Discovering Sensor Space: Constructing Spatial Embeddings That Explain Sensor Correlations

Joseph Modayil
Reinforcement Learning and Artificial Intelligence Laboratory
Department of Computing Science, University of Alberta, Canada

***Abstract*—A fundamental task for a developing agent is to build models that explain its uninterpreted sensory-motor experience. This paper describes an algorithm that constructs a sensor space from sensor correlations, namely the algorithm generates a spatial embedding of sensors where strongly correlated sensors will be neighbors in the embedding. The algorithm first infers a *sensor correlation distance* and then applies the fast maximum variance unfolding algorithm to generate a distance preserving embedding. Although previous work has shown how sensor embeddings can be constructed, this paper provides a framework for understanding sensor embedding, introduces a sensor correlation distance, and demonstrates embeddings for thousands of sensors on intrinsically curved manifolds.**

I. Introduction

A fundamental task for a developing agent is to build models that explain its uninterpreted sensory-motor experience. This problem is important for a developing agent, as the representation provided by a low-level sensory-motor stream is not well-suited for reasoning about high-level abstractions such as time, space, objects, and intents. This paper proposes an algorithm for recovering a sensor space in which the agent constructs a low-dimensional spatial embedding of its sensors that models the correlations between sensors. This work improves on existing methods with a new approach to reasoning about sensor embedding that can reconstruct spatial embeddings that are intrinsically curved and tractably scales up to thousands of sensors. We demonstrate the new approach with simulated and real data, and discuss the implications of this work for developmental robotics.

Spatial models of sensor data are important for people as well as for robots. The cortical homunculus is a region of the human brain where the sensors from the skin surface of the body are smoothly mapped onto a region of the brain [1]. Similar spatial structures exist for the visual cortex and for the auditory cortex. This spatial distribution of sensors in the brain strongly suggests that the spatial organization of sensors is important for human information processing. Notably, these mappings do not exactly preserve the distances between sensors on the physical layout of the body but appear to be distorted in proportion to the number of sensors. Areas of interest are overrepresented both with the number of sensors and in the size of corresponding brain regions. Thus the fovea in visual system and the hand in the somatosensory system command relatively larger processing portions of the brain than their physical counterparts do in space.

The spatial organization of sensors is also relevant for many robot algorithms, including computing perceptual flow, safety reflexes, robot localization. These algorithms assume that the spatial organization of sensors is known *a priori*. Moreover, spatially organized sensors enable better generalization from past experience than element-wise distance comparisons [2].

Earlier work [3] has shown how a robot can self organize its sensory system and learn basic control-laws for navigation. That work demonstrated that sensor correlations can be used to find a spatial embedding for the sensors that strongly corresponds to the positions of the sensors in space. Although the earlier algorithm works on a wide variety of common robot sensor arrays including cameras, lasers, and sonar arrays, the approach has limitations. The algorithm relies on computing accurate distances between all sensors, and this is difficult when sensors are widely separated. It has scaling difficulties as it requires a dense matrix of m^2 distances for m sensors for dimensionality reduction. Although it has been demonstrated to recover sensor configurations on spaces with constant curvature (circles and planes), it has not been demonstrated to work on spaces with varying curvature such a the surface of a human or robot body.

This paper defines a distance between sensors under the assumption that the data is generated by an underlying Gaussian process. Gaussian process models have been successfully used for interpreting many kinds of sensor data due to their ability to robustly and efficiently represent commonly occurring covariance structures [4]. Starting from the common squared-exponential kernel for a Gaussian process, we define a *sensor correlation distance* from the sensor observations. Using this function, distances between sensors are computed for the k-nearest neighbors. Given this sparse set of distance constraints between sensors, we use the fast maximum variance unfolding algorithm [5] to construct a low-dimensional embedding that approximately preserves the sparse distance constraints while maximizing the overall distance between sensors. Although the algorithm relies on the computationally expensive process of semi-definite programming, the computational complexity is circumvented by representing the sparse distance constraints with another low-dimensional approximation.

Perhaps surprisingly, this sequence of approximations preserves the essential structure of the problem and provides a scalable and effective algorithm for computing a low-dimensional sensor embedding that preserves the sensor correlation distances. While the constructed manifold may not be isometric to the positions of the sensors in physical space, it preserves an empirically meaningful distance between sensors. We evaluate this algorithm on synthetic and real data to demonstrate its performance.

The following section describes the background to the problem in more detail. We then describe the theoretical foundations of the method. The experiments are presented in the following section, and the paper concludes with a discussion of the results.

II. Background

We assume that the robot can only interpret its environment through its sensors and motors, and the robot has no *a priori* understanding of the semantics of the sensors. Each sensor $i \in \{1, \ldots, m\}$, at each time step t, generates a real valued observation $z_t^i \in \mathbb{R}$. Denote the entire time-series of observations from sensor i by z_*^i. We make no explicit assumptions on how the observations are gathered, so the observations could arise from some random exploration in the environment.

The task of sensor embedding is formalized as finding a function e that maps the m sensors into a low dimensional embedding space $\mathbb{R}^n$ with $n \ll m$ (for example $m = 1000$ and $n = 3$).

$$e : \{1, \ldots, m\} \to \mathbb{R}^n \tag{1}$$

The embedding can be a partial function where only some of the sensors are embedded into the same space. This is important when a robot has many weakly related sensors (for example sight and touch), and so multiple embeddings may be defined. This can also be relevant for detecting if the sensory streams come from disconnected physical bodies. Without loss of generality, we consider only one embedding at a time.

Since correlation can be used to define a distance, it is natural to ask if there is an embedding e that explains the correlation between sensors.

$$\text{corr}(z_*^i, z_*^j) \propto \exp(-||e(i) - e(j)||^2) \tag{2}$$

This condition is often satisfied in problems modelled by Gaussian processes, where sensor observations can be accurately predicted from knowledge of the environment and the sensor's spatial position.

An embedding can be evaluated in multiple ways. One objective is to accurately recover the physical position of the sensor on the robot [3], [6]. This objective has the benefit that it can be externally evaluated using a robot's manufacturing specifications. Knowledge of the physical placement of a sensor combined with knowledge of the semantics of a sensor reading enable inferences about the structure of the environment, and thus provides the foundation of most mapping algorithms.

Although we do not address the issue in this work, evaluating the recovered geometry based solely on sensor position is limited. One issue is that sensor orientation is as relevant as position for image pixels on cameras, sonar sensors on circular robots, and laser rangefinders. As sensors are often uniformly distributed in both position and orientation in manufactured sensor arrays, further validation is required with irregularly distributed sensors. A second issue is that externally evaluated isometry is often a proxy for measuring the internal utility of the constructed representations for supporting inference and action. Previous research explores this issue with broken-sensor elements [3], solving explicit tasks with the recovered representations [6], and generalizing human motions [7].

III. Theory

The algorithm presented in this section constructs an embedding that approximately preserves a distance between sensors defined from correlations in the observed sensor readings. The *sensor correlation distance* (SCD) embedding constructs a spatial embedding in three steps.

A. The sensor correlation distance is computed between pairs of sensors from a set of sensor observations.
B. Using the sensor correlation distance, the k-nearest neighbors for each sensor are selected to form a sparse set of distance constraints.
C. The fast maximum variance unfolding algorithm [5] is applied to the sparse constraints to generate a spatial embedding of the sensors that approximately preserves the selected sensor correlation distances.

A. The sensor correlation distance

We assume that the agent observes the world with m sensors. We also assume that the sensors have zero mean and unit variance. Let z_t^i be the observed sensor reading from sensor i at time t, and let Z_t^i be the associated random variable. Using a squared-exponential kernel Gaussian process model [4], we assume that the expected covariance (cov_E) of between Z_t^i and Z_u^j is based on a distance in time d_T and space d_S with

$$\text{cov}_E(Z_t^i, Z_u^j) = \sigma_f^2 \exp\left(-d_S(i,j)^2 - d_T(t,u)^2\right) + \delta_{i,j}\delta_{t,u}\sigma_n^2. \tag{3}$$

In the above formula, σ_f^2 corresponds to systematic variance and σ_n^2 corresponds to observation noise. The noise is restricted to a single observation, as δ is the Kronecker delta function that is one when its two arguments are equal and zero otherwise. For example, the distance function d_S could be Euclidean distance between sensors and d_T could be the absolute value of the difference in time.

We note a few identities. First recall that the sample correlation is related to the sample covariance (cov_S) by normalization, so with unit variances they will be the same.

$$\text{corr}(z_*^i, z_*^j) = \frac{\text{cov}_S(z_*^i, z_*^j)}{\sqrt{\text{cov}_S(z_*^i, z_*^i)\text{cov}_S(z_*^j, z_*^j)}} \tag{4}$$

Also, note that the correlation between two time series with zero mean and unit variance is their inner product.

$$\text{corr}(z_*^i, z_*^j) = < z_*^i, z_*^j > \tag{5}$$

Using the observed correlation for the expected covariance, we can attempt to invert Equation 3 to solve for a relevant d_S. Considering only the covariance between observations taken at the same time t but different sensors $i \neq j$, we find

$$\text{cov}_E(Z_t^i, Z_t^j) = \sigma_f^2 \exp(-d_S(i,j)^2) \tag{6}$$

which can be rewritten as

$$d_S(i,j) = \sqrt{-\ln(\text{cov}_E(Z_t^i, Z_t^j)/\sigma_f^2)}. \tag{7}$$

Using the above identities for sample correlation and covariance, and replacing the expected covariance with the sample covariance, we define the *sensor correlation distance* (SCD) between sensors,

$$SCD(i,j) = \sqrt{-\ln(\frac{1}{T}\sum_{t=1}^{T} z_t^i z_t^j)}. \tag{8}$$

This sensor correlation distance function is different from the distance function used by Pierce and Kuipers.

$$d_{PK}(i,j) = \frac{1}{T}\sum_{t=1}^{T} |z_t^i - z_t^j| \tag{9}$$

Several other distance functions could be considered, such as an information distance [8], but the generality of the GP model suggests that it may be widely applicable.

This expression for SCD ignores the sensor noise σ_n as being irrelevant. However, the inferred distances are subject to noise, and at large distances the sensor noise from the σ_n^2 term will dominate the covariance, making estimation of the distance between distant sensors highly inaccurate. As this approach focuses on recovering the distances between nearby sensors, inaccurate inferred distances for widely separated sensors are not a problem for the algorithm.

The expression for SCD also ignores the fact that the observations are not independent in time, but the d_T factor will slow the rate of convergence of the sample correlation to the expected covariance. Assuming that $d_T(t,t') \geq k_T|t-t'|$, the speed of convergence is governed by k_T.We present a brief sketch for why the convergence rate should evolve as $O(1/\sqrt{n})$. If $k_T = \infty$, then the observations are independent in time. Given independent identically distributed data with bounded variance, then the central limit theorem implies that the sample mean converges to the expected mean at a rate of $1/\sqrt{n}$ (see [9] Ch 2, Thm 4.1). For finite k_T, then for any $\epsilon > 0$, we can find a time τ such that for all j, $\text{cov}_E(Z_t^i, Z_{t+\tau}^j)$ is less than ϵ. Using a subsequence with a step size of τ yields essentially independent random variables which will converge as $O(1/\sqrt{n})$. As the convergence of the subsequence is independent of starting position, the full sequence should converge at a rate of $O(1/\sqrt{n})$.

B. Selecting k neighbors

The estimated distance is used to find the k-nearest neighbors for each sensor, thus forming a sparse set of distance constraints between sensors. The restriction to k neighbors guarantees that the number of constraints is linear in the number of sensors. Selecting k neighbors from the set of sensors is straightforward given the sensor distances. By computing all pairwise distances, this $O(m^2T)$ step is the most computationally expensive in the current algorithm. However, there are potentially faster $O(mT \log m)$ algorithms that do not compute all pairwise distances [10].

C. Isometric Embedding

To convert a weighted graph of inferred distances into a low-dimensional embedding, we rely on the technique of fast maximum variance unfolding [5]. This algorithm reduces the computational complexity of semidefinite programming for embedding sensor networks [11]. The algorithm takes as input a sparse set of distance constraints over a large graph and generates as output a low-dimensional embedding.

We present a brief overview of embedding by semidefinite programming, which is an effective way to recover the geometric structure of sensor networks [11]. Local distance constraints between sensors i and j can be represented by $||x_i - x_j||^2 = d_{ij}^2$. Finding positions x_i to optimize this constraint directly is difficult as the equations are non-convex. However, with inner products $X_{ij} = < x_i, x_j >$ the constraints become

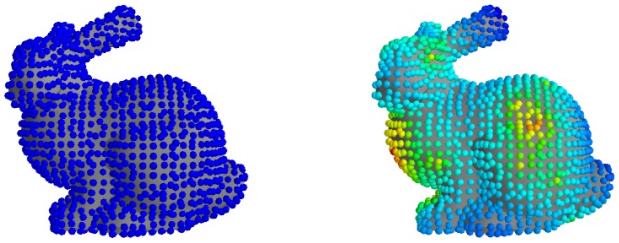
$$|X_{ii} - 2X_{ij} + X_{jj} - d_{ij}^2| = 0, \tag{10}$$

and the hard constraints are softened to an optimization [5].

$$\begin{array}{ll} \text{Maximize} & \text{tr}(X) - \nu \sum_{i\sim j}(X_{ii} - 2X_{ij} + X_{jj} - d_{ij}^2)^2 \\ \text{subject to} & (i)\sum_{ij} X_{ij} = 0 \text{ and } (ii) X \succeq 0 \end{array} \tag{11}$$

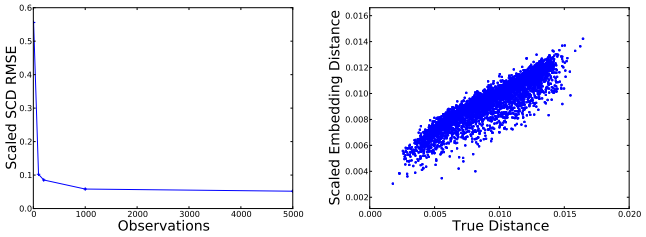
This optimization attempts to maximize the spread of the embedded points ($\text{tr}(X)$) while also minimizing violations of the local distances (weighted by the factor $\nu > 0$). The first constraint centers the point around the origin. The second constraint is that the matrix X_{ij} is symmetric and positive semi-definite, which is equivalent to being a Gramian matrix, namely a matrix of inner products between vectors (an easy result, see for example [12]).

These constraints form a convex semi-definite program which can be solved, although with high complexity in the number of constraints. As is shown in [5], the constraints can be approximated using a semidefinite program with far fewer constraints by approximating the $a \times a$ matrix X with a $b \times b$ matrix Y where $1000 \approx a \gg b \approx 10$, with $x_i \approx \sum_{\alpha=1}^{b} Q_{i\alpha} y_\alpha$. The matrix Q comes from Laplacian eigenvectors for the unweighted sparse connectivity graph of distance constraints, and can be computed at low cost. Additional manipulations are required to remove Q from the semidefinite program and to re-express the original constraints. The problem is thus transformed into a much smaller semidefinite program, one with a $b^2 \times b^2$ matrix inequality with approximately b^2 variables. As this formulation is independent of a,



(a) (b)

Fig. 1. (a) The Stanford bunny with virtual sensors at vertices. (b) A single observation of the virtual sensors. Virtual observations are generated by a sum of exponentially decaying activations starting at randomly selected vertices.



(a) (b)

Fig. 2. Errors in the estimated distances between sensors on the bunny. (a) The number of observation timesteps (horizontal axis) and the root mean-square error of inferred sensor correlation distance to the true sensor distances (vertical axis). (b) A scatterplot of the true distances (horizontal axis) and the scaled embedded distances (vertical axis). Quantitatively, the relative error for the k nearest neighbors is 22%.

the final semi-definite program can be solved with a run-time that is independent of a. For our needs, $a = km$ corresponds to the number of sensor constraints, so the small b approximation allows this expensive optimization to be independent of the number of sensors. The number of eigenvectors required for recovering a manifold does not depend on the number of sensors, but on the complexity of the underlying manifold.

Post-processing steps convert the recovered solution into a solution of the original sensor layout problem. Projecting down to a reduced set of eigenvectors introduces distortions and a conjugate gradient method is used to reduce local errors introduced by the approximation.

The current implementation of the algorithm builds on existing modules. Semidefinite programming is solved by the CSDP library [13], and the conversion of sparse distance constraints into a spatial embedding is performed by the FastMVU Matlab implementation [5]. Although the algorithm is $O(b^4)$ in the size of the reduced problem, in practice the implementation for FastMVU converges in seconds with $b = 13$ and in minutes for $b = 20$ on a 2.5 GHz processor.

IV. Experiments

We evaluated the algorithm on simulated and real data. The simulated data comes from placing virtual sensors on vertices of a commonly used 3D model. The real data comes from a public dataset of video frames from an omnidirectional camera mounted on a mobile robot [14].

A. Simulated sensors

We use the Stanford bunny 3D polygon model [15] at medium resolution (1889 nodes) for our experiments with simulated data. The original bunny is shown in Figure 1(a), and has a size of approximately $0.15 \times 0.15 \times 0.12$ virtual units. The bunny is a complex surface, including holes on the bottom surface. A simulated sensor is placed at each vertex of the bunny. At each time-step t, sensor observations are generated by summing exponentially decaying spreading activations from thirty nodes, c_l, selected randomly with replacement, with each selection having an activation radius, r_t^l drawn from a Gaussian distribution. Letting $e(i)$ be the position of sensor i, the formulas for data generation are listed below.

$$l \in \{1, \ldots, 30\} \tag{12}$$

$$r_t^l \sim N(.01, .002), \quad c_l \sim \text{uniform}\{1, \ldots, m\}, \tag{13}$$

$$z_t^i = \sum_{l=1}^{30} \exp\left(-\frac{||e(i) - e(c_l)||^2}{(r_t^l)^2}\right) \tag{14}$$

An example of the simulated observations is shown in Figure 1(b). Observations are collected for 5000 timesteps, where the data at each timestep is independent. The rate at which the sensor correlation distance converges to the inter-sensor distances is shown in Figure 2(a).

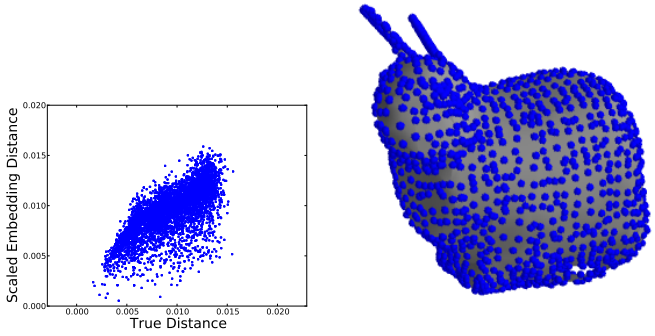
The $k = 20$ nearest neighbors for each sensor were selected to form a sparse graph of distance constraints. The fast maximum variance unfolding algorithm was used to generate an embedding from the sparse set of constraints (using 20 Laplacian eigenvectors for the reduced space). The distribution between sensor distances in the original and recovered models is shown in Figure 2(b) (the recovered distances are scaled to have the same mean as the original). The distribution of data on the graph shows that the true distance and the inferred distances are strongly correlated. For quantitative comparison, we first rescale the estimated distances to match the mean of the true distances. We then compare the estimated values with the true values by

$$Rel\ Err = (\text{true} - \text{estimate})/\text{true}. \tag{15}$$

Using this measure, the average relative error for the distances in the k nearest neighbors is 22%.

The final embedding is shown in Figure 3. As the figure shows, the gross structure of the figure is recovered. The figure appears to be inflated internally, possibly due to the maximum-variance objective increasing the distance between points. The figure also has distortions around the ears and loss of detail on the body, presumably due to the small number of eigenvectors used in the approximation. Using fewer neighbors or fewer eigenvectors resulted in reconstructions that were visibly and quantitatively worse. For comparison, Figure 4 shows the results from using Isomap instead of FastMVU. With Isomap, the computed distances and recovered geometry have more visible errors.

Fig. 3. The bunny reconstructed by SCD embedding. The approximate geometry of the bunny has been recovered, although the bunny appears inflated and somewhat distorted. The embedding recovers only the positions of the nodes, the triangulated surface mesh is



(a) (b)

Fig. 4. The bunny reconstructed with Isomap instead of FastMVU. (a) The recovered local distances have larger errors than under FastMVU. (b) The Isomap embedding produces a body with few details and pointed ears.

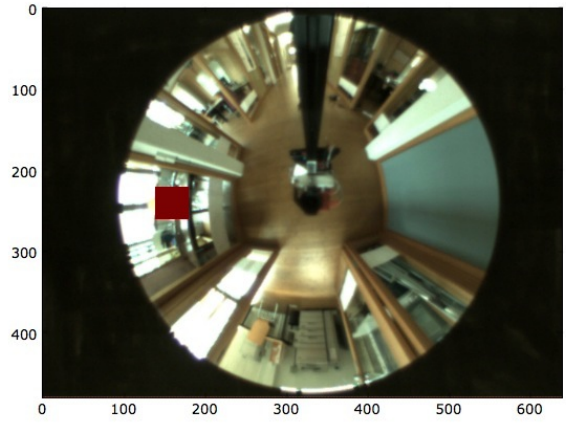
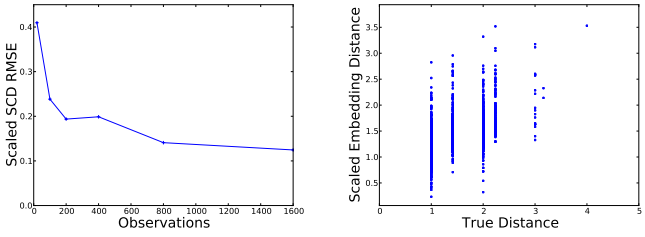


Fig. 5. A sample frame from the image sequence with the region of selected pixels shown in red.



(a) (b)

Fig. 6. Convergence of SCD embedding to pixel distances. (a) The rate at which SCD converges to pixel distances with the number of frames in RMSE. The convergence is slower than for the bunny, potentially due to image noise and correlations in time, (b) The scatter plot for the embedded distances and pixel distance shows a large spread in the reconstruction.

B. Image sensors

We demonstrate the SCD embedding on a 40×40 pixel window of video data. This data comes from a publicly available video dataset [14] (saarbrucken/cloudy sequence1) containing images from an omnidirectional camera mounted on a robot moving between several rooms. A sample image is shown in Figure 5 with the selected 40×40 window. The selected sensors are restricted to the red pixels, and the image sequence is 1297 frames long.

The rate of convergence between the measured SCD distance and the true pixel distance is shown in Figure 6(a). The error reduces more slowly than on the simulated data, presumably due to the presence of temporal correlations and observation noise. The $k = 10$ nearest neighbors were used to form the sparse constraint graph. For the fast maximum variance unfolding algorithm, the first 13 Laplacian eigenvectors were used. Figure 6(b) shows the correlation between true pixel distance and reconstructed embedding distance at the end of the video sequence. The discrete jumps in the true distances make the correlation difficult to visualize. Quantitatively, the method performs reasonably with a relative error for the embedding distances of 25%. Qualitatively, the embedding recovers an approximately correct structure, shown in Figure 7.

V. Discussion

Other methods for constructing low-dimensional sensor representations do not solve the task presented in this paper. Recovery of non-uniformly curved manifolds is not provided by other local distance methods such as ISOMAP [16], or locally linear embedding [17]. Olsson and colleagues [8] construct an embedding with information distance instead of SCD. Although their information distance can handle situations not covered by correlation distance (recovering a single embedding for multiple sensor types), it can also generate undesirable embeddings where negatively correlated sensors become neighbors. Previous research in the area of sensor network localization [11], [5] established the utility of semidefinite programming for recovering sensor geometry when distances are a direct function of the sensor's measured value. and not a function of sensor correlations. Other research has used similar techniques on related problems. Action respecting embedding [18] also uses semi-definite programming, but their

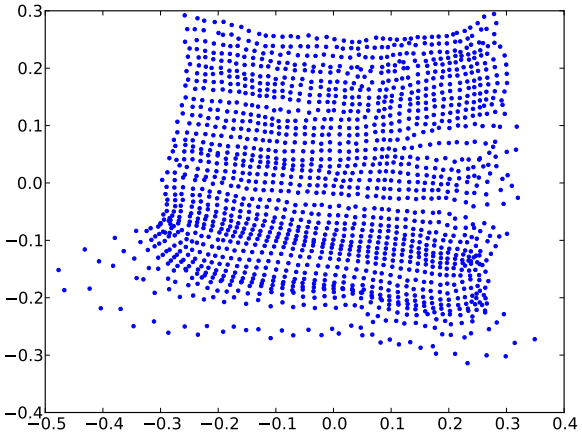


Fig. 7. The SCD embedding reconstructs the coarse structure of the original pixel array but with some distortions.

goal is to recover the global configuration of robot poses that are related by a set of discrete actions. Philipona et al. [19] recover a manifold structure using a Lie algebra induced by actions, but it is not clear if this can be applied to surfaces with non-uniform curvature.

As mentioned earlier, although the SCD embedding may not be isometric to the physical layout of the sensors, the recovered geometry captures some intrinsic structure of the data. Potentially, this method could be used for registering two robots with similar geometry and sensor distributions but distinct sensor placements. One possibility is to simplify the manufacture of robot skin by having sensors randomly distributed within a skin, and then using the SCD embedding to recover the location of the sensors.

Another direction for future research is to explore how the recovered sensor geometry can assist in performing tasks, such as learning to chase an object or to scratch an itch. Following the approach of [3], the recovered geometry could be used to define local control laws or to re-represent the motor system in terms of controlling spatially-defined perceptual features such as maxima, minima, and discontinuities.

There are some weak implications of this work for biological systems. This work shows that there can be enough information in the observations of spatially distributed sensors to recover an approximate sensor geometry. However, using these algorithms directly could be difficult for a brain. A brain might also use one sensing modality is used to bootstrap another (for example an eye can see that a finger is touching a wrist). Alternatively, either in the course of evolution or in an agent's development, an online algorithm might adapt brain processing regions to handle new sensors that are added to the physical body.

VI. Conclusion

The SCD embedding constructs a spatial embedding of the sensors where the observed correlation between sensors is a function of the distance between the sensors in the embedding. The function that relates the sensor correlation with the distance is based on the commonly used squared-exponential kernel Gaussian process model, and thus may be applicable to many problems. This method also provides guidance for the rate of convergence of the sensor distances. Using the maximum variance unfolding algorithm, this technique can generate spatial embeddings for intrinsically curved structures. In practice, this method scales to thousands of sensors and is robust enough to generate approximately correct embeddings from real sensor data.

VII. Acknowledgements

This work has been supported by iCORE (part of Alberta Innovates – Technology Futures), NSERC, and MITACS. This work was performed in part at the University of Rochester and supported by NYSTAR and a gift from Kodak. Thanks to Benjamin Kuipers, Randal Nelson, Dale Schuurmans, Csaba Szepesvari, and Yasin Abbasi for their insights, and to the anonymous reviewers for their detailed comments.

References

[1] A. Nakamura, T. Yamada, A. Goto, T. Kato, K. Ito, Y. Abe, T. Kachi, and R. Kakigi, "Somatosensory homunculus as drawn by MEG," *Neuroimage*, vol. 7, pp. 377–386, 1998.

[2] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Proceedings of the IEEE Int. Conf. on Computer Vision*, 2005.

[3] D. M. Pierce and B. J. Kuipers, "Map learning with uninterpreted sensors and effectors." *Artificial Intelligence*, vol. 92, pp. 169–227, 1997.

[4] C. Rasmussen and C. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[5] K. Weinberger, F. Sha, Q. Zhu, and L. Saul, "Graph Laplacian regularization for large-scale semidefinite programming," in *Advances in Neural Information Processing Systems 19*, B. Schölkopf, J. Platt, and T. Hofmann, Eds. Cambridge, MA: MIT Press, 2007, http://www.cse.wustl.edu/∼kilian/Downloads/FastMVU.html.

[6] J. Stober, L. Fishgold, and B. Kuipers, "Sensor map discovery for developing robots," in *AAAI Fall Symposia Series: Manifold Learning and Its Applications*, 2009.

[7] O. Jenkins and M. Matarić, "A spatio-temporal extension to Isomap," in *Proc. of the 21 International Conference on Machine Learning*, 2004.

[8] L. Olsson, C. Nehaniv, and D. Polani, "From unknown sensors and actuators to actions grounded in sensorimotor perceptions," *Connection Science*, vol. 18, no. 2, pp. 121–144, June 2006.

[9] R. Durrett, *Probability: Theory and Examples*. Duxbury Press, 1996.

[10] P. Vaidya, "An $O(n \log n)$ algorithm for the all-nearest-neighbors problem," *Discrete & Computational Geometry*, vol. 4, pp. 101–115, 1989.

[11] P. Biswas, T.-C. Liang, K.-C. Toh, T.-C. Wang, and Y. Ye, "Semidefinite programming approaches for sensor network localization with noisy distance measurements," *IEEE Transactions on Automation Science and Engineering*, vol. 3, no. 4, pp. 360–371, 2006.

[12] S. Friedberg, A. Insel, and L. Spence, *Linear Algebra*. Prentice Hall, 1989.

[13] B. Borchers, "CSDP, a C library for semidefinite programming," *Optimization Methods and Software*, vol. 11, no. 1, pp. 613–623, 1999.

[14] A. Pronobis and B. Caputo, "COLD: The CoSy localization database," *International Journal of Robotics Research*, pp. 588–594, 2009.

[15] "The Stanford 3D scanning repository," Website http://graphics.stanford.edu/data/3Dscanrep/.

[16] J. B. Tenenbaum, V. de Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, pp. 2319–2323, 2000.

[17] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, pp. 2323–2326, 2000.

[18] M. Bowling, D. Wilkinson, A. Ghodsi, and A. Milstein, "Subjective localization with action respecting embedding," in *Proceedings of the International Symposium of Robotics Research (ISRR)*, 2005.

[19] D. Philipona, J. O'Regan, J. Nadal, and O.-M. Coenen, "Perception of the structure of the physical world using unknown multimodal sensors and effectors," *Advances in Neural Information Processing Systems*, 2004.