Range-based navigation system for a mobile robot

Neil MacMillan, River Allen, Dimitri Marinakis, Sue Whitesides
Department of Computer Science, University of Victoria
neilrqm@gmail.com, riverallen@gmail.com, dmarinak@gmail.com, sue@uvic.ca

***Abstract*—In this paper we present an algorithm for path planning in a fixed range-only beacon field. We define and calculate *entropy* values for regions of interest and provide a method for finding "safe," low-entropy paths between regions. We go on to describe a robotic system for performing range-based localization experiments, developed using inexpensive off-the-shelf components. Our system uses a commercial robot as a mobile platform and custom acoustic beacons for ranging.**

***Keywords*-robot localization, range sensors, robot path planning, Bayesian filtering**

I. Introduction

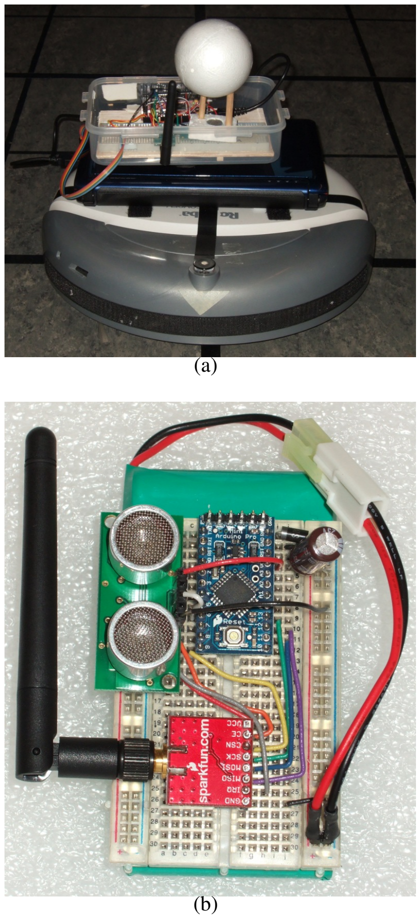
In the future, it can be expected that mobile robots will continue to be applied to automated tasks in indoor industrial settings, *e.g.* see the work of Wurman *et al.* [1]. Given the widespread prevalence of wireless sensor networks for automated data acquisition and control purposes it is likely a typical industrial environment will have *a priori*, statically deployed wireless devices that can be used by a mobile robot as landmarks with *unique* signatures (*e.g.* based on their Media Access Control (MAC) identifiers).

In this paper we investigate approaches that can allow a mobile robot to exploit information provided by wireless devices already present in the environment. We examine a case in which the robot is assumed to be carrying out a routine task that requires it to navigate arbitrarily between a finite number of specified locations, *e.g.* for the transporting of goods in a warehouse application. Under these circumstances, we are interested in identifying *safely navigable* regions and routes in the instrumented environment that can allow the robot to successfully satisfy the requirements of its task. By safely navigable regions we mean areas where the robot's pose uncertainty is relatively low. We use informational *entropy* to quantify uncertainty; this is related to the idea of using regions of high information content to aid navigation discussed by Roy and Thrun [2]. A region of low entropy is considered safely navigable. We use the term *navigation graph* to refer to a set of paths between safely navigable regions.

We assume only that the robot can: 1) determine a range estimate to beacons in the environment and 2) use odometry information to aid its pose estimate. For our investigations we use a custom range-based navigation system designed around a simple commercial robot and off-the-shelf, wireless components (Figure 1).

The contributions of this paper include:

1) an algorithm for constructing a navigation graph that can aid path planning in circumstances of normal operation for robots with minimal sensing capabilities;



(a)

(b)

Fig. 1. (a) iRobot Roomba instrumented with navigation components and (b) time-difference-of-arrival ranging beacon.

2) a detailed presentation of the system details necessary to architect a range-based robot navigation system using affordable, off-the-shelf, components.

In the remainder of this paper we first present our problem formulation and our path planning algorithm. We then proceed to describe our experimental platform, and conclude with simulation and experimental results.

II. Background

Previous work co-authored by one of us (Marinakis) looks at localization issues in 'hybrid' systems that combine both static sensors and a mobile robot [3], [4]. The approach assumes that the locations of the static sensors are unknown *a priori* and uses Markov Chain Monte Carlo (MCMC) based algorithms to find a probability distribution function (PDF) for both the sensor locations and robot positions. The work assumes that in such a hybrid system the robot will continue to operate in the region of the sensor network and that therefore it is reasonable to expend computational effort up front to later facilitate accurate on-line localization. In the work we present in this paper we continue to examine hybrid sensor networks and mobile robot localization systems, but make the differentiating assumption that the sensor locations are known ahead of time.

The concept of evaluating the localizing accuracy available at various regions in the environment has been considered before. Prior work such as that by Jourdan and Roy [5] consider how to deploy a network of static ranging beacons (sensors) for mobile robot localization. They present an algorithm capable of finding an optimal deployment for minimizing the maximum lower bound on the localization accuracy at a single point and show that their approach also works well in practice for finding deployments that provide good localization over a predetermined path. This differs from the problem we consider, in which we wish to find safe paths among specified regions in the environment given a pre-deployed network.

Jourdan and Roy [5] use the Cramer-Rao bound (CRB) on the localization uncertainty as the key metric used to judge relative localization accuracy among different regions. This is a common approach and is also used in the related areas of mobile target tracking, *e.g.*, the work of Martinez and Bullo [6], and in sensor network localization, *e.g.*, Savvides *et al.* [7]. Under simple beacon assumptions such as unlimited range, the relative CRB is proportional to the relative geometric dilution of precision [5]. The derivation of the CRB can be challenging, however, for some beacon assumptions; see Jourdan *et al.* [8] for a result appropriate for ultra-wide bandwidth ranging. In the work we present in this paper, we use a direct estimate of the *entropy* in the robot's PDF given its *true* location as a metric of localization accuracy. This allows us to use an arbitrary beacon model. [1]

Related work by Batalin *et al.* [9] considers robot navigation using a sensor network of known position. The approach uses radio signal strength to guide the robot to regions in the network proximal to destination nodes. Conceptually similar work by Verma *et al.* [10] presents a system that uses a network of pre-deployed devices to construct a navigation field through the broadcasting of packets. The navigation field is used to guide a robot to areas of interest in the region of the network.

Sampling based motion planning approaches based on the classic paper of Kavraki *et al.* [11] aim to build a road map consisting of free configurations in the configuration space of a robot moving in an obstacle-cluttered environment, with free configurations joined by edges (*i.e.* straight line segments) consisting entirely of collision-free positions. If a road map with reasonable coverage of the free space of configurations can be built in a preprocessing step, then it should be possible to join start and goal configurations to the road map; if they lie in the same connected component, then a collision-free path can be determined relatively quickly. A large body of literature elaborates this approach (see [12] ch. 5). The objective is primarily to find a collision free path. Our approach likewise precomputes a map; however, the objective is to find good paths amongst regions known to be of interest, where "goodness" is determined by a measure of robot localization ease and likelihood of traversing a path successfully.

Potential function methods (see LaValle [12]) direct robots to travel in a direction that reduces some potential function. Functions that are guaranteed not to trap the robot in a local minimum are called *navigation functions*. Again, these methods aim to find a path from a start to a goal.

Many graph model approaches to path planning use vertices to represent cells in a cell decomposition of configuration space or physical space [12]. If cell adjacencies are known, and methods for moving within a cell are available, then finding a path becomes a graph theoretic problem. Graph edges may be weighted, and Dijkstra's classic shortest-path algorithm can be used.

By contrast, our graph theoretic model does not decompose the physical space. Rather, it associates vertices with regions within which the robot can be localized accurately as well as regions to which the robot will often be commanded to travel, and associates edges with routes between these regions that are believed to be fairly reliable. Thus the focus is not on finding a path, but rather, on finding what we call a *navigation graph*, i.e., a reliable set of paths that connect regions of frequent operation for the robot. Once precomputed, the navigation graph may be used repeatedly to provide recommended routes for reliable travel between regions of operation shown on the map.

III. The Navigation Graph

We consider the problem of constructing a *navigation graph* $G = (V, E)$ for a beacon-instrumented environment suitable for pragmatically aiding higher level processes such as long term path-planning.

The vertices $V = R \cup \Gamma$ of the navigation graph G represent regions of interest in the environment. Our regions of interest include both initially provided *operating regions* R in the environment that the robot will be expected to visit during routine operations, and additional locations Γ' (from which Γ is selected to form G) that allow the robot to navigate between the operating regions using straight-line transits. The edges $E = \{e_{ij}\}$ of the navigation graph G represent paths through the graph weighted in proportion to the *cost* of the path between the two neighbouring vertices.

Once a cost-weighted navigation graph is constructed for an environment, it can be used for fast, on-line path planning purposes. For example, to plan a route between any two regions, one can apply Dijkstra's algorithm in $|V|log(|V|)$

[1] Later we derive a closed form solution specific to a normally distributed error model. This result gives us a computational speed up, but is not critical to the overall approach.

time [13] to find a suitable path along the edges of the navigation graph.

One can also use the navigation graph obtained from a given set of beacon locations and operations regions to assess the likelihood of operational feasibility. For example, one could construct an unweighted graph G^* by selecting only the edges in G that are under a desired edge cost threshold. An unconnected G^* would then suggest unsuitability. [2]

IV. Algorithm for Navigation Graph Construction

In this section we present an approach for constructing a navigation graph suitable for an environment instrumented with uniquely identifiable beacons of known location from which range estimates can be obtained. We take the following high level steps:

1) Estimate a discretized *entropy map* of the environment;
2) Draw samples of candidate navigation locations (x, y) from Γ';
3) Estimate the *navigation cost* between each pair of locations in $R \cup \Gamma'$;
4) Produce an initial navigation graph G' in which the vertices are $R \cup \Gamma'$;
5) For each pair of operating regions $r_i, r_j \in R$ find the lowest cost path p_{ij};
6) Construct a final navigation graph $G = R \cup \Gamma$ using $P = \bigcup p_{ij}$, where Γ is the set of navigation locations from Γ' that are used by at least one lowest cost path in P.

In the next sections we provide some more details on our approach.

A. Beacon Model

A fundamental input to our approach is a beacon model that specifies when the range to a beacon can typically be obtained and what error is associated with a range estimate when one is obtained. We make the following assumptions:

1) Each beacon position is specified by a location and orientation (x, y, θ), *i.e.* its pose;
2) A range estimate to the beacon can be obtained by the robot if the true distance of the robot to the beacon is less than the beacon's maximum range λ, and the angle α between the beacon and the robot is within an angular range β; i.e. $\alpha \in (\theta - \beta, \theta + \beta)$;
3) Range estimates have zero-mean, normally distributed noise added to them; i.e. $r = r' + N(0, \sigma)$ where r is the distance estimate, r' is the true distance and σ is the variance of the noise.

B. Entropy Map

Given the beacon model specified in section IV-A, we compute a grid map of the environment where the value of each grid square gives an estimate of the entropy of the PDF for the robot's position *should its true location* be the grid square.

First let us consider the question of the uncertainty in the robot's pose given a single observation of the range r to the beacon and the beacon model specified above (analogous to a measurement model). From Bayes' Law we have:

$$\begin{aligned} p(x|r) &= \frac{p(r|x)p(x)}{p(r)} \\ &\propto p(r|x) \end{aligned} \quad (1)$$

where x is the position of the robot and we assume a constant prior over the robot's position and data. By applying Equation 1 one can obtain an estimate of the PDF X for the robot using an occupancy grid over the area being considered. The PDF $X = \{p(x_{ij})\}$ where $p(x_{ij})$ gives the relative probability of the robot being located in grid square X_{ij}. From the PDF estimate, one can then compute an estimate of the entropy associated with the position estimate: $H(X) = \sum p(x_{ij}) \log p(x_{ij})$.

Let us now consider estimating the PDF and associated entropy of an estimated robot position x given the *true* robot position x' for this single beacon case:

$$\begin{aligned} p(x|x') &= \int_0^{+\infty} p(x, r|x')dr \\ &\propto \int_0^{+\infty} p(x|r)p(r|x')dr \\ &\propto \int_0^{+\infty} p(r|x)p(r|x')dr \end{aligned} \quad (2)$$

using Equation 1.

Given a model for $p(x|r)$ and $p(r|x')$, one can estimate the integral shown in Equation 2 using numerical methods.

Since both $p(x|r)$ and $p(r|x')$ are modeled in this work using a normal distribution, a closed form equation can be obtained. Let us assume the measurement model takes the form of: $N(r; x', \sigma)$. In this case:

$$\begin{aligned} p(x|x') &\propto \int_0^{+\infty} p(r|x)p(r|x')dr \\ &\propto \int_0^{+\infty} N(r; x, \sigma)N(r; x', \sigma)dr \quad . \end{aligned} \quad (3)$$

This solution to this integral can be found in a table. After some algebra one arrives at:

$$p(x|x') \quad \propto \quad N(\frac{x' - x}{\sqrt{2}}; 0, \sigma) \quad . \quad (4)$$

Using Equation 4, one can obtain an estimate for the *expected value* of the PDF X^k_{ij} for the robot's location given its *true* location (i, j) and the pose of beacon k. We now assume independence between each of the beacons and assume that the expected PDF X_{ij} for the robot's location is a product of the PDFs obtained from each beacon:

$$X_{ij} \propto \prod X^k_{ij} \quad . \quad (5)$$

[2]From G one could infer such system statistics such as the mean time between failure *etc.*

The entropy of the PDF obtained as a function of the true location of the robot and the beacon poses is then computed and recorded for that grid square. One obtains an *entropy map* of the environment $Z = \{z_{ij}\}$ where $z_{ij} = H(X_{ij})$.

C. Sample-based path planning

Given the discretized *entropy map* Z of the environment, we are interested in finding a set of paths that connect our operating regions R. Motivated by classical sample-based motion planners such as those described in [14], [15] we uniformly sample N candidate points from our environment. We then assess paths through these points for robot navigation purposes.

Given the N sampled points Γ' and the original operating regions R, we arrive at a set of vertices $V' = \Gamma' \cup R$. We now compute the pair-wise *navigation cost* w_{ij} between each pair i, j of vertices; i.e. we assign a weight to the undirected edge e_{ij}. Our weight calculation takes as input: 1.) the entropy that would be observed in the PDF of the robot if it were able to successfully execute a straight line transit between the vertices i and j and 2.) the Euclidean distance between i and j:

$$w_{ij} \approx \int_{e_{ij}} f(H(x))dx \tag{6}$$

where $H(x)$ returns the value for the grid square in the entropy map Z that contains x. The integral is approximated by sampling values for x at regular intervals that are much smaller than the resolution of a single grid square.

For purposes of illustration we consider and later experiment with two arbitrary cost functions $f(H(x))$. One uses the cube of the entropy observed in the grid square:

$$f_1 = H(x)^3 \tag{7}$$

and the second returns a value proportional to the entropy observed, but penalizes areas of high entropy:

$$f_2 = \begin{cases} H(x), & H(x) < \delta \\ C, & H(x) >= \delta \end{cases} \tag{8}$$

where δ is an appropriately selected threshold and $C \gg \delta$ is some very large constant. See Section VI for experimental results using these cost functions.

Once the pair-wise cost between all vertices in V' is computed, the lowest cost path p_{ij} is found between each pair of operating regions $r_i, r_j \in R$ using Dijkstra's algorithm. The path p_{ij} will begin with operating region i and end with operating region j, but may contain a number of sample waypoints and other operating regions. The final navigation graph G consists of the union of all the edges and vertices in this collection of paths between all operating regions.

D. Summary of Algorithm Inputs and Outputs

Our algorithm for constructing a navigation graph takes the following inputs:

1) a metric map of the region in which the robot will be operating; *e.g.* in occupancy grid format
2) a set of known beacon poses
3) a beacon measurement model
4) a set R of task defined operating regions

The output of the algorithm is a navigation graph G which specifies a lowest entropy path between any two operating regions.

V. System Details

We use a custom experimental platform for our localization experiments. We have built a low-cost explorer robot from off-the-shelf components, and use several acoustic beacons for range-based localization.

A. Acoustic Beacons

The acoustic beacon is a simple, low-cost module built from off-the-shelf components (Figure 2). It functions similarly to the well-known Cricket ultrasonic ranger [16]. We use an Arduino Pro Mini microcontroller, and a Nordic Semiconductor nRF24L01+ 2.4 GHz radio for wireless data transmission. A 40 kHz acoustic pulse is generated by the low-power Devantech SRF04 ultrasonic ranger.

The ultrasonic ranger has two piezoelectric transducers: one used for transmitting an acoustic ping, and the other, ignored by our beacons, for receiving. The module has a digital pulse-width interface; we do not perform any analog processing on the acoustic signal. The ranger's datasheet claims a reflection range of 3 cm to 300 cm. In practice, our beacon's acoustic range is approximately 15 cm to 600 cm. Communication delays and processing delays increase the beacon's minimum range. Also, our system sends a one-way acoustic signal rather than reading a reflection, so its maximum range is double the module's specified range. The ranger's resolution is approximately 3 cm.

Each acoustic transducer is a piezoelectric piston driver with a resonant frequency of 40 kHz and a resonant bandwidth of 10 kHz. The transducers generate a directional pressure wave covering an angle of 45°.

The mobile explorer requests a beacon to send an acoustic ping by sending a message over the 2.4 GHz radio. The radio transmits a 41-byte packet at 2 Mbps over the air. The radio's total over-the-air delay is 164 μs. The receiving radio automatically transmits an acknowledgment to the transmitter to indicate that the packet was received. If a packet is dropped the transmitter automatically attempts to resend it, which increases the communications delay unpredictably. This can be compensated for by transmitting the number of retries and adjusting the receiver's range calculation accordingly. For the moment, we ignore the problem due to the radio's good reliability at low range.

The firmware operating on the microcontroller implements a simple control loop interface between the radio and the ultrasonic ranger. As soon as the beacon receives an echo-request packet, it transmits a echo-confirm packet and an acoustic ping.

Each component on the beacon is powered by a 5 V supply, delivered by a regulator built onto the microcontroller. The energy is provided by a rechargeable battery pack. Our battery pack has very high internal resistance, so we place a 470 μF capacitor in parallel with the battery. This ensures that sufficient power is available when the beacon responds to a range request. We place a reverse-protection Schottky diode in series with the battery.

B. Explorer

The explorer robot moves about the environment, fusing its own odometry with the beacon range data to localize itself. We use an iRobot Roomba vacuum cleaner as a mobile platform. The Roomba has a serial interface over which it can be commanded to drive and over which its sensor data can be read. The Roomba has several useful sensors, including an infrared proximity sensor for traveling along walls, an internal temperature sensor, a bumper sensor, and distance and angle odometry sensors. We only use the latter two for localization, but the others could potentially be used as well [17].

The robot's odometry is calculated using shaft encoders mounted on the Roomba's two-wheel differential drive system to detect how many millimetres each wheel has driven. Due to slippage in the drive belts and low resolution in the shaft encoder, the odometry available to an external controller is imprecise. In particular, the Roomba cannot be driven in a straight line and the angular error reported by the angle sensor is inaccurate by several degrees.

We use a Seeed Studio Seeeduino Mega microcontroller to send commands to the Roomba. The microcontroller is also responsible for reading range measurements by interacting with the acoustic beacons. The explorer is equipped with the same radio and ultrasonic ranger as the acoustic beacons.

The ultrasonic ranger module will not detect an acoustic ping until after it receives a request to transmit one. If the explorer actually transmits an acoustic ping it is impossible to tell if the response it detects is from a beacon or is its own reflected signal. In order to eliminate interference from the explorer's transmitter we removed the transmitting piezoelectric transducer so that the explorer would not produce an acoustic signal before engaging its detector. In addition, we modified the ranger by positioning a Styrofoam sphere 4 cm above the receiving transducer. The sphere reflects acoustic pings towards the receiver, enabling the explorer to receive signals from beacons in any direction.

High-level robot control is provided by a netbook mounted on top of the Roomba. The netbook runs a script that reads the position information (*i.e.* odometry and beacon ranges), processes the data with a probabilistic filter, and outputs a movement command back to the microcontroller. Our script offers options to use a Kalman filter or a particle filter to estimate the robot's position.

The explorer's motion is modeled only for a few discrete movements. The robot is able to move forward by 10, 50 or 100 cm, and to rotate clockwise or counterclockwise by $5°$.

Figure 3 shows the sequence of messages that are passed between the netbook and explorer over the serial port, and between the explorer and the beacons during a movement and update routine. The netbook sends a movement command

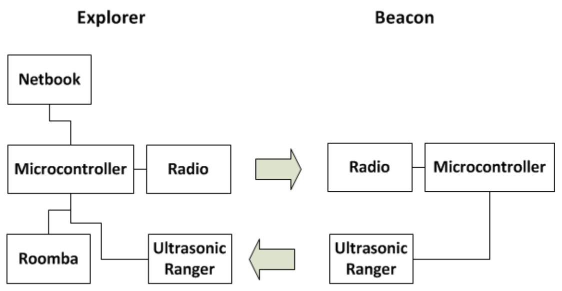


Fig. 2. System block diagram.

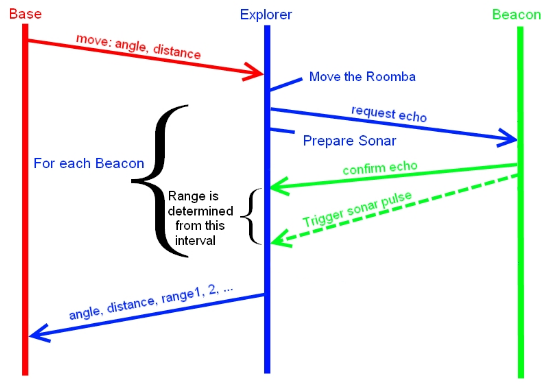


Fig. 3. System message passing diagram.

to the explorer, which executes the movement. When the movement completes the explorer queries each beacon in turn. During a query, the explorer sends a radio message to the beacon and the beacon transmits a response. As soon as the transmitter receives the radio acknowledgment it sends its acoustic ping. The explorer measures its range to the beacon by time difference of arrival between the radio message and the acoustic ping. Once the explorer has queried each beacon in its range, it packages up the range data and its odometry readings and sends them to the netbook, which fuses the new data and produces a new movement command.

C. Data Fusion

We have programmed our netbook to fuse data with Python implementations of a Kalman filter or a particle filter. Given a set of pre-defined waypoints, the control software used by the robot executes periodic course corrections based on its current state estimate; *i.e.* the filtering information is used for real-time navigation. In this section we will provide a brief summary of these probabilistic techniques.

1) The Kalman Filter: The Kalman filter [18] [19] maintains a single estimate of the robot's pose at time $t-1$ represented as an expected value X_{t-1} and a covariance matrix P_{t-1}. It operates in two phases: the prediction phase and the update phase. Given an input control vector u_{t-1}, the prediction phase uses the robot's motion model to predict where the robot will be at the end of the resulting movement, generating a new pose estimate with mean $\hat{X}'_t$ and covariance $\hat{P}'_t$.

During the update phase, the pose estimate calculated in the prediction phase is refined using external sensor data, *e.g.* odometry and beacon range data. First the algorithm calculates a Kalman gain matrix K_t for a particular sensor using the sensor's measurement model H_t and $\hat{P}'_t$. The gain matrix gives more weight to reliable sensors in the update calculations than to unreliable sensors. For each sensor, the update phase refines its estimated pose $\hat{X}'_t$ and covariance $\hat{P}'_t$. After the last sensor, it produces a final estimate X_t and covariance P_t that are used at the next iteration of the filter.

Converting a beacon's range covariance from polar coordinates to Cartesian coordinates is quite difficult, particularly in the case where a beacon produces an omnidirectional signal. Our beacons are directional, but the angle variance can still be spatially large. We use the simplification suggested by Kantor and Singh in [20] to handle this problem. To maintain the angle dimension in our robot's pose we use the technique described by Smith and Cheeseman in [21].

2) The Particle Filter: The particle filter [22], [23] operates similarly to the Kalman filter, but with fewer restrictions: the original Kalman filter is limited to a Gaussian motion and measurement model and only maintains a single mode in the pose estimate.[3] The particle filter allows arbitrary motion and measurement models and maintains many estimates of the robot's pose in the form of particles.

One benefit of using the particle filter is that the complications of maintaining the robot's orientation estimate seen in the Kalman filter do not apply. It does not use a covariance matrix. For each particle, the beacons' polar data are converted directly to Cartesian coordinates. The cost of using the particle filter is that the algorithm is linear in the number of particles in the cloud, and for useful numbers of particles it runs quite slowly. For this reason, our experiments focus on the Kalman filter.

VI. Results

A. Simulation Results

We have implemented a simulation framework to evaluate our navigation graph algorithm. We model the beacons as angular sections radiating from the beacon location with fixed range. Figures 4, 5, and 6 show the result of one simulation result from a trial with 10 randomly placed beacons and 10 randomly placed operating regions. Beacons are represented as blue stars, navigation locations are black crosses, and operating regions are red circles. This result was obtained in less than 15 minutes using an I7 CPU with 4 cores and 6 GB of RAM.

The simulations shown in Figures 7 and 8 model a real section of hallway, placing along the walls four beacons with set positions and orientations. The walls are represented as black regions. We only consider an idealized system with a rough-cut model of the beacon's beam shape and acoustic behaviour (*i.e.* we ignore reflection and obstruction).

[3]Modern versions of the Kalman filter, such as the Extended Kalman filter, allow non-Gaussian models.

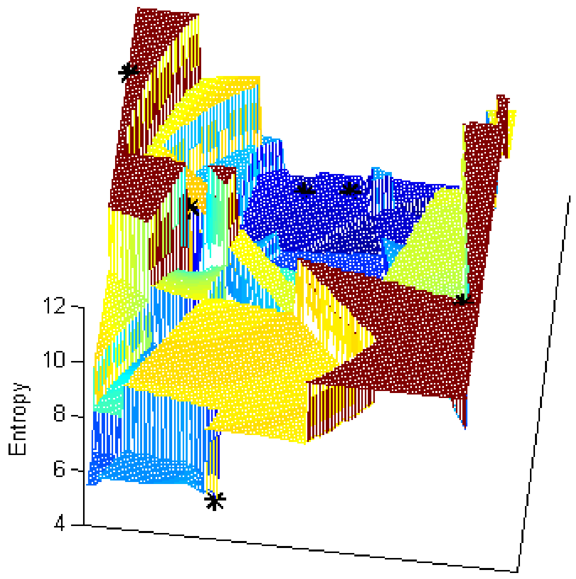


Fig. 4. Estimated entropy throughout the region in Figure 5; black stars depict beacon locations.

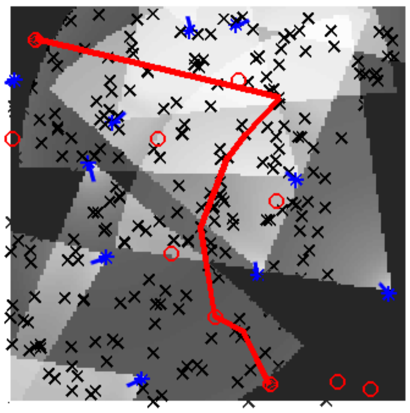


Fig. 5. A path of minimum entropy in a random beacon field with no obstructions; higher entropy is represented by a darker shade.

The simulation results show that the cost function from Equation 7 produces a path that follows the entropy contours closely, whereas the cost function from Equation 8 produces longer straight path segments. This is expected, as the former function rewards low-entropy paths more than the latter; however, the later cost function is more suited to the typical requirements of a robot with poor dead reckoning. The limitation of a few discrete movements means it is easier to successfully execute long, straight movements compared to a larger number of short movements as seen in Figure 7.

Note that in our simulation results the robot aims for low-entropy regions and avoids high-entropy regions. In each example presented, the robot is able to traverse a region that is covered by at least three beacons.

As the number of random navigation locations used to

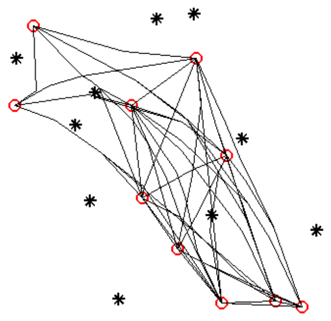


Fig. 6. Full navigation graph for the situation in Figure 5; black stars depict beacons, red circles operating regions, and black lines low entropy paths.

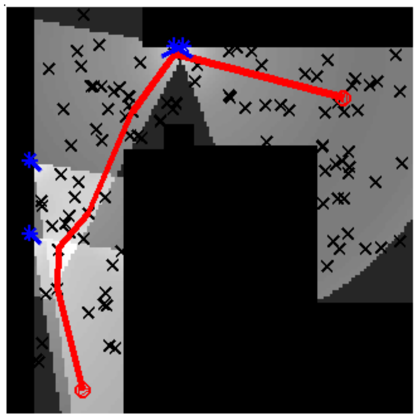


Fig. 7. A path of minimum entropy, with entropy variance cubed at all points (Equation 7).

calculate the navigation graph increases, the mean cost of the minimum-entropy path decreases. It is intuitive that more navigation locations will give the algorithm more options Figure 9 shows the mean costs for paths generated with various numbers of samples, for the situation shown in Figures 7 (left) and 8 (right). The entropy values along the best paths in the two plots in Figure 9 should not be compared numerically as they are calculated using different cost functions.

B. Results from Hardware

Experimentation using our navigation system demonstrates the successful fusion of range data obtained from the beacons into our state estimation system. Due to the poor quality of the odometry data available on our mobile platform, periodic range data is essential to continuous mobile operation; without the range data, the error in the position estimate grows to the point that the robot can not successfully navigate a standard office environment. Within a laboratory of 60 square meters instrumented with 4 beacons, the robot can successfully execute predefined sequences of moves and avoid major obstacles.

We programmed the robot to travel to five operating regions within a 2 m by 3 m area, as a first-order demonstration of the beacons' effectiveness (Figure 10). Results using the Kalman filter are shown in Table I. The experiment included many rotations, exploiting a weakness in our robot's motion

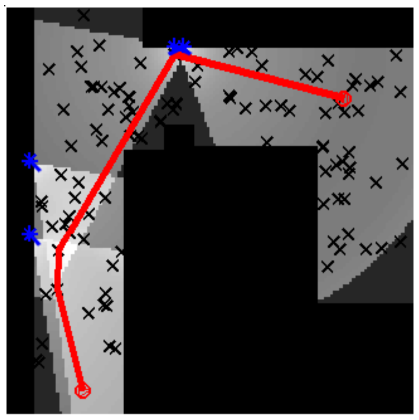


Fig. 8. A path of minimum entropy, with entropy variance cubed only in unseen regions (Equation 8).

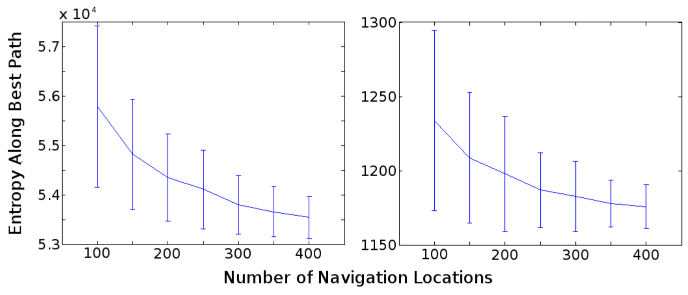


Fig. 9. Mean entropy and standard deviation for random paths at various numbers of samples, using the cost function in Equations 7 (left) and 8 (right).

model to increase the error for each subsequent destination. The large means in positions 4 and 5 observed when using the beacons are caused by large outliers resulting from software bugs; removing these outliers results in a mean error of 9 cm and 8 cm respectively.

Preliminary experiments testing low entropy routes along a navigation graph computed for our indoor test environment have produced mixed results. Often, the final error in the robot's position after executing a low entropy path between two points is slightly worse than a more direct route. An experiment showed a mean error of 33.6 cm (standard deviation 6.2 cm) for a control route with four operating regions and a mean error of 46.4 cm (standard deviation 5.9 cm) for a low-entropy route. This is a result of the typically larger number of operating regions being traversed in a low entropy route, each of which introduces positional error. For example, in the results reported above the control route only demanded

TABLE I
TOTAL ERROR IN CM FOR THE ROBOT WITH AND WITHOUT BEACONS.

	Error (No Beacons)		Error (With Beacons)	
	Mean	Std. Dev.	Mean	Std. Dev.
Pos. 1	14.8	7.8	3.5	5.2
Pos. 2	15.2	8.6	7.3	2.8
Pos. 3	18.0	7.5	7.0	1.6
Pos. 4	24.0	8.8	20.0	22.9
Pos. 5	28.8	9.3	13.3	10.5

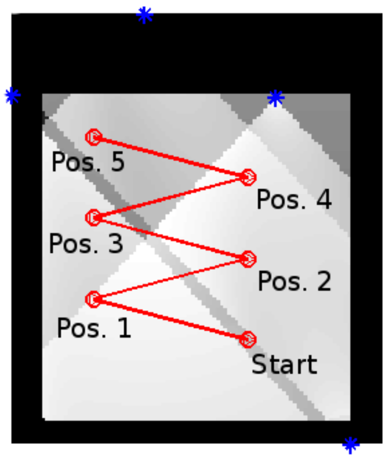


Fig. 10. Operating regions and programmed path for the beacon test summarized in Table I.

three segments whereas the low-entropy route used six. Our motion model was sufficiently accurate for the robot to traverse the high-entropy area successfully. Further testing in a larger, more complex environment using a larger number of beacons is required to expose flaws in the motion model and to further test the merit of the low entropy routing.

VII. Conclusion

In this paper we have considered navigation issues appropriate for a robot carrying out routine tasks and operating in an environment instrumented with range-only beacons. We have presented a pragmatic sample-based path planning approach that can help find low-entropy routes among pre-defined regions in the environment. Additionally, we have described an example of a functional system that exercises these concepts in practice using affordable components and custom state-estimation software implementing modern Bayesian filtering approaches.

In future work on our navigation system, we will implement the unscented Kalman filter [24] which should be better suited to handling the non-linearities of range data and yet retain a computational advantage over the particle filter. Additionally, we will continue to evaluate the navigation graph concept with further experimental work.

Finally, we plan to use simple (and temporally local) environmental parameters such as wireless signal strength at various frequencies, temperature, and ambient light levels to extract features from the environment and use them to augment our current state estimation approach. This will require incorporating algorithms that use environmental parameters for localizing purposes [17] into our current framework.

VIII. Acknowledgments

We thank Sean Chester and Rahnuma Islam Nishat for their comments and help performing experiments. We thank Mantis Cheng for technical assistance and lab space. We acknowledge the Natural Sciences and Engineering Research Council of Canada for their funding.

References

[1] P. R. Wurman, R. D'Andrea, and M. Mountz, "Coordinating hundreds of cooperative, autonomous vehicles in warehouses," in *AAAI'07: Proc. of the 22nd National Conf. on Artificial intelligence*. Vancouver, British Columbia, Canada: AAAI Press, 2007.

[2] N. Roy and S. Thrun, "Coastal navigation with mobile robots," *Advances in Neural Processing Systems*, vol. 12, no. 12, pp. 1043–1049, 1999.

[3] D. Meger, D. Marinakis, I. Rekleitis, and G. Dudek, "Inferring a probability distribution function for the pose of a sensor network using a mobile robot," in *Proc. of ICRA*, Kobe, Japan, May 2009.

[4] D. Marinakis, D. Meger, I. Rekleitis, and G. Dudek, "Hybrid inference for sensor network localization using a mobile robot," in *AAAI'07: Proc. of the 22nd National Conf. on Artificial intelligence*. AAAI Press, 2007, pp. 1089–1094.

[5] D. Jourdan and N. Roy, "Optimal sensor placement for agent localization," *ACM Trans. on Sensor Networks*, vol. 4, no. 3, August 2008.

[6] S. Martinez and F. Bullo, "Optimal sensor placement and motion coordination for target tracking," *Automatica*, vol. 42, no. 4, pp. 661–668, 2006.

[7] A. Savvides, W. Garber, S. Adlakha, R. Moses, and M. B. Srivastav, "On the error characteristics of multihop node localization in ad-hoc sensor networks," in *Proc. of IPSN'03*, 2003.

[8] D. B. Jourdan, D. Dardari, and M. Z. Win, "Position error bound for UWB localization in dense cluttered environments," in *Proc. of IEEE Int. Conf. on Communications*, Instanbul, Turkey, June 2006, pp. 3705–3710.

[9] M. A. Batalin, G. S. Sukhatme, and M. Hattig, "Mobile robot navigation using a sensor network," in *Proc. of ICRA*, New Orleans, April 2004, pp. 636–642.

[10] A. Verma, H. Sawant, and J. Tan, "Selection and navigation of mobile sensor nodes using a sensor network," *Pervasive and Mobile Computing*, vol. 2, no. 1, pp. 65–84, February 2006.

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[12] S. M. LaValle, *Planning algorithms*. Cambridge: Cambridge University Press, 2006. [Online]. Available: http://planning.cs.uiuc.edu

[13] M. L. Fredman and R. E. Tarjan, "Fibonacci heaps and their uses in improved network optimization algorithms," *J. of the ACM*, vol. 34, no. 3, pp. 596–615, July 1987.

[14] B. Siciliano and O. Khatib, Eds., *Springer Handbook of Robotics*. Springer-Verlag, 2008.

[15] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach (2nd Edition)*. Prentice Hall, 2002.

[16] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan, "The cricket location-support system," in *Proc. of the 6th Annual Int. Conf. on Mobile Computing and Networking*. New York, New York, USA: ACM, 2000, pp. 32–43.

[17] D. Marinakis, N. MacMillan, R. Allen, and S. Whitesides, "Simultaneous localization and environmental mapping with a sensor network," in *Proc. of ICRA*, Shanghai, China, May 2011.

[18] R. E. Kalman, "A new approach to linear filtering and prediction problems," *J. of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[19] G. Welch and G. Bishop, "An introduction to the Kalman filter," Chapel Hill, NC, USA, Tech. Rep., 2006.

[20] G. Kantor and S. Singh, "Preliminary results in range-only localization and mapping," in *Proc. of the IEEE Int. Conference on Robotics and Automation 2002*, vol. 2. IEEE, 2002, pp. 1818–1823.

[21] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *The Int. J. of Robotics Research*, vol. 5, no. 4, p. 56, 1986.

[22] A. H. Jazwinski, *Stochastic processes and filtering theory, Volume 63*. Academic Press, 1970.

[23] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE Int. Conf. on Robotics and Automation (ICRA99)*, May 1999.

[24] E. A. Wan and R. V. D. Merwe, "The unscented kalman filter for nonlinear estimation," in *Proc. of IEEE Symposium 2000 (AS-SPCC)*, 2000, pp. 153–158.