Merging occupancy grid maps from multiple robots

Andreas Birk *Member, IEEE*, Stefano Carpin *Member, IEEE*

***Abstract*— Mapping can potentially be speeded up in a significant way by using multiple robots exploring different parts of the environment. But the core question of multi-robot mapping is how to integrate the data of the different robots into a single global map. A significant amount of research exists in the area of multi-robot-mapping that deals with techniques to estimate the relative robots poses at the start or during the mapping process. With map merging the robots in contrast individually build local maps without any knowledge about their relative positions. The goal is then to identify regions of overlap at which the local maps can be joined together. A concrete approach to this idea is presented in form of a special similarity metric and a stochastic search algorithm. Given two maps m and m', the search algorithm transforms m' by rotations and translations to find a maximum overlap between m and m'. In doing so, the heuristic similarity metric guides the search algorithm toward optimal solutions. Results from experiments with up to six robots are presented based on simulated as well as real world map data.**

***Index Terms*— cooperation, mapping, stochastic search**

FINAL VERSION

```
@article{
MultiMap_IEEEproc,
Author = {Birk, Andreas and Carpin, Stefano},
Title = {{Merging occupancy grid maps
          from multiple robots}},
Journal = {IEEE Proceedings,
          special issue on Multi-Robot Systems},
Publisher = {IEEE Press},
Volume = {94},
Number = {7},
Pages = {1384-1397},
Year = {2006} }
```


I. INTRODUCTION

Autonomous mapping is one of the tasks that could benefit more from the effective deployment of cooperative multi-robot systems [1]. Teams of robots can bring more sensors, potentially heterogeneous ones, to the area where robots are performing their task. A properly designed team of robots can significantly reduce the time needed to map a given environment, since they can explore different parts in parallel. In addition, the overall team is more robust, since the failure of one of the robots is not doomed to hinder the overall mission. In order to maintain this robustness, distributed approaches are a must. In fact, each robot has to operate completely autonomously, and there should be no agents that have unique features, in terms of software or hardware, that make them critical for the mission success. In addition, scalability is an important factor. The addition of a new robot to the team should not require too much of restructuring or reconfiguration.

The authors are with the School of Engineering and Science, International University Bremen, 750561, D-28725 Bremen, Germany

Note that the ability to build a map of an unknown environment is one of the fundamental capabilities a robot must exhibit in order to operate outside the well designed and protected laboratory setting. In [2], Thrun provides an extensive coverage of single robot indoor mapping methods. In particular he outlines that "*mapping unstructured, dynamic, or large-scale environments remains largely an open research problem*". The mapping problem is often addressed together with the localization problem. The combination of the two is referred to as simultaneous localization and mapping problem (SLAM). In fact, once a map is given, the task of localizing the robot inside the map using its sensors inputs is solved [3][4]. Conversely, if the robot pose is known, building a map is also a task that can be effectively solved. When the two tasks have to be solved at the same time the problem becomes much more difficult. This motivates the different techniques that have been developed.

We can say that there exist two main approaches to address this challenge. SLAM can be solved using a Kalman filter based approach [5]. In this case the produced map presents the posterior probability of the location of some features (or landmarks) that can be detected by the robot while exploring the environment. The method has some drawbacks. For example, a limited number of features can be handled by the algorithm when building a map in real time while the robot is moving. A map with N features requires N^2 parameters, and this in turn implies that matrices with N^2 elements have to be processed (i.e. inverted and multiplied) at each iteration. In addition the Kalman filter approach relies on specific conditions on the superimposed noise (0 mean Gaussian noise), practically rarely verified. Another challenging aspect of Kalman filter based mapping methods is data association. The algorithm in fact has to be able to identify features from the sensed data, and to associated them with features previously inserted in the map. If no good association is possible, the algorithm has to decide that the observed feature is a new one, and it should be inserted in the map.

A different approach is based on the expectation maximization technique [6]. In this case the mapping task is solved using an algorithm based on the expectation maximization principle (EM). Differently from the previously described Kalman filter algorithm, EM based mapping will not produce a full posterior, but rather the most likely map. Another hard limitation is the fact that EM cannot generate maps incrementally, because of the iterative nature of the EM. On the other hand, EM based mapping is pretty insensitive to the data association problem, can be used to map huge environments, and, notably, can successfully map environments where loops or cycles are present.

The EM algorithm has been also used to address the multi-robot mapping [7][8]. However, some inherent limitations have

been outlined. In particular, it is necessary to assume that all the robots in the team start at positions where there is a significant overlap between their range scans, and in addition they must have an approximate knowledge of their relative positions. Also Kalman filter based approaches have been developed and implemented to address the multi-robot map and localization problems [9][10], as well as other general strategies for multi-robot exploration, mapping, and model acquisition [11][12][13][14]. The re-occurring pattern is the need of information about the relative positions of the robots. This can be in form of the start conditions [7][8], the identification of the position of a robot within the existing map of an other one [15][16] or explicit rendezvous strategies [17][18]. The related general line of research can be dubbed **distributed mapping**.

Here a totally opposite approach to the problem is taken, namely to completely ignore the issue of relative individual robot poses. Robots first operate for some time independently to generate individual local maps, for example in a scenario where multiple robots enter the same building from different locations. Possible applications include surveillance, search and rescue, military reconnaissance and the alike. After the local maps have been acquired, they are merged together to form a global map. According to Konolidge et al. this problem of **map merging**, "*is an interesting and difficult problem, which has not enjoyed the same attention that localization and map building have*" [19]. If the exact initial positions of the robots relative to each other would be known, the task would be trivial. But no information at all about the poses of the robots relative to each other is used here. Instead, regions that appear in more than one local map are used to transform the maps into a global one.

The rather limited amount of work on map merging has concentrated on feature based approaches [20][21][18][19], i.e., they rely on fixed landmarks that can be recognized through suited processing of the robots' data. So-called topological maps for example are graphs where the vertices represent recognizable places, e.g., doorways, different forms of junctions between hallways, and so on. An edge in a topological map represents a passage between the related two places. Map merging of topological maps m_1 and m_2 hence boils down to finding suited identical subgraphs in m_1 and m_2 [20].

Our goal is to combine together maps not based on features, but rather on occupancy grids, i.e., metric arrays where the value in each cell represents whether the related location is free space or part of an obstacle. As outlined by some authors [22], occupancy grids are the predominant paradigm used for environment modeling in robotics. They are indeed very effective when robots are required to explore and map unstructured environments where features extraction is hard to perform. In the rest of the article, we always refer to occupancy grids when using the term map.

Occupancy grids can be thought of as images where the information of whether a cell corresponds to free space or whether it is occupied is represented by a color. Map merging then corresponds to the problem of moving one of the images around until a part of it is aligned with an identical part in an other image. For this purpose we use a function borrowed from a metric ψ introduced before by Birk to measure the similarity of images [23]. There are several commonly used alternatives to ψ, for example computing some form of correlation like mean squared Euclidean distances between pixels of the same color or using special functions like the Hausdorff distance [24], [25]. Unlike these approaches, the similarity function ψ can be very efficiently computed as explained in detail later on. Furthermore, it provides meaningful gradients with respect to rotation, translation and registration, i.e., it can be used to guide a search algorithm to find a template in an image, a process known as registration [26], [27], [28], [29]. Registration has been employed previously in the context of occupancy grid mapping, namely for the purpose of improving localization [30]. In the work of Schultz and Adam, a small local occupancy grid is registered into a large global map to compensate the accumulative error due to an odometry based localization. This registration process searches only a small neighborhood of the current erroneous position of the robot to match the local map into the global one. It therefore can get along with relatively crude matching metrics.

The task of map merging is much harder than registration. Instead of locating a known template in an image, an unknown region of overlap has to be identified in two maps. This is comparable to image stitching [31], i.e., the technique that is for example used to generate panoramic views from several overlapping photographs. Solutions to solve this problem usually need common reference points that are either provided by the user by hand or identified using local image descriptors like intensity patterns [32], [33]. But occupancy grids lack rich textures like photographs. Also, we want to avoid any preprocessing of the map data for efficiency reasons. Hence, a simple combination of the image similarity ψ and a heuristic to identify the alignment of the overlap regions of two maps is used. Furthermore, there is the problem that there may be no overlap at all between the maps. This situation can be clearly identified within our approach, automatically indicating when map merging is prone to fail.

The rest of this article is structured as follows. The problem of map merging is introduced in a formal way in section 2. The search algorithm that is used in our approach is presented in section 3. Section 4 deals with the special heuristic to guide the search algorithm. In section 5, experiments and results of merging the maps from up to six robots into a single one are presented. Section 6 concludes the article.

II. Map Merging

A. Basic definitions

For sake of clearness, we formally define the problem we informally described in the introduction. We start with the definition of map.

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

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

We furthermore denote with $I_{N\times M}$ the set of $N \times M$ maps.

From a practical point of view, a discretization process is needed when a map is processed. This leads to a straightforward representation of a map as a matrix with N rows, M

columns, and storing integer numbers. 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. These assumptions are consistent with the literature on occupancy grids based maps.

We next define a planar transformation which will be used to try different relative placements of two maps to find a good merging. We assume that the location of a point in the plane is expressed in homogeneous coordinates, i.e. the point (x, y) is represented by the vector $[x\ y\ 1]^T$, where the trailing upper T indicates the transpose operation. 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:

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

As known [34], the transformation given in equation 1 corresponds to a counterclockwise rotation about the origin of the the point (x, y) of θ, followed by a translation of (t_x, t_y). We denote with τ the space of possible transformations. Obviously, additional transformations could be used if necessary. When the local maps are produced with approaches that have known deficiencies that for example insert distortions like bended geometries, shear transformations could be employed to generate proper matches. As we will see in the results section, state-of-the-art mapping algorithms produce occupancy grids where rotation and translation transformations are seemingly sufficient for merging real world data.

B. Pairwise Map Merging

In the map merging problem, given two partial maps we look for the transformation that gives the best merging. *Good merging* is defined in terms of overlapping between maps, and is captured by the following definition. The reader should note that the following definition assumes that maps are represented as matrices.

Definition 3: Let m_1 and m_2 be two maps in $I_{N\times M}$. The *overlapping* between m_1 and m_2 is

$$\omega(m_1, m_2) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} Eq(m_1[i, j], m_2[i, j]) \quad (2)$$

where $Eq(a, b)$ is 1 when $a = b$ and 0 otherwise.

The overlapping function ω measures how much two maps agree. In an ideal world, where robots would build maps which correspond to the ground truth and completely cover the operating environment, there exist a transformation which yields a perfect overlapping function, i.e. $\omega(m_1, m_2) = N \times M$. In the real applications this is obviously not the case, so the challenge is to find a transformation which gives the highest overlapping values.

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

Definition 4: Given $m_1 \in I_{N,M}$, $m_2 \in I_{N,M}$, determine the $\{x, y, \theta\}$-map transformation $T_{(x,y,\theta)}$ which maximizes $\omega(m_1, T_{x,y,\theta}(m_2))$.

The devised problem is clearly an optimization problem, where it is required to maximize a goal function, that in our case is the overlapping ω. The optimization has to be performed over a three dimensional space involving two translations and one rotation.

To solve this problem, there are many possible approaches. For example, any of the well-known optimization algorithms can be used to maximize $\omega()$. One problem is that $\omega()$ has many properties that make it badly suited for any optimization technique. The main drawback is that the values of $\omega(m_1, T_{x,y,\theta}(m_2))$ are arbitrarily spread over the space of transformations τ. The optimum may be located right next to the worst case in the search space, for example if the maps consist of spirals. The function $\omega()$ hence delivers no meaningful gradients that for example could be used by hill-climbing.

Therefore a heuristic function Δ is in general likely to be necessary to guide the search process. Δ should provide a kind of attraction between the overlap regions, hence providing some feedback in which direction the search algorithm should proceed. As mentioned in the introduction, several techniques from image registration and image stitching could be adapted for this purpose. Here, a metric ψ introduced before by Birk [23] to measure the similarity of images is used in combination with a heuristic to identify the alignment of the overlap regions. Also for the actual search algorithm, there are many alternatives. Here, Carpin's Gaussian Random Walk [35][36][37] is used for minimizing Δ by searching over τ.

C. Multi-robot Map Merging

Map merging as defined above deals with an integration of two maps into one, i.e., with data coming from two robots. The questions is now how to deal with real multi-robots, i.e., with $k > 2$ robots. The definitions can be extended in a straightforward way to deal with $k > 2$ robots:

Definition 5: Let $m_1, m_2, ..., m_k$ be k maps in $I_{N\times M}$. The *overlapping* between m_1 to m_k is

$$\omega(m_1, ..., m_k) = \sum_{i=0}^{N-1} \sum_{j=0}^{M-1} Eq(m_1[i, j], ..., m_k[i, j]) \quad (3)$$

where $Eq(a_1, ..., a_k)$ is 1 when $a_1 = a_2 = ... = a_k$ and 0 otherwise.

Definition 6: Given k maps m_1 to $m_k \in I_{N,M}$ determine the $k-1$ $\{x, y, \theta\}$-map transformations $T^1_{(x_1,y_1,\theta_2)}$ to $T^{k-1}_{(x_{k-1},y_{k-1},\theta_{k-1})}$ which maximize $\omega(m_1, T^1_{x_1,y_1,\theta_1}(m_2), ..., T^{k-1}_{(x_{k-1},y_{k-1},\theta_{k-1})}(m_k))$.

Fortunately, not only the definitions but also the map merging implementation can be easily extended to deal with $k > 2$ robots. To merge for example the data of some maps m_1 to m_4 of four robots, the simplest way to do so is to merge m_1 and m_2 to get m_{1+2} as well as m_3 and m_4 to get the map m_{3+4}. Then, the maps m_{1+2} and m_{3+4} can get pairwise merged to m_{1to4}. Given k robots this strategy takes $O(k)$ times the time for a pairwise merger. In section V presenting results from various experiments, it is shown that this approach is indeed very successful.

III. Stochastic search of transformations

A. Overview

The described optimization problem for the pairwise merger can be thought of as a process where one map stays fixed while the other one is moved around as a consequence of the differently tried transformation. The process is conceptually similar to the docking problem studied in computational biology. Given a protein, called receptor, and a ligand, the task is to find the so called binding pocket of the receptor. This means moving the ligand to a site where the overall energy of two compounds are minimized. Treating the ligand a rigid body, this problem is nothing but a search in a six dimensional space (three for rotations and three for translations).

Since a few years, there has been a trend to use robot motion planning algorithms to solve this sort of problems [38][39][40]. Though from the computational biology point of view the obtained results are still not comparable with state of the art molecular dynamics based approaches, significant progresses have been achieved. In particular, the algorithmic machinery developed along the years in the field of algorithmic motion planning proved to be suitable to be extend for this apparently unrelated problem. The most critical point is the following. In motion planning both the starting point and the end points are known. In the devised search problem, only the starting point is known. The goal configuration is obviously not available, since it is what we are looking for. There are many possible algorithms that can be used for this purpose. Here, Carpin's Adaptive Random Walk planner is used [35][36][37].

B. Adaptive Random Walk

Given a starting configuration, the algorithm explores the given configuration space using a random walk. At each step a random configuration is generated, and the corresponding heuristic Δ is computed.

The new configuration is generated using a Gaussian distribution whose mean μ_k and whose covariance Σ_k are updated at each step (hence the index k). The updating is a function of the last accepted point and of the last M values obtained for the heuristic Δ, M being one of the few parameters of the algorithm. The new sample is then retained or discarded according to the new value of Δ. Algorithm 1 illustrates the principle.

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: $c_0 \leftarrow \Delta(m_1, T_{t_{start}}(m_2))$
4: **while** $k < numSteps$ **do**
5: Generate a new sample $s \leftarrow t_k + v_k$
6: $c_s \leftarrow \Delta(m_1, T_s(m_2))$
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: $\Sigma_k \leftarrow$ Update$(t_k, t_{k-1}, t_{k-2}, \ldots, t_{k-M})$
10: $\mu_k \leftarrow$ Update$(x_k, t_{k-1}, t_{k-2}, \ldots, t_{k-M})$
11: **else**
12: discard the sample s

The RS function introduced in line 7 is a so called *Random Selector*. Its role is to allow the acceptance of a new sample even if its associated Δ value does not increase the obtained overlapping. The reason for this criteria is to avoid a behavior too similar to hill climbing, but rather like simulated annealing [41]. In fact, it can be proved that by properly tuning the RS function, simulated annealing and multipoint hill climbing are special cases of the adaptive random walk.

In case of a stochastic search algorithm, it is important to guarantee whether it will converge to the global optimum or not. The following theorem, whose proof is omitted, assures the convergence.

Theorem 1: Let $s^* \in S$ an element which maximizes $\Delta(m_1, T_s(m_2))$, and let $\{T_0, T_1, T_2 \ldots\}$ the sequence of transformations generated by the transformation random walk described in algorithm in equation 1. Let T_b^k be the best one generated among the first k transformations, i.e. the one yielding the highest value of Δ.

$$\lim_{k \to +\infty} \Pr[\Delta(m_1, T_b^k(m_2)) \neq \Delta(m_1, T_{s^*})(m_2))] = 0 \quad (4)$$

While reading the former definition, the reader should remember that given two maps, the optimal overlapping value is a finite natural number. It is also important to notice that the theorem only guarantees that when the processing time diverges, the optimal transformation will be found.

IV. The optimization function

A. Overview

As motivated in section II-B, the direct overlap $\omega()$ between two maps is not a very well suited function for guiding the search over the transformation space τ. The values of $\omega(m_1, T_{x,y,\theta}(m_2))$ are spread in an unsystematic way over τ. The optimum that we are looking for may be located right next to the worst case value. Therefore, any stochastic technique that tries to exploit gradient information from $\omega()$ is likely to perform poorly.

A fundamental aspect of our approach to map merging is hence the choice of the heuristic $\Delta()$. $\Delta()$ has two components. One is a metric ψ introduced to measure the similarity of images [23]. As already mentioned in the introduction, there is a large field in computer vision dealing with a problem

somewhat similar to map merging, namely so-called image registration. There are hence alternative metrics that could be used. The choice of ψ is mainly motivated by the fact that it can be very efficiently computed, namely in linear time. But map merging is a harder problem than image registration. Not only a template has to be identified in an image, respectively map, but two completely unknown regions have to be registered with each other. Note that overlapping regions may not even exist. Fortunately, there is a simple but very reliable indicator introduced in section IV-D that clearly indicates when a merger is not successful. Furthermore, ψ is supplemented by additional heuristic presented in section IV-C that identifies well aligned identical regions.

B. The image similarity ψ

Given two matrices m_1 and m_2 containing discrete values. The picture distance function ψ between m_1 and m_2 is defined as follows:

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

with

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

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.

In the work presented here, the matrices m_1 and m_2 are maps in form of occupancy grids. Probabilistic information in the cells representing beliefs is discarded, i.e., a cell is marked as either "free", "occupied" or "unknown". Only occupied and free cells are considered for computing ψ, $\mathcal{C} = \{occ, free\}$. Cells with unknown information are not of interest.

As mentioned before, a strong point about ψ is that it can be computed very efficiently. Concretely, 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 $p_1 = (x_1, y_1)$:

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

Figure 1 shows an example of a distance-map *d-map*$_c$ for a matrix m. Algorithm 2 shows the pseudo-code for the three steps carried out to built it. The underlying principle is illustrated in figure 2. First, all locations in *d-map*$_c$ where cells in m have the value c are set to zero. All other cells of *d-map*$_c$ are set to infinity; for a concrete implementation, ∞ can be substituted by any constant larger than $N \cdot M$ for an $N \times M$-matrix m. Then two relaxation steps follow. In the first one, a pass on *d-map*$_c$ starting from the upper left corner is carried out. During this pass, the value of each cell in *d-map*$_c$ is updated based on the current value of this cell and

0	0	0	0	1	2	3	3	2	1	0	0	1	2	2	1	0	0	1	2	1	1	1	1	2	3	4
0	0	1	1	2	3	4	4	3	2	1	1	2	2	1	1	1	1	2	1	0	0	0	0	1	2	3
0	0	1	2	3	4	4	3	3	2	2	2	2	1	0	0	1	2	3	2	1	1	0	0	1	2	3
1	1	2	3	4	4	3	2	2	1	1	1	2	2	1	1	2	3	2	1	0	0	1	1	2	3	4
2	2	3	4	4	3	2	1	1	0	0	0	1	2	2	2	3	2	2	1	1	1	1	2	3	4	5
3	3	4	4	3	2	1	0	0	0	0	0	0	1	2	3	2	1	1	0	0	0	0	1	2	3	4
4	4	4	3	2	1	1	0	0	0	0	0	0	1	2	2	1	0	0	0	0	1	1	2	3	3	3
5	4	3	2	1	0	0	0	0	0	1	1	1	2	3	2	1	1	1	1	1	2	2	3	3	2	2
4	3	2	1	1	0	0	0	0	0	1	2	2	3	2	1	0	0	0	0	1	2	3	3	2	1	1
3	2	1	0	0	0	1	1	1	1	2	3	3	4	3	2	1	1	1	1	2	3	3	2	1	0	0
3	2	1	0	0	1	2	2	2	2	3	3	3	3	2	1	0	0	0	0	1	2	3	2	1	0	0
4	3	2	1	1	2	2	2	2	2	2	2	2	2	1	0	0	0	0	0	0	1	2	1	0	0	1
5	4	3	2	2	2	1	1	1	1	1	1	1	2	1	1	1	0	0	0	0	0	1	2	1	1	2
6	5	4	3	2	1	0	0	0	0	0	0	0	1	2	2	2	1	0	0	0	0	1	2	2	3	
5	5	4	3	2	1	0	0	0	0	0	0	0	1	2	2	2	1	0	0	0	0	1	2	3	4	
4	4	5	4	3	2	1	0	0	0	0	1	1	1	1	1	1	1	0	0	0	0	1	2	3	4	5
3	3	4	5	4	3	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	2	3	4	4
2	2	3	4	5	4	3	2	1	1	1	0	0	0	0	1	1	1	1	1	2	2	3	3	3	3	4
1	1	2	3	4	5	4	3	2	2	2	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	3
0	0	1	2	3	4	4	3	2	2	2	2	2	1	0	0	1	2	3	2	1	1	1	1	1	1	2
0	0	1	2	3	4	3	2	1	1	1	1	2	2	1	1	2	3	2	1	0	0	0	0	0	0	1
0	0	1	2	3	3	2	1	0	0	0	0	1	2	2	2	3	3	2	1	0	0	1	1	0	0	1
1	1	2	3	4	4	3	2	1	1	1	0	0	1	2	3	4	3	2	1	0	0	1	1	0	0	1
2	2	3	4	5	5	4	3	2	1	0	0	1	2	3	4	4	3	2	1	0	0	0	0	0	0	1
3	3	4	5	5	4	3	2	1	0	0	1	2	3	4	5	5	4	3	2	1	1	1	1	1	1	2

Fig. 1. An example of a distance map that can be used as basis to efficiently compute the similarity between two maps. Given a matrix *m* with cells marked with a particular value *c*, here the color black (upper left figure). The distance map *d-map* (lower center figure) is a matrix containing in every cell the Manhattan distance to the nearest cell with value *c* in *m*. A gray-scale illustration of the *d-map* of *m* is shown in the upper right figure.

Algorithm 2 Computing *d-map*$_c$ for a matrix m

for $y \leftarrow -1$ to n **do**
  for $x \leftarrow -1$ to n **do**
    if $m(x, y) = c$ **then**
      d-map$_c[x][y] \leftarrow 0$
    else
      d-map$_c[x][y] \leftarrow \infty$
for $y \leftarrow 0$ to $n - 1$ **do**
  for $x \leftarrow 0$ to $n - 1$ **do**
    $h \leftarrow \min($*d-map*$_c[x-1][y]+1,$ *d-map*$_c[x][y-1]+1)$
    d-map$_c[x][y] = \min($*d-map*$_c[x][y], h)$
for $y \leftarrow n - 1$ downto 0 **do**
  for $x \leftarrow n - 1$ downto 0 **do**
    $h \leftarrow \min($*d-map*$_c[x+1][y]+1,$ *d-map*$_c[x][y+1]+1)$
    d-map$_c[x][y] = \min($*d-map*$_c[x][y], h)$

Algorithm 3 Computing $d(m_1, m_2, c)$

compute *d-map*$_c$ for m_2
$d(m_1, m_2, c) \leftarrow 0$
for $y \leftarrow -1$ to n **do**
  for $x \leftarrow -1$ to n **do**
    if $m_1(x, y) = c$ **then**
      $d(m_1, m_2, c) \leftarrow d(m_1, m_2, c) +$ *d-map*$_c[x][y]$

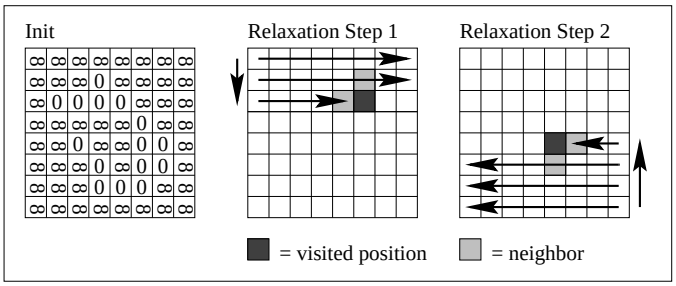

Fig. 2. The working principle for computing *d-map*$_c$. It is a simple relaxation algorithm that just takes one pass for initialization and two passes over the map for processing.

its left and upper neighbor. The second step is very similar to the first one, except that it starts in the lower right corner and that the value of each cell is updated based on the value of the cell and its right and lower neighbor. The distance-map *d-map*$_c$ for a map m can then be used as lookup-table for the computation of the sum over all cells in m_1 with value c, i.e., $d(m_1, m_2, c)$. The according code is illustrated in algorithm 3.

C. Map merging versus registration

Like any other image distance function, ψ is designed to work best for registration, i.e., for finding a template matrix m_T in an input matrix m_I. In an early version of this work, ψ was used as the sole component of Δ [42]. But for merging two maps m_1 and m_2, the situation is quite different from image registration. There is the need to identify a region r_1 in m_1 and a region r_2 in m_2 such that r_1 and r_2 register with each other to merge the maps. The problem is that there is usually no a priori information available about r_1 and r_2. There is even no guarantee that two according regions in m_1 and m_2 exist at all.

First, let us address the problem that overlapping regions r_1 and r_2 are usually smaller than the maps m_1 and m_2 themselves. We denote with m/r the set difference between m and r, e.g., m_1/r_1 is the set of cells of map m_1 excluding the ones from r_1. As illustrated in figure 3, ψ guides the search process to transform the maps toward an overlap of identical regions. As soon as identical regions r_1 and r_2 are aligned, their image distance $\psi(r_1, r_2)$ is zero or for noisy data at least close to zero. But though $\psi(r_1, r_2)$ is zero, there is still a positive image distance $\psi(m_1/r_1, m_2/r_2)$ between the other parts of the maps. It is therefore likely that the "optimization" continues to minimize ψ by trading small increases in $\psi(r_1, r_2)$ with larger decreases in $\psi(m_1/r_1, m_2/r_2)$, hence worsening the result in respect to the merging of the maps. The best solution would be to detect when $\psi(r_1, r_2)$ is zero for two sufficiently large regions r_1 and r_2. But an according check would require to compute $\psi(r_1, r_2)$ for every possible subset r_1 and r_2 of m_1 respectively m_2 in every transformation step.

An easy way out is to count the number of cells in m_1 and m_2 where there is agreement, respectively disagreement

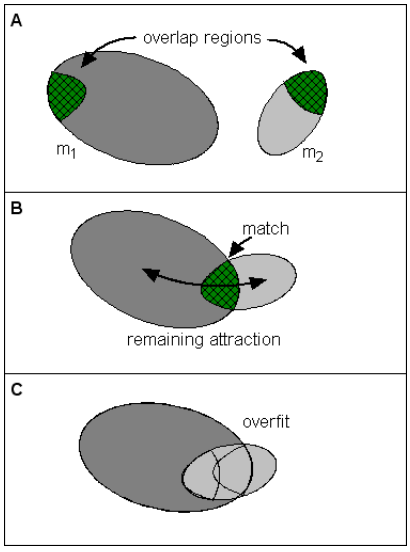

Fig. 3. The image distance function ψ generates a kind of attraction between identical regions r_1 and r_2 in two maps m_1 and m_2 (A), guiding a search algorithm to transform the maps to maximize similarity. This process should encounter a point in time where the identical regions are aligned (B). Then, ψ on these regions is zero indicating the overlap. Nevertheless, it is likely that there still is some attraction between other regions in m_1 and m_2 that have some similarity, e.g., some free space in some rooms. Using solely ψ for Δ is hence likely to lead to some additional "shifting"(C).

whether the cell is occupied or free:

$$agr(m_1, m_2) = \#\{p = (x, y) \mid m_1[p] = m_2[p] \in \mathcal{C}\}$$
$$dis(m_1, m_2) = \#\{p = (x, y) \mid m_1[p] \neq m_2[p] \in \mathcal{C}\}$$

Note that only information is used from map parts that are aligned with each other in the current transformation step. If the content of the cell at position $p = (x, y)$ in m_1 or m_2 is "unknown" then neither $agr()$ nor $dis()$ are affected. For every cell that is "free", respectively "occupied" at a position p in both m_1 and m_2, $agr()$ is incremented. The function $dis()$ is incremented when a cell at a location p is "free" in m_1 and "occupied" in m_2 or vice versa. The according computations can be done in a straightforward manner in linear time.

The function $agr()$ should be as large as possible, $dis()$ as small as possible. In the ideal case when two identical regions r_1 and r_2 are aligned then $dis() = 0$ and $agr()$ is the number of cells in r_1, respectively r_2, i.e., a positive integer that directly reflects the size of the overlap. Dissimilarity is to be minimized, hence $agr()$ is negatively taken into account for the according function Δ:

$$\Delta(m_1, m_2) = \psi(m_1, m_2) + c_{lock} \cdot (dis(m_1, m_2) - agr(m_1, m_2))$$

The constant $c_{lock} \geq 0$ is a scaling factor that allows to trade convergence speed with the amount of necessary overlap between the maps to compute a successful merger. If c_{lock} is zero then the merging algorithm will only merge maps that have a large amount of overlap. If c_{lock} is increased then smaller and smaller amounts of overlap are necessary to get a properly merged map. This is bought at the disadvantage that the time to compute the merging increases. The reason for this is simply that only ψ provides meaningful gradients

for the motion planning, whereas $dis(m_1, m_2) - agr(m_1, m_2)$ only "locks" the two maps in place as soon as the identical regions are aligned. By increasing c_{lock}, the influence of $dis(m_1, m_2) - agr(m_1, m_2)$ on Δ is increased and the influence of ψ decreases. Examples of the influence of c_{lock} are discussed in the results section.

D. Identifying failure

There remains the problem that there is no guarantee of any overlap between the maps that are to be merged. In this case, the algorithm will do its best and determine a "good" match that can only be wrong. Also, as a randomized search algorithm and a heuristic dissimilarity function is used, it can very well be that a bad "solution" is found. Fortunately, there is a very easy way to rule out cases where the merging of m_1 and m_2 failed. The so-called acceptance indicator $ai()$ is defined as

$$ai(m_1, m_2) = 1 - \frac{agr(m_1, m_2)}{agr(m_1, m_2) + dis(m_1, m_2)}$$

Only if $ai(m_1, m_2)$ is very close to 1.0 then there is an actual overlap between a region of m_1 and a region of m_2 that was successfully detected. The results discussed in the following section V show that the distinction between failed attempts and successful merging is indeed very easy. In all experiments, successful runs lead to an $ai()$ of well above 98% while the "best" failed attempt had an $ai()$ of well below 90%.

V. Experimental results

The map merging is implemented in C++ and run on a Pentium IV 2.2 GHz under Linux. Given a pair m and m' of maps, then the center of m is taken as origin of the world coordinate frame. The initial pose of m' in respect to the world frame is determined by transforming m' to 576 different poses that consist of 72 5^o rotations of the orientation of m' times 8 combinations of varying the origin of m' by $+100, 0, -100$ in x-, respectively y-direction. The 576 different poses of m' are evaluated via $\Delta()$ to determine the best one, which is chosen to be the starting point of the optimization process with the Adaptive Random Walk minimizing $\Delta()$. The optimization is stopped if there is no change in Δ for 2000 steps, which is considered as an indication of convergence. In our experiments, this led in several cases to a too early stop of the run, i.e., unsuccessful mergers. All of these unsuccessful runs hat an $a(i)$ between 43.06% and at most 87.33%, i.e., well below the $ai()$ of 98.83% that was the worst case for a successful run; hence failure is clearly detectable.

The implementation is in no way optimized for computation speed. All maps are for example embedded in 400×400 matrices for which every cell is processed, even if the majority of them contains "unknown" values and hence could be disregarded. The processing of the fixed sized matrices has the advantage that every transformation step takes the same amount of time, namely about 4 msec. This allows to compare the results in terms of steps while providing a direct link to the real processing time used. Table I shows the exact runtimes in step, which can be related to true time by multiplying with 4 msec, and the acceptance indicators of all merged maps presented in this section. Each run was hence finished within about a minute or two. For all successful mergers, deviations of the centers of the maps from ground truth are so small that they can not be determined for our experiments, i.e., they are in the order or even below the resolution of the grid cells of 25cm $\times$ 25cm in a building that extends over more than 1,800 m^2.

map	steps	$ai()$
m_{1+2}	14733	99.58%
m'_{3+4}	8345	75.01%
m''_{3+4}	41587	99.46%
m_{5+6}	9935	99.11%
m_{6+7}	37083	98.98%
m_{7+8}	16448	99.29%
m_{5+8}	11989	99.45%
m_{3to8}	81031	98.83%

TABLE I

The run times for merging the different maps on a Pentium IV 2.2 GHz and the related acceptance indicators. As can be nicely seen, m'_{3+4} with an $ai()$ far from 100% is clearly a failed attempt. Most of the maps are pairwise mergers except m_{3to8} where six robots collected the data for the final map.

The maps for the first set of experiments are generated in a special simulator (figure 4). It is based on the Unreal Game engine and it includes a physics engine and realistic noise models [43]. The robot-models in the simulator are based on the IUB rescue robots [44]. The environment is a detailed model of the R1 research building at the International University Bremen (IUB). In the image representation of the maps, the color green corresponds to "free", red to "occupied" and "white" to unknown space. Please note that all maps are horizontally aligned in the following figures for display purposes. As input for the map merging algorithm, the origins of the maps have various locations as well as orientations in respect to the global coordinate frame.

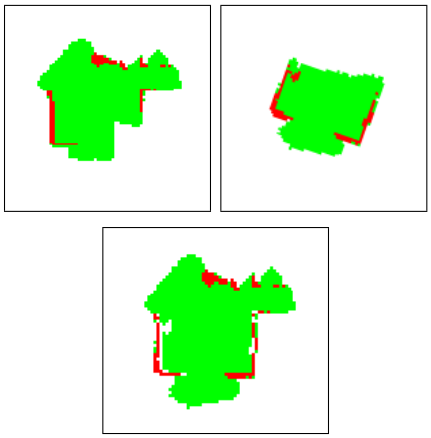

Fig. 5. The maps m_1 (upper left) and m_2 (upper right) showing parts of an entrance hall explored by two robots. The map m_{1+2} (lower center) shows the result of merging the two maps.

In figure 5 a relatively simple case of map merging is shown. The maps m_1 and m_2 have a rather large amount of overlap. In the same figure also the successful merger m_{1+2} of m_1 and

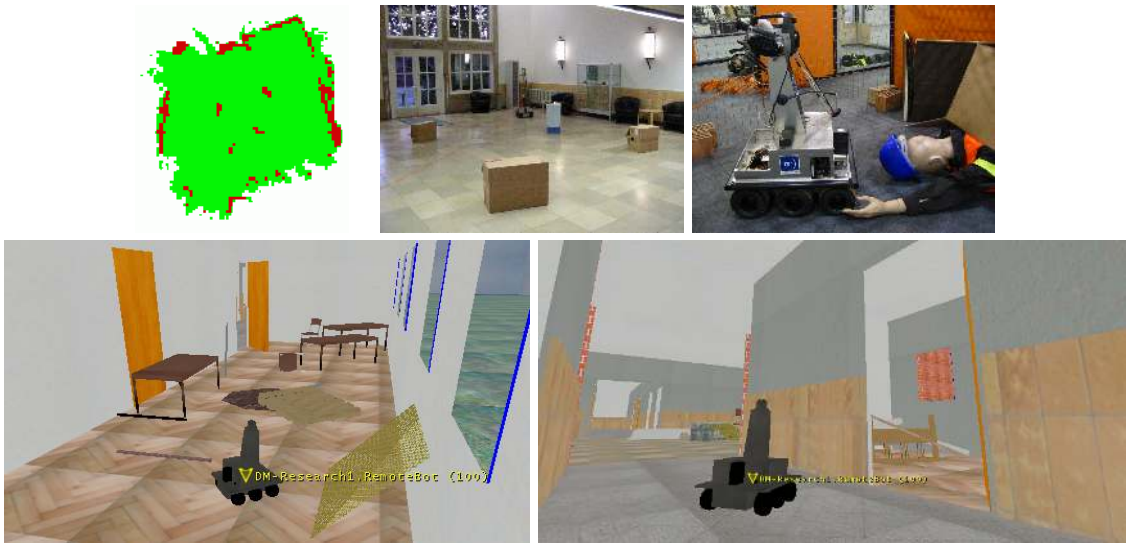

Fig. 4. The map shown in the upper left corner was generated with a real IUB rescue robot (upper right) in the R1 entrance hall with boxes as obstacles (upper middle). As mapping with the real robots is very time consuming, the maps for the experiments are generated in a simulator based on the Unreal Game engine (lower pictures). The simulation includes a physics engine and realistic noise models leading to realistic conditions, making real and simulated maps almost identical in terms of resolution and noise level.

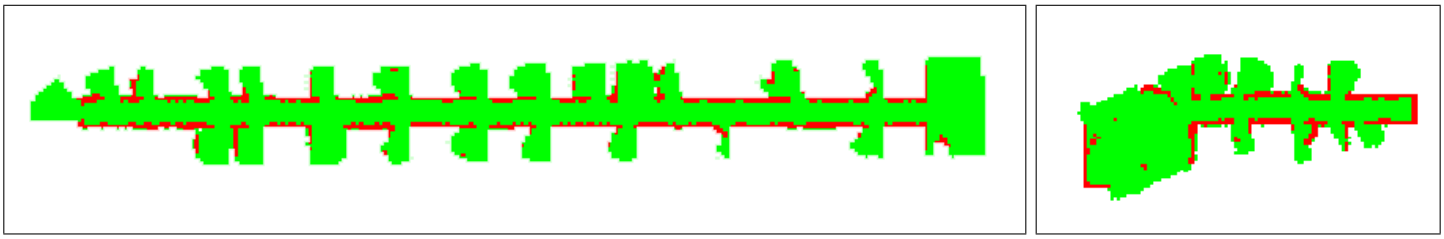

Fig. 6. The maps m_3 (left) and m_4 (right). They show two hallways originating from an entrance hall where both robots started and then wandered off in opposite directions.

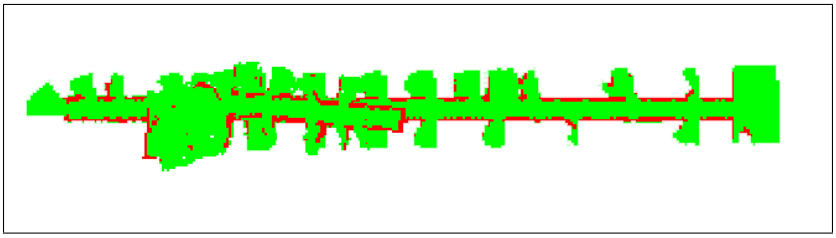

Fig. 7. This map m'_{3+4} shows the result of an unsuccessful attempt to merge m_3 and m_4. For demonstration purposes c_{lock} was set to zero, hence letting ψ overfit by finding the minimum dissimilarity of the overall maps. Note the an acceptance indicator of $ai(m_3, m_4) = 75.01\%$ clearly shows that this attempt failed.

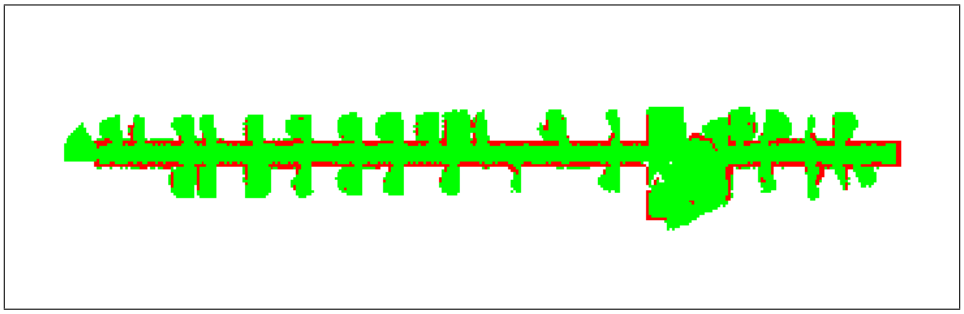

Fig. 8. In this map m''_{3+4} the successful result of merging m_3 and m_4 is shown. Note that this is a very difficult case as the overlap region only consists of 2.71% of the cells of m_3, respectively 8.54% of cells of m_4. This is also reflected by the high lock factor of $c_{lock} = 7.5$ used in this experiment. The acceptance indicator of $ai(m_3, m_4) = 99.46\%$ confirms the success.

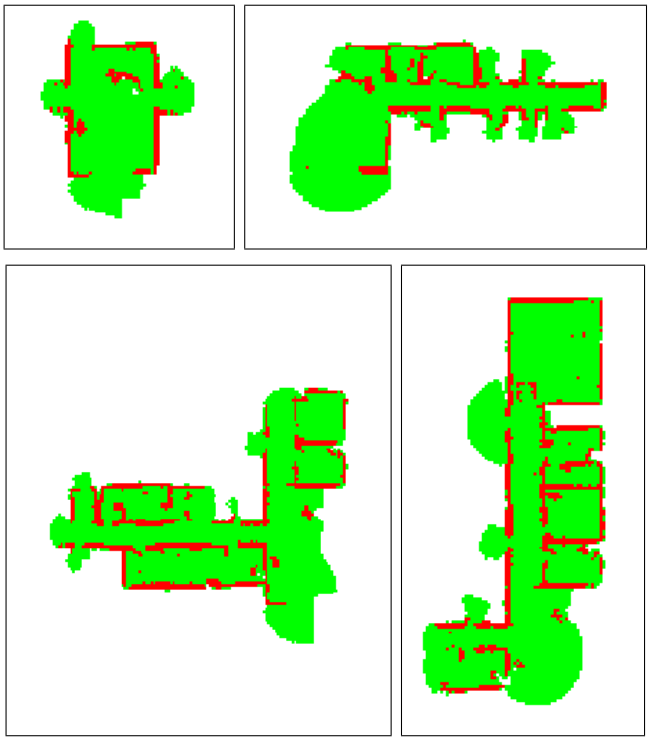

Fig. 9. The maps m_5 (upper left), m_6 (upper right), m_7 (lower left) and m_8 (lower right) gathered by four different robots. Please note that they are here horizontally aligned for the convenience of the reader. They all have different orientations and positions in respect to the global reference frame when the map merging starts.

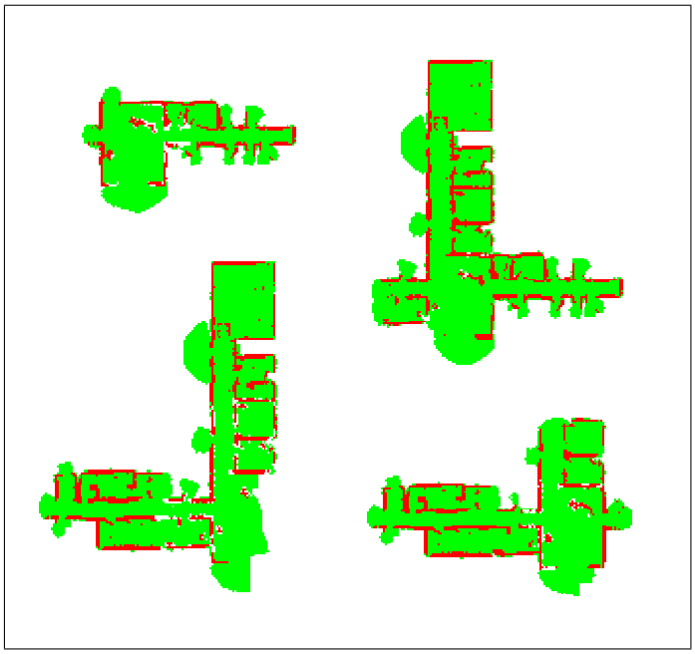

Fig. 10. The results of several pairwise mergers between the maps m_5 to m_8, namely m_{5+6} (upper left), m_{6+7} (upper right), m_{7+8} (lower left) and m_{1+8} (lower right), where m_{i+j} denotes the merger between map m_i and m_j.

m_2 is shown. For the rest of this section, we use the convention to denote the merger of two maps m_i and m_j with m_{i+j}. If $k > 2$ robots collected the maps m_X to m_{X+k}, the map that is generated from the k maps is denoted with $m_{Xto(X+k)}$.

The maps m_3 and m_4 shown in figure 6 pose a much greater challenge to map merging. Note that the overlap region only amounts to 2.71% of the cells of m_3, respectively 8.54% of cells of m_4. This case can serve as an excellent example of the influence of the constant c_{lock} on the algorithm and as an indication that the acceptance indicator indeed does its job. Small values of c_{lock} lead to a strong influence of ψ on Δ. In the run producing the map m'_{3+4} shown in figure 7, c_{lock} was even set to zero. As a consequence, the algorithm tries to transform m_4 to larger regions of similarity with m_3 and to find an "optimum" by mainly aligning large regions of free space. This leads to a small value for $\psi(m_3, m_4)$, but it is far from a usable result. Fortunately, an acceptance indicator of $ai(m_3, m_4) = 75.01\%$ indicates that this attempt indeed failed.

Figure 8 shows that it is possible to merge this extremely difficult case of m_3 and m_4 with our approach. Though it has to be admitted that this is not a typical result and that for this successful run there were several unsuccessful ones in this case. But it is of general interest for the quality of our approach that only the successful run had an optimal acceptance indicator of $ai(m_3, m_4) = 99.46\%$ and that for the other runs the acceptance indicators were well below 80%, thus clearly indicating failure. In addition to simply trying multiple times, the lock factor was increased in this experiment to $c_{lock} = 7.5$. By increasing the contribution of the check for overlap in Δ, mismatches get higher penalities. This advantage of ensuring proper overlap is bought at the disadvantage of increased computation time as the influence of ψ that delivers meaningful gradients for the search is lessened.

The results achieved with the maps m_5 to m_8 (figure 10) are again representative for the performance of the approach presented here. Figure 10 shows several pairwise mergers based on these maps. Finally, let us address the issue of true multi-robot research, i.e., of using more than two robots. As mentioned before, map merging scales very well with increasing numbers of robots. To merge the data of some maps $\hat{m}_1$ to $\hat{m}_4$ of four robots, the simplest way to do so is to merge $\hat{m}_1$ and $\hat{m}_2$ to get $\hat{m}_{1+2}$, $\hat{m}_3$ and $\hat{m}_4$ to get $\hat{m}_{3+4}$, and then $\hat{m}_{1+2}$ and $\hat{m}_{3+4}$ to get $\hat{m}_{1to4}$. Given k robots this strategy takes $O(k)$ times the time for a pairwise merger. The successful result of merging the maps m_3 to m_8 from six robots exploring the IUB R1 building is shown in figure 11. The resulting map m_{3to8} shows nicely the core structure of the building, which can not be recognized from any of the individual maps.

Last but not least, an additional experiment shall indicate that the presented map merging algorithm performs as well with real world robot data as with the maps generated in the high fidelity simulator. The real world data is taken from the Robotics Data Set Repository (Radish) [45]. It is based on the "ap_hill_07b" dataset provided by Andrew Howard, which contains the raw sensor data from four robots. The four individual maps were generated from the raw data with a state-of-the-art SLAM algorithm by Grisetti et al.[46]. The four input maps as well as the successful merger are shown in figure 12.

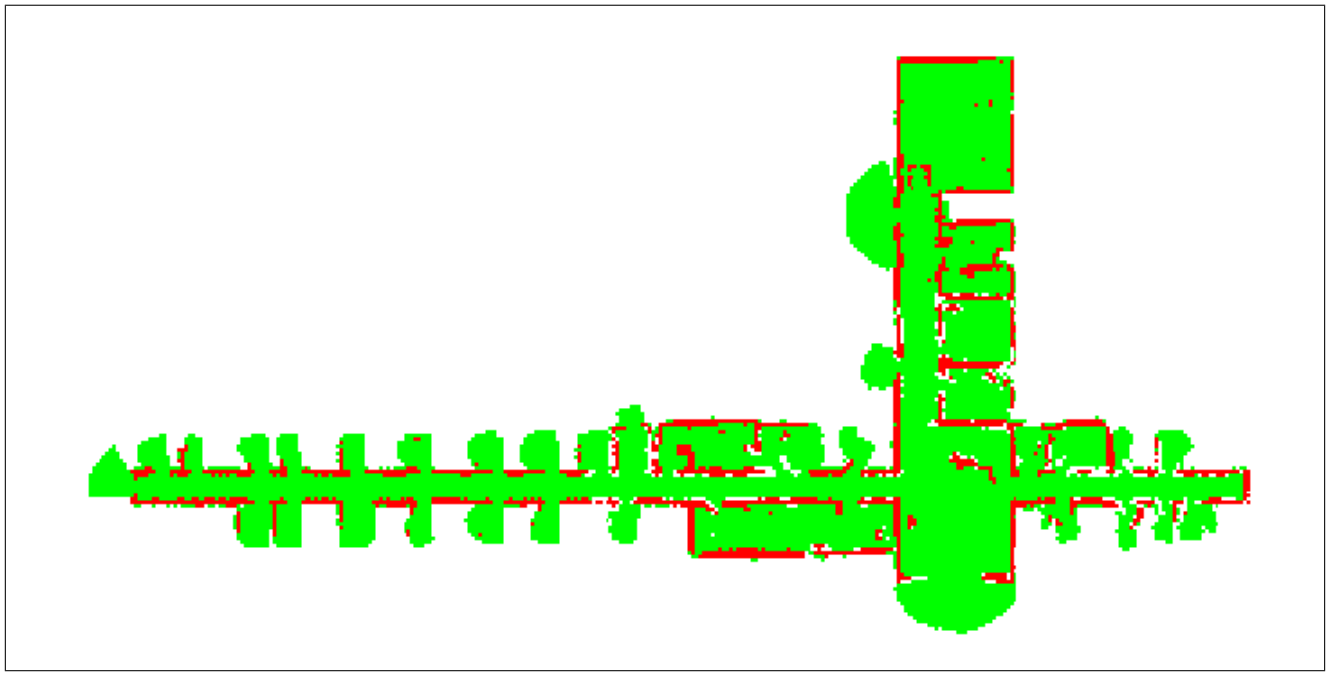

Fig. 11. The result m_{3to8} of merging the maps gathered by six robots exploring the environment. Unlike in the individual maps m_2 to m_8, the structure of the building becomes recognizable. Especially, the large entrance hall and the three corridors can be nicely identified.

For the successful merger of the real world maps, our algorithm performed much like in the previous experiments with the maps from the high fidelity simulator. The main difference is that the real world maps are much larger than the simulated ones, namely 1000x1000 grid cells in contrast to the 400x400 cells in the previous experiments. All parameters, except map size, of the algorithm were exactly the same as in the previous experiments with simulated map data. The computation time of each step hence went up to about 25 msec, mainly due to the increased time needed to compute Δ. The number of steps in each run in contrast was not significantly influenced by neither the larger map sizes nor by the fact that the data has been collected in a real world setting. The merger m'_{1+2} of the maps m'_1 and m'_2 (figure 12 top row) took 53772 steps. Maps m'_3 and m'_4 (figure 12 middle row) were merged in 29634 steps. The final result of map m'_{1-4} (figure 12 bottom row) was generated from m'_{1+2} and m'_{3+4} within 47822 steps.

VI. Conclusion

Multiple robots can be used at first glance in a straightforward way for mapping. Every robot can explore and map a different part of the environment. But the crucial question is how to integrate the data of the different robots. Here we take an approach that simply lets all robots operate individually and then tries to integrate the different local maps into a single global one, i.e., that does so-called map merging. The interesting aspect of the approach presented here is that it needs absolutely no information about the poses of the robots relative to each other. Instead, regions are identified that appear in more than one map. Such regions can then be used to "glue" the maps together.

Concretely, we measure the similarity between maps, as known from computer vision for example for image registration and stitching. This measurement can then be used to guide a search process that transforms one map to achieve a maximum overlap with a second one. There are many possible choices for both the similarity function as well as for the search algorithm. Here, a heuristic based on a special image similarity function is used that can be computed very efficiently. Adaptive Random Walking is used for the search process. Furthermore, a special function is introduced that can indicate whether the merging was successful or not. Last but not least, results were presented where maps from up to six robots are merged.

References

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

[2] S. Thrun, "Proper label: Thrunmapsurvey; robotic mapping: A survey," in *Exploring Artificial Intelligence in the New Millenium*, G. Lakemeyer and B. Nebel, Eds. Morgan Kaufmann, 2002.

[3] S. Thrun, D. Fox, W. Burgard, and F.Dellart, "Robust monte carlo localization for mobile robots," in *Artificial Intelligence*, 2001.

[4] C. Kwok, D. Fox, and M. Meila, "Real-time particle filters," *IEEE proceedings*, vol. 92, no. 3, pp. 469–484, 2004.

[5] G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba., "A solution to the simultaneous localisation and map building (slam) problem," *IEEE Transactions of Robotics and Automation*, vol. 17, no. 3, pp. 229–241, 2001.

[6] S. Thrun, D. Fox, and W. Burgard, "A probabilistic approach to concurrent mapping and localization for mobile robots," *Machine Learning*, vol. 31, pp. 29–53, 1998.

[7] S. Thrun, W. Burgard, and 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.

[8] S. Thrun, "A probabilistic online mapping algorithm for teams of mobile robots," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 335–363, 2001.

[9] R. Madhavan, K. Fregene, and L. Parker, "Terrain aided distributed heterogenous multirobot localization and mapping," *Autonomous Robots*, vol. 17, pp. 23–39, 2004.

Fig. 12. Four maps m'_1 to m'_4 based on real world data (top and middle) and the result of their successful merger m'_{1-4} (bottom). The raw sensor of the robots was taken from an open database, the Robotics Data Set Repository (Radish), and processed with a standard SLAM algorithm to produce the input maps.

[10] S. Roumeliotis and G. Bekey, "Distributed multirobot localization," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 781–795, 2002.

[11] N. Rao, "Terrain model acquisition by mobile robot teams and n-connectivity," in *Distributed Autonomous Systems 4*. Springer, 2000, pp. 231–240.

[12] G. Dedeoglu and G. Sukhatme, "Landmark-based matching algorithm for cooperative mapping by autonomous robots," in *Distributed Autonomous Robotic Systems 4*. Springer, 2000, pp. 251–260.

[13] M. D. Marco, A. Garulli, A. Giannitrapani, and A. Vicino, "Simultaneous localization and map building for a team of cooperative robots: a set membership approach," *IEEE Transactions on robotics and automation*, vol. 19, no. 2, pp. 238–249, 2003.

[14] A. Howard, "Multi-robot mapping using manifold representations," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004, pp. 4198–4203.

[15] S. Williams, G. Dissanayake, and H. Durrant-Whyte, "Towards multi-vehicle simultaneous localisation and mapping," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA*. IEEE Computer Society Press, 2002.

[16] J. Fenwick, P. Newman, and J. Leonard, "Cooperative concurrent mapping and localization," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA*. IEEE Computer Society Press, 2002.

[17] N. Roy and G. Dudek, "Collaborative exploration and rendezvous: Algorithms, performance bounds and observations," *Autonomous Robots*, vol. 11, 2001.

[18] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai, "A practical, decision-theoretic approach to multi-robot mapping and exploration," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003.

[19] K. Konolige, D. Fox, B. Limketkai, J. Ko, and 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.

[20] W. H. Huang and K. R. Beevers, "Topological map merging," in *Proceedings of the 7th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2004.

[21] G. Dedeoglu and G. Sukhatme, "Landmark-based matching algorithm for cooperative mapping by autonomous robots," in *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2000.

[22] S. Thrun, "Learning occupancy grids with forward sensor models," *Autonomous Robots*, vol. 15, pp. 111–127, 2003.

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

[24] W. J. Rucklidge, "Efficiently locating objects using the hausdorff distance," *International Journal of Computer Vision*, vol. 24, no. 3, pp. 251–270, 1997.

[25] D. P. Huttenlocher and W. J. Rucklidge, "A multi-resolution technique for comparing images using the hausdorff distance," in *Proceedings Computer Vision and Pattern Recognition*, New York, NY, 1993, pp. 705–706.

[26] D. Stricker, "Tracking with reference images: a real-time and markerless tracking solution for out-door augmented reality applications," in *Proceedings of the 2001 conference on Virtual reality, archeology, and cultural heritage*. ACM Press, 2001, pp. 77–82.

[27] C. Dorai, G. Wang, A. K. Jain, and C. Mercer, "Registration and integration of multiple object views for 3d model construction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 1, pp. 83–89, Jan., 1998.

[28] L. G. Brown, "A survey of image registration techniques," *ACM Comput. Surv.*, vol. 24, no. 4, pp. 325–376, 1992.

[29] S. Alliney and C. Morandi, "Digital image registration using projections," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 2, pp. 222–233, 1986.

[30] A. C. Schultz and W. Adams, "Continuous localization using evidence grids," in *Proceedings of the 1998 IEEE International Conference on Robotics and Automation, ICRA*. IEEE Computer Society Press, 1998, pp. 2833–2839.

[31] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[32] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," in *Proceedings of Computer Vision and Pattern Recognition*, June, 2003.

[33] L. V. Gool, T. Moons, and D. Ungureanu, "Affine/photometric invariants for planar intensity patterns," in *Proceedings of European Conference on Computer Vision*, 1996.

[34] J. J. Craig, *Introduction to robotics – Mechanics and control*. Prentice Hall, 2005.

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

[36] ——, "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.

[37] ——, "Motion planning using adaptive random walks," *IEEE Transactions on Robotics*, vol. 21, no. 1, pp. 129–136, 2005.
[38] G. Song and N. Amato, "A motion-planning approach to folding: From paper craft to protein folding," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 1, pp. 60–71, 2004.
[39] A. Singh, J. Latombe, and 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.
[40] G. Chirikjian, N. Amato, and L. K. (Eds.), "Special issue: Robotics techniques applied to computational biology," in *International Journal of Robotics Research*, 2004–to appear.
[41] S. Kirkpatrick, C. Gelatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[42] S. Carpin, A. Birk, and V. Jucikas, "On map merging," *International Journal of Robotics and Autonomous Systems*, vol. 53, pp. 1–14, 2005.
[43] S. Carpin, A. Birk, M. Lewis, and A. Jacoff, "High fidelity tools for rescue robotics: results and perspectives," in *RoboCup 2005: Robot Soccer World Cup IX*, ser. Lecture Notes in Artificial Intelligence (LNAI), I. Noda, A. Jacoff, A. Bredenfeld, and Y. Takahashi, Eds. Springer, 2006.
[44] A. Birk, "The IUB 2004 rescue robot team," in *RoboCup 2004: Robot Soccer World Cup VIII*, ser. Lecture Notes in Artificial Intelligence (LNAI), D. Nardi, M. Riedmiller, and C. Sammut, Eds. Springer, 2005, vol. 3276.
[45] A. Howard and N. Roy, "The robotics data set repository (radish)," 2003.
[46] G. Grisetti, C. Stachniss, and W. Burgard, "Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, 2005.