Robot 2D Self-Localization Using Range Pattern Matching via the Discrete Fourier Transform

Andrew Willis
Department of Electrical and Computer Engineering
University of North Carolina at Charlotte
Charlotte, NC 28223
Email: arwillis@uncc.edu

Yunfeng Sui
Department of Electrical and Computer Engineering
University of North Carolina at Charlotte
Charlotte, NC 28223
Email: ysui@uncc.edu

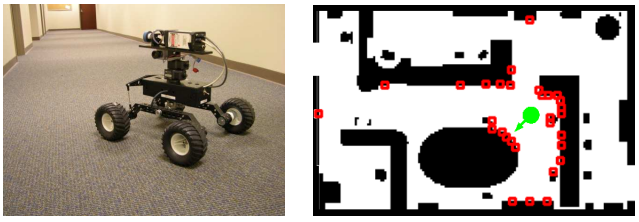
Abstract*—This article describes a novel method for localization of a robot within a 2D scene given as a binary map of the scene and a set of range measurements obtained by the robot from some unknown position and orientation. Theoretically, the algorithm is capable of solving all recognized variants of the robot localization problem: tracking, global localization, and kidnapped robot. This is accomplished by treating each set of range measurements as a unique fingerprint, referred to as a range pattern, that associated with each potential (x, y, θ) pose of the robot. We provide detailed theoretical analysis and an exact solution for the problem when both the range and angle measurements are constrained to come from a discrete set of possible values. Experimental results are obtained using simulated range data taken from synthetic and real-world maps to provide insight on the robustness of our approach and identify situations where the localization solution obtained is not unique. Our solution to this more-constrained problem has low computational complexity and exact solution which makes it appropriate for use in real-time robotic navigation applications. Solutions to this problem are of great importance for successful deployment of autonomous robotic vehicles within a-priori known spaces, e.g., buildings, hospitals, etc.

I. Introduction

Localization is a problem encountered commonly in robotics where the goal is to determine the pose, i.e., the position and orientation, of a robot within its environment (see Figure 1). Note that this problem statement is distinct from the well-known SLAM (Simultaneous Localization and Mapping) problem since we assume that a map of the environment within which the robot is placed is *a-priori* known. Yet, this more-constrained problem has received much consideration in research as it has been deemed of critical importance for successfully using of mobile robot systems [1].

There are several different versions of the localization problem which include increasingly difficult pose estimation problems [2]. These include in *position tracking* problem [3], [4], where the robot's initial pose is assumed to be known and updates for the position are needed as the robot moves through the environment, *global localization* problem, where the robot's initial pose is unknown [5], [6], and finally the *kidnapped robot* problem, where the robot may be instantaneously picked up and moved to another location in the map at any point in time [7], [8].

Most popular approaches to this problem use advanced filtering techniques such as the Kalman filter [9], [10] or a particle filter [11]. These approaches are particularly useful for *tracking* problems where the robot location can be known to some extent and then incrementally updated as the robot moves using the chosen filter. However, these methods cannot effectively treat the problem of global localization since the incremental updates require an initial pose as a starting point. Methods have been devised that solve the *global localization* problems using multi-hypothesis Kalman filters [9], [10] and Monte-Carlo methods [11] that seek to localize the robot's position using a sequence of measurements from different poses. The common thread of these approaches is to store a collection of poses as candidate solutions to the pose problem which may initially consist of a grid that covers all plausible locations for the robot in the environment. The candidate solutions are then iteratively reduced as new sets of measurement data are made available due to robot motion and sensor readings. In [9], [10], multiple candidate initial positions are tracked until only one position is plausible given the map. In [11], a stochastic selection model (the Monte-Carlo method) is used to select candidate solutions of high-probability more often until the selection converges to only one (the most likely) position. The Monte-Carlo localization method has also been shown to solve *kidnapped robot* problems [11], [12] and represents some of the most advanced techniques for solving this problem to date.

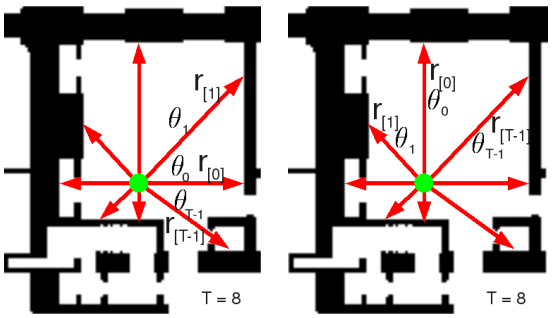


(a) (b)

Figure 1: We propose a method to localize the pose of a robot (a) within an known environment represented by a binary map (b). The unknown robot pose (position and orientation - indicated in (b) by a green circle), must be computed from a sequence of range measurements taken by a 360° range scanner mount on the robot (shown as red squares). Our approach solves this problem directly and typically does not require and initial position information or tracking data (exceptions occur for very simple geometries – see §IV).

II. Range Patterns as a Map Representation

Our approach models the 2D surfaces visible from each (x, y) location via an ordered sequence of range values referred to as a *range pattern*, $r[t] = \begin{bmatrix} r[0] & r[1] & \dots & r[T-1] \end{bmatrix}^t$, that corresponds to a sequence distance readings between the (x, y) position



(a) (b)

Figure 2: (a,b) Show two potential measurement sets taken from a single location (in green) within a binary map. Red arrows indicate the 8 different angles where range measurements are taken to generate a range-pattern associated with the (x, y) position. (a) shows how range patterns are generated for the database, (b) shows a hypothetical range pattern collected by a robot having an unknown orientation. The range patterns from (a) and (b) will differ by an unknown circular-shift in the sequence of observed range values within the range pattern.

of consideration and surfaces in the environment *within line-of-sight of the that point*. For the purposes of discussion, we assume that the robot collects range measurements from the environment using a 360 degree Light Distance and Ranging (LIDAR) scanner which are now available from a number of vendors at reasonable expense (>$6K USD). These scanners produce a sequence of range measurements taken at constant intervals of angle that record the distance between the robot and the surfaces within the robot's line-of-sight that exist in the environment (and map).

We derive our representation assuming that the measurement resolution is infinitesimal, in which case the sequence of range measurements at some (x, y) position is an explicit function $r = f(\theta)$ where r, represents the range of surfaces from the robot and θ is measurement angle;$\theta \in [0, 2\pi)$. Since each (x, y) position for the scanner may generate a unique set of range values, we can represent the map as a union of range scans taken from all possible (x, y) positions which we refer to with the function $V(x, y, r, \theta)$.

The function $V(x, y, r, \theta)$ is an *idealized continuous representation* of the environment, however, we seek to store this representation in a computer for the purposes of localization. Hence, we descretize the independent variables of our representation above. Since the range, r, is an explicit function of angle, it may be discarded to form the function $V(x, y, f(\theta))$ which we then descretize as described in the following two sections.

A. Descretizing the angle

Let θ_t denote the discrete angles associated with the LIDAR sensor measurements. Assuming N measurements per scan, the angles of the measurements are given by $\theta_t = \frac{2\pi t}{T}$ where $t = 0, 1, \ldots, T-1$ and the continuous function $r = f(\theta)$ becomes a discrete function $r[t] = f(\theta)|_{\theta=\theta_t}$. Note that the function $r = f(\theta)$ is 2π-periodic and hence the function $r[t]$ is also periodic with period T. We refer to the sequence of range measurements made at a given (x, y) locations as a *range pattern*.

B. Descretizing the position

Assuming that the provided map covers a region bounded by the rectangle with upper right hand corner (x_1, y_1) and lower left hand corner (x_0, y_0) we continue by descretizing the (x, y) position of the robot into N equally-spaced locations in along the $x - axis$ and M equally-spaced locations along the $y - axis$. Descretization of both the angle,θ, and the (x, y) position generated a discrete version of the map $V(x_n, y_m, r[t])$ where $x_n = \frac{x_1 - x_0}{M-1}n + x_0$, $y_m = \frac{y_1 - y_0}{N-1}m + y_0$, $n = 0, 1, \ldots, N-1$ and $m = 0, 1, \ldots, M-1$. This generates a discrete function which we may index by n, m, and t: $V(n, m, r[t])$ where n and m specify a map grid position, $r[t]$ is a set of range values at that position and t indexes a specific range measurement made at some angular position, θ_t.

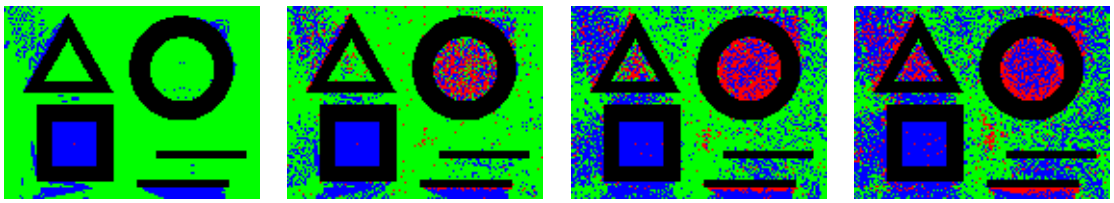
C. Localization with this representation

Using this representation, we seek to perform localization directly from range measurements obtained from an arbitrary (x, y) position and unknown θ orientation, we refer to the unknown (x, y, θ) values as the *query*. Our tabulation of range images $r[t]$ for each grid location generates a database of MN entries where each entry contains T values given by $r[t]$, the range pattern measured at that position (see Figure 2). Localization of the robot's pose is then reduced to a database search problem where we must search through all possible range patterns to find the pattern indicative of our query, the (m, n) index of our database range pattern is indicative of the unknown (x, y) position of our query and the unknown orientation θ can be estimated as a phase difference between the query range pattern and the database range pattern. Hence, the localization problem is reduced to a database query problem where we select a single record and use this record to estimate the unknown (x, y, θ) pose of the robot directly.

Finding the best result for the query constitutes a search over each location, i.e., comparisons with each of the MN positions on the grid, and finally a comparison for all T shifted versions of the *range pattern* that may occur due to the unknown θ orientation of the robot within the environment. This search has computational complexity $O(NMT)$ (or $O(N^3)$ for simplicity) in time which is prohibitively expensive for even modestly sized maps. Fortunately, much of this computation may be avoided by noting the following mathematical properties associated with the *range patterns*:

1) Shifted versions of the *range patterns* will share the same magnitude response under the Discrete Fourier Transform (DFT).
2) *Hashing functions* defined on the *range pattern* magnitude response serve to generate a look-up-table that *greatly* reduces the amount of searching required to find the corresponding range image.

Use of (1) above allows us to directly compare *range patterns* using the DFT. Let $r_i[t]$ denote the range pattern in the database that corresponds to the query range pattern, $r_q[t]$; yet, the two signals are recorded at different initial orientations. Assuming the orientation of both robots is some value ofθ_t, this difference in the orientation of the robot generates a phase-shift between the recorded range samples such that $r_q[t - t_0] = r_i[t]$ (see Figure 2).



(a) (b) (c) (d)

Figure 3: A simulation was run on a synthetic map where range patterns consisted of measurements taken at 32 different angles ($T = 32$). Our simulation places the robot at each (x, y) pixel location and generates a simulated scan assuming a random unknown value for the orientation θ and subsequently localizes the unknown(x, y, θ) pose using the proposed algorithm. (a-d) Show localization results for each (x, y) position where each range measurement has each been perturbed by zero-mean Gaussian white noise with increasing noise variance that starts at 0% (a), 3.3%(b), 6.6%(c), and 10%(d). Green pixels indicate positions where the solution is both unique and correct. Blue pixels indicate positions where the solution is not unique but one of the available solutions is the correct solution. Red pixels indicate positions where the correct solution was not in the set of candidate solutions.

One key computational-saver is that, for comparing two range patterns, $r_q[t-t_0]$ and $r_i[t]$, we may simply compare the magnitude response of the two signals instead of comparing all shifted versions of the two signals. In the following discussion we provide the mathematical proof for this result. Using the DFT shifting theorem, the shifted version of the query range pattern has DFT $e^{-j\frac{2\pi k t_0}{T}} R_q[k]$ and the range pattern in the database has DFT $R_i[k]$. Hence under the DFT we have

$$e^{-j\frac{2\pi k t_0}{T}} R_q[k] = R_i[k] \tag{1}$$

Taking the magnitude of both sides we get $|R_q[k]| = |R_i[k]|$ which motivates the following difference function between two range patterns:

$$e(r_q, r_i) = \sum_{k=0}^{T-1} \left| |R_q[k]| - |R_i[k]| \right|^2 \tag{2}$$

. This function allows us to estimate the position of the query range pattern by summing the squared differences between the *query* and each database entry and results in a savings of T comparisons providing a $O(NM)$ (or $O(N^2)$ for simplicity) algorithm. Localization is then accomplished by applying equation (2) to compute the (x, y) position by matching range patterns and subsequently applying equation (1) to compute the orientation, θ (see §III for details).

We can further reduce computation by hashing the results using a hashing function $h(r[t])$. For our purposes, a hashing function converts a large amount of data into a single measurement where the value of the individual measurements are intended to be as unique as possible. Such functions are commonly used to define hash table data-structures and serve to provide a method to quickly index a specific value within a large group of potential values. While the specific hash function may be arbitrary, we have found that the sum of the magnitudes is sufficiently unique to reduce the computation to nearly $O(BT + 1)$ where B is the expected number of elements associated with a hash table key (see §IV for details). For our implementation the hash function is defined: $h(r[t]) = \frac{1}{T}\sum_{t=0}^{T-1} r[t]$.

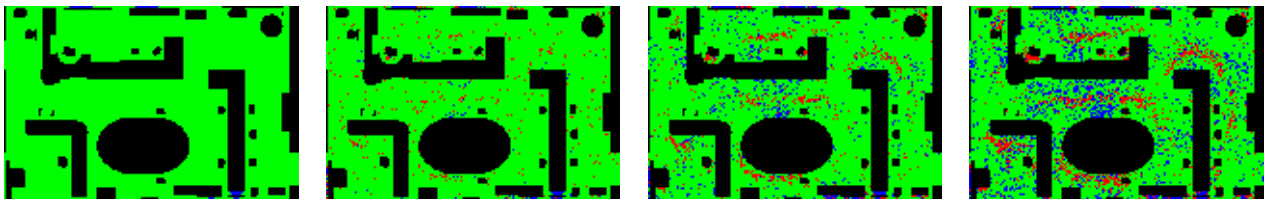
III. Methodology

Implementation of our algorithm proceeds as indicated by the following four steps:

1) For each (x, y) position within the binary map having dimensions $M \times N$ (M rows and N columns), compute a range pattern, $\mathbf{r}_i = \begin{bmatrix} r[0] & r[1] & \dots & r[T-1] \end{bmatrix}^t$, having index $i = yN + x$ that contains a sequence of T radius values recorded at a discrete set of angles: $\theta_t = \frac{2\pi}{T}\ t = 0, 1, \dots, T-1$.
2) We are given a set of measurements $\widetilde{\mathbf{r}} = \begin{bmatrix} r[0] & r[1] & \dots & r[T-1] \end{bmatrix}^t$ obtained from an unknown position $(\widetilde{x}, \widetilde{y})$ and unknown orientation $\widetilde{\theta}$.
3) Estimate position by finding the range pattern index, $\widehat{i}$, that minimizes our difference function:
$$\widehat{i} = \min_i \sum_{k=0}^{T-1} \left\| |R_i[k]| - \left|\widetilde{R}[k]\right| \right\|^2$$
where $|R_i[k]|$ is the magnitude of the DFT coefficient k for vector $\mathbf{r}_i$ and $\left|\widetilde{R}[k]\right|$ is the magnitude of the DFT coefficient k for vector $\widetilde{\mathbf{r}}$. Then the position is $\widetilde{y} = \left\lfloor \frac{\widehat{i}-x}{N} \right\rfloor$ and $\widetilde{x} = \widehat{i} - \widetilde{y}N$.
4) The orientation index can be found using the phase shift DFT property: $r[t - t_0] \Longleftrightarrow e^{-j\frac{2\pi t_0}{T}k} R[k]$. We assume that one signal is a circular-shifted version of the other then $r_{\widehat{i}}[t] = \widetilde{r}[t - t_0]$ where t_0 is the unknown shift that we seek to recover. Using the DFT property, we can compute t_0 as follows:
$$t_0 = -j\frac{N}{2\pi k} \ln\left(\frac{R_{\widehat{i}}[k]}{\widetilde{R}[k]}\right)$$
for any non-zero value of k, i.e., $k = 1, \dots, T-1$ and non-zero value of $\widetilde{R}[k]$. We can then complete our localization computation for the unknown orientation: $\widetilde{\theta} = \frac{2\pi}{T} t_0$.

IV. Results

Figure 3 shows localization results for a synthetic image created for the purposes of development. Localization was



(a) (b) (c) (d)

Figure 4: Experiments are run on a real-world floor plan where range patterns consist of range measurements at 32 different angles ($T = 32$). In each figure, we show the localization estimate result for each (x, y) position in the image. As in Figure (3), localization results are color coded: green=correct, blue=correct but not unique, red=incorrect. Zero-mean Gaussian noise with increasing variance is added to simulated range data for images from left-to-right (0%-(a), 3.3%-(b), 6.6%-(c), 10%-(d). Note that the proposed method shows significant resistance to the additive noise.

performed using range values taken at 11.25° increments (a-d). Analysis of these results show that in some regions of the map (those labeled in blue) the range pattern observed at a single location is not sufficient to uniquely localize the exact position and orientation of the robot. This is occurs when multiple (x, y) positions generate the *exact same* sequence of range values when no noise is present as in Figure 3(a) or *approximately the same* range values when noise is present as in Figure 3(b,c,d). Since these individual range patterns are indistinguishable one must use more than a single range pattern, e.g., two range patterns taken as the robot moves, to uniquely identify the pose of the robot. Due to the simplistic shapes in the synthetic image, many more ambiguous points are present in Figure 3 than would exist in a typical robotic environment.

Figure 4 shows results for the same experiment using a more realistic floor plan map. In this case there are more irregularities in the shape and orientation of the surfaces in the environment which makes the range patterns at each location more distinctive. Hence the localization results are much better than those shown for the synthetic map and include very few non-unique and incorrect localization results. Note that incorrect results are any results that are not *exactly correct* when estimating the unknown pose of the robot.

The computational cost of our localization algorithm is based on the time needed to index into the database to find range range-patterns that correspond to the query pattern. Using the computational accelerations detailed in §II, we computed the value of $O(BT)$ for the range images shown in Figure 4. The expected number of computations using 5 hash keys was $O(BT) = 2245 * 32 \approx 2^{16}$ and for 10 hash keys $O(BT) = 1390 * 32 \approx 2^{15}$ where the error increases modestly as the number of hash keys increases. Note that this includes all arithmetic operations which indicates that the proposed algorithm can be realistically implemented on embedded systems with modest computational resources.

V. Conclusion

We propose a concise algorithm for the localization of a robot within a 2D map. Theoretically, the algorithm is capable of solving all versions of the stated robot localization problems (tracking, global localization, and kidnapped robot – see §I) by treating each range pattern as a unique fingerprint associated with an (x, y, θ) pose of the robot. More concisely, our solution is a significant improvement over existing schemes that are iterative such as Kalman tracking, particle filtering, and Monte Carlo localization as it provides a direct solution to the kidnapped robot problem that uses only a single set of range measurements. In practice, we envision that more range patterns may be needed for noisy range data and in environments that include only simple geometries. Our algorithm has very low computational complexity which makes it appropriate for use in real time systems. Use of such a localization scheme is contingent on the availability of a 360° 2D line-scanner which have recently become available for use by researchers. This can partially explain why such methods have not previously been proposed.

References

[1] I. J. Cox, "Blanche – an experiment in guidance and navigation of an autonomous robot vehicle," *IEEE Trans. on Robotics and Automation*, vol. 7, no. 2, pp. 193–204, 1991.

[2] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust monte carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[3] B. E. J. Borenstein and L. Feng, *Navigating Mobile Robots: Systems and Techniques*. Wellesley, MA: A. K. Peters, Ltd., 1996.

[4] D. H. W. Burgard, D. Fox and T. Schmidt, "Estimating the absolute position of a mobile robot using position probability grids," in *In Proceedings of the Thirteenth National Conference on Artificial Intelligence*, vol. 2, pp. 896–901, 1996.

[5] D. F. W. Burgard, A. Derr and A. Cremers, "Integrating global position estimation and position tracking for mobile robots: The dynamic markov localization approach," in *In Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS' 98)*, vol. 2, pp. 730–735, 1998.

[6] P. Jensfelt and S. Kristensen, "Active global localisation for a mobile robot using multiple hypothesis tracking," in *In Proceedings of the IJCAI Workshop on Reasoning with Uncertainty in Robot Navigation*, (Stockholm, Sweden), pp. 13–22, 1999.

[7] W. B. D. Fox and S. Thrun, "Markov localization for mobile robots in dynamic environments," *Journal of Artificial Intelligence Research*, vol. 11, pp. 391–427, 1999.

[8] S. Engelson and D. McDermott, "Error correction in mobile robot map learning," in *IEEE International Conference on Robotics and Automation*, (Nice, France), pp. 2555–2560, May 1992.

[9] J.-S. Gutmann and C. Schlegel, "Amos: Comparison of scan matching approaches for self-localization in indoor environments," in *In Proceedings of the 1st Euromicro Workshop on Advanced Mobile Robots*, pp. 61–67, 1996.

[10] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Autonomous Robots*, vol. 4, pp. 333–349, 1997.

[11] F. Dallaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 1322–1328, 1999.

[12] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte carlo localization: Efficient position estimation for mobile robots," in *Proc. of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)*, pp. 1–7, 1999.