On Map Merging

Stefano Carpin, Andreas Birk, Viktoras Jucikas

School of engineering and science
International University Bremen
POBOX 750561, D-28725 Bremen, Germany

Abstract

We illustrate our experience in developing and implementing algorithms for map merging, i.e., the problem of fusing two or more partial maps without common reference frames into one large global map. The partial maps may for example be acquired by multiple robots, or during several runs of a single robot from varying starting positions. Our work deals with low quality maps based on probabilistic grids, motivated by the goal to develop multiple mobile platforms to be used in rescue environments. Several contributions to map merging are presented. First of all, we address map merging using a motion planning algorithm. The merging process can be done by rotating and translating the partial maps until similar regions overlap. Second, a motion planning algorithm is presented which is particular suited for this task. Third, a special metric is presented which guides the motion planning algorithm towards the goal of optimally overlapping partial maps. Results with our approach are presented based on data gathered from real robots developed for the RoboCupRescue real robot league.

Key words: stochastic algorithms, rescue robotics, cooperative multi-robot systems

1 Introduction

The ability to build a map of an unknown environment is one of the fundamental enabling capabilities for mobile robots. Having an accurate map enables the robot to perform certain tasks like navigation, localization and so forth much faster and more accurately. In fact, in the past years a significant amount of research has been devoted to the subject (see for example [1] for a survey on mapping algorithms). A generalized problem often addressed in the same context is the so called *SLAM* (Simultaneous Localization And Mapping), where the robot is required to build a map and localize itself at the same time. Some common aspects of the former literature are the following.

First, the map is produced and used by the robot for tasks like navigation, motion and mission planning, etc. Second, the robot is equipped with sensors and algorithms which if correctly used can produce highly precise maps. Finally, the robot is supposed to move in a structured environment, like the interior of a building, where certain features like doors, corridors intersections and so on can be easily extracted and identified.

In this paper we address an important variant of the mapping problem. Our goal is to combine, or merge, two or more planar maps. As pointed out in [2], *map merging is an interesting and difficult problem, which has not enjoyed the same attention that localization and map building have.* The problem of map merging has been partially addressed in [3], where local maps are periodically merged into a global one, to reduce the computational efforts of constantly maintaining a global map. Online multi-robot mapping has been addressed in various researches like [4],[5], but the underlying hypothesis and operating scenarios are different. Our main assumption for map merging is that there is no knowledge about the relative positions of the partial maps which are to be combined. Take for example multiple robots that independently explore an environment. If there is no global reference frame on which the starting poses of the robots can be mapped, there is no known relation between the individual maps the robots generate. Each of them has its own reference frame based on the initial poses of the robots. The same holds if a single robot does multiple runs from different starting positions in the same environment. If the starting positions are not on known locations on a global reference frame, i.e. if no global map already exists, the reference frames of the acquired maps will be based on the unknown initial starting points. Individual maps, from individual robots in the multi-robot case, or individual runs in the single robot case, will typically cover only a part of the environment; we hence dub them *partial maps*. So, the challenge for the merging process is to identify regions that occur in at least two partial maps to join them at this region. Throughout the paper it is assumed that maps to be merged exhibit at least some overlap. The proposed approach is not applicable when the two maps do not share any part. In such case, in fact, the whole map merging problem loses significance.

Our research efforts are motivated by the goal to develop rescue robots, to be used in the Robocup Rescue League at the moment, and in real world applications in a following stage. We therefore want fast and complete exploration, since the ultimate goal is to locate injured persons and to extract them. It is then reasonable to use more than one robot, also to exploit other well known advantages of multi-robot systems, like robustness, heterogeneity and so on [6]. In the context of the Robocup Rescue League, we developed some robots which enter and map the same environment at the same time. Because of the time issue, it is foreseeable that they will end up exploring different parts of the rescue scenario. Then, when their exploration is completed, the partially overlapping maps should be combined to produce a useful tool for humans as

described in [7], where an early form of this work is presented.

The problem of merging two or more maps into a single one can be seen as an optimization problem, as usually more than a single matching is possible and the system is required to determine the best one, according to some metric. In this article, we present several contributions to map merging. First of all, we observe that the map merging problem is conceptually similar to the ligand docking problem studied in computational biology. Both require to determine a transformation that minimizes a given cost (or energy) function. Building on the fact that motion planning algorithms are being currently used to address problems coming from computational biology [8],[9], we propose to use a motion planning algorithm for studying map merging. The merging process can be done by rotating and translating the partial maps until identical regions overlap, somewhat similar to protein docking. Second, a motion planning algorithm is presented which is particular suited for this task. Third, a special metric is presented which guides the motion planning algorithm toward the goal of optimally overlapping partial maps.

In section 2 we elucidate how the map merging problem can be formulated as an optimization problem and we describe a motion planning algorithm for solving this problem. Section 3 describes a metric which guides the motion planning algorithm. The robots we built for the Robocup Rescue competition are briefly depicted in section 4, where also the experimental results are presented. Section 5 concludes the article.

2 Merging via random walks

The maps we aim to merge are based on the mapping software we developed for the Robocup 2003 competition [10]. The mapping algorithm produces a grid where beliefs are encoded, according to the sensor inputs. Each grid cell is assigned a certain belief that it is free or occupied. While one could correctly argue that more refined mapping algorithms might be developed, all the remaining parts of the paper still remain valid. Our goal is to merge maps, independently from how they have been created. In particular, as we wish to merge low quality maps, simple mapping strategies yield more challenging problems and allow to better test and compare different techniques. Moreover, given the peculiar characteristics of the environment being mapped, the use of probabilistic maps appears appropriate [11]. Before getting into the description of the algorithmic aspects, we begin with a formal definition of the problem we are addressing. We start with the 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 $\mathbb{R}^2$ is declared to be its reference point [1] *. The reference point of map m will be indicated as $R(m) = (r_x^m, r_y^m)$.*

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. 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. In the current problem formulation we are for generality assuming that maps are functions from subsets of $\mathbb{R}^2$ to $\mathbb{R}$. It is nevertheless evident that some discretization will be needed when implementing these algorithms. This aspect will be addressed while discussing practical implementation details. We next define a planar transformation which will be used to try different relative placements of two maps to find a good matching. The formal definition is the following.

Definition 2 *Let t_x,t_y and θ be three real numbers. The transformation associated with t_x,t_y and θ is the function*

$$TR_{t_x,t_y,\theta}(x, y) : \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

defined as follows:

$$TR_{t_x,t_y,\theta}(x, y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (1)$$

As known, the transformation given in equation 1 corresponds to a counterclockwise rotation of θ about the origin, followed by a translation of (t_x, t_y). This transformation is an injective function and this fact will be useful in what follows.

Definition 3 *Let t_x,t_y and θ be three real numbers. The associated $\{t_x, t_y, \theta\}$-*map transformation *over $I_{N\times M}$ is the functional*

$$T_{t_x,t_y,\theta} : I_{N\times M} \rightarrow I_{N\times M}$$

[1] the reader should note that the reference point does not necessary belong to the domain. This will make some notation easier later on.

which transforms the map $m_1 \in I_{N\times M}$ *into the map* $m_2 \in I_{N\times M}$ *defined as follows:*

$$m_2(x,y) = \begin{cases} m_1(x',y') \text{ if } \exists (x',y') \in [0,N]\times[0,M] | \\ \qquad\qquad TR_{t_x,t_y,\theta}(x' - r_x^{m_1}, y' - r_y^{m_1}) = (x,y) \\ 0 \qquad\qquad \text{elsewhere} \end{cases} \tag{2}$$

Moreover, the reference point of m_2 *is defined as* $R(m_2) = (t_x, t_y)$.

The first case of equation 2 is well defined in light of the injectivity of equation 1. The rationale behind the former definition is to define how a map is transformed into another when it is rotated and translated. As in general arbitrary translations can be applied, the formulation given in equation 2 assures that the transformed map m_2 is well defined and has the same domain of m_1.

Definition 4 *A* dissimilarity function ψ *over* $I_{N\times M}$ *is a function*

$$\psi : I_{N,M} \times I_{N,M} \rightarrow \mathbb{R}^+ \cup \{0\} \tag{3}$$

such that

- $\forall m_1 \in I_{N,M} \Rightarrow \psi(m_1, m_1) = 0$
- *given two maps* m_1 *and* m_2 *and a map 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 would be able to build maps which correspond to the ground truth and completely cover the operating environment, it would be possible to find a transformation which yields a 0 value for the dissimilarity function. In real applications this is obviously not the case, so the challenge is to find a transformation giving a low dissimilarity value.

Having set the scene, the map merging problem can be defined as follows.
Given $m_1 \in I_{N,M}$*,* $m_2 \in I_{N,M}$*, and a dissimilarity function* ψ *over* $I_{N\times M}$*, determine the* $\{x, y, \theta\}$*-map transformation* $T_{(x,y,\theta)}$ *which minimizes*

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

The devised problem is clearly an optimization problem, where it is required to minimize a cost function, that in our case is the dissimilarity function ψ. The optimization has to be performed over a three dimensional space involving two translations and one rotation. In the following we will indicate with S the set of the possible translations and rotations and with s_i a specific element of S, i.e. $s_i = \{x_i, y_i, \theta_i\}$. We will assume that when we are looking for the

optimal matching between $m_1 \in I_{N\times M}$ and $m_2 \in I_{N\times M}$ the set of possible transformations will be

$$S = [-N, N] \times [-M, M] \times [0, 2\pi]$$

For problems like this, a certain number of algorithms like multi-point hill climbing and simulated annealing have been developed and are part of the standard literature. They will be described in section 2.1 where they will be related to the technique we present in this section. The algorithm was inspired by a motion planning algorithm one of the authors recently developed [12],[13],[14], and by the recent successful application of motion planning algorithms to bioinformatics problems like ligand binding and protein folding [9],[15]. In a sense, the problems of ligand binding and map merging are not so different. In both cases it is required to determine the relative placements of two entities, chemical compounds or maps, so that a certain cost function is minimized. In ligand binding it is required to minimize the total energy, while in map merging we wish to minimize dissimilarity, i.e. maximize the overlap of regions that appear in two or more partial maps. The algorithm explores the space of possible transformations performing a random walk whose probability distribution is a time variant Gaussian. This means that it evolves from transformation s_i to transformation s_{i+1} generating random variations with a Gaussian distribution. The important aspect is that distributions' parameters are not a priori fixed, but rather updated from iteration to iteration. Before describing the algorithm, we provide the theoretical foundations. We start then by defining a probability space 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 5 *Let $\{f_1, f_2, \ldots\}$ be a sequence of mass distributions whose events space is binary. The* random selector *induced by $\{f_1, f_2, \ldots\}$ over a domain D is a function*

$$RS_{f_k}(a, b) : D \times D \rightarrow D$$

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

The *random selector* is the stochastic process which is used to accept or reject a new candidate random transformation. The dependency on f_k evidences that the probability of accepting or refusing is not constant but can rather updated from iteration to iteration.

Definition 6 *Let ψ be a dissimilarity function over $I_{N\times M}$, $\{f_1, f_2, \ldots\}$ be a sequence of mass distributions as in definition 5, and RS_{f_k} be the associated random selector. The* acceptance function *associated with ψ and RS_{f_k} is defined as follows*

$$A_k : S \times S \rightarrow S$$

$$A_k(s_i, s_{i+1}) = \begin{cases} s_{i+1} & \text{if } \psi(m_1, T_{s_{i+1}}(m_2)) < \psi(m_1, T_{s_i}(m_2)) \\ RS_k(s_i, s_{i+1}) & \text{if } \psi(m_1, T_{s_{i+1}}(m_2)) > \psi(m_1, T_{s_i}(m_2)) \end{cases}$$

The *acceptance function* associated with a given random selector and dissimilarity function governs the process which accepts or rejects a candidate transformation which does not bring any benefit, i.e. a transformation associated with an increase in the dissimilarity. From now on the dependency of A_k on ψ and RS_f will be implicit, and we will not explicitly mention it. We now have the mathematical tools to define a Gaussian random walk stochastic process, which will be used to search for the optimal transformation in S.

Definition 7 *Let s_{start} be a point in S, and let A be an acceptance function. We call* transformation random walk *the following discrete time stochastic process $\{T_k\}_{k=0,1,2,3,...}$*

$$\begin{cases} T_0(\omega) = s_{start} \\ T_k(\omega) = A(T_{k-1}(\omega), T_{k-1}(\omega) + v_k(\omega)) \quad k = 1, 2, 3, \dots \end{cases} \tag{4}$$

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 three positive real numbers ε_1, ε_2 and ε_3 such that for each k the following inequalities are satisfied

$$\varepsilon_1 I \leq \Sigma_k \leq \varepsilon_2 I \tag{5}$$

$$||\mu_k||_2 \leq \varepsilon_3 \tag{6}$$

where the matrix inequality $A \leq B$ means that $B - A$ is positive semidefinite. The following theorem proves that the stochastic process defined in equation 4 will eventually discover the optimal transformation in S.

Theorem 1 *Let $s^* \in S$ be an element which minimizes $\psi(m_1, T_s(m_2))$, and let $\{T_0, T_1, T_2 \dots\}$ be the sequence of transformations generated by the transformation random walk defined in equation 4. Let T_b^k be the best one generated among the first k transformations, i.e. the one yielding the smallest value of ψ. Then for each $\varepsilon > 0$*

$$\lim_{k \to +\infty} \Pr[|\psi(m_1, T_b^k(m_2)) - \psi(m_1, T_{s^*})(m_2))| > \varepsilon] = 0 \tag{7}$$

Proof. We first point out that the element s^* indeed exists, as the function ψ is continuous by definition, and we have assumed S to be a closed bounded set. So there must be a transformation $s^* \in S$ which minimizes ψ. In addition it has to be pointed out that such an element is not necessarily unique, as two transformations may result to the same dissimilarity.
Let $B_s(\phi)$ be the ball centered in s and with radius ϕ. As consequence of the continuity of the function ψ, for each $\varepsilon > 0$ it is possible to determine a suitable ball $B_{s^*}(\delta)$ such that

$$\forall \hat{s} \in B_{s^*}(\delta) \quad |\psi(m_1, T_{\hat{s}}(m_2)) - \psi(m_1, T_{s^*})(m_2))| < \varepsilon$$

Let us consider the time step k, when the transformation T_k is generated from T_{k-1}. Because of the assumptions stated in equations 5 and 6, the following inequality holds

$$\Pr[\psi(m_1, T_k(m2)) \in B_{s^*}(\delta)] = \int\limits_{B_{s^*}(\delta)} N_{T_{k-1}+\mu_k, \Sigma_k}(x)dx \geq L$$

where $N_{T_{k-1}+\mu_k, \Sigma_k}$ is the probability density of a Gaussian distribution with mean vector $T_{k-1} + \mu_k$ and covariance matrix Σ_k. It is important to realize that the same L holds for each k. Given the sequence of transformations $\{T_0, T_1, \ldots, T_k\}$, the probability that none of them is in $B_{s^*}(\delta)$ is the following

$$\Pr[T_0 \notin B_{s^*}(\delta) \ldots T_k \notin B_{s^*}(\delta)] \leq (1 - L)^k \tag{8}$$

According to the definition of the stochastic process given in 4, it is easy to verify that this probability is an upper bound for equation 7. Clearly, when k diverges the right side of equation 8 goes to 0 and this proves the theorem. **QED**

One can expect that a random walk evolving over S will eventually cover it completely, thus determining the optimal value of ψ. The important part however is that by using a random walk whose mean and covariance can be updated at each iteration, it is possible to focus the search more effectively. In addition, one is relieved from the daunting task of fine tuning the parameters, i.e. determining good values for μ_k and Σ_k. This was indeed the lesson that was learned while developing the motion planner described in [14].
Algorithm 1 illustrates how the stochastic machinery described before can be turned into an algorithmic searching procedure. As the optimal value for the dissimilarity function is unknown, the overall number of iterations should be limited. This aspect will be later clarified when describing the practical results. Another point of practical importance is the necessity to store the best transformation generated so far, i.e. the one associated with the lowest dissimilarity value. With regard to local minima, which are likely to occur, the use of the random selector RS leaves a chance to escape from them.

Algorithm 1 Random walk

Require: $numSteps \geq 0$

1: $k \leftarrow 0, \quad t_k \leftarrow s_{start}$
2: $\Sigma_0 \leftarrow \Sigma_{init}, \quad \mu_0 \leftarrow \mu_{init}$
3: $c_0 \leftarrow \psi(m_1, T_{t_{start}}(m_2))$
4: **while** $k < numSteps$ **do**
5: Generate a new sample $s \leftarrow t_k + v_k$
6: $c_s \leftarrow \psi(m_1, T_s(m_2))$
7: **if** $c_s < c_k$ OR $RS(t_k, s) = s$ **then**
8: $k \leftarrow k + 1, \quad t_k \leftarrow s, \quad c_k = c_s$
9: $\Sigma_k \leftarrow \text{Update}(t_k, t_{k-1}, t_{k-2}, \ldots, t_{k-M})$
10: $\mu_k \leftarrow \text{Update}(x_k, t_{k-1}, t_{k-2}, \ldots, t_{k-M})$
11: **else**
12: discard the sample s

Lines 9 and 10 in algorithm 1 illustrate that the mean μ_k and the covariance Σ_k can be updated at each step, according to the last accepted M samples. For example, if the last accepted M samples did not decrease the dissimilarity functional it means that the random walk is into a local minima region and then the covariance matrix should be increased in order to escape it. On the other hand, when the last accepted samples have resulted in a decrease of the dissimilarity, it is worth reducing the covariance and to set the mean in the direction of the gradient. In this way the random walk is biased to follow a local gradient descent.

2.1 Relation to other search techniques

As mentioned before, there are other techniques that could be used for this search problem. We here describe two of the most common ones, and we discuss how they relate to the formerly described technique. An obvious first approach to the problem is to perform a gradient descent search following a multipoint paradigm (see for example [16]), as illustrated in algorithm 2. The algorithm operates over a set of candidate transformations, and at each iteration it expands the more promising one toward directions which result in a decrease of the dissimilarity function. Using a set of transformations rather than a single one, the algorithm decreases the chances of getting trapped into local minima. Gradient descent is a particular case of algorithm 1, where the vector the mean μ_k is fixed to be equal to the most promising direction Δs and the covariance Σ_k tends to the 0 matrix [2]. One could argue that this violates the hypothesis of theorem 1. This is indeed true, but at the same time it is necessary to consider that the gradient descent algorithm is not guaranteed

[2] with this we mean that the added Gaussian component is not present.

Algorithm 2 multi-point hill climbing

Require: $numSteps \geq 0$
Require: $numBestSamples > 0$
Require: *samples* array contains different samples for position/rotation and is sorted according to their dissimilarity
Ensure: $samples[1]$ is the best sample found

```
1: step ← 0
2: while step < numSteps do
3:    clear newSamples
4:    for all s in samples do
5:       for all Δs in possible position/rotation changes do
6:          Generate a new sample ns ← s + Δs
7:          c_ns ← ψ(m_1, T_ns(m_2))
8:          if c_ns < c_s then
9:             Add ns to newSamples
10:         else
11:            Discard sample ns
12:   step ← step + 1
13:   samples ← newSamples
14:   sort samples according to their dissimilarity
15:   samples ← samples[1..numBestSamples]
```

to find the optimal solution, then the correctness of the framework is not affected. The multi-point approach illustrated in algorithm 2 can still be seen as a paricular case of algorithm 1. This can be accomodated by fixing $\mu_k = \Delta s$ for *numSteps* steps and then resetting it to a random value uniformly picked over S. Repeating this process *numBestSamples* times will indeed replicate the same search procedure.

An obvious alternative to hill climbing is simulated annealing [17],[18]. The well known advantage is that in the early stage of its execution simulated annealing allows to accept samples which do not give an adavantege in terms of the cost function. Algorithm 3 shows our implementation of the simulated annealing approach for map fusion. It is again straightforward to observe that also simulated annealing can be seen as a special case of the Gaussian random walk. In addition to what has been described for the gradient descent algorithm, it is sufficient to embed the *temperature profile* into the random selector.

3 The dissimilarity function

A fundamental aspect of the depicted framework is the choice of the dissimilarity function ψ. For this purpose we use a function borrowed from a metric

Algorithm 3 simulated annealing

Require: $numSteps \geq 0$
Require: $numBestSamples > 0$
Require: *samples* array contains different samples for position/rotation and is sorted according to their fitness
Require: $fitnessRatio \geq 1$
Ensure: $samples[1]$ is the best sample found

```
step ← 0
bestFitness ← c_samples[1]
lastFitness ← +∞
while step < numSteps AND lastFitness/bestFitness > fitnessRatio do
    clear newSamples
    for all s in samples do
        for all Δs in possible position/rotation changes do
            Generate a new sample ns ← s + Δs/step
            c_ns ← ψ(m_1, T_ns(m_2))
            if c_ns < c_s then
                Add ns to newSamples
            else
                Discard sample ns
    step ← step + 1
    samples ← newSamples
    sort samples according to their fitness
    samples ← samples[1..numBestSamples]
    lastFitness ← bestFitness
    bestFitness ← c_samples[1]
```

introduced before by one of the authors to measure the similarity of images [19]. The metric is conceptually similar to the Grassfire transform used in computer vision. Given two maps m_1 and m_2, the dissimilarity function is defined as follows:

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

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

where

- $\mathcal{C}$ denotes the 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 points 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.

While computing md, the choice of the Manhattan-distance has the advantage to be fast to compute and not to perform worse than other distances. Before computing d, we preprocess the maps m_1 and m_2 setting all positive values to 255 and all negative values to -255. The purpose of this operation is to reduce the set of different values c contained in the map. This simplifies the summation to the sum of two terms, thus speeding up it computation. In our case then $\mathcal{C} = \{-255, 255\}$, i.e., locations mapped as unknown are neglected. As the dissimilarity function has to be evaluated each time a new candidate sample is proposed, it is important to have an efficient algorithm for its computation. Quite surprisingly, it is possible to compute the function ψ in linear time. The algorithm 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 $p_1 = (x_1, y_1)$:

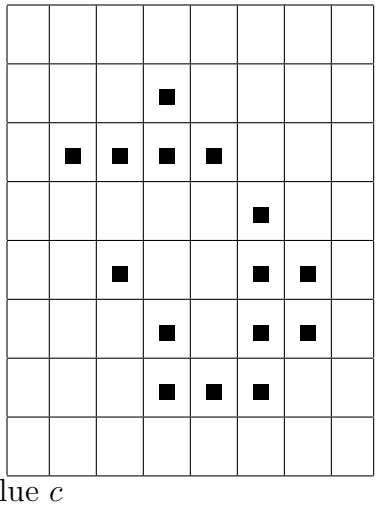
$$\text{d-map}_c[x_1][y_1] = \min\{md(p_1, p_2) | m_2[p_2] = c\}$$

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, while algorithm 4 gives the pseudocode for the three steps carried out to built it. The underlying principle is illustrated in figure 2.

Algorithm 4 The algorithm for computing *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)
```

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.



■ = 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*$_c$

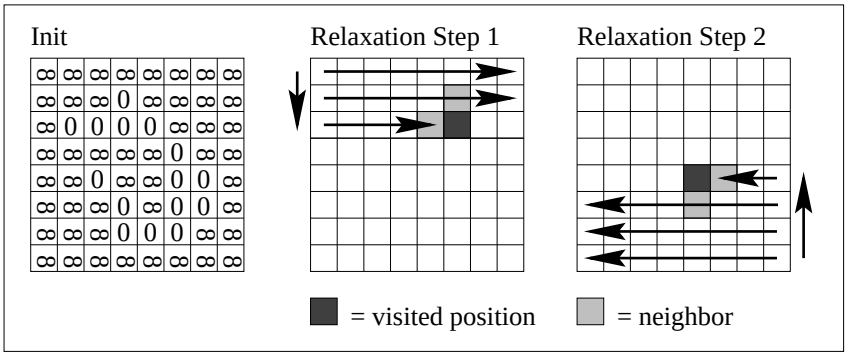
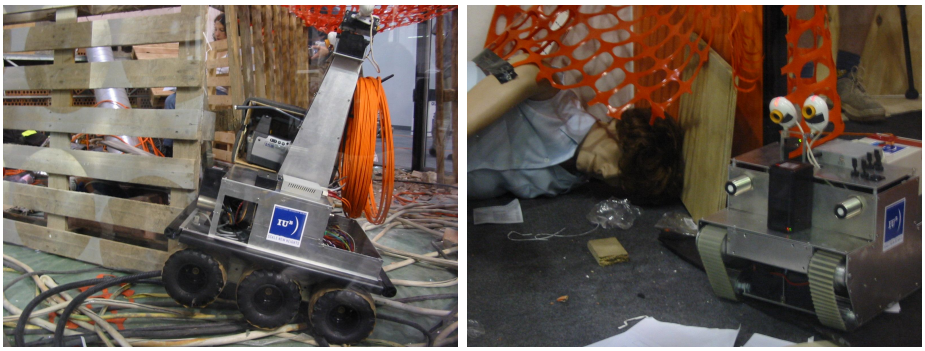


Fig. 2. The working principle for computing *d-map*$_c$

4 Results

4.1 The Rescue Robots

The experimental results are based on real-world data collected with the rescue robots we developed. The robots fall into two classes, the so-called papa goose and the mama goose design. Papa geese (figure 3.a) are 6-wheeled robots with a footprint of $400mm \times 450mm$. Mother geese (figure 3.c) are smaller, with a footprint of $300mm \times 400mm$ and they are based on a tracked drive. In addition to their mechanical differences, they differ in their sensor payload, which is much higher for robots of the papa type. A detailed description of the robots is found in [20] and [21].



(a) Papa goose in the red arena (b) Mama goose in the orange arena

Fig. 3. The IUB robots at the Robocup 2003 competition in Padua, Italy

For the purpose of this paper, the merging of noisy maps of an unstructured environment, both robots can be assumed to behave in the same way. The default localization method is odometry based on motor encoder data and the robots' kinematics. The precision of the odometry is significantly improved by fusing compass data into the orientation estimation of the robot. Nevertheless, the basic pose estimation suffers from the well known problems of accumulative errors, leading to one of the noise sources in the map-building process. For gathering obstacle data, each robot has a low-cost laser scanner. This PB9-11 laser-scanner from Hokuyo Automatic covers 162 degrees in 91 steps up to 3m depth with a 1cm resolution (figure 4). The sensor provides rather accurate data, but it suffers from occasional spurious readings adding hence to the uncertainty in the map data.

Each individual map is created online on board of each robot and can be used for various autonomous behaviors according to the mission of the robot. The data is in addition transmitted in realtime to the operator station via the Framework Architecture for Selfcontrolled and Teleoperated Robots (FAST-Robots)[22], an object-oriented network control architecture framework for robotics experiments that supports mixed teleoperation and autonomy. The

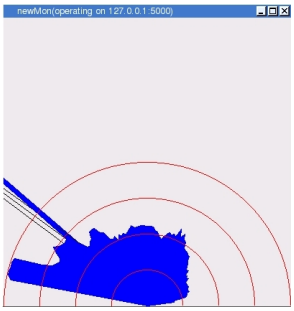


Fig. 4. A snapshot of the data provided by the PB911 sensor. It is possible to see that the robot is facing a corner in the walls (on its right, two meters ahead), as well as spurious readings (on the left)

map merging is then done on the operator station, so that a human rescue worker gets an overview of the overall terrain including location marks for victims and hazards.

4.2 Numerical Results

The algorithms described in the former sections have been implemented and tested over data sets gathered by our robots while performing in the Rescue arena recently set up at the International University Bremen. The maps produced by the mapping software are grids with 200×200 cells, each one containing a belief value between -255 and +255. Positive values indicate a belief that the associated cell is free, while negative values indicate the opposite. As anticipated, the absolute value indicates the degree of the belief. According to this protocol, cells with a 0 belief value do not encode any belief, i.e. no information can be associated with them. In all the illustrated experiments two maps are merged. If more maps are available, the method can be iterated. This means that the map resulting from the merging of the first two is merged with a third one, and so on.

The algorithms have been implemented in C++ and run on a Pentium IV 2.2 GHz running Linux. In addition we implemented also a deterministic brute force algorithm which tries all the possible transformations, and a random walk algorithm without adaptive components. In this way we can better evaluate the tradeoff between speed and accuracy of the different algorithms implemented. Table 1 illustrates the results of the different approaches tried in terms of dissimilarity function, while figure 5 shows the different maps merged. The experimental setup is a square of about 100 m^2, and in the different runs the robots map about half of it. From a qualitative point of view, the merged maps offer more information than the two partial maps, and this closely resembles the area being mapped by the robots.

Case	Brute force	Multi-point	Simulated annealing	Random walk	Gaussian Random Walk
1	4.21	5.89	6.43	4.82	4.86
2	1.23	2.55	3.33	2.27	1.84
3	1.19	1.82	2.96	1.58	1.74
4	3.67	4.42	4.33	4.28	4.20

Table 1
Comparison of the different algorithms implemented. The table shows the dissimilarity value for the optimal merging found. Data of randomized search techniques are averaged over 50 executions.

In all the tests we tuned the algorithms so that the number of generated and analyzed transformations was the same, i.e. is 2000 [3] . This means that each algorithm terminates after the same number of rototranslations have been processed. This number turns out to be more than enough as usually all the algorithms converge to their final point in about half of these iterations. At the same time this allows to get a fair comparison between them as the amount of alloted resources is the same. For what concerns execution time all the analyzed algorithms spend more or less the same, i.e. below three seconds. One should not forget the initial goal of the current research, i.e. to determine the optimal composition of the given maps in a quick time and without too much parameters' tuning. This point will be better addressed in the conclusions, but for the moment it should be clear that the achieved speed is sufficient and it does not make so much sense trying to improve the speed, although technically possible. Figure 6 illustrates an example of maps created by the IUB robots as well as their determined composition using the proposed adaptive random walk algorithm. Subfigure 6.d illustrates how the algorithm behaves. It is possible to observe a repeated pattern of steps where the dissimilarity function decreases followed by spikes where the fitness function increases.

The results indicate that the proposed optimization algorithm behaves better than traditional approaches like multi-point hill climbing or simulated annealing. When compared with the traditional random walk algorithm without adaptive components the advantages in terms of obtained result seem to disappear. In two cases adaptivity seems to pay off and two cases it seems it does not. This is because of an appropriate selection of the parameters for the traditional random walk. In general, however, determining good values for the parameters is not trivial. The great advantage of the proposed algorithm is that one does not have to tune parameters, as they are modified during exe-

[3] Of course this is not the case for the brute force search, where the number of analyzed configurations is 14400000 and the time taken to process all of them is in the order of three hours.

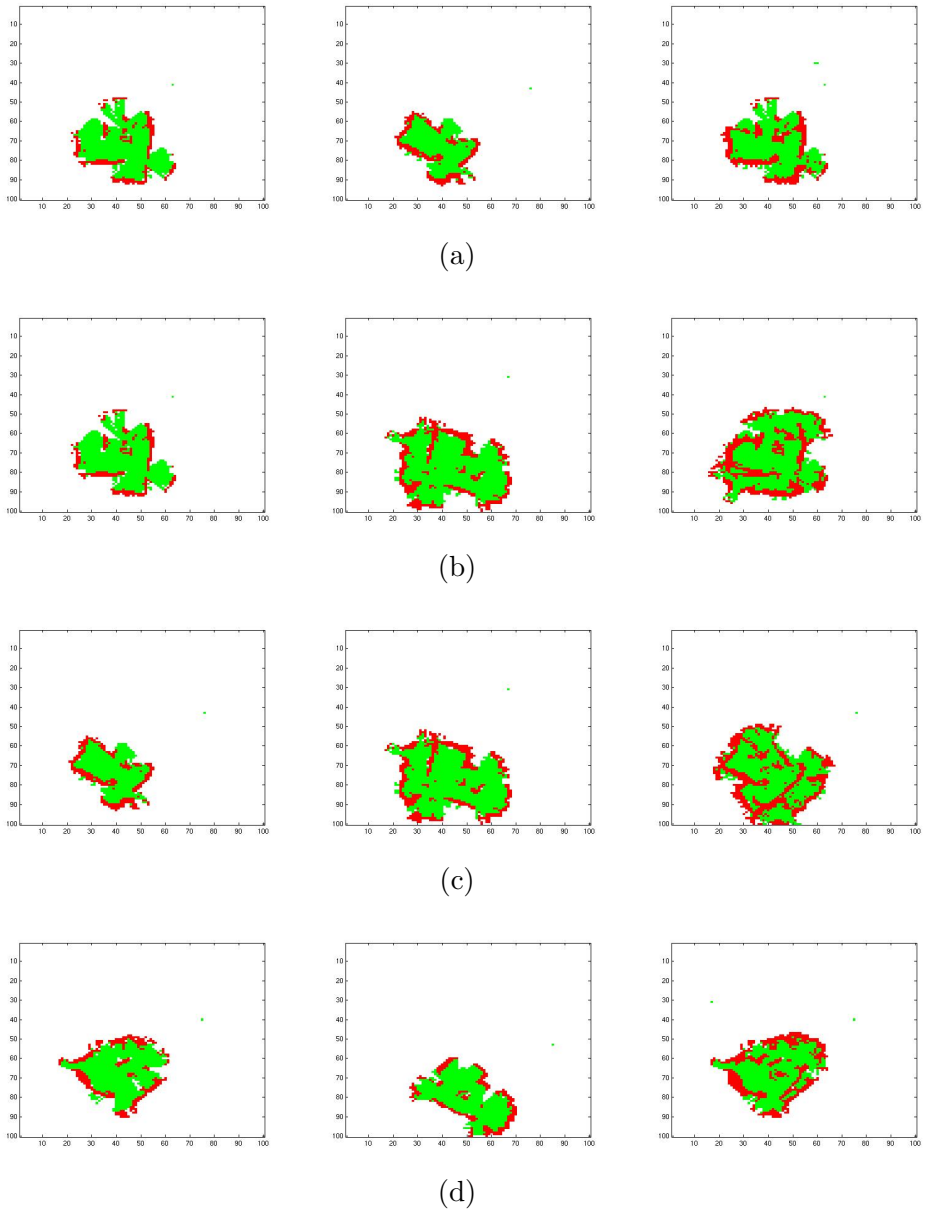


Fig. 5. The four rows illustrate, in sequence, the four case studies whose results are given in table 1. For each row, the first two pictures display the maps to be merged, while the third one illustrates the merged map.

cution. For example, one could choose weak distributions' parameters for the classic random walk algorithm, say picking a too narrow variance which does not allow the procedure to effectively explore the search space. Or one could fix a too big variance, resulting in an algorithm which widely jumps between far apart configuration, without exploring the possible minima it discovers. Good values for these parameters are very much instance dependent and hard to determine, if not by reiterated experience. The adaptive random walk algorithm instead allows the user to fix the values without putting too much efforts on fine tuning. Rather, it will be the algorithm itself to self-tune its

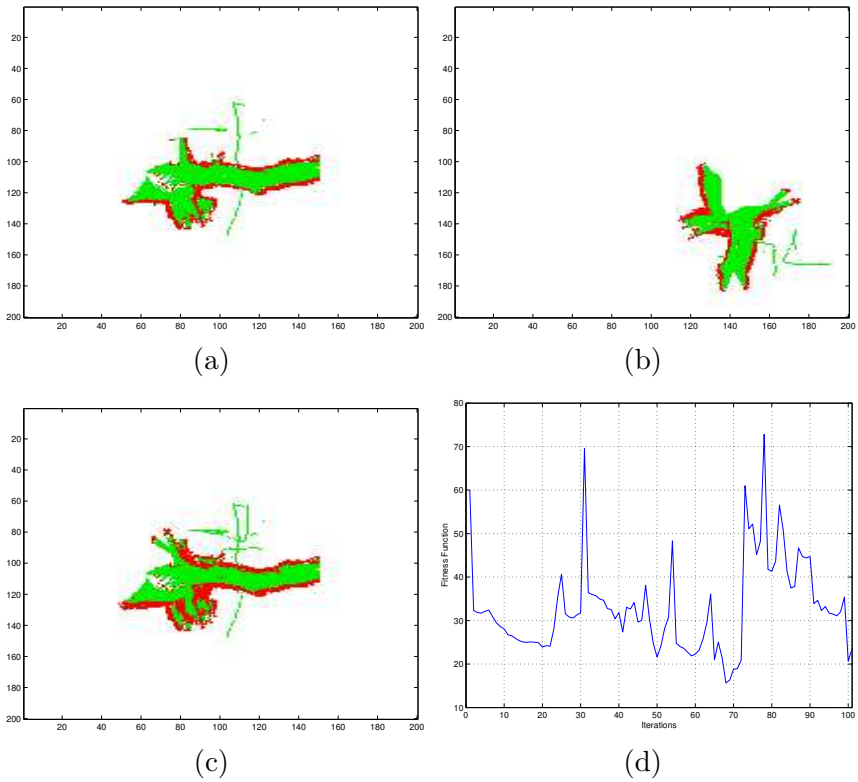


Fig. 6. Subfigures a and b illustrate the maps created by two robots while exploring two different parts of the same environment. To make the matching task more challenging the magnetic compass and the odometry system were differently calibrated. Subfigure c shows the best matching found after 200 iterations of the search algorithm and subfigure d plots the trend of the dissimilarity function during the first 100 iterations.

own distribution. This lesson, which was learned while working on the motion planner previously discussed, turned out be effective also in the apparently very different domain of map merging.

5 Conclusions

Map merging is an important robotics problem, which deals with the fusion of several partial maps into a global one. Partial maps are common, for example when multiple robots explore an environment or when a single robot does multiple runs starting from different locations. It was demonstrated that map merging can be addressed using a motion planning algorithm. The partial maps have to be rotated and translated such that regions which appear in two ore more partial maps overlap. For this purpose, a special motion planning algorithm and a similarity metric were introduced. The special motion planning algorithm is highly efficient, also when compared to alternative techniques, and it does not suffer from the requirement to tune parameters. The similar-

ity metric successfully guides the motion planning algorithm in the presented experiments and it significantly contributes to the overall efficiency as it can be computed in a very fast manner. The experimental results are based on real world data from the IUB rescue robots. The results show that the suggested approach even works with rather low quality maps that were generated without high-end sensors and in a very unstructured environment.

References

[1] S. Thrun, Robotic mapping: A survey, in: G. Lakemeyer, B. Nebel (Eds.), Exploring Artificial Intelligence in the New Millenium, Morgan Kaufmann, 2002.

[2] K. Konolige, D. F. anx B. Limketkai, J. Ko, B. Steward, Map merging for distributed robot navigation, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2003, pp. 212–217.

[3] S. Williams, G. Dissanyake, H. Durrant-Whyte, Towards multi-wehicle simultaneous localisation and mapping, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2002, pp. 2743–2748.

[4] S. Thrun, W. Burgard, D. Fox, A real-time algorithm for mobile robot mapping with applications to multi-robot and 3d mapping, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2000, pp. 321–328.

[5] L. Parker, K. Fregene, Y. Guo, R. Madhavan, Distributed heterogeneous sensing for outdoor multi-robot localization, mapping, and path planning, in: A. Schultz (Ed.), Multi-Robot Systems: From Swarms to Intelligent Automata, Kluwer, 2002, pp. 21–30.

[6] L. Parker, Current state of the art in distributed autonomous mobile robots, in: L. Parker, G. Bekey, J.Barhen (Eds.), Distributed Autonomous Robotic Systems 4, Springer, 2000, pp. 3–12.

[7] S. Carpin, A. Birk, Stochastic map merging in rescue environments, in: Robocup 2004: Robot Soccer World Cup VIII, Springer, 2005.

[8] N. Amato, K. Dill, G.Song, Using motion planning to map protein folding landscapes and analyze folding kinetics of known native structures, in: Proceedings the 6th International Conference on Computational Molecular Biology (RECOMB), 2002, pp. 2–11.

[9] G. Song, N. Amato, A motion-planning approach to folding: From paper craft to protein folding, IEEE Transactions on Robotics and Automation 20 (1) (2004) pp. 60–71.

[10] A. Jacoff, B. Weiss, E. Messina, Evolution of a performance metric for urban search and rescue (2003), in: Performance Metrics for Intelligent Systems, 2003.

[11] H. Moravec, Sensor fusion in certainty grids for mobile robots, AI Magazine 9 (3) (1988), pp. 61–74.

[12] S. Carpin, G. Pillonetto, Robot motion planning using adaptive random walks, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2003, pp. 3809–3814.

[13] S. Carpin, G. Pillonetto, 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, pp. 58–63.

[14] S. Carpin, G. Pillonetto, Motion planning using adaptive random walks, IEEE Transactions on Robotics 21 (1) (2005), pp. 129–136.

[15] A. Singh, J. Latombe, D. Brutlag, A motion planning approach to flexible ligand binding, in: Proceedings of the Int. Conf. on Intelligent Systems for Molecular Biology, 1999, pp. 252–261.

[16] S. Russel, P. Norvig, Artificial Intelligence - A modern approach, Prentice Hall International, 1995.

[17] S. Kirkpatrick, C. Gelatt, M. Vecchi, Optimization by simulated annealing, Science 220 (4598) (1983), pp. 671–680.

[18] M. Locatelli, Simulated annealing algorithms for continuous global optimization, in: P. Pardalos, H. Romeijn (Eds.), Handbook of Global Optimization, Vol. 2, 2002, pp. 179–229.

[19] A. Birk, Learning geometric concepts with an evolutionary algorithm, in: Proc. of The Fifth Annual Conference on Evolutionary Programming, The MIT Press, Cambridge, 1996.

[20] A. Birk, The IUB 2004 Rescue Robot Team, in: Robocup 2004: Robot Soccer World Cup VIII, Springer, 2005.

[21] A. Birk, S. Carpin, Rescue Robotics – a crucial milestone on the road to autonomous systems, Advanced Robotics, accepted for publication.

[22] H. Kenn, S. Carpin, M. Pfingsthorn, B. Liebald, I. Hepes, C. Ciocov, A. Birk, Fast-robots: A rapid-prototyping framework for intelligent mobile robots, in: Proceedings of the 3rd International Conference on Artificial Intelligence and Applications, ACTA Press, 2003, pp. 76–81.