# Maximum Likelihood Tracking of a Personal Dead-Reckoning System

Surat Kwanmuang[1] and Edwin Olson[2]

***Abstract*— We consider the problem of a human-following robot in which a human is equipped with a low-fidelity odometry sensor and a robot follows the human leader— often lagging well behind and out of visual contact with the human. The challenge is for the robot to determine the path taken by the human, despite the relatively noisy odometry data available. Such a system is useful in a "pack mule" application, where the robot carries a heavy load for the human.**

**Our key idea is to equip the robot with sensors allowing it to build a map, and to use observations of the environment structure to constrain the path of the human. We propose and evaluate several approaches: a particle filter method that extends monte-carlo localization approaches, and a multi-hypothesis maximum-likelihood approach based on stochastic gradient descent optimization that efficiently clusters similar trajectories. We demonstrate that our proposed approaches are able to track human trajectories in several synthetic and real-world datasets.**

## I. Introduction

The goal of this paper is to develop semi-autonomous robots capable of following a human leader. In a typical application like a "pack mule", the primary challenge is to track the position of the human leader. In this work, we assume that the human leader may be well outside sensor range but that the leader is equipped with a personal dead-reckoning (PDR) device [1] that transmits odometry-like measurements via radio. Supporting beyond line-of-sight tracking allows the robot much more flexibility in following the human. For example, if the human hops over a ditch that is untraversable by the robot, the robot may need to find an alternative path. With our beyond line-of-sight tracking approach, the human is free to continue: it does not need to wait for the robot. Other applications of a follower-robot include robotic luggage valets, robots for tracking first responders in a dangerous environment, and robots that provide tactical support to a squad of infantry whose rapid motions could make traditional tracking methods difficult.

This paper focuses on leaders equipped with personal dead-reckoning devices. These devices, while quite noisy, are not susceptible to GPS jamming and work in dark or foggy conditions that would be challenging for optical tracking methods.

The time between the moment the human leader starts walking and the moment the robot starts following can be anywhere from a few minutes to multiple hours. Once the leader starts walking, the PDR system onboard the leader transmits the leader's motion estimate to the robot. Once the robot begins, it reconstructs the leader's path by fusing the PDR-derived motion estimates with data obtained by the robot, including LIDAR and inertial measurements. The robot's observations of the environment are informative with respect to the trajectory of the human by virtue of the fact that the human cannot travel through obstacles. Thus, the map constructed by the robot constrains the set of possible trajectories taken by the human.

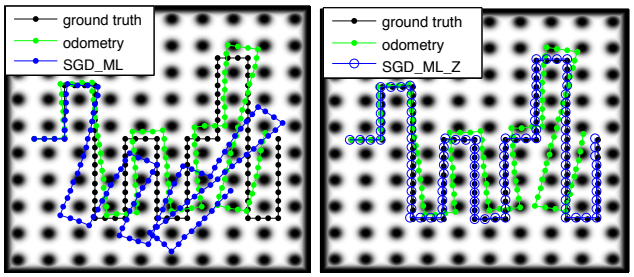


Fig. 1. Forest world results. This figure shows results for the algorithsm proposed in this paper. On the left, SGD_ML (blue) converges to the wrong solution when the odometry noise is large, whereas SGD_ML_Z (right) performs better and converges to the correct solution. Ground truth is plotted in black, and odometry is in red.

Our approach is based on a single-robot Simultaneous Localization and Mapping (SLAM) algorithm [2], but with the added complication that the goal is to estimate the trajectory of the human— recovery of the robot's trajectory is merely a means to this end. A significant challenge is that the robot's observations only indirectly constrain the trajectory of the human: trajectories that do not collide with obstacles are virtually indistinguishable. Recovering the leader's trajectory is thus difficult due to the many local minima and the complex non-linear relationship between plausible leader trajectories and robot observations. The central claim of this paper is that these difficulties can be overcome in many cases.

Specifically, the contributions of this paper include:

- Several methods for estimating the leader's trajectory: one based on a particle filter and two others based on maximum likelihood optimization using Stochastic Gradient Descent (SGD)
- An extension to the maximum-likelihood methods allowing for multiple hypotheses that has improved convergence in the presence of local minima
- An evaluation involving both real and simulated data that demonstrates the performance and tradeoffs of the algorithms

[1]Surat Kwanmuang is with the Department of Mechanical Engineering, Chulalongkorn University, Bangkok, 10330, Thailand. surat.k@chula.ac.th

[2]Edwin Olson is with the Computer Science and Engineering Department, University of Michigan, Ann Arbor, MI 48109, USA. ebolson@umich.edu ; http://april.eecs.umich.edu

## II. PRIOR WORK

Current studies on leader-follower problems use GPS as the primary method for the robot to follow the human [3]. By comparing the robot's current GPS coordinates with the human's coordinates, the robot can follow the leader without line-of-sight requirements. This approach, however, relies on the availability of a GPS signals which are susceptible to obstructions (including the "urban canyon" effect), are generally too attenuated indoors to be used, and can be easily jammed.

Another prevalent method is to use stereo vision on both the leader and the robot [4]. In this scenario the leader sends image features from their camera to the robot to store in its database. The robot compares its current image features with the database to estimate the relative position of the human then tries to move to that position. This method is sensitive to changes in appearance in the environment which can occur if the robot is following at a great distance or if the robot cannot follow the same path as the leader. It is also bandwidth intensive.

The leader-follower problem can also be treated as a multiple robot localization problem [5] in which both the leader and the follower use their sensors to estimate the location of the other. In this case, however, because the leader is a person rather than a robot, mounting sensors such as a laser scanner is not very practical.

Other researchers have used inertial measurement sensors on human leaders [6], [7] and particle filters exploiting a prior map. In this work, we eliminate the need for a prior map.

## III. BACKGROUND

### A. Personal dead-reckoning system

The Personal Dead-Reckoning system (PDR) [1] was developed to track the position of a human subject in real-time. The system consists of an inertial measurement unit (IMU) strapped on the side or embedded under the heel of the subject's boot. When the subject is walking, the PDR system estimates current position and orientation of the system as well as the uncertainty of the estimation. Yaw estimates are derived from low-cost gyroscopes while translations can be recovered by double-integrating acceleration measurements. This double integration would ordinarily result in extremely noisy estimates, however the PDR system exploits the fact that the heel is momentarily stationary during each step. This allows for continuous recalibration of the accelerometer.

### B. Relative poses

The poses from the PDR system are formed by composing the changes measured by inertial measurement sensors. Thus, if we modify one pose, all following poses along the chain are projectively affected.

We can define state variables as relative poses $y_i$ between human poses $x^h_{i-1}$ and $x^h_i$. This variable is initialized to be a function of odometry input $y^0 = f(o)$:

$$y_i = [\tilde{x}_i, \tilde{y}_i, \tilde{\theta}_i,]^T \tag{1}$$

Therefore, a transformation $T(y_i)$ which transforms a pose $x^h_{i-1}$ to $x^h_i$ is given as:

$$x^h_i = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}^h_i = T(y_i) \otimes x^h_{i-1} \tag{2}$$

$$x^h_i = \begin{bmatrix} \cos(\theta_{i-1})\tilde{x}_i - \sin(\theta_{i-1})\tilde{y}_i + x_{i-1} \\ \sin(\theta_{i-1})\tilde{x}_i + \cos(\theta_{i-1})\tilde{y}_i + y_{i-1} \\ \theta_{i-1} + \tilde{\theta}_i \end{bmatrix} \tag{3}$$

Thus, the pose at time $t$ can be calculated from a function:

$$x^h_t = T(y_t) \otimes T(y_{t-1}) \otimes .... \otimes T(y_1) \otimes x^h_0 \tag{4}$$

Given all relative poses $y_{0:n}$, the whole trajectory $x^h_{0:n}$ can also be obtained in the same way $x^h_{0:n} = \{x^h_0, x^h_1, ...., x^h_n\}$.

## IV. METHOD

In this section, we propose several methods for tracking a human leader. First we describe our approach for determining the likelihood of a candidate leader trajectory by proposing the use of "human presence probability" function. We then propose a particle filter algorithm for tracking which, while itself a contribution of this paper, serves as a baseline method for our other proposed methods.

Next, we propose an approach called "Stochastic Gradient Descent for Maximum Likelihood tracking" (SGD_ML) and its variant (SGD_ML_Z). One limitation of SGD_ML and SGD_ML_Z is that they are only able to optimize poses to a single local minima and thus are unable to consider multiple trajectory hypotheses. We then show how these methods can be extended to track multiple hypotheses, which greatly improves the quality of the trajectory estimates.

### A. Human presence probability

The key idea in this paper is that a map of the environment can be used to evaluate the likelihood of candidate leader trajectories. For example, one could imagine that every grid cell in the map is labeled with a "weight" such that the probability of any given trajectory is approximately equal to the product of the weights of the cells through which the trajectory passes. Cells marked as obstacles would receive a weight of zero, since it is impossible for a human to pass through a wall or other obstruction. In principle, cells along "good" paths (sidewalks, perhaps) might have relatively high weights, whereas difficult terrain might be assigned a small positive weight.

We call such a map of "weights" a "human presence probability map". Unfortunately, obtaining such a map is difficult. For example, it could require collecting large amounts of empirical data, or developing autonomous classification systems for sidewalks and other terrain types. Worse, the behavior of humans can vary dramatically: while most humans would likely maintain a comfortable distance between themselves and obstacles, a squad of infantrymen might deliberately cling to any surface that provides cover.

Despite these challenges, we hypothesized that even a simple means of computing a human presence probability map would work in many cases. In particular, we propose

computing weights as a function of the distance to the nearest obstacle:

$$p(x|m_{ML}) = \begin{cases} 0 & d < s \\ \exp(-(d-l)^2/2\sigma^2) & s < d < l \\ 1 & d > l \end{cases} \quad (5)$$

where $d$ is the distance to closest obstacle. We can precompute this probability for each pose inside the map for a given occupancy map using a distance transform and equation (5); see Fig. 2. We use $s = 0.25$ m, $l = 0.6$ m and $\sigma = 0.1$ m, which leads to a function similar to that from an empirical study of human behavior [8].

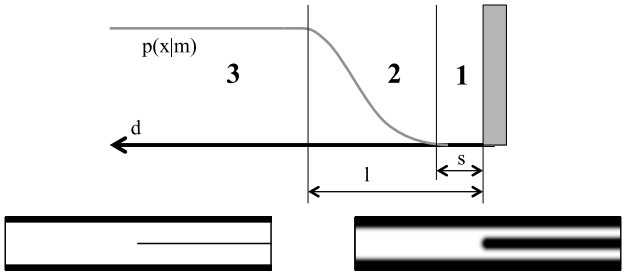


Fig. 2. Human presence probability maps. Top: the human presence probability, $p(x|m)$ as a function of distance to the closest obstacle, $d$. The shaded area to the right is the obstacle. The curve shows the probability of human presence at each distance from the closest obstacles. Bottom: An occupancy map from a simulated corridor with a divider in the middle (left) and the resulting human presence probability map (right).

### B. Graphical model of maximum likelihood tracking

Our goal is to estimate the trajectory of the human given both odometry data from the human and measurement data acquired by the robot. That is, we want $p(x^h|u,z,o)$ where $x^h$ is the human trajectory, $u$ is the robot's odometry input, $z$ is robot laser scanning measurement of an environment $m$ and $o$ is PDR odometry input.

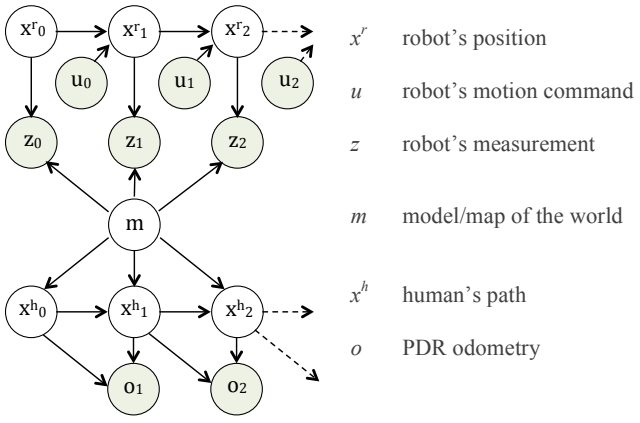


Fig. 3. A probabilistic graphical model of human/robot trajectory. The top part of the graph shows the robot trajectory and its measurements of the map. The bottom part are a path of the human and the PDR odometry measurements of the path. The shaded variables are observed.

The posterior of the human trajectory $p(x^h|u,z,o)$ can be written as:

$$p(x^h|u,z,o) = \int p(x^h,m|u,z,o)dm \quad (6)$$

$$\text{(Bayes' rule)} \propto \int p(u,z,o|x^h,m)p(x^h,m)dm \quad (7)$$

$$\text{(Product rule)} = \int p(o|x^h,m,z,u)p(z,u|x^h,m)p(x^h,m)dm \quad (8)$$

We can apply conditional independence rule to the term $p(o|x^h,m,z,u)$ and Bayes' rule on the $p(z,u|x^h,m)$ term.

$$p(x^h|u,z,o) \propto \int p(o|x^h)p(x^h,m|z,u)dm \quad (9)$$

$$\text{(Product rule)} = \int p(o|x^h)p(x^h|m,z,u)p(m|z,u)dm \quad (10)$$

$$\text{(Cond ind.)} = \int p(o|x^h)p(x^h|m)p(m|z,u)dm \quad (11)$$

As described in section IV-A, the human presence probability map $m_{ML}$ is obtained from the map posterior $p(m|z,u)$ generated from the robot.

$$p(x^h|u,z,o) \approx p(o|x^h)p(x^h|m_{ML}) \quad (12)$$

From the graphical model assumptions in Fig. 3, the posterior $p(x^h|u,z,o)$ can be written as:

$$p(x^h|u,z,o) = \eta \prod p(o_i|x^h_{i-1},x^h_i) \prod p(x^h_i|m_{ML}) \quad (13)$$

where $\eta$ denotes a normalizer.

### C. Particle filter tracking

We first implemented particle filter tracking (Monte Carlo localization) [9]. MCL has been used widely in localization problems due to the effortlessness of implementation and it works well in many situations. MCL can also handle multimodal posteriors.

From Equation (13), using Bayes's rule:

$$p(x^h|u,z,o) \propto \prod p(x^h_i|x^h_{i-1},o_i) \prod p(x^h_i|m_{ML}) \quad (14)$$

We can use the particle filter and sample from a distribution $p(x^h_i|x^h_{i-1},o_i)$. The weight of each particle becomes:

$$w_i = p(x^h_i|m_{ML}) \cdot w_{i-1} \quad (15)$$

Although MCL works well in many scenarios, it can suffer from particle depletion, a situation in which there are not enough particles to accurately represent the distribution. This particle depletion problem is depicted in Fig. 4. Increasing the number of particles can delay the onset of particle depletion, but with increasing computation cost.

## V. Maximum likelihood tracking

To avoid particle depletion in the MCL algorithm, we propose a new tracking algorithm using maximum-likelihood estimation. Since maximum-likelihood tracking only keeps track of one trajectory, it is very memory efficient compared to a particle filter that has to keep track of all the particles.

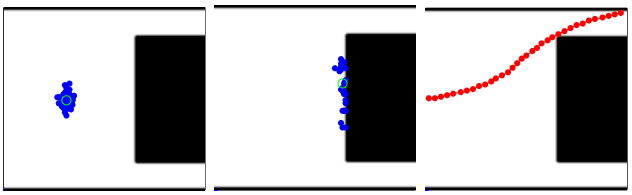

Fig. 4. Particle depletion problem of monte-carlo localization. The human walked from left to right. Blue dots are particles representing human poses. (Right) Particles are all depleted while our maximum-likelihood algorithm (SGD_ML in red) is still able to track the human trajectory.

The closed-form solution of the posterior through induction is given as:

$$p(x^h|u,z,o) = \eta\, p(x_0^h) p(x_0^h|m_{ML}) \prod_{i=1}^{n} [p(x_i^h|m_{ML}) p(o_i|x_{i-1}^h, x_i^h)] \tag{16}$$

A cost function or a negative log likelihood of the posterior can be calculated by taking the logarithm of the equation (16):

$$-\log(p(x^h|u,z,o)) = const - \log(p(x_0^h)) - \log(p(x_0^h|m_{ML})) - \sum_{i=1}^{n} [\log(p(x_i^h|m_{ML})) + \log(p(o_i|x_i,x_i))] \tag{17}$$

The human initial position $x_0^h$ is assumed to be a Gaussian distribution $N(0,\Sigma_0)$. Similarly, the PDR odometry probability $p(o_i|x_{i-1},x_i) = p(o_i|y_i)$ is assumed to be Gaussian $N(y_i^0,\Sigma_i)$ where $y^0 = f(o)$.

$$p(u_i|x_{i-1},x_i) = \eta \exp\left(-\frac{1}{2}(y_i - y_i^0)^T \Sigma_i^{-1} (y_i - y_i^0)\right) \tag{18}$$

By inspection, the initial pose $p(x_0^h)$ is a special case of the above equation, where $y_0 = x_0^h$ and $y_0^0 = 0$:

$$p(x_0) = \eta \exp\left(-\frac{1}{2}(y_0 - 0)^T \Sigma_0^{-1} (y_0 - 0)\right) \tag{19}$$

We now define $d_i \equiv y_i - y_i^0$. The cost of the PDR odometry update becomes:

$$\sum_{i=0}^{n} \log(p(o_i|x_{i-1}^h, x_i^h)) = -\frac{1}{2} \sum_{i=0}^{n} (y_i - y_i^0)^T \Sigma_i^{-1} (y_i - y_i^0) \tag{20}$$

$$= -\frac{1}{2} d^T \Sigma^{-1} d \tag{21}$$

where $d = [d_0, d_1, ..., d_n]$

As for $\log(p(x_i^h|m_{ML}))$, the probability of the poses given a map can be easily looked up in the pre-generated human presence probability map ($map_{ML}$), where $x_i^h$ is inside that cell.

$$p(x_i^h|m_{ML}) \approx h(x_i^h) = map_{ML}(i,j) \qquad x_i^h \in \text{cell}(i^{th}, j^{th}) \tag{22}$$

$$\log(p(x_i^h|m_{ML})) = \log(h(x_i^h)) \tag{23}$$

As a result, the negative log-likelihood of the posterior and the cost function becomes:

$$J = -\log(p(x^h|u,z,o)) \tag{24}$$

$$J = \sum_{i=0}^{n} \left(-\log(h(x_i^h))\right) + \frac{1}{2} d^T \Sigma^{-1} d + const \tag{25}$$

Intuitively, we can see that the cost function is a function of the deviation from the odometry $d^T\Sigma^{-1}d$ and the probability of the human can be presence at a given pose $\log(h(x_i^h))$. If the pose deviates too far from the odometry measurement, the cost will increase as a function of the information matrix $\Sigma^{-1}$. A pose that has high certainty (small $\Sigma$), will incur a larger cost than the uncertain one.

If the human pose $x_i^h$ is close to obstacles and unlikely to be present there, $p(x_i^h|m_{ML})$ approaches zero, and a substantial cost will be incurred compared to poses further away from obstacles.

### A. Stochastic gradient descent (SGD)

To find the poses that have lowest cost function, SGD can be used as an optimization solver. SGD optimizes each pose at a time to reach the cost minima. We use SGD due to its robustness to local minima.

From Equation (25), the cost function for a single pose is:

$$J_i = -\log(h(x_i^h)) + \frac{1}{2(n+1)} d^T \Sigma^{-1} d \tag{26}$$

Using the chain rule, the gradient of the cost function for a single pose is:

$$\nabla J_i = -\frac{1}{h(x_i^h)} \cdot \frac{\partial h(x_i^h)}{\partial x_i^h} \cdot \frac{\partial x_i^h}{\partial y} \cdot \frac{\partial y}{\partial d} + \frac{1}{n+1} d^T \Sigma^{-1} \tag{27}$$

The first term, $h(x_i^h)$ can be easily looked-up from the human presence probability map as in equation (22). Next, $\partial h(x_i^h)/\partial x_i^h$ is the gradient of the map around $x_i^h$. The term $\partial x_i^h/\partial y$ can be calculated from the Jacobian matrix.

$$\frac{\partial x_i^h}{\partial y} = \begin{bmatrix} \frac{\partial x_i^h}{\partial y_0} & \frac{\partial x_i^h}{\partial y_1} & \dots & \frac{\partial x_i^h}{\partial y_n} \end{bmatrix}_{2\times 3n} \tag{28}$$

Lastly, since we define $d \equiv y - y^0$, the term $\partial y/\partial d = 1$.

We can define the learning rate $\lambda$ which will dictate the step size of the state correction. In addition, the state estimate moves in the opposite direction of the gradient (27). As a result, the state correction becomes:

$$d = \lambda \left( \frac{1}{h(x_i^h)} \cdot \frac{\partial h(x_i^h)}{\partial x_i^h} \cdot \frac{\partial x_i^h}{\partial y} - \frac{1}{n+1} d^T \Sigma^{-1} \right) \tag{29}$$

We use a learning rate that is harmonically-decreasing $\lambda = 1/t$, as in [10]. We found that if the initial learning rate $\lambda_0$ is too large, the state will move in large steps and is likely to jump around the solution. The worst case scenario is that it may jump and be trapped outside the convex area of the cost function, and thus fail to converge. On the other hand, smaller $\lambda_0$ will slow the convergence down and the solution may not converge in a reasonable time.

In summary, the overview of the SGD_ML method is presented in the Algorithm 1.

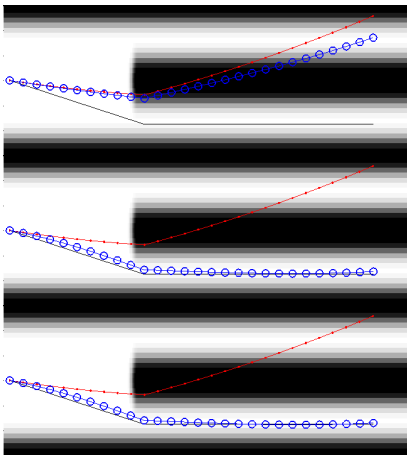

Fig. 5. A simulation result of the single hypothesis SGD_ML algorithm. The blue line is the maximum likelihood trajectory solution at iteration 1 (top), 2 (middle) and 10 (bottom). The red path is the odometry input from the PDR system. The black line is the ground truth. The solution quickly converges close to convergent in iteration 2. ($\lambda_0 = 1e-5$)

**Algorithm 1** SGD_ML

$y \leftarrow y^0$ {$y$ is an initial set of odometry measurements}
$\lambda \leftarrow \lambda_0$ {Initialize learning rate $\lambda_0$}
**while** $y$ not converged **do**
  select pose $i$ at random
  **for** each pose $i$ **do**
    $x \leftarrow T(y_i) \otimes .... \otimes T(y_1) \otimes x_0$
    $h \leftarrow map_{ML}(x)$ {defined in eq.(22)}
    $g \leftarrow$ gradient around $x$
    $J \leftarrow$ Jacobian $\partial x_i^h / \partial y$ {eq. (28)}
    $d \leftarrow y - y^0$
    $\Delta y \leftarrow \lambda \left( h^{-1} \cdot g \cdot J - (n+1)^{-1} d^T \Sigma^{-1} \right)$ {eq. (29)}
    $y \leftarrow y + \Delta y$
  **end for**
  $\lambda \leftarrow \lambda / (\lambda + 1)$
**end while**
**return** $y$

One limitation of both particle filter and SGD_ML is that, without any information from the map, the optimization result will be identical to the odometry path. This is because we do not have any additional information other than the odometry.

### B. Zippering SGD_ML (SGD_ML_Z)

SGD_ML optimizes the whole trajectory at once. For longer trajectories, this leads to a large search space for the optimizer and may contain several local minima. This problem can be seen in Fig. 1 when the odometry noise is large, and SGD_ML converges to the wrong solution.

We implement a "zippering" method to further improve robustness to local minima. We start optimizing from the first section of the trajectory and then incrementally move the window forward like a zipper. By beginning optimization in the best-known part of the trajectory, the optimization is more likely to converge to the correct local minima. The process then repeats and moves to the next section. We show that this method is more likely to converge to the correct solution.

As a result, the SGD_ML_Z method improves the robustness of the SGD_ML. A more detailed discussion can be found in the result section.

### C. Multiple trajectory hypothesis tracking

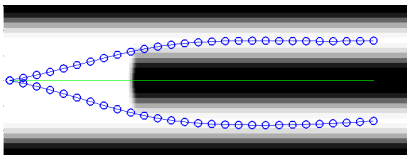

Fig. 6. The simulated odometry input (green) leads to ambiguous solutions. It leads to right in the middle of two corridors. Both top and bottom trajectories converge into different local minima in the cost function.

In many situations, the odometry input from the PDR system can lead to ambiguous trajectories. For example,(considering the two identical corridors in Fig. 6) if the PDR data is uncertain, the human could have taken either path.

For this reason, the robot needs to track multiple hypotheses so it will not commit to the wrong solution when there is a lot of ambiguity. To do this, we will repeat the SGD_ML with different odometry estimates consistent with our odometry noise model, thus exploring other local minima. Specifically, The SGD_ML algorithm is extended by adding two processes, described below.

*1) Perturbation:* The initial condition of the state variable $y$ which is normally equal to odometry input $y^0$ is perturbed to allow the maximum-likelihood solution to converge to different local minima.

We repeatedly sample $y$ from the distribution $N(y^0, \Sigma)$ and perform optimization using SGD_ML. Furthermore, we perform this perturbation when the cost of the current trajectory hypotheses are larger than a threshold indicating poor likelihood. By sampling from the distribution $N(y^0, \Sigma)$, we can make sure that our solutions are unbiased.

*2) Merging Trajectories:* We check that the cost function $J(y)$ between two trajectories ($y_1$,$y_2$) is convex or not by using Eq.(30). If it is convex (indicating there is a local minimum of the cost function between the two trajectories), they are combined and and kept as a new hypothesis.

$$J(t \cdot y_1 + (1-t) \cdot y_2) \leqslant t \cdot J(y_1) + (1-t) \cdot J(y_2); \forall t \in [0,1] \tag{30}$$

First, the combination of two trajectories is simply the average of both. Finally, the combination is once again optimized by SGD_ML to find the maximum-likelihood solution of the average.

By merging trajectories that are close together, we reduce the number of trajectory hypotheses, thus, reducing the computational complexity. The fine-scale detail of each merged trajectory is lost; however, the route that the leader took is much more interesting for the robot than the fine detail of the paths.

The method is presented in Algorithm 2. Fig. 9 shows the maximum-likelihood solution after both perturbation and merging.

**Algorithm 2** Multiple trajectory hypothesis SGD_ML

$n \leftarrow$ number of samples
$s = \{\}$ {$s$ is the sample set}
**for** loop $n$ times **do**
  sample $y$ from $N(y^0, \Sigma)$
  $y_{ml} \leftarrow SGD\_ML(y)$
  append $y_{ml}$ in $s$
**end for**
$s_{merged} \leftarrow merge(s)$
$s_{refined} = \{\}$
**for** each $y$ in $s_{merged}$ **do**
  $y_{ml} \leftarrow SGD\_ML(y)$
  append $y_{ml}$ in $s_{refined}$
**end for**
**return** $s_{refined}$

## VI. RESULTS

We evaluate the performance of both SGD_ML and SGD_ML_Z algorithms by testing them against particle filter tracking as a baseline. First, we repeatedly simulate them in a synthetically generated environment.

### A. Forest world

The forest world is designed to have equally-spaced obstacles ("trees") spread over the map. This obstacle placement has exponentially many possible paths between any two points on the map. The map is also symmetrical in both directions and ambiguous for the tracker. The routes between starting and ending locations are randomly chosen. Furthermore, the simulated odometry measurement is corrupted with noise and fed to all three algorithms.

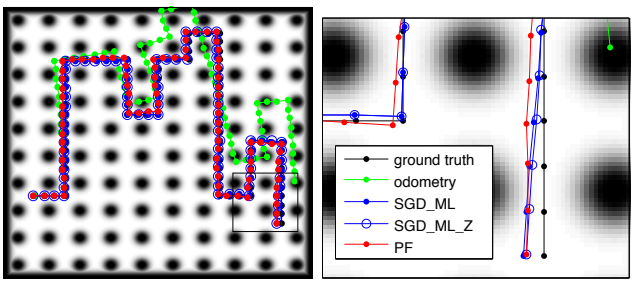


Fig. 7. (Left) A result from forest world experiment: the obstacles are evenly placed to allow exponentially possible paths between randomly selected start and end locations. The simulated odometry input (green) is corrupted with noise. The black line is the ground truth. The blue and red lines are SGD_ML and particle filter output, respectively. (Right) The zoom-in version of the rectangular area in left image.

One particular result is shown in Fig. 7, in which all three algorithms are comparable. At the same time, however, the SGD_ML algorithm converges to the wrong solution in 25 of 100 runs. An example of this problem is in Fig. 1. Since the SGD_ML optimizes the whole trajectory at once, this also means that the optimizer tries to optimize a larger search space which may contains several local-minima. As a result, it is possible that the optimization may converge to a different local-minima. Meanwhile, particle filter and SGD_ML_Z do not suffer this issue and converge to the correct solutions in all runs.

Table I shows the root-mean-square error (RMSE) of the solutions for each algorithm. The RMSE is computed by averaging RMS error of all poses in the trajectory. The SGD_ML algorithm has smaller errors than the particle filter baseline considering only those occasions on which it performs correctly. Compared to the particle filter, it has 35% and 56% improvement in positioning and heading, respectively. Lastly, the SGD_ML_Z performs better than SGD_ML (43% in positioning and 65% in heading) and robustly converges to the correct solutions in all runs. Both improvements over the particle filter are even larger in larger odometry noise simulation.

On average, our SGD_ML algorithm implemented in MATLAB uses 5.6 times the CPU time of the particle filter baseline. Meanwhile, SGL_ML_Z is 4 times slower than SGD_ML since it optimizes the trajectory repeatedly.

TABLE I
ROOT-MEAN-SQUARE ERROR (RMSE) COMPARISON BETWEEN SGD_ML, SGD_ML_Z AND PARTICLE FILTER (PF) FROM 100 WORLD OF DOTS SIMULATION RUNS.

| Method | Heading odometry noise $\sigma$=0.02 radians | | | |
|---|---|---|---|---|
| | Position RMS error (m) | | Angular RMS error (radian) | |
| | Average | Maximum | Average | Maximum |
| PF | 0.2146 | 0.3436 | 0.0532 | 0.0926 |
| SGD_ML* | 0.1585 | 0.2869 | 0.0340 | 0.0768 |
| SGD_ML_Z | 0.1494 | 0.2657 | 0.0322 | 0.0725 |

*Note: SGD_ML only converges correctly in 75 of 100 runs.

| Method | Heading odometry noise $\sigma$=0.05 radians | | | |
|---|---|---|---|---|
| | Position RMS error (m) | | Angular RMS error (radian) | |
| | Average | Maximum | Average | Maximum |
| PF | 1.0743 | 2.5345 | 0.1472 | 0.3053 |
| SGD_ML* | 0.2470 | 0.3813 | 0.0750 | 0.1348 |
| SGD_ML_Z | 0.2312 | 0.3787 | 0.0669 | 0.1382 |

*Note: SGD_ML only converges correctly in 22 of 100 runs.

### B. BBB building

We also test the SGD_ML_Z algorithm with real odometry data from the PDR system. The subject walked inside a typical office building. Due to the non-deterministic nature of these algorithms, we repeatedly process each experiment 5 times for better understanding of the performance variation.

TABLE II
AVERAGE POSITION/ANGULAR ERROR OF BBB BUILDING EXPERIMENTS

| Run | Avg. position error (m) (95% CI) | | Avg. angular error (radian) (95% CI) | |
|---|---|---|---|---|
| | PF | SGD_ML_Z | PF | SGD_ML_Z |
| 1 | 2.250±0.637 | 1.233±0.560 | 0.277±0.032 | 0.070±0.015 |
| 2 | 2.412±0.653 | 1.026±0.352 | 0.267±0.022 | 0.188±0.010 |
| 3 | 1.278±0.213 | 0.623±0.298 | 0.365±0.025 | 0.014±0.006 |
| 4 | 1.241±0.311 | 0.907±0.181 | 0.365±0.073 | 0.205±0.005 |
| Avg. | 1.795±0.610 | 0.947±0.250 | 0.319±0.053 | 0.119±0.009 |

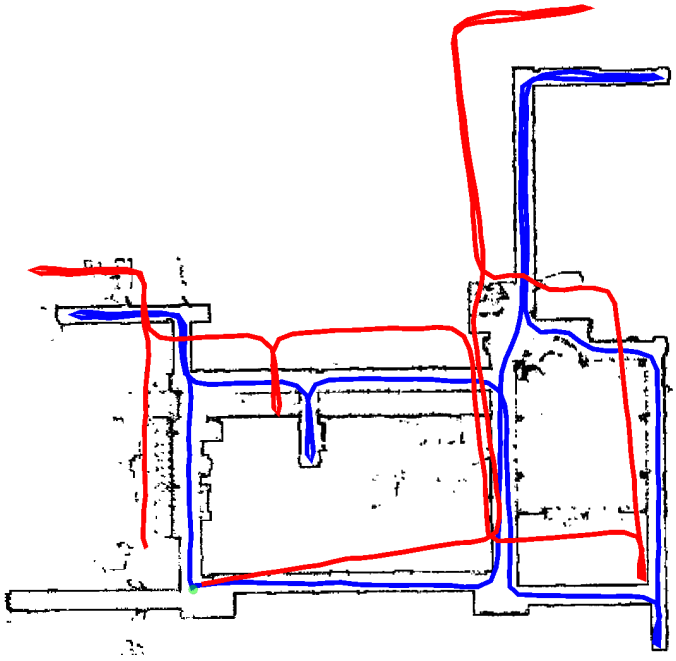

Fig. 8. An experiment of a subject walked inside a building (Run#3 in the Table II). The map was generated using SLAM from a robot driven around the building. The odometry input recorded from the PDR system is in red and corrupted with noise. The output from SGD_ML_Z algorithm (blue) shows an improvement in position estimation.

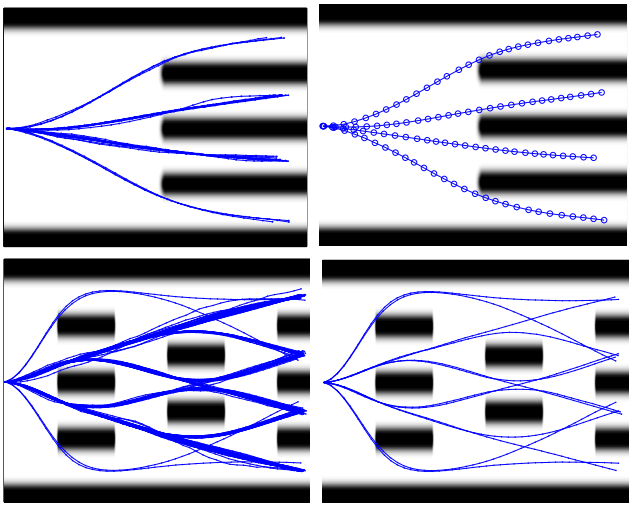

Fig. 9. (Left) Maximum likelihood hypotheses with perturbed initial condition ($n = 300$). (Right) Trajectories after merging. Reducing to 4 and 12 distinct maximum likelihood hypotheses.

### C. Multiple trajectory hypotheses

The multiple trajectory hypotheses version of SGD_ML was also tested in a synthetic map. The map consists of multiple gates that are designed to be ambiguous. The algorithm performs correctly in sampling and merging trajectories to achieve distinct maximum-likelihood solutions. Results from the test can be seen in Fig. 9.

## VII. CONCLUSIONS

In this paper, we showed how map data collected by a robot can be used to improve the trajectory estimates of a human leader equipped with a Personal Dead-Reckoning (PDR) system. We illustrated our approach with several algorithms. We showed that the particle filter approach has the largest margin of error and suffers from particle depletion. Our approach based on maximum likelihood optimization using stochastic gradient descent, SGD_ML, performs well in low noise settings, but is sensitive to large amounts of odometry noise. The SGD_ML_Z significantly improves robustness by exploiting an intuition based on "zippering"— imposing hard constraints incrementally from the beginning of a trajectory, allowing collisions to temporarily occur in later portions of the trajectory.

We showed that additional robustness could be achieved by extending the maximum-likelihood methods to track multiple hypotheses. Like a particle filter, this allows the algorithms to avoid pruning possibilities when the data is ambiguous. However, unlike a particle filter and traditional multi-hypothesis tracking systems, these hypotheses do not represent a *posterior distribution*, but rather a population of maximum likelihood hedges.

## REFERENCES


[1] J. Borenstein, L. Ojeda, and S. Kwanmuang, "Heuristic reduction of gyro drift for personnel tracking systems," *Journal of Navigation*, vol. 62, no. 1, pp. 41–58, 2009.

[2] E. Olson, J. Leonard, and S. Teller, "Fast iterative alignment of pose graphs with poor initial estimates," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2006, no. May. Orlando, FL: IEEE, 2006, pp. 2262–2269.

[3] N. Teck Chew, J. Ibanez-Guzman, S. Jian, G. Zhiming, W. Han, and C. Chen, "Vehicle following with obstacle avoidance capabilities in natural environments," in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, vol. 5. New Orleans, LA: IEEE, 2004, pp. 4283–4288.

[4] O. Naroditsky, Z. Zhu, A. Das, S. Samarasekera, T. Oskiper, and R. Kumar, "Videotrek: A vision system for a tag-along robot," in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2009*. Miami Beach, FL: IEEE, 2009, pp. 1101–1108.

[5] D. Fox, W. Burgard, H. Kruppa, and S. Thrun, "A probabilistic approach to collaborative multi-robot localization," *Autonomous Robots*, vol. 8, no. 3, pp. 325–344, 2000.

[6] O. Woodman, "Pedestrian localisation for indoor environments," Ph.D. dissertation, University of Cambridge, 2010.

[7] S. Beauregard, "Infrastructureless Pedestrian Positioning," Ph.D. dissertation, University of Bremen, 2009.

[8] B. R. Fajen and W. H. Warren, "Behavioral dynamics of steering, obstacle avoidance, and route selection," *Journal of experimental psychology. Human perception and performance*, vol. 29, no. 2, pp. 343–362, 2003.

[9] D. Fox, S. Thrun, W. Burgard, and F. Dellaert, "Particle filters for mobile robot localization," in *Sequential Monte Carlo methods in practice*. Citeseer, 2001, pp. 401–428.

[10] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.