A Manifold Regularization Approach to Calibration Reduction for Sensor-Network Based Tracking

Jeffrey Junfeng Pan, Qiang Yang, Hong Chang and Dit-Yan Yeung
Dept. of Computer Science, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong
{panjf, qyang, hongch, dyyeung}@cs.ust.hk

Abstract

The ability to accurately detect the location of a mobile node in a sensor network is important for many artificial intelligence (AI) tasks that range from robotics to context-aware computing. Many previous approaches to the location-estimation problem assume the availability of calibrated data. However, to obtain such data requires great effort. In this paper, we present a manifold regularization approach known as `LeMan` to calibration-effort reduction for tracking a mobile node in a wireless sensor network. We compute a subspace mapping function between the signal space and the physical space by using a small amount of labeled data and a large amount of unlabeled data. This mapping function can be used online to determine the location of mobile nodes in a sensor network based on the signals received. We use Crossbow MICA2 to setup the network and USB camera array to obtain the ground truth. Experimental results show that we can achieve a higher accuracy with much less calibration effort as compared to several previous systems.

Introduction

Wireless sensor networks have recently attracted great interests in AI communities. Many tasks ranging from robotics (Batalin, Sukhatme, & Hattig 2004; Biswas & Thrun 2005) to context-aware computing (Liao, Fox, & Kautz 2004; Lester *et al.* 2005) can now be realized with the help of distributed wireless sensor networks. Researchers have successfully applied learning techniques in a sensor network from localization (Nguyen, Jordan, & Sinopoli 2005) to activity recognition (Liao, Fox, & Kautz 2004). Many of these applications depend on the ability for a sensor node to be aware of its location.

Accurately tracking *mobile* nodes in wireless sensor networks using radio-signal-strength (RSS) values is a complex and difficult task. Radio signal usually attenuates in a way that is highly nonlinear and uncertain in a complex environment, which may be further corrupted when introducing the mobility of sensor nodes. In the past, many researchers have developed ranging-based algorithms for localizing mobile nodes in a wireless sensor network. These methods (Savvides, Han, & Strivastava 2001; Priyantha *et al.* 2001) usually consist of two main steps, by first transforming sensor reading into a distance measure and then recovering the most probable coordinates of sensor nodes. These approaches usually rely on a sophisticated signal propagation model and extensive hardware support. Learning-based approaches (Nguyen, Jordan, & Sinopoli 2005; Bahl & Padmanabhan 2000; Lorincz & Welsh 2006) can bypass the ranging process but need relatively more calibration data. However, few of them consider the mobility of sensor nodes. Even though (Moore *et al.* 2004) can accurately track *mobile* nodes, it needs special hardware such as ultrasonic transceivers. Adding the required hardware increases the cost and size of sensor nodes. How to estimate the location of mobile nodes when the signal environment is noisy, and when we have much less calibrated (labeled) data and hardware support, is still an open problem.

In this paper, we address the calibration-effort reduction problem in sensor-network based tracking by proposing a semi-supervised learning (Blum & Mitchell 1998) approach for learning a mapping between the signal space and the physical space, which is essentially a regression problem. Our approach, called `LeMan`, is based on *manifold regularization* (Belkin, Niyogi, & Sindhwani 2005), which exploits the manifold structure of data to avoid poor generalization due to limited labeled data. We first use a mobile robot to collect a small quantity of labeled data at various locations. Then, we apply manifold regularization to solve the regression problem, in a semi-supervised manner, using a small amount of labeled data and a large amount of unlabeled data. The mapping function learned can be used online to determine the location of a mobile node in a sensor network based on the signals received.

We have tested our algorithm `LeMan` using a sensor network implemented using the Crossbow MICA2. Experimental results show that we can achieve a higher accuracy with much less calibration effort. We have found that with a much lower proportion of labeled data, `LeMan` achieves much higher accuracy than many state-of-the-art algorithms. We have also found that with an increase of the amount of unlabeled data, the accuracy can be greatly improved.

Background

Related Works in Localization and Tracking

Ranging-based approaches (Savvides, Han, & Strivastava 2001; Priyantha *et al.* 2001) are popular localization methods due to their simplicity and efficiency. Usually, they first

estimate the distance of an unknown-location sensor node to some n *beacon* nodes ($n \geq 3$) based on certain sensor reading, e.g. RSS values, and formulae that model the signal propagation patterns. After that, the most probable location of the unknown-location sensor node can be calculated by solving a system of equations (Savvides, Han, & Strivastava 2001). As mentioned above, these approaches prefer an ideal environment which is often not the case in reality (Nguyen, Jordan, & Sinopoli 2005).

In contrast to ranging-based approaches, learning-based methods to location estimation in a sensor network rely on models that are trained with collected data. The training data typically consist of RSS values associated with locations that are manually collected offline (Bahl & Padmanabhan 2000; Lorincz & Welsh 2006) or automatically collected online with GPS and mobile robots. When environmental factors combine together, they demonstrate some nonlinear statistical patterns, which makes it possible to encode these patterns using kernels (Nguyen, Jordan, & Sinopoli 2005). A major problem with these methods is that to collect training data requires much effort and costs. Furthermore, these offline data are subject to changes when the network is exposed to environmental dynamics.

A viable approach is to collect a small amount of labeled training data and supplement the lack of labels with a large amount of unlabeled data. (Shang *et al.* 2003) tries to recover all the locations of sensor nodes simultaneously through a multidimensional scaling (which is known as MDS). (Patwari & Hero 2005) further generalized the MDS approach to manifold learning-based localization through Laplacian Eigenmap (Belkin & Niyogi 2003). Their approach is able to obtain the global optimum with less computational overhead. However, many of these methods consider *static* sensor nodes only, which performance is unknown in a mobile scenario. The work of (Moore *et al.* 2004) can accurately track *mobile* nodes, but it requires some special hardware such as ultrasonic transceivers, which may not be available on sensor nodes. Adding the required hardware increases the cost and size of sensor nodes.

Manifold Regularization for Learning Functions

Many kernel methods such as support vector machines (SVM) (Burges 1998) can be formulated under the regularization framework. Specifically, the learning problem corresponds to minimizing some regularized risk functional which consists of an empirical risk functional and a regularizer. The empirical risk functional is defined based on some loss function for the labeled training data. Different loss functions are used for different learning problems. For example, the hinge loss function is used for classification problems and the squared loss function or ϵ-insensitive loss function is used for regression problems. The regularizer imposes appropriate constraints to control the complexity of the model and hence the complexity of the solution.

This paper is an attempt to learn a mapping between the signal space and the physical space in sensor-network based tracking under the semi-supervised learning setting. To avoid poor generalization due to limited labeled data, the learning problem is formulated under a regularization framework. Specifically, it is a data-dependent regularization framework called *manifold regularization* (Belkin, Niyogi, & Sindhwani 2005) that exploits the geometric structure of the marginal distribution of the data in the signal space. The basic underlying assumption of manifold regularization is that if two points are close in the intrinsic geometry of the marginal distribution, then their conditional distributions are similar. For classification problems, this implies that they are likely to have the same label. For regression problems, their regression function values are similar. Manifold regularization introduces to the regularized risk functional an additional regularizer that serves to impose this assumption on the learning problem.

Let us now define the learning problem more formally. Suppose we have a set of l labeled examples $\{(r_i, z_i)\}_{i=1}^{l}$ and a set of u unlabeled examples $\{r_j\}_{j=l+1}^{l+u}$, where r_i and r_j are sampled from the input space $\mathcal{X}$ according to the marginal distribution $\mathcal{P}_{\mathcal{X}}$ and $z_i \in \mathbb{R}$ is governed by the conditional distribution $\mathcal{P}(z|r)$. The learning problem corresponds to solving the following optimization problem:

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(r_i, z_i, f) + \gamma_A \|f\|_K^2 + \gamma_I \int_{\mathcal{M}} \langle \nabla_{\mathcal{M}}, \nabla_{\mathcal{M}} \rangle, \quad (1)$$

which finds the optimal function f^* in the reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$ of functions $f : \mathcal{X} \to \mathbb{R}$ corresponding to a Mercer kernel $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.

The first term of the regularized risk functional is defined based on the loss function V which measures the discrepancy between the predicted value $f(r_i)$ and the actual value z_i. The second term controls the complexity of f in terms of the norm $\|\cdot\|_K$, with γ_A being the regularization parameter. The third term is specific to manifold regularization and is based on the assumption that the support of $\mathcal{P}_{\mathcal{X}}$ forms a compact submanifold $\mathcal{M}$. It controls the complexity of f in the intrinsic geometry of $\mathcal{P}_{\mathcal{X}}$, with γ_I being the corresponding regularization parameter.

In (Belkin, Niyogi, & Sindhwani 2005), the third term is approximated using the graph Laplacian (Chung 1997) defined on all $l + u$ labeled and unlabeled examples without using the label information. Hence the optimization problem can be reformulated as:

$$f^* = \arg\min_{f \in \mathcal{H}_K} \frac{1}{l} \sum_{i=1}^{l} V(r_i, z_i, f) + \gamma_A \|f\|_K^2 + \frac{\gamma_I}{(u+l)^2} \hat{f}^T L \hat{f}, \quad (2)$$

where $\hat{f} = (f(r_1), \ldots, f(r_{l+u}))'$ and L is the graph Laplacian.

From the extended Representer Theorem (Belkin, Niyogi, & Sindhwani 2005), the optimal function can be expressed in the following form:

$$f^*(r) = \sum_{i=1}^{l+u} \alpha_i K(r_i, r). \quad (3)$$

When focusing on regression, V in (2) is set to be the squared loss function $V(r_i, z_i, f) = (z_i - f(r_i))^2$ to give the Laplacian Regularized Least Squares (LapRLS) algorithm (Belkin, Niyogi, & Sindhwani 2005). It can be shown that the optimal solution $\alpha^* = (\alpha_1^*, \ldots, \alpha_{l+u}^*)'$ is given by

$$\alpha^* = (JK + \gamma_A l I + \frac{\gamma_I l}{(u+l)^2} LK)^{-1} Z, \quad (4)$$

where K is the $(l+u) \times (l+u)$ Gram matrix over all labeled and unlabeled examples, Z is an $(l+u)$-dimensional label vector given by $Z = (z_1, \ldots, z_l, 0, \ldots, 0)'$, and $J = \text{diag}(1, \ldots, 1, 0, \ldots, 0)$ is an $(l+u) \times (l+u)$ diagonal matrix with the first l diagonal entries being 1 and the rest being 0.

Methodology

Problem Statement

Consider a two-dimensional tracking problem[1]. Assume that there are N sensor nodes fixed in an area $\mathcal{C} \subseteq \mathbb{R}^2$ that we are interested in. These *beacon* nodes periodically send out beacon signals to other nodes. The locations of *beacon* nodes are not necessarily known. There are one or more *mobile* nodes of unknown locations. At time t, a *mobile* node can measure the RSS sent by the N *beacon* nodes by detecting their signals which gives a vector $\mathbf{s}_t = (s_{t1}, s_{t2}, \ldots, s_{tN})' \in \mathbb{R}^N$. In addition, we have collected a small amount l of labeled training data which are signal-location pairs $\{(\mathbf{s}_{t_i}, \boldsymbol{\ell}_{t_i})\}_{i=1}^{l}$ at various locations, and u unlabeled data $\{\mathbf{s}_{t_j}\}_{j=l+1}^{l+u}$.

Our objective is to determine the location $\boldsymbol{\ell}_t = (x_t, y_t)' \in \mathcal{C}$ of a *mobile* node based on the signal vector $\mathbf{s}_t$ measured at time t. Figure 3 shows an example of the floor in our experimental test-bed, which consists of $N = 8$ *beacon* nodes and one *mobile* node.

Domain Characteristics

To establish the feasibility of our manifold regularization approach, we first examine the signal distribution properties of the sensor-network environment. Figure 2(a) shows the distance-signal correspondence between a *mobile* node and a *beacon* node. As can be seen, the signal attenuates in a way that is highly nonlinear and noisy. The uncertainty is relatively larger at a farther distance. Even when the *mobile* node is fixed in one location, the signal would not be stable.

When there is little noise, every two-dimensional location would uniquely determine a signal vector in a high-dimensional signal space and the localization area would be mapped to a two-dimensional manifold. However, when we take into account the uncertainty introduced by environment noise and node mobility, the manifold is corrupted. However, the manifold regularization is still feasible for our problem. As shown in Figure 2(b), the average signal change between two time points which is measured by Euclidean norm $\|\mathbf{s}_{t_i} - \mathbf{s}_{t_j}\|$, grows with the physical distance $\|\boldsymbol{\ell}_{t_i} - \boldsymbol{\ell}_{t_j}\|$. Similarly, the signal change also grows with the time interval $|t_i - t_j|$ because of robot movement, which is shown in Figure 2(c). These two figures support the fact that neighboring locations have more similar signal vectors than those that are far away. This fact also holds at the dimension of time since a *mobile* node can not move too fast. Furthermore, we found that such signal change has a stable statistical pattern. More specifically, the signal change is likely to obey a Gaussian distribution at a fix physical distance or time interval even if the signal itself is not Gaussian at a fixed location or time.

[1]Extension to the three-dimensional case is straight-forward

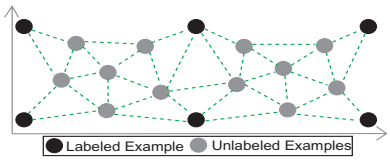


Figure 1: The use of labeled and unlabel examples

One example is shown in Figure 2(d). A more detailed hypothesis testing of the above hypothesis will be provided in our future work. Our problem basically satisfies the assumption of manifold regularization that similar signal vectors in the intrinsic geometry lead to similar location distribution. Figure 1 illustrates how the unlabeled examples can supplement the sparsity of labeled ones in the sample space. A dashed line shows that two examples are similar in signal or time dimensions.

The `LeMan` Algorithm

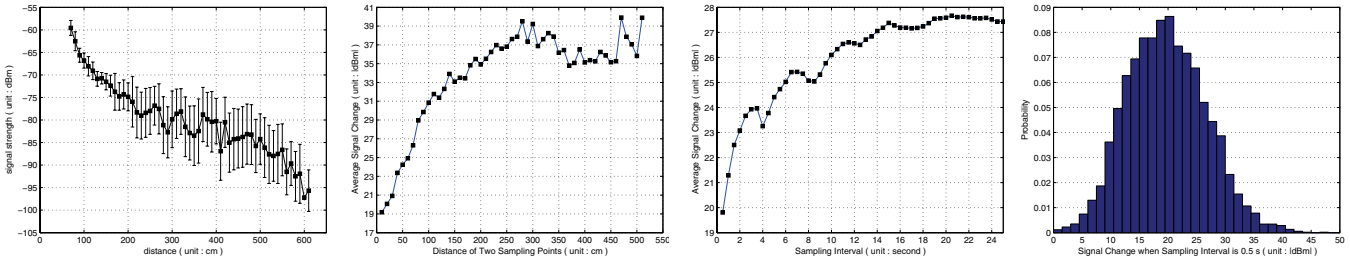
Our location estimation algorithm `LeMan`, which is based on manifold regularization, has two phases : an offline training phase and an online localization phase.

• Offline Training Phase

1. Collect l labeled signal-location pairs $\{(\mathbf{s}_{t_i}, \boldsymbol{\ell}_{t_i})\}_{i=1}^{l}$ at various locations and u unlabeled signal examples $\{\mathbf{s}_{t_j}\}_{j=l+1}^{l+u}$ with a mobile robot, on which top we attach a sensor node. The mobile robot runs and stops around the test-bed so that these signal vectors form a long trace.
2. Scale each component of every signal vector $\mathbf{s}$ to $[0, 1]$ where $\mathbf{s} \in S = \{(s_{t_p 1}, \ldots, s_{t_p N})\}_{p=1}^{l+u}$. This step is commonly used in kernel methods.
3. For each signal vector $\mathbf{s}_{t_p} \in S$, connect $\mathbf{s}_{t_p}$ to its k nearest neighbors, which distance is measured by Euclidean norm $\|\mathbf{s}_{t_p} - \mathbf{s}_{t_q}\|$ where $p, q \in [1, l+u]$. We further link those $\mathbf{s}_{t_p}$ and $\mathbf{s}_{t_q}$ together if $|t_p - t_q| < \Delta T$. These two kinds of connections are based on our analysis from Figure 2(b) and 2(c). After that, the adjacency graph and weight matrix can be used to form the graph Laplacian L.
4. Laplacian Regularized Least Square, with proper choice of kernel, is used to learn the mapping from signal vector $\mathbf{s}$ to location $\boldsymbol{\ell}$. In particular, the optimal α_x^* and α_y^* are obtained from Equation (4) for x and y coordinates.

• Online Localization Phase

1. At time t, the *mobile* node collects a signal vector $\tilde{\mathbf{s}}_t$. For those *beacon* nodes that are far away, which signals are too weak to detect, we fill in a small value, e.g. -95dBm.
2. Scale each component of $\tilde{\mathbf{s}}_t$ to $[0, 1]$ in a similar way as Step 2 in the training phase.
3. $\hat{\boldsymbol{\ell}}_t = (\hat{x}_t, \hat{y}_t)' = (f_{\alpha_x}^*(\tilde{\mathbf{s}}_t), f_{\alpha_y}^*(\tilde{\mathbf{s}}_t))'$ is the location estimation at time t, where α_x^* and α_y^* are obtained from Step 4 in the training phase and f^* is from Equation (3).
4. Optionally, applying a Bayes Filter (Fox *et al.* 2003; Hu & Evans 2004) would be a bonus. For example, Kalman Filter, which encodes a proper motion model, can be used to smooth the trajectory and enhance the performance of *any* localization algorithm.



(a) Signal attenuation (b) Signal change over space (c) Signal change over time (d) Distribution of signal change

Figure 2: Radio Characteristics

Experimental Setup

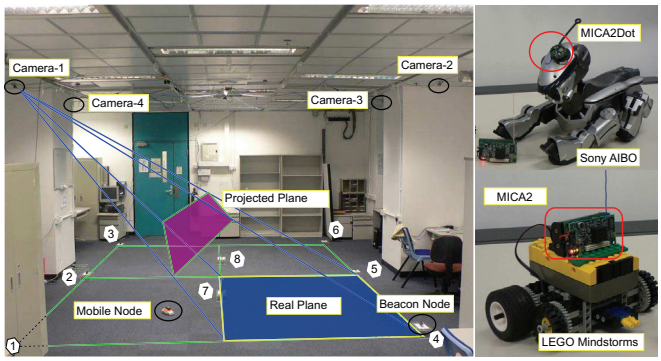


Figure 3: Experimental Test-Bed

Our experiment is performed in a laboratory of HKUST (Figure 3). The room is large enough for us to set up an experimental test-bed of 5.0 meter by 4.0 meter. In Figure 3, $\|P_1P_3\| = \|P_4P_6\| = 5.0m$ and $\|P_1P_4\| = \|P_3P_6\| = 4.0m$. There are three main components of our setup:

- *Wireless Sensor Networks* can be constructed with Crossbow MICA2 or MICA2Dot. Figure 3 shows that there are eight *beacon* sensor nodes deployed on the floor denoted as $1, \ldots, 8$, which are set to broadcast beacon signals periodically. *Mobile* nodes are attached on top of robots.
- *Mobile Robots* are used for supporting mobility. However, in practice, sensor nodes can be attached to any object. We tried different robots that can run freely around the floor such as LEGO Mindstorms and Sony AIBO dogs.
- *A Camera Array* is used to record experiments for supporting time-dependent location information (ground truth) of mobile robots. Figure 3 shows that the whole test-bed is monitored by four USB cameras attached at the ceiling. They can record videos at $20fps$ with resolution 320*240 pixels and track the location of mobile nodes within error distance $2.5cm$ after calibration.

Experimental Results

In this section, experiments are performed to evaluate the location estimation accuracy, calibration effort reduction and the robustness of `LeMan`. For comparison, we also run: (1) Regularized Least Square (RLS); (2) Support Vector Regression (SVR); (3) LANDMARC (Ni *et al.* 2003); (4) RADAR (Bahl & Padmanabhan 2000); (5) MoteTrack (Lorincz & Welsh 2006); (6) Basic Multilateration (Savvides, Han, & Strivastava 2001).

In applications, we can attach sensor nodes to any object. In our experiments, we control a mobile robot to run and stop around the test area (Figure 3) for collecting data with sampling interval $0.5s$. Two data sets are collected at different times within two days, each forming a trace of total length about $1,400$ $meters$ or $7,000$ examples. We spent nearly two months setting up cameras and exporting locations from videos. Labels were expensive to obtain. For every experiment below, we randomly picked up a subset of data from one data set for training and evaluate the performance on the other. To reduce statistical variability, results here are based on averages over 20 repetitions.

We use a Gaussian kernel $exp(-\|\mathbf{s}_{t_p} - \mathbf{s}_{t_q}\|^2/2\sigma^2)$ for Equations (3) and (4) since it is widely used in localization problems for adapting the noisy characteristic of radio signal (Nguyen, Jordan, & Sinopoli 2005) and σ is set to 0.5. The number of nearest neighbors k and the time interval ΔT for constructing graph Laplacian L are set to 5 and $0.5s$ respectively. For γ_A and γ_I in Equation (4), we refer to the handwritten digit and spoken letter recognition experiments in (Belkin, Niyogi, & Sindhwani 2005) and set $\gamma_A l = 0.005$ and $\frac{\gamma_I l}{(u+l)^2} = 0.045$.

Location Estimation Accuracy

In this section, experiments are done on the whole testing set ($7,000$ examples) when using another 100 labeled examples for all compared methods and additional 500 unlabeled examples for `LeMan` from the training set. Figure 4(a) plots the cumulative probability with respect to error distance. More detailed results are summarized in Table 1. As can be seen, `LeMan` has a better performance than the others. Note that RLS is a special case of `LeMan` when no unlabeled examples are used. Furthermore, `LeMan` has the smallest mean error distance and standard deviation. In the worst case, `LeMan` may predict the location within about 290cm. `LeMan`, with the help of unlabeled examples, improves the performance of tracking *mobile* nodes.

Figure 4(b) illustrates an estimated trajectory of about 30 $seconds$. The trajectory is not smooth since we have not yet applied a Bayes Filter (Step 4) in the online phase. However, we test that, by employing a filter, the performance of *all* compared methods can be enhanced by about 9% to 12%. What is more, the maximal error distance can be greatly re-

Table 1: Performance of Different Methods

Method	Mean (cm)	Std. (cm)	Max (cm)	Accuracy at 100cm	Time (ms)
LeMan	* 67	* 39	290	* 82%	0.242
RLS	78	46	358	73%	0.047
SVR	79	40	* 257	75%	0.045
RADAR	86	59	391	68%	0.106
MoteTrack	85	61	418	69%	0.106
Multilateration	108	77	1592	53%	0.125
LANDMARC	118	59	372	42%	0.085

duced because the filter smooths the trajectory. In practice, Bayes Filters are generally used. However, in this paper, we prefer to see the original performance of all compared methods. Thus, experimental results shown in this and the following sections are done without any Bayes Filter.

Figure 4(c) shows the sensitivity of error distance while varying parameters $\gamma_A l$ and $\gamma_I l/(u+l)^2$ in Equation (4). As can be seen, the result is stable when $\gamma = \gamma_A l + \gamma_I l/(u+l)^2$ ranges in $[10^{-1.5}, 10^{-0.5}]$. When the penalty that controls the function complexity in Equation (1) or (2) is higher, say , $\gamma > 10^{-0.5}$, the function mapping from signal to location becomes simpler and tends to underfit. On the other hand, if $\gamma < 10^{-1.5}$, the error distance goes up and overfits.

We test the average time for predicting a new position using the various methods on Matlab with a $3.2GHz$ CPU just for easy control of parameters and evaluation. However, our sensor data are collected on a realistic sensor network based on Crossbow MICA2. The result is shown in Table 1. `LeMan` is relatively slower than the others. However, it is still suitable in real time.

Calibration Effort Reduction

In the first experiment, we test the performance of `LeMan` in reducing the calibration effort. We set the number of labeled examples to 50, 100, 200 and 400 while varying the number of unlabeled ones from 0 to 1000 in each setting, which results are shown in Figure 4(d). For example, the error distance is $75cm$ when given 50 labeled and 250 unlabeled examples. Compared to $87cm$ if no unlabeled examples are available, the performance is enhanced by $(87 - 75)/87 = 14\%$ and is better than the result with 100 labeled examples only. `LeMan`, with the use of unlabeled examples, does help reduce calibration effort when the number of labeled examples is rare.

In the second experiment,we test how the number of labeled examples affects all compared methods. We fix the total number of examples $l + u = 500$ and vary the ratio of labeled part $l/(l + u)$ from 5% to 100%. Figure 4(e) shows that most of the methods benefit from the increasing number of labeled examples. `LeMan` again has a better performance than the others with the help of unlabeled examples. However, if more labeled examples are available, the unlabeled examples become less important.

Robustness to the Number of Beacon Nodes

Low density of nodes due to sparse deployment and node failure may affect the performance. In this experiment, we study how the number of *beacon* nodes affect the performance. We set the number of labeled and unlabeled examples to 100 and 500 respectively and randomly select a subset of beacon nodes for the experiment. The result is shown in Figure 4(f). As can be seen, `LeMan` is relatively more robust than the others.

Conclusion and Future Works

In this paper, we describe `LeMan`, a manifold regularization approach to reducing calibration effort for tracking *mobile* nodes in wireless sensor networks. Our model is based on the observation that similar signals from *beacon* nodes imply close locations. A mapping function between the signal space and the physical space is learned by using a small amount of labeled data and a large amount of unlabeled data. This function can then be used online to determine the location of *mobile* nodes. Experimental results show that we can achieve a higher accuracy with much less calibration effort. It is robust to changes in the number of *beacon* nodes, too. Furthermore, tracking multiple *mobile* nodes would not burden the network since each *mobile* node can estimate its own location by passively listening to *beacon* signals.

We are encouraged by the results and plan to extend this work in a few directions. First, we would like to move the experiment from the lab to a more complex environment to check the performance and robustness. Secondly, we may consider "distributed storage" of calibration data. Each *beacon* node stores some local calibration data nearby and broadcasts these data in their *beacon* frames so that we can reduce the storage cost in each *mobile* node. Third, if *beacon* nodes are densely deployed and their locations can be determined with any available hardware or algorithm (Shang *et al.* 2003; Nguyen, Jordan, & Sinopoli 2005), we can use their signal reading and location as the labeled examples. By combining them with unlabeled examples from *mobile* nodes, our algorithm can be fully automatic.

Acknowledge

We would like to thank Hong Kong RGC for supporting this work under grant HKUST6187/04E. We would also like to thank James Kwok, Lionel Ni, Yunhao Liu for discussions.

References

Bahl, P., and Padmanabhan, V. 2000. RADAR: An in-building RF-based user location and tracking system. In *Proceedings of the Conference on Computer Communications*, volume 2, 775–784.

Batalin, M. A.; Sukhatme, G. S.; and Hattig, M. 2004. Mobile robot navigation using a sensor network. In *IEEE International Conference on Robotics and Automation*, 636–642.

Belkin, M., and Niyogi, P. 2003. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation* 15(6):1373–1396.

Belkin, M.; Niyogi, P.; and Sindhwani, V. 2005. On manifold regularization. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 17–24. Society for Artificial Intelligence and Statistics.