgrade. To do so, we take advantage of the special form of the correct answer, given in Eq. 10: in the upper block of C in Eq. 10, three coordinates are identically zero. Since U is a linear transformation of C, there will be three linear functions of the top block of U that are identically zero (or approximately zero in the presence of noise). We can use SVD on the top block of U to find and remove these linear functions (by setting the smallest three singular values to zero), then proceed as above with the four remaining coordinates.

B. Sample Complexity for the Measurement Model (Robot Map)

Here we provide the details on how our estimation error scales with the number T of training examples—that is, the scaling of the difference between the estimated measurement model $\widehat{U}$, which contains the location of the landmarks, and its population counterpart.

Our bound has two parts. First we use standard concentration bounds to show that each element of our estimated covariance $\widehat{M} = \widehat{Y}\widehat{Y}^\top$ approaches its population value. We use the Azuma-Hoeffding inequality to bound the probability that the sum of random variables deviates from its mean. We start by rewriting the empirical covariance matrix as a vector summed over multiple samples:

$$\text{vec}\left(\widehat{M}\right) = \frac{1}{T}\sum_{t=1}^{T}\Upsilon_{:,t}$$

where $\Upsilon = (\widehat{Y} \odot \widehat{Y})^\top$ is the matrix of column-wise Kronecker products of the observations $\widehat{Y}$. We assume that each element of Υ minus its expectation $\mathbb{E}\Upsilon_i$ is bounded by a constant c; we can derive c from bounds on anticipated errors in distance measurements and odometry measurements.

$$|\Upsilon_{i,t} - \mathbb{E}\Upsilon_i| \le c, \quad \forall_{i,t}$$

Then the Azuma-Hoeffding inequality bounds the probability that the empirical sum differs too much from its population value:

$$\mathbb{P}\left[\left|\sum_{t=1}^{T}(\Upsilon_{i,t} - \mathbb{E}\Upsilon_i)\right| \ge \alpha\right] \le 2e^{-\alpha^2/2Tc^2}$$

If we pick $\alpha = \sqrt{2Tc^2\log(T)}$, then we can rewrite the probability in terms of T:

$$\mathbb{P}\left[\frac{1}{T}\left|\sum_{t=1}^{T}(\Upsilon_{i,T} - \mathbb{E}\Upsilon_i)\right| \ge c\sqrt{\frac{2\log(T)}{T}}\right] \le 2e^{-\log(T)}$$

which means that the probability decreases as $O(\frac{1}{T})$ and the threshold decreases as $\tilde{O}(\frac{1}{\sqrt{T}})$.

We can then use a union bound over all $(2N)^2$ covariance elements j (since $\widehat{Y} \in \mathbb{R}^{2N \times T}$):

$$\forall_j \, \mathbb{P}\left[\left|\frac{1}{T}\sum_{t=1}^{T} \Upsilon_{i,T} - \mathbb{E}\Upsilon_i\right| \geq c\sqrt{\frac{2\log(T)}{T}}\right] \leq 8N^2/T$$

That is, with high probability, the entire empirical covariance matrix $\widehat{M}$ is going to be close (in max-norm) to its expectation.

Next we use the continuity of the SVD to show that the learned subspace approaches its true value. Let $\widehat{M} = M + E$, where E is the perturbation (so the largest element of E is bounded). Let $\widehat{U}$ be the output of SVD, and let U be the population value (the top singular vectors of the true M). Let Ψ be the matrix of canonical angles between range(U) and range($\widehat{U}$). Since we know the exact rank of the true M (either 4 or 7), the last (4th or 7th) singular value of M will be positive; call it $\gamma > 0$. So, by Theorem 4.4 of Stewart and Sun (1998),

$$||\sin\Psi||_2 \leq \frac{||E||_2}{\gamma}$$

This result uses a 2-norm bound on E, but the bound we showed above is in terms of the largest element of E. But, the 2-norm can be bounded in terms of the largest element:

$$||E||_2 \leq N \max_{ij} |E_{ij}|$$

Finally, the result is that we can bound the canonical angle:

$$||\sin\Psi||_2 \leq \frac{Nc\sqrt{\frac{2\log(T)}{T}}}{\gamma}$$

In other words, the canonical angle shrinks at a rate of $\tilde{O}(\frac{1}{\sqrt{T}})$, with probability at least $1 - \frac{8N^2}{T}$.

C. The Robot as a Nonlinear Dynamical System

Once we have learned an interpretable state space via the algorithm of Section 3.3, we can simply write down the nominal robot dynamics in this space. The accuracy of the resulting model will depend on how well our sensors and actuators follow the nominal dynamics, as well as how well we have learned the transformation S to the interpretable version of the state space.

In more detail, we model the robot as a controlled nonlinear dynamical system. The evolution is governed by the following state space equations, which generalize (1):

$$s_{t+1} = f(s_t, a_t) + \epsilon_t \tag{18}$$
$$o_t = h(s_t) + \nu_t \tag{19}$$

Here $s_t \in \mathbb{R}^k$ denotes the hidden state, $a_t \in \mathbb{R}^l$ denotes the control signal, $o_t \in \mathbb{R}^m$ denotes the observation, $\epsilon_t \in \mathbb{R}^k$ denotes the state noise, and $\nu_t \in \mathbb{R}^m$ denotes the observation noise. For our range-only system, following the decomposition of Section 3, we have:

$$s_t = \begin{bmatrix} 1 \\ -x_t \\ -y_t \\ (x_t^2 + y_t^2)/2 \\ -\cos(\theta_t) \\ -\sin(\theta_t) \\ \frac{x_{t+1}^2 - x_t^2 + y_{t+1}^2 - y_t^2}{2v_t} \end{bmatrix}$$

$$o_t = \begin{bmatrix} d_{1t}^2/2 \\ \vdots \\ d_{Nt}^2/2 \\ \frac{d_{1t+1}^2 - d_{1t}^2}{2v_t} \\ \vdots \\ \frac{d_{Nt+1}^2 - d_{Nt}^2}{2v_t} \end{bmatrix}$$

$$a_t = \begin{bmatrix} v_t \\ \cos(\omega_t) \\ \sin(\omega_t) \end{bmatrix} \tag{20}$$

Here v_t and ω_t are the translation and rotation calculated from the robot's odometry. A nice property of this model is that expected observations are a *linear* function of state:

$$h(s_t) = Cs_t \tag{21}$$

The dynamics, however, are *nonlinear*: see Eq. 22, which can easily be derived from the basic kinematic motion model for a wheeled robot (Thrun et al., 2005).

$$f(s_t, a_t) =$$

$$\begin{bmatrix} 1 \\ -x_t - v_t\cos(\theta_t) \\ -y_t - v_t\sin(\theta_t) \\ \frac{x_t^2+y_t^2}{2} + v_t x_t\cos(\theta_t) + v_t y_t\sin(\theta_t) + \frac{v_t^2\cos^2(\theta_t) + v_t^2\sin^2(\theta_t)}{2} \\ -\cos(\theta_t)\cos(\omega_t) + \sin(\theta_t)\sin(\omega_t) \\ -\sin(\theta_t)\cos(\omega_t) + \cos(\theta_t)\sin(\omega_t) \\ [x_t\cos(\theta_t)\cos(\omega_t) - x_t\sin(\theta_t)\sin(\omega_t) + v_t\cos^2(\theta_t)\cos(\omega_t) + \\ y_t\sin(\theta_t)\cos(\omega_t) - y_t\sin(\omega_t)\cos(\theta_t) + v_t\sin^2(\theta_t)\cos(\omega_t) - \\ 2v_t\cos(\theta_t)\sin(\theta_t)\sin(\omega_t)] \end{bmatrix} \tag{22}$$

C.0.1. Robot System Identification

To apply the model of Section C, it is essential that we maintain states in the physical coordinate frame, and not just the linearly transformed coordinate frame—i.e., $\widehat{C}$ and not $\widehat{U} = \widehat{C}S^{-1}$. However, it is possible instead to use *system identification* to learn to filter directly in the raw state space $\widehat{U}$. We conjecture that it may be more robust to do so, since we will not be sensitive to errors in the metric upgrade process (errors in learning S), and since we can learn to compensate for some deviations from the nominal model of Section C.

To derive our system identification algorithm, we can explicitly rewrite $f(s_t, a_t)$ as a nonlinear feature-expansion map followed by a linear projection. Our algorithm will then just be to use linear regression to learn the linear part of f.

First, let's look at the dynamics for the special case of $S = I$. Each additive term in Eq. 22 is the product of at most two terms in s_t and at most two terms in a_t. Therefore, we define $\phi(s_t, a_t) := s_t \otimes s_t \otimes \bar{a}_t \otimes \bar{a}_t$, where $\bar{a}_t = [1, a_t]^\mathsf{T}$ and $\otimes$ is the Kronecker product. (Many of the dimensions of $\phi(s_t, a_t)$ are duplicates; for efficiency we would delete these duplicates, but for simplicity of notation we keep them.) Each additive term in Eq. 22 is a multiple of an element of $\phi(s_t, a_t)$, so we can write the dynamics as:

$$s_{t+1} = N\phi(s_t, a_t) + \epsilon_t \tag{23}$$

where N is a linear function that picks out the correct entries to form Eq. 22.

Now, given an invertible matrix S, we can rewrite $f(s_t, a_t)$ as an *equivalent* function in the transformed state space:

$$Ss_{t+1} = \bar{f}(Ss_t, a_t) + S\epsilon_t \tag{24}$$

To do so, we use the identity $(Ax) \otimes (By) = (A \otimes B)(x \otimes y)$. Repeated application yields

$$\begin{aligned}\phi(Ss_t, a_t) &= Ss_t \otimes Ss_t \otimes \bar{a}_t \otimes \bar{a}_t \\ &= (S \otimes S \otimes I \otimes I)(s_t \otimes s_t \otimes \bar{a}_t \otimes \bar{a}_t) \\ &= \bar{S}\,\phi(s_t, a_t)\end{aligned} \tag{25}$$

where $\bar{S} = S \otimes S \otimes I \otimes I$. Note that $\bar{S}$ is invertible (since $\mathrm{rank}(A \otimes B) = \mathrm{rank}(A)\,\mathrm{rank}(B)$); so, we can write

$$\bar{f}(Ss_t, a_t) = SN\bar{S}^{-1}\bar{S}\phi(s_t, a_t) = Sf(s_t, a_t) \tag{26}$$

Using this representation, we can *learn* the linear part of f, $SN\bar{S}^{-1}$, directly from our state estimates: we just do a linear regression from $\phi(Ss_t, a_t)$ to Ss_{t+1}.

Algorithm 2 Robot System Identification

In: T *i.i.d.* pairs of observations $\{o_t, a_t\}_{t=1}^T$, measurement model for 4 landmarks $\mathcal{C}_{1:4}$ (by e.g. GPS)

Out: measurement model $\widehat{C}$, motion model $\widehat{N}$, robot states $\widehat{X}$ (the tth column is state s_t)

1: Collect observations and odometry into a matrix $\widehat{Y}$ (Eq. 8)
2: Find the the top 7 singular values and vectors: $\langle \widehat{U}, \widehat{\Lambda}, \widehat{V}^\top \rangle \leftarrow \mathrm{SVD}(\widehat{Y}, 7)$
3: Find the transformed measurement matrix $\widehat{C}S^{-1} = \widehat{U}$ and robot states $S\widehat{X} = \widehat{\Lambda}\widehat{V}^\top$
4: Compute a matrix Φ with columns $\Phi_t = \phi(Ss_t, a_t)$.
5: Compute dynamics: $S\widehat{N}\bar{S}^{-1} = S\widehat{X}_{2:T}(\Phi_{1:T-1})^\dagger$
6: Compute the partial S^{-1}: $\widehat{S}^{-1} = \mathcal{C}_{1:4}^{-1}(\widehat{C}_{1:4}S^{-1})$ where $\widehat{C}S^{-1}$ comes from step 3. $\widehat{S}^{-1}\widehat{X}$ gives us the x, y coordinates of the states. These can be used to find $\widehat{X}$ (see Section 3.2)
7: Given $\widehat{X}$, we can compute the full S as $S = (S\widehat{X})\widehat{X}^\dagger$
8: Finally, from steps 3,5, and 7, we find the interpretable measurement model $(\widehat{C}S^{-1})S$ and motion model $N = S^{-1}(SN\bar{S}^{-1})\bar{S}$.

For convenience, we summarize the entire learning algorithm (state space discovery followed by system identification) as Algorithm 1.

C.0.2. Filtering with the Extended Kalman Filter

Whether we learn the dynamics through system identification or simply write them down in the interpretable version of our state space, we will end up with a transition model of the form (23) and an observation model of the form (21). Given these models, it is easy to write down an EKF which tracks the robot state. The measurement update is just a standard Kalman filter update (see, e.g., (Thrun et al., 2005)), since the observation model is linear. For the motion update, we need a Taylor approximation of the expected state at time $t+1$ around the current MAP state $\hat{s}_t$, given the current action a_t:

$$s_{t+1} - s_t \approx N[\phi(\hat{s}_t, a_t) + \tfrac{d\phi}{ds}\big|_{\hat{s}_t}(s_t - \hat{s}_t)] \tag{27}$$

$$\tfrac{d\phi}{ds}\big|_{\hat{s}} = (\hat{s} \otimes I + I \otimes \hat{s}) \otimes \bar{a}_t \otimes \bar{a}_t \tag{28}$$

We simply plug this Taylor approximation into the standard Kalman filter motion update (e.g., (Thrun et al., 2005)).

D. Synthetic Experiments

Our simulator randomly places 6 landmarks in a 2-D environment. A robot then randomly moves through the environment for 500 time steps and receives a range reading to each one of the landmarks at each time step. The range readings are perturbed by noise sampled from a Gaussian distribution with variance equal to 1% of the range. Given this data, we apply the algorithm from Section 3.3 to solve the SLAM problem. We use the coordinates of 4 landmarks to learn the linear transform S and recover the true state space, as shown in Figure 1A. The results indicate that we can accurately recover both the landmark locations and the robot path.

We also investigated the empirical convergence rate of our observation model (and therefore the map) as the number of range readings increased. To do so, we generated 1000 different random pairs of environments and robot paths. For each pair, we repeatedly performed our spectral SLAM algorithm on increasingly large numbers of range readings and looked at the difference between our estimated measurement model (the robot's map) and the true measurement model $||\widehat{C} - C||_{\mathcal{F}}$. The results are shown in Figure 1B, and show that our estimates steadily converge to the true model, corroborating our theoretical results (in Section 3.3 and the Appendix).