RAWSEEDS

Robotics Advancement through Web-publishing of Sensorial and Elaborated Extensive Data Sets

WorkPackage 5

Deliverable D5.2
Final Benchmark Solutions

Project no. 045144
Instrument: Specific Support Action
Thematic Priority: IST-2005-2.6.1 Advanced Robotics

date due:	end of month 33 (June 30st, 2009)
authors:	Michael Ruhnke, Giorgio Grisetti, Cyrill Stachniss, ALUFR
internal reviewers:	Giorgio Grisetti, ALUFR
	Matteo Matteucci, POLIMI
	Domenico Sorrenti, UNIMIB

contributors:
Giorgio Grisetti, Michael Ruhnke, Cyrill Stachniss, Wolfram Burgard, ALUFR
César Cadena, José A. Castellanos, Javier Civera, Dorian Gálvez, Oscar García, José M. M. Montiel, Ana C. Murillo, José Neira, Pedro Piniés, UNIZAR
Domenico Sorrenti, UNIMIB

Date of preparation of this document: Wed, September 30, 2009

Contents

1 Introduction

This deliverable reports about the *benchmark solutions* (BS) for the RAWSEEDS project. According to the naming convention used in the project a *benchmark solution* consists in:

- The description and the software implementation of a SLAM algorithm.
- The output of the algorithm on a given dataset. Such a dataset is named *benchmarking problem* according to the Annex I.
- Computation of the score according to a quality measure for evaluating the output of the algorithm according to the benchmarking problem.

In the remainder of this document, we first recall the concept at the base of our performance metric. Subsequently, we describe the set of algorithms provided by the members of the RAWSEEDS consortium. For each of these algorithms, we present a set of benchmark solutions. We grouped the benchmarking solutions in this document based on the sensor used by the corresponding algorithms (laser data as well as monocular, stereo, and trinocular camera data) .

2 Performance Metric

This section gives a brief review of the used metric which is included in D5.2 to make this document self-containing but should not be regarded as a contribution of D5.2.

2.1 Absolute Trajectory Error

ATE compares the trajectory of a robot, as reconstructed by an algorithm using real sensor data as its input, to the actual trajectory (ground truth). ATE is a mandatory performance measure.

2.2 Mapping Error

ME compares the map of an environment, as reconstructed by an algorithm using real sensor data as its input, to the actual map of the location (ground truth). ME is a recommended performance measure.

2.3 Relative Pose Error

RPE measures the accuracy of a SLAM result, as reconstructed by an algorithm using real sensor data as its input, by comparing the reconstructed relative transformations between nearby poses to the actual relative transformations (ground truth). RPE is a recommended performance measure.

2.4 Rough Estimate of Complexity

REC provides a basic estimate of how the running time of an algorithm (which uses real sensor data as its input) scales as the quantity of data available to be processed increases. REC is a mandatory performance measure.

2.5 Self Localization Error

SLE aims to evaluate the overall quality of a SLAM algorithm by actually using its output in a realistic application. The SLAM algorithm, fed with real sensor data from a robot, is used to build a map of the explored environment; then a self-localization algorithm, fed with different sensor data streams collected in the same environment, is used to localize the robot within the map. The precision of such localization is evaluated by comparing it with the actual pose of the robot (ground truth). SLE is a recommended performance measure.

3 Benchmark Solutions: Laser-Based SLAM

In this section, we present benchmarking solutions of laser-based SLAM on three different estimation algorithms and different datasets. As algorithms, we consider scan matching as well as two state-of-the-art SLAM approaches for learning 2D grid maps from a sequence of laser observations and odometry measurements.

3.1 Scan Matching

Scan matching is the computation of the incremental, open loop maximum likelihood trajectory of the robot by matching consecutive scans [29, 7].

The general idea of this approaches can be summarized as follows. At any point $t-1$ in time, the robot is given an estimate of its pose $\hat{x}_{t-1}$ and a map $\hat{m}(\hat{x}_{1:t-1}, z_{1:t-1})$, constructed using the incremental trajectory estimate $\hat{x}_{1:t-1}$. After the robot moves further on and after taking a new measurement z_t, the robot determines the most likely new pose $\hat{x}_t$ as

$$\hat{x}_t = \underset{x_t}{\operatorname{argmax}} \left[p(z_t \mid x_t, \hat{m}(\hat{x}_{1:t-1}, z_{1:t-1})) \cdot p(x_t \mid u_{t-1}, \hat{x}_{t-1}) \right]. \quad (1)$$

The idea is to trade off the consistency of the measurement with the map (first term on the right-hand side in (1)) and the consistency of the new pose with the control action and the previous pose (second term on the right-hand side in (1)). The map is then extended by the new measurement z_t, using the pose $\hat{x}_t$ as the pose at which this measurement was taken. The key limitation of these approaches lies in the greedy maximization step. Once the location x_t at time t has been computed it is not revised afterward so that the robot cannot recover from errors affecting the past pose from which the map is computed (registration errors). Although they have been proved to be able to correct enormous errors in odometry, the resulting maps often are globally inconsistent,

In small environments, a scan matching algorithm is generally sufficient to obtain accurate maps with a comparably small computational effort. However, the estimate of the robot trajectory computed by scan matching is affected by an increasing error which becomes visible whenever the robot reenters in known regions after visiting large unknown areas (loop closure or place revisiting).

3.2 Rao-Blackwellized Particle Filters for Map Learning

Particle filters are a frequently used technique in robotics for dynamical system estimation. They have been used to localize robots [14], to build both feature-maps [34, 36] and grid-maps [16, 22, 24], and to track objects based on vision data [25]. A particle filter approximates the posterior by a set of random samples and updates it in a recursive way. The particle filter framework specifies how to update the sample set but leaves open how to choose the so-called proposal distribution. The proposal is used to draw the next generation of samples at the subsequent time step in the dynamical process. In practice, the design of the proposal has a major influence on the performance and robustness of the filtering process. On the one hand, the closer the proposal is to the target distribution, the better is the estimation performance of the filter. On the other hand, the computational complexity of the calculation of the proposal distribution should be small in order to run the filter online. For this reason, the majority of particle filter applications restrict the proposal distribution to a Gaussian since one can efficiently draw samples from such a distribution.

In our recent work [46] (see attachment to D5.1), we analyzed how well such Gaussian proposal distributions approximate the optimal proposal in the context of mapping. It turns out that Gaussians are often an appropriate choice but there exist situations in which multi-modal distributions are needed to appropriately sample the next generation of particles. Based on this insight, we present an alternative sampling technique that can appropriately capture distributions with multiple modes, resulting in more robust mapping systems.

The mapping system has been implemented and is available as an open source implementation under the name GMapping at [47]. GMapping applies a particle filter that requires three sequential steps to update its estimate. Firstly, one draws the next generation of samples from the so-called proposal distribution π. Secondly, one assigns a weight to each sample. The weights account for the fact that the proposal distribution is in general not equal to the target distribution. The third step is the resampling step in which the target distribution is obtained from the weighted proposal by drawing particles according to their weight.

In the context of the SLAM problem, one aims to estimate the trajectory of the robot as well as a map of the environment. The key idea of a Rao-Blackwellized particle filter for SLAM is to separate the estimate of the trajectory $x_{1:t}$ of the robot from the map m of the environment. This is done by the following factorization

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) \quad = \quad p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} \mid z_{1:t}, u_{1:t-1}), \tag{2}$$

where $z_{1:t}$ is the observation sequence and $u_{1:t-1}$ the odometry information. In practice, the first term of Eq. (2) is estimated using a particle filter and the second term turns into "mapping with known poses".

One of the main challenges in particle filtering is to choose an appropriate proposal distribution. The closer the proposal is to the true target distribution, the more precise is the estimate represented by the sample set. Typically, one requires the proposal π to fulfill the assumption

$$\pi(x_{1:t} \mid z_{1:t}, u_{1:t-1}) \;=\; \pi(x_t \mid x_{1:t-1}, z_{1:t}, u_{1:t-1})\pi(x_{1:t-1} \mid z_{1:t-1}, u_{1:t-2}). \tag{3}$$

According to Doucet [15], the distribution

$$p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) \;=\; \frac{p(z_t \mid m_{t-1}^{(i)}, x_t)p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})} \tag{4}$$

is the optimal proposal for particle i with respect to the *variance of the particle weights* that satisfies Eq. (3). This proposal minimizes the degeneracy of the algorithm (Proposition 4 in [15]). As a result, the computation of the weights turn into

$$\begin{aligned}
w_t^{(i)} &= w_{t-1}^{(i)} \frac{\eta p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)})p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1})} && (5)\\
&\propto w_{t-1}^{(i)} \frac{p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)})p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{\frac{p(z_t \mid m_{t-1}^{(i)}, x_t)p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})}} && (6)\\
&= w_{t-1}^{(i)} \cdot p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}) && (7)\\
&= w_{t-1}^{(i)} \cdot \int p(z_t \mid x')p(x' \mid x_{t-1}^{(i)}, u_{t-1})\, dx'. && (8)
\end{aligned}$$

Unfortunately, the optimal proposal distribution is in general not available in closed form or in a suitable form for efficient sampling. As a result, most efficient mapping techniques use a Gaussian approximation of the optimal proposal. This approximation is easy to compute and allows the robot to sample efficiently. As we will showed in [46], the Gaussian assumption is not always justified. Therefore, we implemented a technique that is similar to the Gaussian proposal approximation but is still able to cover multiple modes.

Our approach is equivalent to computing a sum of weighted Gaussians to model the proposal but does not require the explicit computation of a sum of Gaussians.

Our previous method [22] first applies scan matching on a per-particle basis. It then computes a Gaussian proposal *for each sample* by evaluating poses around the pose reported by the scan-matcher. This technique yields accurate results in case of a uni-modal distribution, but encounters problems in that it focuses only on the dominant mode to which the scan matching process converges. The left image in Figure 1 illustrates an example in which the scan matching process converges to the dominant peak denoted as "mode 1". As a result, the Gaussian proposal samples only from this mode and at most a few particles cover "mode 2" (and only if the modes are spatially close). Even if such situations are rarely encountered in practice, we found in our experiments that they are one of the major reasons for filter divergence.

One of the key ideas integrated into our new approach is to adapt the scan matching/sampling procedure to better deal with multiple modes. It consists of a two step sampling. First, only the odometry motion model is used to propagate the samples. This technique is

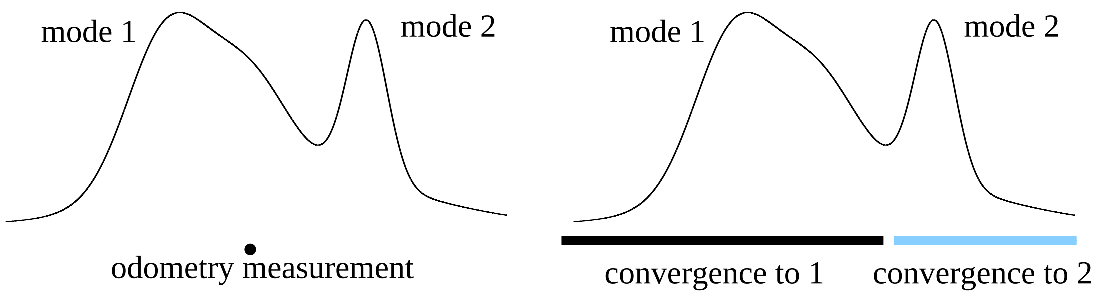


Figure 1: The left image illustrates a 1D likelihood function and an odometry measurement. Conventional informed sampling first performs scan matching starting from the odometry measurement. In this situation, the scan-matcher will find a local peak in the likelihood function (most likely mode 1) and the future sample will be drawn from a Gaussian centered at this single mode. The right image illustrates the new approach. It draws the sample first from the odometry model and applies scan matching afterwards. When a drawn sample falls into the area colored black, the scan-matcher will converge to mode 1, otherwise, it will converge to mode 2. By sampling first from the odometry, then applying scan matching, and finally computing local Gaussian approximations, multiple modes in the likelihood function are likely to be covered by the overall sample set.

known from standard Monte-Carlo localization approaches (c.f. [14]) and allows the particles to cover possible movements of the robot. In a second step, gradient descent scan matching is applied based on the observation likelihood and the denominator of Eq. (4). As a result, each sample converges to the mode in the likelihood function that is closest to its own starting position. Since the individual particles start from different locations, they are likely to cover the different modes in their corresponding likelihood functions as illustrated in the right image of Figure 1. Our approach leads to sample sets distributed according to a Gaussian *around the modes* in the observation likelihood functions. As we demonstrated in the experimental results in [46], this technique leads to proposal distributions which are closer to the optimal proposal given in Eq. (4) than the Gaussian approximations; when the distribution has only a single mode, the solution is equivalent to previous approaches [22].

Experimental results obtained with GMapping are depicted in Section 3.4 of this deliverable.

3.3 Graph-Based SLAM

Approaches to graph-based SLAM focus on estimating the most likely configuration of the nodes and are therefore referred to as maximum-likelihood (ML) techniques [18, 30, 41]. The approach briefly described here belongs to this class of methods. For a detailed description see [23, 20, 21, 48].

The goal of graph-based ML mapping algorithms is to find the configuration of the nodes that maximizes the likelihood of the observations. Let $\mathbf{x} = (x_1 \ \cdots \ x_n)^T$ be a vector of parameters which describes a configuration of the nodes. Let δ_{ji} and Ω_{ji} be respectively the mean and the information matrix of an observation of node j seen from node i. Let $f_{ji}(\mathbf{x})$

be a function that computes a zero noise observation according to the current configuration of the nodes j and i.

Given a constraint between node j and node i, we can define the *error* e_{ji} introduced by the constraint as

$$e_{ji}(\mathbf{x}) \quad = \quad f_{ji}(\mathbf{x}) - \delta_{ji} \tag{9}$$

as well as the *residual* $r_{ji} = -e_{ji}(\mathbf{x})$. Let $\mathcal{C} = \{\langle j_1, i_1 \rangle, \ldots, \langle j_M, i_M \rangle\}$ be the set of pairs of indices for which a constraint $\delta_{j_m i_m}$ exists. The goal of a ML approach is to find the configuration $\mathbf{x}^*$ of the nodes that minimized the negative log likelihood of the observations. Assuming the constraints to be independent, this can be written as

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \sum_{\langle j,i \rangle \in \mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). \tag{10}$$

Solving the SLAM problem in its graph-based formulation requires to address two problems:

- Determining a set of spatial relations δ_{ij} between adjacent robot positions from the laser observations. This step is often referred to as *graph construction*,
- Computing the configuration of poses $\mathbf{x}^*$ which best explain the pairwise relations in the graph and it is called *graph optimization* or *network optimization*.

In the remainder of this section we first discuss how to construct the graph from a sequence of laser range observations. Subsequently we introduce our novel graph optimization approach which is based on stochastic gradient descent. Our approach reaches higher convergence speeds by embedding the loopy structure of the SLAM problem in the parametrization of the graph.

Graph Construction To construct the graph of relations out of a sequence of measurements we determine the relative motion between subsequent scans by refining the odometry position via scan matching. In other words, given a pair of subsequent robot positions x_i and x_{i+1}, we determine the transformation $\delta_{i+1,i}$ as

$$\delta_{i+1,i} = \operatorname*{argmax}_{\delta} p(\delta \mid z_i, z_{i+1}, u_i). \tag{11}$$

Here z_i and z_{i+1} are the laser readings acquired at the times i and $i+1$ and u_i is the odometry measurement between the pose x_i and the pose x_{i+1}. $\delta i + 1, i$ represents the transformation which leads to the best overlap of the scans z_{i+1} and z_i, under the constraint derived from the odometry measurement u_i. To determine these constraints we use the scan matching algorithm *vasco* which is part of the open source navigation framework CARMEN [35].

We determine the so called loop closure constraints $\delta_{j,i}$ between the current robot location x_j and some previous location x_i by running Monte Carlo Localization [14] in a grid map obtained from all scans which intersect the current uncertainty ellipse of the robot pose. When MCL converges, we select the previous node x_i which is closer to the MCL estimate x_j^* of x_j and we add a new constraint $\delta_{j,i} = x_j \ominus x_i$ to the graph.

Network Optimization using Stochastic Gradient Descent Olson *et al.* [41] propose to use a variant of the preconditioned stochastic gradient descent (SGD) to address the compute the most likely configuration of the network's nodes. The approach minimizes Eq. (10) by iteratively selecting a constraint $\langle j, i \rangle$ and by moving the nodes of the network in order to decrease the error introduced by the selected constraint. Compared to the standard formulation of gradient descent, the constraints are not optimized as a whole but individually. The nodes are updated according to the following equation:

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^{t} + \lambda \cdot \mathbf{H}^{-1} J_{ji}^{T} \Omega_{ji} r_{ji} \tag{12}$$

Here $\mathbf{x}$ is the set of variables describing the locations of the poses in the network and $\mathbf{H}^{-1}$ is a preconditioning matrix. J_{ji} is the Jacobian of f_{ji}, Ω_{ji} is the information matrix capturing the uncertainty of the observation, r_{ji} is the residual, and λ is the learning rate which decreases with the iteration. For a detailed explanation of Eq. (12), we refer the reader to our previous works [23, 41].

In practice, the algorithm decomposes the overall problem into many smaller problems by optimizing subsets of nodes, one subset for each constraint. Whenever time a solution for one of these sub problems is found, the network is updated accordingly. Obviously, updating the different constraints one after each other can have antagonistic effects on the corresponding subsets of variables. To avoid infinitive oscillations, one uses the learning rate λ to reduce the fraction of the residual which is used for updating the variables. This makes the solutions of the different sub-problems to asymptotically converge towards an equilibrium point that is the solution reported by the algorithm.

Tree Parametrization The poses $\mathbf{p} = \{p_1, \ldots, p_n\}$ of the nodes define the configuration of the network. The poses can be described by a vector of *parameters* $\mathbf{x}$ such that a bidirectional mapping between $\mathbf{p}$ and $\mathbf{x}$ exists. The parametrization defines the subset of variables that are modified when updating a constraint. Therefore, the way the nodes are parametrized has a serious influence on the performance of the system. We proposed to use a tree [23] as an efficient way of parametrize the nodes. One can construct a spanning tree (not necessarily a minimum one) from the graph of poses. Given such a tree, we define the parametrization for a node as

$$x_i \quad = \quad p_i - p_{\mathrm{parent}(i)}, \tag{13}$$

where $p_{\mathrm{parent}(i)}$ refers to the parent of node i in the spanning tree. As defined in Eq. (13), the tree stores the differences between poses. This is similar in the spirit to the incremental representation used in the Olson's original formulation, in that the difference in pose positions (in global coordinates) is used rather than pose-relative coordinates or rigid body transformations.

To obtain the difference between two arbitrary nodes based on the tree, one needs to traverse the tree from the first node upwards to the first common ancestor of both nodes and then downwards to the second node. The same holds for computing the error of a constraint. We refer to the nodes one needs to traverse on the tree as the path of a constraint. For example, $\mathcal{P}_{ji}$ is the path from node i to node j for the constraint $\langle j, i \rangle$. The path can be

divided into an ascending part $\mathcal{P}_{ji}^{[-]}$ of the path starting from node i and a descending part $\mathcal{P}_{ji}^{[+]}$ to node j. We can then compute the residual in the global frame by

$$r'_{ji} = \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} x_{k^{[-]}} - \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} x_{k^{[+]}} + R_i \delta_{ji}. \tag{14}$$

Here R_i is the homogeneous rotation matrix of the pose p_i. It can be computed according to the structure of the tree as the product of the individual rotation matrices along the path to the root. Note that this tree does not replace the graph as an internal representation. The tree only defines the parametrization of the nodes.

Let $\Omega'_{ji} = R_i \Omega_{ji} R_i^T$ be the information matrix of a constraint in the global frame. According to [41], we compute an approximation of the Jacobian as

$$J'_{ji} = \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} \mathcal{I}_{k^{[+]}} - \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} \mathcal{I}_{k^{[-]}}, \tag{15}$$

with $\mathcal{I}_k = (0 \ \cdots \ 0 \ \underbrace{I}_{k^{\text{th}} \text{ element}} \ 0 \ \cdots \ 0)$. Then, the update of a constraint turns into

$$\mathbf{x}^{t+1} = \mathbf{x}^t + \lambda |\mathcal{P}_{ji}| \mathbf{M}^{-1} \Omega'_{ji} r'_{ji}, \tag{16}$$

where $|\mathcal{P}_{ji}|$ refers to the number of nodes in $\mathcal{P}_{ji}$. In Eq. (16), we replaced the preconditioning matrix $\mathbf{H}^{-1}$ with its scaled approximation $\mathbf{M}^{-1}$ as described in [41]. This prevents from a computationally expensive matrix inversion.

Let the *level* of a node be the distance in the tree between the node itself and the root. We define the *top node* of a constraint as the node on the path with the smallest level. Our parametrization implies that updating a constraint will never change the configuration of a node with a level smaller than the level of the top node of the constraint.

To summarize, with our approach named TORO [23, 20, 21] (see attachment to D5.1), which is available as open source at [48], we presented a highly efficient solution to the problem of learning maximum likelihood maps for mobile robots. Our technique is based on the graph-formulation of the simultaneous localization and mapping problem and applies a gradient descent based optimization scheme. Our approach extends Olson's algorithm by introducing a tree-based parametrization for the nodes in the graph. This has a significant influence on the convergence speed and execution time of the method. Furthermore, it enables us to correct arbitrary graphs and not only a list of sequential poses. In this way, the complexity of our method depends on the size of the environment and not directly on the length of the input trajectory. This is an important precondition to allow a robot lifelong map learning in its environment. Our method has been implemented and exhaustively tested on simulation experiments as well as on real robot data. We furthermore compared our method to two existing, state-of-the-art solutions which are multi-level relaxation and Olson's algorithm. Our approach converges significantly faster than both approaches and yields accurate maps with low errors.

3.4 Results of Laser-Based Approaches

We ran open source implementations of the approaches described above on the RAWSEEDS datasets obtained at the Bicocca and Bovisa locations.

3.4.1 Relations for RPE

For obtaining close to ground truth information of the outdoor RAWSEEDS datasets, we extracted a set of ground truth relations by manually matching sets of nearby scans. Additionally we integrated relations at locations where recorded ground truth data were available for the indoor datasets. We finally measured the performances of each algorithm on each dataset by using RPE.

Often, a "weighting-factor" is used to combine both error terms into a single number. In this evaluation, however, we provide both terms separately for a better transparency of the results.

3.4.2 Overview RPE results

We processed the benchmark datasets mentioned above using the algorithms described at the beginning of this section. A condensed view of each algorithm's performance is given by the averaged error over all relations. In Table 1 (top) we give an overview on the translational error of the various algorithms, while Table 1 (bottom) shows the rotational error. As expected, it can be seen that the more advanced algorithms (Rao-Blackwellized particle filter and graph mapping) usually outperform scan matching. This is mainly caused by the fact, that scan matching only locally optimizes the result and will introduce topologically errors in the maps, especially when large loops have to be closed. Whenever the RBPF was able to close the loops the results were comparable with the results of GraphSLAM. Running on the challenging RAWSEEDS datasets the RBPF was often not able to close the large loops. Only GraphSLAM was able to generate consistent maps for 10 of 11 datasets.

Since graph mapping uses the TORO graph optimization framework and the evaluation poses are also generated using that framework there is the theoretical chance that graph mapping might achieve an advantage over other approaches by picking the same poses encoded in the relations. Since the relations are generated from the front laser data we decided to evaluate graph mapping based on the rear laser data. Using an independent sensor ensures a fair benchmarking in respect to other approaches.

To visualize the results and to provide more insights about the benchmark solutions, we do not provide the scores only but also plots showing the error of each relation. This enables us to see not only where an algorithm fails, but might also provide insights why it fails. Inspecting those situations in correlation with the map helps to understand the properties of algorithms and gives valuable insights on its capabilities. For the 11 RAWSEEDS datasets, a detailed analysis using these plots is presented in the following sections.

Table 1: Quantitative results of different approaches/datasets (translational error).

Trans. error m	Scan Matching	RBPF (50 part.)	Graph Mapping
Bicocca 2009-02-25a	3.784 ± 5.801	4.400 ± 6.800	0.540 ± 0.618
Max. absolute error of a relation	21.898	21.502	4.214
Bicocca 2009-02-25b	0.963 ± 1.607	1.118 ± 1.978	0.560 ± 0.677
Max. absolute error of a relation	9.163	15.757	6.391
Bicocca 2009-02-26a	1.577 ± 3.406	0.641 ± 0.820	0.601 ± 0.713
Max. absolute error of a relation	21.632	13.692	8.672
Bicocca 2009-02-26b	1.734 ± 3.593	2.531 ± 10.475	0.533 ± 0.659
Max. absolute error of a relation	25.024	67.547	12.517
Bicocca 2009-02-27a	1.856 ± 3.283	0.582 ± 0.757	0.557 ± 0.870
Max. absolute error of a relation	13.698	9.002	12.564
Bovisa 2008-09-01 Static	1.682 ± 5.508	3.591 ± 8.566	0.520 ± 0.723
Max. absolute error of a relation	44.080	33.358	23.788
Bovisa 2008-10-06 Dynamic	1.417 ± 3.141	0.864 ± 1.198	0.723 ± 0.921
Max. absolute error of a relation	23.390	17.779	13.472
Bovisa 2008-10-11a Static	3.667 ± 14.248	2.443 ± 7.670	0.805 ± 0.950
Max. absolute error of a relation	96.529	50.201	22.606
Bovisa 2008-10-04 Static	5.977 ± 22.446	0.802 ± 1.077	0.767 ± 0.954
Max. absolute error of a relation	137.027	23.674	24.763
Bovisa 2008-10-07 Dynamic	1.240 ± 4.645	1.703 ± 6.046	1.611 ± 6.416
Max. absolute error of a relation	41.471	$52,694$	59.916
Bovisa 2008-10-11b Static	1.995 ± 8.998	2.148 ± 9.182	2.332 ± 10.012
Max. absolute error of a relation	87.441	93.938	101.162

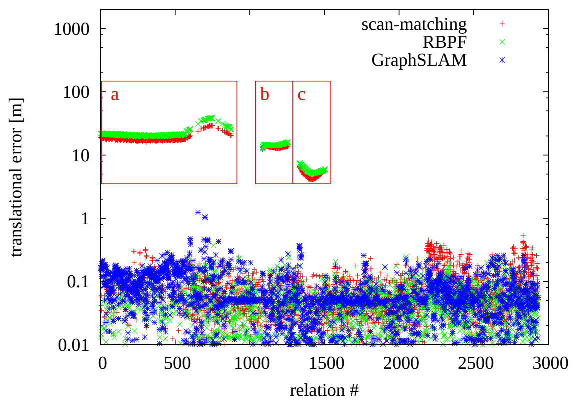
Table 2: Quantitative results of different approaches/datasets (rotational error).

Rot. error *degree*	Scan Matching	RBPF (50 part.)	Graph Mapping
Bicocca 2009-02-25a	4.359 ± 6.800	3.538 ± 3.997	0.540 ± 0.618
Max. absolute error of a relation	21.502	70.825	4.214
Bicocca 2009-02-25b	1.118 ± 1.978	2.408 ± 2.366	0.560 ± 0.677
Max. absolute error of a relation	15.757	37.060	6.391
Bicocca 2009-02-26a	0.641 ± 0.820	2.822 ± 3.234	0.601 ± 0.713
Max. absolute error of a relation	13.692	131.907	8.672
Bicocca 2009-02-26b	3.170 ± 8.776	1.776 ± 8.108	0.586 ± 0.704
Max. absolute error of a relation	49.497	67.547	8.726
Bicocca 2009-02-27a	1.959 ± 5.031	0.473 ± 0.635	0.495 ± 0.758
Max. absolute error of a relation	$21,646$	9.318	16.342
Bovisa 2008-09-01 Static	14.116 ± 38.512	3.591 ± 8.556	0.466 ± 0.605
Max. absolute error of a relation	166.501	9.318	6.258
Bovisa 2008-10-06 Dynamic	13.358 ± 40.764	0.0864 ± 1.198	0.297 ± 0.427
Max. absolute error of a relation	179.965	17.779	10.777
Bovisa 2008-10-11a Static	4.736 ± 19.928	2.443 ± 7.670	0.194 ± 0.409
Max. absolute error of a relation	124.149	50.201	15.685
Bovisa 2008-10-04 Static	11.295 ± 36.546	0.802 ± 1.077	0.171 ± 0.267
Max. absolute error of a relation	179.750	23.674	7.250
Bovisa 2008-10-07 Dynamic	1.771 ± 9.159	1.703 ± 6.046	0.174 ± 0.291
Max. absolute error of a relation	101.242	$52,694$	11.913
Bovisa 2008-10-11b Static	3.620 ± 18.200	2.148 ± 9.182	2.647 ± 15.991
Max. absolute error of a relation	173.051	93.938	108.738

[1] scan matching has been applied as a preprocessing step.

3.4.3 Results for Bicocca 2009-02-25a

In the Bicocca datasets the robot traverses mainly long corridors and hallways as it can be found in many education or office buildings. This allows for multiple nested loops along the robots trajectory. We extracted close to 3000 nearby evaluation positions of the robot. Figure 2 shows the error distributions for the given relation set. Regions in the map with high inconsistencies correspond to relations having a high error. We labeled the error peaks in Figure 2 and the corresponding areas in the resulting maps in the following Figures 3,4,5 to illustrate the association between the maps and the error plots.



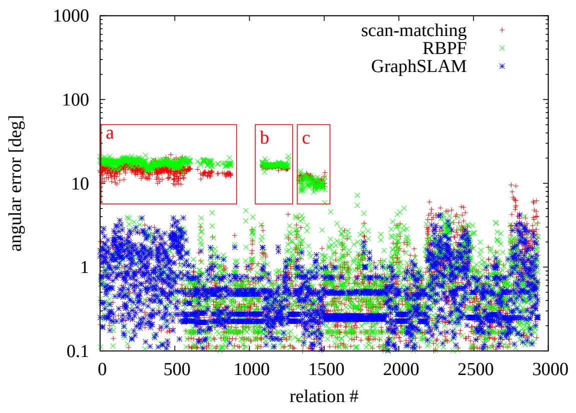


Figure 2: Evaluation of the Bicocca-2009-02-25a dataset using the RPE metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees. The red marked areas a,b and c show the error peaks for the loop closure relations.

Figure 3 shows the resulting map for the scan matching algorithm with colored relations. Relations for translational errors below 20 cm are colored green and relations for errors equal or above 20 cm are colored red. Most of the incremental relations that follow the path of the robot have an error below 20 cm. In contrast most of the loop closure relations report an error over 20 cm. This highlights the aspect that scan matching optimizes only the local consistence of a map and has no mechanism to consider loop closures in the trajectory of the robot. In this case scan matching leads to an inconsistent map. The error plots in Figure 2 show high translational and rotational errors for the relations in the three red marked loop closure areas a,b and c.

The results for RBPF with the Bicocca 2009-02-25a dataset are illustrated in Figure 4. Most of the errors of the RBPF are reported by the loop closure relations. Considering the quantity of errors the map is far from being globally consistent. This can also be observed by the peaks in the error plots for a,b and c. The particle filter might have missed the loop closures because no particle survived that could close the loop and generate a consistent map. This is a general drawback of particle filters for loop closures after long trajectories. The dataset is quite hard for a particle filter approach, since the loops are closed at a very late part of the trajectory.

The resulting map for GraphSlam shown in Figure 5 looks globally consistent. All three loop closures are detected and no major global inconsistency is visible. Also the dense relations from

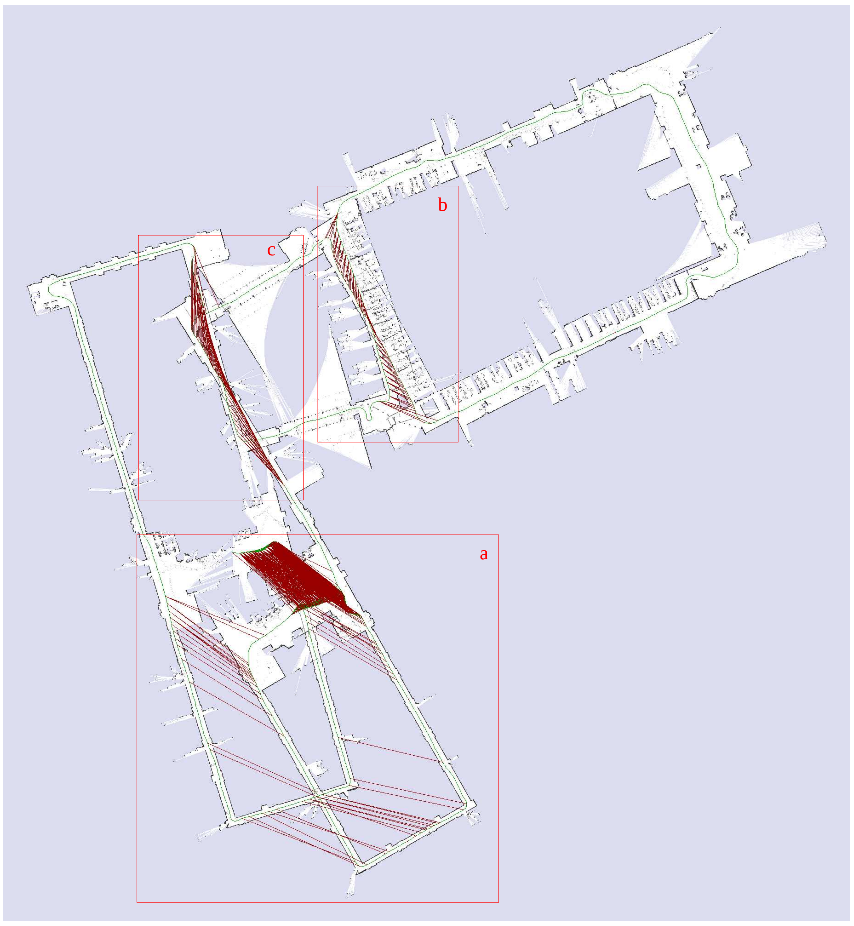


Figure 3: Result of the scan matching algorithm for the Bicocca 2009-02-25a dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green. The red rectangles a,b and c mark the map parts with high errors in the loop closure relations. The corresponding error regions are also marked in the error plot in Figure 2.

the externally recorded ground truth area show only a few errors over 20 cm. The GraphSLAM map is sufficient for navigation tasks.

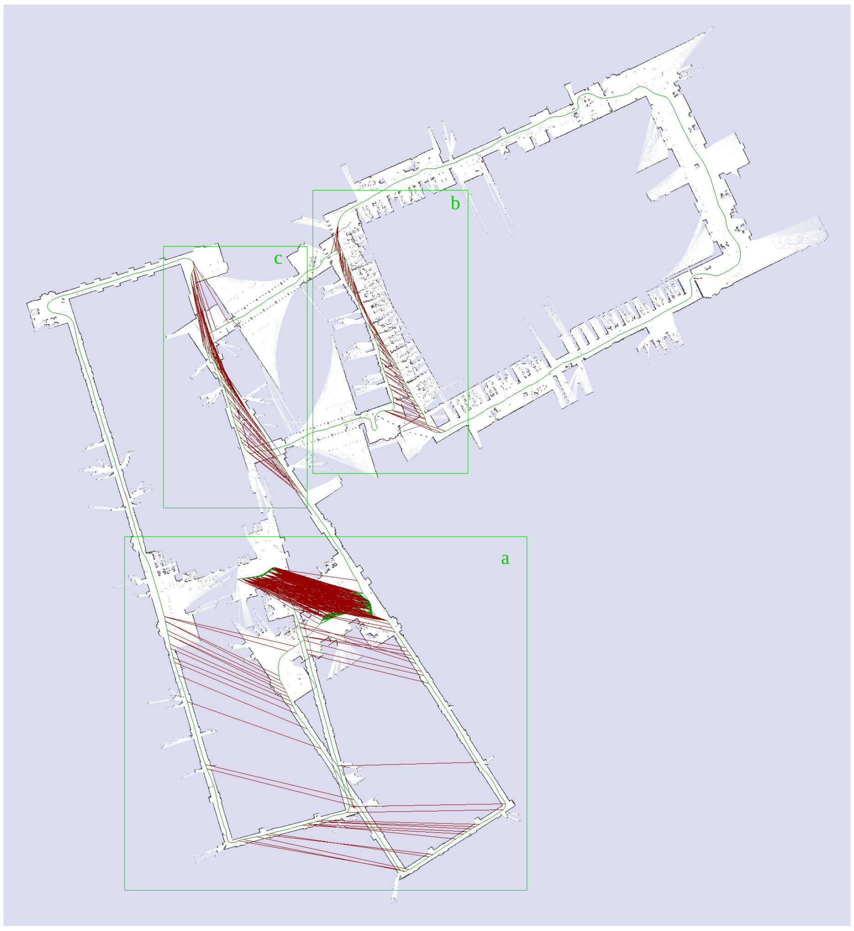


Figure 4: Result of RBPF for the Bicocca 2009-02-25a dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green. The green boxes a,b and c mark the map parts with high errors in the loop closure relations. The corresponding error regions are also marked in the error plot in Figure 2.

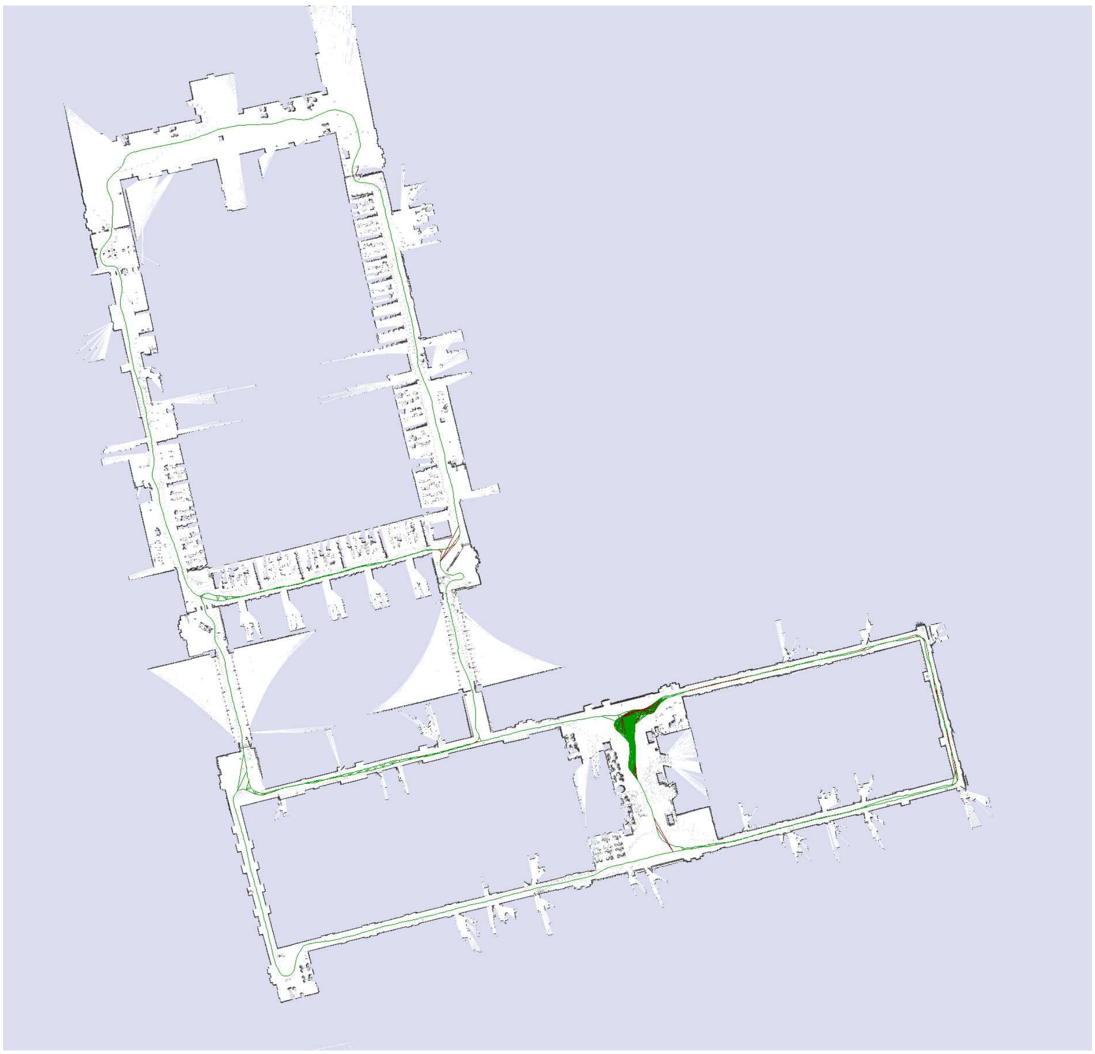


Figure 5: Result of GraphSLAM for the Bicocca 2009-02-25a dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green.

3.4.4 Results for Bicocca 2009-02-25b

The relations collection for the Bicocca 2009-02-25b consists of more than 9000 relations. Figure 6 shows the error distributions for the relations collection.

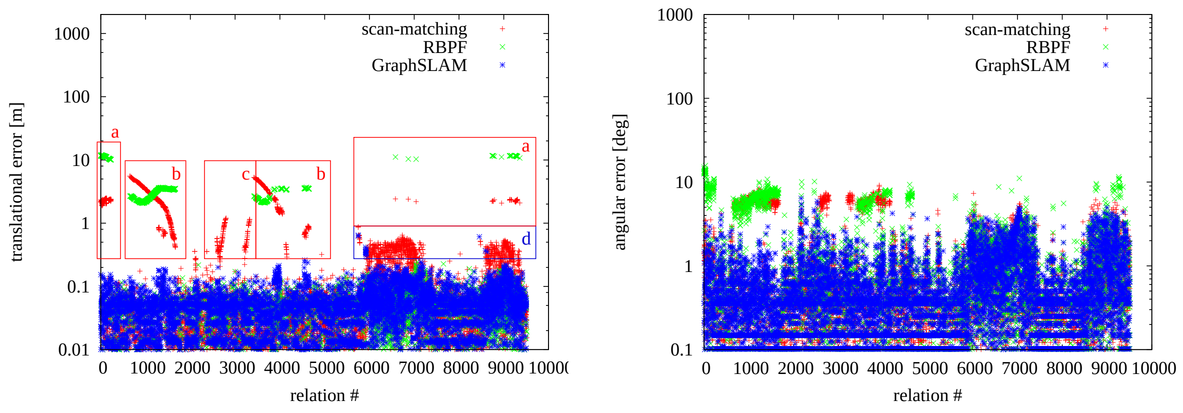


Figure 6: Evaluation of the Bicocca-2009-02-25b dataset using the defined metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees. The corresponding loop closure areas are marked with red boxes.

Figure 7 shows the resulting map for the scan matching algorithm. The small alignment errors of the scan matching procedure are accumulated over the complete trajectory and result in global inconsistencies, as already shown in the first dataset. Therefore the loop closure areas a,b and c are inconsistent. The corresponding error peaks are shown in Figure 6.

The result for the RBPF is shown in Figure 8. The small loop closure in this dataset is successfully detected as illustrated in the green marked area c in Figure 8. In consequence the error plots in Figure 6 show no large errors for RBPF in area c. Again the particle filter misses the loop closures for the large scale loops in areas a and b.

The GraphSLAM result is shown in Figure 9. The resulting map looks topological correct and should be sufficient for navigation tasks. The blue marked area d has higher errors (red lines) what indicates some inconsistency and can be observed in the map as double walls.

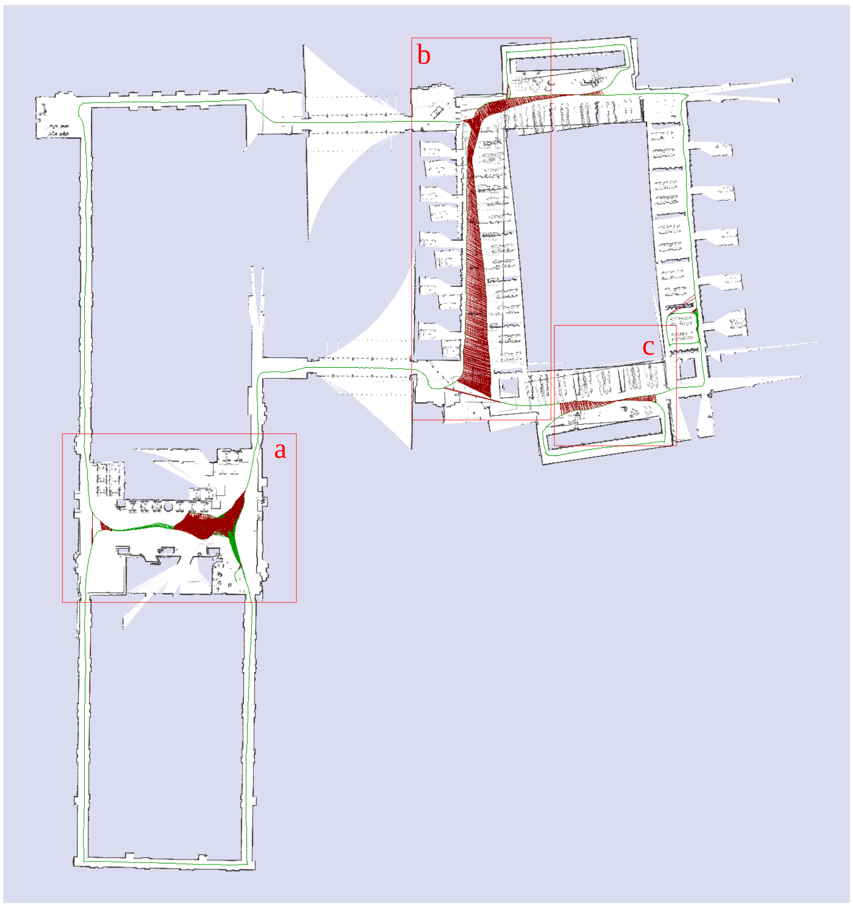


Figure 7: Result of the scan matching algorithm for the Bicocca 2009-02-25b dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green. The three loop closure areas a,b and c are inconsistent.

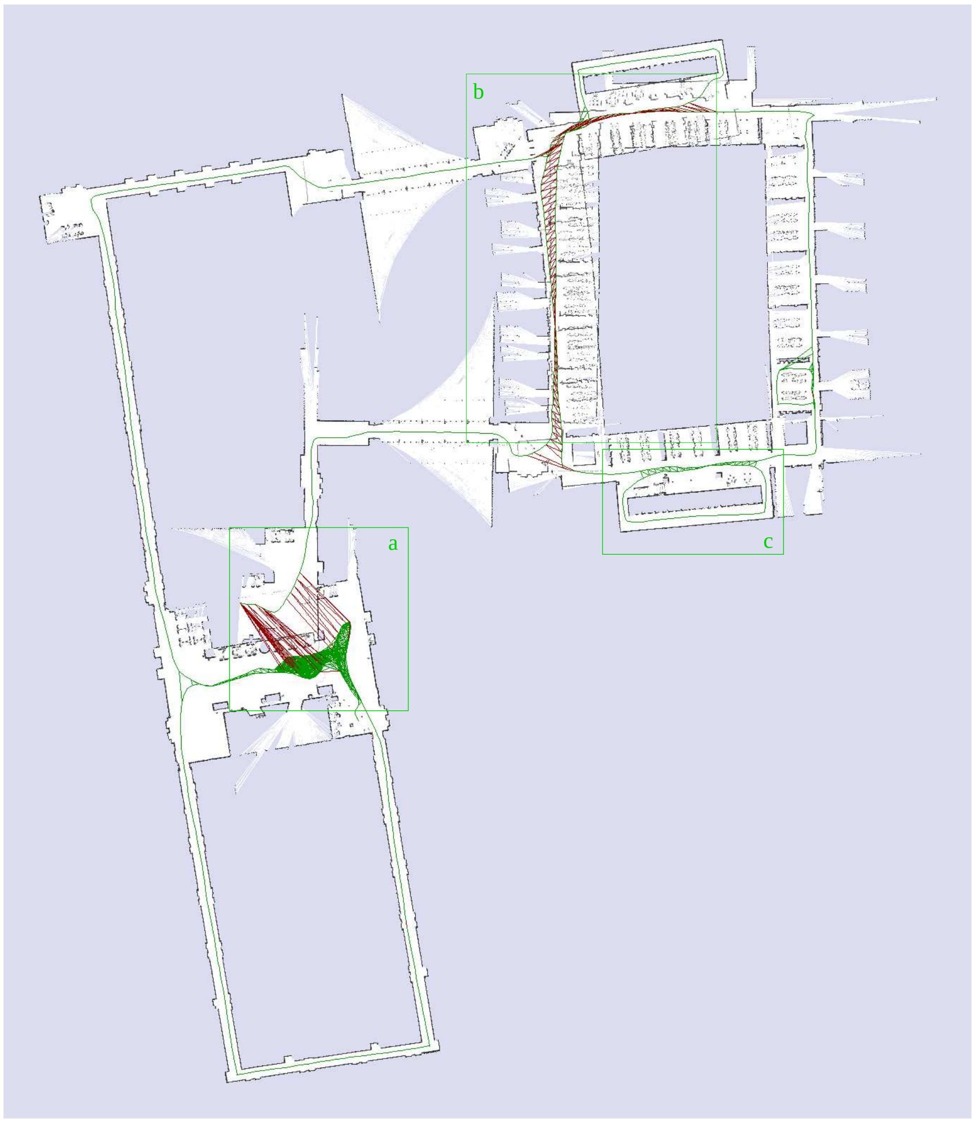


Figure 8: Result of RBPF for the Bicocca 2009-02-25b dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green. The large loops in the areas a and b are not closed and the map is inconsistent in this areas. The small loop in area c is successfully closed.

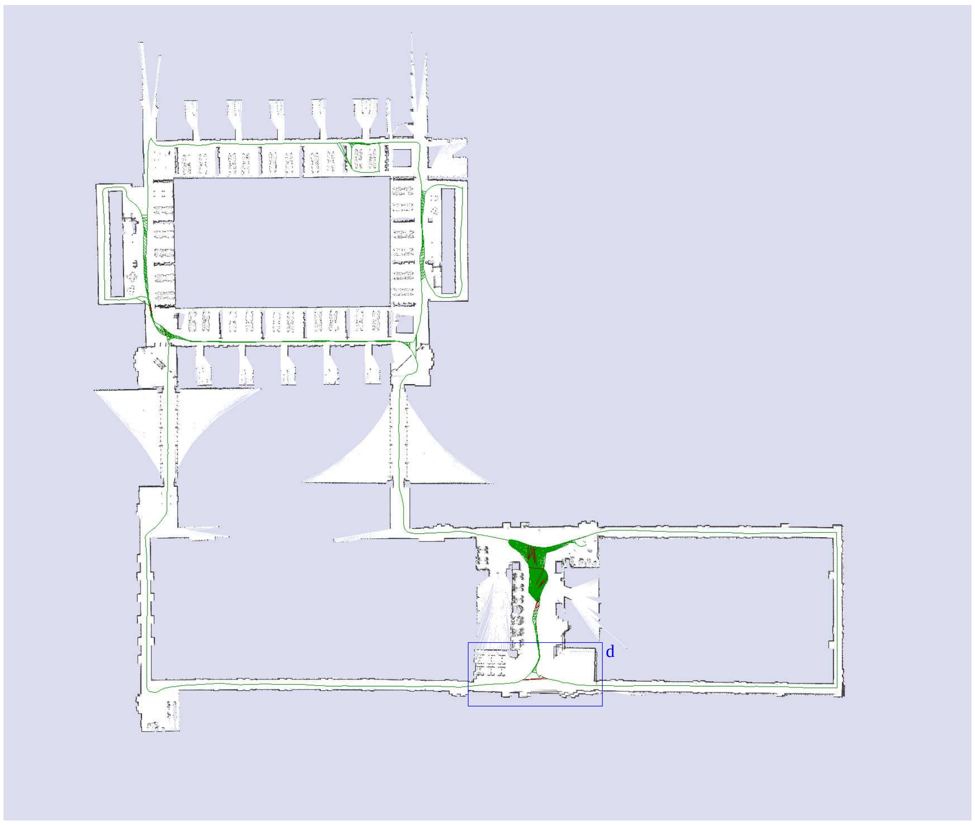
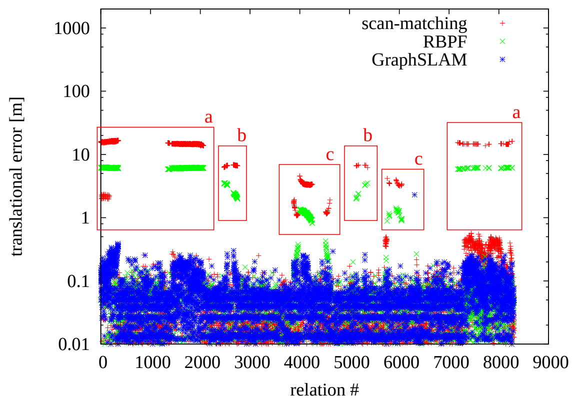


Figure 9: Result of Graph SLAM for the Bicocca 2009-02-25b dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green.

3.4.5 Results for Bicocca 2009-02-26a

We extracted close to 8000 relations for the Bicocca 2009-02-26a dataset to evaluate it with the RPE metric. The corresponding error distributions are shown in Figure 10. The three loop closure areas a,b and c are marked with red rectangles.



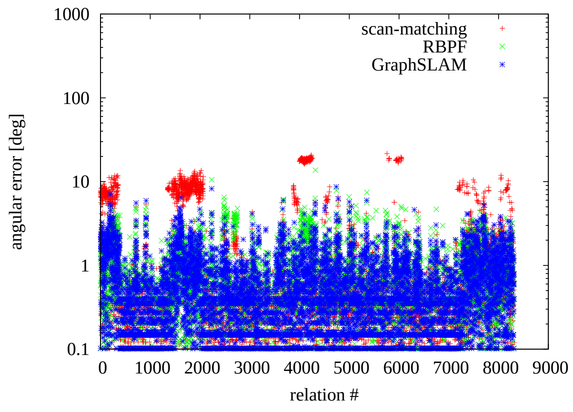


Figure 10: Evaluation of the Bicocca-2009-02-26a dataset using the defined metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees. The three loop closure areas a,b and c are marked with red boxes.

Figure 11 shows the resulting map for the scan matching algorithm. The three loop closure areas have also been highlighted with red rectangles. The small alignment errors of the scan matching procedure are accumulated over the complete trajectory and results in an inconsistend map.

The result of the RBPF is shown in Figure 12 and the three loop closure areas are marked with green rectangles. The loop closures a,b and c are not detected by the algorithm and therefore the map is inconsistent. The error peaks in Figure 10 are lower than the corresponding errors for scan matching. The resulting map looks subjectively more consistent than the scan-matching result in terms of topology.

The GraphSLAM result in Figure 13 looks topological correct and has only minor inconsistencies. As reminder the results of GraphSLAM are produced using the rear laser.

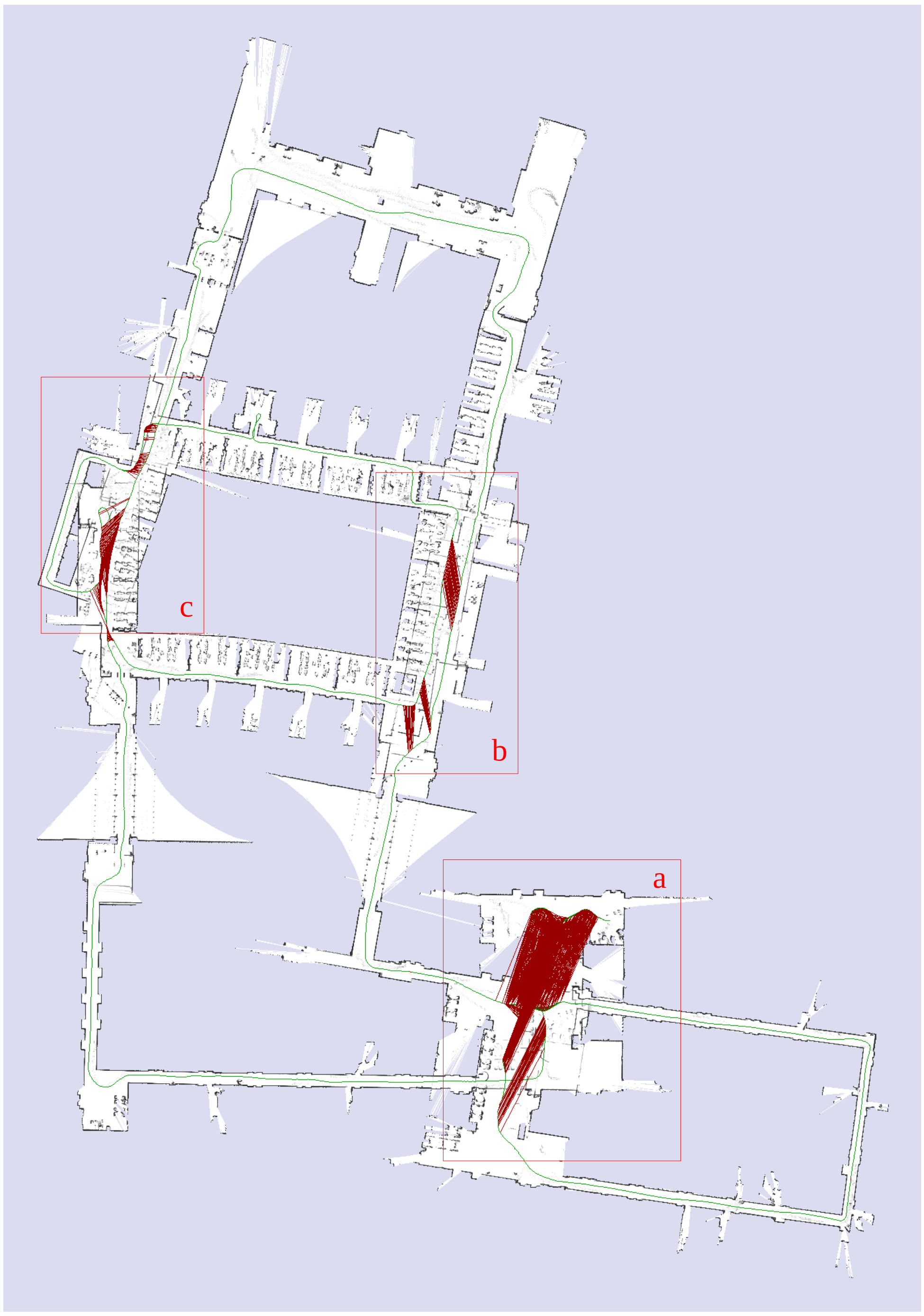


Figure 11: Result of the scan matching algorithm for the Bicocca 2009-02-26a dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green.

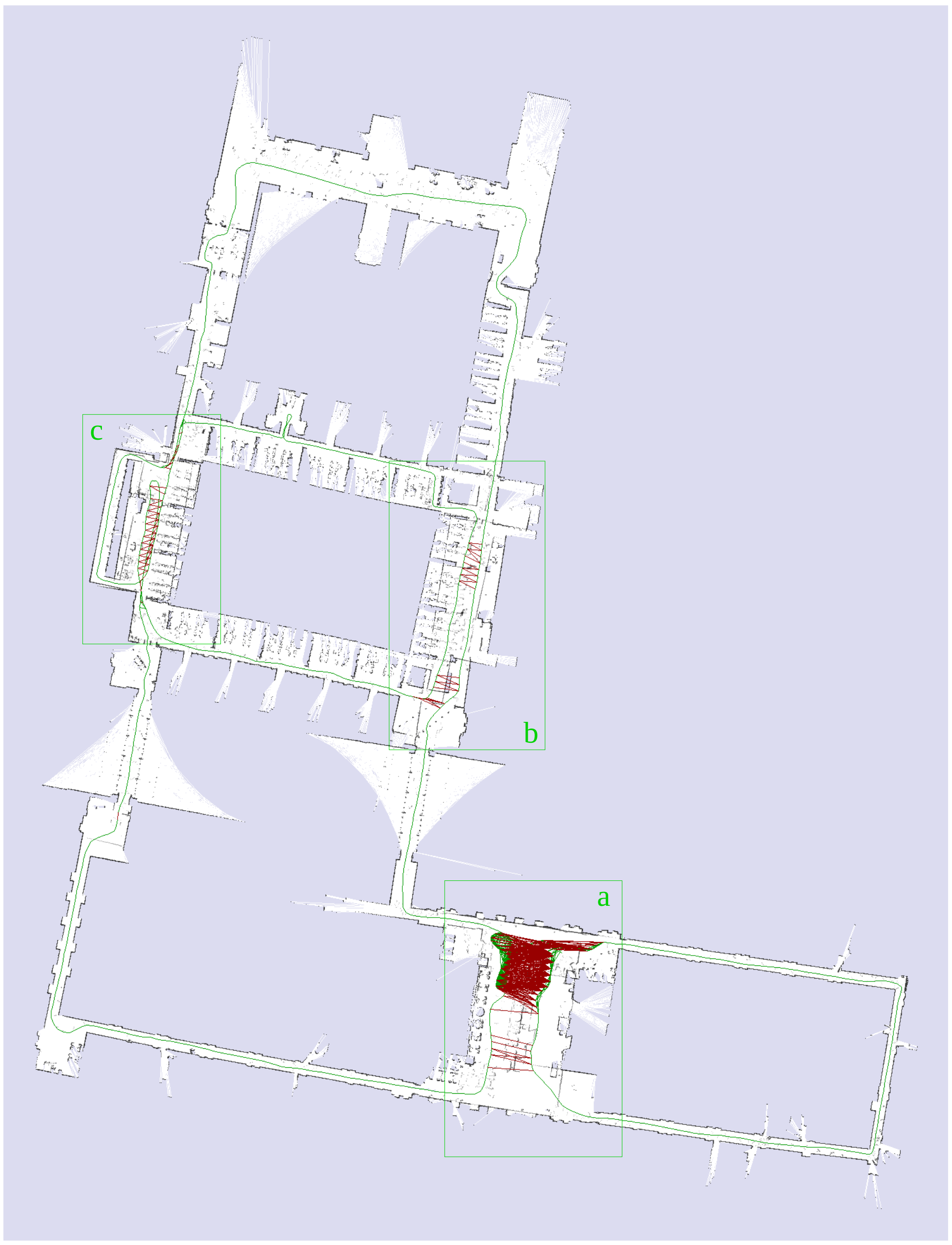


Figure 12: Result of RBPF for the Bicocca 2009-02-26a dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green.

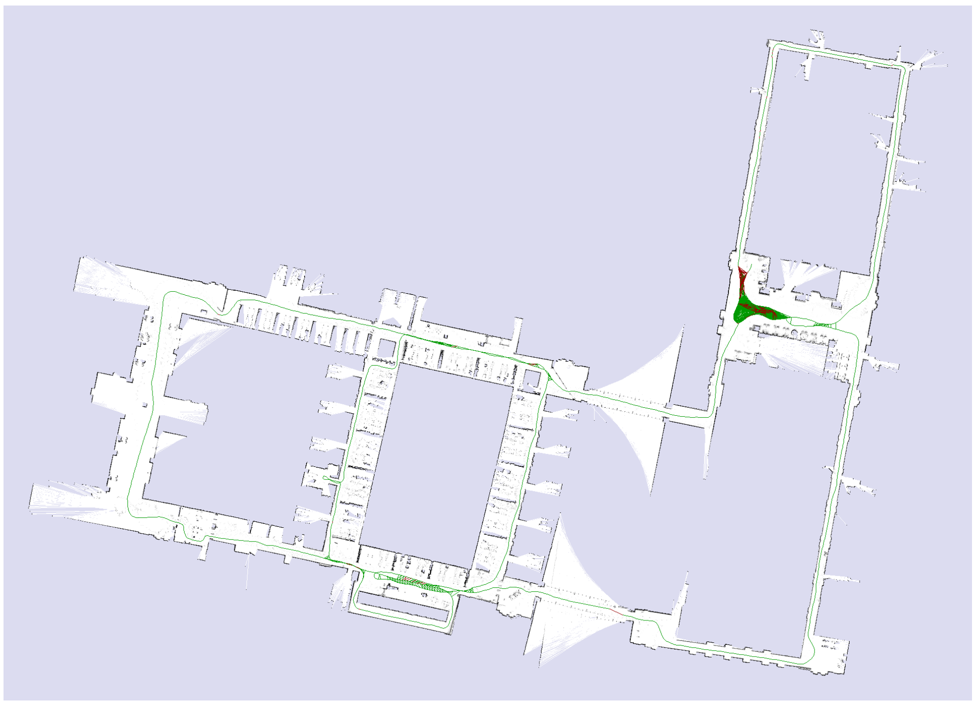
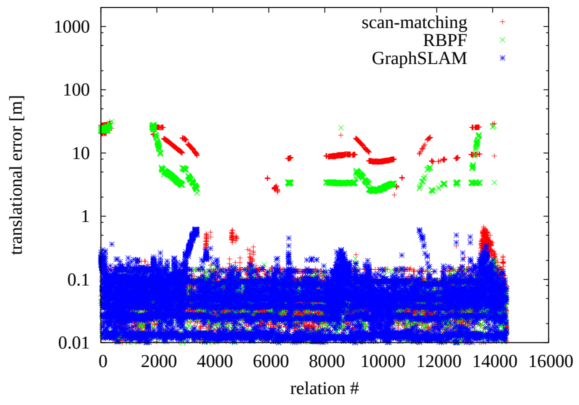


Figure 13: Result of GraphSLAM for the Bicocca 2009-02-26a dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green.

3.4.6 Results for Bicocca 2009-02-26b

The relations dataset for Bicocca 2009-02-26b consists of 14000 positions. The error distributions for the three evaluated SLAM approaches are shown in Figure 14. Since there is only one large loop closure area we did not mark the corresponding areas and errors. Figure 15 shows the scan matching result. The resulting map has major inconsistencies and is a good example for the significance of the accumulated error over long trajectories. The result of the RBPF in Figure 16 looks better than the scan matching result, but is still not good enough for navigation tasks of autonomous robots, since the topology has major errors. The GraphSLAM result in Figure 17 looks topological correct and has only some minor errors in the map.



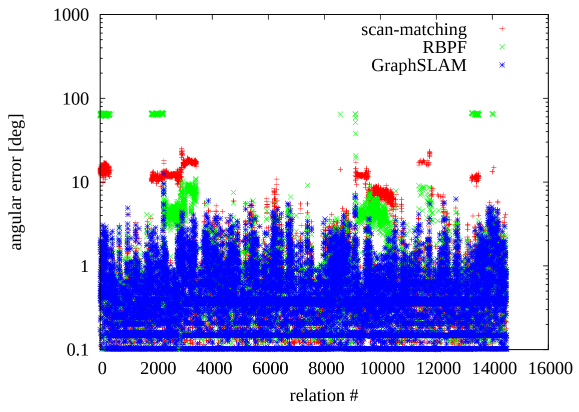


Figure 14: Evaluation of the Bicocca-2009-02-26b dataset using the defined metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees.

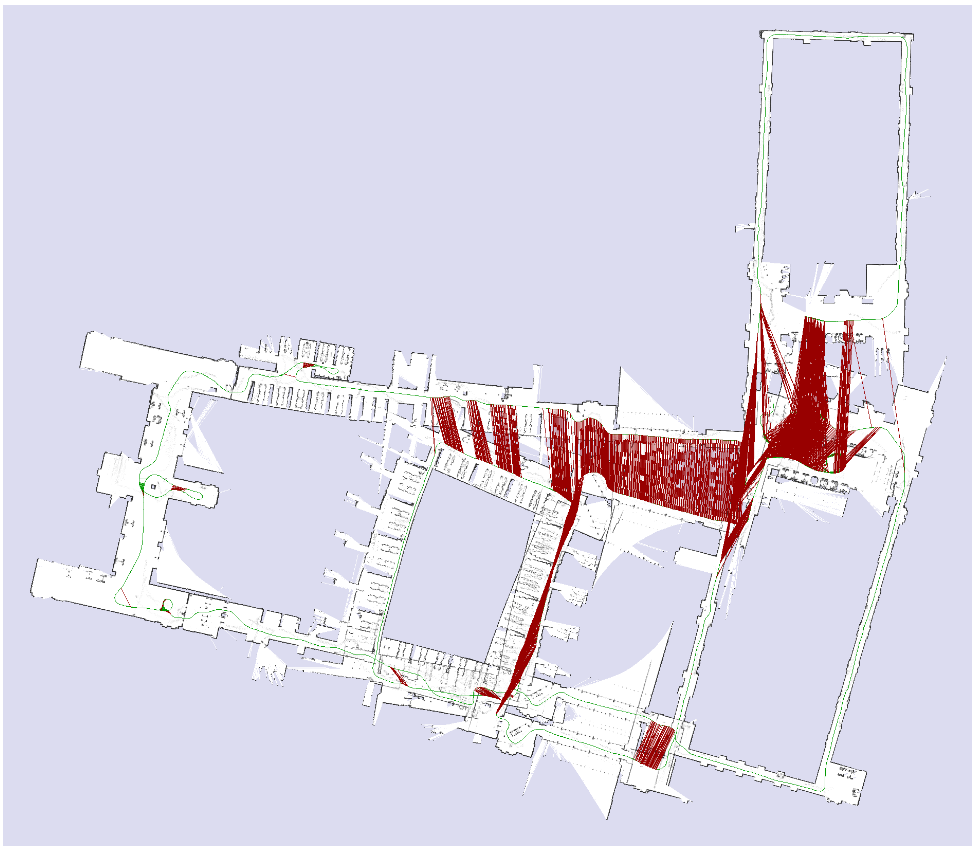


Figure 15: Result of the scan matching algorithm for the Bicocca 2009-02-26b dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green.

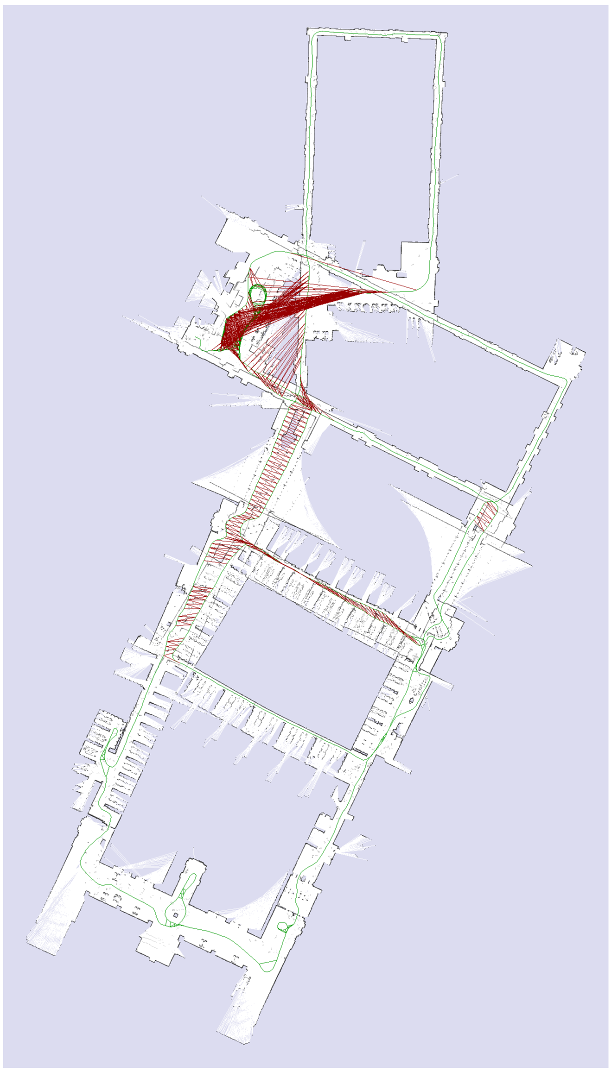


Figure 16: Result of RBPF for the Bicocca 2009-02-26b dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green.

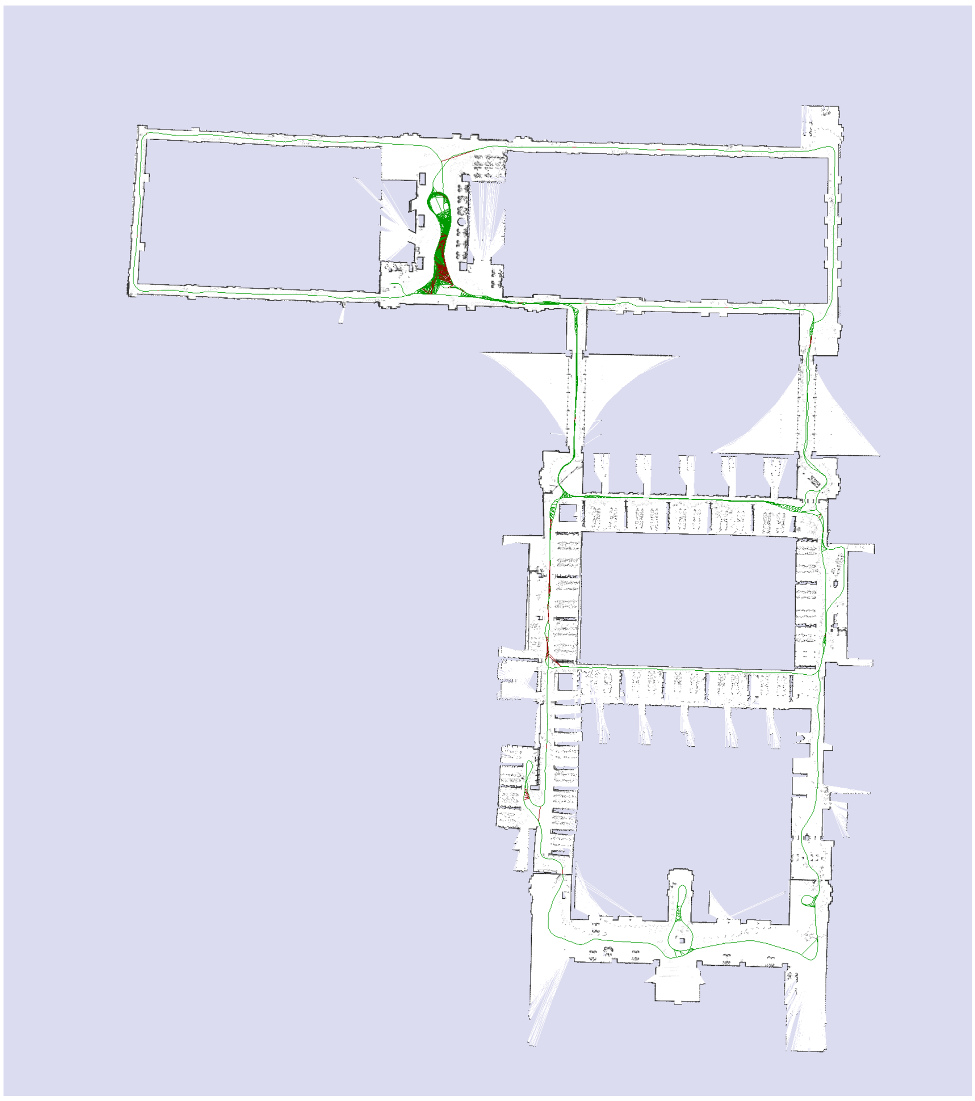
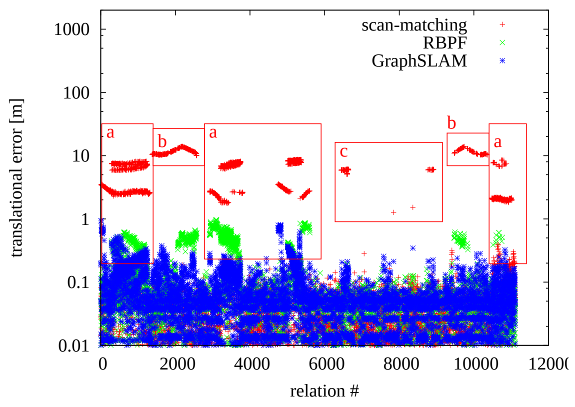


Figure 17: Result of GraphSLAM for the Bicocca 2009-02-26b dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green.

3.4.7 Results for Bicocca 2009-02-27a

We extracted close to 11000 evaluation positions for the Bicocca 2009-02-27a dataset. The error distributions for the three SLAM approaches are illustrated in Figure 18. The scan matching result is shown in Figure 19. Again the scan matching algorithm is not able to generate a consistent map since it can not close the loops. The resulting map of the RBPF is shown in Figure 20 and the loop closure areas are marked with green rectangles. The map is topological correct and has only some minor inconsistencies. Those minor inconsistencies are in the regions of a and b and can be observed by the red relations and the double walls occurring in that areas. The error plots in Figure 18 show only errors below one meter for a and b and no error peak in area c. The GraphSLAM result for the dataset is illustrated in Figure 21. The three loop closing areas are marked by blue rectangles. The map looks consistent in the three areas and no double walls are visible. The relations acquired from the ground truth recording system in area a report errors over 20 cm. Since the map looks consistent in that area it is possible that those errors are caused by a small error in the angle reported of the ground truth system.



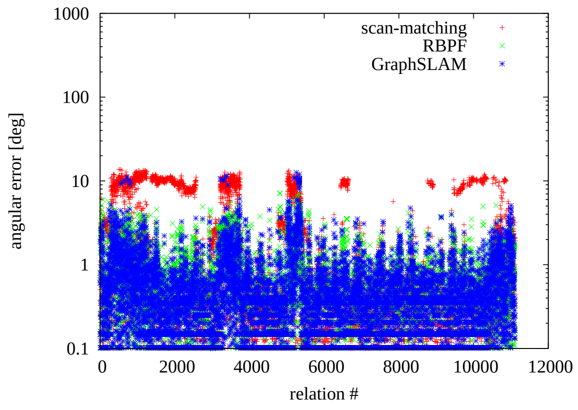


Figure 18: Evaluation of the Bicocca-2009-02-27a dataset using the defined metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees. The three areas with major errors a,b and c are marked with red rectangles.

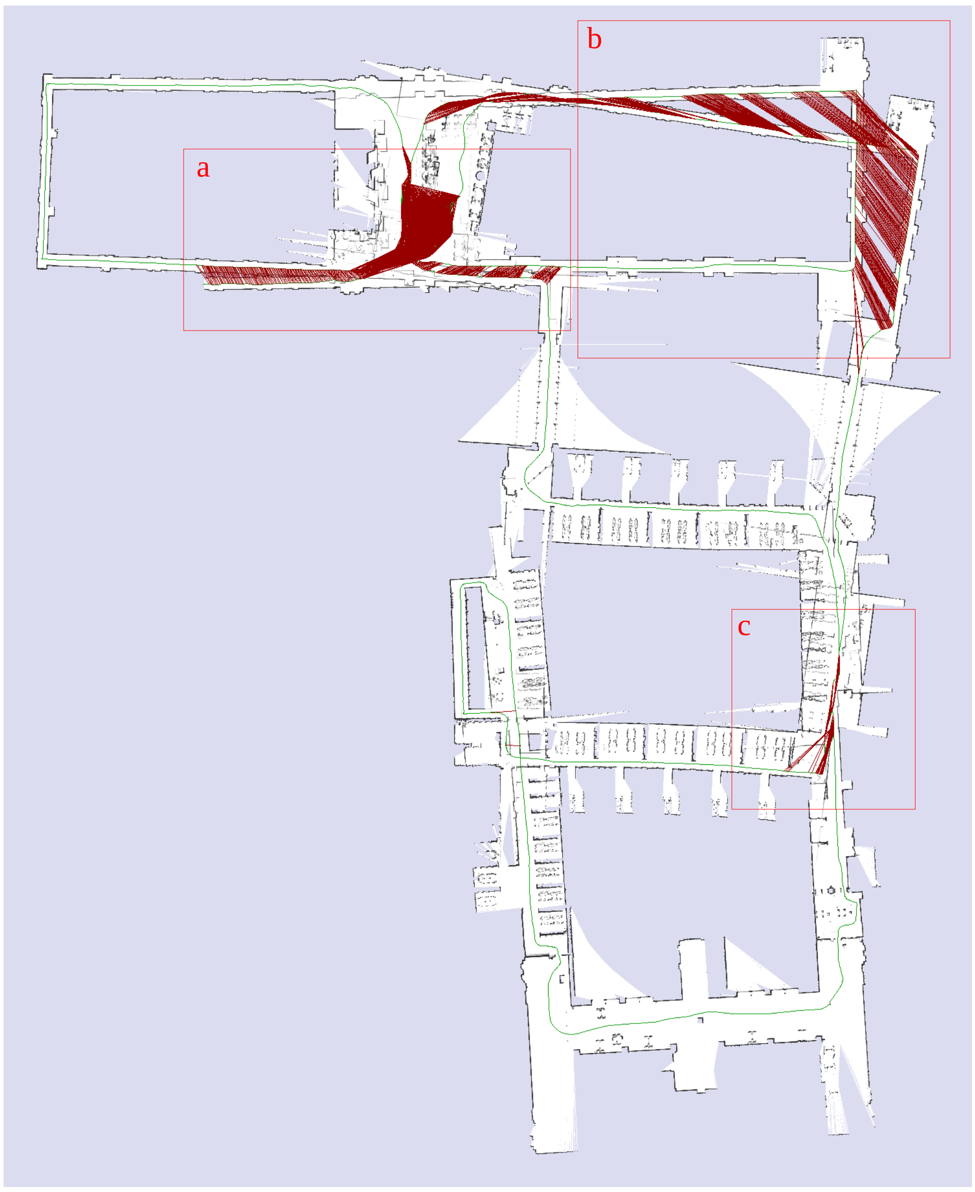


Figure 19: Result of the scan matching algorithm for the Bicocca 2009-02-27a dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green. The three loop closure areas are marked with red rectangles. The major errors are reported by the loop closing relations in the areas a,b and c.

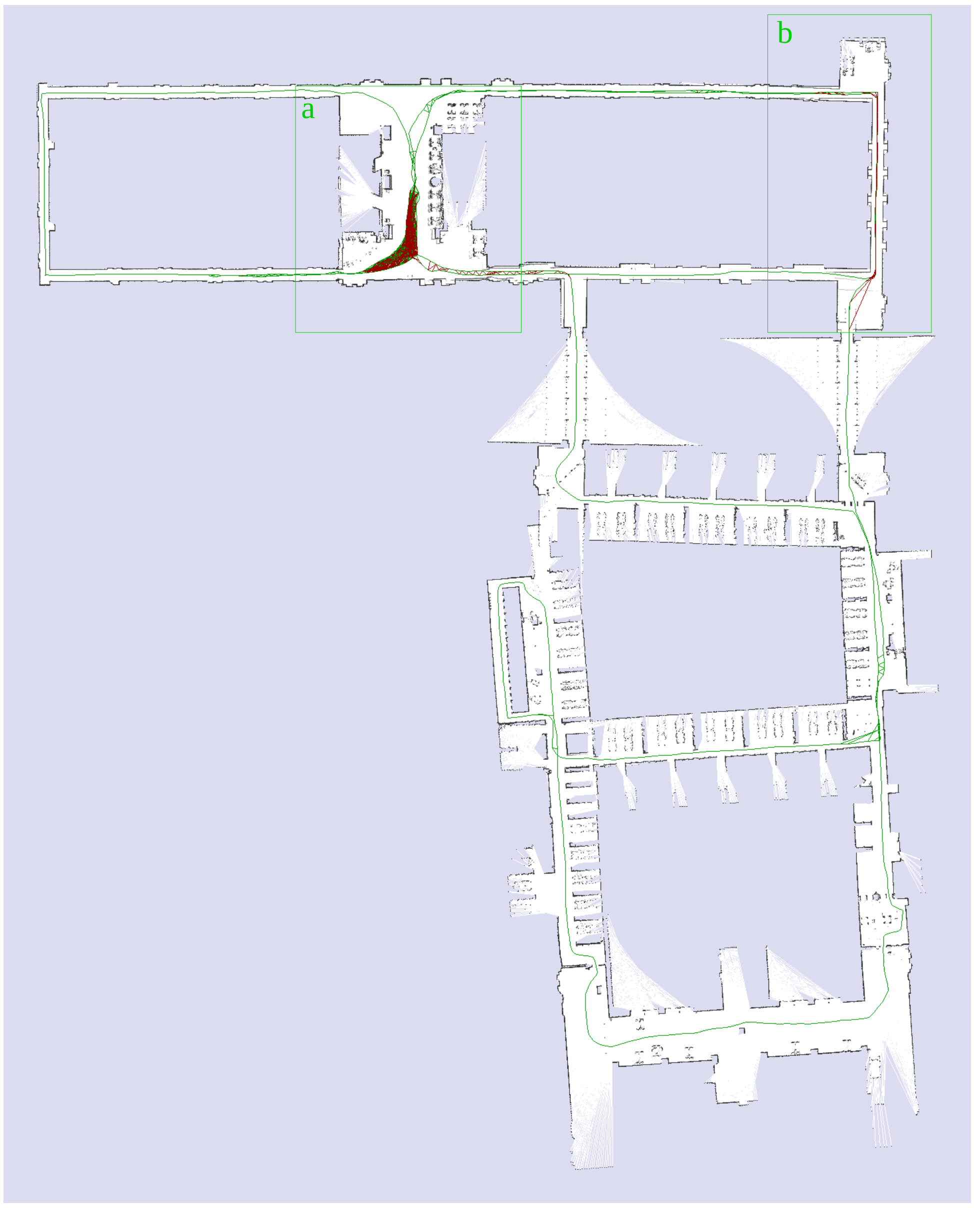


Figure 20: Result of RBPF for the Bicocca 2009-02-27a dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green. Most of the errors are in the areas a and b that comes from the loop closing relations.

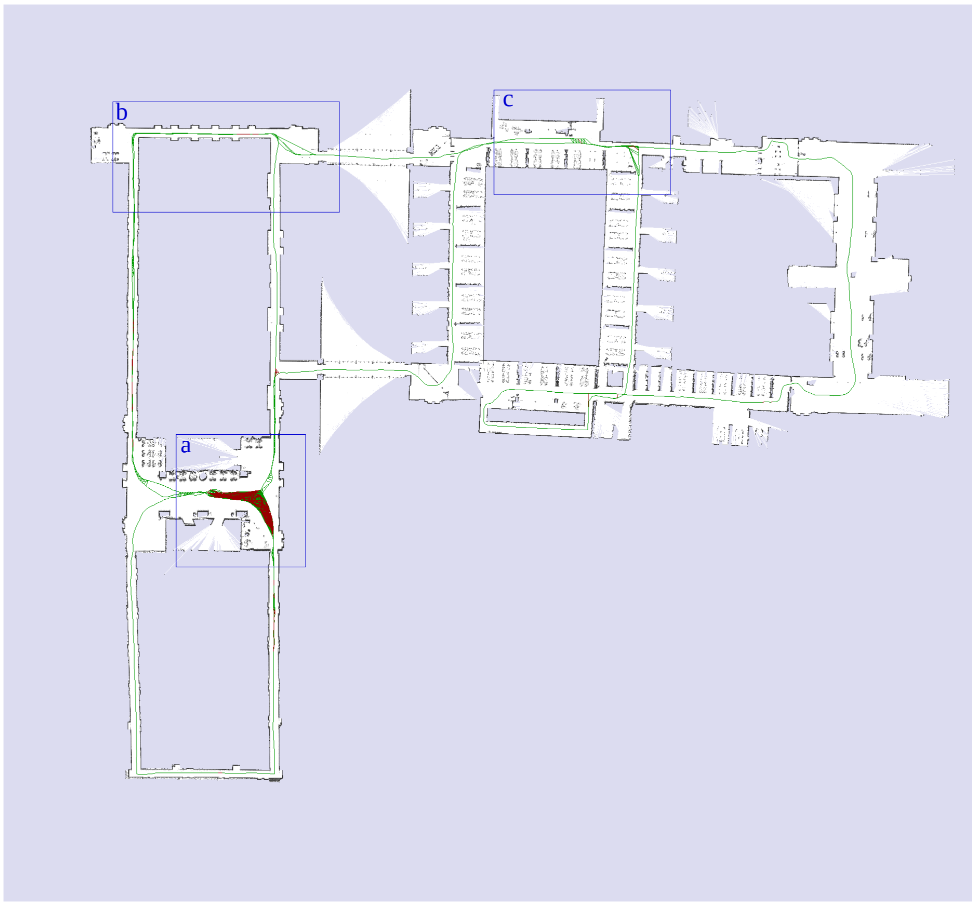


Figure 21: Result of GraphSLAM for the Bicocca 2009-02-27a dataset. Relations with an error over 0.2 m are colored red and below 0.2 m are colored green. The map looks consistent in all three loop closing relation areas. Those areas a,b and c are marked with blue rectangles.

3.4.8 Results for Bovisa 2008-09-01

The Bovisa 2008-09-01 dataset was recorded at the Bovisa location in Milan. That location allows for outdoor and indoor data recording and therefore it is possible to generate challenging mixed indoor/outdoor datasets. The Bovisa 2008-09-01 is one of those mixed indoor/outdoor datasets. The relations collection of the Bovisa 2009-09-01 dataset consists of more than 17000 nearby evaluation positions.

The result of the scan matching approach is illustrated in Figure 23. The accumulated error grows over the trajectory and no loop closures are considered what leads to an globally inconsistent map. Figure 24 shows the resulting map for the RBPF which is also globally inconsistent. This inconsistency is introduced by an alignment error in the blue marked area in Figure 24. The RBPF was not able to correct that error at the next loop closure. This might be caused by the fact that no good hypotheses survived that would lead to a consistent map. Figure 25 shows the resulting map for the GraphSLAM approach, which looks consistent and shows only few minor errors. The error distributions are shown in Figure 22.

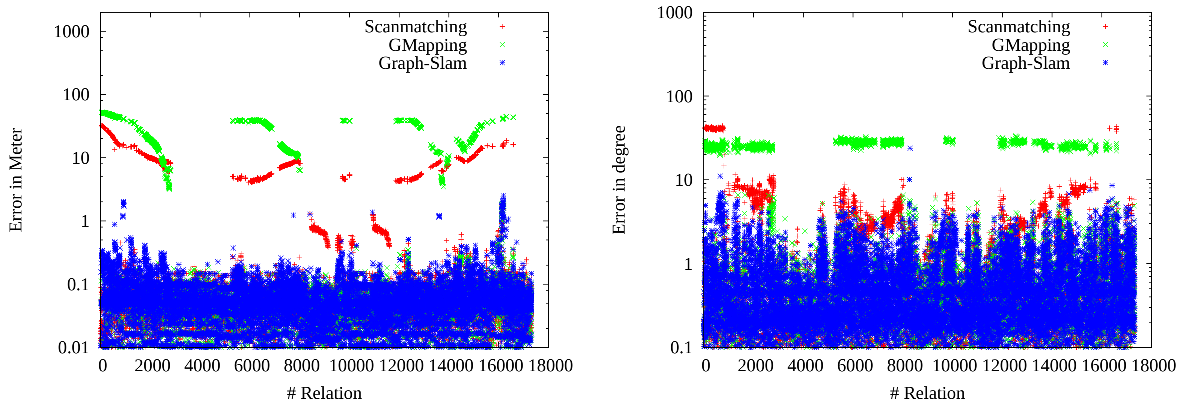


Figure 22: Evaluation of the Bovisa-2008-09-01 dataset using the defined metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees.

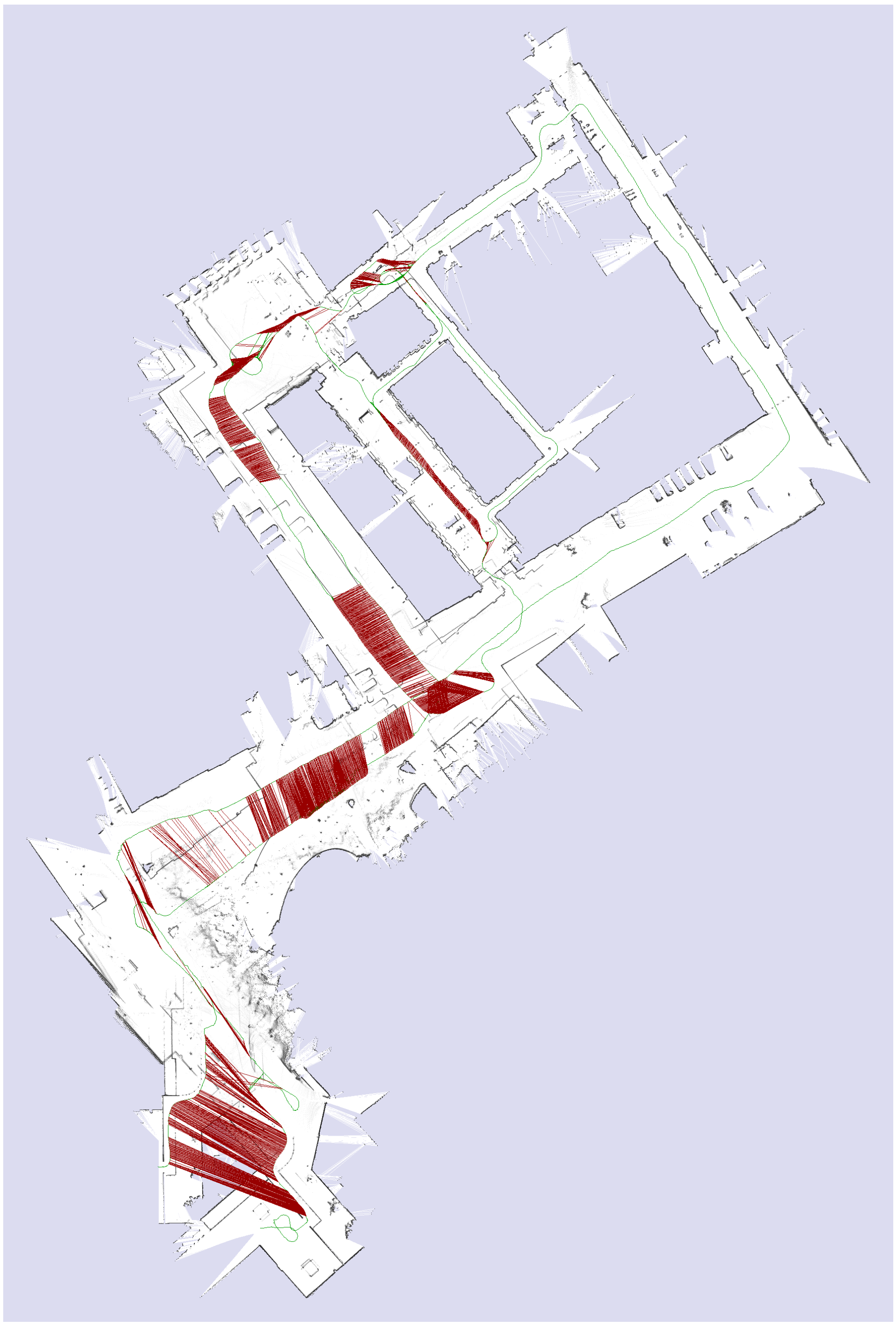


Figure 23: Result of the scan matching algorithm for the Bovisa 2008-09-01 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green. The shown map has major inconsistencies and has also topological errors.

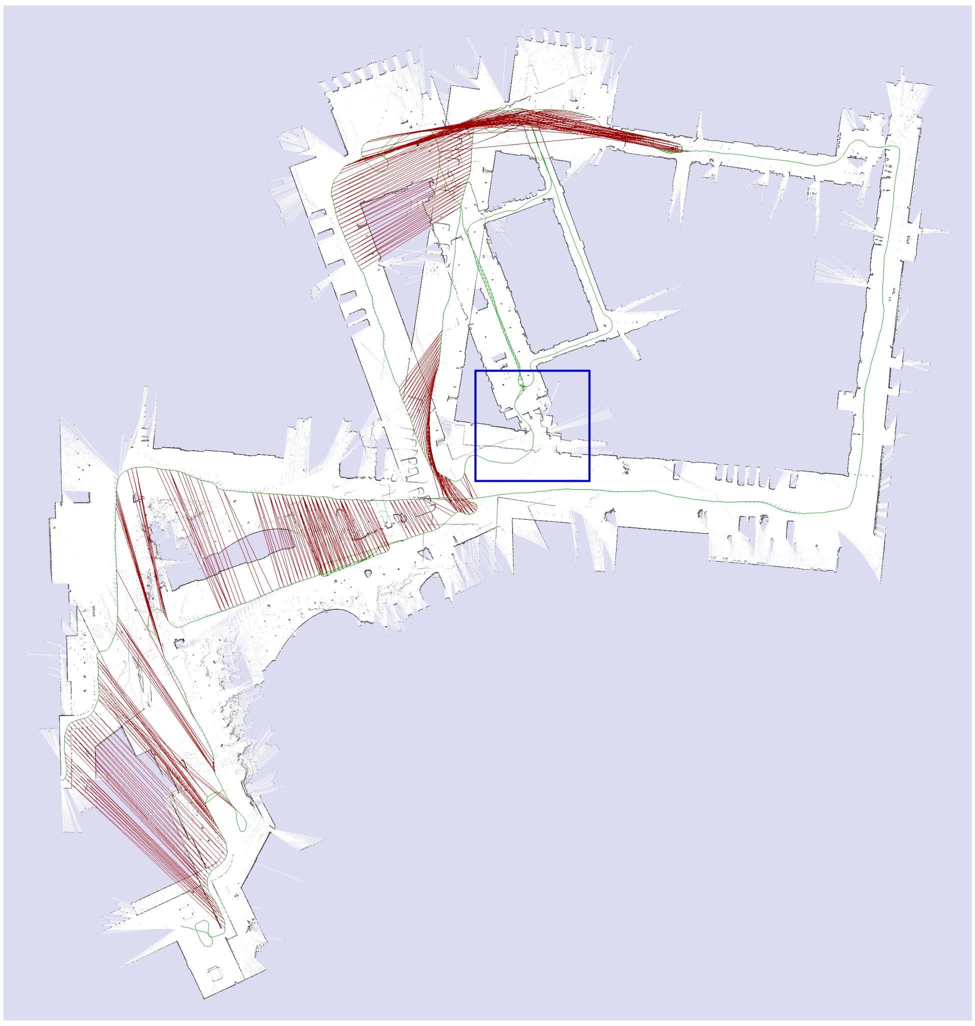


Figure 24: Result of RBPF for the Bovisa 2008-09-01 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green. A drift in the blue marked region causes the inconsistent map. The RBPF is not able to correct that error at the next loop closure.

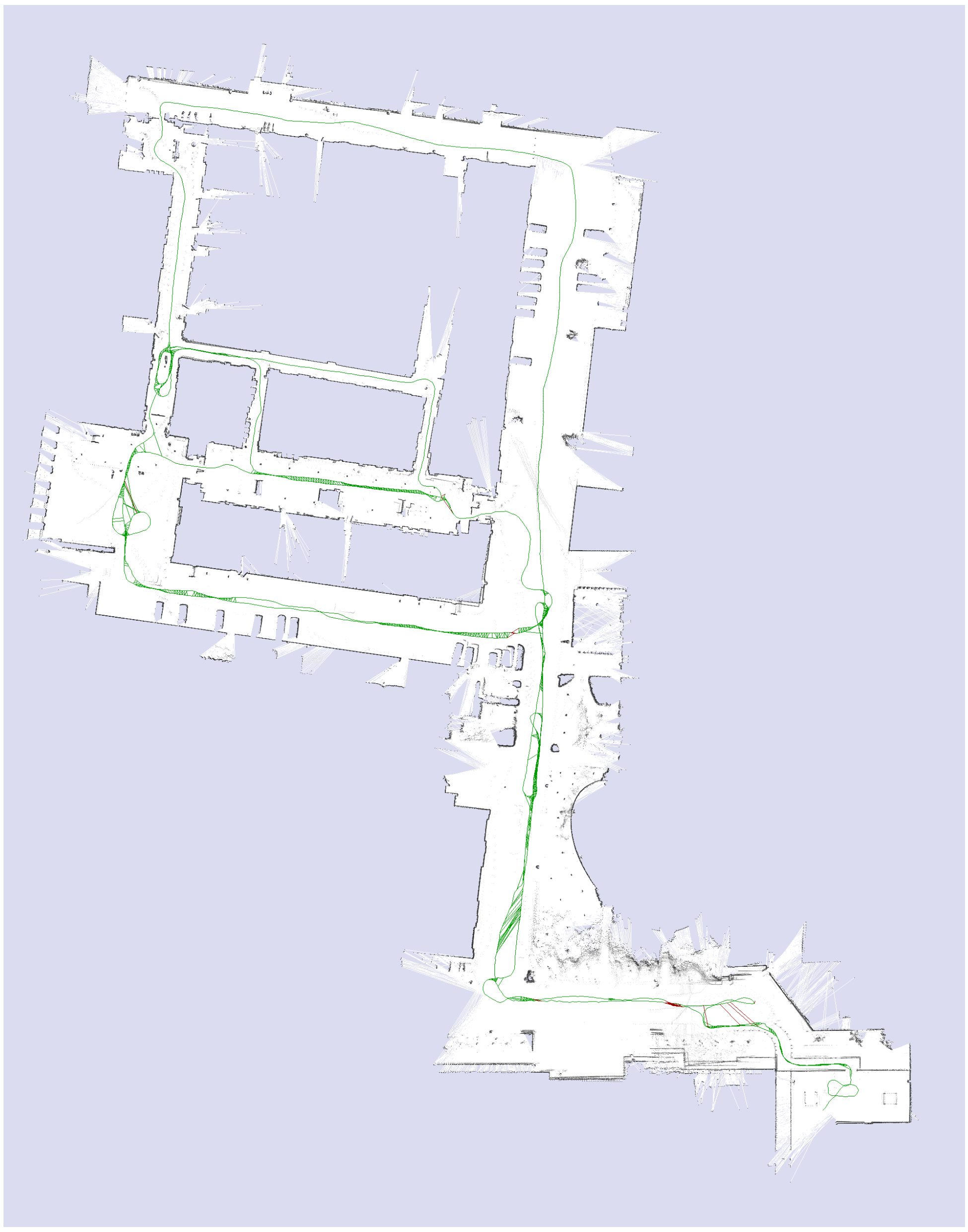
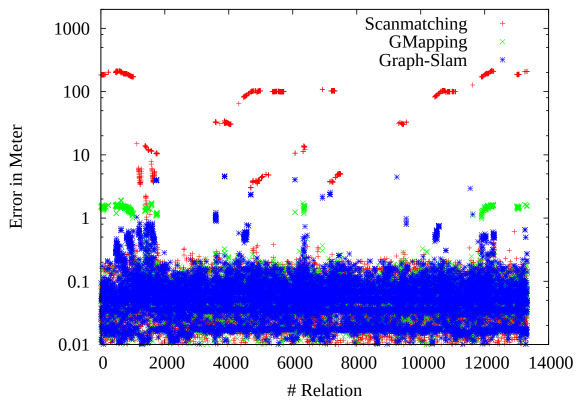


Figure 25: Result of GraphSLAM for the Bovisa 2008-09-01 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green. The shown map has only minor errors and is topological correct.

3.4.9 Results for Bovisa 2008-10-04

We extracted more than 13000 relations for the Bicocca 2009-10-04 dataset. The resulting scan-matching map is shown in Figure 27. The map is topological incorrect and has major inconsistencies. The resulting map for RBPF is shown in Figure 28. The map looks topological correct but has some local inconsistencies in the lower left part of the map. The resulting map of GraphSLAM shown in Figure 29 looks also topological correct. There is one region that is locally inconsistent, one explanation for that could be that the scan matcher of GraphSLAM is confused by the vegetated area in that region. Figure 26 shows the corresponding error distributions



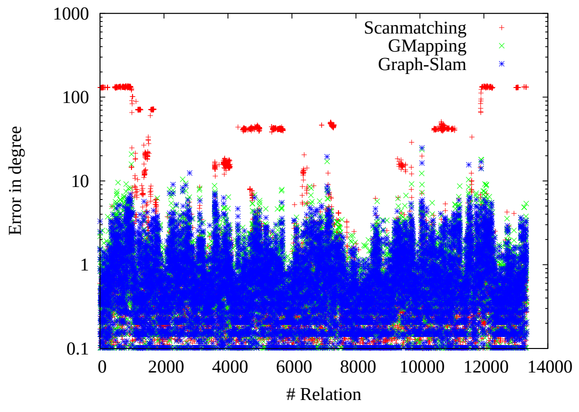


Figure 26: Evaluation of the Bovisa-2008-10-04 dataset using the defined metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees.

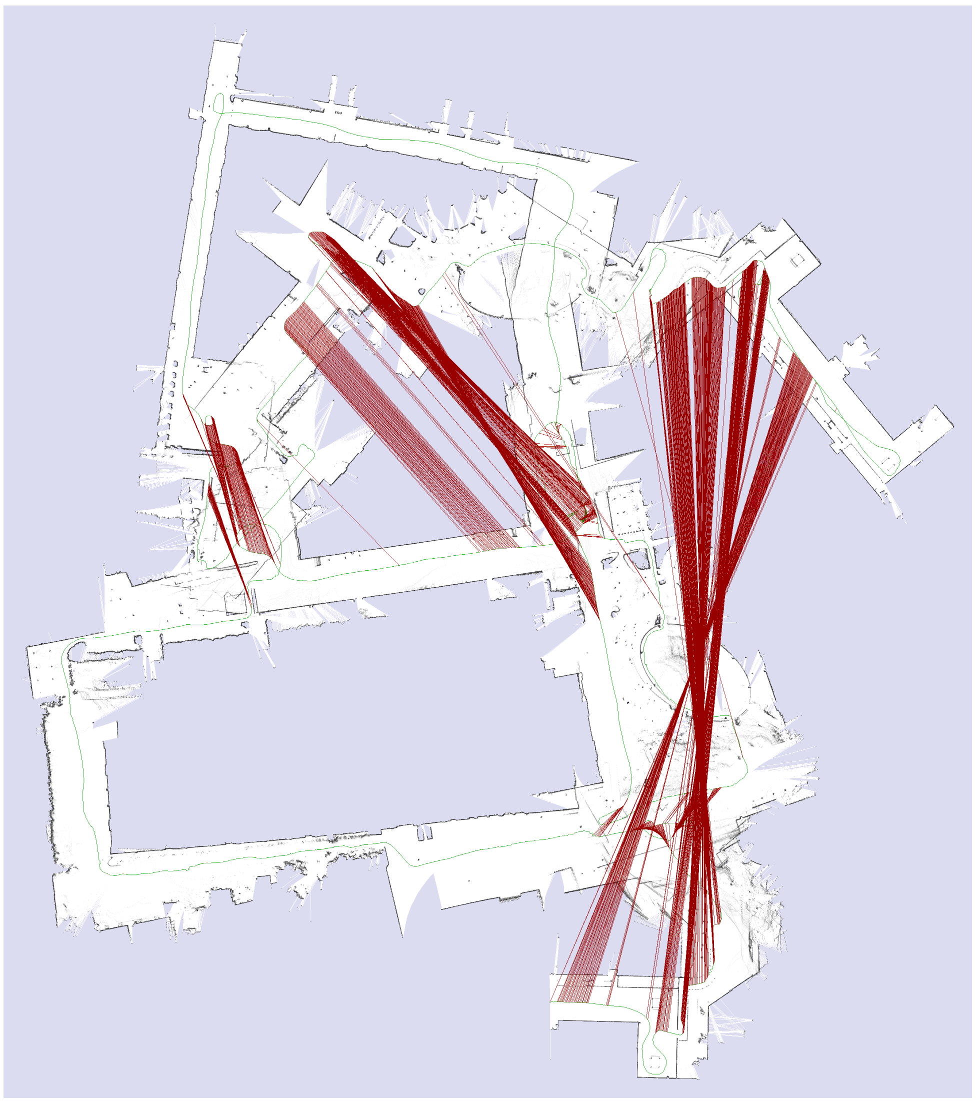


Figure 27: Result of the scan matching algorithm for the Bovisa 2008-10-04 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

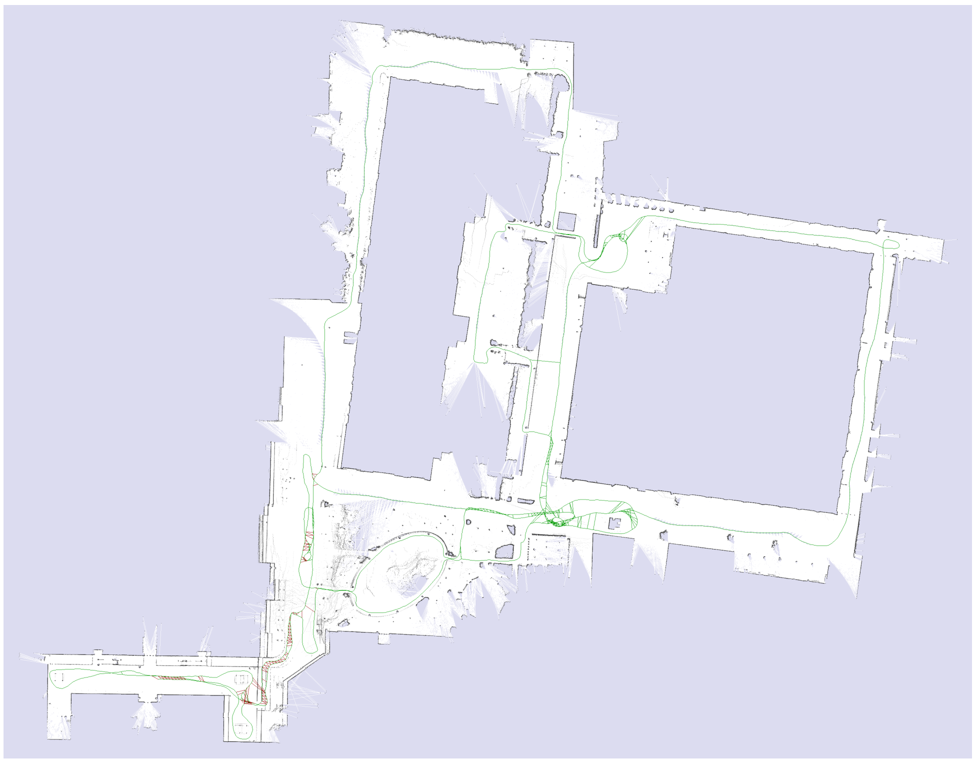


Figure 28: Result of RBPF for the Bovisa 2008-10-04 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

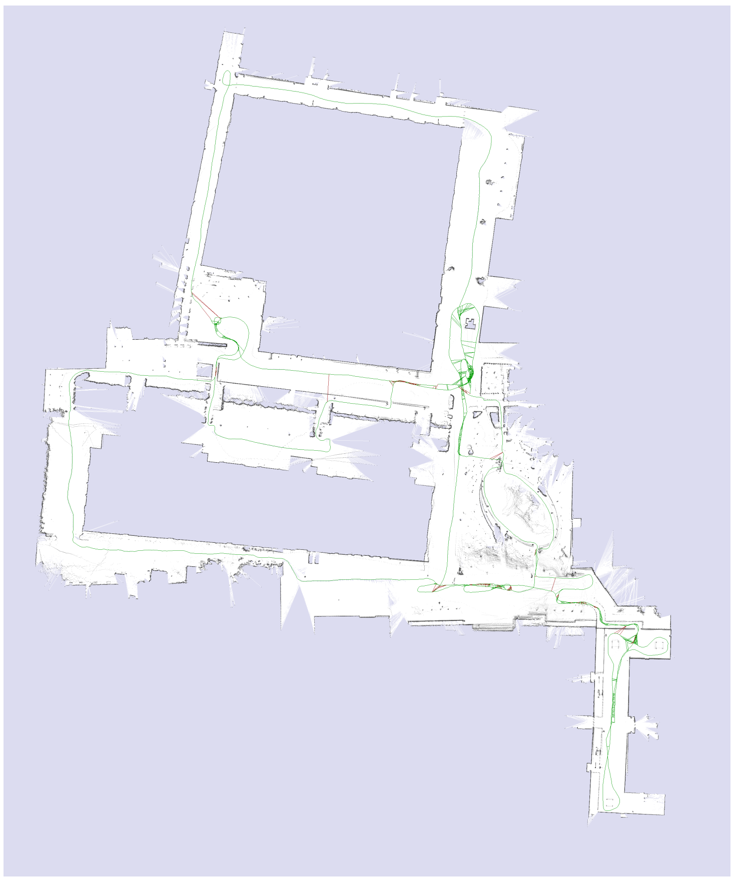
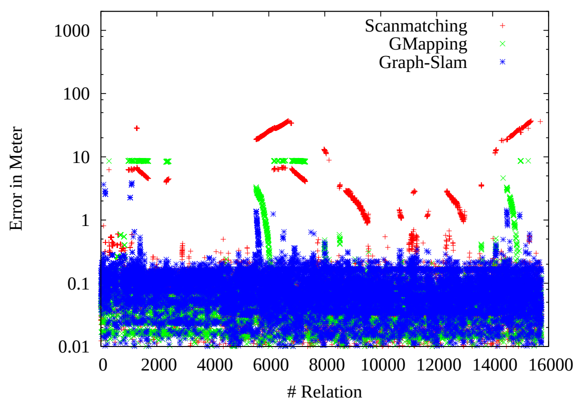


Figure 29: Result of GraphSLAM for the Bovisa 2008-10-04 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

3.4.10 Results for Bovisa 2008-10-06

The Bicocca 2009-10-06 relations set consists of close to 16000 nearby evaluation positions. Figure 31 shows the resulting scan matching map. Again the accumulated alignment error grows along the trajectory of the robot and this results in an inconsistent map. The resulting map for RBPF is shown in Figure 32. There are inconsistencies in the lower left area of the map, caused by a missed loop closure. The resulting map of GraphSLAM is shown in Figure 33. The map looks consistent and no major errors can be observed on it. Figure 30 shows the corresponding error distributions.



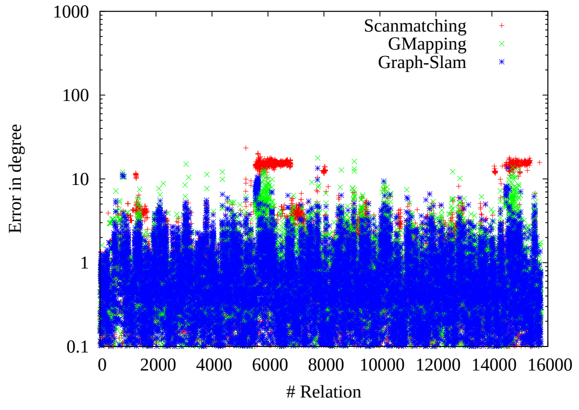


Figure 30: Evaluation of the Bovisa-2008-10-06 dataset using the defined metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees.

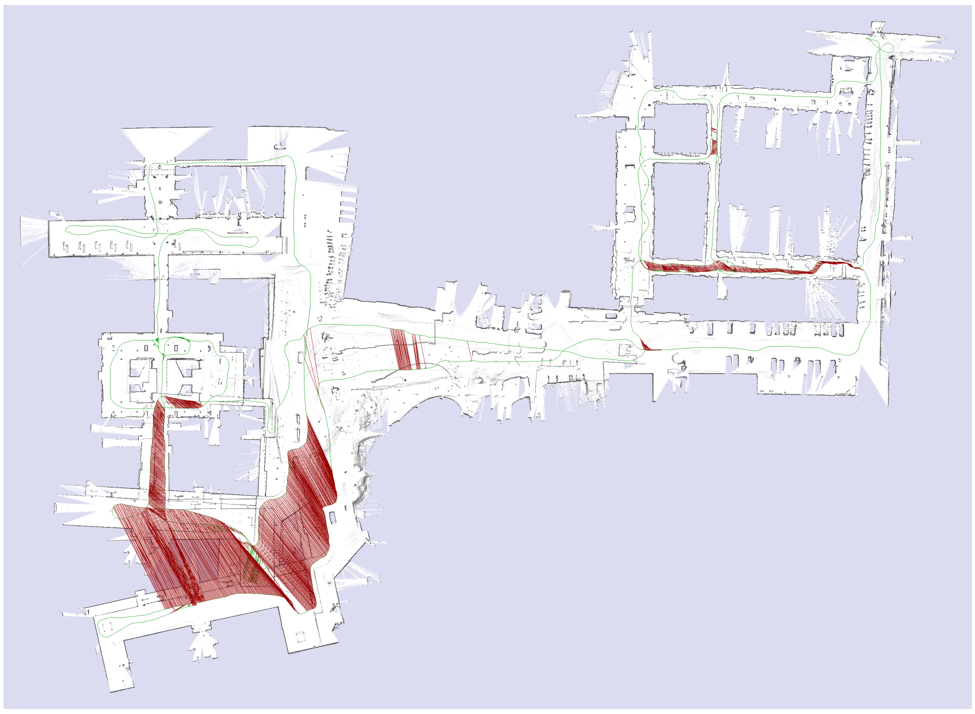


Figure 31: Result of the scan matching algorithm for the Bovisa 2008-10-06 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

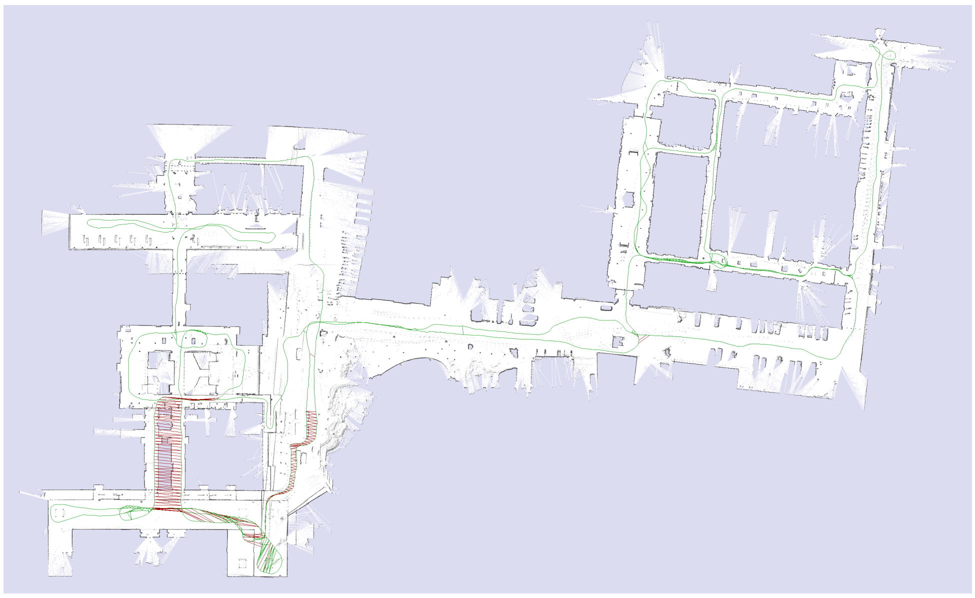


Figure 32: Result of RBPF for the Bovisa 2008-10-06 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

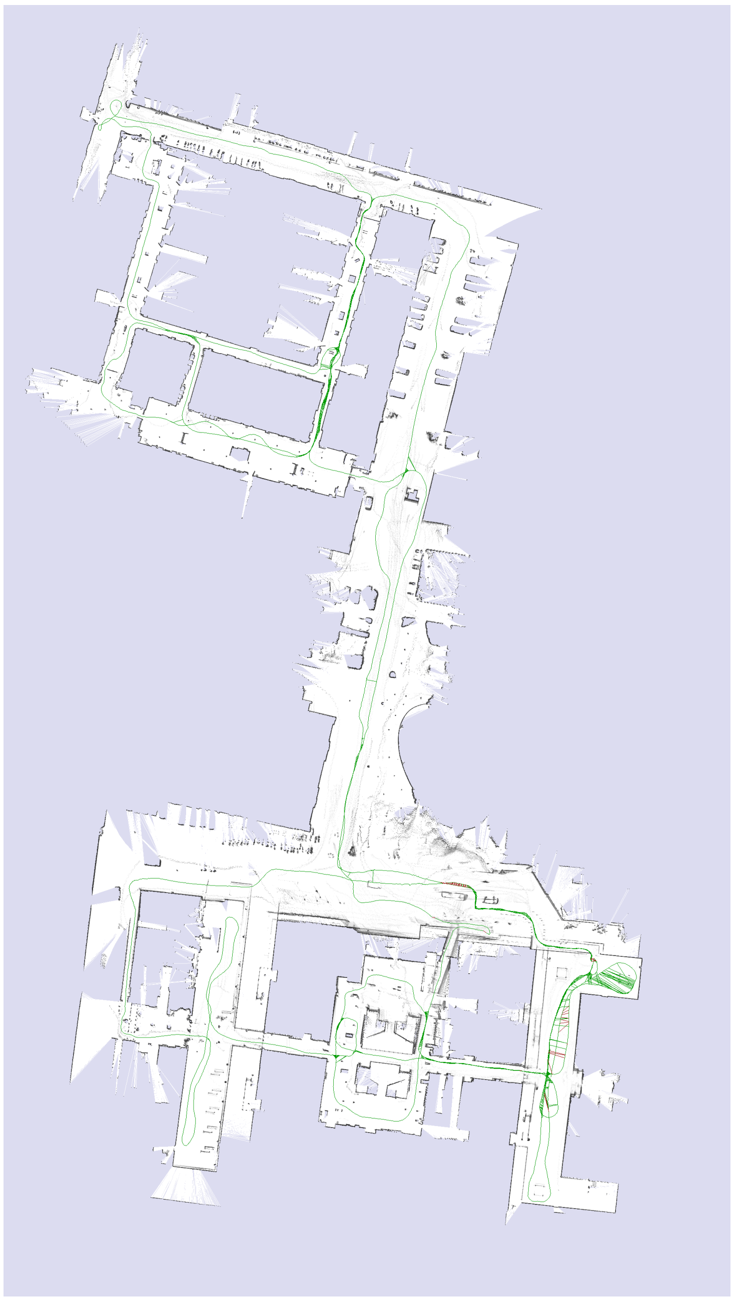


Figure 33: Result of GraphSLAM for the Bovisa 2008-10-06 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

3.4.11 Results for Bovisa 2008-10-07

For the Bicocca 2009-10-07 dataset we extracted more than 10000 nearby positions for the evaluation. The resulting scan matching map is shown in Figure 35. The accumulated scan matching error grows along the trajectory of the robot and results in a inconsistent map. The RBPF is not able to close the loops as can be seen in the resulting map in Figure 36. The RBPF map is inconsistent and topological incorrect. Figure 37 shows the resulting map for the GraphSLAM. The map has no major inconsistencies and is topological correct. Figure 34 shows the error distributions. In this setting the RBPF with 50 particles has the highest error peaks in Figure 34 and the resulting map also looks a bit worse than the scan matching map. Usually RBPF performs better than scan matching, but in this instance the particle filter diverged.

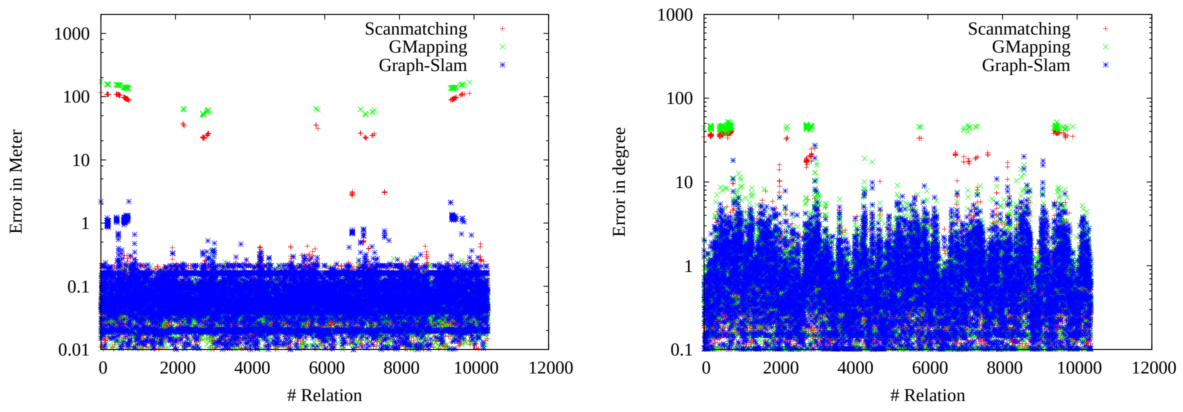


Figure 34: Evaluation of the Bovisa-2008-10-07 dataset using the defined metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees.

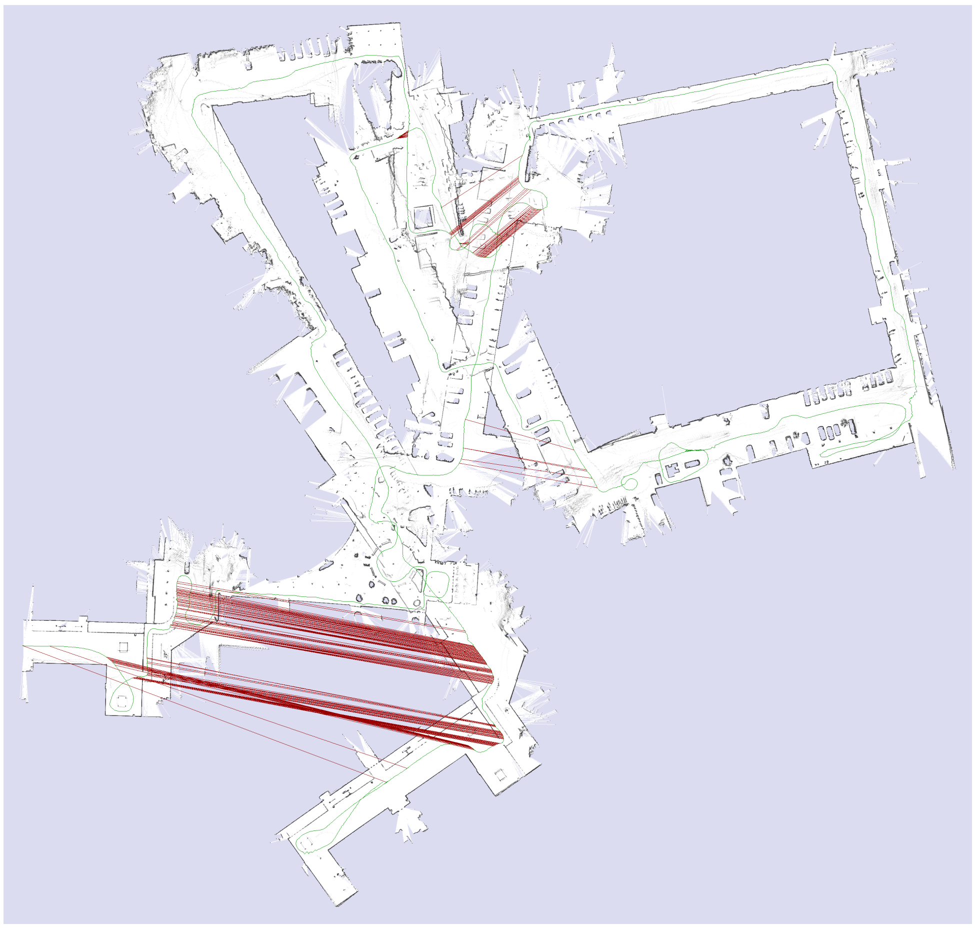


Figure 35: Result of the scan matching algorithm for the Bovisa 2008-10-07 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

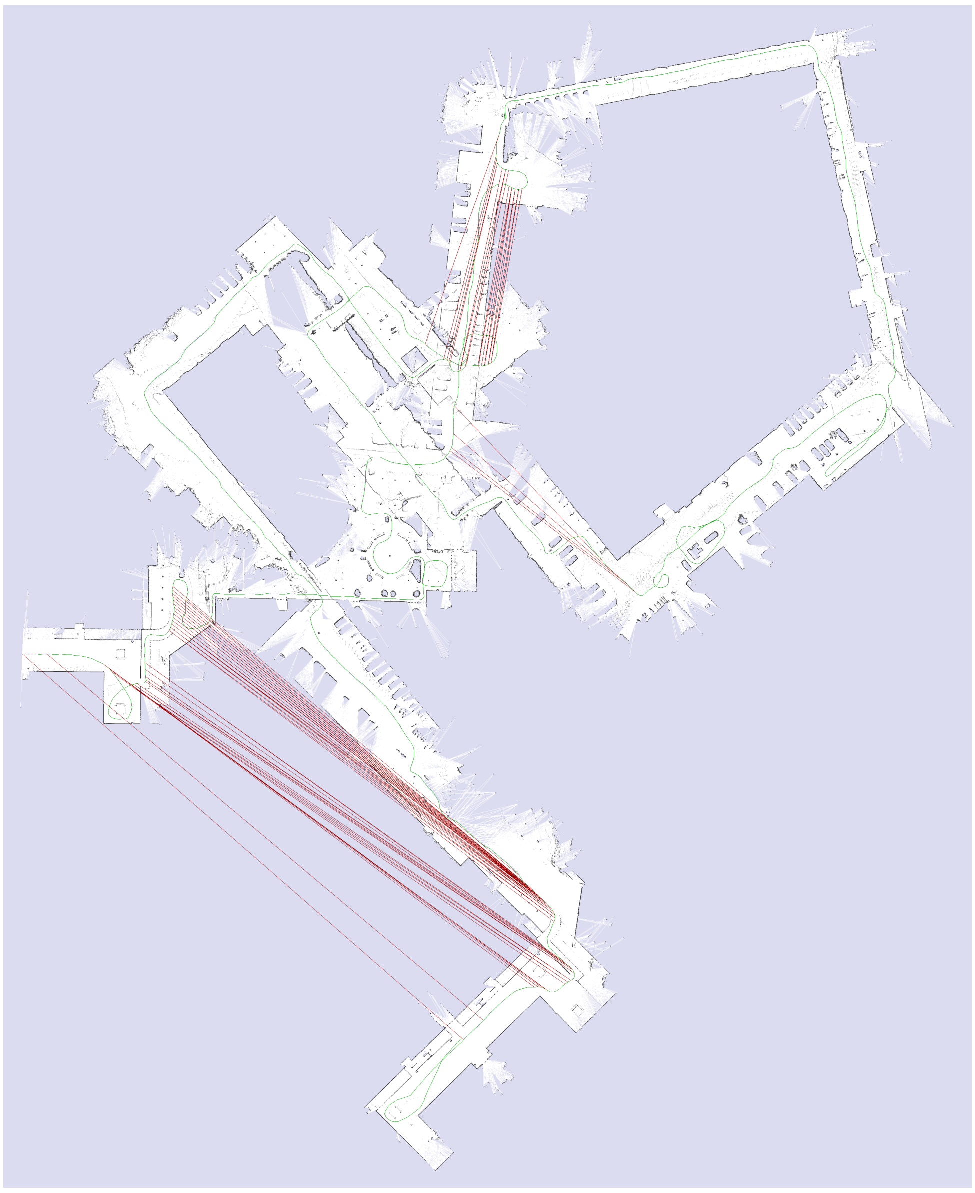


Figure 36: Result of RBPF for the Bovisa 2008-10-07 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

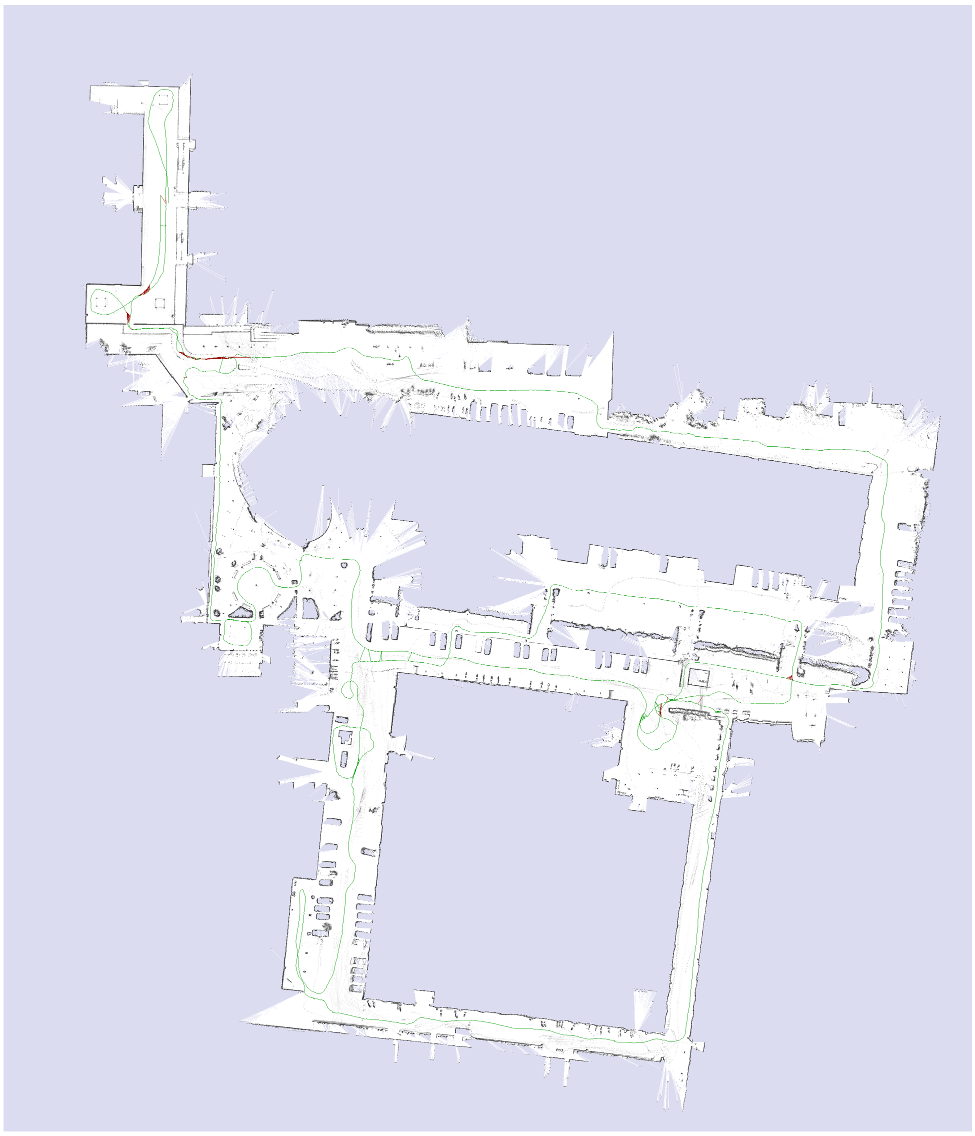


Figure 37: Result of GraphSLAM for the Bovisa 2008-10-07 dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

3.4.12 Results for Bovisa 2008-10-11a

We extracted close to 16000 nearby evaluation positions for the Bicocca 2009-10-11a dataset. The resulting maps for scan matching and RBPF shown in Figures 39, 40 are inconsistent and topological incorrect. It is hard to decide which map is better than the other. In contrast the error plots in Figure 38 tell that the result for the RBPF is worse than scan matching. The resulting GraphSLAM map is shown in Figure 41. The map is topological correct and no major errors can be observed.

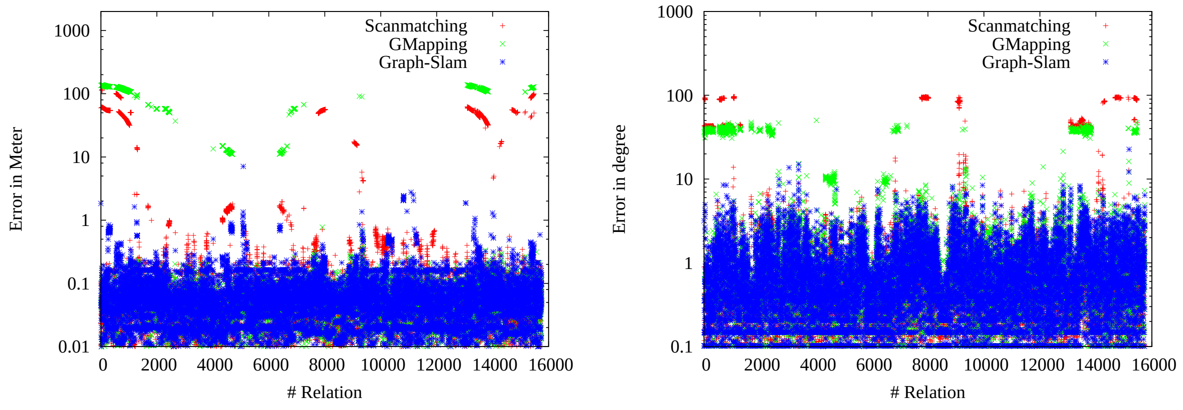


Figure 38: Evaluation of the Bovisa-2008-10-11a dataset using the defined metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees.

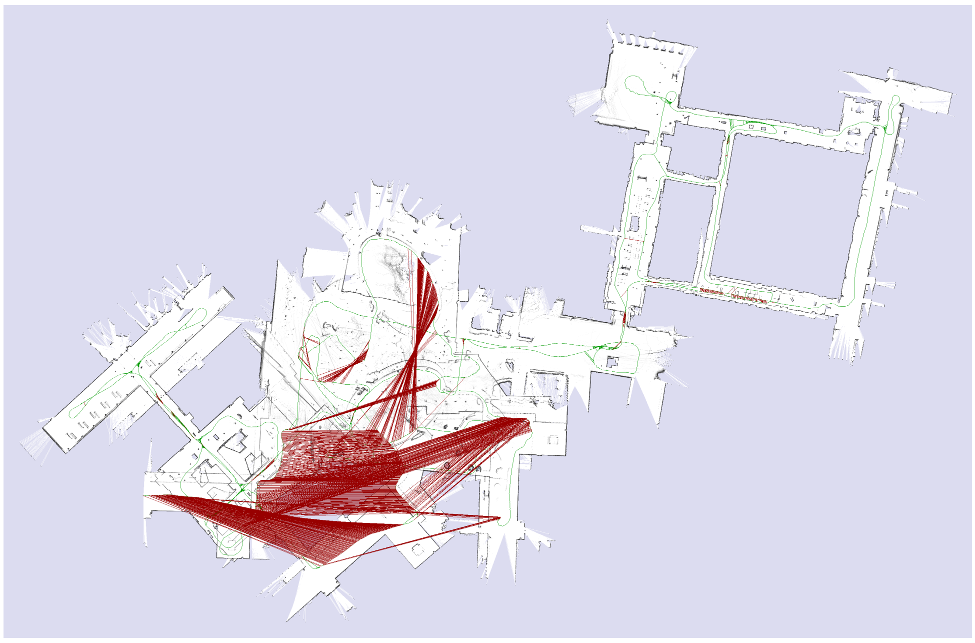


Figure 39: Result of the scan matching algorithm for the Bovisa 2008-10-11a dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

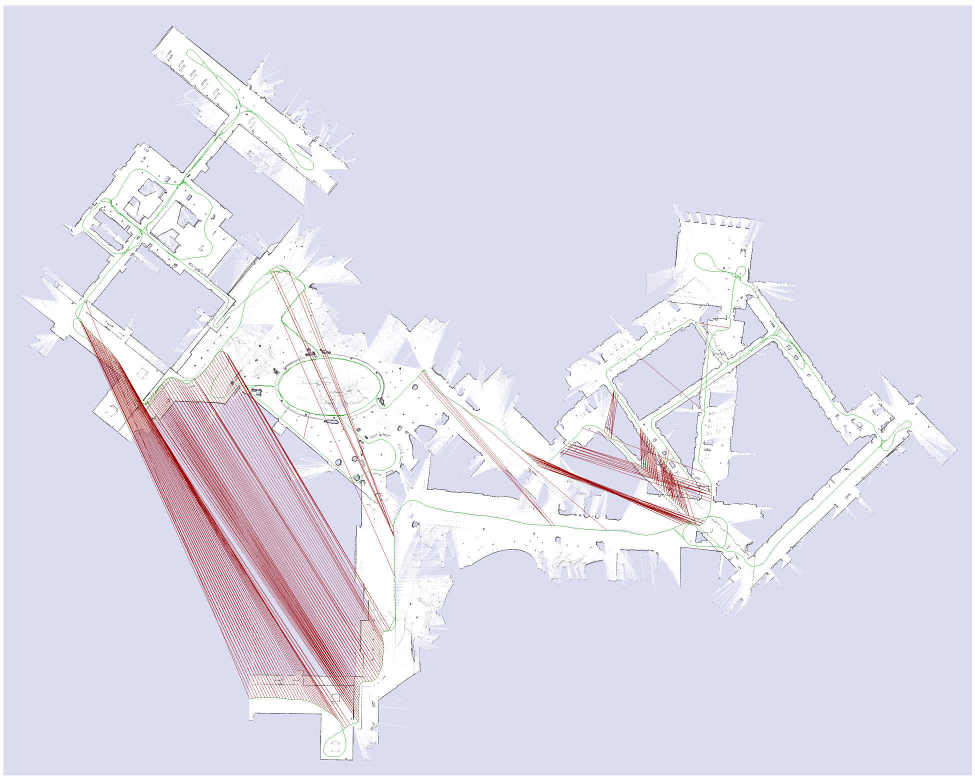


Figure 40: Result of RBPF for the Bovisa 2008-10-11a dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

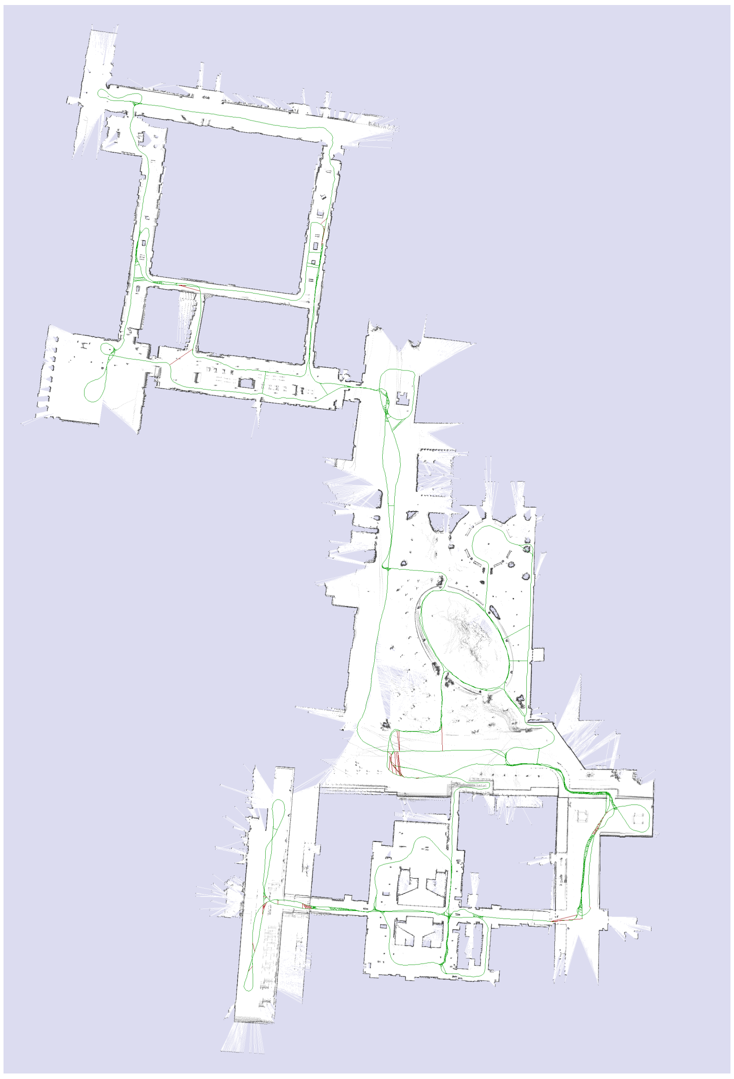
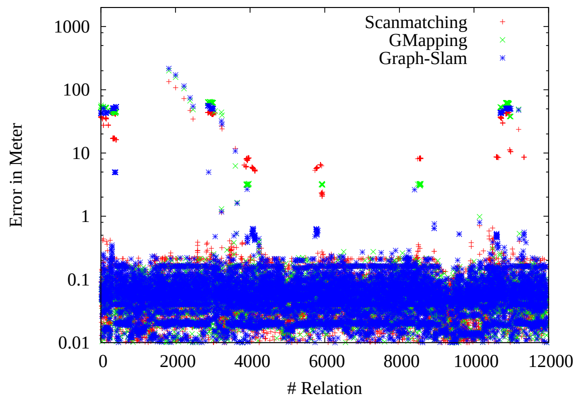


Figure 41: Result of GraphSLAM for the Bovisa 2008-10-11a dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

3.4.13 Results for Bovisa 2008-10-11b

The Bovisa 2008-10-11b is a more challenging dataset, because it has a known odometry error of about 90 degree in one occasion. This makes it quite hard for laser based SLAM approaches and should be an advantage for vision based odometry approaches. The Bicocca 2009-10-11b relations set consists of over 12000 relative evaluation position pairs. Figures 43, 44, 45 shows the results for scan matching, RBPF and GraphSLAM. All maps are inconsistent, this is caused by the known odometry problem. Figure 42 shows the error distributions. One interesting aspect is that GraphSLAM performs worse than RBPF looking at the error peaks in Figure 42 and RBPF also performs worse than pure scan matching.



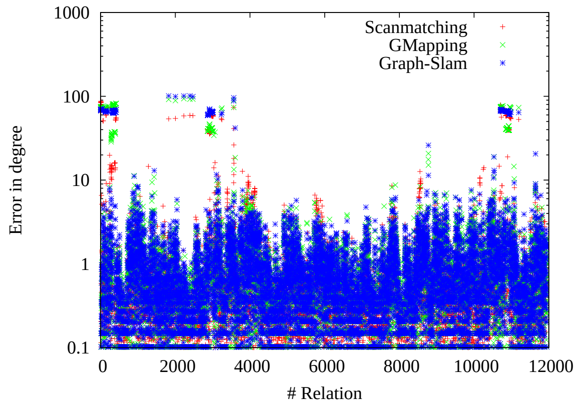


Figure 42: Evaluation of the Bovisa-2008-10-11b dataset using the defined metric. The left picture shows the translational error in meters and the right picture shows the rotational error in degrees.

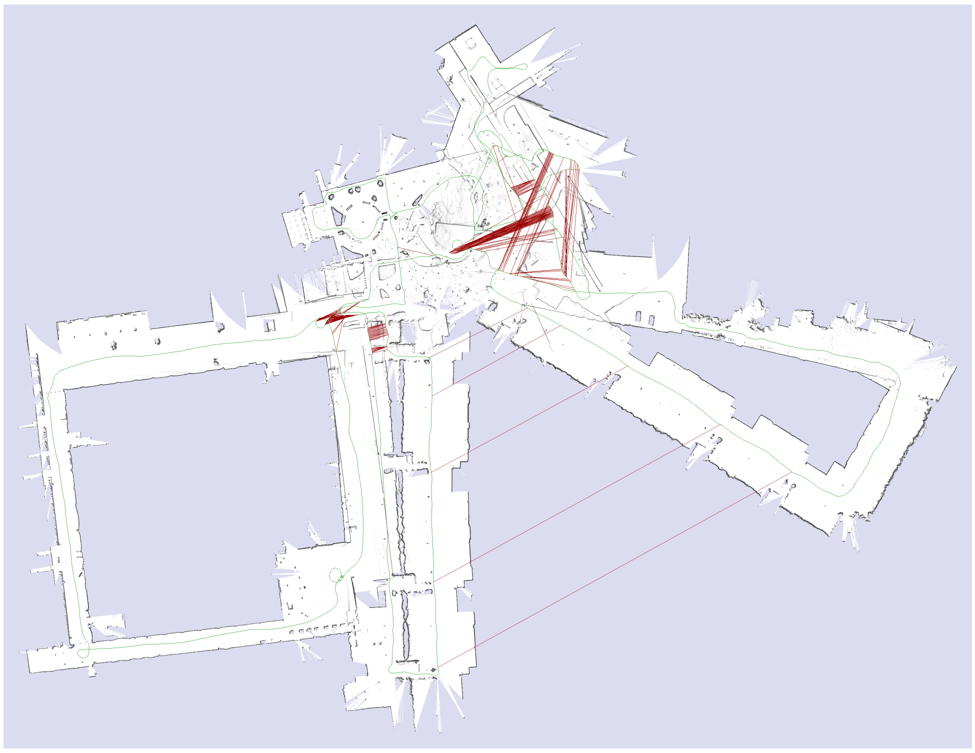


Figure 43: Result of the scan matching algorithm for the Bovisa 2008-10-11b dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

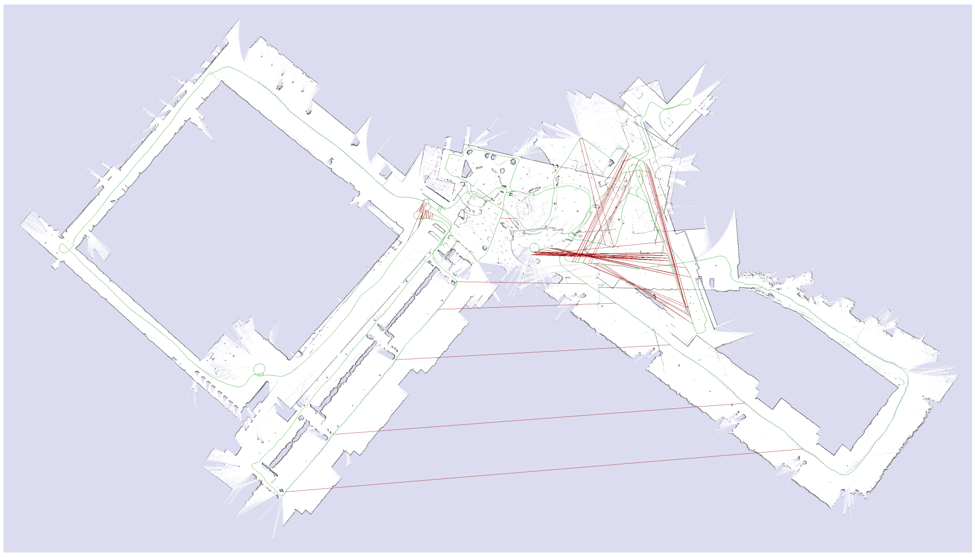


Figure 44: Result of RBPF for the Bovisa 2008-10-11b dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

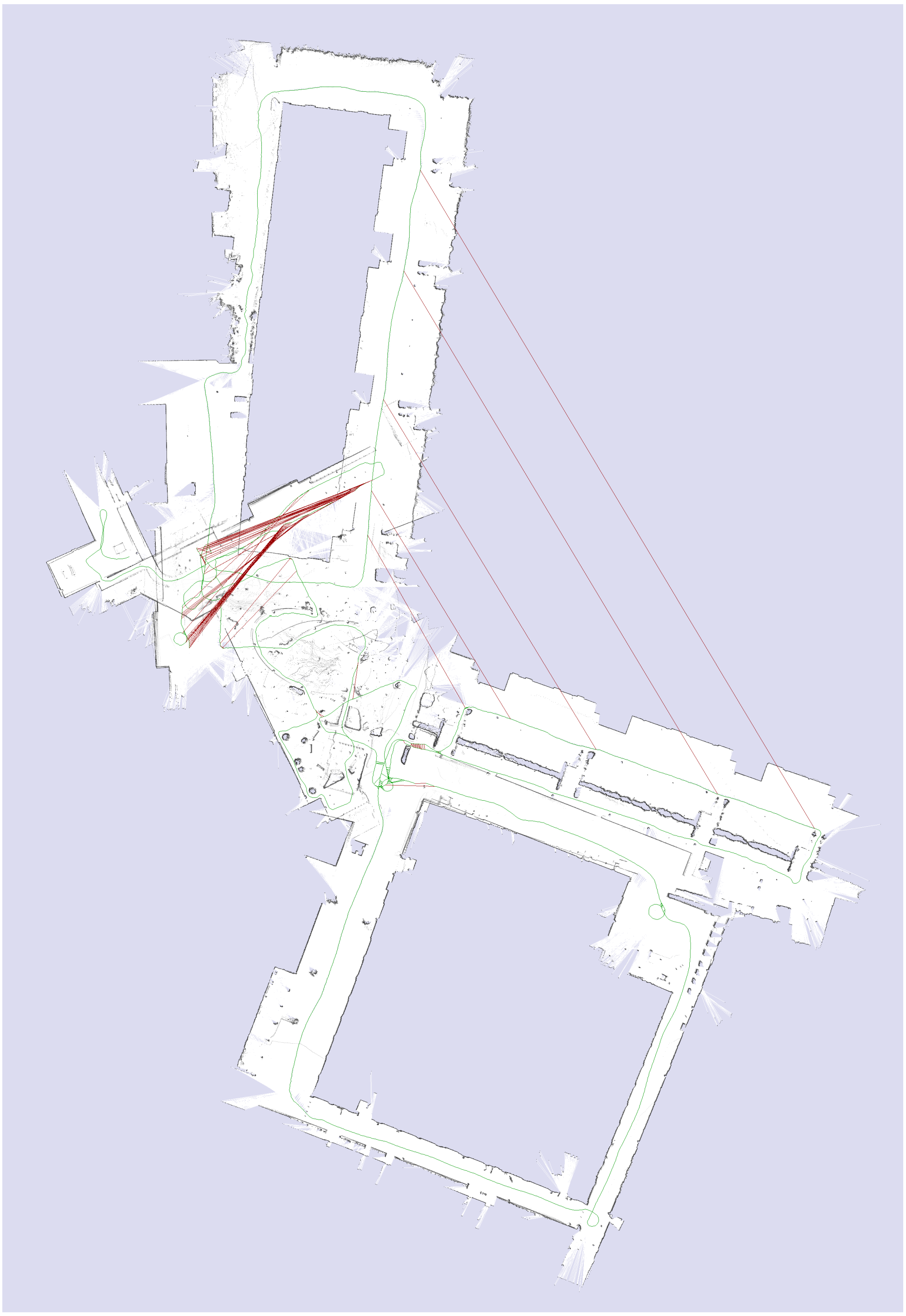


Figure 45: Result of GraphSLAM for the Bovisa 2008-10-11b dataset. Relations with an error over 1 m are colored red and errors below 1 m are colored green.

4 Benchmark solution: Monocular EKF-SLAM

This section briefly describes the algorithm used to provide benchmark solutions to the datasets using monocular and wheel odometry streams. We begin by giving full detail of the algorithm used in next subsection; and present in the following one results obtained over the outdoor dataset Bovisa_2008-10-04. It should be noticed that the algorithm presented is based on novel contributions developed in the framework of the RAWSEEDS project and that constitute the state of the art in filtering monocular sequences. Specifically, the algorithm is based on the following publications: [11, 12, 13].

4.1 Camera-Centered EKF + 1-Point RANSAC

An illustrative scheme of the algorithm used is shown in algorithm 1, and is fully detailed along the subsection. The recent key filtering contributions that are combined in this scheme are: first, a camera-centered representation of the geometric entities in the estimation [6] that reduces the linearization error for long exploration trajectories, which will be the case in the large environments of the dataset. Second; inverse depth parametrization for point features [11] that will allow undelayed point initialization and low-parallax points mapping improving the accuracy of the estimation. Finally, a 1-Point RANSAC algorithm [12, 13] will perform efficient outlier rejection based on filtering priors.

In the camera-centered representation, the estimation at every step k is parameterized as a multidimensional Gaussian $\mathbf{x}_k \sim \mathcal{N}\left(\hat{\mathbf{x}}_k, \mathbf{P}_k\right)$ that includes the location of the world reference frame $\mathbf{x}_W^C$ as a non-observable feature and the map $\mathbf{y}^C$, both in the current camera reference frame.

$$\hat{\mathbf{x}}_k^{C_k} = \begin{pmatrix} \hat{\mathbf{x}}_W^{C_k} \\ \hat{\mathbf{y}}^{C_k} \end{pmatrix} ; \quad \mathbf{P}_k^{C_k} = \begin{pmatrix} \mathbf{P}_W^{C_k} & \mathbf{P}_{Wy}^{C_k} \\ \mathbf{P}_{yW}^{C_k} & \mathbf{P}_y^{C_k} \end{pmatrix} . \tag{17}$$

The map $\mathbf{y}^{C_k}$ is composed of n point features $\mathbf{y}_i^{C_k}$ which are parametrized using inverse depth coordinates as detailed in [11]:

$$\hat{\mathbf{y}}^{C_k} = \begin{pmatrix} \hat{\mathbf{y}}_1^{C_k} \\ \vdots \\ \hat{\mathbf{y}}_n^{C_k} \end{pmatrix} ; \quad \mathbf{P}_y^{C_k} = \begin{pmatrix} \mathbf{P}_{y_1}^{C_k} & \cdots & \mathbf{P}_{y_1 y_n}^{C_k} \\ \vdots & \ddots & \vdots \\ \mathbf{P}_{y_n y_1}^{C_k} & \cdots & \mathbf{P}_{y_n}^{C_k} \end{pmatrix} . \tag{18}$$

Location for the world reference frame with respect to the current camera frame is represented by its position vector and quaternion orientation

$$\hat{\mathbf{x}}_W^{C_k} = \begin{pmatrix} \hat{\mathbf{r}}_W^{C_k} \\ \hat{\mathbf{q}}_W^{C_k} \end{pmatrix} . \tag{19}$$

The algorithm integrating Extended Kalman Filter and 1-Point RANSAC outlier rejection can be divided in five stages, which are described below.

Algorithm 1 Camera-Centered EKF + 1-Point RANSAC

```
INPUT: $\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}$ {EKF estimate at step $k-1$}
          $th$ {Threshold for low-innovation points.}
               {In this paper, $th = 1.0\ pixels$}
OUTPUT: $\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}$ {EKF estimate at step $k$}

{A. EKF prediction and individually compatible matches}
$[\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}] = EKF\_prediction(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1}, \mathbf{u})$
$[\hat{\mathbf{h}}_{k|k-1}, \mathbf{S}_{k|k-1}] = measurement\_prediction(\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})$
$\mathbf{z}^{IC} = search\_IC\_matches(\hat{\mathbf{h}}_{k|k-1}, \mathbf{S}_{k|k-1})$

{B. Get a reliable set of low-innovation inliers}
$\mathbf{z}^{li-inliers} = [\,]$
for $i = 0$ to $n_{hyp}$ do
    $\mathbf{z}_i = select\_match(\mathbf{z}^{IC})$
    $\hat{\mathbf{x}}_i = EKF\_state\_update(\mathbf{z}_i, \hat{\mathbf{x}}_{k|k-1})$ {Notice: only state update; NO covariance update}
    $\mathbf{h}_i = predict\_all\_measurements(\hat{\mathbf{x}}_i)$
    $\mathbf{z}_i^{th} = find\_matches\_below\_a\_threshold(\mathbf{z}^{IC}, \mathbf{h}_i, th)$
    if $size(\mathbf{z}_i^{th}) > size(\mathbf{z}^{li-inliers})$ then
        $\mathbf{z}^{li-inliers} = \mathbf{z}_i^{th}$
    end if
end for

{C. Partial EKF update using low-innovation inliers}
$[\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}] = EKF\_update(\mathbf{z}^{li-inliers}, \hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1})$

{D. Partial EKF update with high-innovation inliers}
$\mathbf{z}^{hi-inliers} = [\,]$
for every match $j$ above a threshold $th$ do
    $[\mathbf{h}^j, \mathbf{S}^j] = point\_j\_prediction\_and\_covariance(\hat{\mathbf{x}}, \mathbf{P}, j)$
    $\nu^{\mathbf{j}} = \mathbf{z}^j - \mathbf{h}^j$
    if $\nu^{\mathbf{j}\top} \mathbf{S}^{\mathbf{j}^{-1}} \nu^{\mathbf{j}} < \chi^2_{\mathbf{2},\mathbf{0.01}}$ then
        $\mathbf{z}^{hi-inliers} = add\_match\_j\_to\_inliers(\mathbf{z}^{hi-inliers}, \mathbf{z}^j)$ {If individually compatible, add to
        inliers}
    end if
end for
if $size(\mathbf{z}^{hi-inliers}) > 0$ then
    $[\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}] = EKF\_update(\mathbf{z}^{hi-inliers}, \hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k})$
end if

{E. Composition step}
$[\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}] = composition(\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k})$
```

4.1.1 EKF Prediction and Individually Compatible Matches

For the prediction step at time k, the world reference frame and feature map are kept in the reference frame at time $k-1$ and a new feature that represents the motion of the sensor between $k-1$ and k is added:

$$\hat{\mathbf{x}}_{k|k-1}^{C_{k-1}} = \begin{pmatrix} \hat{\mathbf{x}}_W^{C_{k-1}} \\ \hat{\mathbf{y}}^{C_{k-1}} \\ \hat{\mathbf{x}}_{C_k}^{C_{k-1}} \end{pmatrix} \tag{20}$$

Predicted camera motion will be taken from the odometry measurements in the dataset; and predicted covariance will be obtained by linearizing the model and adding zero-mean Gaussian noise.

For each mapped feature $\hat{\mathbf{y}}_i^{C_k}$, its correspondence $\mathbf{z}_i$ will be searched in the 99% probability region from the predicted Gaussian pdf $\mathcal{N}\left(\hat{\mathbf{h}}_i, \mathbf{S}_i\right)$

$$\hat{\mathbf{h}}_i = \mathbf{h}_i\left(\hat{\mathbf{x}}_{k|k-1}^{C_{k-1}}\right) \tag{21}$$

$$\mathbf{S}_i = \mathbf{H}_i \mathbf{P}_{k|k-1}^{C_{k-1}} \mathbf{H}_i^\top + \mathbf{R}_i \ , \tag{22}$$

where $\mathbf{h}_i$ is the projection model –a pinhole camera with radial distortion as in [11] will be used–; $\mathbf{H}_i$ is the derivative of the projection model by the state vector and $\mathbf{R}_i$ is the covariance of the measurement noise.

This correspondence set, searched in the 99% probability region for each feature, is said to be individually compatible: that means, each match is compatible with the a priori model separately. But that does not imply that the whole set is jointly compatible; that is, compatible with the prior information when the whole set of matches is considered. There can exist outliers that are individually compatible –when evaluated separately from other matches– but not jointly compatible if they are evaluated with other matches. The algorithm described from here will extract from the initial set of individually compatible matches a jointly compatible set rejecting spurious data (see Fig. 46).

4.1.2 Selection of Low-Innovation Inliers Using 1-Point RANSAC

The hypothesize-and-verify loop follows here, and it is where the key difference with standard RANSAC arises. While standard RANSAC needs a certain number of points that depends on the particular problem to hypothesize a model, having prior knowledge coming from filtering will permit to propose hypothesis using only 1 data point. The advantage of requiring 1 point resides on the fact that it is easier to randomly select an spurious-free random sample; reducing then the number of samples and the associated computational cost.

Another key aspect for the efficiency of the algorithm is that only a state vector update with one match is needed to compute an hypothesis, which is computationally cheap compared with the quadratic complexity of the whole EKF covariance update. This means that the hypothesis generation process will have in general a negligible cost compared with the standard EKF operations.

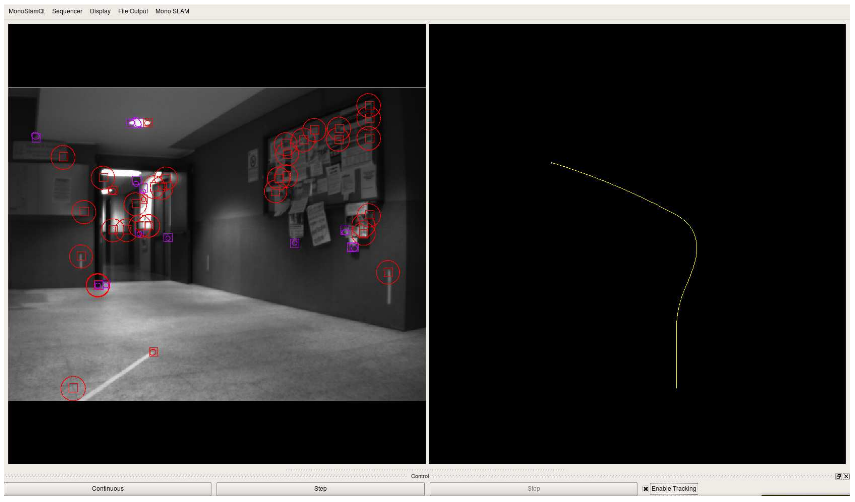


Figure 46: Capture of the Monocular SLAM system performing tracking of features on the sequence (left). A few set of predicted features (magenta) are rejected by 1-Point RANSAC while most of the features are correctly associated (red). Part of the estimated trajectory (right).

Following with the algorithm; support given by the matches for each hypothesis is computed simply counting the number of data points inside a heuristic threshold. Threshold should be chosen close to the measurement noise, which is what was done in this paper setting the value to 1 pixel.

4.1.3 Partial Update with Low-Innovation Inliers

Most supported hypothesis among the randomly constructed in previous step is selected. As threshold was chosen to be close to the measurement noise, all the matches $\mathbf{z}_i$ inside can be considered to be inliers having low innovation $\mathbf{z}^{li-inliers}$. The rest of the high-innovation measurements will be either high-innovation inliers, corresponding to either close points affected by translation or recently initialized points, or outliers.

A partial update of the covariance using this low innovation inliers will reduce most of the correlated priors in the Gaussian prediction. It is important to notice here that, while JCBB uses correlations between parameters to reject outliers, the approach of this paper first remove most of these correlations. After this removal, checking individual compatibility is enough to discard outliers.

4.1.4 Partial Update with High-Innovation Inliers

Individually compatible matches after the first partial update will be finally classified as inliers. Computation done in the first update serves as the starting point for the second partial update using rescued high-innovation inlier measurements. As in the first partial update, standard EKF update formulation will be applied.

It is important to remark here that, if it is true that dividing the covariance update step into two parts will introduce a extra computational overhead with respect to the case of a global update, such overhead will be of little importance compared with other computations of the filter –particularly covariance update, which is the most expensive computation.

4.1.5 Composition

After the update, a final composition step will be necessary in order to transform all the geometric entities in the filter from previous camera reference frame to the current one.The rigid transformation between the previous frame of reference and the current one is removed from the estimation. The resulting state vector is:

$$\hat{\mathbf{x}}_k^{C_k} = \begin{pmatrix} \hat{\mathbf{x}}_W^{C_k} \\ \hat{\mathbf{x}}_v^{C_k} \\ \hat{\mathbf{y}}^{C_k} \end{pmatrix} , \tag{23}$$

where $\hat{\mathbf{x}}_W^{C_k}$, $\hat{\mathbf{x}}_v^{C_k}$ and $\hat{\mathbf{y}}^{C_k}$ have been computed by composition with the motion between frames $\hat{\mathbf{x}}_{C_k}^{C_{k-1}}$:

$$\begin{aligned} \hat{\mathbf{x}}_W^{C_k} &= \ominus\hat{\mathbf{x}}_{C_k}^{C_{k-1}} \oplus \hat{\mathbf{x}}_W^{C_{k-1}} & (24)\\ \hat{\mathbf{x}}_v^{C_k} &= \ominus\hat{\mathbf{x}}_{C_k}^{C_{k-1}} \oplus \hat{\mathbf{x}}_v^{C_{k-1}} & (25)\\ \hat{\mathbf{y}}^{C_k} &= \ominus\hat{\mathbf{x}}_{C_k}^{C_{k-1}} \oplus \hat{\mathbf{y}}^{C_{k-1}} . & (26) \end{aligned}$$

The final covariance is computed using the Jacobian of the composition equation $\mathbf{J}_{C_{k-1}\to C_k}$:

$$\mathbf{P}_k^{C_k} = \mathbf{J}_{C_{k-1}\to C_k}\mathbf{P}_k^{C_{k-1}}\mathbf{J}_{C_{k-1}\to C_k}^\top . \tag{27}$$

4.2 Results using Bovisa_2008-10-04 dataset

4.2.1 Methodology

In order to compare with RTK GPS, the trajectory provided by the filtering has to be previously aligned applying a rotation transformation over it

$$\begin{bmatrix} \mathbf{r}_{C_k}^W \\ 1 \end{bmatrix} = \begin{bmatrix} x_{C_k}^W \\ y_{C_k}^W \\ z_{C_k}^W \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{C_0}^W & \mathbf{t}_{C_0}^W \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} x_{C_k}^{C_0} \\ y_{C_k}^{C_0} \\ z_{C_k}^{C_0} \\ 1 \end{bmatrix} . \tag{28}$$

Translation offset $\mathbf{t}_{C_0}^W$ will be taken from the first GPS measurement, while rotation $\mathbf{R}_{C_0}^W$ will be obtained by minimizing the distance between the two trajectories –whole GPS and filtered trajectories– allowing a rotation motion.

Finally, the error of each camera position in the reconstructed path is computed as the Euclidean distance between each point of the estimated camera path and GPS path, both in the W reference,

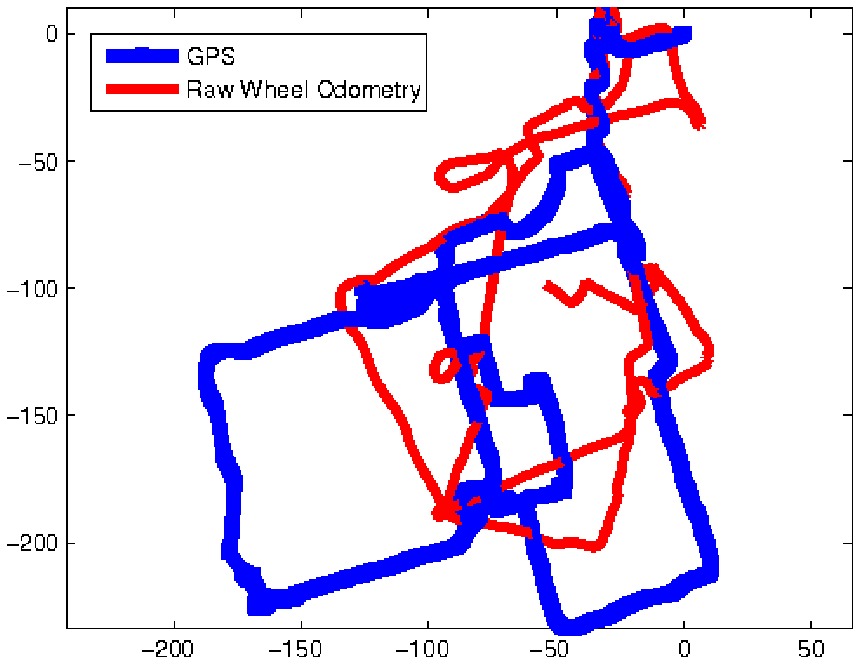


Figure 47: Raw odometry measurements (thin red) and GPS ground truth (thick blue). Errors of raw odometry are caused by early drift typical from proprioceptive sensors.

$$e_k = \sqrt{\left(\mathbf{r}^W_{C_k} - \mathbf{r}^W_{GPS_k}\right)^\top \left(\mathbf{r}^W_{C_k} - \mathbf{r}^W_{GPS_k}\right)}. \tag{29}$$

4.2.2 Monocular SLAM Results

The dataset chosen to provide a benchmark solution is Bovisa_2008-10-04. The length of the estimated trajectory is about 1310 meters long and was covered by RAWSEEDS mobile robot in 30 minutes being then the length of the sequence 54000 frames. Trajectory obtained from the combination of raw odometry and monocular information explained above is shown in figure 49; together with GPS ground truth for comparison. Maximum and mean error compared against GPS ground truth are 23.6 and 9.8 meters respectively.

Next figures are devoted to highlight the main inconveniences of raw odometry and monocular camera alone; and how the combination of the two sensors is able to overcome the drawbacks that both of them show separately. Figure 47 shows raw odometry lectures as a red thin line and GPS ground truth with a blue thick line for comparison. It can be observed that early drift appears and plotted trajectory is rather far from the ground truth value.

Figure 48 shows pure monocular estimation in thin red and GPS measurements in thick green. Observing carefully this plot, it can be observed that monocular camera is able to very accurately estimate orientation; but the unobservability of the scale produces drift in this parameter for the number of tracked features considered (25). Tracking a larger number of features as in [12] will reduce this scale drift, but real-time capabilities of the algorithm would be lost. Also, global scale is not recovered: in figure 48, scale had to be adjusted to a factor

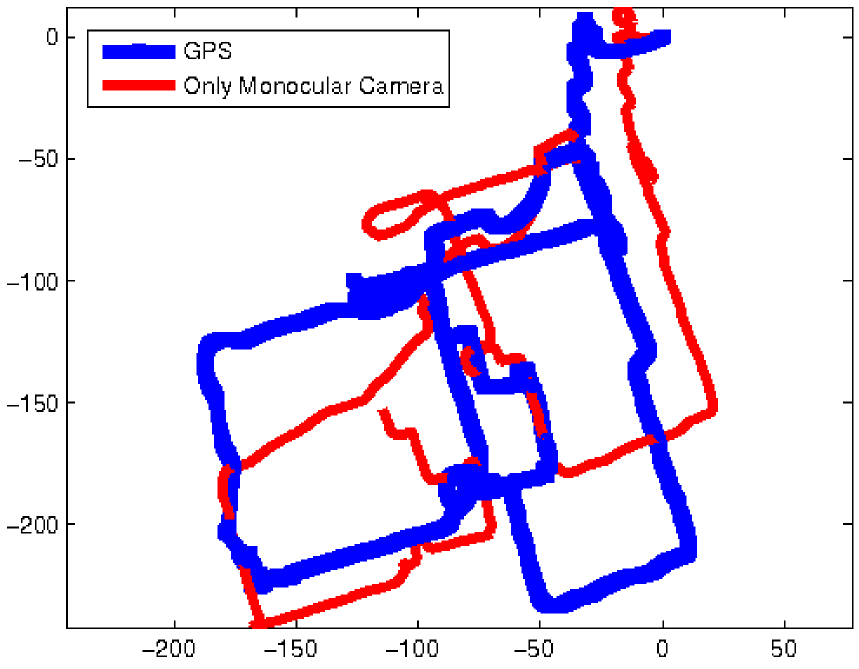


Figure 48: Pure monocular estimation (thin red) tracking 25 features and GPS ground truth (thick blue). Errors in this case are caused by scale drift, which is unobservable by a monocular camera.

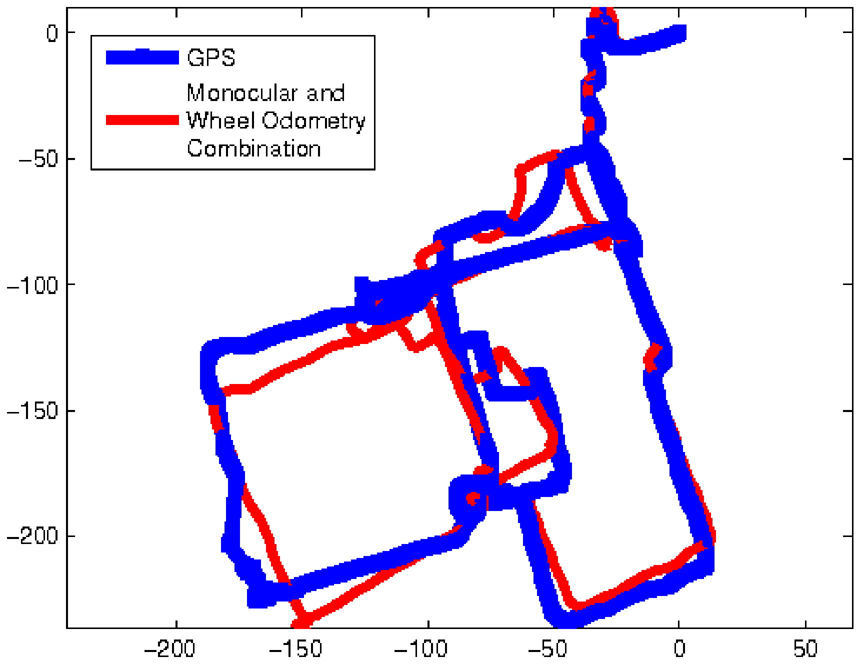


Figure 49: Monocular SLAM estimation from the combination of monocular camera plus wheel odometry (thin red) and GPS trajectory (thick blue).

of 2.7 via a minimization process.

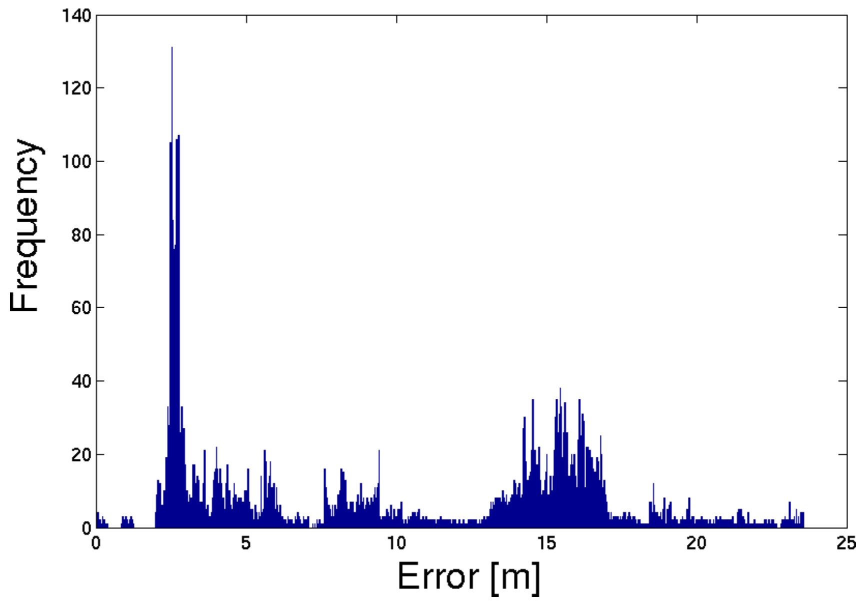


Figure 50: Histogram of the Monocular SLAM errors compared with GPS ground truth.

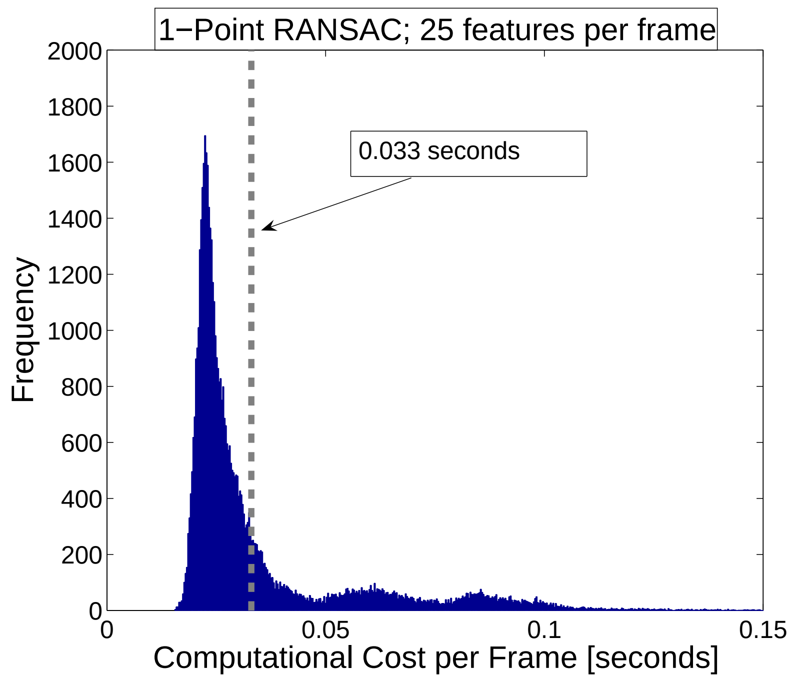


Figure 51: Histogram of the computational cost for 1-Point RANSAC when 25 image features are tracked.

Finally, figure 49 details the estimated trajectory that can be achieved from the combination of the two sensors. It can be seen that problems commented in two previous figures disappear. An accurate estimation is achieved for a trajectory of 1.3 kilometers.

Figure 50 shows the histogram of the errors for the sequence. The number of tracked features per frame was kept in 25 for this experiment.

The processing time per frame for this experiment can be observed in figure 51 in the form of an histogram. It can be noticed that, although the proposed algorithm still does not entirely run at real time at 30 frames per second, it is really close: 70% of the frames are already under 33 miliseconds.

5 Benchmark Solutions: Stereo CI-Graph SLAM

In this section we describe our multicamera SLAM system which integrates a set of novel technologies allowing us to gather most of the information available in the images. We consider information from features both close and far from the cameras [42]. Given a bunch of cameras and the rigid transformation between them, 3D information from nearby scene points can be obtained, at the same time each camera can also provide bearing only information from distant scene points. Both types of information are relevant to obtain good estimates of the translation and the attitude of the camera system. The first main contribution of the proposed method is that it can easily deal with any number of cameras since each camera is treated independently.

The second main contribution is a novel SLAM algorithm that allows us to efficiently build maps of large environments when the camera system follows complex trajectories. Our algorithm is able to operate in large environments by decomposing the whole map in local-maps of limited size. Instead of building independent submaps we build conditionally Independent submaps [44], which allow the system to share both camera velocity information and current feature information during local map initialization. This adds robustness to the system without sacrificing precision or consistency in any way. Finally, by using the CI-Graph algorithm [43] we can extend the properties of the CI-submaps to more complex robot trajectories and map topologies.

We validated the approach on the proposed indoor dataset Bicooca_2009-02-25b for benchmarking which has been described and tested in Deliverable 3.2 of the RAWSEEDS project. Due to the reported lack of texture in the corresponding dataset validation, we integrate odometry readings to drive the stereo system in those places where features can not be extracted from the images.

5.1 CI-Graph Algorithm Description

In order to work with complex topologies, the algorithm proposed is based on building an undirected graph of the CI-submaps. An undirected graph is defined as a pair $\mathcal{G} = (\mathcal{N}, \mathcal{E}_\mathcal{G})$ where $\mathcal{N}$ are the *nodes* of $\mathcal{G}$ and $\mathcal{E}_\mathcal{G}$ are its undirected *edges*. In our graph, $\mathcal{N}$ is the set of CI-submaps $\mathbf{m}_i$ with $i = 1 \ldots N$. An edge connecting two nodes is created either because the robot makes a transition between the corresponding submaps or because being the robot in a submap, it observes a feature that belongs to the other submap.

In addition, the algorithm builds a spanning tree $\mathcal{T}(\mathcal{N}, \mathcal{E}_\mathcal{T})$ of the graph $\mathcal{G}$, where $\mathcal{E}_\mathcal{T} \subset \mathcal{E}_\mathcal{G}$. A spanning tree $\mathcal{T}$ of a connected undirected graph $\mathcal{G}$ is defined as a subgraph of $\mathcal{G}$ which is a tree (it contains no cycles) and connects all the nodes. Our algorithm ensures that, by construction, any pair of submaps $(\mathbf{m}_i, \mathbf{m}_j)$ that are adjacent in $\mathcal{T}$ have a conditionally independent structure, sharing some vehicle and feature states. Each edge in $\mathcal{E}_\mathcal{T}$ will be labeled with the corresponding shared states. Given any pair of submaps, $\mathbf{m}_i$ and $\mathbf{m}_j$, there is a unique path in $\mathcal{T}$ connecting them. This path allows us to transmit information from map to map without loosing the conditional independence property between submaps. In figure 52, spanning tree edges $\mathcal{E}_\mathcal{T}$ will be depicted using a continuous line while the remaining edges of $\mathcal{G}$, i.e. $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$, will be traced with a dashed line.

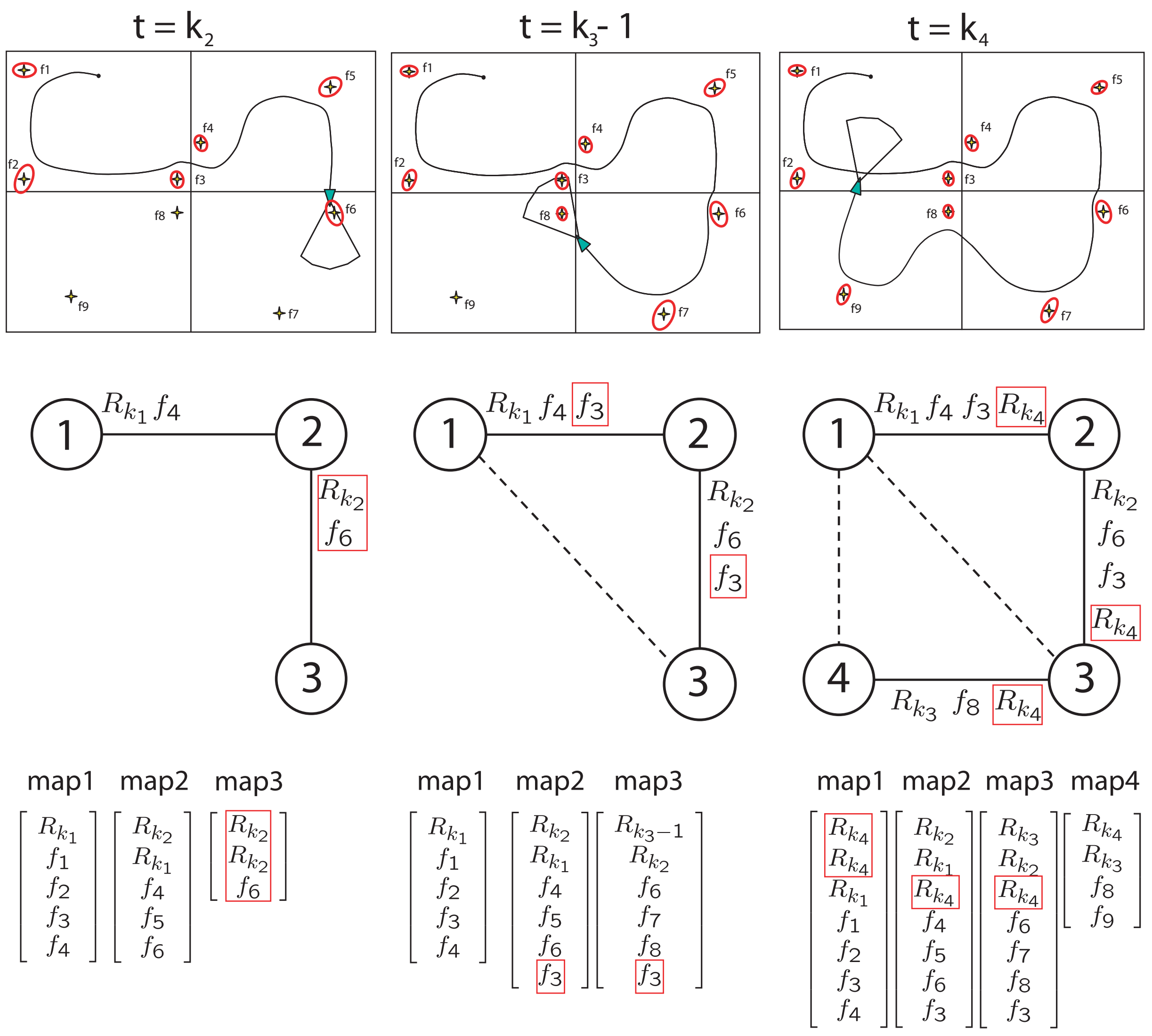


Figure 52: Example using CI-Graph SLAM. The figure is divided in three rows that show information about the state of a simulated experiment at three different instants of time (columns). In the first row, the map of the simulated environment with the current robot position is shown. In the second row, the graph of relations between submaps will be created according to the state of the estimation. In the last row we will show the state vectors of the estimated submaps at different moments of time.

Two operational levels can be distinguished in the algorithm. Local operations that are only applied to the current submap $\mathbf{m}_i$, and graph operations that are performed through the graph involving at least two submaps. Most of the time, the operations carried out when the robot moves inside a CI-submap are local operations corresponding to standard EKF-SLAM equations. Graph operations are more sporadic and can be considered as the interface between CI-submaps. In the following subsections, the graph operations are explained in detail as presented in Algorithm 2.

5.1.1 Starting a new submap

Suppose that robot is in submap $\mathbf{m}_i$ and we decide to start a new submap $\mathbf{m}_j$. The steps followed in the algorithm are:

- Add $\mathbf{m}_j$ to $\mathcal{N}$
- Add edge $\langle \mathbf{m}_i, \mathbf{m}_j \rangle$ to $\mathcal{E}_\mathcal{T}$
- Copy robot pose and last seen features from $\mathbf{m}_i$ to $\mathbf{m}_j$

In fact, the robot pose is copied twice in submap $\mathbf{m}_j$. The first copy will represent the current robot position which changes as the robot moves through the new map. The second copy will represent the initial position of the robot when it entered the map. This initial pose remains fixed as a common element with map $\mathbf{m}_i$.

An example can be seen in figure 52. At time k_2, submaps $\mathbf{m}_1$ and $\mathbf{m}_2$ have been already explored and a new submap is being created $\mathbf{m}_3$. Nodes $\mathbf{m}_1$ and $\mathbf{m}_2$ share in common a robot position R_{k_1} and a feature f_4. Submap 3 is initialized with robot R_{k_2} and feature f_6 from submap 2.

5.1.2 Re-observing a feature from a different map

This situation occurs when the robot is at submap $\mathbf{m}_i$ and observes *for the first time* a feature that is already included in a previous submap $\mathbf{m}_j$. The process followed is:

- Copy the feature from $\mathbf{m}_j$ to $\mathbf{m}_i$ along all nodes of the path in $\mathcal{T}$
- Add $\langle \mathbf{m}_j, \mathbf{m}_i \rangle$ to $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$

If $\langle \mathbf{m}_k, \mathbf{m}_l \rangle \in \mathcal{T}$ represents an edge in the path, to copy the feature from $\mathbf{m}_k$ to $\mathbf{m}_l$, the feature is first updated with the information contained in $\mathbf{m}_l$ using *back-propagation* equations and the correlations with the elements of $\mathbf{m}_l$ are also calculated [44].

Figure 52 at time $k_3 - 1$ shows an example of this case. Feature f_3 that belongs to submap $\mathbf{m}_1$ is measured by the robot when it is traversing submap $\mathbf{m}_3$. Since edge $\langle \mathbf{m}_1, \mathbf{m}_3 \rangle \notin \mathcal{T}$, f_3 is transmitted along the path $\langle \mathbf{m}_1, \mathbf{m}_2 \rangle$, $\langle \mathbf{m}_2, \mathbf{m}_3 \rangle$ that connects both nodes. Observe that the feature is replicated in all intermediate nodes. Finally, edge $\langle \mathbf{m}_1, \mathbf{m}_3 \rangle$ is included in $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$.

Algorithm 2 : `CI-Graph SLAM`

```
$\mathbf{z}_0, \mathbf{R}_0 = getObservations$
$\mathbf{m}_0 = initMap(\mathbf{z}_0, \mathbf{R}_0)$
$[\mathcal{G}, \mathcal{T}] = initGraph(\mathbf{m}_0)$ $\{\mathcal{G}(\mathcal{N} = \mathbf{m}_0, \mathcal{E}_\mathcal{G} = \emptyset)\}$
$i = 0$ $\{i$ for current submap$\}$
for $k = 1$ to steps do
  $\mathbf{u}_{k-1}, \mathbf{Q}_{k-1} = getOdometry$
  $\mathbf{m}_i = ekfPrediction(\mathbf{m}_i, \mathbf{u}_{k-1}, \mathbf{Q}_{k-1})$
  $\mathbf{z}_k, \mathbf{R}_k = getObservations$
  $\mathcal{DA}_k = dataAssociation(\mathbf{m}_i, \mathbf{z}_k, \mathbf{R}_k)$
  if revisiting $\mathbf{m}_j$ then
    {Subsection 5.1.3}
    for $\langle \mathbf{m}_k, \mathbf{m}_l \rangle$ in $path(\mathbf{m}_i, \mathbf{m}_j)$ do
      $backPropagation(\mathbf{m}_k, \mathbf{m}_l)$
      $copyRobot(\mathbf{m}_k, \mathbf{m}_l)$
    end for
    $addEdge(\langle \mathbf{m}_i, \mathbf{m}_j \rangle, \mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T})$
    $i = j$ {Map change}
  else if newMap $\mathbf{m}_j$ then
    {Subsection 5.1.1}
    $addNode(\mathbf{m}_j, \mathcal{N})$
    $addEdge(\langle \mathbf{m}_i, \mathbf{m}_j \rangle, \mathcal{E}_\mathcal{T})$
    $copyRobot(\mathbf{m}_i, \mathbf{m}_j)$
    $copyActiveFeat(\mathbf{m}_i, \mathbf{m}_j)$
    $i = j$ {Map change}
  end if
  if reobserved $\mathbf{f} \notin \mathbf{m}_i$ & $\mathbf{f} \in \mathbf{m}_j$ then
    {Subsection 5.1.2}
    for $\langle \mathbf{m}_k, \mathbf{m}_l \rangle$ in path$(\mathbf{m}_j, \mathbf{m}_i)$ do
      $copyFeat(\mathbf{f}, \mathbf{m}_k, \mathbf{m}_l)$
    end for
    $addEdge(\langle \mathbf{m}_j, \mathbf{m}_i \rangle, \mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T})$
  end if
  $\mathbf{m}_i = ekfUpdate(\mathbf{m}_i, \mathbf{z}_k, \mathbf{R}_k, \mathcal{DA}_k)$
  $\mathbf{m}_i = addNewFeatures(\mathbf{m}_i, \mathbf{z}_k, \mathbf{R}_k, \mathcal{DA}_k)$
end for
{Subsection 5.1.4}
$updateAllMaps(\mathbf{m}_i, \mathcal{T})$ {Updates $\mathcal{T}$ starting from $\mathbf{m}_i$}
```

5.1.3 Revisiting a previous submap

When the algorithm detects that the robot revisits an already traversed area $\mathbf{m}_j$, the transition from the current submap $\mathbf{m}_i$ to $\mathbf{m}_j$ is as follows:

- Update all nodes in the path from $\mathbf{m}_i$ to $\mathbf{m}_j$
- Copy the current robot pose along all nodes of the path
- Add $\langle \mathbf{m}_i, \mathbf{m}_j \rangle$ to $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$

As in the previous subsection, to update submaps in the path we use the *back-propagation* equations and to copy the current robot pose, correlations with submaps elements are calculated as well.

Figure 52 at time k_4 shows an example of this operation. When the robot makes a transition between submaps $\mathbf{m}_4$ and $\mathbf{m}_1$, current robot position R_{k_4} is replicated along all nodes that are in the path, i.e., along $\mathbf{m}_3$, $\mathbf{m}_2$ and $\mathbf{m}_1$. Finally, edge $\langle \mathbf{m}_4, \mathbf{m}_1 \rangle$ is added to $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$ and submap $\mathbf{m}_1$ becomes the current map.

5.1.4 Updating all maps from the current submap

Using the Graph operations just described, we can assure that the current submap is always updated with all available information. In addition, the CI property between submaps is preserved. An interesting property of the back-propagation equations is that they can be applied at any moment. They work correctly even if we back-propagate twice the same information. This allows us to schedule the back-propagation in moments with low CPU loads, or when graph operations are required. If the whole map has to be updated, the *back-propagation* equations are recursively applied starting from the current node and following the spanning tree $\mathcal{T}$.

5.2 Working with several cameras

In order to allow the system to easily scale with the number of cameras, each camera is treated independently in the algorithm. The only interaction between cameras is when a new feature is initialized. The feature is first initialized in one of the cameras. Since no depth information is available, this feature is included in the state vector using inverse depth parametrization [11]. Using the known relative transformation between cameras, the recently introduced feature is predicted and searched in the images of the other cameras and correspondingly updated when found. The rigid transformation between the cameras allows us to obtain the depth information of nearby features. For the rest of the steps of the SLAM algorithm, each camera predicts and updates features in the map independently. In addition, to improve the computational cost of the algorithm, inverse depth features are transformed to 3D cartesian parametrization according to the parallax index explained in [10].

5.3 Appearance-based loop closing

In order to close loops in the trajectory, a visual procedure is used. This method consists of three stages: first, one image per second is acquired from the stereo camera and converted into an appearance-based representation; then, it is checked if the current scene was seen

before, so that a loop is detected, and finally, the loop is closed by obtaining a transformation between the robot's poses by solving a perspective-n-point problem.

5.3.1 Appearance-based representation

An appearance-based representation of an image is obtained by using a visual bag of words [45]. This is a technique that represents an image by using a numeric vector created from the local features this contains. We use *SURF* points [3] as image features. A *SURF* feature is a point in the image associated to a real 64-dimensional descriptor which summarizes the distribution of the intensity content within the point neighbourhood.

The bag of words technique consists of clustering the image descriptor space (the 64-dimensional *SURF* space, in our case) into a fixed number C of clusters. The centers of the resulting clusters are named *visual words*; after clustering, a *visual vocabulary* is obtained. Now, a set of image features can be represented in the visual vocabulary by means of a vector v of length C. For that, each feature is associated to its closest visual word; then, each component v_i is set to a value in accordance with the relevance of the i-th word in the vocabulary and the given set, or 0 if that word is not associated to any of the image descriptors. In general, the more a word appears in the data used to create the visual vocabulary, the lower its relevance is. The vector v is the *bag of words* representation of the given set of image descriptors. This way, the appearance of an image can be simply described by a numeric vector.

This method is very suitable for managing big amounts of images; moreover, [40] presents a hierarchical version which improves efficiency. In this version, the descriptor space clustering is done hierarchically, obtaining a visual vocabulary arranged in a tree structure, with a branching factor k and L depth levels. This way, the comparisons for converting an image descriptor into a visual word only need to be done in a branch and not in the whole discretized space, shrinking the search complexity logarithmically.

We use a hierarchical vocabulary with $k = 9$, $L = 6$ and the *kmeans++* algorithm [1] as clustering function. This vocabulary was created from a set of 1300 images obtained from the *Mixed / Bovisa_2008-09-01_Static* dataset. These images represent indoor and outdoor scenes, so that the vocabulary is generic enough to be used with any other dataset.

5.3.2 Loop detection

The loop detection procedure runs independently of the rest of the system, at a frequency of 1Hz. In order to detect a loop, one stereo pair is acquired at time t. The image from the left camera is converted into its vector representation, named v_t. This vector is compared with the set of all the vectors of the images obtained before, W, to check if any of them is similar enough to consider both scenes the same. If there is a satisfactory match with some vector $w_{t'} \in W$ acquired at time $t' \leq t - c$, a loop may be found. To confirm it, there must be consistency with the images previously matched. The constant c is the time interval that must pass to consider an already seen scene as revisited; it is set to 20 seconds. Finally, the current vector v_t is added to the list of already visited scene vectors W.

To make vector similarity comparisons faster, an inverted file is maintained. This file keeps a record of in which vectors each visual word is present. This way, when the vector v_t is going to be compared with vectors from W, comparisons are only made with those vectors $w_{t'} \in W$ which have at least one visual word in common with v_t. When the vector v_t is added to W, the inverted file is updated by including v_t in the lists of the visual words it contains.

The resemblance between two vectors is scored when they are compared. This score grows as the similarity between the two images is higher. Given two image vectors v_t and $w_{t'} \in W$, the score of its match is related to the normalized distance between the two vectors [40]:

$$s(v_t, w_{t'}) = 1 - \frac{\left|\left| \frac{v_t}{|| v_t ||} - \frac{w_{t'}}{|| w_{t'} ||} \right|\right|}{2} \tag{30}$$

We use the L_1-norm to compute this score, so that it is defined between 0 (completely different words) and 1 (perfect match). Vectors from matches $< v_t,\ w_{t'} >$ whose score is above a threshold $\lambda = 0.037$ are likely to come from images that represent the same scene, so they are considered loop candidates. The rest of the matches are discarded.

We impose a temporal constraint to reliably detect loops and to avoid mismatches. A loop in a place visited at time t and t'_0 is detected if there is a match $< v_t,\ w_{t'_0} >$, as well as previous matches $< v_{t-1},\ w_{t'_1} >$, ..., $< v_{t-N+1},\ w_{t'_{N-1}} >$ such that $\max(|t'_0 - t'_1|, \ldots, |t'_{N-2} - t'_{N-1}|) \leq 2$ seconds. The constant N is the minimum time duration of the loop trajectory to be found, and is set to 3 seconds.

Figure 53 shows the images from the left hand side of the stereo camera matched by the loop detection algorithm on the *Indoor / Bicocca_2009-02-25b* dataset. Each group of close dots represents each one of the five loops detected in the whole trajectory. Figure 54 shows two of the images matched in the first loop. There is a sixth loop in this dataset, marked with a cross in the graph, which cannot be detected. This is due to a limitation of this appearance-based approach: it cannot handle places seen from very different points of view. Figure 55 illustrates this difficulty with the non-matched images from the missed loop.

5.3.3 Loop closing

Once a loop is detected at time t and we know that the current place was previously visited at time t', the loop is closed by finding a transformation between the current robot's pose and the one at time t'.

For that, the two images from the current stereo pair, I_t^1 and I_t^2, and one of the images from the stereo pair acquired at t', $I_{t'}$, are searched for *SURF* features. Those features which are not present in the three images are removed. The rest of the features are be reconstructed in the 3D space by using stereo triangulation and their pixel coordinates in I_t^1 and I_t^2. This way, we obtain the set of points in the space in the current camera reference, C_t. Given those points and their projections in $I_{t'}$, we can find the transformation ${}^{C_t}T_{C_{t'}}$ by solving the perspective-n-point (PnP) problem [37]. This problem consists of estimating the pose of a calibrated camera from n 3D-to-2D point correspondences. After obtaining ${}^{C_t}T_{C_{t'}}$, the constant transformation between the robot and the stereo camera can be applied to find the final transformation between robot's poses at t and t', and successfully close the loop.

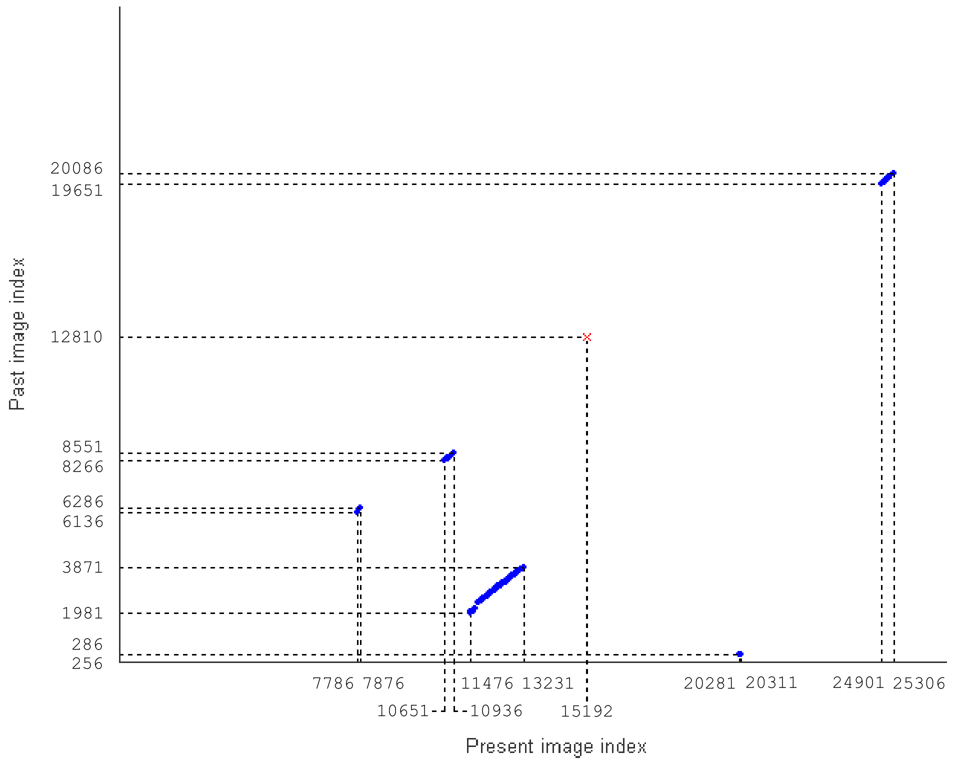


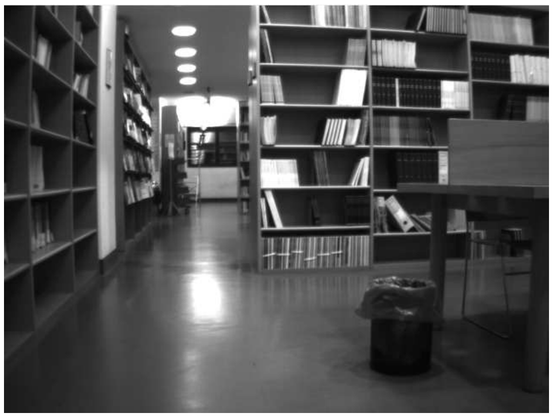
Figure 53: Indices of images matched by the loop detection algorithm (dots), and undetected loops (crosses).

5.4 Stereo SLAM results

In order to obtain a benchmark solution we ran our Stereo CI-Graph SLAM system on dataset Biccoca_2009-02-25b. The dataset consists of 26335 trinocular image frames collected during 30 minutes at 15 FPS. Figure 56 shows an example of the system performance when building a local map along the library. We can see how features in the map are predicted and search over right, left and top images in order to update the state vector. A reconstruction is also shown both in top and lateral view for the resulting submap.

Figure 57 shows the results obtained after running our loop closing approach. Pairings between past and current images are highlighted with red points and green crosses respectively on the odometry path. In order to overcome the lack of texture in critical parts of environment, the odometry readings were used along with the CI-Graph method over the full path. A total map of the dataset is shown in figure 58 where each local submap is represented in absolute coordinates before applying the loop closing constraints. The estimated trajectory is also presented in Figure 59.

We compared our estimation with the provided Ground Truth solution when this was available (see Fig. 61 and Fig. 60). Figure 62 shows the Absolute Trajectory Error (ATE) evaluation where our Stereo SLAM produces an error of $1.38119m$ in position and $0.04012rad$ in orientation with a maximum error of $1.92394m$ and $0.14556rad$ correspondingly. The error


(a) Input image (7786)

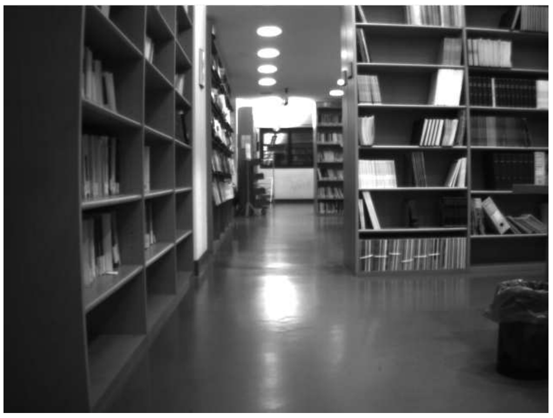
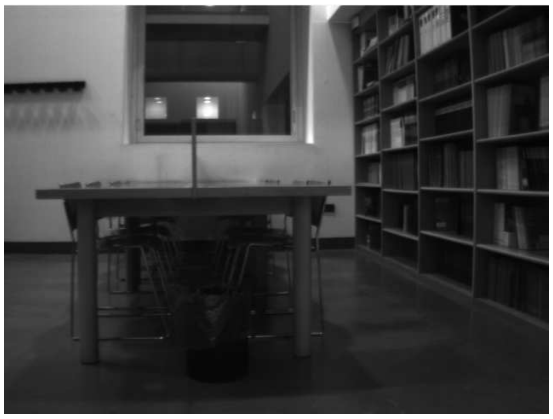

(b) Matched image (6136)

Figure 54: Example of a successful loop detection


(a) Input image (15192)

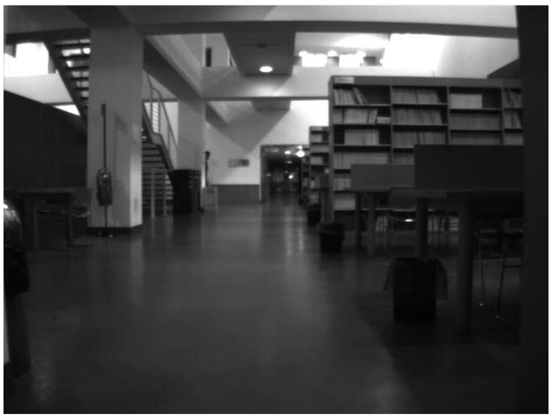

(b) Expected match (12810)

Figure 55: The appearance-based approach cannot detect a loop with these images, acquired in very close positions but from different points of view.

distribution is shown in Figure 64 considering a 3σ error bound for the uncertainty. In addition the Relative Pose Error (RPE) was computed producing a Mean Square Error of $3.5468m^2$ in translation and $0.6002rad^2$ in orientation (see Fig. 63).

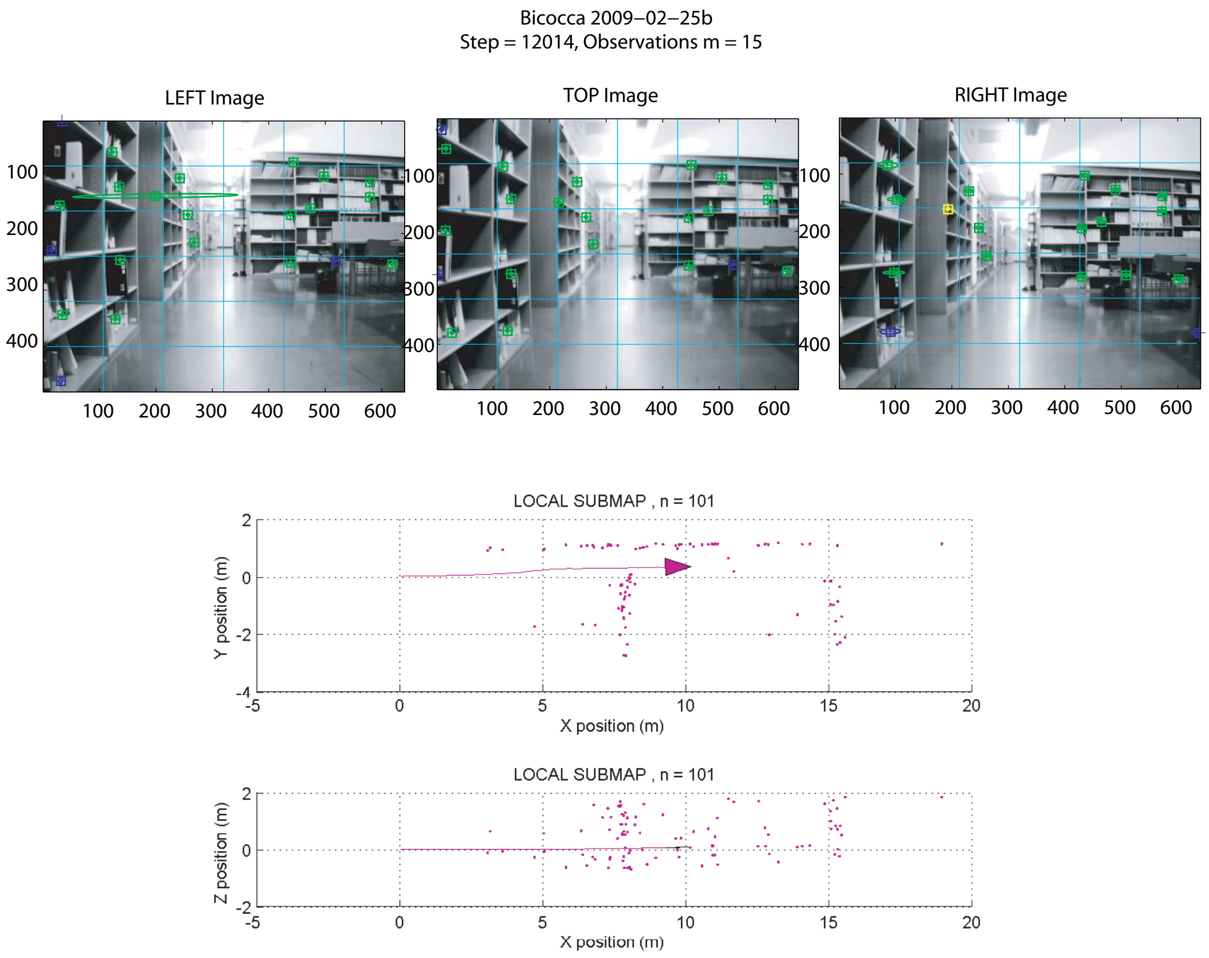


Figure 56: SLAM system performing trinocular tracking (top). Top and lateral views of the local submap reconstruction (bottom).

6 Benchmark Solutions: Trinocular SLAM

We present hereafter a benchmark solution (BS) for the benchmark problem "Stereo or trinocular SLAM - Bicocca_2009-02-25b"; this solution has been generated in the framework of the Rawseeds project. This solution is based on detecting segments as the image features of interest. The BS could then be called "Stereo (Trinocular) SLAM with Segments". The BS is based on the robot trinocular streams, as well as the odometric information. By processing these streams we incrementally build a 3D map of the observed environment and estimate robot poses with respect to this map, detect loops, and perform loop closure, and finally calculate the performance measurements.

This document is articulated as follows: first we shortly review the input data, we then introduce the incremental map building algorithm and analyze the resulting 3D map; we then introduce the loop closure algorithm, and analyze the corrected 3D map. We finally present the performance measurements.

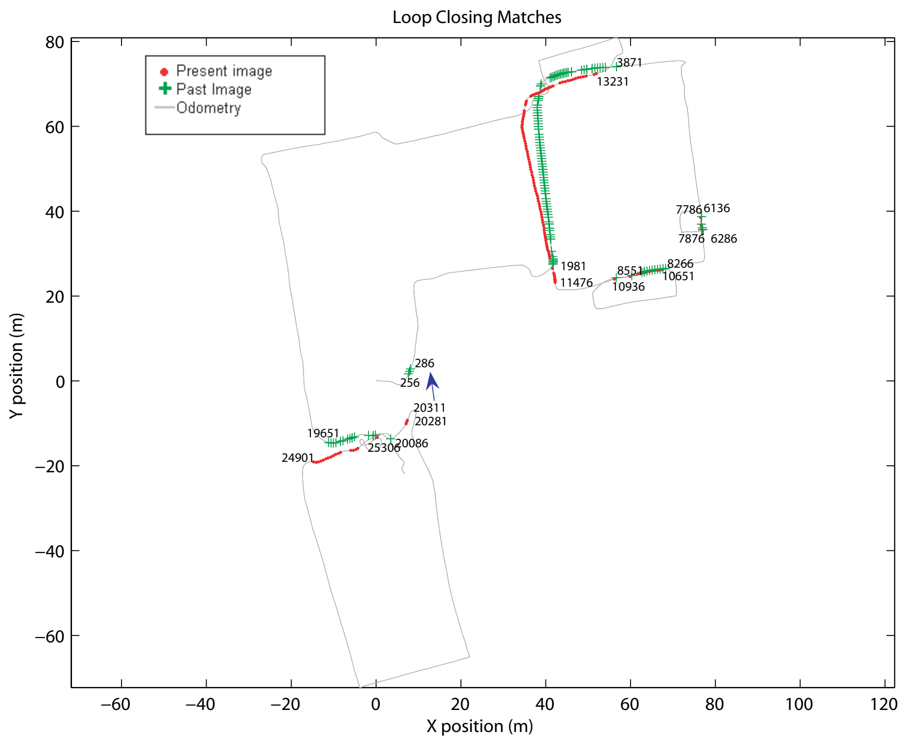


Figure 57: Obtained loop closing matches. Results are shown along the odometry path

6.1 Input data

Input data of our system are the odometric readings of the robot, and the image sequences generated by the trinocular stereo rig mounted on the front side of the robot. In Figure 65 an example of image triplets; in Figure 66 an example of odometric data.

6.2 The incremental map building algorithm

The approach used in map building is the well known Hierarchical SLAM, adapted to the case of 3D segments from trinocular stereo. For a paper-like description of the algorithm see [5]. We hereafter present the software structure and the relevant parameters of the algorithm.

6.2.1 Main loop

The main loop iterates over image triplets. We first load the images and the odometry, attention has to be devoted to the selection of the appropriate image, as the data have not been captured at exactly the same time (microseconds); we select, for each image odometry reading nearest to the image time-stamp. The captured images are then rectified, exploiting the camera calibration parameters that have been loaded during the initial configuration. Next we retrieve the last robot movement, from the odometry readings, and here we have both the nominal value and its covariance. With this information we can now begin the EKF(iltering). The first important step of EKF is the prediction.

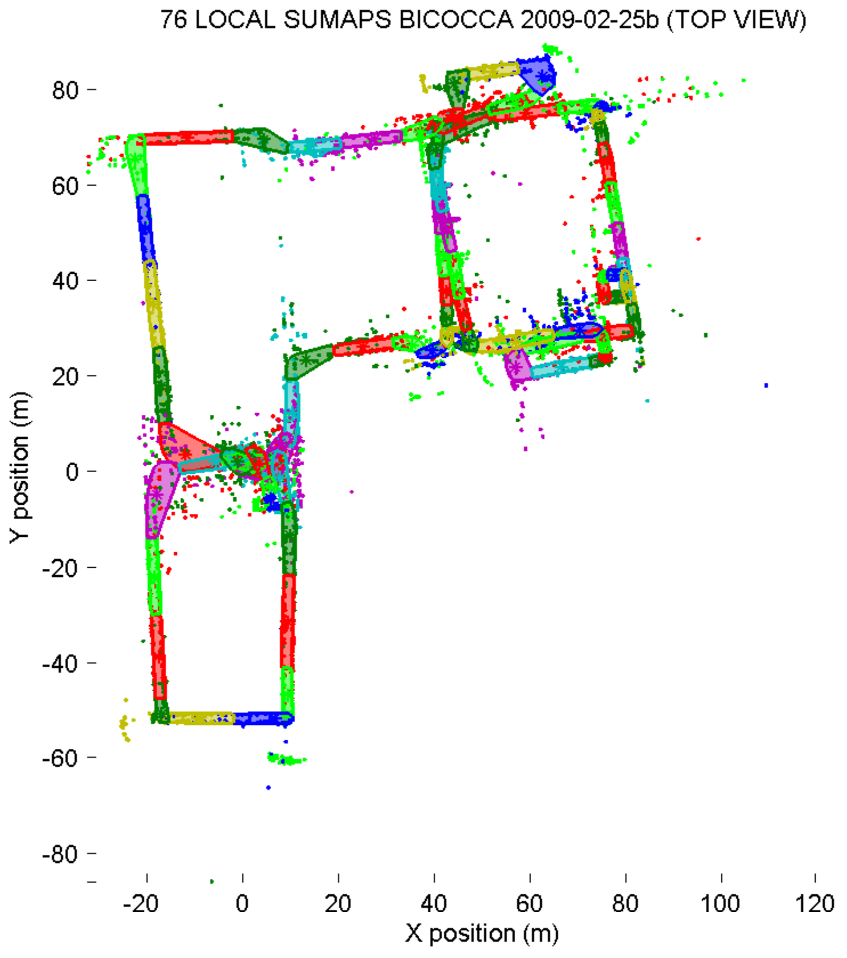


Figure 58: Map result with Trinocular sequence and odometry on Bicocca_2009-02-25b.

6.2.2 Prediction step

We perform a prediction over the whole state, starting from the state at the previous time step and predicting the new robot pose and feature position on the basis of the odometric information. This operation results in a new state, which consists of the robot pose, position of features, and whole state covariance matrix. The operations are the following:

1. The previous time step gives the robot pose, position of features and covariance matrix;
2. The state (robot pose and map features) and its covariance matrix are updated basing on the odometric information;
3. The current state represents the prediction about where the robot and the features might be found in the 3D space;
4. We use this information to predict where the features could be found in the images, by using the camera projection matrix.

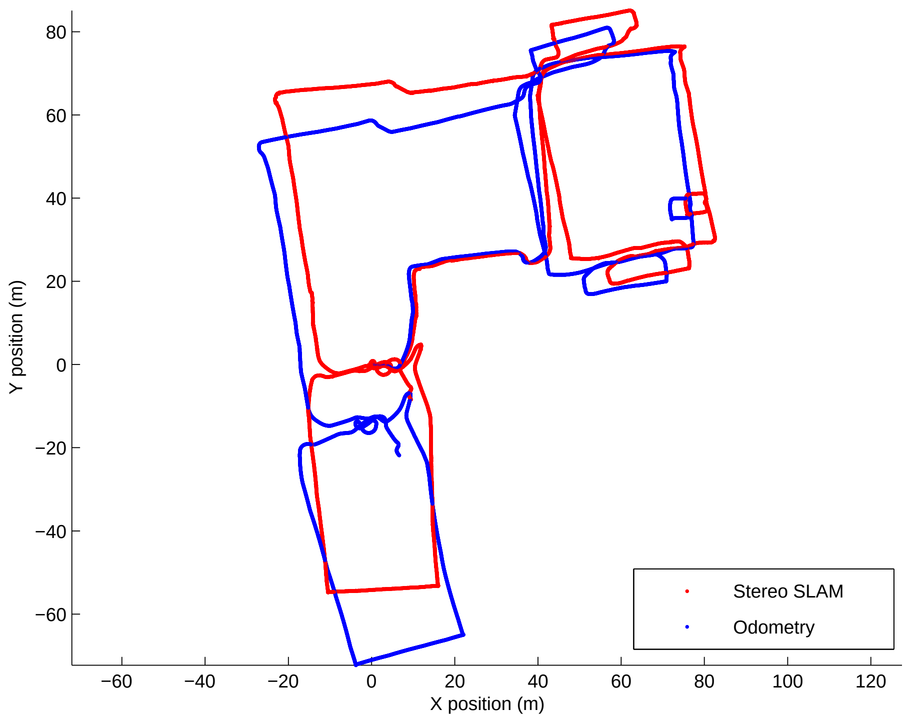


Figure 59: Estimated trajectory with Stereo SLAM

Now we have the information required for associating the features observed up to now with the current measurements.

6.2.3 Feature extraction

Each image of the trinocular stereo rig has to be elaborated in order to extract features; this processing can be executed independently on each image. We determine image segments, likely images of 3D segments in the observed scene. The feature extraction algorithm is based on the work of [26] for matlab. On each image we perform the following computations.

1. Edge detection with the so-called Canny detector (single scale), with the following parameters:
    (a) Sigma of the Gaussian low-pass filter = 1
    (b) Low threshold = 0.1;
    (c) High threshold = 0.2;
2. Linking of the edge points. This operation results in a set of pixel chains, each represented by the coordinates of all the pixels on that edge. During this processing we also erase chains of edge pixels shorter than 50 pixels
3. Polygonal Approximation. This step is performed on each pixel chain. Line segments are fitted with a maximum deviation of 2 pixels.

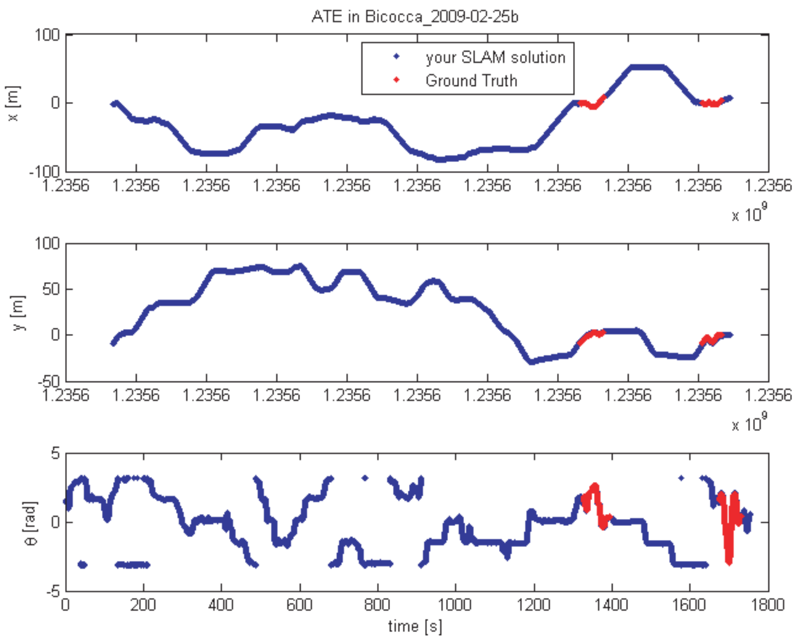


Figure 60: Comparison between Ground Truth Position and Stereo SLAM before closing the loop. The comparison has been done when Ground Truth is available.

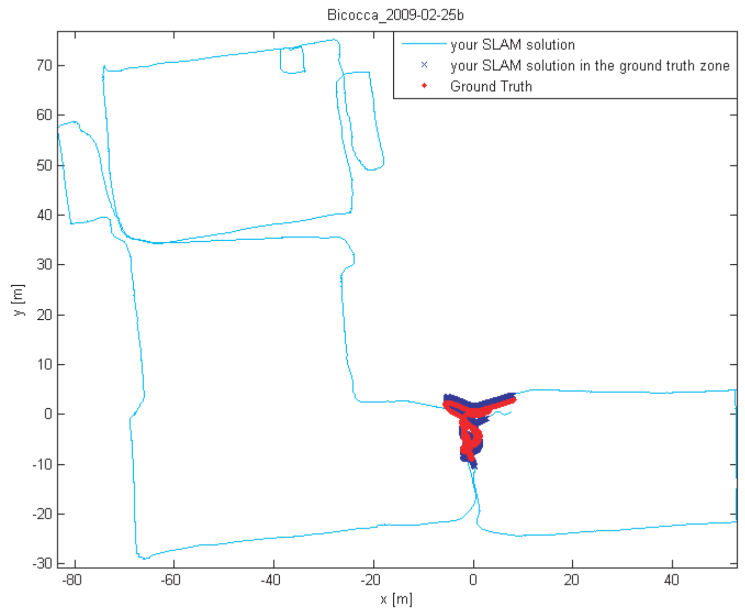


Figure 61: Full estimated trajectory and Ground Truth.

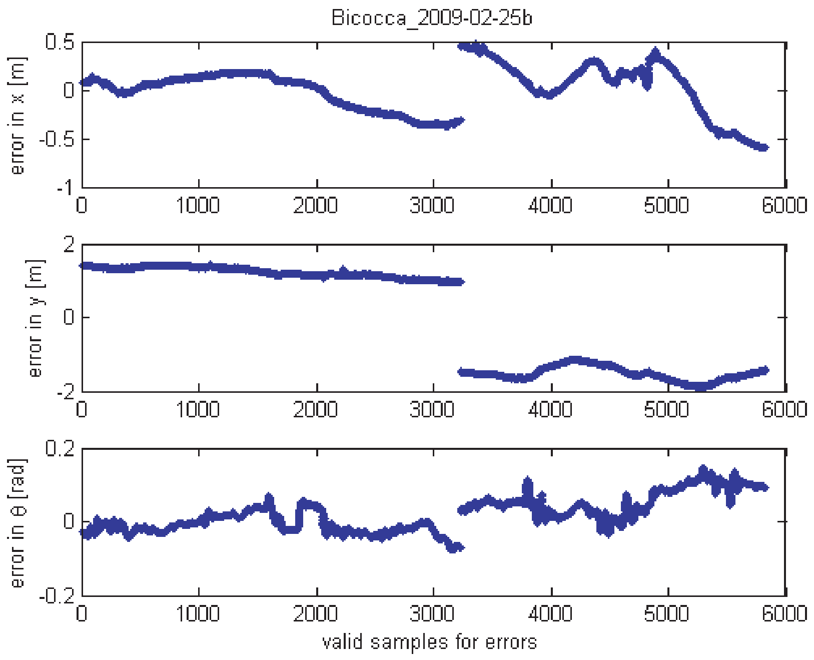


Figure 62: Position and Angular Root Square Error in the available samples.

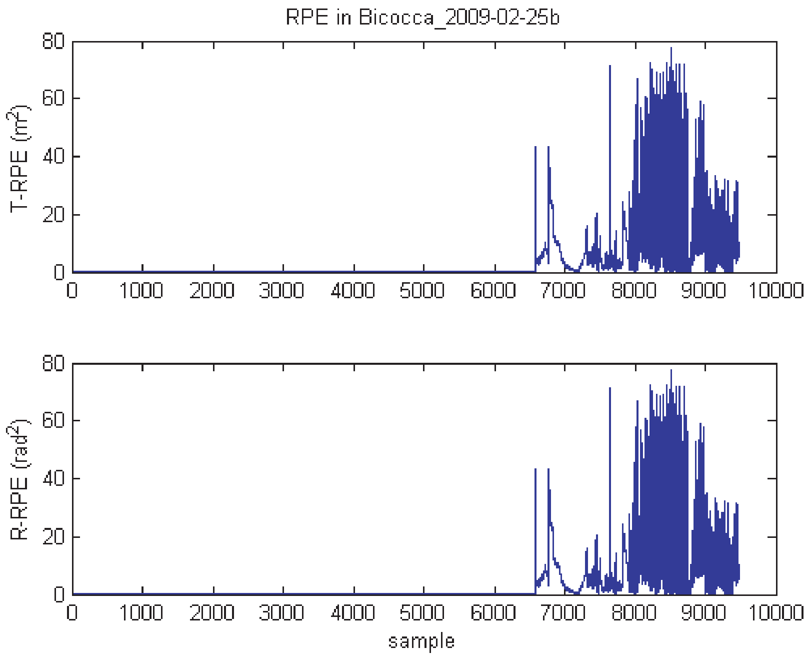


Figure 63: Position and Angular Root Square Error in the available samples.

4. Computation of the average gradient of each segment. This is performed by dividing each segment in a fixed number of sub-segments and calculating the gradient across the

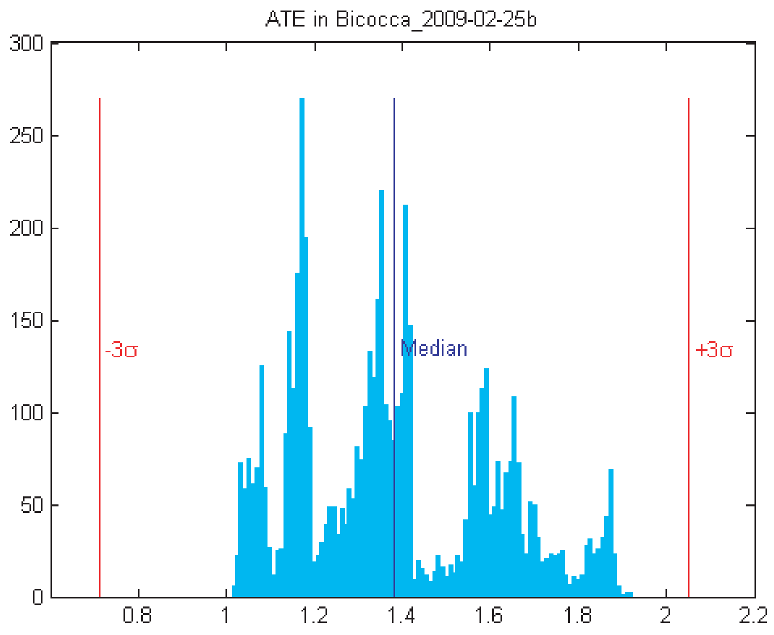


Figure 64: Error Distribution.

segment in these points; we currently use 3 sub-segments, i.e., 4 points. The result is then averaged.

5. Elimination of segments that are too close each other. This processing helps in preventing wrong associations of features lying on particular surfaces (e.g., on door frames).

After this processing we have, for each image, a set of 2D segments. The next step has to associate these features with the predicted ones. This is a deviation from classical trinocular stereo, which would first stereo-associate the segments in the three images, so to be able to 3D-reconstruct the associated ones. The approach we developed is more similar to apply a monocular slam algorithm to each camera stream, integrated with the trinocular stereo for the features not matched by their prediction, i.e., new ones.

6.2.4 Data association

The data association is performed on each image separately. For each predicted 2D segment (projection of a 3D map segment), we test all the observed features in the image, searching for the nearest one. The comparison is based on a 2D variation of the three criteria described, e.g., in [5]. In particular, we consider the third criterion first, i.e., we check if the (measured) image segment can be considered a continuation of the 3D one or a subpart of it, so to exclude the case of the image segment being the observation of another scene segment. We need not to associate an image segment to a 3D one, even if they are collinear, whenever there is no superimposition between the two, along the supporting line. If there is no such superimposition

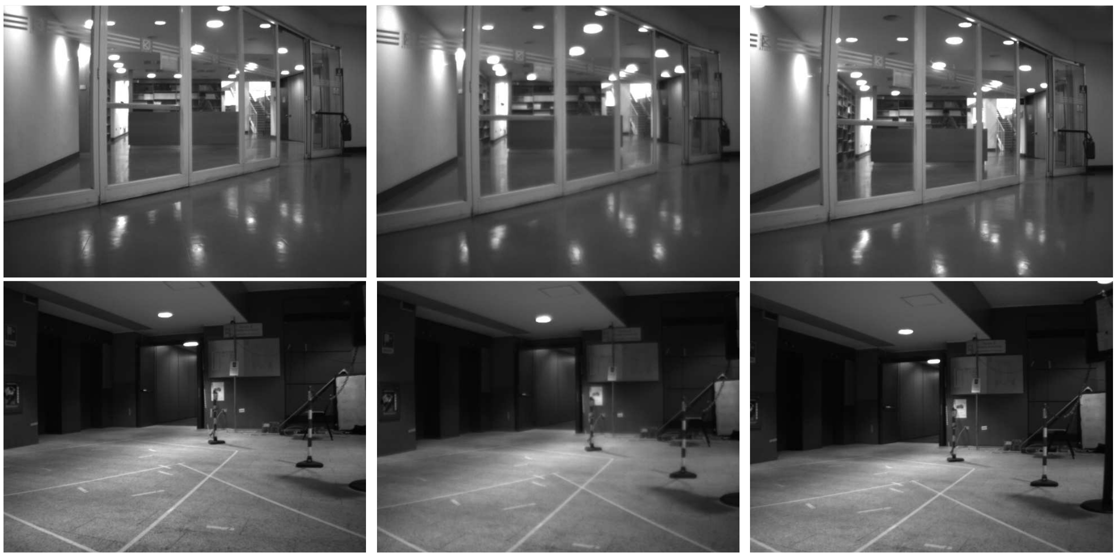
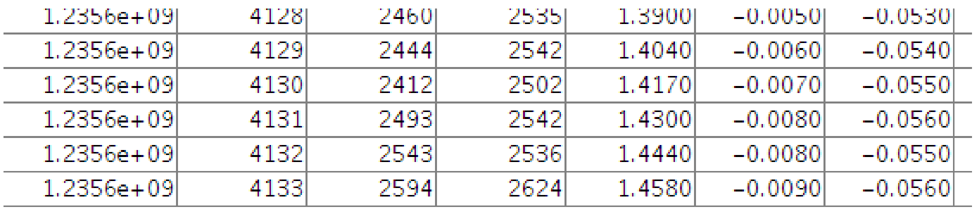


Figure 65: Example of triplets from the three trinocular streams, i.e., (left to right) the left, the top, and the right camera images.



1.2356e+09	4128	2460	2535	1.3900	-0.0050	-0.0530
1.2356e+09	4129	2444	2542	1.4040	-0.0060	-0.0540
1.2356e+09	4130	2412	2502	1.4170	-0.0070	-0.0550
1.2356e+09	4131	2493	2542	1.4300	-0.0080	-0.0560
1.2356e+09	4132	2543	2536	1.4440	-0.0080	-0.0550
1.2356e+09	4133	2594	2624	1.4580	-0.0090	-0.0560

Figure 66: Example of odometry readings; in the columns, respectively: time-stamp, rolling counter, tics right, tics left, x [m], y [m], ϑ [rad], the latter three items are relative to the starting pose [0, 0, 0]

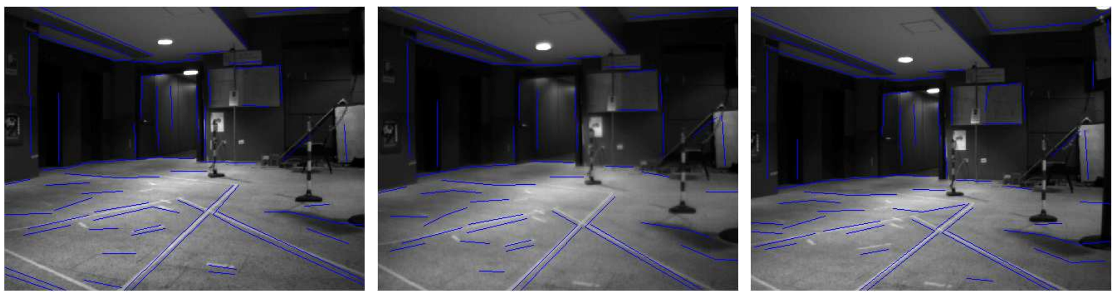


Figure 67: Example of segments detected on an image triplet.

the image segment is discarded. Secondly, we calculate the Mahalanobis distance between the two projected extrema of the 3D segment, with respect to the 2D measured segment. If this distance is beyond a threshold, the segment is discarded. For more details on this Mahalanobis distance see e.g., [32]. Finally we check if the gradient direction of the two segments is the same. Otherwise the image segment is discarded. As a result of these operations, the 3D map segment, by means of its predicted image appearance, is associated to the image segment (if any) that is matching it with the minimum Mahalanobis distance. If it cannot be associated to any image segment, then a counter is incremented, so that 3D segments which are not associated for a long time can be suppressed. Usually only a sub-set of image segments are associated to 3D segments. The unassociated segments will be marked as new observations.

6.2.5 Update step

We use a canonical EKF step for updating the state by using the predictions coming from the previous steps, and the associated measurements. By doing so we obtain an update of the whole state (features and robot pose), integrating the new observations.

Next, we need to add the new segments, which were not associated to any map segment. These segments were not taken into consideration during the previous update step. So we need to generate new 3D segments from these observations, and add them to the state. 3D segments are generated exploiting implicitly the trinocular approach, i.e., the trinocular epipolar constraint. The first step to find the associations of a segment in the left image is to project it into the right image. This process bases on the left 2D segment for the initialization of a 3D segment in inverse scaling coordinates. Then this 3D segment is projected into the right image by using the calibrated projection parameters. Afterwards we determine the set of segments, in the right image, which are sufficiently close (in Mahalanobis distance terms) to the projected segment. If this set is not empty, i.e., at least one right image segment could be associated to the segment in the left image, then we proceed repeating this search in the top image. By applying the same search to the top image we obtain, for each segment of the left camera, the set of potential correspondents in the top. Concluding we have two sets of potential correspondents, one for the right camera (namely set-LR) and one for the top camera (namely set-LT). Next, we build a set of potential matching segments by verification of the distance (again Mahalanobis) between the 3D segment obtained from the top camera projected in the right and the segments in the right. This is then repeated starting from the right camera, and deleting the pairs involving the same segments found before (top into right). Notice that the pairs are actually triples of potential matches, by considering the original segment in the left image. Finally, we select among the potential triples, by choosing the one that minimizes the sum of the previous Mahalanobis distances. Once we have obtained the associations, the next step is to generate the 4D (inverse scaling) world segment from each triplet. To this aim we setup a local filter, whose initial state is composed by 4D segments generated by projecting the associated segments of the left image into the world. The innovation is determined with respect to the re-projection of these segments in the right and top images and computing the differences to the observed ones. This filtering allows the generation of a set of new 4D map segments, generated by the trinocular vision system. Then we add this set to the global state, where all features are parameterized with the 4D inverse scaling. The representation is then moved back to conventional euclidean 3D only when required, e.g., for display.

3D segments, which are in front of the camera, and which have not been associated to any image segment for a long time, are deleted.

6.2.6 Sub-map saving

The sub-map currently being elaborated is closed and saved when a given number of image frames have been processed or when a maximum number of segments in the sub-map is reached. The latter is the approach usually taken in the literature, while we used the first for the parallelization of the computation; in such case subsequent data-chunks can be actually processed in parallel,each resulting in a different sub-map. This step concludes the sub-map generation. As typical of hierarchical SLAM, each sub-map is only a part of the global map. The global map is represented as an oriented graph, where each node includes the base reference of the sub-map and the edges represent the roto-translations between two adjacent sub-maps. The roto-translation is computed on the final poses of the robot at the closure of that sub-map. Concluding, the global map is, at this stage, a chain of sub-maps.

6.2.7 Analysis of the resulting 3D map

As it can be easily seen in picture 68, there are many gross errors. If we analyze the locations of this errors and compare them to the odometry readings (image 69 below), we can notice that they primarily occur when the robot path is curve.

6.2.8 The loop closure algorithm

In the previous section we have seen that a global map, obtained as the concatenation of sub-maps, presents some gross errors. The causes of these errors can be found in the odometry readings (which are not reliable) and in the data-association errors, during the incremental map building, which are often, but not exclusively, due to the odometry errors. We therefore need a loop detection and closure algorithm, in order to try to overcome such errors. We sequentially load the sub-maps generated by the incremental map building algorithm. For each sub-map we check if its bounding box overlaps with the bounding box of another sub-maps previously analyzed. If this condition is satisfied, and the overlapping sub-map is not the previous sub-map, then we can consider that the robot might have previously visited this part of world, i.e., it has already observed this place.

Loading sub-maps and finding overlaps Each sub-map contains a large number of segments. This is because we prefer to generate more detailed sub-maps during the incremental map building, and eventually filter out useless segments during the loop closure. Therefore, when loading sub-maps, we perform a selection over the set of segments, discarding the segments that are not reliable enough. Hence segments whose extrema have a variance larger than 600mm and segments which are partially or completely misplaced, e.g., underground, are eliminated. We say that two sub-maps overlap if the intersection of their bounding box corresponds at least to the 40% of the size of the bounding box of the sub-map being evaluated.

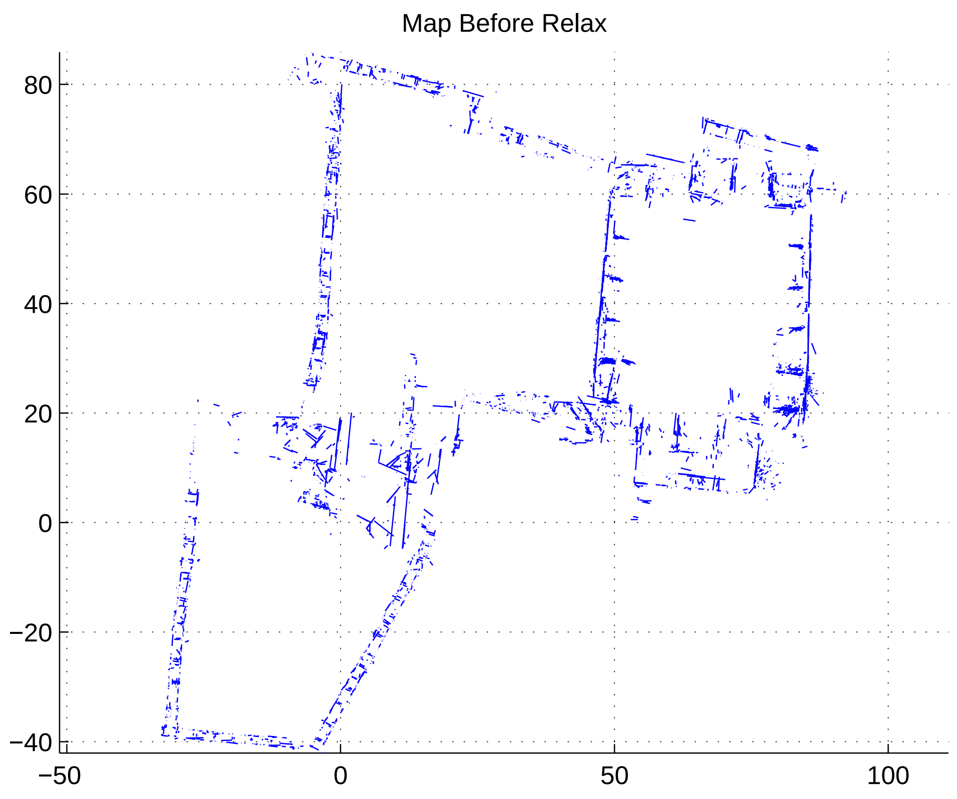


Figure 68: In the picture we show the sub-maps generated by the incremental map building algorithm, concatenated to form the global map; both axes in meters.

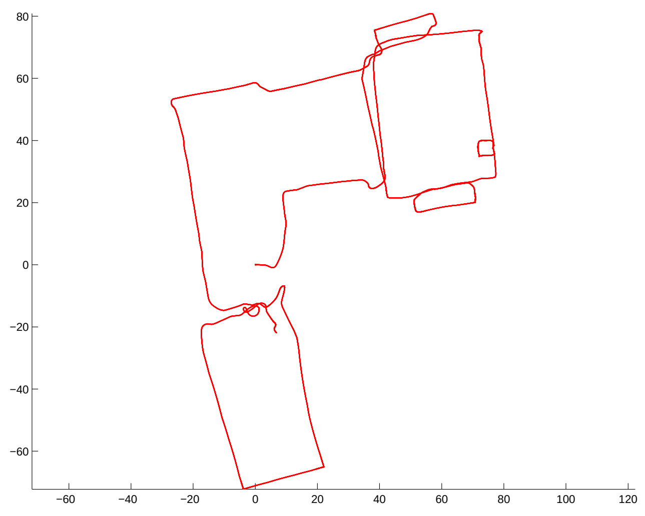


Figure 69: Odometric robot path; both axes in meters

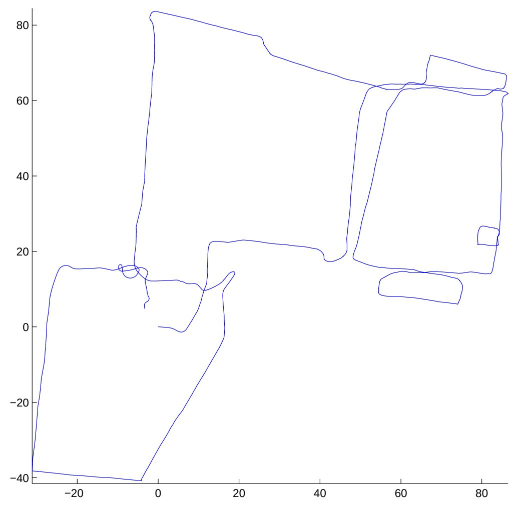


Figure 70: The robot path as resulting after the incremental map building, i.e., before loop closure; both axes in meters.

Closure If the current sub-map overlaps with one or more of the sub-maps already considered, then we start a more accurate loop detection algorithm, based on checking if some subset of the 3D features in a sub-map match some of those in the current sub-map. In other words, we search for correspondences among features of the overlapping sub-maps; this is done using the JCBB algorithm [38]. If the current sub-map (say E) overlaps with more than one of the previous sub-maps, we select the sub-map for which we could find more correspondences. Let us call C the sub-map being revisited. As stated in [17], we adopt local map joining [49] to join and fuse E and C, and to relocate the robot in the updated sub-map C. We then perform a global relaxation over the relative pose of each sub-map, with an Iterated Extended Kalman Filter, as proposed in [17] as an adaptation from the original form by Bar-Shalom et al. [2]. The final result is a global 3D map where the gross errors caused by unreliable odometry readings and data association errors have been mitigated by the loop closure constraint. Notice that further details about the quite complex loop detection and closure algorithm can be found in [39], as we adapted this proposal to the 3D segment from stereo problem.

Analysis of the corrected 3D map In our view the resulting map is quite poor; many errors, which we believe can be attributed to data association, are present. Notice in particular the area around $x = 80, y = 20$.

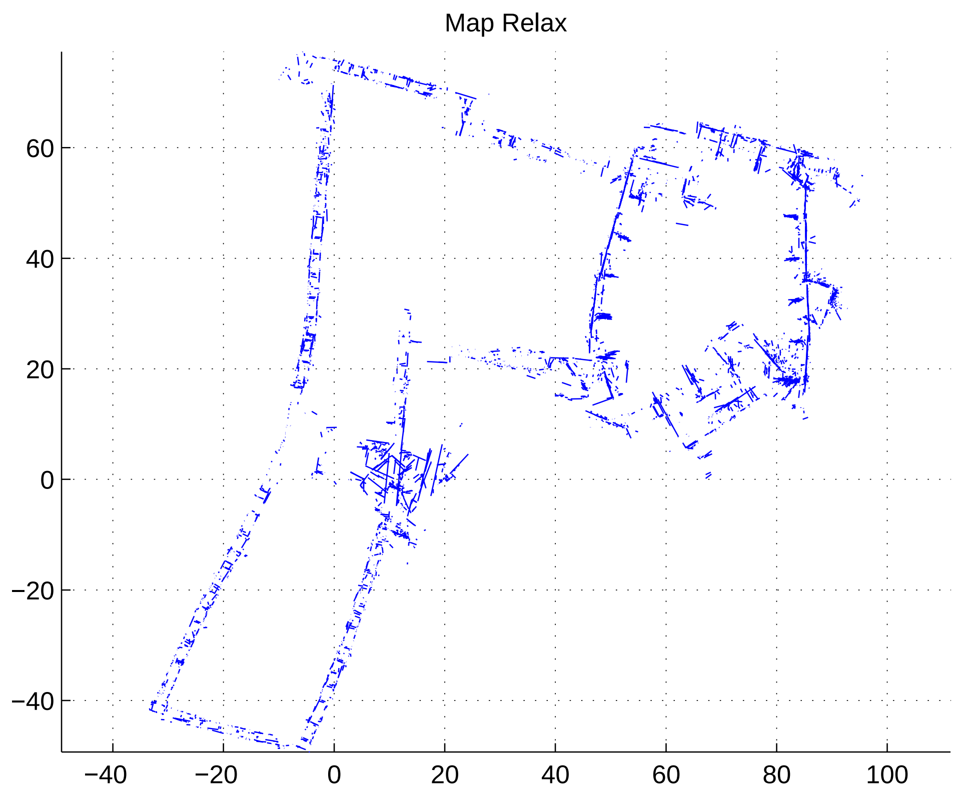


Figure 71: The map after loop closure; both axes in meters.

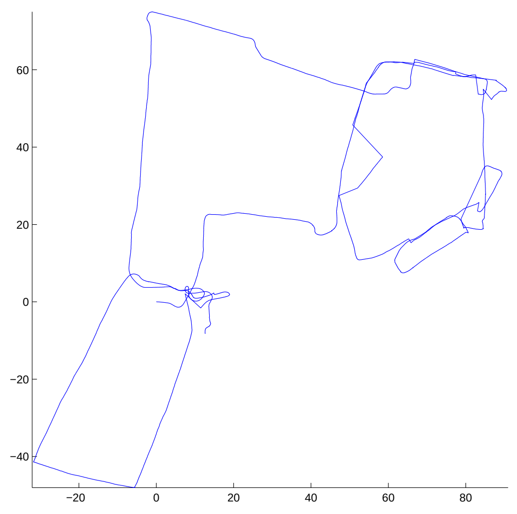


Figure 72: The robot path as resulting after the loop closure; both axes in meters.

6.3 Benchmark measurements

6.3.1 Absolute Trajectory Error (ATE)

The Absolute Trajectory Error (ATE) is a useful performance measure that captures at the same time both the accuracy in mapping and in localization. It is a compact, although indirect, representation of the accumulation of errors due to data associations, biases in the resulting map and robot pose estimates. It can be reduced by loop closures, which are informative events that happen during the execution of SLAM algorithms.

In order to compute this measure, the measures of the estimated robot pose has to be provided at the poseGT frequency (50Hz), i.e., a robot pose estimate has to be provided for each poseGT value available. the values have been computed for both the map obtained by the incremental map building, i.e., before the loop closure, and after the loop closure.

The computation of the value of the ATE requires to follow these steps:

1. For each instant, provide the robot pose estimate. Our trinocular stereo SLAM algorithm takes as input image triplets generated at 15Hz. Therefore our robot pose estimation is generated at the same frequency. In order to synchronize our data to the GT robot poses we need to interpolate missing information. We do this by performing a linear interpolation between subsequent estimated robot poses. We sample this interpolation at the robot poseGT frequency.
2. Put all reconstructed robot pose in a file, to be provided as part of the BS; the file is a list of lines, one for each pose; for each pose the format is $< timestamp, [x_j, y_j \theta_j] >$.
3. For each instant where the poseGT is available, compute the distance, in terms of translation, between the poseGT and the reconstructed robot pose; $d_j = ||trans(x_j) - trans(x_j^{GT})||$, the orientation has been considered implicitly taken into account by the high sampling rate of the position.
4. Put all error distances in a file, to be provided as part of the BS; the file is a list of lines, one for each pose; for each pose the format is $< timestamp, d_j >$.
5. ATE =
   (a) mean of the translation error d_j;
   (b) standard deviation of the translation error d_j;
   (c) confidence interval of the translation error d_j;

Mean and standard deviation of the translation error are computed in the standard way. For the confidence interval we consider these values (mean and standard deviation) and use them to calculate the CI with the built-in Matlab function "paramci".

$$\begin{aligned}
ATE &= \left[\bar{d_j} \quad \sigma_{d_j} \quad conf.int.d1_{3\sigma} \quad conf.int.d2_{3\sigma}\right]^T; \\
ATE_{pre-loopclosure} &= [8.68309 \quad 9.35616 \quad 8.4429 \quad 8.9233]^T \\
ATE_{post-loopclosure} &= [1.81892 \quad 2.37795 \quad 1.7579 \quad 1.8800]^T
\end{aligned}$$

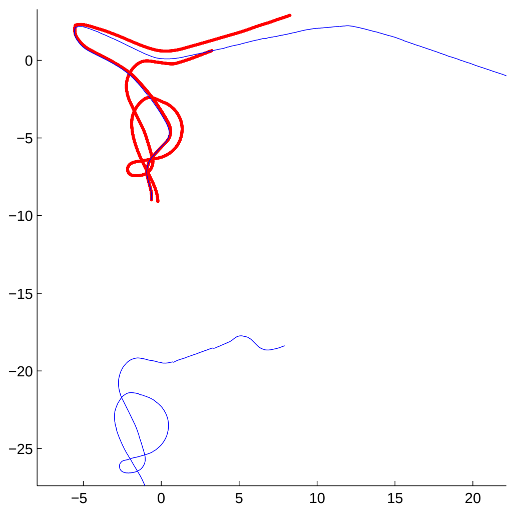


Figure 73: In this image we show the poses of the poseGT (in red) and the estimated robot poses (in blue); both axes in meters. Note that the estimated robot poses, in the second pass in the GT area, are translated of some meters. This is because the robot path used here is the one before loop closure.

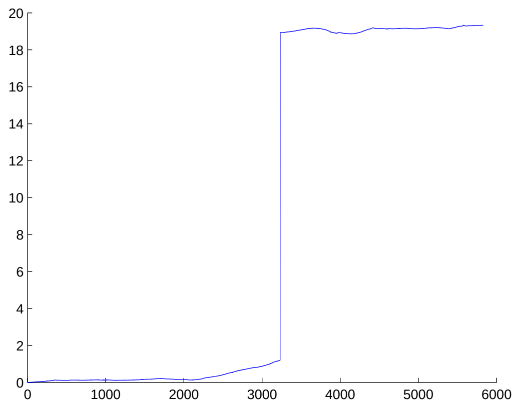


Figure 74: In this image we show the ATE error vector, i.e. the distance between the estimated robot pose and the poseGT at a given time-stamp; time on the x axis, meters on the y. This is again for the incremental map building case. It is clearly visible where the robot passes the GT area the second time.

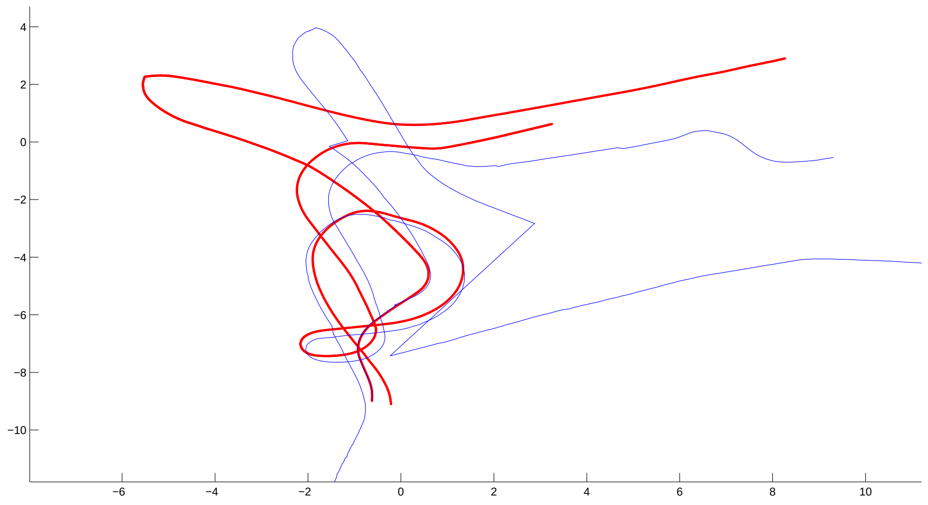


Figure 75: In this image we show the poses of the poseGT (in red) and the estimated robot poses (in blue), for the robot path obtained after the loop closure; both axes in meters. Note that the estimated robot poses, in the second pass in the GT area, are now translated much less than meters.

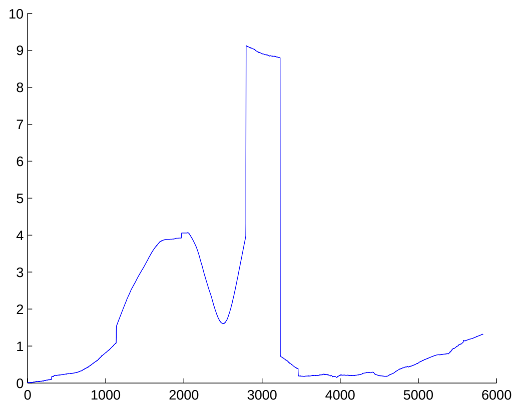


Figure 76: In this image we show the ATE error vector, i.e., the distance between the estimated robot pose and the poseGT at a given time-stamp, for the robot path obtained after the loop closure; time on the x axis, meters on the y. Notice that here the robot enters the GT area for the second time, on the falling edge just after time $= 3000$, and that here the error is very small.

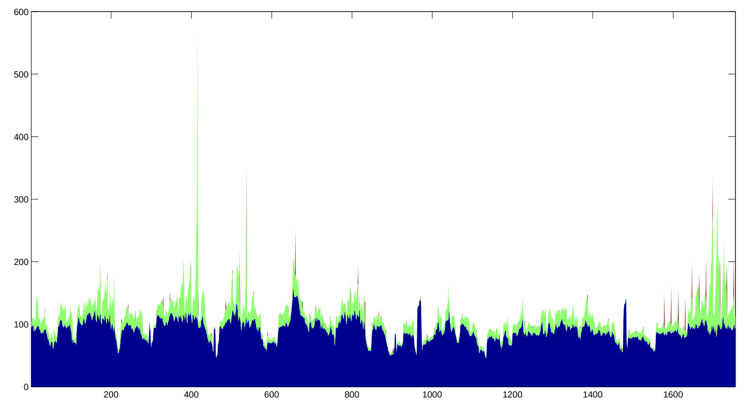


Figure 77: Timings. Acquisition time instant on the x, computation time on the y. Note that in this image a very relevant part of the computation is not included; the missing processing time is devoted to the determination of the associations between segments of the two maps that passed the preliminary check on the superimposition of the bounding boxes

6.3.2 Rough Estimate of Complexity (REC)

inserire la descrizione di come si sono ottenute le varie sottoparti del tempo complessivo.

The images 77, 78 represent the running time for our Trinocular Stereo SLAM algorithm. On the horizontal axis we have the data acquisition time. On the vertical axis we have the processing time required by the algorithm, differentiated by colour. Here the blue represents the time required by the image processing for segment extraction. The green represents the sum of the computation time of the incremental map building and of the trinocular matching. The red represents the simple bounding box check between potentially superimposed sub-maps plus the whole map relaxation. What is missing here is the very demanding association of data in the two sub-maps. The huge requirement of this phase is clarified in the following Figures 79, 80.

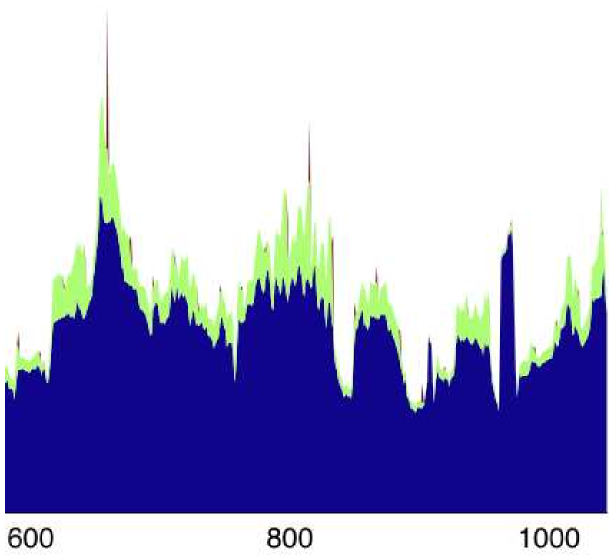


Figure 78: Timings. Acquisition time instant on the x, computation time on the y. Magnification of the previous image

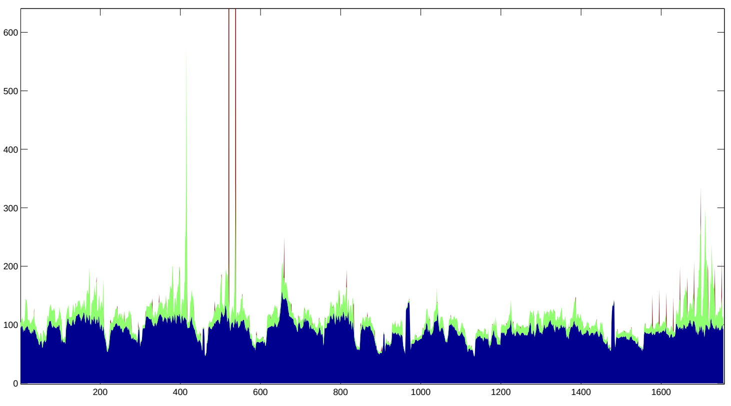


Figure 79: Timings. Acquisition time instant on the x, computation time on the y. The red spikes that go beyond the ceiling of the picture are due to a particular step of the loop closure, i.e., the search for correspondences between segments of two overlapping sub-maps. In this picture just two searches are reported, the corresponding timings come from the execution from a different (faster) machine, and are here presented in order to give an idea of their magnitude, in comparison to the other processing.

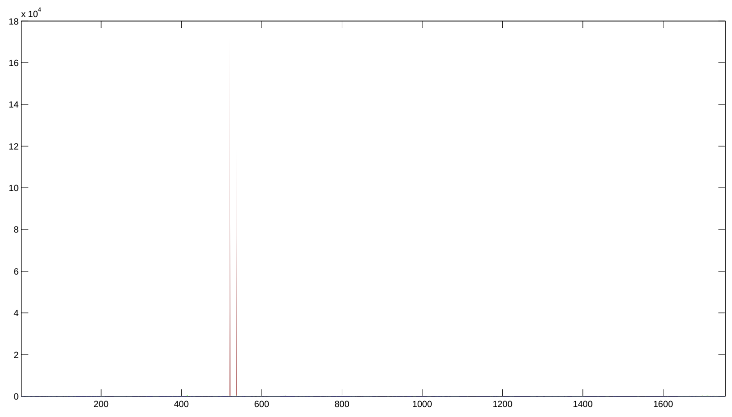


Figure 80: Timings. Acquisition time instant on the x, computation time on the y. Here you can better appreciate the magnitude of the computational requirements of the most cumbersome part of the loop closure, in comparison to the other processing.

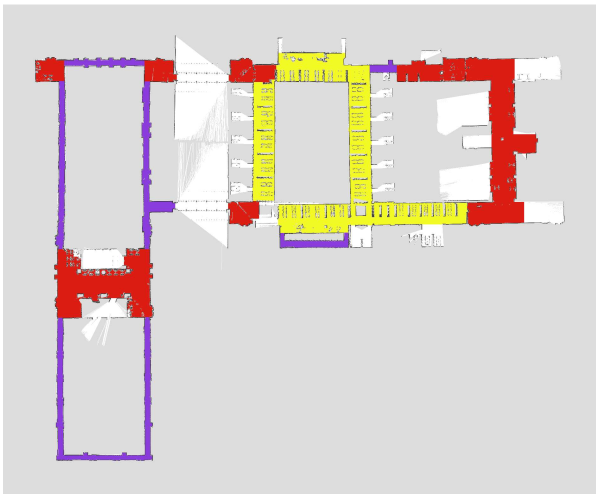


Figure 81: This figure illustrates the training dataset for the RAWSEEDS maps. The classes are labeled as corridors (purple), hallways (red) and cluttered hallways (yellow)

7 Semantic Place Labeling

In this section, we present the results of the semantic place labeling approach of Martínez Mozos *et al.* [31] on the RAWSEEDS indoor datasets. This techniques has not been developed within RAWSEEDS and thus we explain this method only briefly here. More details can be founnd in [31].

The method of Martínez Mozos *et al.* runs on closed 2D grid maps and uses manually labeled maps for learning the semantic labels. In a second step, the method samples robot positions in the known part of the maps and simulates laser readings. This laser readings are classified according to the learned labels and by considering the neighborhood around the current observation pose. While this method is well suited for closed indoor maps, the increasing number of max ranges in outdoor environments makes it unusable in large outdoor maps. Therefore, we processed the maps of the five RAWSEEDS indoor datasets, taken in the Bicocca location. We choose three different classes for the place labeling to distinguish between corridors (purple), hallways (red) and cluttered hallways (yellow). Cluttered hallways are chosen as class, since we have only very few rooms in the datasets. Figure 81 shows the training dataset used to learn the classification parameters.

Figure 82 shows the result for the RAWSEEDS indoor datasets. Over all the hallway areas are correctly classified and marked with red dots. In contrast the corridor classification for the maps is poor. Large corridor areas are false classified with yellow dots. A explanation for that could be that the training dataset had no open doors in the corridors and that areas with open doors look like cluttered hallways. The classification of cluttered hallways is also stable.

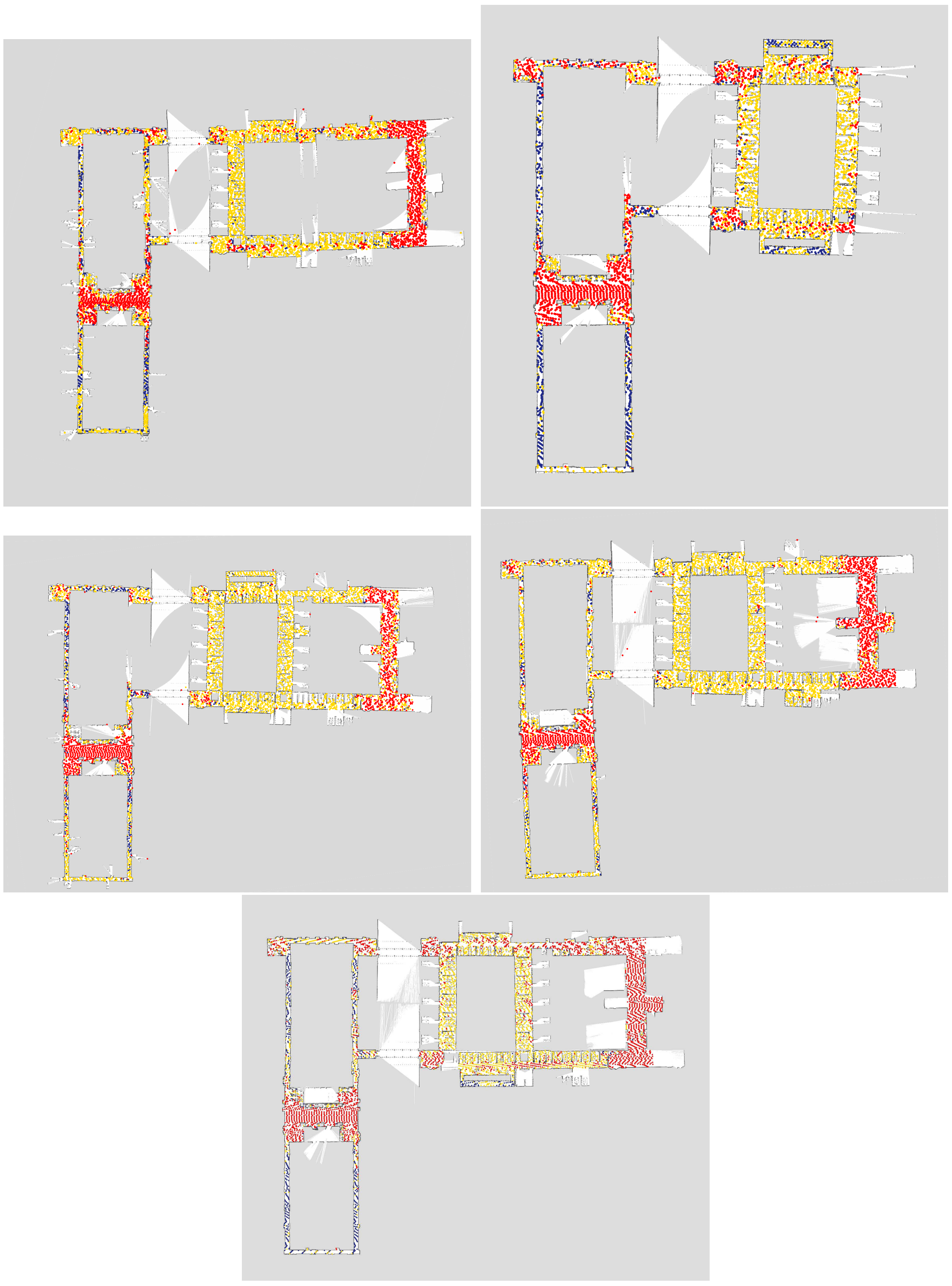


Figure 82: This Figure shows the results of the semantic place labeling algorithm for the RAWSEEDS indoor datasets. Bicocca-2009-02-25a is shown top left, Bicocca-2009-02-25b is shown top right, Bicocca-2009-02-26a is shown middle left, Bicocca-2009-02-26b is shown middle right and Bicocca-2009-02-27a is shown at the bottom.

8 Coordinated Multi-Robot Exploration

In this section we describe the approach of Wurm *et al.* [51] that had been applied on the RAWSEEDS indoor location. This method is not suited to run on outdoor datasets since it needs closed maps. The video showing the results will be available at the RAWSEEDS website.

The key issue in coordinated multi-robot exploration is how to assign target locations to the individual robots such that the overall mission time is minimized. This approach takes into account the structure of the environment to distribute the robots over the environment. To achieve this, it partitions the space into segments, for example, corresponding to individual rooms. Instead of only considering frontiers between unknown and explored areas as target locations, the approach sends the robots to the individual segments with the task to explore the corresponding area.

We ran the described approach on one outdoor and one indoor map using with X simulated robots exploring the environment. We chose the maps which covers most of the area for both locations. Videos of the coordinated Multi-Robot Exploration will be made available on the RAWSEEDS website.

8.1 Target Assignment using the Hungarian Method

In 1955, Kuhn [27] presented a general method, which is often referred to as the Hungarian method, to assign a set of jobs to a set of machines given a fixed cost matrix. Consider a given $n \times n$ cost matrix which represents the cost of all individual assignments of jobs to machines. The Hungarian method, which is able to find the optimal solution with the minimal cost given this matrix, can be summarized by the following three steps:

1. Compute a reduced cost matrix by subtracting from each element the minimal element in its row. Afterwards, do the same with the minimal element in each column.
2. Find the *minimal number* of horizontal and vertical lines required to cover all zeros in the matrix. In case exactly n lines are required, the optimal assignment is given by the zeros covered by the n lines. Otherwise, continue with Step 3.
3. Find the smallest nonzero element in the reduced cost matrix that is not covered by a horizontal or vertical line. Subtract this value from each uncovered element in the matrix. Furthermore, add this value to each element in the reduced cost matrix that is covered by a horizontal and a vertical line. Continue with Step 2.

The computationally difficult part lies in finding the minimum number of lines covering the zero elements (Step 2). The overall algorithm has a complexity of $O(n^3)$. The method described above can directly be applied to assign a set of target locations (tasks) to the individual robots (machines). Here, each entry in the cost matrix can be the length of the path the corresponding robot has to travel to reach the designated target point.

Since the implementation of the Hungarian method described above requires the number of jobs and the number of machines to be equal, the cost matrix needs to be slightly adapted to compute it that way. This can be achieved by expanding the cost matrix using "dummy machines" (which will result in target locations that are not approached by any of the robots) and by duplicating existing targets. The Hungarian Method is then able to compute the optimal assignment, given the cost matrix.

8.2 Map Segmentation

Several researchers investigated the problem of segmenting maps based on the partitioning of a graph [4, 19, 28, 50, 52]. A very popular graph-based representation in this context are *Voronoi Graphs* (VGs) [8]. To compute the Voronoi Graph $G(m) = (V, E)$ of a given map m, we consider the set $O_p(m)$ which contain for each point p in the free-space C of m the set of closest obstacle points. The Voronoi Graph then is given by the set of points in $O_p(m)$ for which there are at least two obstacle points with an equal minimal distance:

$$V = \{p \in C \mid \; |O_p(m)| \geq 2\} \tag{31}$$
$$E = \{(p, q) \mid \; p, q \in V, \; p \text{ adjacent } q \text{ in } m\} \tag{32}$$

For each pair of nodes in $G(m)$ we add an edge if their corresponding points in m are adjacent. The Voronoi Graph can be generated from metric maps of the environment such as occupancy grid maps [9, 50]. In a practical implementation this can be efficiently done by applying the Euclidean distance transformation [33] to an occupancy grid map. This transformation results in a distance map which holds for each grid cell the distance to the closest obstacle. A Voronoi Graph can then be constructed using skeletonization on the distance map. Figure 83 illustrates the process of generating a Voronoi Graph for an example occupancy grid map.

After generating the Voronoi Graph the next interesting step is the partitioning of the graph into k disjoint sets $V_1, V_2, \ldots, V_k$ with

$$V = \bigcup_{i=1}^{k} V_i \tag{33}$$

such that each cluster of nodes V_i corresponds to a segment which can be assigned to a robot. Thrun *et al.* suggest the graph to be separated at so-called *critical points* [50]. Here, critical points are those nodes in the Voronoi Graph at which the distance to the closest obstacle in the map is a local minimum. This is usually the case in doorways or other narrow passages.

Whereas this approach is able to reliably find doorways, it also generates a lot of false positive candidates in cluttered environments. The false positives are reduced using the following constrains: First, critical points have to be nodes of degree 2 (two edges) and second, need to have a neighbor of degree 3 (a junction node). In addition the points have to lead from known into unknown areas, since segments which do not contain unknown areas can safely be ignored in an exploration task. This constraint is verified by computing the distance to the closest reachable unknown cell for each point. This can be done efficiently in a similar way as the computation of the distance map. Figure 84 shows a pruned version of the Voronoi graph and the critical points found by the algorithm. All doorways have been selected as candidates and the number of false positives is much smaller than the number of critical points according to the definition of Thrun *et al.* [50] which includes distance minima in the Euclidean distance transformation within corridors and rooms.

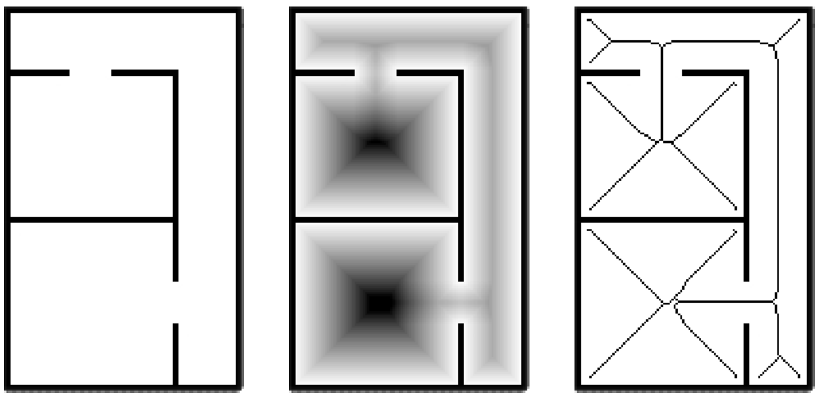


Figure 83: Generation of the Voronoi Graph. Left: Example grid-map. Center: Map plus distance transform (the darker a point the larger the distance to the closest obstacle). Right: Map and Voronoi Graph generated from the distance transform using skeletonization.

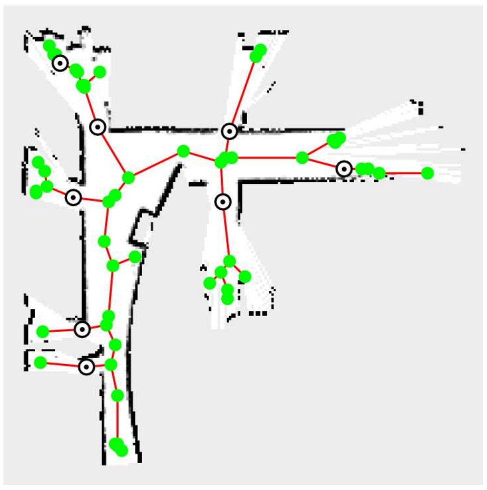


Figure 84: Example segmentation of a small fraction of an environment. The marked nodes are the candidates for the partitioning of the graph calculated by the described approach.

8.3 Assignment of Robots to Target Areas

Indoor environments are in general structured environments. Buildings are usually divided into rooms which can be reached via corridors. In many cases, it can be a disadvantage to assign more than one robot to one room. The room might, for example, be too small for a second robot to speed up it's exploration even though there initially is more than one frontier in the room. When the room is fully explored, robots might even block each other while trying to leave the room which will result in an increase in exploration time.

The described approach assigns individual robots to different segments of unexplored space. Segments could be separate rooms, corridors, or parts of larger corridors or rooms. This takes into account the structure of the environment and prevents the forming of inefficient clusters of robots.

Algorithm 3 Target Assignment Using Map Segmentation.

```
1: Determine segmentation $S = \{s_1, ..., s_n\}$ of map.
2: Determine the set of frontier targets for each segment.
3: Compute for each robot $i$ the cost $C^i_s$ for reaching each map segment $s \in S$.
4: Discount cost $C^i_s$ if robot $i$ is already in segment $s$.
5: Assign robots to segments using the Hungarian Method.
6: for all segments $s$ do
7:    Assign robot(s) to frontier targets in $s$ w.r.t. path costs using the Hungarian Method.
8: end for
```

The assignment algorithm is summarized in Algorithm 3. An assignment is determined whenever one of the robots requests a new exploration target. First, a partition of the partial map of the environment is created using the graph-based method described in Section 8.2. To generate targets within the segments, we then determine the set of frontier cells. The cost C^i_s for reaching segment s with robot i is defined as the expected path cost to the nearest frontier cell within s. This cost is discounted by a constant factor if robot i is already located in segment s. This has the effect that the robots stay in their assigned segment until it is completely explored. After computing the costs of a segment, an assignment is calculated by applying the Hungarian method (see Section 8.1) based on the cost matrix.

The Hungarian method does not assign more than one robot to the same segment unless there are more robots available than there are unexplored segments. To appropriately handle those cases in which multiple robots are assigned to a single segment, we apply a local assignment based on the cost-optimal frontier within a segment. For this reason, our algorithm is equivalent to a purely frontier-based assignment if the environment cannot be partitioned, i.e., there is only one segment.

By assigning robots to separate segments, an appropriate distribution of the robots can be achieved. Instead of aiming at the closest frontier, robots share work more efficiently. A typical office environment, for example, contains corridors and rooms. Using this approach, each of the corridors is explored completely by one of the robots. In this way, the rough structure of the building will quickly be revealed. Meanwhile other robots will be assigned to the rooms reachable from the corridors, one at a time.

9 Conclusion

This deliverable D5.2 “Final Benchmark Solutions” provides according to the description of work (Annex I) a set of benchmarking solutions, which is (i) a description and the software implementation of the corresponding SLAM algorithms, (ii) the output of the algorithm on a given benchmarking problem (a dataset), and (iii) the score of rating the output of the algorithm according to a quality measure defined in the benchmarking problem.

For laser based SLAM, we provided benchmarking solutions for scan matching, a mapping system based on a Rao-Blackwellized particle filter (“GMapping”, see attached papers) and a graph mapping system (“TORO”, see attached papers) evaluation of the 11 RAWSEEDS datasets. We furthermore carried out the evaluations for vision-based SLAM systems. We provided algorithms for monocular, stereo, and trinocular SLAM. We also presented the results of our semantic place labeling techniques as well as our multi-robot exploration approach.

References

[1] David Arthur and Sergei Vassilvitskii. k-means++: the advantages of careful seeding. In *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[2] Yaakov Bar-Shalom, X. Rong Li, and Thiagallingam Kirubarajan. *Estimation with Applications to Tracking and Navigation*. Wiley-Interscience, June 2001.

[3] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. *Computer Vision and Image Understanding (CVIU)*, 110(3):346–359, 2008.

[4] P. Beeson, N.K. Jong, and B. Kuipers. Towards autonomous topological place detection using the extended voronoi graph. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2005.

[5] Wolfram Burgard, Michael Ruhnke, Giorgio Grisetti, and Cyrill Stachniss. Rawseeds deliverable d5.1: Preliminary benchmark solutions, 2009. Available from: <http://www.rawseeds.org>, verified Oct. 1st 2009.

[6] J. A. Castellanos, R. Martínez-Cantín, J. Neira, and J. D. Tardós. Robocentric map joining: Improving the consistency of EKF-SLAM. *Robotics and Autonomous Systems*, 55(1):21–29, January 2007.

[7] A. Censi. Scan matching in a probabilistic framework. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2291–2296, Orlando, Florida, 2006.

[8] H. Choset, , and Burdick J. Sensor-based exploration: The hierarchical generalized voronoi graph. *Int. Journal of Robotics Research*, 19(2), 2000.

[9] H. Choset and J. Burdick. Sensor based planning, part i: The generalized voronoi graph. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Nagoya, Japan, 1995.

[10] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth to depth conversion for monocular SLAM. In *IEEE International Conference on Robotics and Automation, 2007*, pages 2778–2783, April 2007.

[11] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Transactions on Robotics*, 2008. Submitted to IEEE Transactions on Robotics.

[12] J. Civera, O. García-Grasa, A. J. Davison, and J. M. M. Montiel. 1-point RANSAC for EKF-based structure from motion. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[13] J. Civera, O. García-Grasa, and J. M. M. Montiel. 1-point RANSAC for filtering. application to EKF-based visual odometry. In *Submitted to IEEE International Conference on Robotics and Automation*, 2010.

[14] F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte carlo localization for mobile robots. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Leuven, Belgium, 1998.

[15] A. Doucet. On sequential simulation-based methods for bayesian filtering. Technical report, Signal Processing Group, Dept. of Engeneering, University of Cambridge, 1998.

[16] A. Eliazar and R. Parr. DP-SLAM: Fast, robust simultainous localization and mapping without predetermined landmarks. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1135–1142, Acapulco, Mexico, 2003.

[17] Carlos Estrada, José M. Neira, and Juan D. Tardós. Hierarchical slam: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, August 2005.

[18] U. Frese, P. Larsson, and T. Duckett. A multilevel relaxation algorithm for simultaneous localisation and mapping. *IEEE Transactions on Robotics*, 21(2):1–12, 2005.

[19] S. Friedman, H. Pasula, and D. Fox. Voronoi random fields: Extracting topological structure of indoor environments via place labeling. In Manuela M. Veloso, editor, *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 2109–2114, 2007.

[20] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard. Efficient estimation of accurate maximum likelihood maps in 3D. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, San Diego, CA, USA, 2007.

[21] G. Grisetti, D. Lodi Rizzini, C. Stachniss, E. Olson, and W Burgard. Online constraint network optimization for efficient maximum likelihood map learning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Pasadena, CA, USA, 2008.

[22] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with rao-blackwellized particle filters. *IEEE Transactions on Robotics*, 23(1):34–46, 2007.

[23] G. Grisetti, C. Stachniss, S. Grzonka, and W. Burgard. A tree parameterization for efficiently computing maximum likelihood maps using gradient descent. In *Proc. of Robotics: Science and Systems (RSS)*, Atlanta, GA, USA, 2007.

[24] D. Hähnel, W. Burgard, D. Fox, and S. Thrun. An efficient FastSLAM algorithm for generating maps of large-scale cyclic environments from raw laser range measurements. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 206–211, 2003.

[25] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. of the European Conference on Computer Vision*, 1996.

[26] Peter D. Kovesi. MATLAB and Octave functions for computer vision and image processing. School of Computer Science & Software Engineering, The University of Western Australia, 2000. Available from: <http://www.csse.uwa.edu.au/∼pk/research/matlabfns/>, verified Oct. 1st 2009.

[27] H.W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1):83–97, 1955.

[28] B. Kuipers and Y.-T. Byun. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics & Autonomous Systems*, 8:47–63, 1991.

[29] F. Lu and E. Milios. Robot pose estimation in unknown environments by matching 2d range scans. In *IEEE Computer Vision and Pattern Recognition Conference (CVPR)*, pages 935–938, 1994.

[30] F. Lu and E. Milios. Globally consistent range scan alignment for environment mapping. *Journal of Autonomous Robots*, 4:333–349, 1997.

[31] O. Martínez-Mozos, C. Stachniss, A. Rottmann, and W. Burgard. *Robotics Research*, volume 28 of *STAR Springer tracts in advanced robotics*, chapter Using AdaBoost for Place Labelling and Topological Map Building. Springer Verlag, 2007.

[32] D. Marzorati, M. Matteucci, and D. G. Sorrenti. Particle-based sensor modeling for 3d-vision slam. In *Proceedings of IEEE International Conference on Robotics and Automation 2007*, April 2007.

[33] A. Meijster, J.B.T.M. Roerdink, and W.H. Hesselink. *Mathematical Morphology and its Applications to Image and Signal Processing*, chapter A General Algorithm for Computing Distance Transforms in Linear Time, pages 331–340. Kluwer Academic Publishers, 2000.

[34] M. Montemerlo, S. Thrun D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proc. of the Int. Conf. on Artificial Intelligence (IJCAI)*, pages 1151–1156, Acapulco, Mexico, 2003.

[35] M. Montemerlo, N. Roy, S. Thrun, D. Hähnel, C. Stachniss, and J. Glover. CARMEN – the carnegie mellon robot navigation toolkit. http://carmen.sourceforge.net, 2002.

[36] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to simultaneous localization and mapping. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, pages 593–598, Edmonton, Canada, 2002.

[37] Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Accurate non-iterative o(n) solution to the pnp problem. *Computer Vision, IEEE International Conference on*, 0:1–8, 2007.

[38] J. Neira and J. D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. R&A*, 17(6):890–897, 2001.

[39] José M. Neira, Juan D. Tardós, and José A. Castellanos. Linear time vehicle relocation in slam. In *Proceedings of IEEE International Conference on Robotics and Automation 2003*, September 2003.

[40] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 2, pages 2161–2168, 2006.

[41] E. Olson, J. Leonard, and S. Teller. Fast iterative optimization of pose graphs with poor initial estimates. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 2262–2269, 2006.

[42] Lina M. Paz, Pedro Piniés, Juan D. Tardós, and José Neira. Large Scale 6DOF SLAM with Stereo in Hand. *Transactions on Robotics*, 24(5):946–957, October 2008.

[43] P. Piniés, L.M. Paz, and J. D. Tardós. CI-Graph: An efficient approach for large scale SLAM. In *IEEE Int. Conf. on Robotics and Automation*, Kobe, Japan, 2009. To appear.

[44] P. Piniés and J. D. Tardós. Large scale slam building conditionally independent local maps: Application to monocular vision. *IEEE Trans. on Robotics*, 24(5):1094–1106, October 2008.

[45] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International Conference on Computer Vision*, volume 2, pages 1470–1477, October 2003.

[46] C. Stachniss, G. Giorgio, W. Burgard, and N. Roy. Analyzing gaussian proposal distributions for mapping with rao-blackwellized particle filters. In *Proc. of the Int. Conf. on Intelligent Robots and Systems (IROS)*, 2007.

[47] C. Stachniss and G. Grisetti. GMapping project at OpenSLAM.org. http://openslam.org/gmapping.html, 2007.

[48] C. Stachniss and G. Grisetti. TORO project at OpenSLAM.org. http://openslam.org/toro.html, 2008.

[49] Juan D. Tardós, José M. Neira, Paul M. Newman, and John J. Leonard. Robust mapping and localization in indoor environments using sonar data. *The International Journal of Robotics Research*, 21(4):311–330, April 2002.

[50] S. Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artificial Intelligence*, 99(1):21–71, 1998.

[51] K.M. Wurm, R Kuemmerle, C. Stachniss, and W. Burgard. Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, St. Louis, MO, USA, October 2009. Accepted for publication.

[52] Z. Zivkovic, B. Bakker, and B. Kröse. Hierarchical map building and planning based on graph partitioning. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, pages 803–809, 2006.

10 Attached Documents

The remainder of this deliverable consists of selected scientific RAWSEEDS publications of members of the consortium that provide more details on the individual algorithm described above.