Stochastic Map Merging in Rescue Environments

Stefano Carpin and Andreas Birk

School of Engineering and Science,
International University of Bremen – Germany
{s.carpin, a.birk}@iu-bremen.de

Abstract. We address the problem of merging multiple noisy maps in the rescue environment. The problem is tackled by performing a stochastic search in the space of possible map transformations, i.e. rotations and translations. The proposed technique, which performs a time variant Gaussian random walk, turns out to be a generalization of other search techniques like hill-climbing or simulated annealing. Numerical examples of its performance while merging partial maps built by our rescue robots are provided.

1 Introduction

One of the main tasks to be carried out by robots engaged in a rescue scenario is to produce useful maps to be used by human operators. Among the characteristics of such environment is the lack of a well defined structure, because of collapsed parts and debris. Robots are supposed to move in uneven surfaces and to face significant skidding while operating. It follows that maps generated using odometric information and cheap proximity range sensors turn out to be very inaccurate. In the robotic systems we have developed this aspect is even more emphasized by our choice to implement simple mapping algorithms for their real-time execution on devices with possible low computational power [1],[2]. One of the possible ways to overcome this problem is to use multiple robots to map the same environment. The multi-robot approach has some well known advantages in itself, most notably robustness [3]. In the rescue framework, multi-robot systems are even more appealing because of the possibility to perform a faster exploration of the inspected area, thus increasing the chances to quickly locate victims and hazards. As the goal is to gather as much information as possible, it is evident that the maps produced by different robots will only partially overlap, as they are likely to spread around in different regions and not to stick together for the whole mission. It is then a practical issue of enormous importance to merge together such partially overlapping maps before they are used by the human operators. To solve the map matching problem we have borrowed some ideas from a recent randomized motion planning algorithm recently developed by one of the authors which have turned out to work very efficiently [4],[5]. The algorithm performs a Gaussian random walk, but its novel aspect is that it updates its distribution parameters so that it can take advantage from its recent

D. Nardi et al. (Eds.): RoboCup 2004, LNAI 3276, pp. 483–490, 2005.

history. Section 2 formally defines the problem and describes the algorithmic machinery used to solve it, together with convergence results. Next, section 3 offers details about the implementation of the proposed technique and numerical results. A final discussion is presented in section 4.

2 Theoretical Foundations

We start with the formal definition of a map.

Definition 1. *Let N and M be two positive real numbers. An $N \times M$ map is a function*

$$m : [0, N] \times [0, M] \rightarrow \mathbb{R}.$$

We furthermore denote with $I_{N \times M}$ the set of $N \times M$ maps. Finally, for each map, a point from its domain is declared to be the reference point*. The reference point of map m will be indicated as $R(m)$.*

The function m is a model of the beliefs encoded in the map. For example, one could assume that a positive value of $m(x, y)$ is the belief that the point (x, y) in the map is free, while a negative value indicates the opposite. Moreover, the absolute value indicates the degree of belief. The important point is that we assume that if $m(x, y) = 0$ no information is available. From now on, for sake of simplicity, we will assume $N = M$, but the whole approach holds also for $N \neq M$.

Definition 2. *Let x,y and θ be three real numbers and $m_1 \in I_{N \times N}$. We define the* $\{x, y, \theta\}$-transformation *to be the functional which transforms the map m_1 into the map m_2 obtained by the translation of $R(m_1)$ to the point (x, y) followed by a rotation of θ degrees. We will indicate it as $T_{x,y,\theta}$, and we will write $m_2 = T_{x,y,\theta}(m_1)$ to indicate that m_2 is obtained from m_1 after the application of the given $\{x, y, \theta\}$-transformation.*

Definition 3. *A* dissimilarity function *ψ over $I_{N \times N}$ is a function*

$$\psi : I_{N,N} \times I_{N,N} \rightarrow \mathbb{R}^+ \cup \{0\}$$

such that

- $\forall m_1 \in I_{N,N}\ \psi(m_1, m_1) = 0$
- *given two maps m_1 and m_2 and a transformation $T_{x,y,\theta}$, then $\psi(m_1, T_{x,y,\theta}(m_2))$ is continuous with respect to x, y and θ.*

The dissimilarity function measures how much two maps differ. In an ideal world, where robots are able to build two perfectly overlapping maps, their dissimilarity will be 0. When the maps cannot be superimposed the ψ function will return positive values.

Having set the scene, the map matching problem can be defined as follows.

Given $m_1 \in I_{N,N}$, $m_2 \in I_{N,N}$ and a dissimilarity function ψ over $I_{N\times N}$, determine the $\{x, y, \theta\}$-transformation $T_{(x,y,\theta)}$ which minimizes

$$\psi(m_1, T_{(x,y,\theta)}(m_2)).$$

The devised problem is clearly an *optimization* problem over $\mathbb{R}^3$. Traditional AI oriented techniques for addressing this problem include genetic algorithms, multipoint hill-climbing and simulated annealing (see for example [6]). We hereby illustrate how a recent technique developed for robot motion planning can be used to solve the same problem. In particular, we will also show that multipoint hill-climbing and simulated annealing can be seen as two special cases of this broader technique.

From now we assume that the values x, y and θ come from a subset of $S \subset \mathbb{R}^3$ which is the Cartesian product of three intervals. In symbols,

$$(x, y, \theta) \in S = [a_0, b_0] \times [a_1, b_1] \times [a_2, b_2].$$

Also, to simplify the notation we will often indicate with $s \in S$ the three parameters which identify a transformation, and we will then write T_s. Before moving into the stochastic part, we define a probability space [7] as the triplet (Ω, Γ, η) where Ω is the sample space, whose generic element is denoted ω. Γ is a $\sigma - algebra$ on Ω and η a probability measure on Γ.

Definition 4. *Let $\{f_1, f_2, \ldots\}$ be a sequence of mass distributions whose events space consists of just two events. The* random selector *induced by $\{f_1, f_2, \ldots\}$ over a domain D is a function*

$$RS_k(a, b) : D \times D \to D$$

which randomly selects one of its two arguments according to the mass distribution f_k.

Definition 5. *Let ψ be a dissimilarity function over $I_{N\times N}$, and RS_f be a random selector over S induced by the sequence of mass distributions $\{f_1, f_2, \ldots\}$. The* acceptance function *associated with ψ and RS_f is defined as follows*

$$A_k : S \times S \to S$$

$$A_k(s_1, s_2) = \begin{cases} s_2 & \text{if } \psi(m_1, T_{s_2}(m_2)) < \psi(m_1, T_{s_1}(m_2)) \\ RS_k(s_1, s_2) & \text{if } \psi(m_1, T_{s_2}(m_2)) > \psi(m_1, T_{s_1}(m_2)) \end{cases}$$

From now on the dependency of A on ψ and RS_f will be implicit, and we will not explicitly mention it. We now have the mathematical tools to define Gaussian random walk stochastic process, which will be used to search for the optimal transformation in S.

Definition 6. *Let t_{start} be a point in S, and let A be an acceptance function. We call* Gaussian random walk *the following discrete time stochastic process $\{T_k\}_{k=0,1,2,3,\ldots}$*

$$\begin{cases} T_0(\omega) = t_{start} \\ T_k(\omega) = A(T_{k-1}(\omega), T_{k-1}(\omega) + v_k(\omega)) \quad k = 1, 2, 3, \ldots \end{cases} \tag{1}$$

where $v_k(\omega)$ is a Gaussian vector with mean μ_k and covariance matrix Σ_k.

From now on the dependence on ω will be implicit and then we will omit to indicate it.

Assumption. We assume that there exist two positive real numbers ε_1 and ε_2 such that for each k the covariance matrix Σ_k satisfies the following inequalities:

$$\varepsilon_1 I \leq \Sigma_k \leq \varepsilon_2 I. \tag{2}$$

where the matrix inequality $A \leq B$ means that $B - A$ is positive semidefinite. The following theorem proves that the stochastic process defined in 1 will eventually discover the optimal transformation in S. The proof is omitted for lack of space.

Theorem 1. *Let $\hat{s} \in S$ be the element which minimizes $\psi(m_1, T_s(m_2))$, and let $\{T_0, T_1, \ldots, T_k\}$ the sequence of transformations generated by the Gaussian random walk defined in equation 1. Let T_b^k be the best transformation generated among the first k elements, i.e. the one yielding the smallest value of ψ. Then for each $\varepsilon > 0$*

$$\lim_{k \to +\infty} \Pr[|\psi(m_1, T_b^k(m_2)) - \psi(m_1, T_{\hat{s}})(m_2))| > \varepsilon] = 0 \tag{3}$$

Algorithm 1 depicts the procedure used for exploring the space of possible transformations accordingly to the stochastic process illustrated. As the optimal value

```
 1: k ← 0,   t_k ← t_start,   Σ_0 ← Σ_init,   μ_0 ← μ_init
 2: c_0 ← ψ(m_1, T_{t_start}(m_2))
 3: loop
 4:    Generate a new sample s ← x_k + v_k
 5:    c_s ← ψ(m_1, T_s(m_2))
 6:    if c_s < c_k OR RD(t_k, s) = s then
 7:       k ← k + 1,   t_k ← s,   c_k = c_s
 8:       Σ_k ← Update(t_k, t_{k-1}, t_{k-2}, ..., t_{k-M})
 9:       μ_k ← Update(x_k, t_{k-1}, t_{k-2}, ..., t_{k-M})
10:    else
11:       discard the sample s
```

Algorithm 1: Basic Gaussian Random Walk Exploration algorithm

of the dissimilarity is not known, practically the algorithm will be bounded to a certain number of iterations and it will return the transformation producing the lowest ψ value.

We wish to outline that this algorithm is a modification of the Adaptive Random Walk motion planner we have recently introduced [4]. The fundamental difference is that in motion planning one has to explore the space of configurations in order to reach a known target point, while in this case this information is not available.

3 Numerical Results

The results presented in this section are based on real-world data collected with the IUB rescue robots. A detailed description of the robots is found in [1]. We describe how we implemented the algorithm described in section 2 and we sketch the results we obtained. In our implementation a map is a grid of 200 by 200 elements, whose elements can assume integer values between -255 and 255. This is actually the output of the mapping system we described in [2]. According to such implementation, positive values indicate free space, while negative values indicate obstacles. As anticipated, the absolute value indicates the belief, while a 0 value indicates lack of knowledge. The function ψ used for driving the search over the space S is defined upon a map distance function borrowed from picture distance computation [8]. Given the maps m_1 and m_2, the function is defined as follows

$$\psi(m_1, m_2) = \sum_{c \in \mathcal{C}} d(m_1, m_2, c) + d(m_2, m_1, c)$$

$$d(m_1, m_2, c) = \frac{\sum_{m_1[p_1]=c} \min\{md(p_1, p_2) | m_2[p_2] = c\}}{\#_c(a)}$$

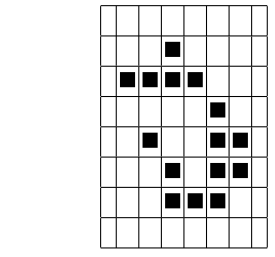
where

- $\mathcal{C}$ denotes a set of values assumed by m_1 or m_2,
- $m_1[p]$ denotes the value c of map m_1 at position $p = (x, y)$,
- $md(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$ is the Manhattan-distance between p_1 and p_2,
- $\#_c(m_1) = \#\{p_1 | m_1[p_1] = c\}$ is the number of cells in m_1 with value c.

Before computing D, we preprocess the maps m_1 and m_2 setting all positive values to 255 and all negative values to -255. In our case then $\mathcal{C} = \{-255, 255\}$, i.e., locations mapped as unknown are neglected. A less obvious part of the linear time implementation of the picture distance function is the computation of the numerator in the $d(m_1, m_2, c)$-equation. It is based on a so called distance-map *d-map*$_c$ for a value c. The distance-map is an array of the Manhattan-distances to the nearest point with value c in map m_2 for all positions $p_1 = (x_1, y_1)$:

$$\text{d-map}_c[x_1][y_1] = \min\{md(p_1, p_2) | m_2[p_2] = c\}$$

The distance-map *d-map*$_c$ for a value c is used as lookup-table for the computation of the sum over all cells in m_1 with value c. Figure 1 shows an example of a distance-map. Algorithm 2 gives the pseudocode for the three steps carried out to built it, while the underlying principle is illustrated in 2.



■ = map cell with value c

3	2	2	1	2	3	4	5
2	1	1	0	1	2	3	4
1	0	0	0	0	1	2	3
2	1	1	1	1	0	1	2
2	1	0	1	1	0	0	1
3	2	1	0	1	0	0	1
3	2	1	0	0	0	1	2
4	3	2	1	1	1	2	3

d-map$_c$

Fig. 1. A distance-map *d-map*$_c$

```
 1: for  y ← 0 to n − 1  do
 2:    for  x ← 0 to n − 1  do
 3:       if  M(x, y) = c  then
 4:          d-map_c[x][y] ← 0
 5:       else
 6:          d-map_c[x][y] ← ∞
 7: for  y ← 0 to n − 1  do
 8:    for  x ← 0 to n − 1  do
 9:       h ← min(d-map_c[x − 1][y] + 1, d-map_c[x][y − 1] + 1)
10:       d-map_c[x][y] = min(d-map_c[x][y], h)
11: for  y ← n − 1 downto 0  do
12:    for  x ← n − 1 downto 0  do
13:       h ← min(d-map_c[x + 1][y] + 1, d − map_c[x][y + 1] + 1)
14:       d-map_c[x][y] = min(d-map_c[x][y], h)
```

Algorithm 2: The algorithm for computing *d-map*$_c$

It can be appreciated that to build the lookup map it is necessary just to scan the target map for three times. In this case it is possible to avoid the quadratic matching of each grid cell in m_1 against each grid cell in m_2.

While implementing the Gaussian random walk algorithm one has to choose how to update μ_k, Σ_k and the sequence of mass distributions $\{f_1, f_2, \ldots\}$ used to accept or refuse sampled transformation which lead to an increment in the dissimilarity function ψ. For the experiments later illustrated we update μ_k at each stage to be a unit vector in the direction of the gradient. Only two different Σ_k matrices are used. If the last accepted sample was accepted, $\Sigma_k = 0.1I$, where I is the 3×3 identity matrix. This choice pushes the algorithm to perform a gradient descent. If the last sample has not been accepted, $\Sigma_k = 10I$. This second choice gives the algorithm the possibility to perform big jumps when it has not been able to find a promising descent direction. The random decisor accepts a sampled transformation s with probability

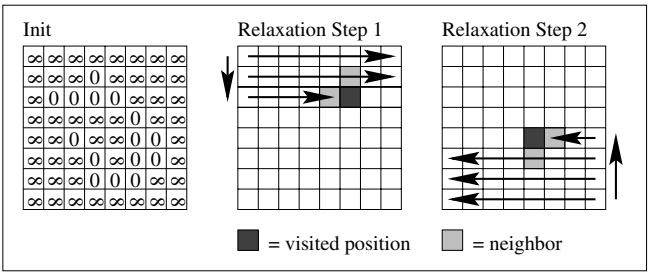
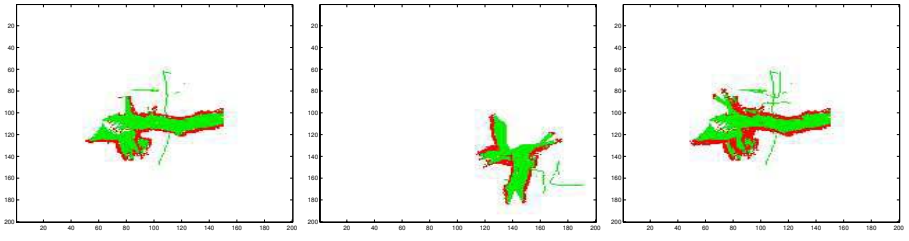


Fig. 2. The working principle for computing $d\text{-}map_c$



(a) (b) (c)

Fig. 3. Subfigures a and b illustrate the maps created by two robots while exploring two different parts of the same environment. To make the matching task more challenging the magnetic compass and the odometry system were differently calibrated. Subfigure c shows the best matching found after 200 iterations of the search algorithm

$$\frac{2(BD - \psi(m_1, T_s(m_2))}{BD}$$

where BD is the best dissimilarity value generated up to current step. Figure 3 illustrates the result of the search procedure.

In particular, subfigure d shows the trend of the dissimilarity function for the sampled transformation generated by the algorithm. Many recurrent gradient descents stages can be observed, interleaved by exploring stages where wide spikes are generated as a consequence of samples generated far from the point currently being explored. In the devised example 200 iterations are enough to let the search algorithm find a transformation almost identical to the best one (which was determined by applying a brute force algorithm).

4 Conclusions

We addressed the problem of map fusion in rescue robotics. The specific problem is to find a good matching of partially overlapping maps subject to a significant amount of noise. This means finding a suitable rotation and translation which

optimize an overlapping quality index. It is then required to perform a search in order to detect the parameters which optimizes a quality index. We introduced a theoretical framework, called Gaussian random walk. We outlined that it generalizes some well known approaches for iterative improvement. In fact, it incorporates a few parameters that when properly tuned can result in techniques like hill-climbing or simulated annealing. The novel aspect of the proposed algorithm is in the possibility to use time variant random distributions, i.e. the distributions' parameters can be updated and tuned accordingly to the already generated samples. It has in fact to be observed that most of the random based approaches use stationary distributions, or distributions whose time dynamics is not influenced by the partial results already obtained.

The proposed algorithm has been applied for fusing maps produced by the robots we are currently using in the Real Rescue competition. Preliminary results confirm the effectiveness of the proposed technique, both in terms of result accuracy and computation speed.

References

1. Birk, A., Carpin, S., Kenn, H.: The iub 2003 rescue robot team. In: Robocup 2003. Springer (2003)
2. Carpin, S., Kenn, H., Birk, A.: Autonomous mapping in the real robot rescue league. In: Robocup 2003. Springer (2003)
3. Parker, L.: Current state of the art in distributed autonomous mobile robots. In Parker, L., Bekey, G., J.Barhen, eds.: Distributed Autonomous Robotic Systems 4. Springer (2000) 3–12
4. Carpin, S., Pillonetto, G.: Motion planning using adaptive random walks. IEEE Transactions on Robotics and Automation (To appear)
5. Carpin, S., Pillonetto, G.: Learning sample distribution for randomized robot motion planning: role of history size. In: Proceedings of the 3rd International Conference on Artificial Intelligence and Applications, ACTA press (2003) 58–63
6. Russel, S., Norwig, P.: Artificial Intelligence - A modern approach. Prentice Hall International (1995)
7. Papoulis, A.: Probability, Random Variables, and Stochastic Processes. McGraw-Hill (1991)
8. Birk, A.: Learning geometric concepts with an evolutionary algorithm. In: Proc. of The Fifth Annual Conference on Evolutionary Programming, The MIT Press, Cambridge (1996)