Range-only SLAM with Interpolated Range Data

Ath. Kehagias, J. Djugash, S. Singh

CMU-RI-TR-06-26

May 2006

Robotics Institute

Carnegie Mellon University

Pittsburgh, Pennsylvania 15213

Range-only SLAM with Interpolated Range Data

Ath. Kehagias, J. Djugash, S. Singh

June 13, 2006

Abstract

In a series of recent papers Singh et al. have explored the idea of Simultaneous Localization and Mapping (SLAM) using range-only measurements. These measurements, obtained from radio or sonar sensors, come at *irregular time intervals*. In this report we explore the use of interpolation to generate data equally spaced in time, in order to improve the performance of SLAM algorithms. We test this idea on several (simulated and real) robot paths and two SLAM algorithms: an online Extended Kalman Filter (EKF) algorithm and an offline batch optimization algorithm.

1 Introduction

A robot can *localize* itself (determine its spatial coordinates) using a map of landmarks and an onboard sensor which senses these landmarks; conversely, from accurate localization of the robot / sensor, it is possible to build a *map* of the landmarks (determine their spatial coordinates). Performing both tasks simultaneously constitutes the *Simultaneous Localization and Mapping*[1] (SLAM) problem; when the aforementioned sensor can only detect *range* to landmarks we have the special (and important) case of *range-only SLAM*. In this report we evaluate a particular approach to performing range-only SLAM.

For a more extended description of SLAM problems in general see [1, 5, 6] and especially [12]; for range-only SLAM in particular see [10, 11, 13, 14] and [9] which includes references to many related publications; let us mention here that range-only SLAM has been studied in a recent series of papers [2, 3, 7, 8, 16] by Singh and his collaborators; finally there is a connection between range-only SLAM and localization in wireless sensor networks [4, 15].

The basic ideas of the current report can be described as follows.

1. Range measurements are collected by the onboard sensors at *irregular* time intervals;
2. we want to generate data approximately equally spaced in time by interpolation;
3. using such data will (hopefully) improve the performance of SLAM algorithms – more specifically, we hope that we can use interpolation to produce the "right" amount of data uniformly over the time interval of interest, since we want neither "too few" data (with too little information the SLAM algorithms will not work properly) nor "too many" data (this can cause computational difficulties);
4. we also want to apply the same idea to odometry measurements;
5. our *goal* is to test whether this interpolation-based approach improves the performance of SLAM algorithms;
6. we test *two* algorithms, an online EKF algorithm reported in [3] and an offline "classical" batch optimization algorithm.

[1] We emphasize that "mapping"means the determinatiuon of the spatial coordinates of landmarks, beacons etc.

This report is organized as follows. In Section 2 we give a brief description of the two SLAM algorithms we have used; in Section 3 we describe the interpolation method; in Section 4 we present various experiments (using both synthetic and real data) to evaluate the usefulness of interpolation.

2 SLAM Algorithms

In this section we describe the two SLAM algorithms we have used, namely SLAM by *Extended Kalman Filtering* (EKF) and SLAM by *batch optimization*.

2.1 SLAM by Extended Kalman Filtering

Assuming that a wheeled robot moves on a plane, the robot state at time t (we assume time evolves in discrete units $t = 0, 1, 2, ...$) can be represented by the state vector $q_t = [x_t, y_t, \theta_t]^T$, where (x_t, y_t) are the Cartesian plane coordinates of the robot position and θ_t is the robot orientation. Let the distance travelled between time t and $t+1$ be denoted by ΔD_t and the change in heading by $\Delta\theta_t$; then the input vector is $u_t = [\Delta D_t, \Delta\theta_t]^T$. The motion of the robot is described by the following set of nonlinear equations:

$$\begin{bmatrix} x_{t+1} \\ y_{t+1} \\ \theta_{t+1} \end{bmatrix} = \begin{bmatrix} x_t + \Delta D_t \cos(\theta_t) \\ y_t + \Delta D_t \sin(\theta_t) \\ \theta_t + \Delta\theta_t \end{bmatrix} + \nu_t, \tag{1}$$

where ν_t is a noise vector. Range measurement is effected by a network of sensors which uses radio to communicate data and ultrasonic measurements to range in. Assume that N stationary *beacons* (e.g. sonar/radio beacons, such as the *Parrots* [3]) are available and the robot carries an additional sensor, which can range to the beacons. Hence we have the following range *measurement* equation

$$\hat{r}_t = \sqrt{(x_t - X_{b(t)})^2 + (y_t - Y_{b(t)})^2} + \omega_t \tag{2}$$

where $b(t) \in \{1, 2, ..., N\}$ is the id of the beacon to which range is measured at time t, $\left(X_{b(t)}, Y_{b(t)}\right)$ are the coordinates of said beacon and ω_t is a noise vector.

Localization can be achieved, in the above formulation, by estimation of the robot state, which can be performed by extended Kalman filtering ("extended" because eqs. (1), (2) are nonlinear and hence not amenable to the "standard" linear Kalman filter). The approach used to implement EKF is rather standard (a detailed description appears in [9]).

SLAM can be performed by EKF as well, if we augment the state vector as follows:

$$q_t = \left[\; x_t, \quad y_t, \quad \theta_t, \quad X_{1,t}, \quad Y_{1,t}, \quad \ldots \quad X_{N,t}, \quad Y_{N,t} \; \right]^T \tag{3}$$

where $(X_{n,t}, Y_{n,t})$ is the position of the n-th beacon at time t. Since the beacons are immobile, the state equations are augmented by

$$\begin{array}{l} X_{n,t+1} = X_{n,t}, \\ Y_{n,t+1} = Y_{n,t}, \end{array} \qquad n = 1, 2, ..., N. \tag{4}$$

The measurement equation requires a minimal modification:

$$\hat{r}_t = \sqrt{(x_t - X_{b(t),t})^2 + (y_t - Y_{b(t),t})^2} + \omega_t. \tag{5}$$

In short, our state / measurement model is given by eqs.(1), (4), (5). EKF can be applied to these equations, so we have reduced the SLAM problem to an EKF problem. Two important implementation details must be discussed at this point.

1. There are times t when range measurements are *not* received. This requires a change in the EKF algorithm; namely at no-measurement times the state estimate simply propagates forward, without receiving a Kalman correction (for more details see [9]).

2. The behavior of EKF can crucially depend on *good initial conditions*. In the range-only SLAM problem discussed here it is necessary to perform some type of batch process to initialize beacon locations, which are then refined within the filter. This is discussed in more detail in [3].

2.2 SLAM by Batch Optimization

The beacons we have described in the previous section have an additional capability: to measure and communicate ranges from one to the other ("inter-beacon" measurements[3]). More specifically, at time t the m-th beacon could contact the the n-th beacon and report their measured range to be $\widehat{R}_{t,m,n}$. We will use this capability in the formualtion of SLAM by batch optimization.

Let us rewrite the state and measurement equations in slightly different form, ignoring the noise term (for reasons wich will presently become obvious).

$$x_{t+1} - x_t = \Delta D_t \cdot \cos(\theta_t) \tag{6}$$
$$y_{t+1} - y_t = \Delta D_t \cdot \sin(\theta_t) \tag{7}$$
$$\theta_{t+1} - \theta_t = \Delta\theta_t, \tag{8}$$
$$\widehat{r}_t = \sqrt{\left(x_t - X_{b(t)}\right)^2 + \left(y_t - Y_{b(t)}\right)^2} \tag{9}$$
$$\widehat{R}_{t,m,n} = \sqrt{(X_m - X_n)^2 + (Y_m - Y_n)^2} \tag{10}$$

In our formulation we will ignore the θ variables; our goal will be to simply estimate the position of the robot (x_t, y_t) (for $t = 0, 1, 2, \ldots$), but not the orientation. Hence we ignore eq.(8); to eliminate θ from (6), (7), we square both sides of the equations, add and take square roots; we obtain

$$\sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2} = \Delta D_t.$$

In short, we want the the variables x_0, y_0, x_1, y_1, ..., x_T, y_T, X_1, Y_1, ..., X_N, Y_N to satisfy the following equations (for $t = 0, 1, ..., T-1$ and $m, n = 1, 2, ...,$):

$$\Delta D_t = \sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2} \tag{11}$$
$$\widehat{r}_t = \sqrt{\left(x_t - X_{b(t)}\right)^2 + \left(y_t - Y_{b(t)}\right)^2} \tag{12}$$
$$\widehat{R}_{t,m,n} = \sqrt{(X_m - X_n)^2 + (Y_m - Y_n)^2} \tag{13}$$

Equations (11)–(13) will probably *not* be exactly satisfied; but we can try to minimize the following *error* (of the equations) *function*:

$$\begin{aligned} J(x_0, y_0, ..., x_T, y_T, X_1, Y_1, ..., X_N, Y_N) \;=\; & \sum_{k=0}^{T-1} \left(\Delta D_t - \sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2} \right)^2 \\ & + \sum_{k=0}^{T-1} \left(\widehat{r}_t - \sqrt{\left(x_t - X_{b(k)}\right)^2 + \left(y_t - Y_{b(k)}\right)^2} \right)^2 \\ & + \sum_{k=0}^{T-1} \sum_{m,n=1}^{N} \left(\widehat{R}_{k,m,n} - \sqrt{(X_m - X_n)^2 + (Y_m - Y_n)^2} \right)^2 \end{aligned} \tag{14}$$

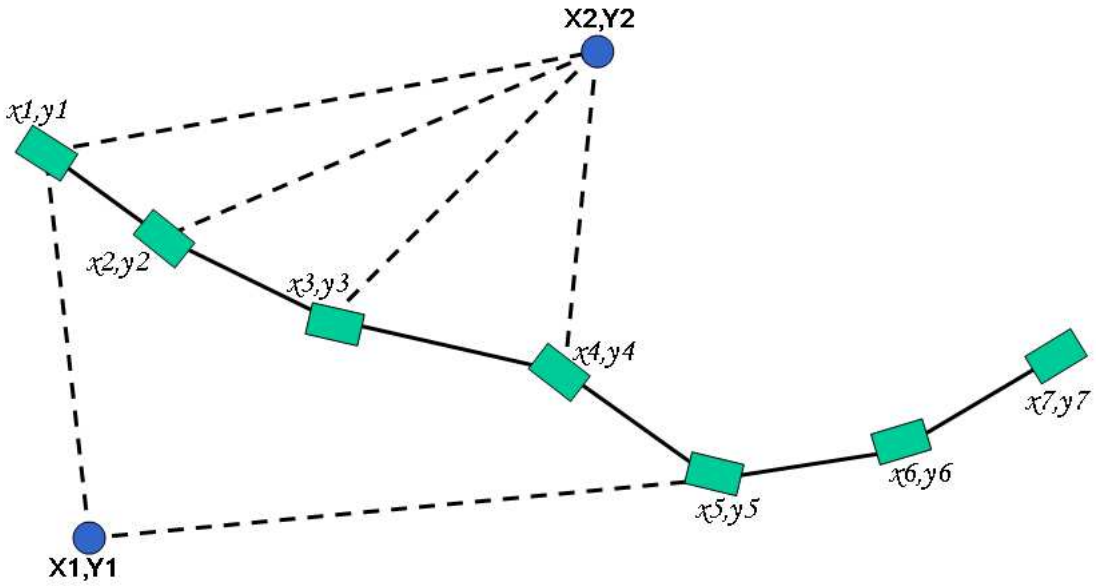

Figure 1: Plot of a part of the path of the robot and the range measurements it receives. The green rectangles are robot positions at successive times $t = 1, 2, ...$; the blue circles are the stationary beacons. Solid lines indicate the robot path (and the respective odometry measurements), dashed lines indicate beacon-to-robot measurements.

i.e. the *total square error*. Hence we define our estimates of the robot path and beacon coordinates by

$$(x_0^*, y_0^*, ..., x_T^*, y_T^*, X_1^*, Y_1^*, ..., X_N^*, Y_N^*) = \arg\min J\left(x_0, y_0, ..., x_T, y_T, X_1, Y_1, ..., X_N, Y_N\right).$$

Many algorithms can be used to actually minimize the cost function (14). For the experiments of this report we have used Gauss-Newton minimization as implemented by the Matlab function `fminunc`.

The above formulation of SLAM admits a simple geometric interpretation, illustrated in Figs. 1, 2. Think of the path of the robot plotted out on the x-y plane, with the positions of the robot at times $t = 0, 1, 2, ..., T$ marked as *nodes*; between each pair of these nodes there is an *edge* of length ΔD_t. The positions of the beacons are also marked as nodes, with additional edges between the beacon nodes (the beacon-to-beacon range measurements) and between beacon nodes and robot nodes (the robot-to-beacon range measurements). Hence the entire plot can be visualized as a *graph*, where edge lengths are known (up to some error) and node coordinates are the unknown variables; the SLAM problem consists in estimating the node coordinates. Compare now the graphs of Figs.1 and 2. In Fig.1 we see the robot path and range measurements *before* interpolation is applied; in Fig.2 *after* interpolation, with additional edges (and their lengths) inserted; these new edges have been obtained by interpolation. It is intuitively obvious that, all other things being equal, as the number of edges increases the above problem will admit a "better" (more accurate, more robust etc.) solution; in other words, it should be easier to specify the coordinates of the graph in Fig.2. This is the motivation for the use of interpolation, which will be further discussed in Section 3.

Similarly to the EKF case, we have given here the equi-spaced data formulation of the batch optimization algorithm. In case range measurements are not received at particular times, the following modification is necessary: in the error function definition (14) the summations of the second and third term are performed only over times at which range measurements are received.

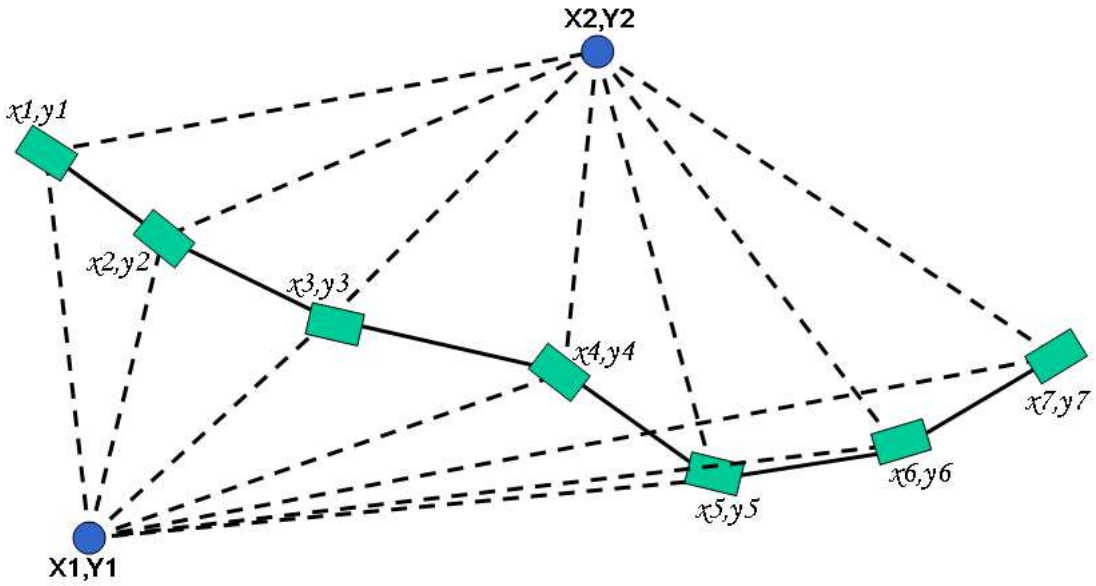

Figure 2: The same plot as in Fig.1, but with additional edge lengths (range measurements) obtained by interpolation.

3 Preprocessing by Interpolation

In this section we first describe *how* to use interpolation to preprocess the range and odometry data; then we discuss about the *potential advantages* of using the interpolated rather than the raw data.

Suppose that range measurements from beacon k to the robot are available at times $t_{k,1}$, $t_{k,2}$, $t_{k,3}$, ... In other words we have the range measurements

$$r_{k,n} = r_{t_{k,n}}, \quad n = 1, 2, 3, ...$$

Note that the time differences $t_{k,2} - t_{k,1}$, $t_{k,3} - t_{k,2}$, are not necessary equal. We can choose some *time step* Δt, take a sequence of times

$$t_i = i \cdot \Delta t \quad (i = 0, 1, 2, 3, ...)$$

and use the range measurements $r_{k,n}$ to interpolate (by *straight line segments*) new ranges $\hat{r}_{k,i}$ at the times t_i. Subsequently, the *interpolation-augmented data* $\hat{r}_{i,n}$'s (instead of the original $r_{k,n}$'s.) can be used by the EKF and optimization algorithms (which in fact will have a simpler implementation when applied to data equally spaced in time). Also note that interpolation can be (and has been) applied to the odometry data (Δd and $\Delta\theta$) as well. Some smoothing, removing of outliers etc. has also been performed.

In Figs.3–5 we plot three series of true range measurements and the corresponding interpolated series. Each figure illustrates some aspect of the interpolation process.

1. In Fig. 3 we see a case where interpolation works to complete the data. The blue stars (actual range measurements) are sparse and equally spaced. The red dots (interpolated range measurements) are equally spaced, closer to each other than the original data and give a better picture of the actual time series of ranges. Note however that there are time intervals (for example for $t \in [130, 140]$, $t \in [200, 210]$) where the true time series changes abruptly and the linear interpolation has a distorting effect.

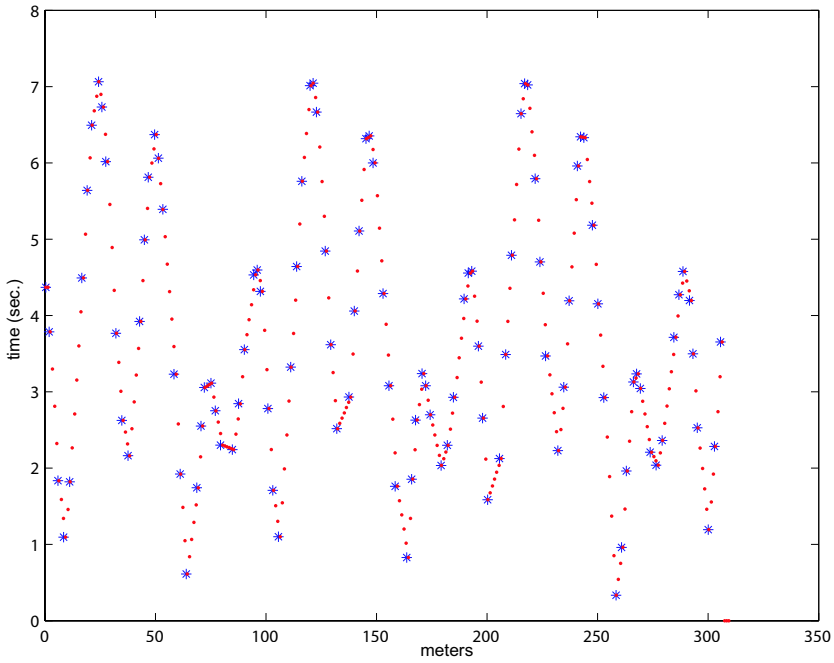

Figure 3: Graph of the true (blue stars) and interpolated (red dots) range measurement time series. The horizontal axis denotes time and the vertical range (meters) from the robot to beacon no.1. It can be seen that the red dots interpolate the true range measurements quite accurately, with a few exceptions; e.g. in time interval [130,140] the range measurements change quickly and the interpolation is not very accurate.

2. In Fig. 4 we see that the actual range measurements are spaced quite irregularly in time. There are intervals with very dense measurements (for example for $t \in [0, 20]$, $t \in [165, 185]$); in this case interpolation thins the data out. In other intervals the measurements are sparse (for example for $t \in [60, 90]$); and interpolation completes the data in a (presumably) non-distorting way. Finally in the interval $t \in [130, 165]$ we have *sensor silence*) and interpolation cannot be performed.

3. Finally, in Fig. 5 we see a case of very sparse measurements, where no useful interpolation can be performed.

In short, we can think of interpolation as a "connect-the-dots" step, which can be performed over a particular time interval *as long as there are enough dots to connect*. In this sense, interpolation does not really create new information; it just rearranges existing information. Another way to look at interpolation is as a form of filtering.

In our current implementation, interpolation is performed *offline*, as a preprocessing step. However, there is no reason, at least in principle, that it could not be performed *online*, perhaps with a small time delay; namely, as soon as the datum $r_{k,n}$ becomes available (at time $t_{k,n}$) , we can interpolate range at every desired time $t \in [t_{k,i-1}, t_{k,i}]$. Note also that we have used straight line interpolation, but we could have used quadratic interpolation, splines or any other desirable function. Finally, note that the odometry data (Δd and $\Delta\theta$) can be (and are) interpolated in exactly the same manner.

Let us now restate the expected *advantages* of using interpolated rather than raw data. We have already discussed this issue in Section 2.2, and it will be ultimately answered by the experiments of Section 4; however our *motivation* can be explained as follows.

1. We can use the interpolation to produce the *"right" amount* of data. We do not want "too few" data; with too little information the SLAM algorithms will not work properly. We do

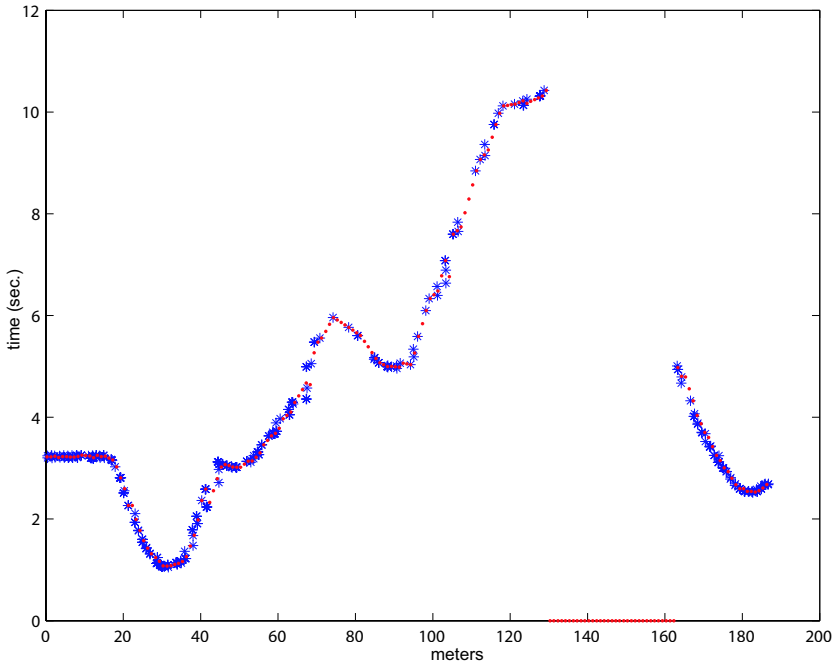

Figure 4: Graph of the true (blue stars) and interpolated (red dots) range measurement time series. The horizontal axis denotes time and the vertical range (meters) from the robot to beacon no.1. It can be seen that in the time interval [130,165] there is sensor silence and interpolation is not possible.

not want "too many" data either; this can cause computational difficulties, particularly for the optimization algorithm: with too many measurements the error function will depend on too many variables, which in turn may result in a complicated error surface with many local minima, excessive computational load etc. By choosing the interpolation step we can control the exact amount of data that we provide to the algorithm (however, the amount of *information* should remain the same).

2. Also, interpolation spreads data *uniformly* over the time interval of interest. We conjecture this will make the algorithms perform better; most estimation, optimization etc. algorithms have originally been designed to operate on uniformly spaced data; extensions to irregularly spaced data are indeed possible, but they may violate some of the basic assumptions underlying the algorithms (an example would be the assumption of constant covariance matrix in the Kalman filter; this implicitly assumes equal times between measurements).

3. A final possible advantage of the interpolation approach is *simplicity of implementation*. Applying the SLAM algorithms to irregularly spaced data requires nonstandard implementations, which are somewhat harder to code (this however is not a decisive advantage, merely an added convenience).

4 Experiments

We now present several experiments (using simulated and real robot paths) to evaluate the usefulness of interpolation.

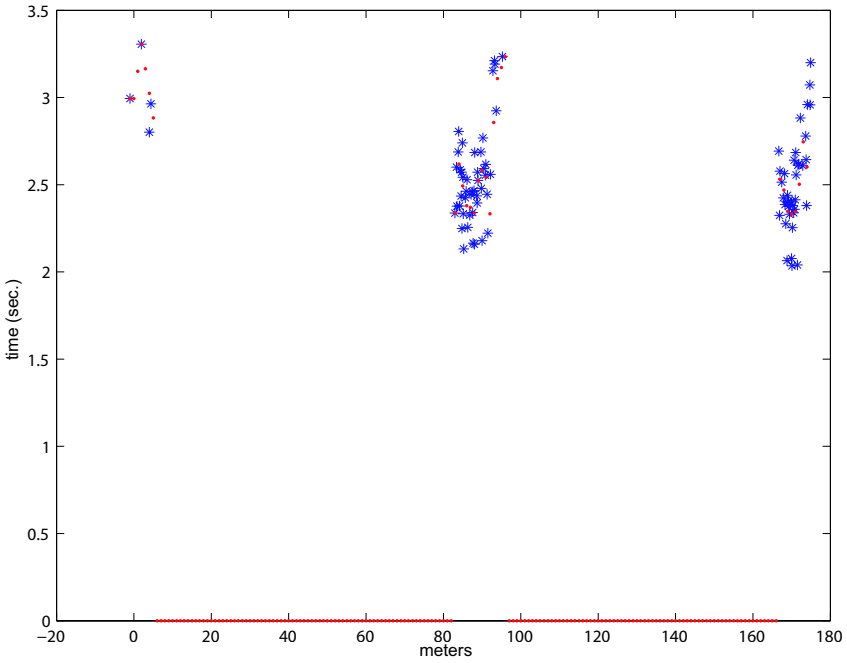

Figure 5: Graph of the true (blue stars) and interpolated (red dots) range measurement time series. The horizontal axis denotes time and the vertical range (meters) from the robot to beacon no.1. It can be seen that the true measurements are very sparse and useful interpolation is not possible.

4.1 Experiment Setup

To evaluate the usefulness of interpolation, we run each of the two SLAM algorithms of Section 2 in two variations on the original, irregularly spaced data as well as on the interpolated, uniformly spaced data. We compare the "plain" and "interpolation enhanced" algorithms using two criteria: goodness of *localization* and goodness of *mapping*. For goodness of localization we use a subjective criterion, namely plotting the true and estimated path and visually comparing them. (Ideally, we should compare the trajectories by an objective criterion, e.g. cross-track, tangential or RMS error. However, this comparison is not easy, because we do *not* know the actual "ground truth" (but only an approximation to it) ans also because the length of the original and interpolated time series is different). For goodness of mapping we can actually compute an objective criterion. Recall that (X_n, Y_n) are the true coordinates of the n-th beacon $(n = 1, 2, ..., N)$; call (X_n^*, Y_n^*) the estimates of the coordinates. The RMS error defined by

$$E_{RMS} = \frac{\sum_{n=1}^{N} \sqrt{\left[(X_n - X_n^*)^2 + (Y_n - Y_n^*)^2\right]}}{N}$$

measures the accuracy of the estimates. An important point to note is that an *affine transform* must be performed on the final beacon locations estimates in order to re-align the locally accurate solution into a global coordinate frame; (X_n^*, Y_n^*) are the estimates after the affine transform has been applied.

The parameters of the two algorithms are as follows:

1. For the EKF algorithm, the statistics of the noise processes are estimated using a method detailed in [3].

2. An important EKF parameter is the estimate of the initial robot position. In this report we assume that this is known with zero error.

3. The batch optimization algorithm also requires initialization (of the entire vector $[x_0, y_0,$ ..., $x_T, y_T,$ $X_1, Y_1,$ ..., $X_N, Y_N]^T$). We initialize $x_0, y_0, ..., x_T, y_T$ using the *dead reckoning* estimate. We initialize $X_1, Y_1, ..., X_N, Y_N$ randomly, by sampling a $2N$-dimensional normal distribution with zero mean and a covariance matrix Σ, which we always take to be diagonal with all elements equal to 5.

4. Regarding the "interpolation enhanced" algorithms, the only additional parameters are the time step Δt (which we take to be 1 for the EKF algorithm and 2 for the batch optimization algorithm) and a cutoff parameter Δd which is used to reject outliers (and is taken equal to 1).

In all experiments reported here we have used 7 beacons ($N = 7$).

Table 1: RMS mapping error in meters (i.e. RMS error of the estimated beacon locations) for datasets A6 and A8.

	EKF	EKF + Interp.	Optim.	Optim. + Interp.
A6	0.0203	0.0228	0.0069	0.0155
A8	0.0077	0.0370	0.0081	0.0348

4.2 Experiment no.1: Simulated Data

The first data set on which we experimented consists of two *simulated* robot paths. In other words, using eqs. (1), (4), (5) and appropriate $\Delta D_t, \Delta\theta_t$ inputs we simulate two robot paths.

1. An elliptical robot path (dataset A6); the path is plotted in Figs.6–9. It covers a terrain of approximately 10-by-15 m, takes 429 sec to complete and has total length 85 m (hence average speed is around 0.20 m/sec). There is a total of 1000 odometry measurements (to which additive Gaussian white noise with $\mu = 0$ and $\sigma = 0.01$ is superimposed) and 1000 range measurements (hence 140 measurements per beacon on the average). One range measurement is produced every 0.5 sec on the average; the beacon which generates this measurement is selected randomly. Beacon-to-beacon range measurements are also randomly generated.

2. A "figure-eight" robot path (dataset A8); the path is plotted in Figs. 10–13. It covers a terrain of approximately 10-by-20 m, takes 309 sec to complete and has total length 155 m (hence average speed is around 0.50 m/sec). There is a total of 1000 odometry measurements and 1000 range measurements (hence 140 measurements per beacon on the average). The details regardng the odometry and range data are the same, except that one range measurement is produced approximately every 0.30 sec.

In Figs. 6-13 we present the SLAM results. It can be seen in these figures that the "interpolation enhanced" algorithms give good localization and mapping and the same is true of the standard (without interpolation) EKF; optimization without interpolation does not work as well. Indeed, it is worth comparing in greater detail the batch optimization algorithms without and with interpolation. Their output appears in Figs.8-9 (dataset A6) and Figs.12-13 (dataset A8). The reader will observe that in Figs.8 and 12 we only plot a small part of the entire trajectory and its estimate. The reason is that we have *not* been able to solve the entire problem, because of the computational burden. It is worth explaining the issue in detail.

1. Consider dataset A6. *Without interpolation*, we have 1000 range measurements, each of these coming at a separate time instant. We can "syncrhonize" odometry measurements to range measurements (by integrating odometry between time instants of range measurements) but this still leaves us with 1000 time instants (i.e. $t = 1, 2, ..., 1000$), which means that our error function depends on 2000 variables (one x and one y for every time step) plus 14 more variables corresponding to the coordinates of the seven beacons. An nonlinear optimization problem with 2014 variables is *hard* even under "normal" circumstances; the fact that the corresponding graph (see Fig.1) has relatively few edges makes the problem even harder. Hence it is not surprising that `fminunc` fails to converge to a solution of this problem, or converges to a local, obviously "bad" minimum. To overcome the problem we ran `fminunc` on a subset of the problem, i.e. on the odometry and range measurements corresponding to 200 (rather 1000) consecutive time steps (a "path segment"). In this case, our cost function depends on 414 variables (a much smaller problem) and `fminunc` converges to the solutions illustrated in Fig.8. It can be seen that path estimates are reasonable and beacon coordinates excellent. However, it must be noted that these

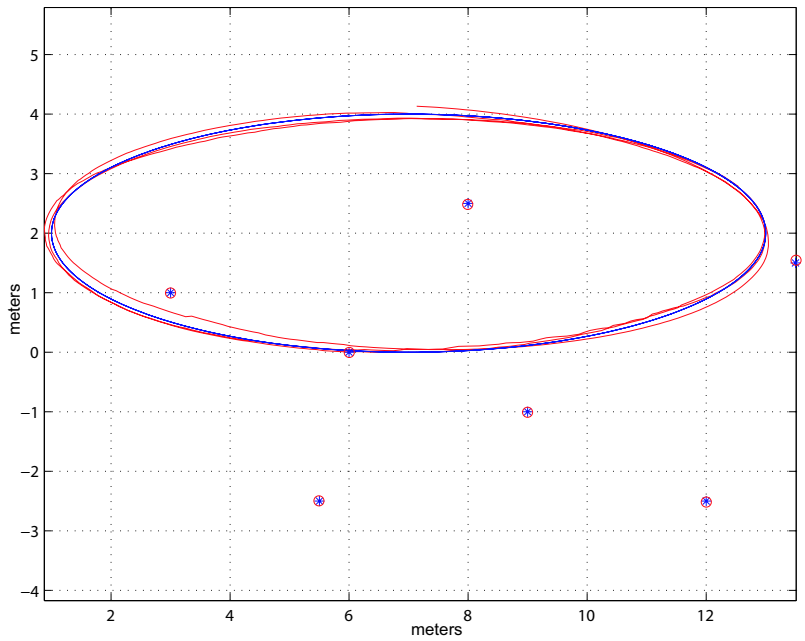

Figure 6: The true A6 path (blue line) and its estimate (red line) by the EKF algorithm. Black stars denote the true beacon locations and red circles their estimates.

results were obtained after 100000 iterations of the batch optimization algorithm, which took close to 15 min of computing time.

2. *With interpolation* things become a lot easier. By using a $\Delta t = 2$ sec, we have 210 distinct time steps *for the entire robot path*, which means 434 variables; however, the 215 time steps correspond to 215*7=1505 range measurements (since we complete "missing" measurements by interpolation). As a consequence the problem is much better posed and a solution is obtained in 30000 iterations which, moreover, are executed a lot faster; an entire batch optimization run takes less than 3 min. The results (illustrated in Fig.9) are comparable to the ones obtained by optimization without interpolation (actually, path estimates appear to be better), but concern the entire path and are obtained much faster. Hence it is rather clear that interpolation brings a definite advantage.

3. The above remarks concern dataset A6; similar things are true for dataset A8 (results are plotted in Figs. 8 and 12).

In short, it appears from the figures and the above discussion that the "interpolation enhanced" algorithms give better path estimates. Mapping results are summarized in Table 1 and show that "standard" EKF performs better than the "interpolation-enhanced" one. Interpolation-based batch optimization compares evenly with EKF without interpolation, outperforming it on A6, but being outperformed on A8. Optimization without interpolation gives mediocre localization results on a *subset* of the robot path; it gives very good mapping results, but at the cost of long execution time.

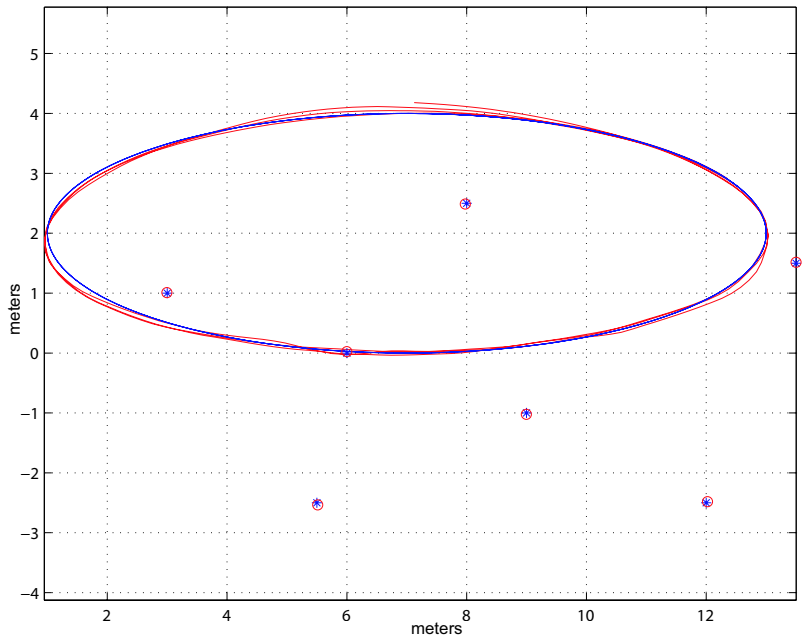

Figure 7: The true A6 path (blue line) and its estimate (red line) by the EKF algorithm with interpolation. Black stars denote the true beacon locations and red circles their estimates.

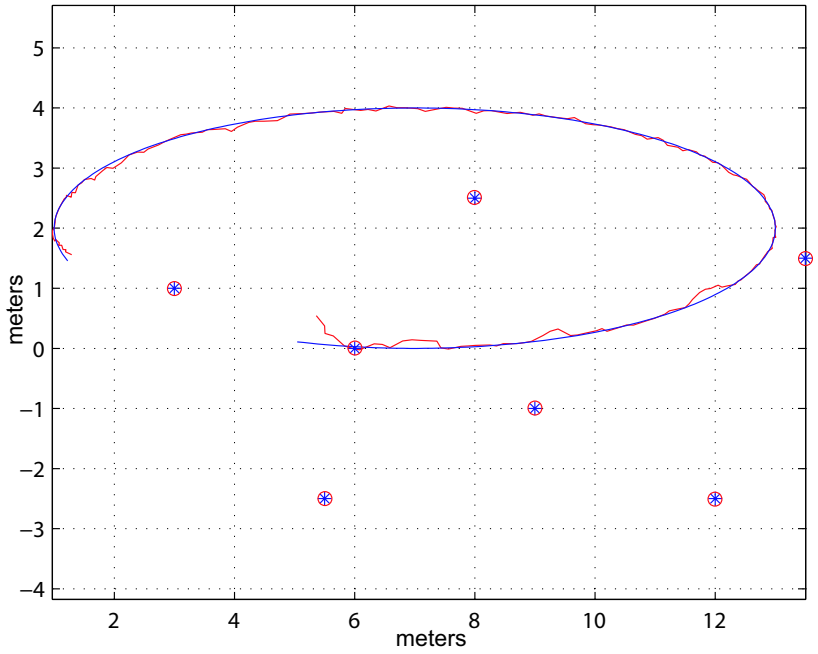

Figure 8: The odometry estimated A6 path (blue line) and the estimate (red line) by the batch optimization algorithm. Black stars denote the true beacon locations and red circles their estimates. Because the computational burden for estimating the entire robot path is too heavy, only a path segment is estimated and plotted.

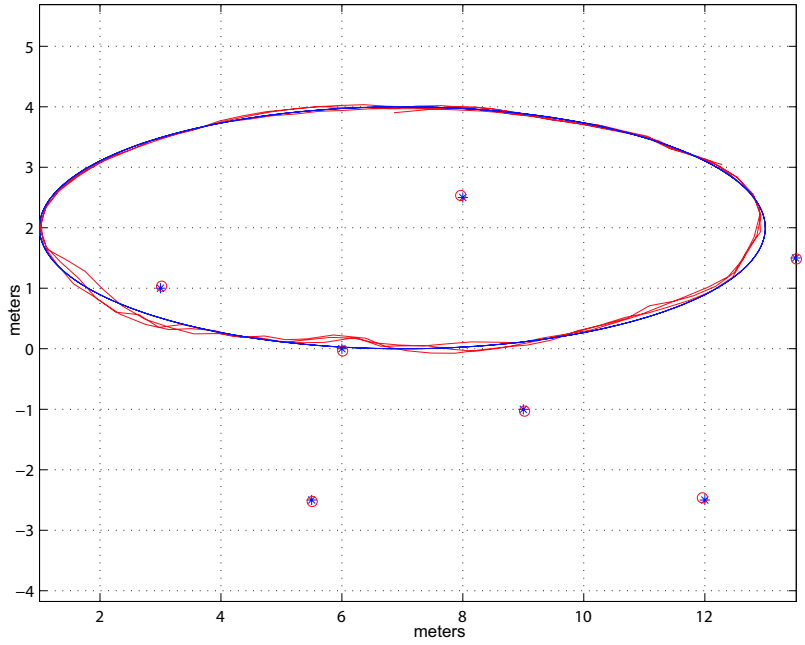

Figure 9: The true A6 path (blue line) and its estimate (red line) by the batch optimization algorithm with interpolation. Black stars denote the true beacon locations and red circles their estimates.

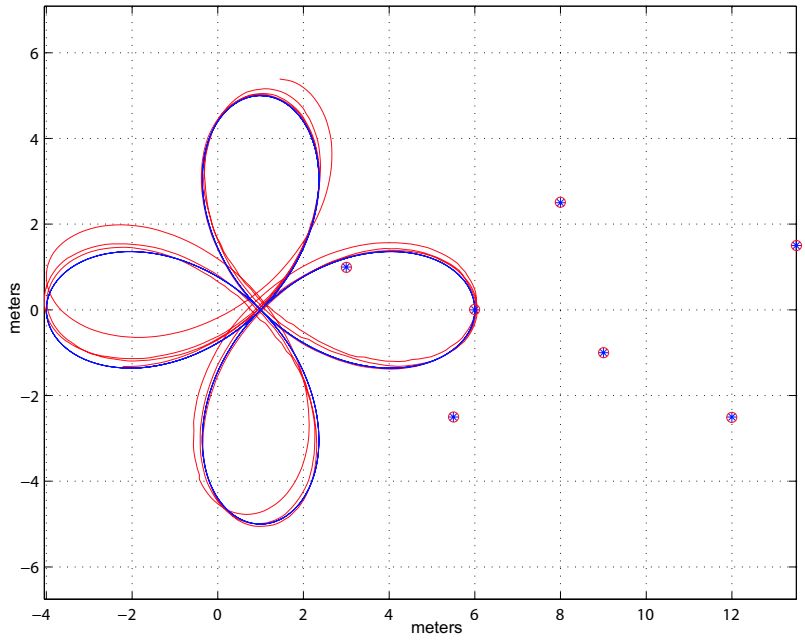

Figure 10: The true A8 path (blue line) and its estimate (red line) by the EKF algorithm. Black stars denote the true beacon locations and red circles their estimates.

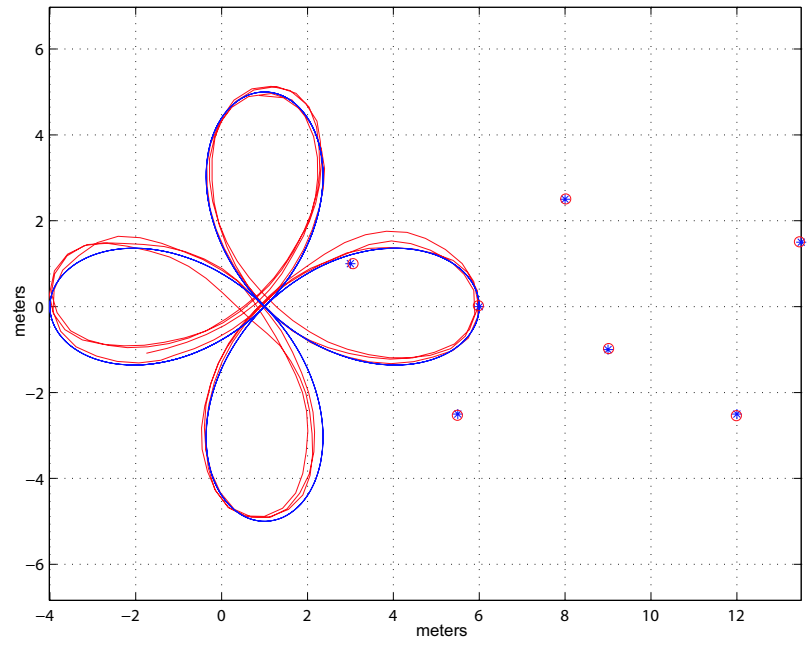

Figure 11: The true A8 path (blue line) and its estimate (red line) by the EKF algorithm with interpolation. Black stars denote the true beacon locations and red circles their estimates.

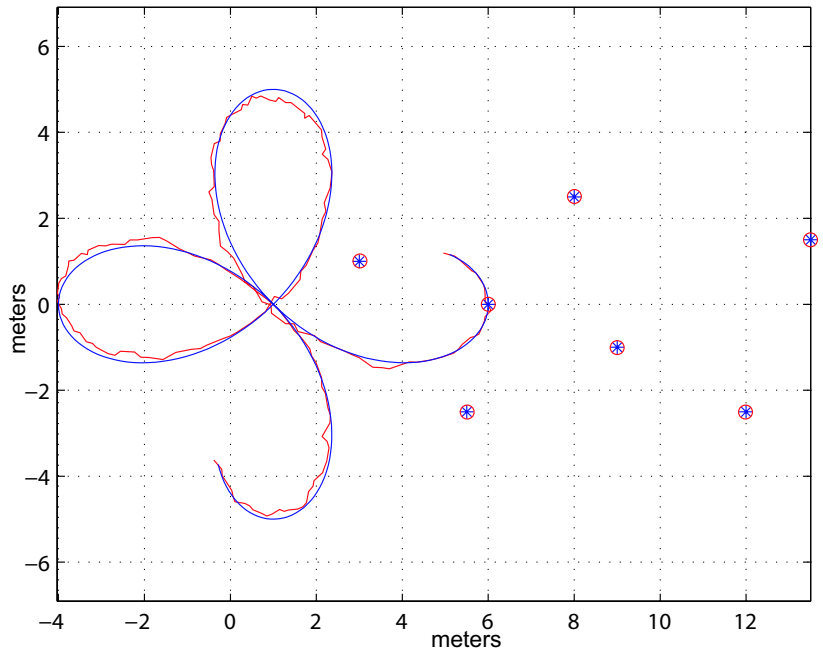

Figure 12: The odometry estimated A8 path (blue line) and the estimate (red line) by the batch optimization algorithm. Black stars denote the true beacon locations and red circles their estimates. Because the computational burden for estimating the entire robot path is too heavy, only a path segment is estimated and plotted.

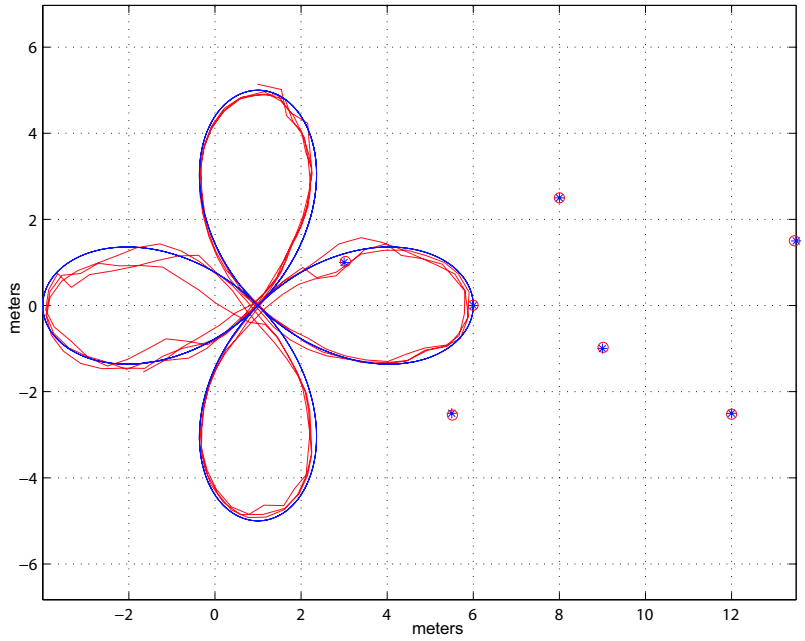

Figure 13: The true A8 path (blue line) and its estimate (red line) by the batch optimization algorithm with interpolation. Black stars denote the true beacon locations and red circles their estimates.

Table 2: RMS mapping error in meters (i.e. RMS error of the estimated beacon locations) for datasets B1 and B4.

	EKF	EKF + Interp.	Optim.	Optim. + Interp.
B1	0.1287	0.1289	0.2054	0.0992
B4	0.1767	0.1495	2.7502	0.1003

4.3 Experiment no.2: Real Rectilinear Data

This group of experiments uses data collected from a real robot and beacons. We carried out our experiments on a Pioneer 1 robot from ActivMedia. It is equipped with a sonar/ radio beacon (Parrot) that is placed on top of the robot at about 1 ft above the floor. This beacon is capable of measuring range to other beacons in the environment using a combination of sonar and radio signals, [17]. The robot was controlled by an operator who drove the robot using a wireless link. The robot was driven around within a large indoor area with partial clutter in and around the area. There was no ground truth available for the exact path taken by the robot, however, the robot's intended path was surveyed. The true robot path differs somewhat from the intended path, however, it is close enough for the purposes of judging the validity of the localization and SLAM algorithm. In addition to the beacon that was placed on the robot, several other beacons were placed around the environment on top of boxes, 1 ft above the floor. The locations of these beacons were accurately surveyed to allow proper evaluation of the accuracy of our SLAM mapping results. We collected two datasets (B1, B4).

1. Dataset B1 (plotted in Figs.14–17 covers a terrain of approximately 15-by-5 m, and travels a total distance of about 35 m in 190 sec; it contains about 5000 range measurements (including both robot-to-beacon and beacon-to-beacon).

2. Dataset B4 (plotted in Figs.18–21) covers a terrain of approximately 20-by-10 m, travels a total distance of about 50 m in 240 sec and contains about 4500 range measurements (including both robot-to-beacon and beacon-to-beacon). Note that here range-measurement density is lower than for B1, because we have fewer measurements over a longer time interval. Also, because of the presence of obstructions, some beacons give a quite smaller number of range measurements than average.

In Figs. 14-21 we present the SLAM results. The following conclusions can be obtained from these figures and Table 2 which summarizes mapping results.

1. *EKF without interpolation* gives reasonable, but not good path estimates; it gives good mapping results.

2. *EKF with interpolation*, when compared to "plain" EKF, improves very significantly the path estimates and also improves somewhat the mapping results .

3. *Optimization without interpolation* will (similarly to datasets A6, A8) *not* work on entire paths, so we use "path segments". For B1 the path estimates (localization) are rather poor but mapping (beacon coordinates) is reasonable. For B4, the method fails: both path and beacon estimate is very bad. Furthermore, these results (for B1, B4) require *heavy* computation: 600000 iterations, which take over one hour of computing.

4. *Optimization with interpolation* gives the overall best mapping results and quite good localization as well; computation time is less than 5 minutes in every case.

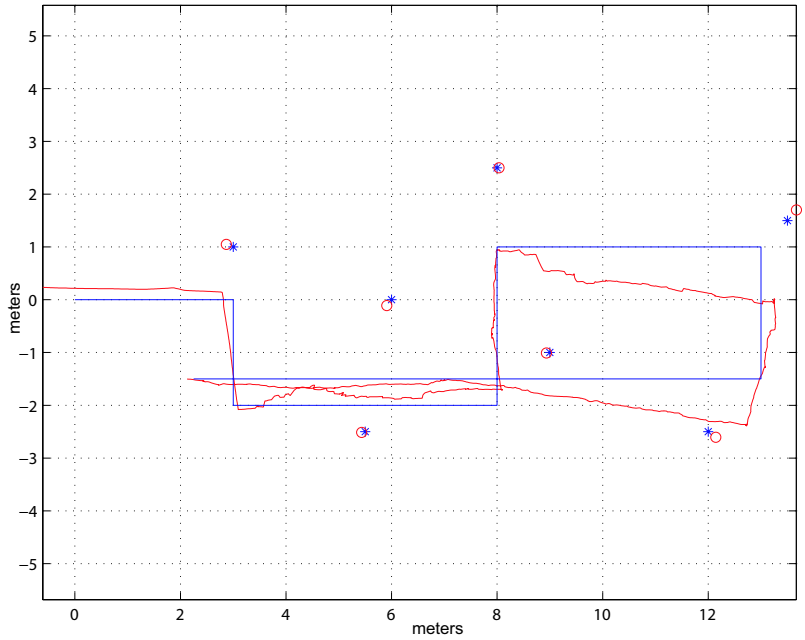

Figure 14: The true B1 path (blue line) and its estimate (red line) by the EKF algorithm. Black stars denote the true beacon locations and red circles their estimates.

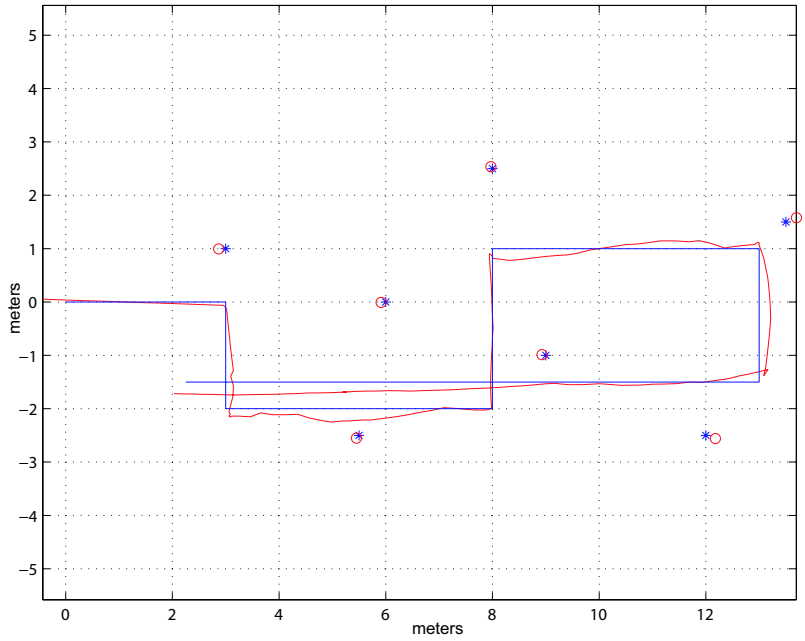

Figure 15: The true B1 path (blue line) and its estimate (red line) by the EKF algorithm with interpolation. Black stars denote the true beacon locations and red circles their estimates.

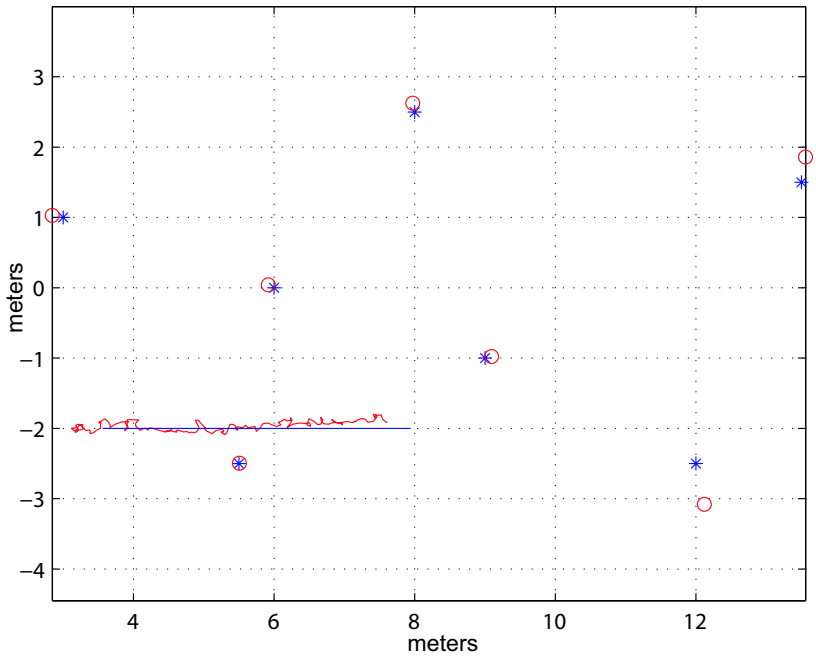

Figure 16: The odometry estimated B1 path (blue line) and the estimate (red line) by the batch optimization algorithm. Black stars denote the true beacon locations and red circles their estimates. Because the computational burden for estimating the entire robot path is too heavy, only a path segment is estimated and plotted.

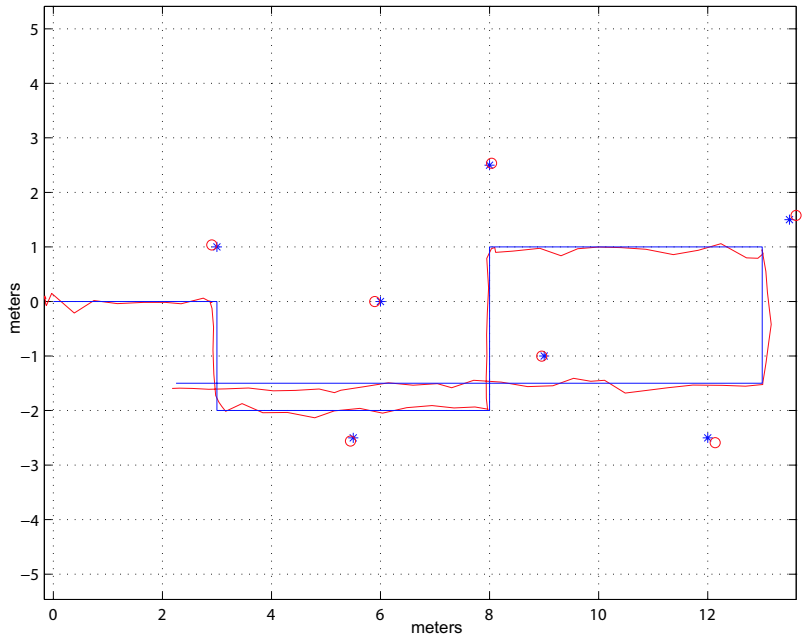

Figure 17: The true B1 path (blue line) and its estimate (red line) by the batch optimization algorithm with interpolation. Black stars denote the true beacon locations and red circles their estimates.

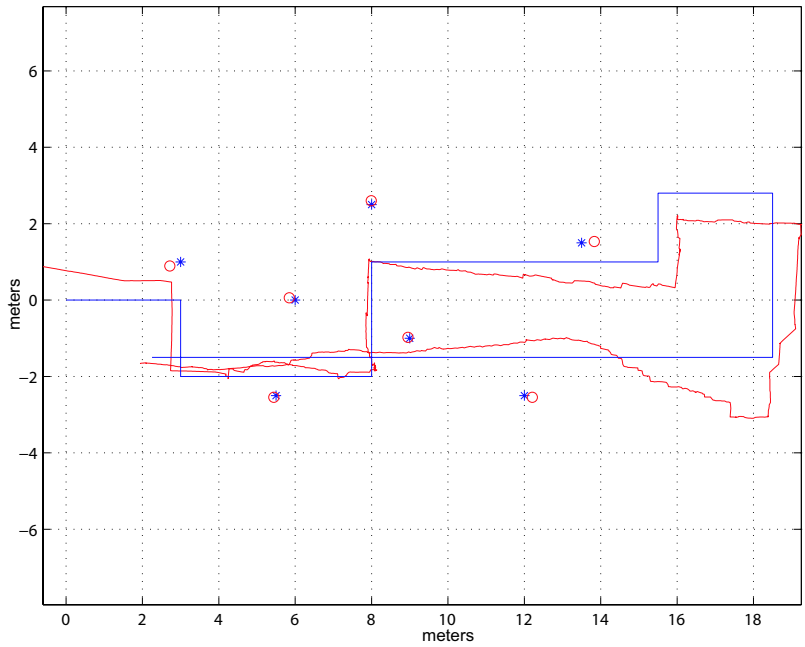

Figure 18: The true B4 path (blue line) and its estimate (red line) by the EKF algorithm. Black stars denote the true beacon locations and red circles their estimates.

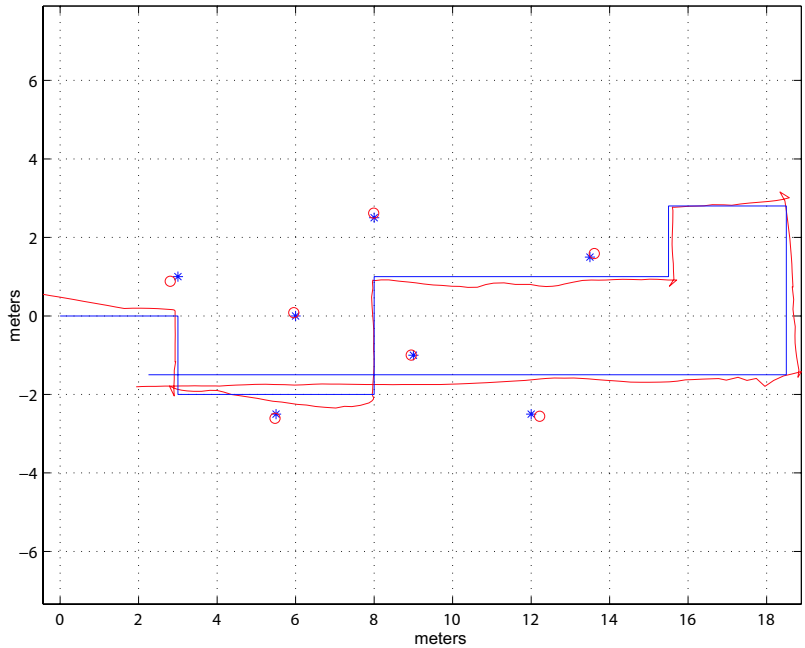

Figure 19: The true B4 path (blue line) and its estimate (red line) by the EKF algorithm with interpolation. Black stars denote the true beacon locations and red circles their estimates.

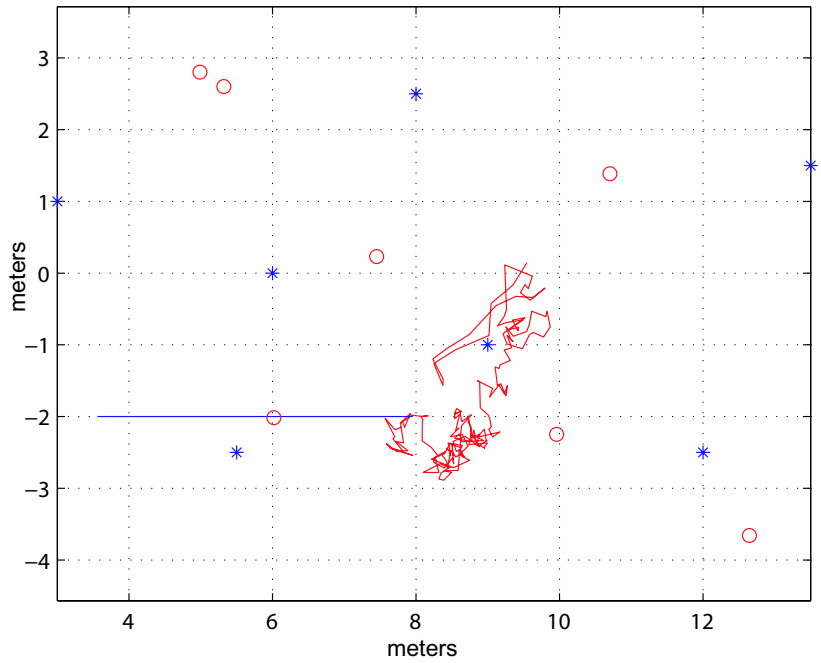

Figure 20: The odometry estimated B4 path (blue line) and the estimate (red line) by the batch optimization algorithm. Black stars denote the true beacon locations and red circles their estimates. Because the computational burden for estimating the entire robot path is too heavy, only a path segment is estimated and plotted.

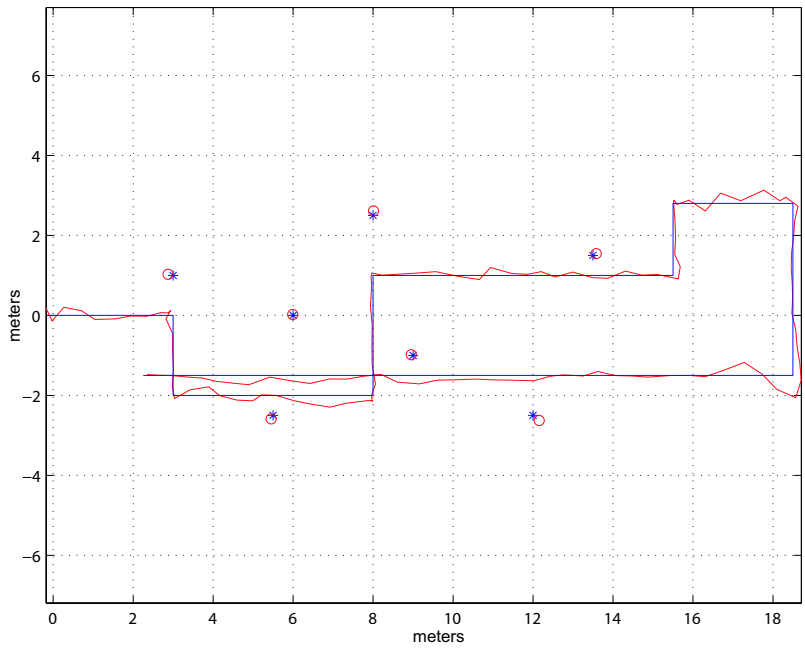

Figure 21: The true B4 path (blue line) and its estimate (red line) by the batch optimization algorithm with interpolation. Black stars denote the true beacon locations and red circles their estimates.

Table 3: RMS mapping error in meters (i.e. RMS error of the estimated beacon locations) for datasets C1 and C2.

	EKF	EKF + Interp.	Optim.	Optim. + Interp.
C1	2.0809	——	——	0.1893
C2	4.5319	——	——	0.2576

4.4 Experiment no.3: Real Curvilinear Data

The final two datasets come from the same robot and terrain as those of Section 4.3. The robot performs a "figure eight" motion (dataset C1) and a collision avoidance motion which resembles a random walk (dataset C2).

1. The "figure-eight" path takes 174 sec to complete and has total length of 49 m. There is a total of 1729 odometry measurements and 2896 range measurements (including robot-to-beacon and beacon-to-beacon).
2. The "collision avoidance" path takes 239 sec to complete and has total length of 71 m. There is a total of 2342 odometry measurements and 3875 range measurements (including robot-to-beacon and beacon-to-beacon).

These datasets are harder than the ones presented in previous sections. Odometry measurements are unreliable, probably because of frequent (and sometimes abrupt) changes in direction. The difficulty of the datasets is reflected in the quality of the results, which can be summarized as follows

1. First of all, it must be noted that (a) EKF with interpolation and (b) batch optimization without interpolation failed to produce any usable results.
2. Also (a) EKF without interpolation and (b) batch optimization with interpolation produced poor path estimates (localization).
3. The interpolation based batch optimization algorithm gives *quite good* estimates of the beacons (*mapping*), but not as good as the ones on datasets A6, A8, B1 and B4.
4. All other methods give poor beacon estimates. We summarize RMS error on beacon coordinate estimates in Table 3.

5 Conclusion

The experiments of Section 4 indicate that interpolation *does* improve SLAM results. We can look at localization and mapping separately.

1. Regarding *localization*, a subjective inspection of the estimated paths shows that in nearly all datasets the interpolation based algorithms give better approximations to the assumed true paths, or at least very comprable to the ones obtained by the standard algorithms; the sole exception is the performance of the EKF algorithm with interpolation on datasets C1 and C2, where it failed.
2. Regarding *mapping*, the objective RMS criterion shows that the interpolation based algorithms give better estimates of beacon coordinates than the ones without interpolation, except for the fact that the EKF algorithm with interpolation failed to produce such estimates on datasets C1 and C2 (however the "straight" EKF algorithm also gave very bad estimates on this dataset).

Hence it appears that the use of interpolation is justified and worth of further research.

References

[1] M. Deans and M. Hebert, "Experimental comparison of techniques for localization and mapping using a bearing–only sensor," in *Seventh International Symposium on Experimental Robotics*. Honolulu, HI: Springer Verlag, December 2000, pp. 393–404.

[2] J. Djugash, S. Singh, and P. Corke, "Further results with localization and mapping using range from radio," in *Proceedings of FSR2005*, Cairns, Australia, August 2005.

[3] J. Djugash, S. Singh, and G. Kantor, "Range-only SLAM for robots operating cooperatively with sensor networks" in *Proceedings of ICRA2005*, 2005.

[4] L. Doherty, K. S. J. Pister, and L. E. Ghaoui, "Convex position estimation in wireless sensor networks."

[5] J. Guivant, F. Masson, and E. Nebot, "Simultaneous localization and map building using natural features and absolute information," *Robotics and Autonomous Systems*, 2002.

[6] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "A highly efficient fast SLAM algorithm for generating cyclic maps of large-scale environments from raw laser range measurements," in *Proceedings IROS 2003*, Las Vegas, NV, 2003.

[7] G. Kantor and S. Singh, "Preliminary results in range-only localization and mapping", *Proceedings of ICRA 2002*, 2002.

[8] D. Kurth, G. Kantor and S. Singh, "Experimental results in range-only localization with radio", *Proceedings of IROS 2003*, 2003.

[9] D. Kurth. *Range-only robot localization and SLAM with Radio*. Tech. Report CMU-RI-TR-04-29, Robotics Institure, Carnegie Mellon Univ., 2004.

[10] J. Leonard and H. Feder, "A computationally efficient method for large-scale concurrent mapping and localization," in *Robotics Research: The Ninth International Symposium*. Snowbird,UT: Springer Verlag, 2000, pp. 169–176.

[11] J. Leonard and H. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," in *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, Washington D.C., USA, May 1991.

[12] M. Montemerlo, "FastSLAM: A factored solution to the simultaneous localization and mapping problem with unknown data association," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, July 2003.

[13] P. Newman and J. Leonard, "Pure range-only sub-sea slam," in *Proceedings of IEEE Conference on Robotics and Automation*, Boston, USA, September 2003.

[14] E. Olson, J. Leonard, and S. Teller, "Robust range-only beacon localization," in *Proceedings of Autonomous Underwater Vehicles, 2004*, 2004.

[15] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-free distributed localization in sensor networks," MIT LCS, Tech. Rep. TR-892, April 2003.

[16] S. Singh, G. Kantor and D. Strelow. "Recent results in extensions to simultaneous localization and mapping". In *Proceedings of Int. Symposium on Experimental Robotics*, 2002.

[17] W. Zhang and J. Djugash and S. Singh. "Parrots: A Range Measuring Sensor Network". Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-06-05, in progress.