RAWSEEDS Deliverable D5.1: Preliminary Benchmark Solutions

Wolfram Burgard Michael Ruhnke Giorgio Grisetti Cyrill Stachniss

1 Introduction

This deliverable reports about the *preliminary 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 current version, we performed our evaluation based on a set of publicly available datasets and on parts of the RAWSEEDS datasets (due to reasons of availability). For the final deliverable of WP5 (D5.2), we will construct a set of benchmark solutions based all datasets acquired within RAWSEEDS and we will additionally provide the results of our semantic place labeling techniques as well as on multi-robot exploration according to the Annex I.

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 preliminary 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) and based upon the underlying basic principle (Extended Kalman filter, particle filter, or graph-optimization).

2 Performance Metric

This section gives a brief review of the used metric which is included in D5.1 to make this document self-containing but should not be regarded as a contribution of D5.1.

One attractive way for measuring the performance of a SLAM algorithm is to consider the poses of the robot during data acquisition compared to ground truth information rather than comparing maps. In this way, we gain two important properties: First, it allows us to compare the result of algorithms that generate different types of metric map representations, such as feature-maps or occupancy grid maps. Second, the method is invariant to the sensor setup of the robot. Thus, a result of a graph-based SLAM approach working on laser range data can be compared, for example, with the result of vision-based FastSLAM. The only property we require is that the SLAM algorithm estimates the trajectory of the robot given by a set of poses.

2.1 Benchmarking based on the Trajectory of the Robot

Let $x_{1:T}$ be the poses of the robot estimated by a SLAM algorithm from time step 1 to T. Let $x^*_{1:T}$ be the reference poses of the robot, ideally the true locations. We can define a measure that uses the deformation energy that is needed to deform the estimated trajectory into the ground truth. This can be done – similar to the ideas of the graph mapping introduced by Lu and Milios [24] – by considering the nodes as masses and connections between them as springs. Thus, our metric is based on the *relative* displacement between poses. Instead of comparing x to x^* in the global reference frame, we do the operation based on δ and δ^* as

$$\varepsilon(\delta) \;=\; \frac{1}{N}\sum_{i,j} trans(\delta_{i,j} \ominus \delta^*_{i,j})^2 + rot(\delta_{i,j} \ominus \delta^*_{i,j})^2, \qquad (1)$$

where N is the number of relative relations and $trans(\cdot)$ and $rot(\cdot)$ are used to separate the translational and rotation components. We suggest to provide both quantities individually. The mathematical definition of this metric, however, leaves open which relative displacements $\delta_{j,i}$ are included in the summation in Eq. 1. We propose that the relative displacements have to be provided by the *benchmarking problem*, which according to the Annex I, requires to provide the log file and the ground truth information.

In addition to Eq. 1, one can define the metric according to the absolute error rather than the energy. Then the metric can be specified accordingly as

$$\varepsilon(\delta) \;=\; \frac{1}{N}\sum_{i,j} ||trans(\delta_{i,j} \ominus \delta^*_{i,j})|| + ||rot(\delta_{i,j} \ominus \delta^*_{i,j})|| \qquad (2)$$

The squared error measures to the average energy per constraint required to deform the current estimate in the ground truth. The absolute error can be interpreted as the average metric displacement between the estimated and the true relative transformations.

2.2 Benchmarking based on the Trajectory of the Estimated Map

Some SLAM approaches do not estimate the trajectory of the robot. Especially for "camera-in-hand" systems, only the resulting map of features is typically considered. In this case, we can, however, adapt our metric to operate on the landmark locations accordingly and not on the trajectory of the robot. One then considers the relative displacements of landmarks. The disadvantage of this technique is that the data association between the estimated map and the ground truth feature map has to be generated by the user. However, this is the only possibility for benchmarking given that no trajectory estimate is available.

2.3 Selecting Relative Displacements for Evaluation

The benchmarking problem defines the relative displacements $\delta_{j,i}$ used in Eq. 1 for a given dataset. The information can be extracted from accurate building blue print or by manual measurements or manual scan alignment. This, however, is not part of the benchmarking solution and has to be provided by the benchmarking problem.

3 Benchmark Solutions for 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.

Due to the unavailability of validated RAWSEEDS datasets especially in the beginning of this WP, we present in this section the results of the algorithms on five publicly available datasets. For the final deliverable of WP5 (D5.2), the evaluation will be done for all RAWSEEDS datasets by comparing the outputs of the SLAM algorithms according to the metric of Eq. 1.

3.1 Scan-Matching

Scan matching is the computation of the incremental, open loop maximum likelihood trajectory of the robot by matching consecutive scans [23, 4].

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

$$\hat{x}_t \quad = \quad \underset{x_t}{\operatorname{argmax}} \left[p(z_t \mid x_t, \hat{m}(\hat{x}_{1:t-1}, z_{1:t-1})) \cdot p(x_t \mid u_{t-1}, \hat{x}_{t-1}) \right]. \tag{3}$$

The idea is to trade off the consistency of the measurement with the map (first term on the right-hand side in (3)) and the consistency of the new pose with the control action and the previous pose (second term on the right-hand side in (3)). 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 closing 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 [9], to build both feature-maps [25, 27] and grid-maps [11, 17, 19], and to track objects based on vision data [21]. 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 [34] (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 [35]. 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

$$p(x_{1:t}, m \mid z_{1:t}, u_{1:t-1}) \quad = \quad p(m \mid x_{1:t}, z_{1:t}) \cdot p(x_{1:t} \mid z_{1:t}, u_{1:t-1}), \tag{4}$$

where $z_{1:t}$ is the observation sequence and $u_{1:t-1}$ the odometry information. In practice, the first term of Eq. (4) 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

$$\pi(x_{1:t} \mid z_{1:t}, u_{1:t-1}) \quad = \quad \pi(x_t \mid x_{1:t-1}, z_{1:t}, u_{1:t-1})\pi(x_{1:t-1} \mid z_{1:t-1}, u_{1:t-2}). \tag{5}$$

According to Doucet [10], the distribution

$$p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) = \frac{p(z_t \mid m_{t-1}^{(i)}, x_t) p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})} \tag{6}$$

is the optimal proposal for particle i with respect to the *variance of the particle weights* that satisfies Eq. (5). This proposal minimizes the degeneracy of the algorithm (Proposition 4 in [10]). As a result, the computation of the weights turn into

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \frac{\eta p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1})} && (7) \\ &\propto w_{t-1}^{(i)} \frac{p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{\frac{p(z_t \mid m_{t-1}^{(i)}, x_t) p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})}} && (8) \\ &= w_{t-1}^{(i)} \cdot p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}) && (9) \\ &= w_{t-1}^{(i)} \cdot \int p(z_t \mid x') p(x' \mid x_{t-1}^{(i)}, u_{t-1}) \, dx'. && (10) \end{aligned}$$

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 [34], 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 [17] 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 known from standard Monte-Carlo localization approaches (c.f. [9]) 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. (6). As a result, each sample converges to the mode in the likelihood function that is closest to its own starting position. Since the individual

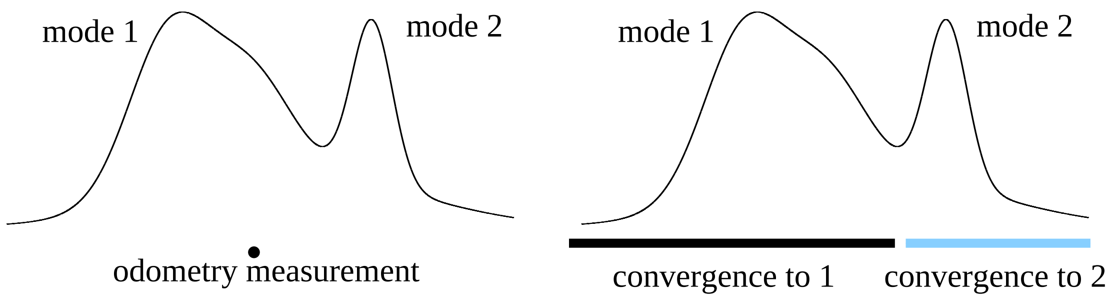


Figure 1: The left image illustrates a 1D likelihood function and an odometry measurement. Conventional informed sampling first performs scan-matching starting from the odometry measurement. In this situation, the scan-matcher will find a local peak in the likelihood function (most likely mode 1) and 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.

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 [34], this technique leads to proposal distributions which are closer to the optimal proposal given in Eq. (6) than the Gaussian approximations; when the distribution has only a single mode, the solution is equivalent to previous approaches [17].

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 [14, 24, 29]. The approach briefly described here belongs to this class of methods. For a detailed description see [18, 15, 16, 36].

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

$$e_{ji}(\mathbf{x}) \quad = \quad f_{ji}(\mathbf{x}) - \delta_{ji} \tag{11}$$

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

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} \sum_{\langle j,i \rangle \in \mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). \tag{12}$$

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 parameterization 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

$$\delta_{i+1,i} = \operatorname*{argmax}_{\delta} p(\delta \mid z_i, z_{i+1}, u_i). \tag{13}$$

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 [26].

We determine the so called loop closing constraints $\delta_{j,i}$ between the current robot location x_j and some previous location x_i by running Monte Carlo Localization [9] 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.* [29] 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. (12) 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:

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^{t} + \lambda \cdot \mathbf{H}^{-1} J_{ji}^{T} \Omega_{ji} r_{ji} \tag{14}$$

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. (14), we refer the reader to our previous works [18, 29].

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 subproblems 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 Parameterization 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 parameterization defines the subset of variables that are modified when updating a constraint. Therefore, the way the nodes are parameterized has a serious influence on the performance of the system. We proposed to use a tree [18] as an efficient way of parameterizing 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 parameterization for a node as

$$x_i \quad = \quad p_i - p_{\mathrm{parent}(i)}, \tag{15}$$

where $p_{\mathrm{parent}(i)}$ refers to the parent of node i in the spanning tree. As defined in Eq. (15), 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

$$r'_{ji} \quad = \quad \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} x_{k^{[-]}} - \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} x_{k^{[+]}} + R_i \delta_{ji}. \tag{16}$$

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 parameterization 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 [29], we compute an approximation of the Jacobian as

$$J'_{ji} \quad = \quad \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} \mathcal{I}_{k^{[+]}} - \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} \mathcal{I}_{k^{[-]}}, \tag{17}$$

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

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t + \lambda |\mathcal{P}_{ji}| \mathbf{M}^{-1} \Omega'_{ji} r'_{ji}, \tag{18}$$

where $|\mathcal{P}_{ji}|$ refers to the number of nodes in $\mathcal{P}_{ji}$. In Eq. (18), we replaced the preconditioning matrix $\mathbf{H}^{-1}$ with its scaled approximation $\mathbf{M}^{-1}$ as described in [29]. 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 parameterization 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 [18, 15, 16] (see attachment to D5.1), which is avaliable as open source at [36], 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 parameterization 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.

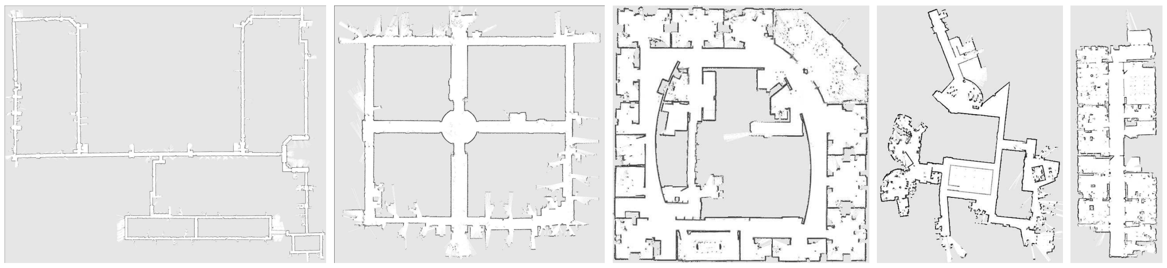


Figure 2: Maps obtained by the reference datasets used construct RAWSEEDS Benchmark Solutions. From left to right: MIT Killian Court, ACES Building at the University of Texas, Intel Research Lab Seattle, MIT CSAIL Building, and building 079 University of Freiburg.

3.4 Performances of Laser-Based Approaches

We run open source implementations of the approaches described above on well known and publicly available datasets (see Fig. 2). For obtaining close to ground truth information of these non RAWSEEDS datasets, we extracted a set of ground truth relations by manually matching sets of nearby scans. We finally measured the performances of each algorithm on each dataset by using Eq 1 and Eq 2.

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.

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. A distinction between RBPF and graph mapping seems difficult as both algorithms perform well in general. On average, graph mapping seems to be slightly better than a RBPF for mapping.

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. In case of high errors in a block of relations, we label the relations in the maps. This enables us to see not only where an algorithm fails, but might also provide insides why it fails. Inspecting those situations in correlation with the map helps to understand the properties of algorithms and give valuable insights on its capabilities. For two datasets, a detailed analysis using these plots is presented in the following sections.

MIT Killian Court In the MIT Killian Court dataset (also called the infinite corridor dataset), the robot mainly observed corridors with only few structures that support accurate pose correction. The robot traverses multiple nested loops – a challenge especially for the RBPF-based technique. We extracted close to 5000 relations between nearby poses that are used for evalua-

Table 1: Quantitative results of different approaches/datasets.

Trans. error m / m^2	Scan Matching	RBPF (50 part.)	Graph Mapping
Aces (abs)	0.173 ± 0.614	0.060 ± 0.049	0.044 ± 0.044
Aces (sqr)	0.407 ± 2.726	0.006 ± 0.011	0.004 ± 0.009
Intel (abs)	0.220 ± 0.296	0.070 ± 0.083	0.031 ± 0.026
Intel (sqr)	0.136 ± 0.277	0.011 ± 0.034	0.002 ± 0.004
MIT (abs)	1.651 ± 4.138	0.122 ± 0.386[1]	0.050 ± 0.056
MIT (sqr)	19.85 ± 59.84	0.164 ± 0.814[1]	0.006 ± 0.029
CSAIL (abs)	0.106 ± 0.325	0.049 ± 0.049[1]	0.004 ± 0.009
CSAIL (sqr)	0.117 ± 0.728	0.005 ± 0.013[1]	0.0001 ± 0.0005
FR 79 (abs)	0.258 ± 0.427	0.061 ± 0.044[1]	0.056 ± 0.042
FR 79 (sqr)	0.249 ± 0.687	0.006 ± 0.020 [1]	0.005 ± 0.011

Rot. error $\mathbf{10^{-2}}$ $rad/^2$	Scan Matching	RBPF (50 part.)	Graph Mapping
Aces (abs)	2.0 ± 2.7	2.1 ± 2.3	0.7 ± 0.7
Aces (sqr)	0.1 ± 0.3	0.09 ± 0.21	0.01 ± 0.03
Intel (abs)	1.5 ± 1.5	4.3 ± 6.3	0.7 ± 0.7
Intel (sqr)	0.04 ± 0.08	0.06 ± 2.9	0.01 ± 0.03
MIT (abs)	4.0 ± 7.8	1.4 ± 1.7[1]	1.0 ± 1.5
MIT (sqr)	0.7 ± 2.0	0.04 ± 0.55[1]	0.03 ± 0.57
CSAIL (abs)	2.4 ± 7.9	5.0 ± 5.4[1]	0.1 ± 0.2
CSAIL (sqr)	0.7 ± 3.4	0.5 ± 1.4[1]	0.0003 ± 0.0013
FR 79 (abs)	2.9 ± 3.7	2.6 ± 2.7[1]	2.6 ± 2.7
FR 79 (sqr)	0.2 ± 0.4	0.1 ± 0.3[1]	0.1 ± 0.3

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

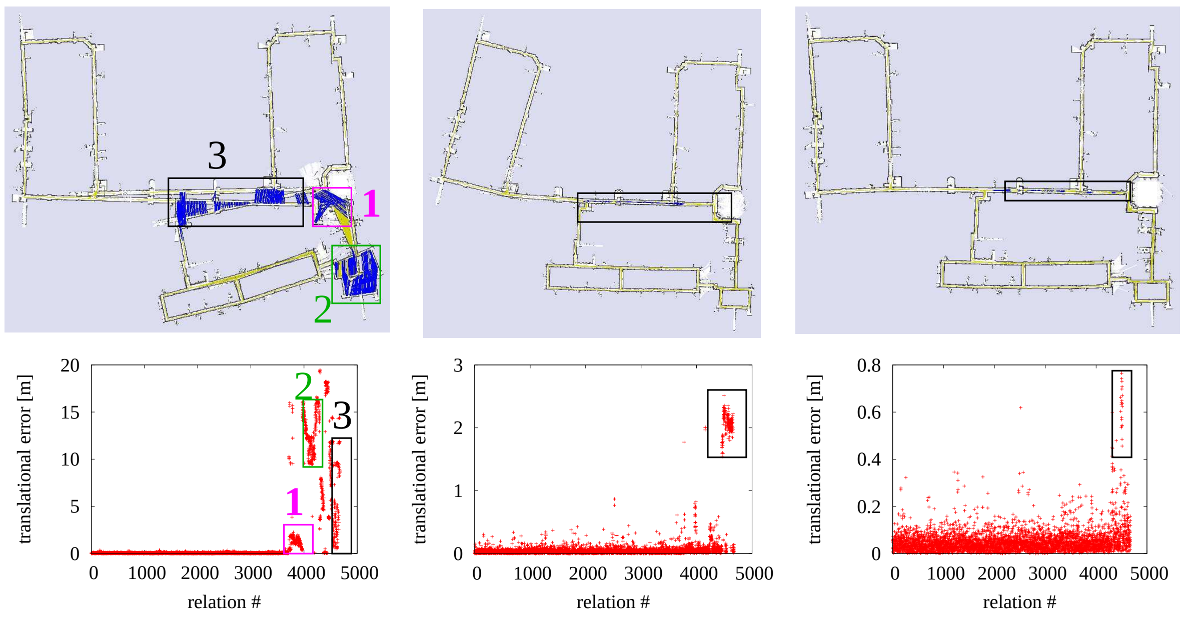


Figure 3: The MIT Killian Court dataset. The reference relations are depicted in light yellow. The left column shows the results of scan-matching, the middle column the result of a GMapping using 50 samples, and the right column shows the result of a graph-based approach. The regions marked in the map (boxes and dark blue relations) correspond to regions in the error plots having high error. The rotational error is not plotted due to space reasons.

tion. Figure 3 shows three different results and the corresponding error distributions to illustrate the capabilities of our method. Regions in the map with high inconsistencies correspond to relations having a high error. The absence of significant structure along the corridors results in a small or medium re-localization error of the robot in all compared approaches. In sum, we can say the graph-based approach outperforms the other methods and that error reflects the impression of a human about map quality obtained by visually inspecting the mapping results (the vertical corridors in the upper part are supposed to be parallel).

Freiburg Indoor Building 079 The building 079 of the University of Freiburg is an example for a typical office environment. The building consists of one corridor which connects the individual rooms. Figure 4 depicts the results of the individual algorithms (scan matching, RBPF, graph-based). In the first row of Figure 4, the relations having a translational error greater than 0.15 m are highlighted in blue.

In the left plot showing the scan matching result, the relations plotted in blue are generated when the robot revisits an already known region. These relations are visible in the corresponding error plots (Figure 4 first column, second and third row). As can be seen from the error plots, these relations with a number greater than 1000 have a larger error than the rest of the dataset. The fact that the pose estimate of the robot is sub-optimal and that the error accumulates can also be seen by the rather blurry map and that some walls occur twice. In contrast to that, the more

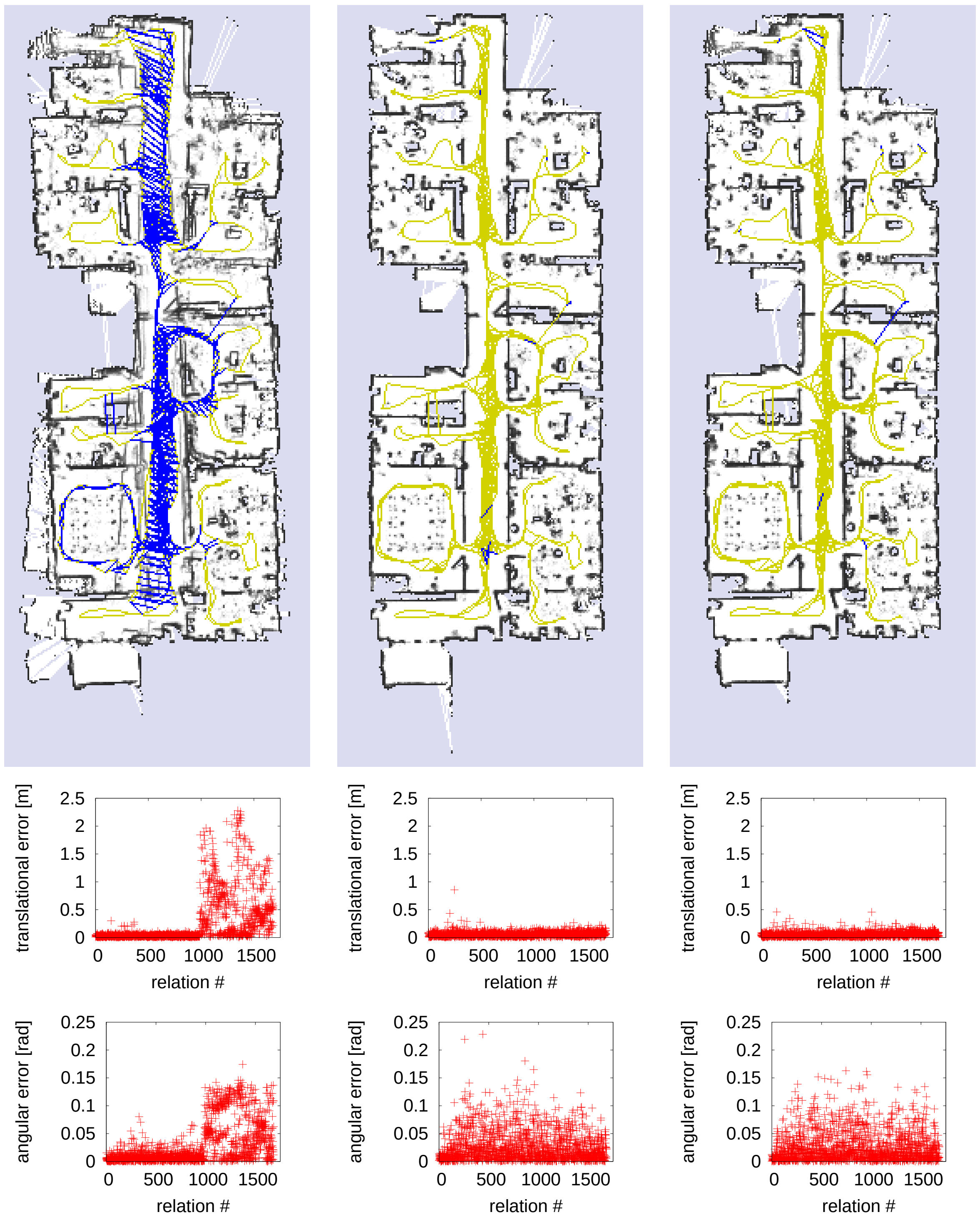


Figure 4: This figure shows the Freiburg Indoor Building 079 dataset. Each column reports the results of one approach. Left: scan-matching, middle: RBPF and right a graph based algorithm. Within each column, the top image shows the map, the middle plot is the translational error and the bottom one is the rotational error.

sophisticated algorithms, namely RBPF and graph mapping, are able to produce consistent and accurate maps in this environment. Only very few relations show an increased error (illustrated by dark blue relations).

All datasets, the manually verified relations, and map images are available online [22].

4 Benchmark Solutions for Monocular SLAM

In this section, we shortly describe a state-of-the-art SLAM algorithm that operates exclusively on a stream of single-camera images. Subsequently, we explain how to solve the inherent scaling problem of monocular SLAM to apply the performance measure of Eq. 1. We conclude this section with an evaluation of this algorithm on some datasets acquired in-house as well as on some RAWSEEDS datasets.

4.1 Single Camera SLAM

We used a sequential Bayesian approach to visual odometry described in [6],[7]. For completeness we attach the related papers to this deliverable.

The algorithm presented makes use of an Extended Kalman Filter with feature points represented using inverse-depth [6] and considers several hundreds of point features per frame, camera-centered. Compared to the usual EKF-SLAM, always referred to a world reference frame, we use here a sensor-centered approach, that greatly reduces the linearization error.

Another key difference of our approach is the number of measured features, which we rise from tens to about a hundred of them (see figure 5). The availability of ground truth in the RAWSEEDS datasets allowed us to verify that this highly improves the accuracy of the estimation and also makes the typical monocular SLAM scale drift, previously reported in [8, 30], almost vanish.

As in any visual estimation problem, the spurious rejection plays a fundamental role in our algorithm. Spurious rejection algorithms well suited for Bayesian estimation –like the classical JCBB [28] or the recent Active Matching [5]– become computationally intractable due to the high number of measured features. To overcome this problem, we adopt a RANSAC-based spurious rejection algorithm.

4.2 Solving the Scaling Problem of Monocular SLAM for Measuring the Performance

In order to facilitate posterior comparisons, the publicly available *RAWSEEDS* datasets have been used to benchmark the monocular SLAM algorithm. The great advantage of the RAWSEEDS datesets is that they have been captured with a multisensor platform both in indoors and outdoors environment, and ground truth is available. In this paper, three different outdoor sequences have been selected in order to make use of the Real Time Kinematics Differencial GPS data existing in those datasets.

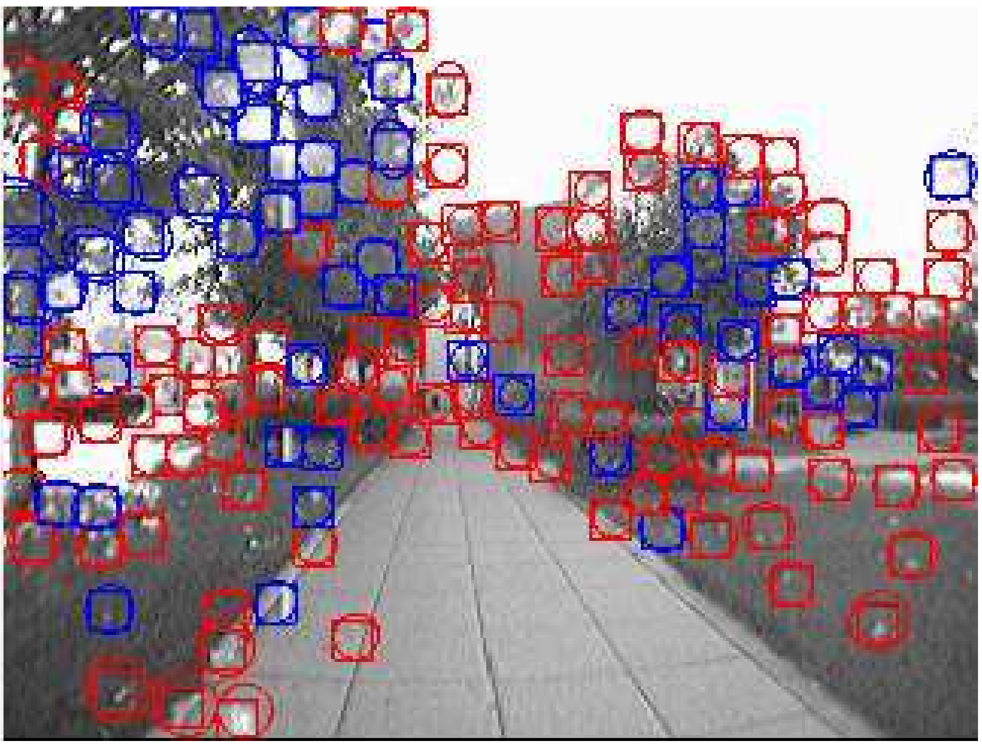


Figure 5: Example image from the monocular sequence in a RAWSEEDS dataset. Whereas squares represent the image patches of the tracked features, ellipses show the predicted uncertainty region where the correspondences will be looked for.

Our goal is to compare the trajectory estimated by the EKF-based visual odometry against the GPS data which we will consider as ground truth. The EKF-SLAM trajectory will be referred to the first camera frame of reference C_0. To do the comparison, a rigid transformation composed of a rotation R, translation t and scale factor s has to be applied. Such transformation converts every estimated camera position to the ground truth frame of reference.

$$\begin{bmatrix} x_{C_i} \\ y_{C_i} \\ z_{C_i} \\ 1 \end{bmatrix} = \begin{bmatrix} sR & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{C_i}^{C_0} \\ y_{C_i}^{C_0} \\ z_{C_i}^{C_0} \\ 1 \end{bmatrix} \tag{19}$$

The translation is computed to align the starting position with its ground truth value, and the unknown rotation and scale are obtained using a non-linear minimization algorithm, where these parameters are modified to minimize the error between the estimated trajectory and the GPS ground truth data.

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 the GPS ground truth, ignoring the vertical coordinate.

4.3 Benchmark results

Three different sequences from the mentioned *RAWSEEDS* datasets have been used to test the validity of the algorithm. All sequences have been recorded with the same camera, a 320×240 Unibrain with a wide-angle lens capturing at 30 fps. Camera calibration is provided in the dataset.

In the first sequence (*sequence 1*), composed by 3900 images, the robot translates for about 125 meters. The second sequence (*sequence 2*) has 5400 images and the robot describes a larger trajectory, about 185 meters. Finally, a very large challenging sequence (*sequence 3*) is evaluated that is composed by 22350 frames –12 minutes of video– in which the robot moves around a loopy trajectory of 630 meters. It is remarkable in this latter sequence that, although the accumulated drift already makes the error noticeable when plotted with the GPS ground truth data, the relative error with respect to the trajectory keeps the same low value as the other two shorter sequences.

Figures 7,8 and 9 shows the estimated trajectory (in red) and the GPS ground truth (in green) over a top view extracted from Google maps for each one of the sequences. From plain visual inspection, it can be seen in these figures that the estimated trajectory is not very far from the GPS trajectory. Table 2 gives detailed information about the errors obtained in the trajectory. As an illustration of the distribution of the errors, figure 10 shows its histogram for the 630 meters monocular sequence.

5 Benchmark Solutions for Stereo SLAM

In this section we describe our preliminary benchmark solution for stereo-based SLAM. This family of approaches utlilzes exclusively a sequence of stereo image pairs to compute a feature

Figure 6: Example of images taken from the 630 metres monocular sequence.

Table 2: EKF-based visual odometry estimation errors in the three experiments.

Trajectory length [m]	Mean error [m]	Maximum error [m]	% mean error over the trajectory
125	2.0	4.6	1.6%
185	2.1	6.0	1.1%
630	13.2	23.1	2.1%

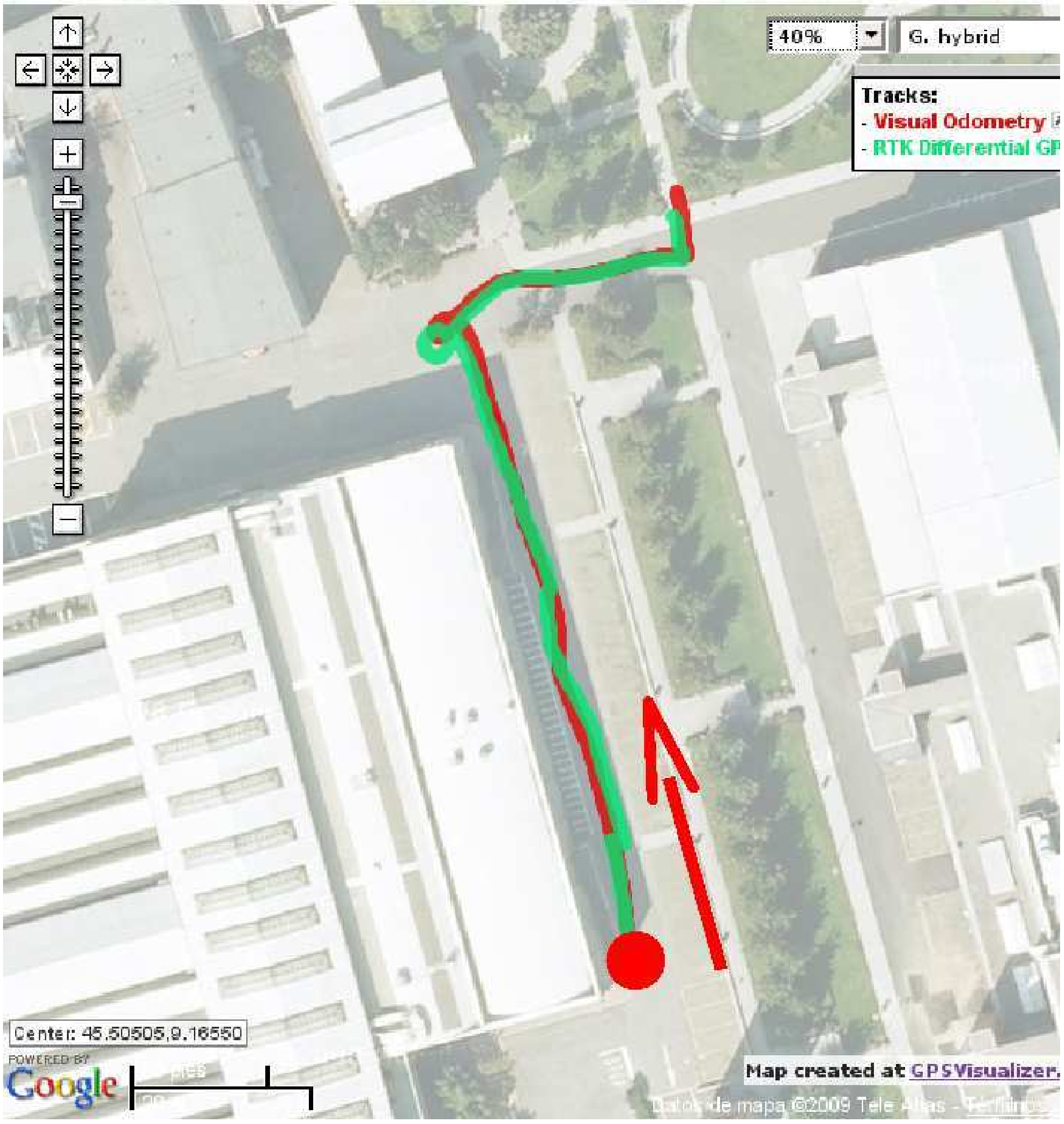


Figure 7: Estimation results in a 125 metres trajectory over a Google maps top view. Estimated trajectory is shown in red; GPS data is plotted in green. Mean error of the estimated trajectory is 2.0 metres. Red dot shows the beginning of the trajectory, and red arrow shows its initial direction.

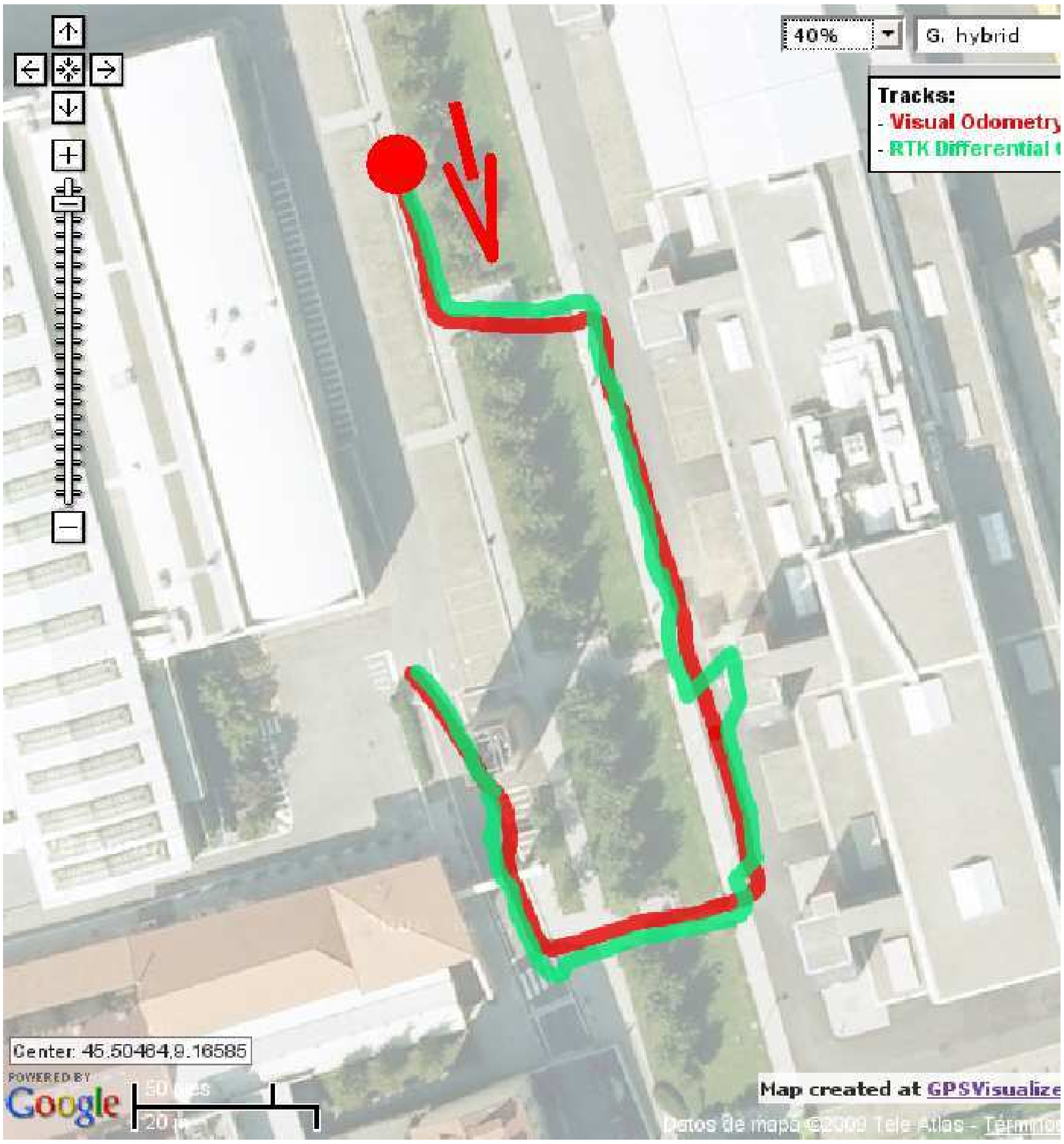


Figure 8: Estimation results in a 185 metres trajectory over a Google maps top view. Estimated trajectory is shown in red; GPS data is plotted in green. Mean error of the estimated trajectory is 2.1 metres. Red dot shows the beginning of the trajectory, and red arrow shows its initial direction.

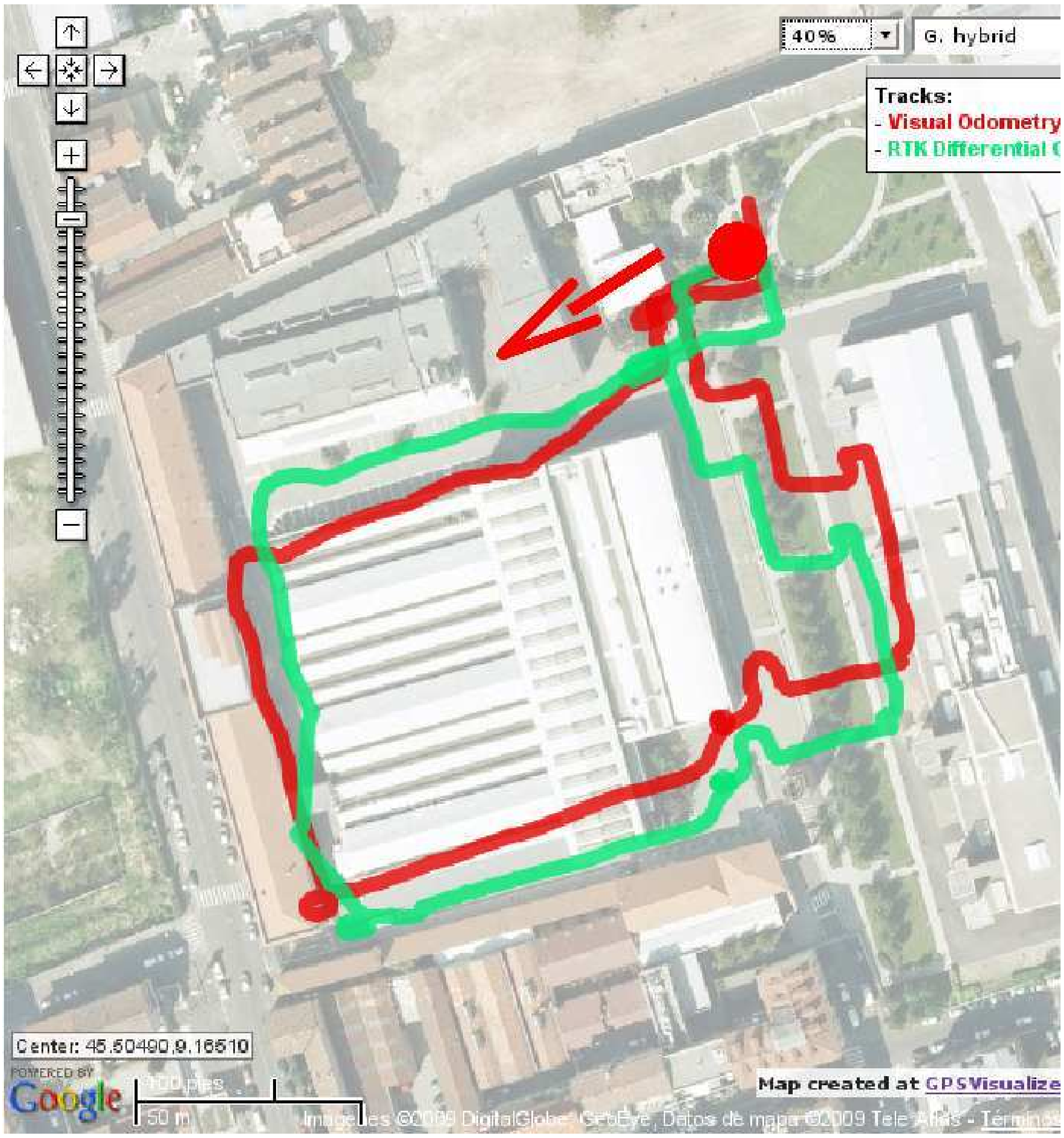


Figure 9: Estimation results in a 630 metres trajectory over a Google maps top view. Estimated trajectory is shown in red; GPS data is plotted in green. Mean error of the estimated trajectory is 13.2 metres. Red dot shows the beginning of the trajectory, and red arrow shows its initial direction.

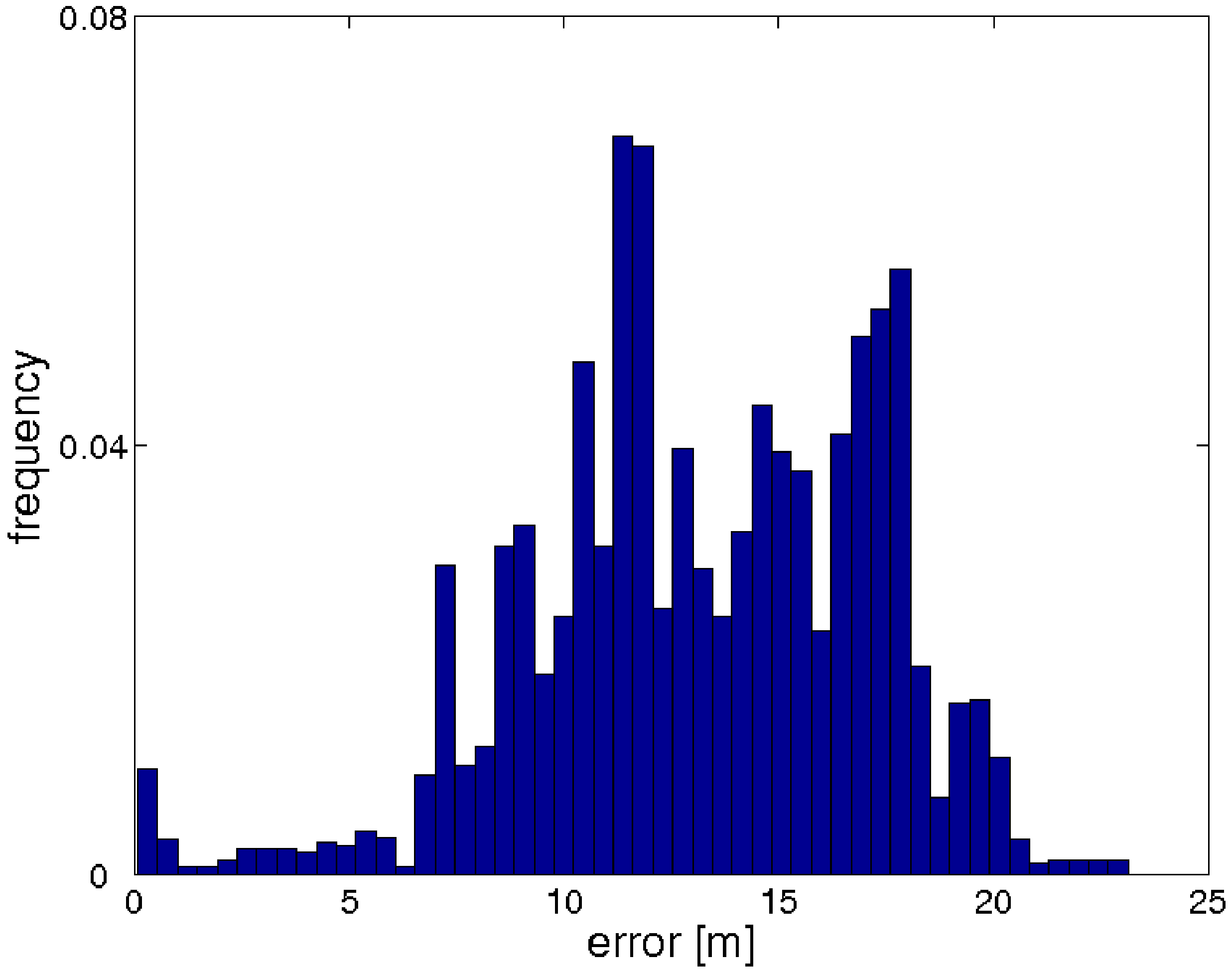


Figure 10: Histogram of the visual odometry errors for the 630 metres sequence.

map of the environment. In the following we present a state-of-the-art approach which is able to construct maps of large scale environments. We validated the approach on some datasets acquired in house. The results of our approach are shown in Section 5.2. For the final deliverable of WP5, we plan to run the approach on the RAWSEEDS datasets and to construct a set of benchmark solutions using the metric described in Section 4.2.

5.1 Stereo Camera SLAM using Conditionally Independent Local Maps

In this section we describe our stereo-SLAM system which integrates a set of novel technologies which allow us to gather most of the information available in the stereo features.

We consider information from features both close and far from the cameras. Stereo provides 3D information from nearby scene points, and each camera can also provide bearing only information from distant scene points. Both types of information are incorporated into the map and used to improve the estimation of both the camera pose and velocity, as well as the map. Nearby scene points provide scale information through the stereo baseline, eliminating the intrinsic scale unobservability problem of monocular systems.

Furthermore our system is able to operate in large environments by decomposing the whole map in local-maps of limited size. We use Conditionally Independent SLAM [33], that allows the system to maintain 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. Using the CI-Graph algorithm we can extend the properties of the CI-submaps to more complex robot trajectories and map topologies.

Feature Detection and Representation A stereo camera can provide depth estimation of points up to a certain distance determined by the baseline between left and right cameras. Therefore, we differentiate two regions: a region close to the cameras and visible by both, in which stereo behaves as a range and bearing sensor. The second is the region of features far from the cameras or seen by only one, in which the stereo becomes a monocular camera, only providing bearing measurements of such points. To take advantage of both types of information, we combine 3D points and inverse depth points (*ID*) (see [6]) in the state vector in order to build a map and estimate the camera trajectory. Despite its properties, each inverse depth point needs an over-parametrization of six values instead of a simpler three coordinates spatial representation. This produces a computational overhead in the EKF. Working with a stereo camera, which can estimate the depth of points close to the camera, raises the subtle question of when a feature should be initialized using a $3D$ or an *ID* representation.

The system produces sequences of local maps of limited size containing both types of features using an EKF SLAM algorithm. Most recent submapping techniques are based on building local maps of limited size that are statistically *independent* [37, 38, 20, 31]. This requirement imposes important constraints to the submaps structure. Valuable information present in a submap cannot be used to improve other submap estimates since, otherwise, the independence property could not be preserved. In addition, same environment features observed in different maps have independent estimations in each map.

CI-SLAM The *Conditionally Independent* SLAM algorithm works with maps that are not statistically independent, but rather *conditionally independent* [33], and thus allow to share the valuable information with no increment in computational cost or loss of precision whatsoever. In Visual SLAM it is very useful to share some state vector components between consecutive submaps: some camera states, such as linear and angular velocities, as well as features that are in the transition region between adjacent submaps and are currently being tracked. This allows us to improve the estimate of relative location between the submaps and continue tracking the observed features with no interruptions. At the same time, CI-submaps inherit the computational efficiency of submapping techniques that, taking into account a subgroup of the map elements, can work with covariance/information submatrices of limited size. The details of this algorithm are explained in [32, 33].

5.2 Performances of Stereo-Camera SLAM

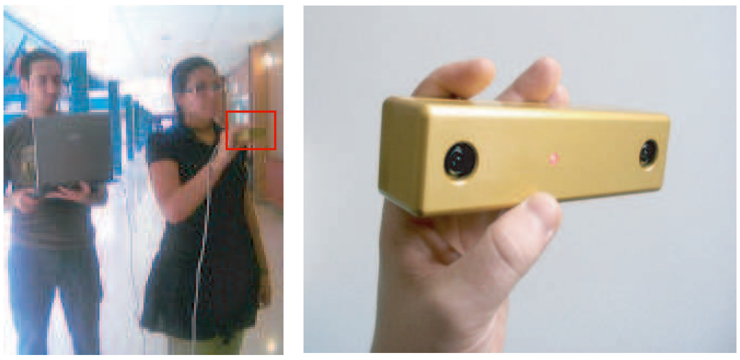


Figure 11: Stereo vision system used to acquire the image sequences. Picture on the left shows the experimental setup during the data acquisition for the indoor experiment.

In this section we describe a preliminary validation of the system performed on datasets acquired in-house with the following sensor setup.

The hardware system consists of a stereo camera carried in hand and a laptop to record and process a sequence of images (see fig. 11). Since the camera moves in 6DOF, we define the camera state using 12 variables: camera position in 3D cartesian coordinates, camera orientation in Euler angles, and linear and angular velocities.

We tested the SLAM algorithm with two 320×240 image sequences obtained with the Point Grey Bumblebee stereo system (see fig. 11). The system provides a 65×50 degree field of view per camera, has a baseline of $12cm$, limiting the 3D point features initialization up to a distance close to $5m$.

An indoor loop (at 48 fps) and an urban outdoor (at 25 fps) loop sequences were captured carrying the camera in hand, at normal walking speeds of $4 - 5km/hour$. Both sequences were processed in MATLAB with the proposed algorithms on a desktop computer with an Intel 4 processor at 2,4GHz.

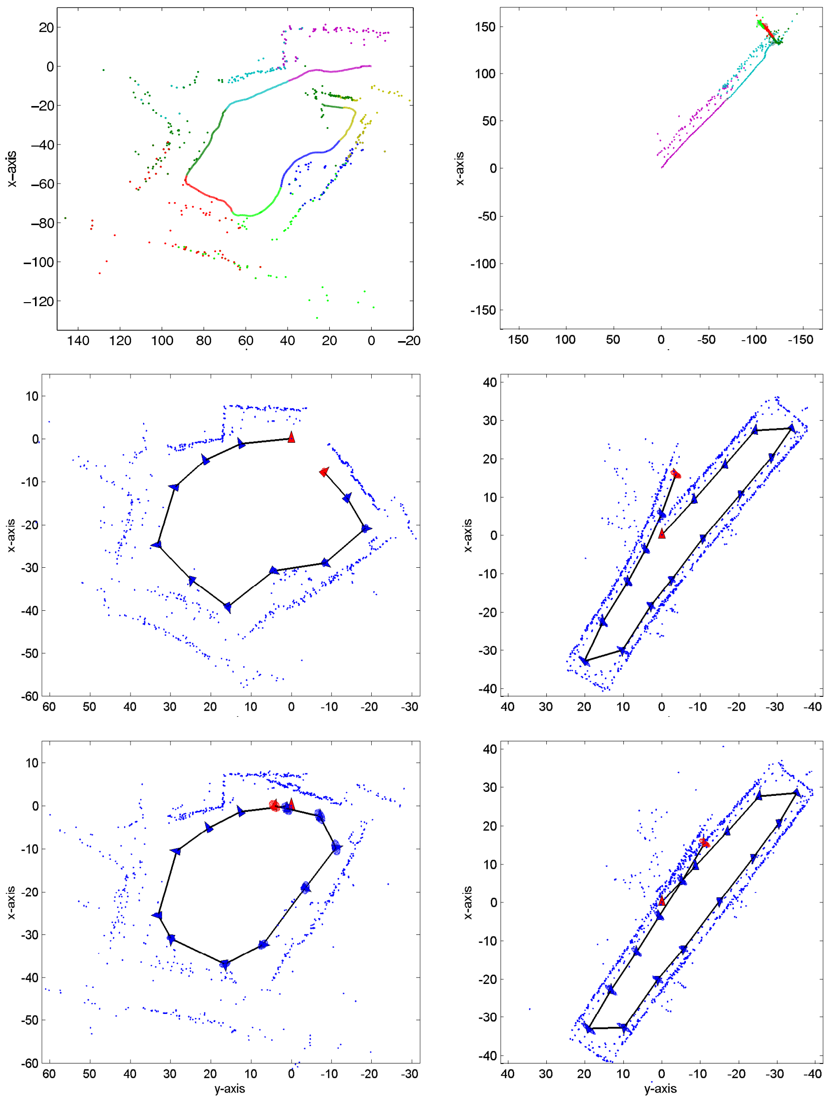
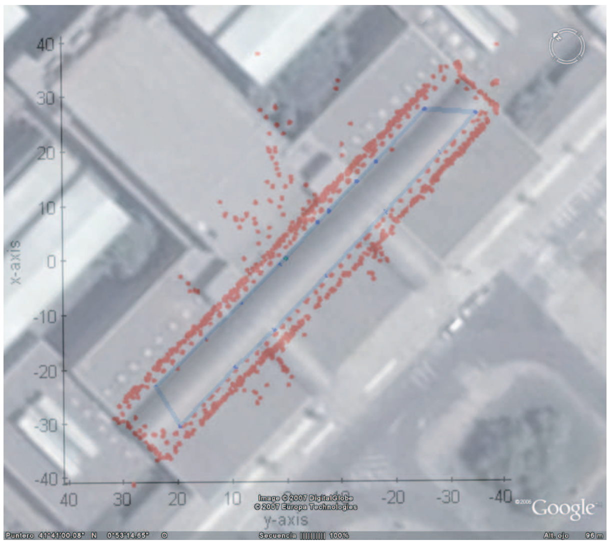


Figure 12: Outdoors experiment: 6DOF stereo SLAM on a public square (left). Indoor experiment along a building environment (right). The sequence of CI Local maps is represented with respect to the initial reference (top); results obtained after running the D&C algorithm that joins and corrects the estimates (middle); final map obtained when the loop closing constraint is imposed (bottom). The scale factor and camera positions are well recovered thanks to the combined observations of 3D points and inverse depth points.



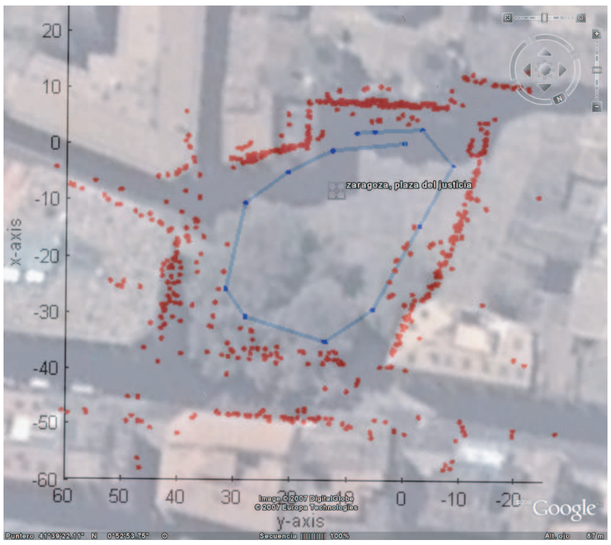

x-axis
y-axis
20
10
0
-10
-20
-30
-40
-50
-60
60
50
40
30
20
10
0
-10
-20
zaragoza, plaza del justicia
Image © 2007 DigitalGlobe
© 2007 Europa Technologies
Google

Figure 13: Stereo visual SLAM recovers the true scale: the building environment (top) and the Public square (bottom) overlapping Google Earth.

The outdoor sequence is composed of 3441 stereo pairs gathered in a public square of our home town (see fig. 12, left). The full trajectory is approximately $140m$ long from the initial camera position. Figure 12 left column, shows the sequence of conditional independent local maps obtained with the technique described above. Each map contains 100 features combining inverse depth and 3D points. The total number of maps built during the stereo sequence is 11. The result of CI SLAM without applying the loop closing constraint is shown in fig. 12, left, middle. As it can be observed, the precision of the map obtained is good enough to almost align the first and last submaps after all the trajectory has been traversed, even without applying loop closing constraints. Fig. 12, left, bottom, presents the final result after closing the loop.

The second experiment was carried out inside one of our campus buildings in a walk of approximately $210m$ (see fig. 12, right). The same process was run in order to obtain a full map from 8135 stereo pairs. This environment has a particular degree of difficulty due to ambiguous texture and the presence of extensive zones of glass windows such as offices, corridors and cafeterias. This can be noticed in the long distance points estimated in some of the maps, which are actually inside offices and the cafeteria (see fig.12, right, top). The result of CI SLAM is shown in fig. 12 right, middle, and the final result after loop closing is shown in fig. 12, right, bottom.

Using the Google Earth tool we can see that the map scale obtained and the trajectory followed by the camera is very close to the real scale. Fig. 13 illustrates comparative results. We loaded the MATLAB figure in Google Earth and set the scale parameter to the real scale.

In the final version of this deliverable, we will provide benchmark solutions based on the RAWSEEDS datasets.

6 Benchmark Solutions for Trinocular SLAM

In this section we provide a benchmark solution for SLAM systems which operate on trinocular camera images and odometry. The benchmark solution presented are based on a system which uses 3D segments as features. The map is estimated by a hierarchical approach which combines a set of local maps (esimtated using EKF) into a global map by means of graph-optimization.

6.1 Trinocular SLAM with 3D segments

3D segments turn out to be quite a stable primitive, at the image and also at the scene level. On the other hand, 3D points are computed in the order of thousands per 3D segment, so they are not a synthetic measure of a human-built scene. The presented work proves that visual SLAM is possible and reliable, when based on 3D segments from stereo. The work extends the known hierarchical SLAM algorithm to deal with 3D segments from stereo with a full 6DoF observer pose; the latter, beside being obvious for outdoor conditions, is more appropriate for properly handling the small imperfections present in indoor conditions.

Extraction of 3D Segments Our system computes the extrema of each segment as a 3D position. The uncertainty in the perception of each extrema is assumed to be Gaussian, which is

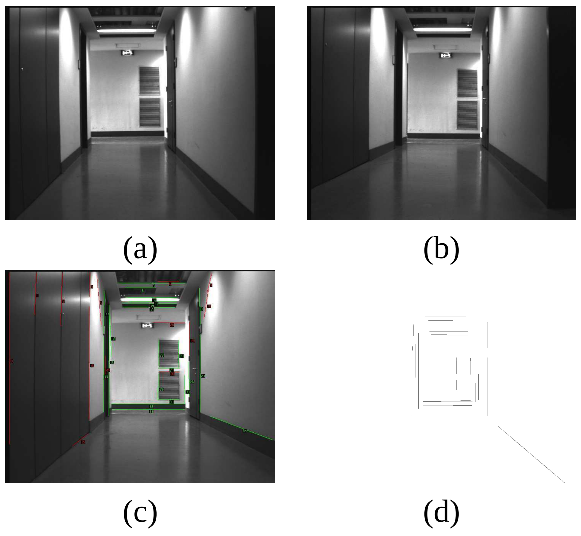


Figure 14: Images captured from: (a): Camera left. (b): Camera top. (c): Camera right (with stereo matched segments). (d): 3D segments extraced by the sensor.

computed by applying the Jacobians of the trinocular projection operation [39], [40], [1], [13], [2], [3] to the noise affecting the pixels in the image. The three cameras are calibrated with respect to a common reference system, located on the robot. We use a DLT technique to determine the three projection matrices. Figure 14 shows a typical result of our segment extraction routine.

Hierarchical SLAM with 3D segments In our system, we use 3D data from the perception system mentioned above. The pose of the each extrema of a segment is modeled as a 3D point. With this information we can approach the full DoF SLAM problem to build a 3D map and we do not constrain the robot to move on the plane.

The estimation technique used in our system is the popular Hierarchical-SLAM approach proposed by Neira *et al.* [12]. The main idea is to subdivide the whole map in two levels: a local level and a global level. The local level consists in a set of submaps. Each of these submaps is estimated using an Extended Kalman Filter. To reduce the complexity of the approach, the individual maps are treated as independent. The global level is represented by a graph where each node describes one submap and the edges represent the existing relations between them (i.e., the relative position and its uncertainty). At any point in time a global map can be recovered by graph optimization. These two levels are the two abstraction level through which it is possible to observe the world.

This hierarchical decomposition of the problem, is shown to limit the influence of linearization errors in the EKF process. Furthermore, it reduces the computational cost of the whole procedure. See attached paper for a more detailed description.

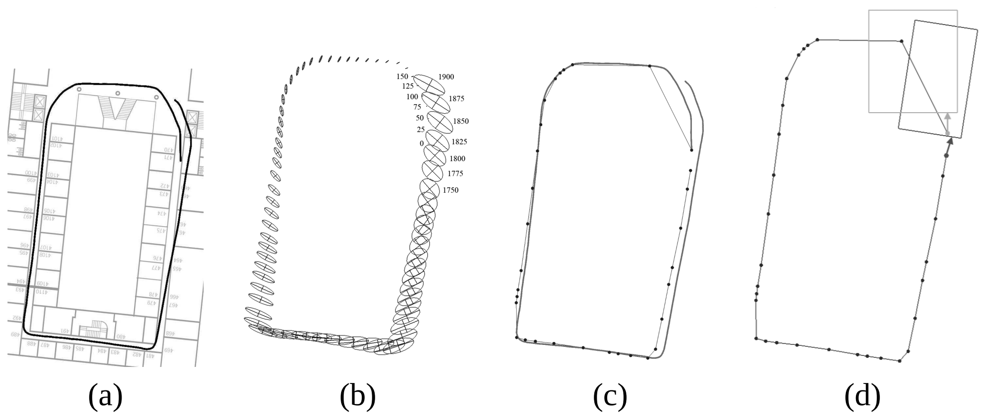


Figure 15: Odometry superimposed to the map of the environment (a), uncertainty in odometry ($\pm 3\sigma$), with view_id (b), odometry (full) superimposed to the base references of the submaps depicted as circles connected by lines (c), bounding boxes of last (dark) and first (bright) submaps (d), and the base references of the submaps after graph relaxation (e).

6.2 Performances of Trinocular SLAM

We validated our approach by processing a medium-scale indoors dataset acquired at 4th floor of building U7, Univ. Milano - Bicocca, Milano, Italy. We used a Robuter mobile robot from Robosoft and a trinocular system which delivers grab three 704x558 gray-scale images simultaneously. We recorded an trinocular image every time the robot moved for approximately 5 cm. The overall length of the trajectory is approximatively 200 m. The results of our approach are shown in Figures 16 and 17.

In the final version of this deliverable we will evaluate the method on multi-sensorial RAWSEEDS datasets, and we will compare the results with laser based methods using the metric described in Section 2.

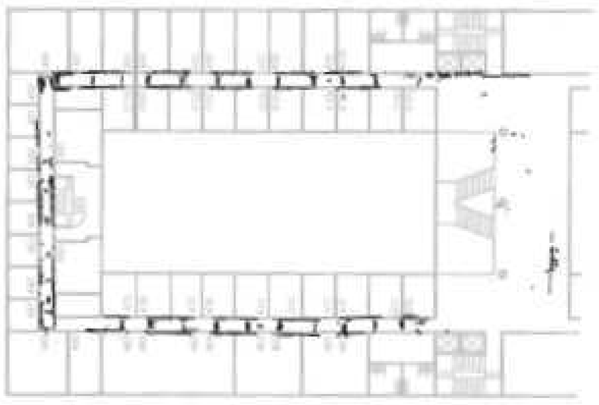


Figure 16: Birds-eye view of the 6DoF-pose final reconstruction after graph relaxation, superimposed to planimetry.

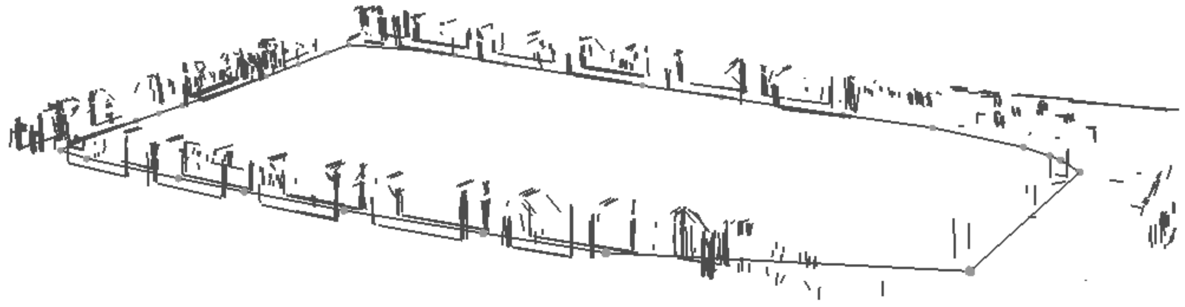


Figure 17: A 3D view of the 6DoF-pose final reconstruction; the solid-circle line is the same as in Figure 15e.

7 Conclusion

This deliverable D5.1 “Preliminary 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 a set of different datasets. We furthermore carried out the evaluations for vision-based SLAM systems. We provided algorithms for monocular, stereo, and trinocular SLAM.

For the final deliverable (D5.1) of WP5, we will carry out the evaluations on all RAWSEEDS datasets and will furthermore present results of our semantic place labeling techniques as well as on multi-robot exploration according to the Annex I.

References

[1] N. Ayache. *Artificial Vision for Mobile Robots*. The MIT Press, 1991.

[2] N. Ayache and F. Lustman. Trinocular stereo vision for robotics. *IEEE Trans. on PAMI*, 12(1), 1991.

[3] N. J. Ayache and O. D. Faugeras. Maintaining representations of the environment of a mobile robot. *IEEE Trans. on R&A*, 5(6):804–819, 1989.

[4] 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.

[5] M. Chli and A.J. Davison. Active matching. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2008.

[6] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth parametrization for monocular SLAM. *IEEE Trans. on Robotics*, 24(5), October 2008.

[7] J. Civera, A. J. Davison, and J. M. M. Montiel. Ekf-based sequential bayesian visual odometry for a monocular camera. Technical Report report, Dept. Informatica e Ing. Sistemas, Universyty of Zaragoza, 2009.

[8] L. A. Clemente, A. J. Davison, I. D. Reid, J. Neira, and J. D. Tardos. Mapping large loops with a single hand-held camera. In *Proceedings of Robotics: Science and Systems*, 2007.

[9] 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.

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

[11] 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 (IJ-CAI)*, pages 1135–1142, Acapulco, Mexico, 2003.

[12] C. Estrada, J. Neira, and J. D. Tardós. Hierarchical slam: real-time accurate mapping of large environments. *IEEE Trans. on Robotics*. to appear.

[13] O. D. Faugeras. *Three Dimensional Computer Vision - A geometric viewpoint*. The MIT Press, 1993.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] S. Huang, Z. Wang, and G. Dissanayake. Sparse Local Submap Joining Filters for building large-scale maps. *Accepted for publication in IEEE Transactions on Robotics*, 2008.

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

[22] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner. Slam benchmarking webpage. http://ais.informatik.uni-freiburg.de/slamevaluation, 2009.

[23] 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.

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

[25] 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.

[26] 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.

[27] 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.

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

[29] 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.

[30] L. M. Paz, P. Piniés, J.D. Tardós, and J. Neira. Large-scale 6-dof slam with stereo-in-hand. *IEEE Trans. on Robotics*, 24(5):946–957, October 2008.

[31] L. M. Paz, J. D. Tardós, and J. Neira. Divide and conquer: EKF SLAM in O(n). *IEEE Trans. on Robotics*, 24(5), October 2008.

[32] 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.

[33] 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.

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

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

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

[37] J. D. Tardós, J. Neira, P. Newman, and J. Leonard. Robust mapping and localization in indoor environments using sonar data. *Int. J. Robotics Research*, 21(4):311–330, 2002.

[38] S. B. Williams, G. Dissanayake, and H. Durrant-Whyte. An efficient approach to the simultaneous localisation and mapping problem. In *IEEE Int. Conf. on Robotics and Automation, ICRA*, volume 1, pages 406–411, Washington DC, 2002.

[39] M. Yachida. 3-d data acquisition by multiple views. In O. D. Faugeras and G. Giralt, editors, *Robotics Research: Third Int. Symp.*, pages 11–18, 1986.

[40] M. Yachida, Y. Kitamura, and M. Kimachi. Trinocular vision: New approach for correspondence problem. In *Proc. IEEE ICPR*, pages 1041–1044, Oct 1986.

8 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.

Analyzing Gaussian Proposal Distributions for Mapping with Rao-Blackwellized Particle Filters

Cyrill Stachniss* Giorgio Grisetti* Wolfram Burgard* Nicholas Roy†

Abstract— **Particle filters are a frequently used filtering technique in the robotics community. They have been successfully applied to problems such as localization, mapping, or tracking. The particle filter framework allows the designer to freely choose the proposal distribution which is used to obtain the next generation of particles in estimating dynamical processes. This choice greatly influences the performance of the filter. Many approaches have achieved good performance through informed proposals which explicitly take into account the current observation. A popular approach is to approximate the desired proposal distribution by a Gaussian. This paper presents a statistical analysis of the quality of such Gaussian approximations. We also propose a way to obtain the optimal proposal in a non-parametric way and then identify the error introduced by the Gaussian approximation. Furthermore, we present an alternative sampling strategy that better deals with situations in which the target distribution is multi-modal. Experimental results indicate that our alternative sampling strategy leads to accurate maps more frequently that the Gaussian approach while requiring only minimal additional computational overhead.**

I. Introduction

Particle filters are a frequently used technique in robotics for dynamical system estimation. They have been used to localize robots [4], to build both feature-maps [12], [13] and grid-maps [7], [8], [9], and to track objects based on vision data [10]. 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. For example, in the context of localizing a robot, the odometry motion model is a good choice for the proposal in that it can be easily sampled and then easily transformed into the target distribution by such techniques as weighted importance sampling. 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.

Murphy, Doucet, and colleagues [6], [14] introduced factored particle filters, known as "Rao-Blackwellization", as an effective means to solve the simultaneous localization and mapping (SLAM) problem. By applying this factorization, several efficient mapping algorithms have been presented [7], [8], [9], [12] and we can note that all of these algorithms have used Gaussians to obtain the next generation of particles.

In this paper, we analyze how well such Gaussian proposal distributions approximate the optimal proposal in the context of mapping. We apply well-founded statistical measures to carry out the comparisons. To the best of our knowledge, this question has not been addressed in the context of particle filter applications in robotics so far. 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 has the same complexity as the Gaussian approximation but can appropriately capture distributions with multiple modes, resulting in more robust mapping systems.

This paper is organized as follows. After a discussion of related approaches, we briefly introduce in Section III the ideas of mapping with Rao-Blackwellized filters. In Section IV, we explain how to actually represent and sample from the optimal proposal. We then present an efficient variant that allows us to deal with multi-modal proposals in an efficient way. In Section VI, we introduce the statistical tests that are used in the experimental section for evaluation.

II. Related Work

Particle filters have been applied to various kinds of robotic state estimation problems such as localization [4], mapping [7], [8], [9], [12], visual tracking [10], or data association problems [20]. Murphy, Doucet, and colleagues were the first that presented an approach based on a Rao-Blackwellized particle filter that learns grid maps [6], [14]. The first efficient approach for mapping with Rao-Blackwellized particle filters was the FastSLAM algorithm by Montemerlo *et al.* [13]. It uses a set of Kalman filters to represent the map features conditioned on a sampled robot pose. A Gaussian process model is used to sample the odometry motion model and generate the proposal distribution on the next step. The grid-based variant presented by Haehnel *et al.* [9] performs scan-matching as a preprocessing step. In this way, they are able to draw samples from Gaussians with lower variances compared to proposals computed based on the odometry only. This reduces the number of required particles and allows a robot to maintain a map estimate online. In contrast to that, Eliazar *et al.* [7] focus on an efficient grid map representation which allows the particles

*University of Freiburg, Department of Computer Science, D-79110 Freiburg. †MIT, 77 Massachusetts Ave., Cambridge, MA 02139-4307

to share a map. Subsequently, Montemerlo *et al.* published FastSLAM2 [12] that uses an informed proposal based on the most recent sensor observation to restrict the space for sampling. Again, to efficiently draw the next generation of particles, the distribution is assumed to be Gaussian. Grisetti *et al.* [8] extended FastSLAM2 to deal with large-scale occupancy grid maps. This technique combines scan-matching on a per particle basis with informed Gaussian proposal distributions.

To the best of our knowledge, there exists no evaluation of how well the Gaussian proposal distributions approximate the optimal proposal which in general is non-Gaussian in the context of mapping. There exist approaches that show that the uncertainty of certain SLAM techniques monotonically decreases over time. For example, Newman proved this property for the relative map filter and also showed that "in the limit, as the number of observations increases, the relative map becomes perfectly known" [15]. In the context of particle filters for SLAM, Montemerlo *et al.* [12] showed that FastSLAM2 "converges [...] for a restricted class of linear Gaussian problems". It, however, makes no statement about the validity of Gaussian approximations in real world settings.

III. Learning Maps with Rao-Blackwellized Particle Filters

A particle filter 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

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

where $z_{1:t}$ is the observation sequence and $u_{1:t-1}$ the odometry information. In practice, the first term of Eq. (1) 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

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

According to Doucet [5], the distribution

$$p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1}) = \frac{p(z_t \mid m_{t-1}^{(i)}, x_t) p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})} \tag{3}$$

is the optimal proposal for particle i with respect to the *variance of the particle weights* that satisfies Eq. (2). This proposal minimizes the degeneracy of the algorithm (Proposition 4 in [5]). As a result, the computation of the weights turn into

$$\begin{aligned} w_t^{(i)} &= w_{t-1}^{(i)} \frac{\eta p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{p(x_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, z_t, u_{t-1})} && (4) \\ &\propto w_{t-1}^{(i)} \frac{p(z_t \mid m_{t-1}^{(i)}, x_t^{(i)}) p(x_t^{(i)} \mid x_{t-1}^{(i)}, u_{t-1})}{\frac{p(z_t \mid m_{t-1}^{(i)}, x_t) p(x_t \mid x_{t-1}^{(i)}, u_{t-1})}{p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1})}} && (5) \\ &= w_{t-1}^{(i)} \cdot p(z_t \mid m_{t-1}^{(i)}, x_{t-1}^{(i)}, u_{t-1}) && (6) \\ &= w_{t-1}^{(i)} \cdot \int p(z_t \mid x') p(x' \mid x_{t-1}^{(i)}, u_{t-1}) \, dx'. && (7) \end{aligned}$$

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 show in this paper, the Gaussian assumption is not always justified. To provide examples for this statement, we first compute the optimal proposal explicitly and then compare it to the Gaussian approximation. Using the optimal proposal in a mapping system leads to computationally expensive operations which are explained in the next section in more detail.

IV. Computing and Sampling from the Optimal Proposal

This section explains how to compute the optimal proposal and how to sample from that distribution. In mapping as well as in many other problems, there is no closed form solution available but we can arrive at a high-fidelity numerical solution for the likelihood function. In our case, the numerator of Eq. (3) is the product of the observation likelihood and the odometry motion model. When using laser range finders, the dominating factor is the observation likelihood. To point-wise evaluate the observation likelihood, we use the so called "beam endpoint model" [19]. In this model, the individual beams within a scan are considered to be independent. Furthermore, the likelihood of a beam is computed based on the distance between the endpoint of the beam and the closest obstacle from that point. Using this point-wise evaluation of the observation likelihood, we can compute a three-dimensional histogram providing the observation likelihood for the different poses.

The second term in Eq. (3) is the robot motion model. In this paper, we consider the "banana-shaped" distribution known from most approaches to Monte-Carlo localization [4]. The likelihood for the individual poses is computed point-wise and is stored in a histogram. This histogram describes the likelihood function in a non-parametric form. Histograms, however, are affected by discretization errors. To smooth this effect, we furthermore apply the Parzen window/kernel estimator [1] based on the evaluated data points. Let x^j be the evaluated poses, then this estimator is defined as

$$\hat{p}(x) \quad = \quad \frac{p(x^j)}{h} \sum_{j=1}^{n} K\left(\frac{x - x^j}{h}\right) \tag{8}$$

where h is called Parzen window. We chose the kernel $K(u)$ as

$$K(u) \quad = \quad \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right). \tag{9}$$

This technique allows us to smooth the histogram data and in this way avoid the discontinuities which are inherent in the histogram representation itself. Furthermore, we can make the likelihood of the smoothed histogram arbitrarily close to the optimal distribution of Eq. (3) by increasing the resolution of the local grid map and reducing the size of the histogram bins.

Given this non-parametric estimator, we can perform rejection sampling to draw the next generation of particles. Obviously, this results in a highly inefficient mapping system with respect to the computation time. However, it allows us to sample from an arbitrarily close approximation to the optimal proposal distribution and to compare it to its Gaussian approximation.

As we will illustrate in the experiments, in most cases the proposal can be safely approximated by a Gaussian. This explains why existing methods based on this particular approximation have been so successful. In certain situations, however, the distribution is highly non-Gaussian and often multi-modal so that the Gaussian does not properly approximate the true distribution which in turn can lead to the divergence of the filter. To overcome this problem, we present an alternative sampling method in the following section. This sampling strategy is able to handle multiple modes in the likelihood functions used as the proposal distribution. Note that our approach does not require any significant computational overhead compared to existing mapping systems that apply scan-matching in combination with a Gaussian proposal [8].

V. Efficient Mapping with Multi-Modal Proposal Distributions

In this section, we present our alternative sampling strategy that can handle multiple modes in the distributions while at the same time keeping the efficiency of a Gaussian proposal distribution. 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. Note that an open source implementation of our mapping system using this technique is available online [18].

Our previous method [8] 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.

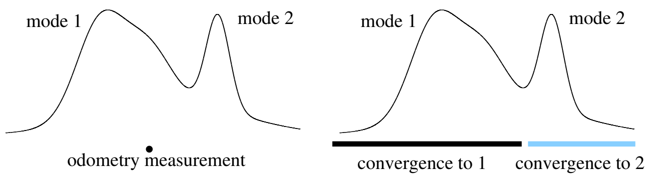


Fig. 1. The left image illustrates a 1D likelihood function and an odometry measurement. Conventional informed sampling first performs scan-matching starting from the odometry measurement. In this situation, the scan-matcher will find a local peak in the likelihood function (most likely mode 1) and the 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.

One of the key ideas of our 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 known from standard Monte-Carlo localization approaches (c.f. [4]) 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. (3). 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 will demonstrate in the experimental results, this technique leads to proposal distributions which are closer to the optimal proposal given in Eq. (3) than the Gaussian approximations; when the distribution has only a single mode, the solution is equivalent to previous approaches [8].

VI. Statistical Tests

To analyze how close the Gaussian proposal as well as our new proposal are to the optimal proposal distribution, we make use of three statistical measures. First, we apply the Anderson-Darling test on normality [2]. This test is reported to be one of the most powerful tests in statistics for detecting most departures from normality. This test is superior to the Kolmogorov-Smirnov test and has a similar performance than the Shapiro-Wilk test [16]. Second, we use the Kullback-Leibler divergence [11] to measure the distance between distributions. Third, we make use of a measure taken from the Cramér-von-Mises test [3], [21] to identify differences between distribution.

Given a set of n samples $\{y^1 < \ldots < y^n\}$ in ascending order of magnitude, the Anderson-Darling (AD) test computes the A statistic as

$$A = -n - \sum_{k=1}^{n} \frac{2k-1}{n} \left[\ln F(y^k) + \ln(1 - F(y^{n+1-k}))\right], \quad (10)$$

where F is the cumulated density function (CDF) of the distribution that is assumed to have generated the samples. In our case, F is the CDF of the normal distribution.

To determine if the samples are generated by a Gaussian or not, one needs to test if

$$A \cdot \left(1 + \frac{0.75}{n} + \frac{2.25}{n^2}\right) \quad \leq \quad c, \quad (11)$$

where c is the Anderson-Darling test value for normal distributions corresponding to a desired level of significance. For example, for a 95% confidence test of normality, the corresponding c is 0.752.

This test allows us to check if the optimal proposal is in fact a Gaussian distribution. An interesting property of the AD test is that it also provides a confidence level for its result. To apply this test, we only need to draw a sample set from the optimal proposal and compute Eq. (10) and Eq. (11). Performing this test for all proposals generated during a mapping experiment provides a measure of how often a sample set is generated from a wrong distribution.

Besides the Anderson-Darling test, we apply the Kullback-Leibler divergence (KLD) which is a frequently used technique to measure the distance between two arbitrary distributions. This allows us to also compare our proposal given in the previous section to the optimal proposal distribution. A KLD value of zero indicates that the distributions are equal and the higher the KLD, the bigger is the difference between them. The KLD between p and f is defined as

$$KLD(p, f) \quad = \quad \int p(x) \cdot \log\left(\frac{p(x)}{f(x)}\right) dx. \quad (12)$$

The KLD takes into account a quotient between two distributions. This can give a high weight to differences in the tails of the distributions (see Eq. (12), where $f(x)$ is small).

An alternative measure for comparison is used in the Cramér-von-Mises test [3], [21]. It measures the disparity of two distributions by taking into account their cumulative density functions (CDF). Since it does not use a quotient as the KLD does, it gives less weight to the tails of the distribution. It computes the integral over the squared distances between the CDFs. Let p and f be the distributions to compare and P and F the corresponding CDFs. Then,

$$d(p, f) \quad = \quad \int [P(x) - F(x)]^2 \, dP(x) \quad (13)$$

provides a measure about the similarity of both distributions which is zero if both are equal.

The three techniques presented here are used in our experiments to identify the differences between the individual proposals and to illustrate potential weaknesses of the Gaussian proposals.

TABLE I

PROPOSAL DISTRIBUTIONS WHICH ARE REGARDED AS GAUSSIANS ACCORDING TO THE ANDERSON-DARLING TEST (95% CONFIDENCE).

Dataset	Gaussian proposal	Non-Gauss (unimodal)	Multi-modal proposal
Intel Research Lab	89.2%	7.2%	3.6%
FHW Museum	84.5%	10.4%	5.1%
Belgioioso	84.0%	10.4%	5.6%
MIT CSAIL	78.1%	15.9%	6.0%
MIT Killian Court	75.1%	19.1%	5.8%
Freiburg Bldg. 79	74.0%	19.4%	6.6%

VII. EXPERIMENTS

The experiments presented in this paper are all based on real world data. We furthermore used freely available datasets to perform our analysis. The learned maps and the datasets used here are available online [17].

A. *Quality of Gaussian Proposals*

In the first experiment, we carried out the Anderson-Darling (AD) test with a confidence of 95% to determine if the optimal proposal can be considered as Gaussian. The results of the test are described in Table I. As can be seen, depending on the dataset, in the optimal proposal was non-Gaussian in 10% to 26% of all cases.

By visually inspecting the datasets and resulting maps, we observed two different scenarios in which non-Gaussian situations occurred. Firstly, we often observed non-Gaussian observation likelihood functions in highly cluttered environments where small changes in the position led to substantial changes of the likelihood. Multi-modal distributions are likely to occur and Gaussians are not well suited to serve as a proposal in these cases. Secondly, non-Gaussian proposals occurred when the robot was moving in environments with long corridors, a fact that surprised us. At first sight, this may appear counterintuitive since corridors are well-structured environments. However, in positions where the robot cannot observe the end of the corridor with its sensor, the likelihood along the main axis of the corridor is almost constant which is highly non-Gaussian and can lead to a negative result of the AD test. One example is MIT Killian Court, consisting mainly of long corridors. Note that even if the AD test fails in such situations, Gaussians can be still good proposals.

In addition to testing acceptance as a Gaussian distribution, we analyzed the distance between the optimal proposal and its Gaussian approximation based on the KLD and the measure from the Cramér-von-Mises test (which is referred to as CvM in the remainder of this paper). Figure 2 plots the frequencies of the individual KLD and CvM values for the Intel and FHW datasets. As can be seen, the approximation error was small (values close to zero) in 94% to 97% of

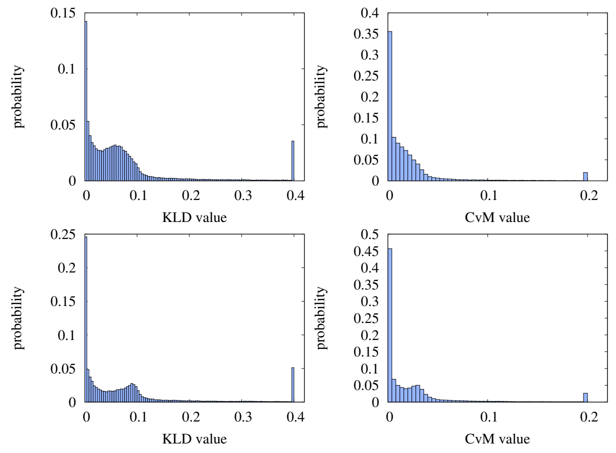


Fig. 2. Difference between the optimal proposal and the Gaussian approximation based on the Intel Research Lab (first row) and the FHW dataset (second row). The images on the left depict the frequencies of the individual Kullback-Leibler divergence values and the images on the right show the frequencies of the distance measure based on the Cramér-von-Mises test (see Eq. (13)). The right-most bin contains also all values larger or equal 0.4 (KLD) and 0.2 (CvM).

all cases. In all other cases, however, the distributions were substantially different. This fact is represented by the peak in the right-most bin of the histograms which contains all values larger or equal than 0.4 (KLD) and 0.2 (CvM). This peak corresponds to situations with multi-modal distributions which can only be badly approximated by a Gaussian. Note that similar results were obtained for the other datasets (see first row of Figure 3).

B. Multi-Modal Proposal Distribution

In the next experiment, we evaluated the alternative sampling strategy proposed in this paper. We used the KLD to compare our new proposal to the optimal proposal distribution. To actually perform the comparison, we computed all modes of the distribution explicitly, which is not required in the mapping system itself as described in Section V. To do so, we drew a set of samples and performed a gradient ascent in the likelihood function to find the individual modes. The modes were then approximated by Gaussians according to the sampled points.

The results of the comparison are shown in Figure 3 for different datasets. The plots in the first row show the KLD distance between the optimal proposal and its Gaussian approximation. The plots in the second row depict the corresponding comparison of our new proposal to the optimal one.

As can be seen, we obtained distributions that no longer approximated a significant fraction of the proposal distributions with large error (i.e., the right-most bin of the distance histograms). In contrast to this, the Gaussian approach approximates the optimal proposal inappropriately in 3% to 6% of all cases. The comparisons using the CvM value showed similar results and are omitted due to reasons of space.

Approaches using the Gaussian proposal have shown to build highly accurate maps of most datasets (compare the experiments in [8]) but there exist situations in which such techniques are likely to fail. This is especially the case if the dominant mode in the likelihood function is not the correct one. Such a situation occurs, for example, in the CSAIL dataset [17] recorded at MIT. Our expectation is that modeling multiple modes in the proposal distribution leads to more robust filters. We carried out 10 experiments with different random seeds and evaluated the success rate of the approach using the Gaussian proposal and our new method. Using the Gaussian approximation for the proposal distribution, the final map had the correct topology (all loops closed, etc.) in only 20% of trials whereas our new approach generated a correct map every time. Figure 4 shows example maps using the Gaussian proposal (left) and our new approach (right).

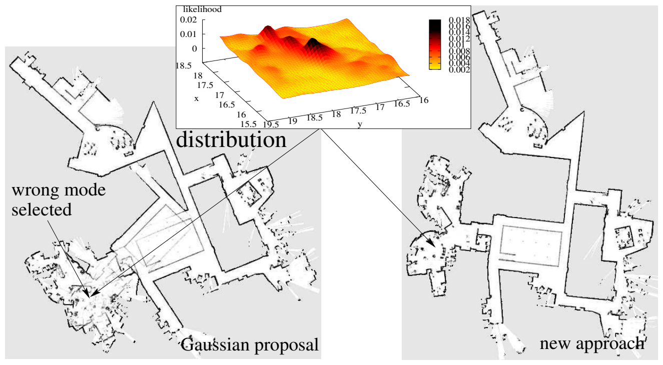


Fig. 4. Resulting map of the MIT CSAIL dataset using a Gaussian proposal (left) and our new approach (right). The Gaussian approach fails due to highly non-Gaussian likelihood functions in the cluttered room (illustrated for a given orientation θ in the top image). Trajectory length: 385m, recording time: 7 min, average speed: 0.9m/s.

TABLE II
EXECUTION TIME ON A 2.8 GHZ PC WITH A P4 SINGLE CORE CPU.

Dataset	N	Execution time		
		optimal	[8]	new method
MIT Killian Court	80	155 h	112 min	113 min
Freiburg Bldg. 79	30	84 h	62 min	62 min
Intel Research Lab	30	40 h	29 min	29 min
FHW Museum	30	38 h	27 min	27 min
Belgioioso	30	18 h	13 min	13 min
MIT CSAIL	30	10 h	7 min	7 min

C. Runtime

In principle, it is possible to avoid Gaussian approximations in the proposal distribution. The main disadvantage when sampling from the optimal proposal is the high computational overhead. To illustrate this overhead, Table II shows the execution time for the individual approaches as well as the number of samples used (N). As can be seen, sampling from the optimal proposal is not suitable for practical applications since it took up to one week to correct a single dataset. In contrast to this, the computational overhead of our new approach is negligible. It allows a robot to learn an accurate map online while moving through the environment.

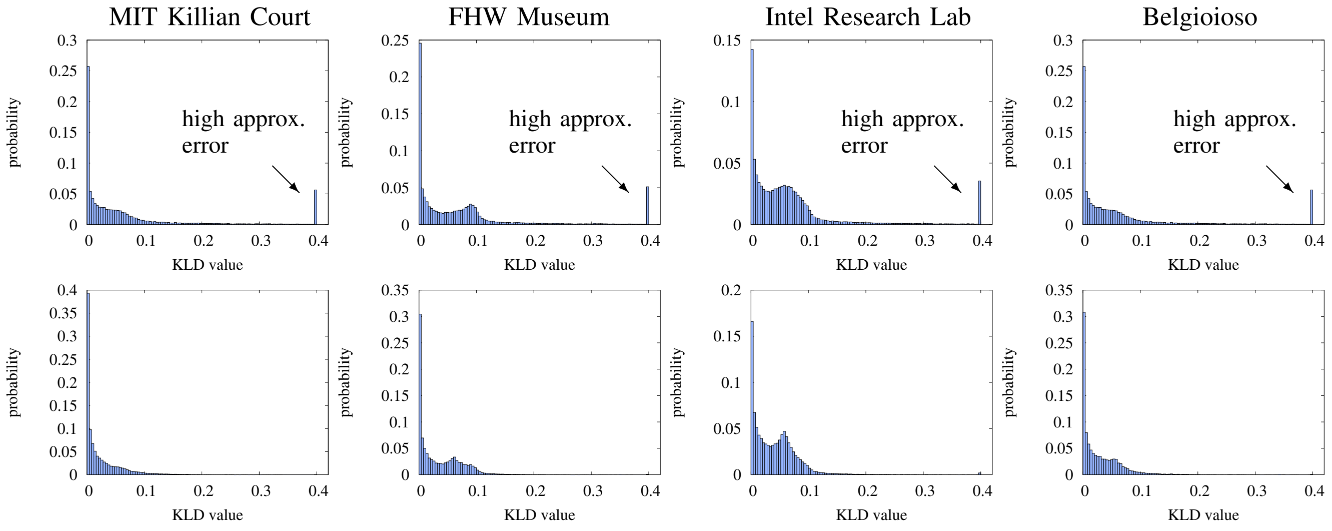


Fig. 3. The plots in the first row show the KLD between optimal proposal and its Gaussian approximation for different datasets. The plots in the second row depict the corresponding KLD between the optimal proposal and the proposal proposed in this paper. The right-most bin contains also all values larger or equal to 0.4. The right-most bin illustrates the mayor drawback of the Gaussian approximation since it described the situations in which the optimal proposal is highly non-Gaussian (e.g., multi-modal). Our new approach, however, can better deal with such situations.

VIII. Conclusion

In this paper, we analyzed how well Gaussian proposal distributions approximate the optimal proposal in the context of the application of Rao-Blackwellized particle filters to the simultaneous localization and mapping problem. We demonstrated that in around 5% of all cases, the Gaussian approximation is not sufficient to model the likelihood function. As such situations are one of the sources for the divergence of the filter, we presented an alternative sampling technique that is able to deal with multi-modal distributions while maintaining the same efficiency as the Gaussian proposal. This resulted in a more robust approach to mapping with Rao-Blackwellized particle filters. In experiments carried out with real data, we showed the efficiency and robustness of our approach.

Acknowledgment

This work has partly been supported by the DFG under contract number SFB/TR-8, by the EC under contract number FP6-IST-34120-muFly (action line: 2.5.2.: micro/nano based subsystems) and FP6-2005-IST-6-RAWSEEDS, and by the NSF under CAREER grant 0546467. Thanks to Dirk Hähnel for providing the Intel and the Belgioioso dataset as well as to Mike Bosse for the Killian Court dataset.

References

[1] E. Alaydin. *Introduction to Machine Learning*, chapter Nonparametric Density Estimation, pages 157–161. MIT Press, 2004.

[2] T. W. Anderson and D. A. Darling. Asymptotic theory of certain goodness-of-fit criteria based on stochastic processes. *Annals of Mathematical Statistics*, 23:193–212, 1952.

[3] H. Cramér. On the composition of elementary errors. ii: statistical applications. *Skandinavisk Aktuarietidskrift*, 11:141–180, 1928.

[4] 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.

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

[6] A. Doucet, J.F.G. de Freitas, K. Murphy, and S. Russel. Rao-Blackwellized partcile filtering for dynamic bayesian networks. In *Proc. of the Conf. on Uncertainty in Artificial Intelligence (UAI)*, 2000.

[7] 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, 2003.

[8] 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.

[9] 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 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 206–211, 2003.

[10] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *Proc. of the European Conference on Computer Vision*, pages 343–356, 1996.

[11] S. Kullback and R. A Leibler. On information and sufficiency. *Annals of Mathematical Statistics*, 22:79–86, 1951.

[12] 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, 2003.

[13] 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)*, 2002.

[14] K. Murphy. Bayesian map learning in dynamic environments. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, pages 1015–1021, Denver, CO, USA, 1999.

[15] P.M. Newman. *On the structure and solution of the simultaneous localization and mapping problem.* PhD thesis, University of Sydney, Australia, 1999.

[16] S. S. Shapiro and M. B. Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52:591–611, 1965.

[17] C. Stachniss. Robotic datasets. http://www.informatik.uni-freiburg.de/$\sim$stachnis/datasets, 2007.

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

[19] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*, chapter Robot Perception, pages 171–172. MIT Press, 2005.

[20] G.D. Tipaldi, A. Farinelli, L. Iocchi, and D. Nardi. Heterogeneous feature state estimation with rao-blackwellized particle filters. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[21] R. von Mises. *Wahrscheinlichkeitsrechnung und Ihre Anwendung in der Statistik und Theoretischen Physik.* Deuticke, Leipzig, Germany, 1931. In German.

A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent

Giorgio Grisetti Cyrill Stachniss Slawomir Grzonka Wolfram Burgard
University of Freiburg, Department of Computer Science, 79110 Freiburg, Germany

Abstract— **In 2006, Olson *et al.* presented a novel approach to address the graph-based simultaneous localization and mapping problem by applying stochastic gradient descent to minimize the error introduced by constraints. Together with multi-level relaxation, this is one of the most robust and efficient maximum likelihood techniques published so far. In this paper, we present an extension of Olson's algorithm. It applies a novel parameterization of the nodes in the graph that significantly improves the performance and enables us to cope with arbitrary network topologies. The latter allows us to bound the complexity of the algorithm to the size of the mapped area and not to the length of the trajectory as it is the case with both previous approaches. We implemented our technique and compared it to multi-level relaxation and Olson's algorithm. As we demonstrate in simulated and in real world experiments, our approach converges faster than the other approaches and yields accurate maps of the environment.**

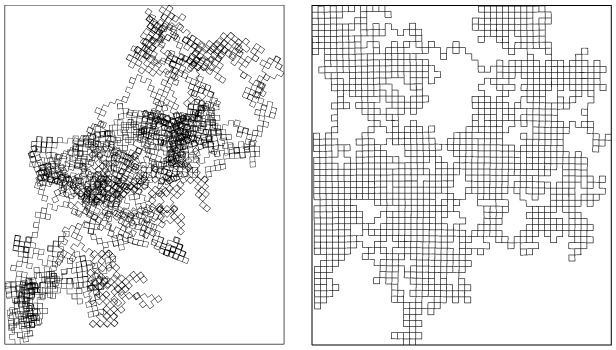


Fig. 1. The left image shows an uncorrected network with around 100k poses and 450k constraints. The right image depicts the network after applying our error minimization approach (100 iterations, 17s on a P4 CPU with 1.8GHz).

I. Introduction

Models of the environment are needed for a wide range of robotic applications, including search and rescue, automated vacuum cleaning, and many others. Learning maps has therefore been a major research focus in the robotics community over the last decades. Learning maps under uncertainty is often referred to as the simultaneous localization and mapping (SLAM) problem. In the literature, a large variety of solutions to this problem can be found. The approaches mainly differ due to the underlying estimation technique such as extended Kalman filters, information filters, particle filters, or least-square error minimization techniques.

In this paper, we consider the so-called "graph-based" or "network-based" formulation of the SLAM problem in which the poses of the robot are modeled by nodes in a graph [2, 5, 6, 7, 11, 13]. Constraints between poses resulting from observations or from odometry are encoded in the edges between the nodes.

The goal of an algorithm designed to solve this problem is to find the configuration of the nodes that maximizes the observation likelihood encoded in the constraints. Often, one refers to the negative observation likelihood as the error or the energy in the network. An alternative view to the problem is given by the spring-mass model in physics. In this view, the nodes are regarded as masses and the constraints as springs connected to the masses. The minimal energy configuration of the springs and masses describes a solution to the mapping problem. Figure 1 depicts such a constraint network as a motivating example.

A popular solution to this class of problems are iterative approaches. They can be used to either correct all poses simultaneously [6, 9, 11] or to locally update parts of the network [2, 5, 7, 13]. Depending on the used technique, different parts of the network are updated in each iteration. The strategy for defining and performing these local updates has a significant impact on the convergence speed.

Our approach uses a tree structure to define and efficiently update local regions in each iteration. The poses of the individual nodes are represented in an incremental fashion which allows the algorithm to automatically update successor nodes. Our approach extends Olson's algorithm [13] and converges significantly faster to a network configuration with a low error. Additionally, we are able to bound the complexity to the size of the environment and not to the length of the trajectory.

The remainder of this paper is organized as follows. After discussing the related work, Section III explains the graph-based formulation of the mapping problem. Subsequently, we explain the usage of stochastic gradient descent to find network configurations with small errors. Section V introduces our tree parameterization and in Section VI we explain how to obtain such a parameterization tree from robot data. We finally present our experimental results in Section VII.

II. Related Work

Mapping techniques for mobile robots can be classified according to the underlying estimation technique. The most popular approaches are extended Kalman filters (EKFs), sparse extended information filters, particle filters, and least square error minimization approaches. The effectiveness of the EKF approaches comes from the fact that they estimate a fully correlated posterior about landmark maps and robot poses [10, 14]. Their weakness lies in the strong assumptions that have to be made on both, the robot motion model and the sensor noise. Moreover, the landmarks are assumed to be uniquely identifiable. There exist techniques [12] to deal with unknown data association in the SLAM context, however, if certain assumptions are violated the filter is likely to diverge [8].

Frese's TreeMap algorithm [4] can be applied to compute nonlinear map estimates. It relies on a strong topological assumption on the map to perform sparsification of the information matrix. This approximation ignores small entries in the information matrix. In this way, Frese is able to perform an update in $\mathcal{O}(\log n)$ where n is the number of features.

An alternative approach is to find maximum likelihood maps by least square error minimization. The idea is to compute a network of relations given the sequence of sensor readings. These relations represent the spatial constraints between the poses of the robot. In this paper, we also follow this way of formulating the SLAM problem. Lu and Milios [11] first applied this approach in robotics to address the SLAM problem using a kind of brute force method. Their approach seeks to optimize the whole network at once. Gutmann and Konolige [6] proposed an effective way for constructing such a network and for detecting loop closures while running an incremental estimation algorithm. Howard *et al.* [7] apply relaxation to localize the robot and build a map. Duckett *et al.* [2] propose the usage of Gauss-Seidel relaxation to minimize the error in the network of constraints. In order to make the problem linear, they assume knowledge about the orientation of the robot. Frese *et al.* [5] propose a variant of Gauss-Seidel relaxation called multi-level relaxation (MLR). It applies relaxation at different resolutions. MLR is reported to provide very good results and is probably the best relaxation technique in the SLAM context at the moment.

Note that such maximum likelihood techniques as well as our method focus on computing the best map and assume that the data association is given. The ATLAS framework [1] or hierarchical SLAM [3], for example, can be used to obtain such data associations (constraints). They also apply a global optimization procedure to compute a consistent map. One can replace such optimization procedures by our algorithm and in this way make ATLAS or hierarchical SLAM more efficient.

The approach closest to the work presented here is the work of Olson *et al.* [13]. They apply stochastic gradient descent to reduce the error in the network. They also propose a representation of the nodes which enables the algorithm to perform efficient updates. The approach of Olson *et al.* is one of the current state-of-the-art approaches for optimizing networks of constraints. In contrast to their technique, our approach uses a different parameterization of the nodes in the network that better takes into account the topology of the environment. This results in a faster convergence of our algorithm.

Highly sophisticated optimization techniques such as MLR or Olson's algorithm are restricted to networks that are built in an incremental way. They require as input a sequence of robot poses according to the traveled path. First, this makes it difficult to use these techniques in the context of multi-robot SLAM. Second, the complexity of the algorithm depends on the length of the trajectory traveled by the robot and not on the size of the environment. This dependency prevents to use these approaches in the context of lifelong map learning.

One motivation of our approach is to build a system that depends on the size of the environment and not explicitely on the length of the trajectory. We designed our approach in a way that it can be applied to arbitrary networks. As we will show in the remainder of this paper, the ability to use arbitrary networks allows us to prune the trajectory so that the complexity of our approach depends only on the size of the environment. Furthermore, our approach proposes a more efficient parameterization of the network when applying gradient descent.

III. On Graph-based SLAM

Most 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 [2, 5, 6, 11, 13]. They do not consider to compute the full posterior about the map and the poses of the robot. The approach presented in this paper also belongs to this class of methods.

The goal of graph-based ML mapping algorithms is to find the configuration of the nodes that maximizes the likelihood of the observations. For a more precise formulation consider the following definitions:

- $\mathbf{x} = (x_1 \ \cdots \ x_n)^T$ is a vector of parameters which describes a configuration of the nodes. Note that the parameters x_i do not need to be the absolute poses of the nodes. They are arbitrary variables which can be mapped to the poses of the nodes in real world coordinates.
- δ_{ji} describes a constraint between the nodes j and i. It refers to an observation of node j seen from node i. These constraints are the edges in the graph structure.
- Ω_{ji} is the information matrix modeling the uncertainty of δ_{ji}.
- $f_{ji}(\mathbf{x})$ is a function that computes a zero noise observation according to the current configuration of the nodes j and i. It returns an observation of node j seen from node i.

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

$$e_{ji}(\mathbf{x}) \quad = \quad f_{ji}(\mathbf{x}) - \delta_{ji} \tag{1}$$

as well as the *residual* r_{ji}

$$r_{ji}(\mathbf{x}) \quad = \quad -e_{ji}(\mathbf{x}). \tag{2}$$

Note that at the equilibrium point, e_{ji} is equal to 0 since $f_{ji}(\mathbf{x}) = \delta_{ji}$. In this case, an observation perfectly matches the current configuration of the nodes. Assuming a Gaussian observation error, the negative log likelihood of an observation f_{ji} is

$$\begin{aligned} F_{ji}(\mathbf{x}) &\propto (f_{ji}(\mathbf{x}) - \delta_{ji})^T \, \Omega_{ji} \, (f_{ji}(\mathbf{x}) - \delta_{ji}) && (3) \\ &= e_{ji}(\mathbf{x})^T \Omega_{ji} e_{ji}(\mathbf{x}) && (4) \\ &= r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). && (5) \end{aligned}$$

Under the assumption that the observations are independent, the overall negative log likelihood of a configuration $\mathbf{x}$ is

$$\begin{aligned} F(\mathbf{x}) &= \sum_{\langle j,i \rangle \in \mathcal{C}} F_{ji}(\mathbf{x}) && (6) \\ &= \sum_{\langle j,i \rangle \in \mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). && (7) \end{aligned}$$

Here $\mathcal{C} = \{\langle j_1, i_1 \rangle, \ldots, \langle j_M, i_M \rangle\}$ is 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 maximizes the likelihood of the observations. This can be written as

$$\mathbf{x}^* = \operatorname*{argmin}_{\mathbf{x}} F(\mathbf{x}). \tag{8}$$

IV. Stochastic Gradient Descent for Maximum Likelihood Mapping

Olson *et al.* [13] propose to use a variant of the preconditioned stochastic gradient descent (SGD) to address the SLAM problem. The approach minimizes Eq. (8) 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:

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t + \underbrace{\lambda \cdot \mathbf{H}^{-1} J_{ji}^T \Omega_{ji} r_{ji}}_{\Delta \mathbf{x}_{ji}} \tag{9}$$

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, and r_{ji} is the residual.

Reading the term $\Delta \mathbf{x}_{ji}$ of Eq. (9) from right to left gives an intuition about the sequential procedure used in SGD:

- r_{ji} is the residual which is the opposite of the error vector. Changing the network configuration in the direction of the residual r_{ji} will decrease the error e_{ji}.
- Ω_{ji} represents the information matrix of a constraint. Multiplying it with r_{ji} scales the residual components according to the information encoded in the constraint.
- J_{ji}^T: The role of the Jacobian is to map the residual term into a set of variations in the parameter space.
- $\mathbf{H}$ is the Hessian of the system and it represents the curvature of the error function. This allows us to scale the variations resulting from the Jacobian depending on the curvature of the error surface. We actually use an approximation of $\mathbf{H}$ which is computed as

  $$\mathbf{H} \quad \simeq \quad \sum_{\langle j,i \rangle} J_{ji} \Omega_{ji} J_{ji}^T. \tag{10}$$

  Rather than inverting the full Hessian which is computationally expensive, we approximate it by

  $$\mathbf{H}^{-1} \quad \simeq \quad [\operatorname{diag}(\mathbf{H})]^{-1}. \tag{11}$$

- λ is a learning rate which decreases with the iteration of SGD and which makes the system to converge to an equilibrium point.

In practice, the algorithm decomposes the overall problem into many smaller problems by optimizing the constraints individually. Each time a solution for one of these subproblems is found, the network is updated accordingly. Obviously, updating the different constraints one after each other can have opposite effects on a subset of variables. To avoid infinitive oscillations, one uses the learning rate 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.

This framework allows us to iteratively reduce the error given the network of constraints. The optimization approach, however, leaves open how the nodes are represented (parameterized). Since the parameterization defines also the structure of the Jacobians, it has a strong influence on the performance of the algorithm.

The next section addresses the problem of how to parameterize a graph in order to efficiently carry out the optimization approach.

V. Network Parameterizations

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 bijective mapping g between $\mathbf{p}$ and $\mathbf{x}$ exists

$$\mathbf{x} = g(\mathbf{p}) \qquad\qquad \mathbf{p} = g^{-1}(\mathbf{x}). \tag{12}$$

As previously explained, in each iteration SGD decomposes the problem into a set of subproblems and solves them successively. In this work, a subproblem is defined as the optimization of a single constraint. Different solutions to the individual subproblems can have antagonistic effects when combining them.

The parameterization g defines also the subset of variables that are modified by a single constraint update. A good parameterization defines the subproblems in a way that the combination step leads only to small changes of the individual solutions.

A. Incremental Pose Parameterization

Olson *et al.* propose the so-called incremental pose parameterization. Given a set of node locations p_i and given a fixed order on the nodes, the incremental parameters x_i can be computed as follows

$$x_i \quad = \quad p_i - p_{i-1}. \tag{13}$$

Note that x_i is computed as the difference between two subsequent nodes and not by motion composition. Under this parameterization, the error in the global reference frame (indicated by primed variables) has the following form

$$e'_{ji} \quad = \quad p_j - (p_i \oplus \delta_{ji}) \tag{14}$$

$$\quad = \quad \left(\sum_{k=i+1}^{j} x_k \right) + \underbrace{\left(\prod_{k=1}^{i} \tilde{R}_k \right)}_{R_i} \delta_{ji}, \tag{15}$$

where $\oplus$ is the motion composition operator according to Lu and Milios [11] and $\tilde{R}_k$ the homogenous rotation matrix of the *parameter* x_k. The term R_k is defined as the rotation matrix of the *pose* p_k. The information matrix in the global reference frame can be computed as

$$\Omega'_{ji} \quad = \quad R_i \Omega_{ji} R_i^T. \tag{16}$$

According to Olson *et al.* [13], neglecting the contribution of the angular terms of $x_0, \ldots, x_i$ to the Jacobian results in the following simplified form

$$J'_{ji} = \sum_{k=i+1}^{j} \mathcal{I}_k \quad \text{with} \quad \mathcal{I}_k = (0 \cdots 0 \underbrace{I}_{k} 0 \cdots 0). \quad (17)$$

Here 0 is the 3 by 3 zero matrix and I is the 3 by 3 identity.

Updating the network based on the constraint $\langle j, i \rangle$ with such an Jacobian results in keeping the node i fixed and in distributing the residual along all nodes between j and i.

Olson *et al.* weight the residual proportional to $j - i$ which is the number of nodes involved in the constraint. The parameter x_k of the node k with $k = i+1, \ldots, j$ is updated as follows

$$\Delta x_k \quad = \quad \lambda w_k \Omega'_{ji} r'_{ji}, \quad (18)$$

where the weight w_k is computed as

$$w_k = (j - i) \left[\sum_{m=i+1}^{j} D_m^{-1} \right]^{-1} D_k^{-1}. \quad (19)$$

In Eq. (19), D_k are the matrices containing the diagonal elements of the k^{th} block of the Hessian $\mathbf{H}$. Intuitively, each variable is updated proportional to the uncertainty about that variable. Note that the simple form of the Jacobians allows us to update the parameter vector for each node individually as expressed by Eq. (18).

The approach presented in this section is currently one of the best solutions to ML mapping. However, it has the following drawbacks:

- In practice, the incremental parameterization cannot deal with arbitrarily connected networks. This results from the approximation made in Eq. (17), in which the angular components are ignored when computing the Jacobian. This approximation is only valid if the subsequent nodes in Eq. (13) are spatially close. Furthermore, the way the error is distributed over the network assumes that the nodes are ordered according to poses along the trajectory. This results in adding a large number of nodes to the network whenever the robot travels for a long time in the same region. This requirement prevents an approach from merging multiple nodes into a single one. Merging or pruning nodes, however, is a necessary precondition to allow the robot lifelong map learning.
- When updating a constraint between the nodes j and i, the parameterization requires to change the j-i nodes. As a result, each node is likely to be updated by several constraints. This leads to a high interaction between constraints and will typically reduce the convergence speed of SGD. For example, the node k will be updated by all constraints $\langle j', i' \rangle$ with $i' < k \leq j'$. Note that using an intelligent lookup structure, this operation can be carried out in $\mathcal{O}(\log n)$ time where n is the number of nodes in the network [13]. Therefore, this is a problem of convergence speed of SGD and not a computational problem.

B. Tree Parameterization

Investigating a different parameterization which preserves the advantages of the incremental one but overcomes its drawbacks is the main motivation for our approach. First, our method should be able to deal with arbitrary network topologies. This would enable us to compress the graph whenever robot revisits a place. As a result, the size of the network would be proportional to the visited area and not to the length of the trajectory. Second, the number of nodes in the graph updated by each constraint should mainly depend on the topology of the environment. For example, in case of a loop-closure a large number of nodes need to be updated but in all other situations the update is limited to a small number of nodes in order to keep the interactions between constraints small.

Our idea is to first construct a spanning tree from the (arbitrary) graph. Given such a tree, we define the parameterization for a node as

$$x_i \quad = \quad p_i - p_{\text{parent}(i)}, \quad (20)$$

where $p_{\text{parent}(i)}$ refers to the parent of node i in the spanning tree. As defined in Eq. (20), the tree stores the differences between poses. As a consequence, one needs to process the tree up to the root to compute the actual pose of a node in the global reference frame.

However, to obtain only the difference between two arbitrary nodes, 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 error in the global frame by

$$\begin{aligned} e'_{ji} &= p_j - (p_i \oplus \delta_{ji}) & (21) \\ &= p_j - (p_i + R_i \delta_{ji}) & (22) \\ &= \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} x_{k^{[+]}} - \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} x_{k^{[-]}} - R_i \delta_{ji}. & (23) \end{aligned}$$

Here R_i is the 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 parameterization of the nodes. It can furthermore be used to define an order in which the optimization algorithm can efficiently process the constraints as we will explain in the remainder of this section. For illustration, Figure 2 (a) and (b) depict two graphs and possible parameterization trees.

Similar to Eq. (16), we can express the information matrix associated to a constraint in the global frame by

$$\Omega'_{ji} \quad = \quad R_i \Omega_{ji} R_i^T. \quad (24)$$

As proposed in [13], we neglect the contribution of the rotation matrix R_i in the computation of the Jacobian. This approximation speeds up the computation significantly. Without

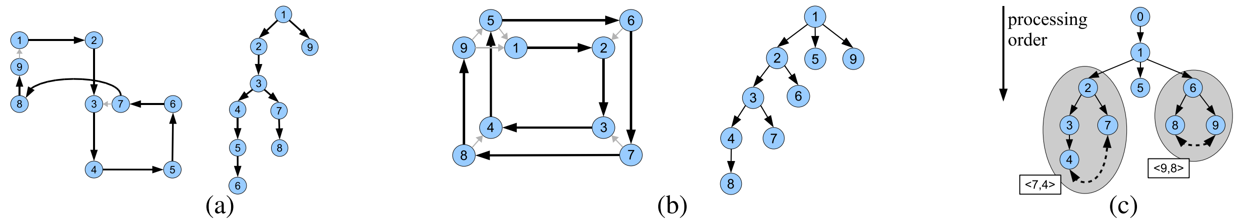


Fig. 2. (a) and (b): Two small example graphs and the trees used to determine the parameterizations. The small grey connections are constraints introduced by observations where black ones result from odometry. (c) Processing the constraints ordered according to the node with the smallest level in the path avoids the recomputation of rotational component of all parents. The same holds for subtrees with different root nodes on the same level.

this approximation the update of a single constraint influences the poses of all nodes up to the root.

The approximation leads to the following Jacobian:

$$J'_{ji} \quad = \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} \mathcal{I}_{k^{[+]}} - \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} \mathcal{I}_{k^{[-]}} \tag{25}$$

Compared to the approach described in the previous section, the number of updated variables per constraint is in practice smaller when using the tree. Our approach updates $|\mathcal{P}_{ji}|$ variables rather than $j - i$. The weights w_k are computed as

$$w_k = |\mathcal{P}_{ji}| \left[\sum_{m \in \mathcal{P}_{ji}}^{j} D_m^{-1} \right]^{-1} D_k^{-1}, \tag{26}$$

where D_k is the k-th diagonal block element of $\mathbf{H}$. This results in the following update rule for the variable x_k

$$\Delta x_k \quad = \quad \lambda w_k \cdot \mathrm{s}(x_k, i, j) \cdot \Omega'_{ji} r'_{ji}, \tag{27}$$

where the value of $\mathrm{s}(x_k, j, i)$ is $+1$ or -1 depending on where the parameter x_k is located on the path $\mathcal{P}_{ji}$:

$$\mathrm{s}(x_k, j, i) \quad = \quad \begin{cases} +1 & \text{if } x_k \in \mathcal{P}_{ji}^{[+]} \\ -1 & \text{if } x_k \in \mathcal{P}_{ji}^{[-]} \end{cases} \tag{28}$$

Our parameterization maintains the simple form of the Jacobians which enables us to perform the update of each parameter variable individually (as can be seen in Eq. (27)). Note that in case one uses a tree that is degenerated to a list, this parameterization is equal to the one proposed by Olson *et al.* [13]. In case of a non-degenerated tree, our approach offers several advantages as we will show in the experimental section of this paper.

The optimization algorithm specifies how to update the nodes but does not specify the order in which to process the constraints. We can use our tree parameterization to sort the constraints which allows us to reduce the computational complexity of our approach.

To compute the residual of a constraint $\langle j, i \rangle$, we need to know the rotational component of the node i. This requires to traverse the tree up to the first node for which the rotational component is known. In the worst case, this is the root of the tree.

Let the *level* of a node be the distance in the tree between the node itself and the root. Let z_{ji} be the node in the path of the constraint $\langle j, i \rangle$ with the smallest level. The level of the constraint is then defined as the level of z_{ji}.

Our parameterization implies that updating a constraint will never change the configuration of a node with a level smaller than the level of the constraint. Based on this knowledge, we can sort the constraints according to their level and process them in that order. As a result, it is sufficient to access the parent of z_{ji} to compute the rotational component of the node i since all nodes with a smaller level than z_{ji} have already been corrected.

Figure 2 (c) illustrates such a situation. The constraint $\langle 7, 4 \rangle$ with the path $4, 3, 2, 7$ does not change any node with a smaller level than the one of node 2. It also does not influence other subtrees on the same level such as the nodes involved in the constraint $\langle 9, 8 \rangle$.

In the following section, we describe how we actually build the tree given the trajectory of a robot or an arbitrary network as input.

VI. Construction of the Spanning Tree

When constructing the parameterization tree, we distinguish two different situations. First, we assume that the input is a sequence of positions belonging to a trajectory traveled by the robot. Second, we explain how to build the tree given an arbitrary graph of relations.

In the first case, the subsequent poses are located closely together and there exist constraints between subsequent poses resulting from odometry or scan-matching. Further constraints between arbitrary nodes result from observations when revisiting a place in the environment. In this setting, we build our parameterization tree as follows:

1) We assign a unique id to each node based on the timestamps and process the nodes accordingly.
2) The first node is the root of the tree (and therefore has no parent).
3) As the parent of a node, we choose the node with the smallest id for which a constraint to the current node exists.

This tree can be easily constructed on the fly. The Figures 2 (a) and (b) illustrates graphs and the corresponding trees. This tree has a series of nice properties when applying our optimization algorithm to find a minimal error configuration of the nodes. These properties are:

- The tree can be constructed incrementally: when adding a new node it is not required to change the existing tree.
- In case the robot moves through nested loops, the interaction between the updates of the nodes belonging to the individual loops depends on the number of nodes the loops have in common.

- When retraversing an already mapped area and adding constraints between new and previously added nodes, the length of the path in the tree between these nodes is small. This means that only a small number of nodes need to be updated.

The second property is illustrated in Figure 2 (a). The two loops in that image are only connected via the constraint between the nodes 3 and 7. They are the only nodes that are updated by constraints of both loops.

The third property is illustrated in Figure 2 (b). Here, the robot revisits a loop. The nodes 1 to 4 are chosen as the parents for all further nodes. This results in short paths in the tree when updating the positions of the nodes while retraversing known areas.

The complexity of the approach presented so far depends on the length of the trajectory and not on the size of the environment. These two quantities are different in case the robot revisits already known areas. This becomes important whenever the robot is deployed in a bounded environment for a long time and has to update its map over time. This is also known as lifelong map learning. Since our parameterization is not restricted to a trajectory of sequential poses, we have the possibility of a further optimization. Whenever the robot revisits a known place, we do not need to add new nodes to the graph. We can assign the current pose of the robot to an already existing node in the graph.

Note that this can be seen as an approximation similar to adding a rigid constraint neglecting the uncertainty of the corresponding observation. However, in case local maps (e.g., grid maps) are used as nodes in the network, it makes sense to use such an approximation since one can localize a robot in an existing map quite accurately.

To also avoid adding new constraints to the network, we can refine an existing constraint between two nodes in case of a new observation. Given a constraint $\delta_{ji}^{(1)}$ between the nodes j and i in the graph and a new constraint $\delta_{ji}^{(2)}$ based on the current observation. Both constraints can be combined to a single constraint which has the following information matrix and mean:

$$\begin{aligned} \Omega_{ji} &= \Omega_{ji}^{(1)} + \Omega_{ji}^{(2)} && (29) \\ \delta_{ji} &= \Omega_{ji}^{-1}(\Omega_{ji}^{(1)} \cdot \delta_{ji}^{(1)} + \Omega_{ji}^{(2)} \cdot \delta_{ji}^{(2)}) && (30) \end{aligned}$$

As a result, the size of the problem does not increase when revisiting known locations. As the experiments illustrate, this node reduction technique leads to an increased convergence speed.

In case the input to our algorithm is an arbitrary graph and no natural order of the nodes is provided, we compute a minimal spanning tree to define the parameterization. Since no additional information (like consecutive poses according to a trajectory) is available, we cannot directly infer which parts of the graph are well suited to form a subtree in the parameterization tree. The minimal spanning tree appears to yield comparable results with respect to the number of iterations needed for convergence in all our experiments.

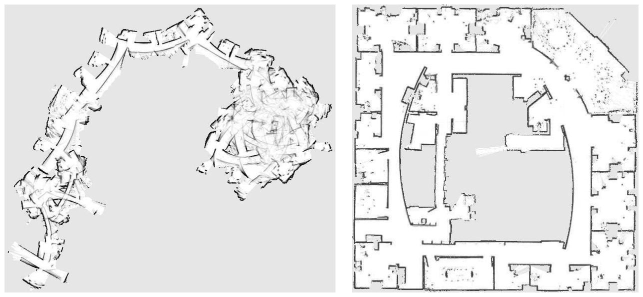


Fig. 3. The map of the Intel Research Lab before (left) and after (right) execution of our algorithm (1000 nodes, runtime <1s).

VII. Experiments

This section is designed to evaluate the properties of our tree parameterization for learning maximum likelihood maps. We first show that such a technique is well suited to generate accurate occupancy grid maps given laser range data and odometry from a real robot. Second, we provide simulation experiments on large-scale datasets. We furthermore provide a comparison between our approach, Olson's algorithm [13], and multi-level relaxation by Frese *et al.* [5]. Finally, we analyze our approach and investigate properties of the tree parameterization in order to explain why we obtain better results then the other methods.

A. *Real World Experiments*

The first experiment is designed to illustrate that our approach can be used to build maps from real robot data. The goal was to build an accurate occupancy grid map given the laser range data obtained by the robot. The nodes of our graph correspond to the individual poses of the robot during data acquisition. The constraints result from odometry and from the pair-wise matching of laser range scans. Figure 3 depicts two maps of the Intel Research Lab in Seattle. The left one is constructed from raw odometry and the right one is the result obtained by our algorithm. As can be seen, the corrected map shows no inconsistencies such as double corridors. Note that this dataset is freely available on the Internet.

B. *Simulated Experiments*

The second set of experiments is designed to measure the performance of our approach quantitatively. Furthermore, we compare our technique to two current state-of-the-art SLAM approaches that work on constraint networks, namely multi-level relaxation by Frese *et al.* [5] and Olson's algorithm [13]. In the experiments, we used the two variants of our method: the one that uses the node reduction technique described in Section VI and the one that maintains all the nodes in the graph.

In our simulation experiments, we moved a virtual robot on a grid world. An observation is generated each time the current position of the robot was close to a previously visited location. We corrupted the observations with a variable amount of noise for testing the robustness of the algorithms. We simulated different datasets resulting in graphs with a number of constraints between around 4,000 and 2 million.

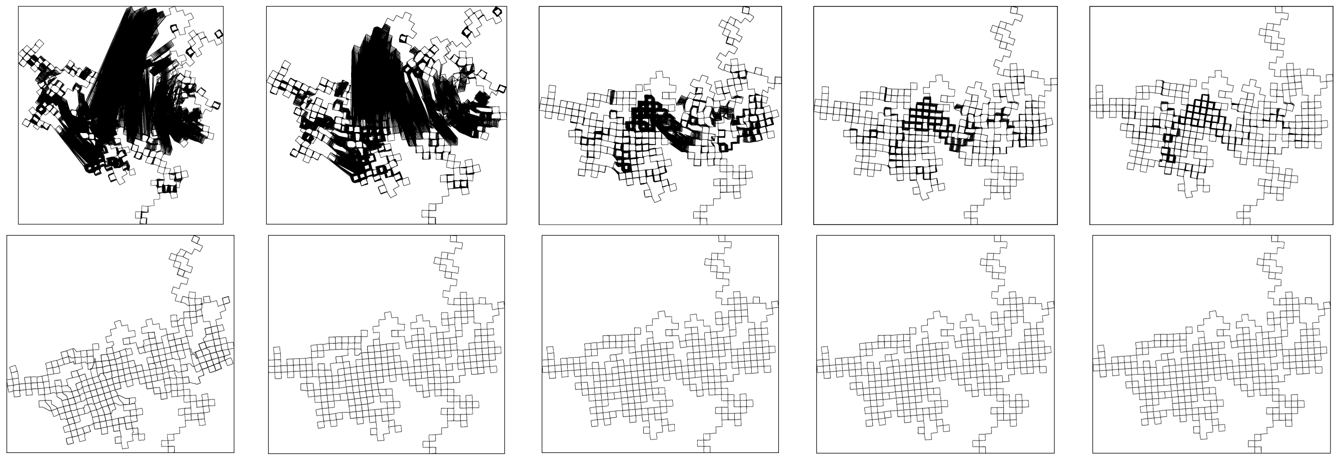


Fig. 4. Results of Olson's algorithm (first row) and our approach (second row) after 1, 10, 50, 100, 300 iterations for a network with 64k constraints. The black areas in the images result from constraints between nodes which are not perfectly corrected after the corresponding iteration (for timings see Figure 6).

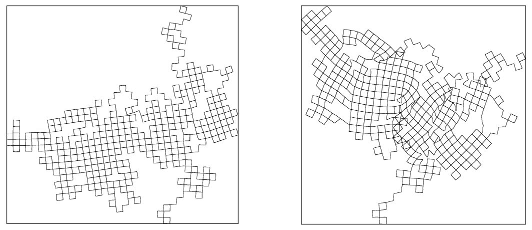


Fig. 5. The result of MLR strongly depends on the initial configuration of the network. Left: small initial pose error, right: large initial pose error.

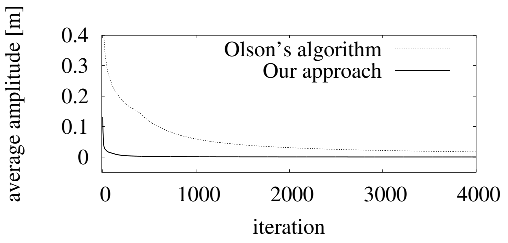


Fig. 7. The average amplitude of the oscillations of the nodes due to the antagonistic effects of different constraints.

Figure 4 depicts a series of graphs obtained by Olson's algorithm and our approach after different iterations. As can be seen, our approach converges faster. Asymptotically, both approaches converge to a similar solution.

In all our experiments, the results of MLR strongly depended on the initial positions of the nodes. In case of a good starting configuration, MLR converges to an accurate solution similar to our approach as shown in Figure 5 (left). Otherwise, it is likely to diverge (right). Olson's approach as well as our technique are more or less independent of the initial poses of the nodes.

To evaluate our technique quantitatively, we first measured the error in the network after each iteration. The left image of Figure 6 depicts a statistical experiments over 10 networks with the same topology but different noise realizations. As can be seen, our approach converges significantly faster than the approach of Olson *et al.* For medium size networks, both approaches converge asymptotically to approximatively the same error value (see middle image). For large networks, the high number of iterations needed for Olson's approach prevented us from showing this convergence experimentally. Due to the sake of brevity, we omitted comparisons to EKF and Gauss Seidel relaxation because Olson *et al.* already showed that their approach outperforms such techniques.

Additionally, we evaluated in Figure 6 (right) the average computation time per iteration of the different approaches. As a result of personal communication with Edwin Olson, we furthermore analyzed a variant of his approach which is restricted to spherical covariances. It yields similar execution times *per iteration* than our approach. However, this restricted variant has still the same converge speed with respect to the number of iterations than Olson's unrestricted technique. As can be seen from that picture, our node reduction technique speeds up the computations up to a factor of 20.

C. Analysis of the Algorithm

The experiments presented above illustrated that our algorithm offers significant improvements compared to both other techniques. The goal of this section is to experimentally point out the reasons for these improvements.

The presented tree parameterization allows us to decompose the optimization of the whole graph into a set of weakly interacting problems. A good measure for evaluating the interaction between the constraints is the average number l of updated nodes per constraint. For example, a network with a large value of l has typically a higher number of interacting constraints compared to networks with low values of l. In all experiments, our approach had a value between 3 and 7. In contrast to that, this values varies between 60 and 17,000 in Olson's approach on the same networks. Note that such a high average path length reduces the convergence speed of Olson's algorithm but does not introduce a higher complexity.

The optimization approach used in this paper as well as in Olson's algorithm updates for each constraint the involved nodes to minimize the error in the network. As a result, different constraints can update poses in an antagonistic way during one iteration. This leads to oscillations in the position

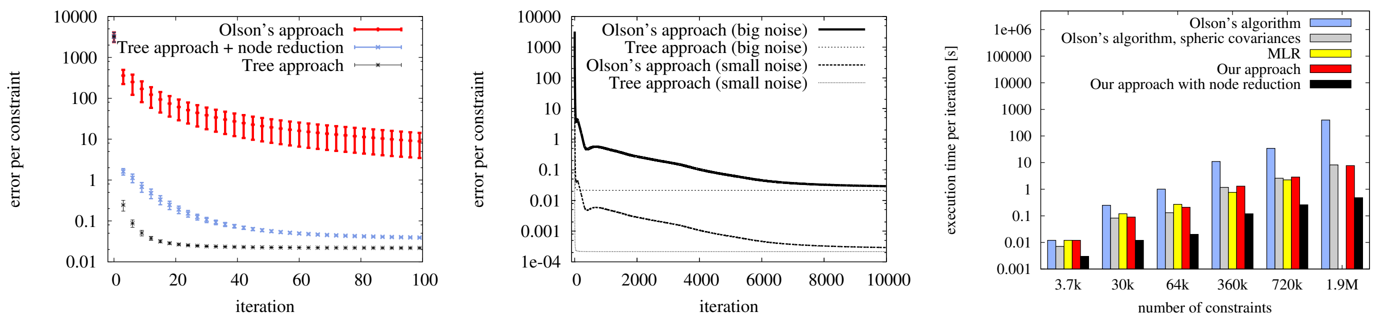


Fig. 6. The left image illustrates shows the error of our and Olson's approach in a statistical experiment ($\sigma = 0.05$ confidence). The image in the middle shows that both techniques converge asymptotically to the same error. The right image shows the average execution time *per iteration* for different networks. For the 1.9M constraints network, the executing of MLR required memory swapping and the result is therefore omitted.

of a node before convergence. Figure 7 illustrates the average amplitude of such an oscillations for Olson's algorithm as well as for our approach. As can be seen, our techniques converges faster to an equilibrium point. This a further reason for the higher convergence speed of our approach.

D. Complexity

Due to the nature of gradient descent, the complexity of our approach per iteration depends linearly on the number of constraints. For each constraint $\langle j, i \rangle$, our approach modifies exactly those nodes which belong to the path $\mathcal{P}_{ji}$ in the tree. Since each constraint has an individual path length, we consider the average path length l. This results in an complexity per iteration of $\mathcal{O}(M \cdot l)$, where M is the number of constraints. In all our experiments, l was approximatively $\log N$, where N is the number of nodes. Note that given our node reduction technique, M as well as N are bounded by the size of the environment and not by the length of the trajectory.

A further advantage of our technique compared to MLR is that it is easy to implement. The function that performs a single iteration requires less than 100 lines of C++ code. An open source implementation, image and video material, and the datasets are available at the authors' web-pages.

VIII. Conclusion

In this paper, 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 parameterization 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.

Acknowledgment

The authors would like to gratefully thank Udo Frese for his insightful comments and for providing us his MLR implementation for comparisons. Further thanks to Edwin Olson for his helpful comments on this paper. This work has partly been supported by the DFG under contract number SFB/TR-8 (A3) and by the EC under contract number FP6-2005-IST-5-muFly and FP6-2005-IST-6-RAWSEEDS.

References

[1] M. Bosse, P.M. Newman, J.J. Leonard, and S. Teller. An ALTAS framework for scalable mapping. In *Proc. of the IEEE Int. Conf. on Robotics & Automation*, pages 1899–1906, Taipei, Taiwan, 2003.

[2] T. Duckett, S. Marsland, and J. Shapiro. Fast, on-line learning of globally consistent maps. *Autonomous Robots*, 12(3):287 – 300, 2002.

[3] C. Estrada, J. Neira, and J.D. Tardós. Hierachical slam: Real-time accurate mapping of large environments. *IEEE Transactions on Robotics*, 21(4):588–596, 2005.

[4] U. Frese. Treemap: An $o(logn)$ algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122, 2006.

[5] 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.

[6] J.-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *Proc. of the IEEE Int. Symp. on Comp. Intelligence in Robotics and Automation*, pages 318–325, Monterey, CA, USA, 1999.

[7] A. Howard, M.J. Matarić, and G. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 1055–1060, 2001.

[8] S. Julier, J. Uhlmann, and H. Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proc. of the American Control Conference*, pages 1628–1632, Seattle, WA, USA, 1995.

[9] J. Ko, B. Stewart, D. Fox, K. Konolige, and B. Limketkai. A practical, decision-theoretic approach to multi-robot mapping and exploration. In *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 3232–3238, Las Vegas, NV, USA, 2003.

[10] J.J. Leonard and H.F. Durrant-Whyte. Mobile robot localization by tracking geometric beacons. *IEEE Transactions on Robotics and Automation*, 7(4):376–382, 1991.

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

[12] J. Neira and J.D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[13] E. Olson, J.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*, pages 2262–2269, 2006.

[14] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial realtionships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer Verlag, 1990.

Online Constraint Network Optimization for Efficient Maximum Likelihood Map Learning

Giorgio Grisetti* Dario Lodi Rizzini‡ Cyrill Stachniss* Edwin Olson† Wolfram Burgard*

Abstract— **In this paper, we address the problem of incrementally optimizing constraint networks for maximum likelihood map learning. Our approach allows a robot to efficiently compute configurations of the network with small errors while the robot moves through the environment. We apply a variant of stochastic gradient descent and use a tree-based parameterization of the nodes in the network. By integrating adaptive learning rates in the parameterization of the network, our algorithm can use previously computed solutions to determine the result of the next optimization run. Additionally, our approach updates only the parts of the network which are affected by the newly incorporated measurements and starts the optimization approach only if the new data reveals inconsistencies with the network constructed so far. These improvements yield an efficient solution for this class of online optimization problems.**

Our approach has been implemented and tested on simulated and on real data. We present comparisons to recently proposed online and offline methods that address the problem of optimizing constraint network. Experiments illustrate that our approach converges faster to a network configuration with small errors than the previous approaches.

I. Introduction

Maps of the environment are needed for a wide range of robotic applications such as search and rescue, automated vacuum cleaning, and many other service robotic tasks. Learning maps has therefore been a major research focus in the robotics community over the last decades. Learning maps under uncertainty is often referred to as the simultaneous localization and mapping (SLAM) problem. In the literature, a large variety of solutions to this problem can be found. The approaches mainly differ in the underlying estimation technique. Typical techniques are Kalman filters, information filters, particle filters, network based methods which rely on least-square error minimization techniques.

Solutions to the SLAM problem can be furthermore divided into online an offline methods. Offline methods are so-called batch algorithms that require all the data to be available right from the beginning [1], [2], [3]. In contrast to that, online methods can re-use an already computed solution and update or refine it. Online methods are needed for situations in which the robot has to make decisions based on the model of the environment during mapping. Exploring an unknown environment, for example, is a task of this category. Popular online SLAM approaches such as [4], [5] are based on the Bayes' filter. Recently, also incremental maximum-likelihood approaches have been presented as an effective alternative [6], [7], [8].

*Department of Computer Science, University of Freiburg, Germany. †MIT, 77 Massachusetts Ave., Cambridge, MA 02139-4307, USA. ‡Department of Information Engineering, University of Parma, Italy.

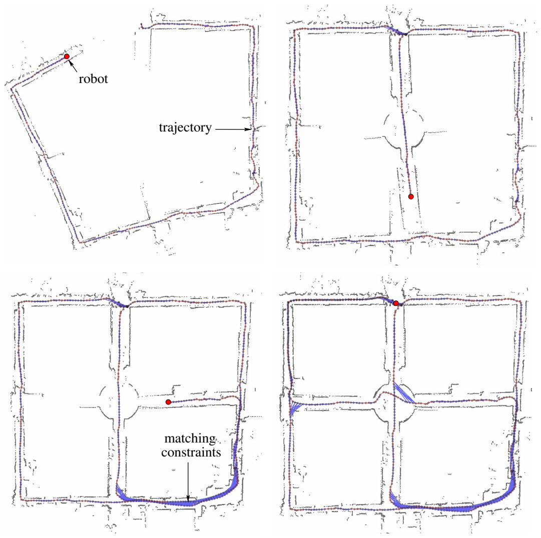


Fig. 1. Four snapshots created while incrementally learning a map.

In this paper, we present an efficient online optimization algorithm which can be used to solve the so-called "graph-based" or "network-based" formulation of the SLAM problem. Here, the poses of the robot are modeled by nodes in a graph and constraints between poses resulting from observations or from odometry are encoded in the edges between the nodes. Our method belongs to the same class of techniques of Olson's algorithm or MLR [8]. It focuses on computing the best map and it assumes that the constraints are given. Techniques like the ATLAS framework [9] or hierarchical SLAM [10], for example, can be used to obtain the necessary data associations (constraints). They also apply a global optimization procedure to compute a consistent map. One can replace these optimization procedures by our algorithm and in this way make them more efficient.

Our approach combines the ideas of adaptive learning rates with a tree-based parameterization of the nodes when applying stochastic gradient descent. This yields an online algorithm that can efficiently compute network configurations with low errors. An application example is shown in Figure 1. It depicts four snapshots of our online approach during a process of building a map from the ACES dataset.

II. Related Work

A large number of mapping approaches has been presented in the past and a variety of different estimation techniques have been used to learn maps. One class of approaches uses constraint networks to represent the relations between poses and observations.

Lu and Milios [1] were the first who used constraint networks to address the SLAM problem. They proposed a brute force method that seeks to optimize the whole network at once. Gutmann and Konolige [11] presented an effective way for constructing such a network and for detecting loop closures while running an incremental estimation algorithm. Frese *et al.* [8] described a variant of Gauss-Seidel relaxation called multi-level relaxation (MLR). It applies relaxation at different resolutions.

Olson *et al.* [2] were the first who applied a variant of stochastic gradient descent to compute solutions to this family of problems. They propose a representation of the nodes which enables the algorithm to perform efficient updates. Our previously presented method [3] introduced the tree parameterization that is also used in this paper. Subsequently, Olson *et al.* [6] presented an online variant of their method using adaptive learning rates. In this paper, we integrate such learning rates into the tree-based parameterization which yields a solution to the online SLAM problem that outperforms the individual methods.

Kaess *el al.* [7] proposed an on-line version of the smoothing and mapping algorithm for maximum likelihood map estimation. This approach relies on a QR factorization of the information matrix and integrates the new measurements as they are available. Using the QR factorization, the poses of the nodes in the network can be efficiently retrieved by back substitution. Additionally they keep the matrices sparse via occasional variable reordering. Frese [12] proposed the Treemap algorithm which is able to perform efficient updates of the estimate by ignoring the weak correlations between distant locations.

The contribution of this paper is an efficient online approach for learning maximum likelihood maps. It integrates adaptive learning rates into a tree-based network optimization technique using a variant of stochastic gradient descent. Our approach presents an efficient way of selecting only the part of the network which is affected by newly incorporated data. Furthermore, it allows to delay the optimization so that the network is only updated if needed.

III. Stochastic Gradient Descent for Maximum Likelihood Mapping

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 [8], [1], [2]. The approach presented in this paper also belongs to this class of methods.

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

$$e_{ji}(\mathbf{x}) \quad = \quad f_{ji}(\mathbf{x}) - \delta_{ji} \tag{1}$$

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

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \sum_{\langle j,i \rangle \in \mathcal{C}} r_{ji}(\mathbf{x})^T \Omega_{ji} r_{ji}(\mathbf{x}). \tag{2}$$

In the remainder of this section we describe how the general framework of stochastic gradient descent can be used for minimizing Eq. (2) and how to construct a parameterization of the network which increases the convergence speed.

A. Network Optimization using Stochastic Gradient Descent

Olson *et al.* [2] 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. (2) 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:

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t + \lambda \cdot \mathbf{H}^{-1} J_{ji}^T \Omega_{ji} r_{ji} \tag{3}$$

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. (3), we refer the reader to our previous works [3], [2].

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 subproblems 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.

B. Tree Parameterization

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 parameterization defines the subset of variables that are modified when updating a constraint. An efficient way of parameterizing the node is to use a tree. One can construct a spanning tree (not necessarily a minimum one) from the graph of poses. Given such a tree, we define the parameterization for a node as

$$x_i \quad = \quad p_i - p_{\mathrm{parent}(i)}, \tag{4}$$

where $p_{\mathrm{parent}(i)}$ refers to the parent of node i in the spanning tree. As defined in Eq. (4), 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

$$r'_{ji} \quad = \quad \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} x_{k^{[-]}} - \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} x_{k^{[+]}} + R_i \delta_{ji}. \tag{5}$$

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 parameterization 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 [2], we compute an approximation of the Jacobian as

$$J'_{ji} \quad = \quad \sum_{k^{[+]} \in \mathcal{P}_{ji}^{[+]}} \mathcal{I}_{k^{[+]}} - \sum_{k^{[-]} \in \mathcal{P}_{ji}^{[-]}} \mathcal{I}_{k^{[-]}}, \tag{6}$$

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

$$\mathbf{x}^{t+1} \quad = \quad \mathbf{x}^t + \lambda |\mathcal{P}_{ji}| \mathbf{M}^{-1} \Omega'_{ji} r'_{ji}, \tag{7}$$

where $|\mathcal{P}_{ji}|$ refers to the number of nodes in $\mathcal{P}_{ji}$. In Eq. (7), we replaced the preconditioning matrix $\mathbf{H}^{-1}$ with its scaled approximation $\mathbf{M}^{-1}$ as described in [2]. 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 parameterization 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.

In principle, one could apply the technique described in this section as a batch algorithm to an arbitrarily constructed spanning tree of the graph. However, our proposed method uses a spanning tree which can be constructed incrementally, as described in the next section.

IV. Online Network Optimization

The algorithm presented in the previous section is a batch procedure. At every iteration, the poses of all nodes in the network are optimized. The fraction of the residual used in updating every constraint decreases over time with the learning rate λ, which evolves according to an harmonic progression. During online optimization, the network is dynamically updated to incorporate new movements and observations. In theory, one could also apply the batch version of our optimizer to correct the network. This, however, would require to compute a solution from scratch each time the robot moves or makes an observation which would obviously lead to an inefficient algorithm.

In this section we describe an incremental version of our optimization algorithm, which is suitable for solving online mapping problems. As pointed in [6] an incremental algorithm should have the following properties:

1) Every time a constraint is added to the network, only the part of the network which is affected by that constraint should be optimized. For example, when exploring new terrain, the effects of the optimization should not perturb distant parts of the graph.
2) When revisiting a known region of the environment it is common to re-localize the robot in the previously built map. One should use the information provided by the re-localization to compute a better initial guess for the position of the newly added nodes.
3) To have a consistent network, performing an optimization step after adding each constraint is often not needed. This happens when the newly added constraints are adequately satisfied by the current network configuration. Having a criterion for deciding when to perform unnecessary optimizations can save a substantial amount of computation.

In the remainder of this section, we present four improvements to the algorithm so that it satisfies the discussed properties.

A. Incremental Construction of the Tree

When constructing the parameterization tree online, we can assume that the input is a sequence of poses corresponding to a trajectory of the robot. In this case, subsequent poses are located closely together and there exist constraints between subsequent poses resulting from odometry or scan-matching. Further constraints between arbitrary nodes result from observations when revisiting a place in the environment.

We proceed as follows: the oldest node is the root of the tree. When adding a node i to the network, we choose as its parent the oldest node for which a constraint to the node i exists. Such a tree can be constructed incrementally since adding a new node does not require to change the existing parts of the tree.

The pose p_i and parameter x_i of a newly added node i is initialized according to the position of the parent node and the connecting constraint as

$$p_i = p_{\text{parent}(i)} \oplus \delta_{i,\text{parent}(i)} \quad (8)$$

$$x_i = p_i - p_{\text{parent}(i)}. \quad (9)$$

The parent node represents an already explored part of the environment and the constraint between the new node and the parent can be regarded as a localization event in an already constructed map, thus satisfying Property 2. As shown in the experiments described below, this initialization appears to be a good heuristic for determining the initial guess of the pose of a newly added node.

B. Constraint Selection

When adding a constraint $\langle j, i\rangle$ to the graph, a subset of nodes needs to be updated. This set depends on the topology of the network and can be determined by a variant of breadth first visit. Let $\mathcal{G}_{j,i}$ be the minimal subgraph that contains the added constraint and has only one constraint to the rest of the graph. Then, the nodes that need to be updated are all nodes of the minimal subtree that contains $\mathcal{G}_{j,i}$. The precise formulation on how to efficiently determine this set is given by Algorithm 1.

```
Data: $\langle j, i\rangle$: the constraint, $\mathcal{G}$: the graph, $\mathcal{T}$: the tree.
Result: $\mathcal{N}_{ji}$: the set of affected nodes, $\mathcal{E}_{ji}$: the affected
        constraints.
Queue $f = \text{childrenOf}(\text{topNode}(\langle j, i\rangle))$;
$\mathcal{E}_{ji} := \text{edgesToChildren}(\text{topNode}(\langle j, i\rangle))$;
foreach $\langle a, b\rangle \in \mathcal{E}_{ji}$ do
    $\langle a, b\rangle .mark = true$;
end
while $f \neq \{\}$ do
    Node $n := \text{first}(f)$;
    $n.mark := true$
    foreach $\langle a, b\rangle \in \text{edgesOf}(n)$ do
        if $\langle a, b\rangle .mark = true$ then
            continue;
        end
        Node $m := (a = n)?b : a$;
        if $m = \text{parent}(n)$ or $m.mark = true$ then
            continue;
        end
        $\langle a, b\rangle .mark = true$;
        $\mathcal{E}_{ji} := \mathcal{E}_{ji} \cup \{\langle a, b\rangle\}$;
        if $\langle a, b\rangle \in \mathcal{T}$ then
            $f := f \cup \{m\}$;
        else
            $f := f \cup \text{childrenOf}(\text{topNode}(\langle a, b\rangle))$;
        end
    end
    $f := \text{removeFirst}(f)$;
    $\mathcal{N}_{ji} := \mathcal{N}_{ji} \cup \{n\}$;
end
```

Algorithm 1: Construction of the set of nodes affected by a constraint. For readability we assume that the frontier f can contain only the nodes which are not already marked.

Note that the number of nodes in $\mathcal{G}_{j,i}$ does depend only on the root of the tree and on the overall graph. It contains all variables which are affected by adding the new costraint $\langle i, j\rangle$.

C. Adaptive Learning Rates

Rather than using one learning rate λ for all nodes, the incremental version of the algorithm uses spatially adaptive learning rates introduced in [6]. The idea is to assign an individual learning rate to each node, allowing different parts of the network to be optimized at different rates. These learning rates are initialized when a new constraint is added to the network and they decrease with each iteration of the algorithm. In the following, we describe how to initialize and update the learning rates and how to adapt the update of the network specified in Eq. (7).

a) Initialization of the learning rates: When a new constraint $\langle j, i\rangle$ is added to the network, we need to update the learning rates for the nodes $\mathcal{N}_{ji}$ determined in the previous section. First, we compute the learning rate λ'_{ji} for the newly introduced information. Then, we propagate this learning rate to the nodes $\mathcal{N}_{ji}$.e

A proper learning rate is determined as follows. Let β_{ji} be the fraction of the residual that would appropriately fuse the previous estimate and the new constraint. Similar to a Kalman filter, β_{ji} is determined as

$$\beta_{ji} = \Omega_{ji}(\Omega_{ji} + \Omega^{\text{graph}}_{ji})^{-1}, \quad (10)$$

where Ω_{ji} is the information matrix of the new constraint, and $\Omega^{\text{graph}}_{ji}$ is an information matrix representing the uncertainty of the constraints in the network. Based on Eq. (10), we can compute the learning rate λ'_{ji} of the new constraint as

$$\lambda'_{ji} = \text{maxrow}\left(\frac{1}{|\mathcal{P}_{ji}|}(\beta_{ji} \oslash \mathbf{M}\Omega'_{ji})\right). \quad (11)$$

Here $\oslash$ represents the row by row division (see [6] for further details). The learning rate of the constraint is then propagated to all nodes $k \in \mathcal{N}_{ji}$ as

$$\lambda_k \leftarrow \max(\lambda_k, \lambda'_{ji}), \quad (12)$$

where λ_k is the learning rate of the node k. According to Eq. (11) constraints with large residuals result in larger learning rate increases than constraints with small residuals.

b) Update of the network: When updating the network, one has to consider the newly introduced learning rates. During an iteration, we decrease the individual learning rates of the nodes according to a generalized harmonic progression [13]:

$$\lambda_k \leftarrow \frac{\lambda_k}{1 + \lambda_k} \quad (13)$$

In this way, one guarantees the strong monotonicity of λ_k and thus the convergence of the algorithm to an equilibrium point.

The learning rates of the nodes cannot be directly used for updating the poses since Eq. (7) requires a learning rate for each constraint and not for each node. When updating the network given the constraint $\langle j, i\rangle$, we obtain an average learning rate $\tilde{\lambda}_{ji}$ from the nodes on $\mathcal{P}_{ji}$ as

$$\tilde{\lambda}_{ji} = \frac{1}{|\mathcal{P}_{ji}|} \sum_{k \in \mathcal{P}_{ji}} \lambda_k. \quad (14)$$

Then, the constraint update turns into

$$\Delta \mathbf{x}_k = \tilde{\lambda}_{ji} |\mathcal{P}_{ji}| \mathbf{M}^{-1} \Omega'_{ji} r'_{ji}. \tag{15}$$

D. Scheduling the Network Optimization

When adding a set of constraints $\langle j, i \rangle \in \mathcal{C}_{\text{new}}$ to a network without performing an optimization, we can incrementally compute the error of the network as

$$e_{\text{new}} = \sum_{\langle j,i \rangle \in \mathcal{C}_{\text{old}}} r_{ji}^T \Omega_{ji} r_{ji} + \sum_{\langle j,i \rangle \in \mathcal{C}_{\text{new}}} r_{ji}^T \Omega_{ji} r_{ji}. \tag{16}$$

Here e_{new} is the new error and $\mathcal{C}_{\text{old}}$ refers to the set of constraints before the modification.

To avoid unnecessary computation, we perform the optimization only if needed. This is the case when the newly incorporated information introduced a significant error compared to the error of the network before. We perform an optimization step if

$$\frac{e_{\text{new}}}{|\mathcal{C}_{\text{new}}| + |\mathcal{C}_{\text{old}}|} > \alpha \max_{\langle j,i \rangle \in \mathcal{C}_{\text{old}}} r_{ji}^T \Omega_{ji} r_{ji}. \tag{17}$$

Here α is a user-defined factor that allows the designer of a mapping system to adapt the quality of the incremental solutions to the needs of the specific application.

If we assume that the network in $\mathcal{C}_{\text{old}}$ has already converged, this heuristic triggers an optimization only if a significant inconsistency is revealed. Furthermore, the optimization only needs to be performed for a subset of the network and not for the whole network. The subset is given by

$$\mathcal{E} = \bigcup_{\langle j,i \rangle \in \mathcal{C}_{\text{new}}} \mathcal{E}_{ji}. \tag{18}$$

Here $\mathcal{E}_{ji}$ is the set of constraints to be updated given a new constraint $\langle j, i \rangle \in \mathcal{C}_{\text{new}}$. The sets $\mathcal{E}_{ji}$ are computed according to Algorithm 1. This criterion satisfies Property 3 and leads to an efficient algorithm for incrementally optimizing the network of constraints.

V. EXPERIMENTS

This section is designed to evaluate the effectiveness of the proposed methods to incrementally learn maximum likelihood maps. We first show that such a technique is well suited to generate accurate grid maps given laser range data and odometry from a real robot. Second, we provide simulation experiments to evaluate the evolution of the error and provide comparisons to our previously proposed techniques [3], [2], [6]. Finally, we illustrate the computational advantages resulting from our algorithm.

A. Real World Experiments

To illustrate that our technique can be used to learn maps from real robot data, we used the freely available ACES dataset. The motivating example shown in Figure 1 depicts four different maps computed online by our incremental mapping technique. During this experiment, we extracted constraints between consecutive poses by means of pairwise scan matching. Loop closures were determined by localizing the robot in the previously built map by means of a particle filter.

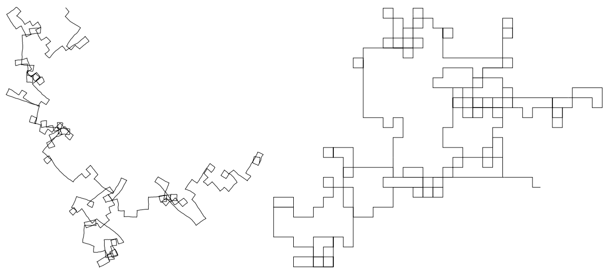


Fig. 2. Network used in the simulated experiments. Left: initial guess. Right: ground truth.

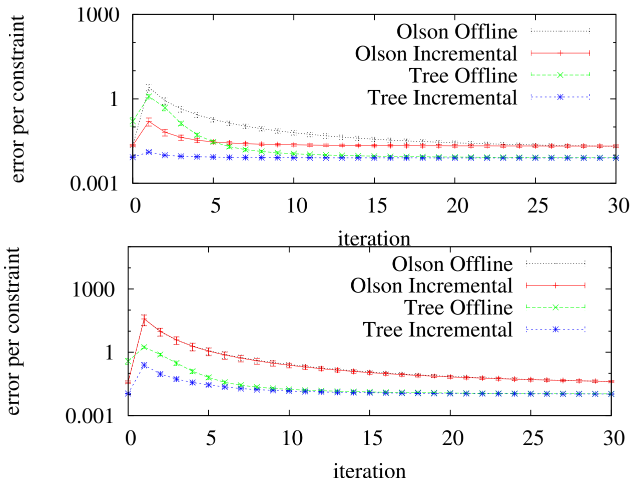


Fig. 3. Statistical experiments showing the evolution of the error per iteration of the algorithm. Top: situation in which the robot is closes a small loop. Bottom: closure of a large loop. The statistics have been generated by considering 10 different realizations of the observation noise along the same path.

As can be seen, our approach leads to accurate maps for real robot data. Similar results were obtained with all datasets we found online or recorded on our own.

B. Statistical Experiments on the Evolution of the Error

In the these experiments, we moved a virtual robot on a grid world. An observation is generated each time the current position of the robot was close to a previously visited location. The observations are corrupted by a given amount of Gaussian noise. The network used in this experiment is depicted in Figure 2.

We compare our approach named *Tree Incremental* with its offline variant [3] called *Tree Offline* which solves the overall problem from scratch. In addition to that, we compare it to the offline version without the tree optimization [2] called *Olson Offline* as well as its incremental variant [6] referred to as *Olson Incremental*. For space reasons, we omit comparisons to LU decomposition, EKF, and Gauss-Seidel. The advantages of our method over these other methods is similar to those previously reported [2].

To allow a fair comparison, we disabled the scheduling of the optimization of Eq. (17) and we performed 30 iterations every time 16 constraints were added to the network. During the very first iterations, the error of all approaches may show

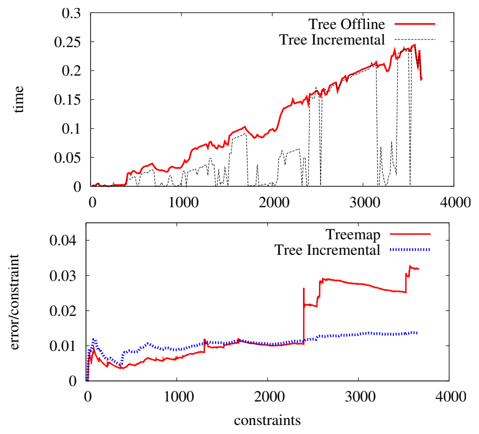


Fig. 4. Top: runtime comparison of the offline and the incremental approaches using a tree parameterization. The optimization is performed only when the error condition specified by Eq. (17) was verified. Bottom: Comparison of the evolution of the global error between Treemap[12] and the online version of our approach.

an increase, due to the bigger correction steps which result from increasing the learning rates.

Figure 3 depicts the evolution of the error for all four techniques during a mapping experiment. We depicted two situations. In the first one, the robot closed a small loop. As can be seen, the introduced error is small and thus our approach corrects the error within 2 iterations. Both incremental techniques perform better than their offline variants. The approach proposed in this paper outperforms the other techniques. The same holds for the second situation in which the robot was closing a large loop. Note that in most cases, one iteration of the incremental approach can be carried out faster, since only a subpart of the network needs to be updated.

C. *Runtime Comparison*

Finally, we evaluated our incremental version and its offline variant with respect to the execution time. Both methods where executed only when needed according to our criterion specified by Eq. (17). We measured the time needed to run the individual approach until convergence to the same low error configuration, or until a maximum number of iterations (30) was reached. As can be seen in Figure 4(top), the incremental technique requires significantly less operations and thus runtime to provide equivalent results in terms of error. Figure 4(bottom) shows the error plot of a comparison of our approach and Treemap [12] proposed by Frese. As shown in the error-plot, in the beginning Treemap performs slightly better than our algorithm, due to the exact calculation of the Jacobians. However, when closing large loops Treemap is more sensitive to angular wraparounds (see increase of the error at constraint 2400 in Figure 4). This issue is typically better handled by our iterative procedure. Overall, we observed that for datasets having a small noise Treemap provides slightly better estimates, while our approach is generally more robust to extreme conditions.

VI. Conclusion

In this paper, we presented an efficient online solution to the optimization of constraint networks. It can incrementally learn maps while the robot moves through the environment. Our approach optimizes a network of constraints that represents the spatial relations between the poses of the robot. It uses a tree-parameterization of the nodes and applies a variant of gradient descent to compute network configurations with low errors.

A per-node adaptive learning rate allows the robot to reuse already computed solutions from previous steps, to update only the parts of the network, which are affected by the newly incorporated information, and to start the optimization approach only if the new data causes inconsistencies with the already computed solution. We tested our approach on real robot data as well as with simulated datasets. We compared it to recently presented online and offline methods that also address the network-based SLAM problem. As we showed in practical experiments, our approach converges faster to a configuration with small errors.

Acknowledgment

The authors gratefully thank Udo Frese for providing us his Treemap implementation. This work has partly been supported by the DFG under contract number SFB/TR-8 (A3), by the EC under contract number FP6-IST-34120-muFly, and FP6-2005-IST-6-RAWSEEDS.

References

[1] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Journal of Autonomous Robots*, vol. 4, 1997.

[2] 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)*, 2006, pp. 2262–2269.

[3] 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. [Online]. Available: http://www.informatik.uni-freiburg.de/ stachnis/pdf/grisetti07rss.pdf

[4] R. Smith, M. Self, and P. Cheeseman, "Estimating uncertain spatial realtionships in robotics," in *Autonomous Robot Vehicles*, I. Cox and G. Wilfong, Eds. Springer Verlag, 1990, pp. 167–193.

[5] M. Montemerlo and S. Thrun, "Simultaneous localization and mapping with unknown data association using FastSLAM," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Taipei, Taiwan, 2003.

[6] E. Olson, J. Leonard, and S. Teller, "Spatially-adaptive learning rates for online incremental slam," in *Robotics: Science and Systems*, Atlanta, GA, USA, 2007.

[7] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Fast incremental smoothing and mapping with efficient data association," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2007.

[8] U. Frese, P. Larsson, and T. Duckett, "A multilevel relaxation algorithm for simultaneous localisation and mapping," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 1–12, 2005.

[9] M. Bosse, P. Newman, J. Leonard, and S. Teller, "An ALTAS framework for scalable mapping," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, Taipei, Taiwan, 2003.

[10] C. Estrada, J. Neira, and J. Tardós, "Hierachical slam: Real-time accurate mapping of large environments," *IEEE Transactions on Robotics*, vol. 21, no. 4, pp. 588–596, 2005.

[11] J.-S. Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proc. of the IEEE Int. Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, Monterey, CA, USA, 1999, pp. 318–325.

[12] U. Frese, "Treemap: An $o(logn)$ algorithm for indoor simultaneous localization and mapping," *Journal of Autonomous Robots*, vol. 21, no. 2, pp. 103–122, 2006.

[13] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.

Inverse Depth Parametrization for Monocular SLAM

Javier Civera, Andrew J. Davison, and J. M. Martínez Montiel

***Abstract*—We present a new parametrization for point features within monocular simultaneous localization and mapping (SLAM) that permits efficient and accurate representation of uncertainty during undelayed initialization and beyond, all within the standard extended Kalman filter (EKF). The key concept is direct parametrization of the inverse depth of features relative to the camera locations from which they were first viewed, which produces measurement equations with a high degree of linearity. Importantly, our parametrization can cope with features over a huge range of depths, even those that are so far from the camera that they present little parallax during motion—maintaining sufficient representative uncertainty that these points retain the opportunity to "come in" smoothly from infinity if the camera makes larger movements. Feature initialization is undelayed in the sense that even distant features are immediately used to improve camera motion estimates, acting initially as bearing references but not permanently labeled as such. The inverse depth parametrization remains well behaved for features at all stages of SLAM processing, but has the drawback in computational terms that each point is represented by a 6-D state vector as opposed to the standard three of a Euclidean *XYZ* representation. We show that once the depth estimate of a feature is sufficiently accurate, its representation can safely be converted to the Euclidean *XYZ* form, and propose a linearity index that allows automatic detection and conversion to maintain maximum efficiency—only low parallax features need be maintained in inverse depth form for long periods. We present a real-time implementation at 30 Hz, where the parametrization is validated in a fully automatic 3-D SLAM system featuring a hand-held single camera with no additional sensing. Experiments show robust operation in challenging indoor and outdoor environments with a very large ranges of scene depth, varied motion, and also real time 360° loop closing.**

***Index Terms*—Monocular simultaneous localization and mapping (SLAM), real-time vision.**

Manuscript received February 27, 2007; revised September 28, 2007. First published October 3, 2008; current version published October 31, 2008. This work was supported in part by the Spanish Grant PR2007-0427, Grant DPI2006-13578, and Grant DGA(CONSI+D)-CAI IT12-06, in part by the Engineering and Physical Sciences Research Council (EPSRC) Grant GR/T24685, in part by the Royal Society International Joint Project grant between the University of Oxford, University of Zaragoza, and Imperial College London, and in part by the Robotics Advancement through Web-Publishing of Sensorial and Elaborated Extensive Datasets (RAWSEEDS) under Grant FP6-IST-045144. The work of A. J. Davison was supported the EPSRC Advanced Research Fellowship Grant. This paper was recommended for publication by Associate Editor P. Rives and Editor L. Parker upon evaluation of the reviewers' comments.

J. Civera and J. M. Martínez Montiel are with the Departamento de Informática, University of Zaragoza, 50018 Zaragoza, Spain (e-mail: josemari@unizar.es; jcivera@unizar.es).

A. J. Davison is with the Department of Computing, Imperial College London, SW7 2AZ London, U.K. (e-mail: ajd@doc.ic.ac.uk).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the author. This material includes the following video files. *inverseDepth_indoor.avi* (11.7 MB) shows simultaneous localization and mapping, from a hand-held camera observing an indoor scene. All the processing is automatic, the image sequence being the only sensorial information used as input. It is shown as a top view of the computed camera trajectory and 3-D scene map. Image sequence is acquired with a hand-held camera 320 £ 240 at 30 frames/second. *Player information* XviD MPEG-4. *inverseDepth_outdoor.avi* (12.4 MB) shows real-time simultaneous localization and mapping, from a hand-held camera observing an outdoor scene, including rather distant features. All the processing is automatic, the image sequence being the only sensorial information used as input. It is shown as a top view of the computed camera trajectory and 3-D scene map. Image sequence is acquired with a hand-held camera 320 £ 240 at 30 frames/second. The processing is done with a standard laptop. *Player information* XviD MPEG-4. *inverseDepth_loopClosing.avi* (10.2MB) shows simultaneous localization and mapping, from a hand-held camera observing a loop-closing indoor scene. All the processing is automatic, the image sequence being the only sensorial information used as input. It is shown as a top view of the computed camera trajectory and 3-D scene map. Image sequence is acquired with a hand-held camera 320 £ 240 at 30 frames/second. *Player information* XviD MPEG-4. *inverseDepth_loopClosing_ID_to_XYZ_conversion.avi* (10.1 MB) shows simultaneous localization and mapping, from a hand-held camera observing the same loop-closing indoor sequence as in inverseDepth loopClosing.avi, but switching from inverse depth to *XYZ* parameterization when necessary. All the processing is automatic, the image sequence being the only sensorial information used as input. It is shown as a top view of the computed camera trajectory and 3-D scene map. Image sequence is acquired with a hand-held camera 320 £ 240 at 30 frames/second. *Player information* XviD MPEG-4. *inverseDepth_indoorRawImages.tar.gz* (44 MB) shows indoor sequence raw images in .pgm format. Camera calibration in an ASCII file. *inverseDepth_outdoorRawImages.tar.gz* (29 MB) shows outdoor sequence raw images in .pgm format. Camera calibration in an ASCII file. *inverseDepth_loopClosingRawImages.tar.gz* (33 MB) shows loop-closing sequence raw images in .pgm format. Camera calibration in an ASCII file. *Contact information* jcivera@unizar.es; josemari@unizar.es.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2008.2003276

I. Introduction

A MONOCULAR camera is a projective sensor that measures the bearing of image features. Given an image sequence of a rigid 3-D scene taken from a moving camera, it is now well known that it is possible to compute both a scene structure and a camera motion up to a scale factor. To infer the 3-D position of each feature, the moving camera must observe it repeatedly each time, capturing a ray of light from the feature to its optic center. The measured angle between the captured rays from different viewpoints is the feature's *parallax*—this is what allows its depth to be estimated.

In offline "structure from motion (SFM)" solutions from the computer vision literature (e.g., [11] and [23]), motion and structure are estimated from an image sequence by first applying a robust feature matching between pairs or other short overlapping sets of images to estimate relative motion. An optimization procedure then iteratively refines global camera location and scene feature position estimates such that features project as closely as possible to their measured image positions (bundle adjustment). Recently, work in the spirit of these methods, but with "sliding window" processing and refinement rather than global optimization, has produced impressive real-time "visual odometry" results when applied to stereo sequences in [21] and for monocular sequences in [20].

An alternative approach to achieving real-time motion and structure estimation are online visual simultaneous localization and mapping (SLAM) approaches that use a probabilistic filtering approach to sequentially update estimates of the positions of features (the map) and the current location of the camera. These SLAM methods have different strengths and weaknesses to visual odometry, being able to build consistent and drift-free global maps, but with a bounded number of mapped features. The core single extended Kalman filter (EKF) SLAM technique, previously proven in multisensor robotic applications, was first applied successfully to real-time monocular camera tracking by Davison *et al.* [8], [9] in a system that built sparse room-sized maps at 30 Hz.

A significant limitation of Davison's and similar approaches, however, was that they could only make use of features that were close to the camera relative to its distance of translation, and therefore exhibited significant parallax during motion. The problem was in initializing uncertain depth estimates for distant features: in the straightforward Euclidean *XYZ* feature parametrization adopted, position uncertainties for low parallax features are not well represented by the Gaussian distributions implicit in the EKF. The depth coordinate of such features has a probability density that rises sharply at a well-defined minimum depth to a peak, but then, tails off very slowly toward infinity—from low parallax measurements, it is very difficult to tell whether a feature has a depth of 10 units rather than 100, 1000, or more. For the rest of the paper, we refer to Euclidean *XYZ* parametrization simply as *XYZ*.

There have been several recent methods proposed for coping with this problem, relying on generally undesirable special treatment of newly initialized features. In this paper, we describe a new feature parametrization that is able to smoothly cope with initialization of features at all depths—even up to "infinity"—within the standard EKF framework. The key concept is direct parametrization of inverse depth relative to the camera position from which a feature was first observed.

A. Delayed and Undelayed Initialization

The most obvious approach to coping with feature initialization within a monocular SLAM system is to treat newly detected features separately from the main map, accumulating information in a special processing over several frames to reduce depth uncertainty before insertion into the full filter with a standard *XYZ* representation. Such *delayed initialization* schemes (e.g., [3], [8], and [14]) have the drawback that new features, held outside the main probabilistic state, are not able to contribute to the estimation of the camera position until finally included in the map. Further, features that retain low parallax over many frames (those very far from the camera or close to the motion epipole) are usually rejected completely because they never pass the test for inclusion.

In the delayed approach of Bailey [2], initialization is delayed until the measurement equation is approximately Gaussian and the point can be safely triangulated; here, the problem was posed in 2-D and validated in simulation. A similar approach for a 3-D monocular vision with inertial sensing was proposed in [3]. Davison [8] reacted to the detection of a new feature by inserting a 3-D semiinfinite ray into the main map representing everything about the feature except its depth, and then, used an auxiliary particle filter to explicitly refine the depth estimate over several frames, taking advantage of all the measurements in a high frame rate sequence, but again with new features held outside the main state vector until inclusion.

More recently, several *undelayed initialization* schemes have been proposed, which still treat new features in a special way but are able to benefit immediately from them to improve camera motion estimates—the key insight being that while features with highly uncertain depths provide little information on camera translation, they are extremely useful as bearing references for orientation estimation. The undelayed method proposed by Kwok and Dissanayake [15] was a multiple hypothesis scheme, initializing features at various depths and pruning those not reobserved in subsequent images.

Sola *et al.* [24], [25] described a more rigorous undelayed approach using a Gaussian sum filter approximated by a federated information sharing method to keep the computational overhead low. An important insight was to spread the Gaussian depth hypotheses along the ray according to inverse depth, achieving much better representational efficiency in this way. This method can perhaps be seen as the direct stepping stone between Davison's particle method and our new inverse depth scheme; a Gaussian sum is a more efficient representation than particles (efficient enough that the separate Gaussians can all be put into the main state vector), but not as efficient as the single Gaussian representation that the inverse depth parametrization allows. Note that neither [15] nor [25] considers features at very large "infinite" depths.

B. Points at Infinity

A major motivation of the approach in this paper is not only the efficient undelayed initialization, but also the desire to cope with features at *all* depths, particularly in outdoor scenes. In SFM, the well-known concept of a point at infinity is a feature that exhibits no parallax during camera motion due to its extreme depth. A star for instance would be observed at the same image location by a camera that translated through many kilometers pointed up at the sky without rotating. Such a feature cannot be used for estimating camera translation but is a perfect bearing reference for estimating rotation. The homogeneous coordinate systems of visual projective geometry used normally in SFM allow explicit representation of points at infinity, and they have proven to play an important role during offline structure and motion estimation.

In a sequential SLAM system, the difficulty is that we do not know in advance which features are infinite and which are not. Montiel and Davison [19] showed that in the special case where *all features are known to be infinite*—in very-large-scale outdoor scenes or when the camera rotates on a tripod— SLAM in pure angular coordinates turns the camera into a real-time visual compass. In the more general case, let us imagine a camera moving through a 3-D scene with observable features at a range of depths. From the estimation point of view, we can think of all features starting at infinity and "coming in" as the camera moves far enough to measure sufficient parallax. For nearby indoor features, only a few centimeters of movement will be sufficient. Distant features may require many meters or even kilometers of motion before parallax is observed. It is important that these features are not permanently labeled as infinite—a feature that seems to be at infinity should always have the chance to prove its finite depth given enough motion, or there will be the serious risk of systematic errors in the scene map. Our probabilistic SLAM algorithm must be able to represent the uncertainty in depth of seemingly infinite features. Observing no parallax for a feature after 10 units of camera translation does tell us something about its depth—it gives a reliable lower bound, which depends on the amount of motion made by the camera (if the feature had been closer than this, we *would* have observed parallax). This explicit consideration of uncertainty in the locations of points has not been previously required in offline computer vision algorithms, but is very important in a more difficult online case.

C. Inverse Depth Representation

Our contribution is to show that, in fact, there is a unified and straightforward parametrization for feature locations that can handle both initialization and standard tracking of both close and very distant features within the standard EKF framework. An explicit parametrization of the *inverse depth* of a feature along a semiinfinite ray from the position from which it was first viewed allows a Gaussian distribution to cover uncertainty in depth that spans a depth range from nearby to infinity, and permits seamless crossing over to finite depth estimates of features that have been apparently infinite for long periods of time. The unified representation means that our algorithm requires no special initialization process for features. They are simply tracked right from the start, immediately contribute to improved camera estimates, and have their correlations with all other features in the map correctly modeled. Note that our parameterization would be equally compatible with other variants of Gaussian filtering such as sparse information filters.

We introduce a linearity index and use it to analyze and prove the representational capability of the inverse depth parametrization for both low and high parallax features. The only drawback of the inverse depth scheme is the computational issue of increased state vector size since an inverse depth point needs six parameters rather than the three of *XYZ* coding. As a solution to this, we show that our linearity index can also be applied to the *XYZ* parametrization to signal when a feature can be safely switched from inverse depth to *XYZ*; the usage of the inverse depth representation can, in this way, be restricted to low parallax feature cases where the *XYZ* encoding departs from Gaussianity. Note that this "switching," unlike in delayed initialization methods, is purely to reduce computational load; SLAM accuracy with or without switching is almost the same.

The fact is that the projective nature of a camera means that the image measurement process is nearly linear in this inverse depth coordinate. Inverse depth is a concept used widely in computer vision: it appears in the relation between image disparity and point depth in a stereo vision; it is interpreted as the parallax with respect to the plane at infinity in [12]. Inverse depth is also used to relate the motion field induced by scene points with the camera velocity in optical flow analysis [13]. In the tracking community, "modified polar coordinates" [1] also exploit the linearity properties of the inverse depth representation in a slightly different, but closely related, problem of a target motion analysis (TMA) from measurements gathered by a bearing-only sensor with known motion.

However, the inverse depth idea has not previously been properly integrated in sequential, probabilistic estimation of motion, and structure. It has been used in EKF-based sequential depth estimation from camera-known motion [16], and in a multibaseline stereo, Okutomi and Kanade [22] used the inverse depth to increase matching robustness for scene symmetries; matching scores coming from multiple stereo pairs with different baselines were accumulated in a common reference coded in inverse depth, this paper focusing on matching robustness and not on probabilistic uncertainty propagation. Chowdhury and Chellappa [5] proposed a sequential EKF process using inverse depth, but this was in some way short of full SLAM in its details. Images are first processed pairwise to obtain a sequence of 3-D motions that are then fused with an individual EKF per feature.

It is our parametrization of inverse depth *relative to the positions from which features were first observed*, which means that a Gaussian representation is uniquely well behaved, this is the reason why a straightforward parametrization of monocular SLAM in the homogeneous coordinates of SFM will not give a good result—that representation only meaningfully represents points that appear to be infinite relative to the coordinate origin. It could be said in projective terms that our method defines separate but correlated projective frames for each feature. Another interesting comparison is between our method, where the representation for each feature includes the camera position from which it was first observed, and smoothing/full SLAM schemes, where all historical sensor pose estimates are maintained in a filter.

Two recently published papers from other authors have developed methods that are quite similar to ours. Trawny and Roumeliotis [26] proposed an undelayed initialization for 2-D monocular SLAM that encodes a map point as the intersection of two projection rays. This representation is overparametrized but allows undelayed initialization and encoding of both close and distant features, the approach validated with simulation results.

Eade and Drummond presented an inverse depth initialization scheme within the context of their FastSLAM-based system for monocular SLAM [10], offering some of the same arguments about advantages in linearity as in our paper. The position of each new partially initialized feature added to the map is parametrized with three coordinates representing its direction and inverse depth relative to the camera pose at the first observation, and estimates of these coordinates are refined within a set of Kalman filters for each particle of the map. Once the inverse depth estimation has collapsed, the feature is converted to a fully initialized standard *XYZ* representation. While retaining the differentiation between partially and fully initialized features, they go further and are able to use measurements of partially initialized features with unknown depth to improve estimates of camera orientation and translation via a special epipolar update step. Their approach certainly appears appropriate within a FastSLAM implementation. However, it lacks the satisfying unified quality of the parametrization we present in this paper, where the transition from partially to fully initialized need not be explicitly tackled and full use is automatically made of all of the information available in measurements.

This paper offers a comprehensive and extended version of our work previously published as two conference papers [7], [18]. We now present a full real-time implementation of the inverse depth parameterization that can map up to 50–70 features in real time on a standard laptop computer. Experimental validation has shown the important role of an accurate camera calibration to improve the system performance, especially with wide-angle cameras. Our results section includes new real-time experiments, including the key result of vision-only loop closing. Input test image sequences and movies showing the computed solution are included in the paper as multimedia material.

Section II is devoted to defining the state vector, including the camera motion model, *XYZ* point coding, and inverse depth point parametrization. The measurement equation is described in Section III. Section IV presents a discussion about measurement equation linearization errors. Next, feature initialization from a single-feature observation is detailed in Section V. In Section VI, the switch from inverse depth to *XYZ* coding is presented, and in Section VII, we present experimental validations over real-image sequences captured at 30 Hz in large-scale environments, indoors and outdoors, including real-time performance, and a loop closing experiment; links to movies showing the system performance are provided. Finally, Section VIII is devoted to conclusions.

II. State Vector Definition

A. Camera Motion

A constant angular and linear velocity model is used to model handheld camera motion. The camera state $\mathbf{x}_v$ is composed of pose terms: $\mathbf{r}^{WC}$ camera optical center position and $\mathbf{q}^{WC}$ quaternion defining orientation, and linear and angular velocity $\mathbf{v}^W$ and ω^C relative to world frame W and camera frame C, respectively.

We assume that linear and angular accelerations $\mathbf{a}^W$ and α^C affect the camera, producing at each step, an impulse of linear velocity $\mathbf{V}^W = \mathbf{a}^W \Delta t$ and angular velocity $\Omega^C = \alpha^C \Delta t$, with zero mean and known Gaussian distribution. We currently assume a diagonal covariance matrix for the unknown input linear and angular accelerations.

The state update equation for the camera is

$$\mathbf{f}_v = \begin{pmatrix} \mathbf{r}^{WC}_{k+1} \\ \mathbf{q}^{WC}_{k+1} \\ \mathbf{v}^{W}_{k+1} \\ \omega^{C}_{k+1} \end{pmatrix} = \begin{pmatrix} \mathbf{r}^{WC}_{k} + \left(\mathbf{v}^W_k + \mathbf{V}^W_k\right)\Delta t \\ \mathbf{q}^{WC}_{k} \times \mathbf{q}\left(\left(\omega^C_k + \Omega^C\right)\Delta t\right) \\ \mathbf{v}^W_k + \mathbf{V}^W \\ \omega^C_k + \Omega^C \end{pmatrix} \tag{1}$$

where $\mathbf{q}((\omega^C_k + \Omega^C)\Delta t)$ is the quaternion defined by the rotation vector $(\omega^C_k + \Omega^C)\Delta t$.

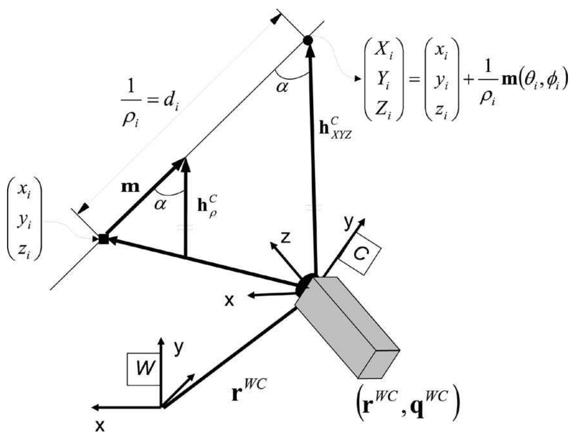

$\frac{1}{\rho_i} = d_i$
α
$\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i}\mathbf{m}(\theta_i, \phi_i)$
$\mathbf{h}^C_{XYZ}$
$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix}$
$\mathbf{m}$
α
$\mathbf{h}^C_\rho$
z
y
C
x
W
y
x
$\mathbf{r}^{WC}$
$\left(\mathbf{r}^{WC}, \mathbf{q}^{WC}\right)$

Fig. 1. Feature parametrization and measurement equation.

B. Euclidean XYZ Point Parametrization

The standard representation for scene points i in terms of Euclidean *XYZ* coordinates (see Fig. 1) is

$$\mathbf{x}_i = \begin{pmatrix} X_i & Y_i & Z_i \end{pmatrix}^\top. \tag{2}$$

In this paper, we refer to the Euclidean *XYZ* coding simply as *XYZ* coding.

C. Inverse Depth Point Parametrization

In our new scheme, a scene 3-D point i can be defined by the 6-D state vector:

$$\mathbf{y}_i = \begin{pmatrix} x_i & y_i & z_i & \theta_i & \phi_i & \rho_i \end{pmatrix}^\top \tag{3}$$

which models a 3-D point located at (see Fig. 1)

$$\mathbf{x}_i = \begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i}\mathbf{m}\left(\theta_i, \phi_i\right) \tag{4}$$

$$\mathbf{m} = \left(\cos\phi_i \sin\theta_i, -\sin\phi_i, \cos\phi_i \cos\theta_i\right)^\top. \tag{5}$$

The $\mathbf{y}_i$ vector encodes the *ray from the first camera position from which the feature was observed* by x_i, y_i, z_i, the camera optical center, and θ_i, ϕ_i azimuth and elevation (coded in the world frame) defining unit directional vector $\mathbf{m}(\theta_i, \phi_i)$. The point's depth along the ray d_i is encoded by its inverse $\rho_i = 1/d_i$.

D. Full State Vector

As in standard EKF SLAM, we use a single-joint state vector containing camera pose and feature estimates, with the assumption that the camera moves with respect to a static scene. The whole state vector $\mathbf{x}$ is composed of the camera and all the map features

$$\mathbf{x} = \left(\mathbf{x}_v^\top, \mathbf{y}_1^\top, \mathbf{y}_2^\top, \ldots, \mathbf{y}_n^\top\right)^\top. \tag{6}$$

III. Measurement Equation

Each observed feature imposes a constraint between the camera location and the corresponding map feature (see Fig. 1).

Observation of a point $\mathbf{y}_i(\mathbf{x}_i)$ defines a ray coded by a directional vector in the camera frame $\mathbf{h}^C = (h_x \quad h_y \quad h_z)^\top$. For points in *XYZ*

$$\mathbf{h}^C = \mathbf{h}^C_{XYZ} = \mathbf{R}^{CW} \left(\begin{pmatrix} X_i \\ Y_i \\ Z_i \end{pmatrix} - \mathbf{r}^{WC} \right). \tag{7}$$

For points in inverse depth

$$\mathbf{h}^C = \mathbf{h}^C_{\rho} = \mathbf{R}^{CW} \left(\rho_i \left(\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} - \mathbf{r}^{WC} \right) + \mathbf{m}(\theta_i, \phi_i) \right) \tag{8}$$

where the directional vector has been normalized using the inverse depth. It is worth noting that (8) can be safely used even for points at infinity, i.e., $\rho_i = 0$.

The camera does not directly observe $\mathbf{h}^C$ but its projection in the image according to the pinhole model. Projection to a normalized retina, and then, camera calibration is applied:

$$\mathbf{h} = \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} u_0 - \dfrac{f}{d_x}\dfrac{h_x}{h_z} \\ v_0 - \dfrac{f}{d_y}\dfrac{h_y}{h_z} \end{pmatrix} \tag{9}$$

where u_0, v_0 is the camera's principal point, f is the focal length, and d_x, d_y is the pixel size. Finally, a distortion model has to be applied to deal with real camera lenses. In this paper, we have used the standard two parameters distortion model from photogrammetry [17] (see the Appendix for details).

It is worth noting that the measurement equation in inverse depth has a sensitive dependency on the parallax angle α (see Fig. 1). At low parallax, (8) can be approximated by $\mathbf{h}^C \approx \mathbf{R}^{CW}(\mathbf{m}(\theta_i, \phi_i))$, and hence, the measurement equation only provides information about the camera orientation and the directional vector $\mathbf{m}(\theta_i, \phi_i)$.

IV. Measurement Equation Linearity

The more linear the measurement equation is, the better a Kalman filter performs. This section is devoted to presenting an analysis of measurement equation linearity for both *XYZ* and inverse depth codings. These linearity analyses theoretically support the superiority of the inverse depth coding.

A. Linearized Propagation of a Gaussian

Let x be an uncertain variable with Gaussian distribution $x \sim N(\mu_x, \sigma_x^2)$. The transformation of x through the function f is a variable y that can be approximated with Gaussian distribution:

$$y \sim N(\mu_y, \sigma_y^2), \ \mu_y = f(\mu_x), \ \sigma_y^2 = \left.\frac{\partial f}{\partial x}\right|_{\mu_x} \sigma_x^2 \left.\frac{\partial f}{\partial x}\right|_{\mu_x}^\top \tag{10}$$

if the function f is linear in an interval around μ_x (Fig. 2). The interval size in which the function has to be linear depends on σ_x; the bigger σ_x the wider the interval has to be to cover a significant fraction of the random variable x values. In this paper, we fix the linearity interval to the 95% confidence region defined by $[\mu_x - 2\sigma_x, \mu_x + 2\sigma_x]$.

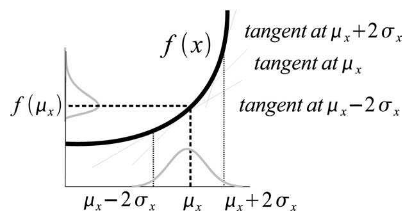
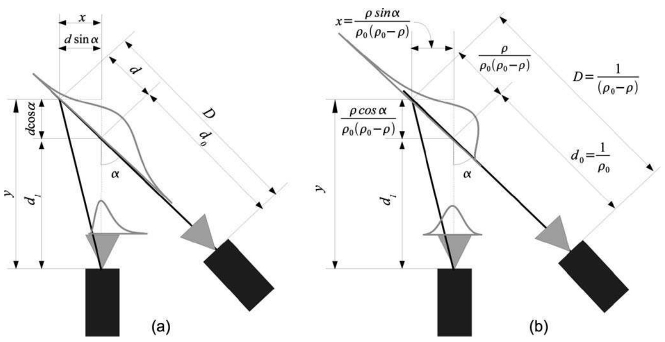


Fig. 2. First derivative variation in $[\mu_x - 2\sigma_x, \mu_x + 2\sigma_x]$ codes the departure from Gaussianity in the propagation of the uncertain variable through a function.



x
$d \sin\alpha$
d
D
d_0
$d \cos\alpha$
y
d_1
α
(a)
$x = \frac{\rho \sin\alpha}{\rho_0(\rho_0-\rho)}$
$\frac{\rho}{\rho_0(\rho_0-\rho)}$
$D = \frac{1}{(\rho_0-\rho)}$
$\frac{\rho \cos\alpha}{\rho_0(\rho_0-\rho)}$
$d_0 = \frac{1}{\rho_0}$
(b)

Fig. 3. Uncertainty propagation from the scene point to the image. (a) *XYZ* coding. (b) Inverse depth coding.

If a function is linear in an interval, the first derivative is constant in that interval. To analyze the first derivative variation around the interval $[\mu_x - 2\sigma_x, \mu_x + 2\sigma_x]$, consider the Taylor expansion for the *first derivative*:

$$\frac{\partial f}{\partial x}(\mu_x + \Delta x) \approx \left.\frac{\partial f}{\partial x}\right|_{\mu_x} + \left.\frac{\partial^2 f}{\partial x^2}\right|_{\mu_x} \Delta x. \tag{11}$$

We propose to compare the value of the derivative at the interval center μ_x with the value at the extremes $\mu_x \pm 2\sigma_x$, where the deviation from linearity will be maximal, using the following dimensionless linearity index:

$$L = \left| \frac{\left.\dfrac{\partial^2 f}{\partial x^2}\right|_{\mu_x} 2\sigma_x}{\left.\dfrac{\partial f}{\partial x}\right|_{\mu_x}} \right|. \tag{12}$$

When $L \approx 0$, the function can be considered linear in the interval, and hence, Gaussianity is preserved during transformation.

B. Linearity of XYZ Parametrization

The linearity of the *XYZ* representation is analyzed by means of a simplified model that only estimates the depth of a point with respect to the camera. In our analysis, a scene point is observed by two cameras [Fig. 3(a)], both of which are oriented toward the point. The first camera detects the ray on which the point lies. The second camera observes the same point from a distance d_1; the parallax angle α is approximated by the angle between the cameras' optic axes.

The point's location error d is encoded as Gaussian in depth

$$D = d_0 + d, \qquad d \sim N(0, \sigma_d^2). \tag{13}$$

This error d is propagated to the image of the point in the second camera u as

$$u = \frac{x}{y} = \frac{d \sin\alpha}{d_1 + d\cos\alpha}. \tag{14}$$

The Gaussianity of u is analyzed by means of (12), giving the following linearity index:

$$L_d = \left| \frac{(\partial^2 u/\partial d^2) 2\sigma_d}{\partial u/\partial d} \right| = \frac{4\sigma_d}{d_1} |\cos\alpha| . \tag{15}$$

C. Linearity of Inverse Depth Parametrization

The inverse depth parametrization is based on the same scene geometry as the direct depth coding, but the depth error is encoded as Gaussian in inverse depth [Fig. 3(b)]:

$$D = \frac{1}{\rho_0 - \rho}, \qquad \rho \sim N\left(0, \sigma_\rho^2\right) \tag{16}$$

$$d = D - d_0 = \frac{\rho}{\rho_0\left(\rho_0 - \rho\right)} \qquad d_0 = \frac{1}{\rho_0}. \tag{17}$$

So, the image of the scene point is computed as

$$u = \frac{x}{y} = \frac{d\sin\alpha}{d_1 + d\cos\alpha} = \frac{\rho\sin\alpha}{\rho_0 d_1\left(\rho_0 - \rho\right) + \rho\cos\alpha} \tag{18}$$

and the linearity index L_ρ is now

$$L_\rho = \left| \frac{(\partial^2 u/\partial \rho^2) 2\sigma_\rho}{\partial u/\partial \rho} \right| = \frac{4\sigma_\rho}{\rho_0} \left| 1 - \frac{d_0}{d_1}\cos\alpha \right| . \tag{19}$$

D. Depth Versus Inverse Depth Comparison

When a feature is initialized, the depth prior has to cover a vast region in front of the camera. With the inverse depth representation, the 95% confidence region with parameters ρ_0, σ_ρ is

$$\left[\frac{1}{\rho_0 + 2\sigma_\rho}, \frac{1}{\rho_0 - 2\sigma_\rho} \right]. \tag{20}$$

This region cannot include zero depth but can easily extend to infinity.

Conversely, with the depth representation, the 95% region with parameters d_0, σ_d is $[d_0 - 2\sigma_d, d_0 + 2\sigma_d]$. This region can include zero depth but cannot extend to infinity.

In the first few frames, after a new feature has been initialized, little parallax is likely to have been observed. Therefore, $d_0/d_1 \approx 1$ and $\alpha \approx 0 \Longrightarrow \cos\alpha \approx 1$. In this case, the L_d linearity index for depth is high (bad), while the L_ρ linearity index for inverse depth is low (good): during initialization, the inverse depth measurement equation linearity is superior to the *XYZ* coding.

As estimation proceeds and α increases, leading to more accurate depth estimates, the inverse depth representation continues to have a high degree of linearity. This is because in the expression for L_ρ, the increase in the term $|1 - (d_0/d_1)\cos\alpha|$ is compensated by the decrease in $4\sigma_\rho/\rho_0$. For inverse depth features, a good linearity index is achieved along the whole estimation history. So, the inverse depth coding is suitable for both low and high parallax cases if the feature is continuously observed.

The *XYZ* encoding has low computational cost, but achieves linearity only at low depth uncertainty and high parallax. In Section VI, we explain how the representation of a feature can be switched over such that the inverse depth parametrization is only used when needed—for features that are either just initialized or at extreme depths.

V. Feature Initialization

From just a single observation, no feature depth can be estimated (although it would be possible in principle to impose a very weak depth prior by knowledge of the type of scene observed). What we do is to assign a general Gaussian prior in inverse depth that encodes probabilistically the fact that the point has to be in front of the camera. Hence, due to the linearity of inverse depth at low parallax, the filter can be initialized from just one observation. Experimental tuning has shown that infinity should be included with reasonable probability within the initialization prior, despite the fact that this means that depth estimates can become negative. Once initialized, features are processed with the standard EKF prediction-update loop—even in the case of negative inverse depth estimates—and immediately contribute to camera location estimation within SLAM.

It is worth noting that while a feature retains low parallax, it will automatically be used mainly to determine the camera orientation. The feature's depth will remain uncertain with the hypothesis of infinity still under consideration (represented by the probability mass corresponding to negative inverse depths). If the camera translates to produce enough parallax, then the feature's depth estimation will be improved and it will begin to contribute more to the camera location estimation.

The initial location for a newly observed feature inserted into the state vector is

$$\hat{\mathbf{y}}\left(\hat{\mathbf{r}}^{WC}, \hat{\mathbf{q}}^{WC}, \mathbf{h}, \rho_0\right) = \left(\hat{x}_i \quad \hat{y}_i \quad \hat{z}_i \quad \hat{\theta}_i \quad \hat{\phi}_i \quad \hat{\rho}_i\right)^\top \tag{21}$$

a function of the current camera pose estimate $\hat{\mathbf{r}}^{WC}, \hat{\mathbf{q}}^{WC}$, the image observation $\mathbf{h} = (\,u \quad v\,)^\top$, and the parameters determining the depth prior ρ_0, σ_ρ.

The endpoint of the initialization ray (see Fig. 1) is taken from the current camera location estimate

$$\left(\hat{x}_i \quad \hat{y}_i \quad \hat{z}_i\right)^\top = \hat{\mathbf{r}}^{WC} \tag{22}$$

and the direction of the ray is computed from the observed point, expressed in the world coordinate frame

$$\mathbf{h}^W = \mathbf{R}_{WC}\left(\mathbf{q}^{\hat{W}C}\right)\left(\upsilon \quad \nu \quad 1\right)^\top \tag{23}$$

where υ and ν are normalized retina image coordinates. Despite $\mathbf{h}^W$ being a nonunit directional vector, the angles by which we parametrize its direction can be calculated as

$$\begin{pmatrix} \theta_i \\ \phi_i \end{pmatrix} = \begin{pmatrix} \arctan\left(\mathbf{h}_x^W, \mathbf{h}_z^W\right) \\ \arctan\left(-\mathbf{h}_y^W, \sqrt{\mathbf{h}_x^{W\,2} + \mathbf{h}_z^{W\,2}}\right) \end{pmatrix}. \tag{24}$$

The covariance of $\hat{x}_i, \hat{y}_i, \hat{z}_i, \hat{\theta}_i$, and $\hat{\phi}_i$ is derived from the image measurement error covariance $\mathbf{R}_i$ and the state covariance estimate $\hat{\mathbf{P}}_{k|k}$.

The initial value for ρ_0 and its standard deviation are set empirically such that the 95% confidence region spans a range of depths from close to the camera up to infinity. In our experiments, we set $\hat{\rho}_0 = 0.1, \sigma_\rho = 0.5$, which gives an inverse depth confidence region $[1.1, -0.9]$. Notice that infinity is included in this range. Experimental validation has shown that the precise values of these parameters are relatively unimportant to the accurate operation of the filter as long as infinity is clearly included in the confidence interval.

The state covariance after feature initialization is

$$\hat{\mathbf{P}}^{\text{new}}_{k|k} = \mathbf{J} \begin{pmatrix} \hat{\mathbf{P}}_{k|k} & 0 & 0 \\ 0 & \mathbf{R}_i & 0 \\ 0 & 0 & \sigma_\rho^2 \end{pmatrix} \mathbf{J}^\top \tag{25}$$

$$\mathbf{J} = \left(\begin{array}{c|c} I & 0 \\ \hline \frac{\partial \mathbf{y}}{\partial \mathbf{r}^{WC}}, \frac{\partial \mathbf{y}}{\partial \mathbf{q}^{WC}}, 0, \ldots, 0, & \frac{\partial \mathbf{y}}{\partial \mathbf{h}}, \frac{\partial \mathbf{y}}{\partial \rho} \end{array} \right). \tag{26}$$

The inherent scale ambiguity in a monocular SLAM has usually been fixed by observing some known initial features that fix the scale (e.g., [8]). A very interesting experimental observation we have made using the inverse depth scheme is that sequential monocular SLAM can operate successfully without *any* known features in the scene, and in fact, the experiments we present in this paper do not use an initialization target. In this case, of course, the overall scale of the reconstruction and camera motion is undetermined, although with the formulation of the current paper, the estimation will settle on a (meaningless) scale of some value. In a very recent work [6], we have investigated this issue with a new dimensionless formulation of monocular SLAM.

VI. Switching From Inverse Depth to *XYZ*

While the inverse depth encoding can be used at both low and high parallax, it is advantageous for reasons of computational efficiency to restrict inverse depth to cases where the *XYZ* encoding exhibits nonlinearity according to the L_d index. This section details switching from inverse depth to *XYZ* for high parallax features.

A. Conversion From Inverse Depth to XYZ Coding

After each estimation step, the linearity index L_d (15) is computed for every map feature coded in inverse depth

$$\mathbf{h}^W_{XYZ} = \hat{\mathbf{x}}_i - \hat{\mathbf{r}}^{WC} \qquad \sigma_d = \frac{\sigma_\rho}{\rho_i^2} \qquad \sigma_\rho = \sqrt{\mathbf{P}_{\mathbf{y}_i\mathbf{y}_i}(6,6)}$$
$$d_i = \left\|\mathbf{h}^W_{XYZ}\right\| \qquad \cos\alpha = \mathbf{m}^\top \mathbf{h}^W_{XYZ} \left\|\mathbf{h}^W_{XYZ}\right\|^{-1}. \tag{27}$$

where $\hat{\mathbf{x}}_i$ is computed using (4) and $\mathbf{P}_{\mathbf{y}_i\mathbf{y}_i}$ is the submatrix 6×6 covariance matrix corresponding to the considered feature.

If L_d is below a switching threshold, the feature is transformed using (4) and the *full state* covariance matrix $\mathbf{P}$ is transformed with the corresponding Jacobian:

$$\mathbf{P}_{\text{new}} = \mathbf{JPJ}^\top \qquad \mathbf{J} = \operatorname{diag}\left(\mathbf{I}, \frac{\partial \mathbf{x}_i}{\partial \mathbf{y}_i}, \mathbf{I}\right). \tag{28}$$

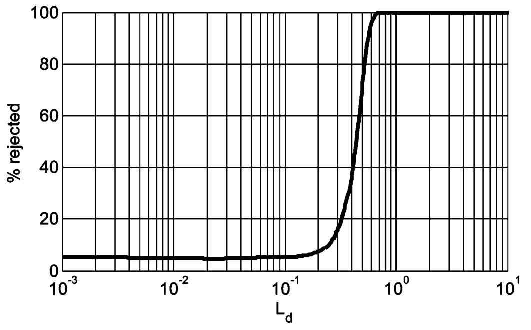


Fig. 4. Percentage of test rejections as a function of the linearity index L_d.

B. Linearity Index Threshold

We propose to use index L_d (15) to define a threshold for switching from inverse depth to *XYZ* encoding at the point when the latter can be considered linear. If the *XYZ* representation is linear, then the measurement u is Gaussian distributed (10), i.e.,

$$u \sim N\left(\mu_u, \sigma_u^2\right) \qquad \mu_u = 0 \qquad \sigma_u^2 = \left(\frac{\sin\alpha}{d_1}\right)^2 \sigma_d^2. \tag{29}$$

To determine the threshold in L_d that signals a lack of linearity in the measurement equation, a simulation experiment has been performed. The goal was to generate samples from the uncertain distribution for variable u, and then, apply a standard Kolmogorov–Smirnov Gaussianty [4] test to these samples, counting the percentage of rejected hypotheses h. When u is effectively Gaussian, the percentage should match the test significance level α_{sl} (5% in our experiments); as the number of rejected hypotheses increases, the measurement equation departs from linearity. A plot of the percentage of rejected hypotheses h with respect to the linearity index L_d is shown in Fig. 4. It can be clearly seen than when $L_d > 0.2$, h sharply departs from 5%. So, we propose the $L_d < 10\%$ threshold for switching from inverse depth to *XYZ* encoding.

Notice that the plot in Fig. 4 is smooth (log scale in L_d), which indicates that the linearity index effectively represents the departure from linearity.

The simulation has been performed for a variety of values of α, d_1, and σ_d; more precisely, all triplets resulting from the following parameter values:

$$\alpha(\text{deg}) \in \{0.1, 1, 3, 5, 7, 10, 20, 30, 40, 50, 60, 70\}$$
$$d_1(\text{m}) \in \{1, 3, 5, 7, 10, 20, 50, 100\}$$
$$\sigma_d(\text{m}) \in \{0.05, 0.1, 0.25, 0.5, 0.75, 1, 2, 5\}.$$

The simulation algorithm detailed in Fig. 5 is applied to every triplet $\{\alpha, d_1, \sigma_d\}$ to count the percentage of rejected hypotheses h and the corresponding linearity index L_d.

VII. Experimental Results

The performance of the new parametrization has been tested on real-image sequences acquired with a handheld-low-cost Unibrain IEEE1394 camera, with a 90° field of view and

```
input: $\alpha, d_1, \sigma_d$
output: $h$, $L_d$
$\sigma_u = \left|\frac{\sin\alpha}{d_1}\right|\sigma_d$; $\mu_u = 0$; //(29)
$\alpha_{sl} = 0.05$; // Kolm. test sign. level
$L_d = \frac{4\sigma_d}{d_1}|\cos\alpha|$
n_rejected=0 ;
N_GENERATED_SAMPLES=1000;
SAMPLE_SIZE=1000;

for j=1 to N_GENERATED_SAMPLES repeat
    $\{d_i\}_j$=random_normal(0,$\sigma_d^2$,SAMPLE_SIZE);
        //generate a normal sample from $N\left(0,\sigma_d^2\right)$;
    $\{u_i\}_j$=propagate_from_dept_to_image($\{d_i\}_j$,$\alpha$,$d_1$);//(14)
    if rejected==Kolmogorov_Smirnov($\{u_i\}_j$,$\mu_u$,$\sigma_u$,$\alpha_{sl}$)
        n_rejected=n_rejected+1;
endfor
$h$=100$\frac{[n\_rejected]}{[N\_GENERATED\_SAMPLES]}$;
```

Fig. 5. Simulation algorithm to test the linearity of the measurement equation.

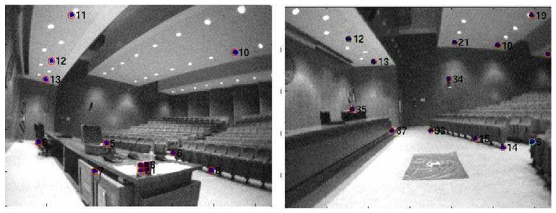

(a) (b)

Fig. 6. First (a) and last (b) frame in the sequence of the indoor experiment of Section VII-A. Features 11, 12, and 13 are analyzed. These features are initialized in the same frame but are located at different distances from the camera.

320×240 resolution, capturing monochrome image sequences at 30 fps.

Five experiments were performed. The first was an indoor sequence processed offline with a Matlab implementation, the goal being to analyze initialization of scene features located at different depths. The second experiment shows an outdoor sequence processed in real time with a C++ implementation. The focus was on distant features observed under low parallax along the whole sequence. The third experiment was a loop closing sequence, concentrating on camera covariance evolution. Fourth was a simulation experiment to analyze the effect of switching from inverse depth to *XYZ* representations. In the last experiment, the switching performance was verified on the real loop closing sequence. This section ends with a computing time analysis. It is worth noting that no initial pattern to fix the scale was used in any of the last three experiments.

A. Indoor Sequence

This experiment analyzes the performance of the inverse depth scheme as several features at a range of depths are tracked within SLAM. We discuss three features, which are all detected in the same frame but have very different depths. Fig. 6 shows the image where the analyzed features are initialized (frame 18 in the sequence) and the last image in the sequence. Fig. 7 focuses on the evolution of the estimates corresponding to the features, with labels 11, 12, and 13, at frames 1, 10, 25, and 100. Confidence regions derived from the inverse depth representation (thick red line) are plotted in the *XYZ* space by numerical Monte Carlo propagation from the 6-D multivariate Gaussians representing these features in the SLAM EKF. For comparison, standard Gaussian *XYZ* acceptance ellipsoids (thin black line) are linearly propagated from the 6-D representation by means of the Jacobian of (28). The parallax α in degrees for each feature at every step is also displayed.

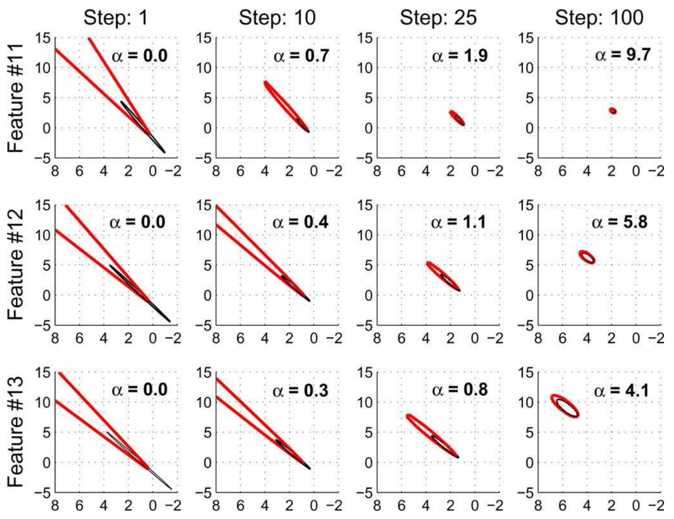


Fig. 7. Feature initialization. Each column shows the estimation history for a feature horizontal components. For each feature, the estimates after 1, 10, 25, and 100 frames since initialization are plotted; the parallax angle α in degrees between the initial observation and the current frame is displayed. The thick (red) lines show (calculated by a Monte Carlo numerical simulation) the 95% confidence region when coded as Gaussian in inverse depth. The thin (black) ellipsoids show the uncertainty as a Gaussian in the *XYZ* space propagated according to (28). Notice how at low parallax, the inverse depth confidence region is very different from the elliptical Gaussian. However, as the parallax increases, the uncertainty reduces and collapses to the Gaussian ellipse.

When initialized, the 95% acceptance region of all the features includes $\rho = 0$, so infinite depth is considered as a possibility. The corresponding confidence region in depth is highly asymmetric, excluding low depths but extending to infinity. It is clear that Gaussianity in inverse depth is not mapped to Gaussianity in *XYZ*, so the black ellipsoids produced by Jacobian transformation are far from representing the true depth uncertainty. As stated in Section IV-D, it is at low parallax that the inverse depth parametrization plays a key role.

As rays producing bigger parallax are gathered, the uncertainty in ρ becomes smaller but still maps to a nonGaussian distribution in *XYZ*. Eventually, at high parallax, for all of the features, the red confidence regions become closely Gaussian and well approximated by the linearly propagated black ellipses—but this happens much sooner for nearby feature 11 than distant feature 13.

A movie showing the input sequence and estimation history of this experiment is available as multimedia data `inverseDepth_indoor.avi`. The raw input image sequence is also available at `inverseDepth_indoorRawImages.tar.gz`.

B. Real-Time Outdoor Sequence

This 860 frame experiment was performed with a C++ implementation that achieves real-time performance

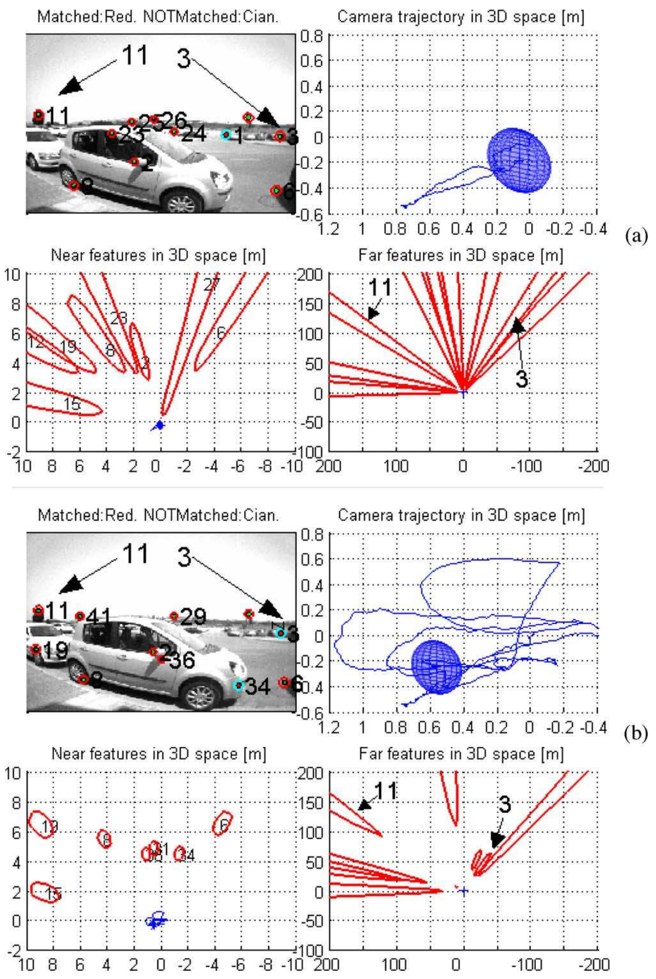


Fig. 8. (a) and (b) show frames #163 and #807 from the outdoor experiment of Section VII-B. This experiment was processed in real time. The focus was two features: 11 (tree on the left) and 3 (car on the right) at low parallax. Each of the two figures shows the current images and top-down views illustrating the horizontal components of the estimation of camera and feature locations at three different zoom scales for clarity: the top-right plots (maximum zoom) highlight the estimation of the camera motion; bottom-left (medium zoom) views highlight nearby features; and bottom-right (minimum zoom) emphasizes distant features.

at 30 fps with the handheld camera. Here, we highlight the ability of our parametrization to deal with both close and distant features in an outdoor setting. The input image sequence is available at multimedia material `inverseDepth_outdoorRawImages.tar.gz`. A movie showing the estimation process is also available at `inverseDepth_outdoor.avi`.

Fig. 8 shows two frames of the movie illustrating the performance. For most of the features, the camera ended up gathering enough parallax to accurately estimate their depths. However, being outdoors, there were distant features producing low parallax during the whole camera motion.

The inverse depth estimation history for two features is highlighted in Fig. 9. It is shown that distant, low parallax features are persistently tracked through the sequence, despite the fact that their depths cannot be precisely estimated. The large depth uncertainty, represented with the inverse depth scheme, is

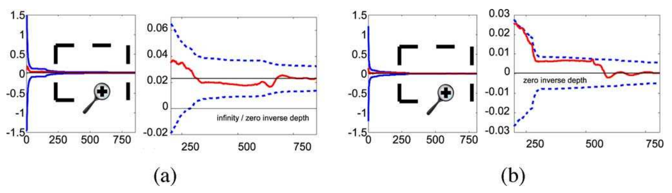


Fig. 9. Analysis of outdoor experiment of Section VII-B. (a) Inverse depth estimation history for feature 3, on the car, and (b) for feature 11, on a distant tree. Due to the uncertainty reduction during estimation, two plots at different scales are shown for each feature. It shows the 95% confidence region, and with a thick line, the estimated inverse depth. The thin solid line is the inverse depth estimated after processing the whole sequence. In (a), for the first 250 steps, zero inverse depth is included in the confidence region, meaning that the feature might be at infinity. After this, more distant but finite locations are gradually eliminated, and eventually, the feature's depth is accurately estimated. In (b), the tree is so distant that the confidence region always includes zero since little parallax is gathered for that feature.

successfully managed by the SLAM EKF, allowing the orientation information supplied by these features to be exploited.

Feature 3, on a nearby car, eventually gathers enough parallax enough to have an accurate depth estimate after 250 images, where infinite depth is considered as a possibility. Meanwhile, the estimate of feature 11, on a distant tree and never displaying significant parallax, never collapses in this way and zero inverse depth remains within its confidence region. Delayed intialization schemes would have discarded this feature without obtaining any information from it, while in our system, it behaves like a bearing reference. This ability to deal with distant points in real time is a highly advantageous quality of our parametrization. Note that what does happen to the estimate of feature 11 as translation occurs is that hypotheses of nearby depths are ruled out—the inverse depth scheme correctly recognizes that measuring little parallax while the camera has translated some distance allows a minimum depth for the feature to be set.

C. *Loop Closing Sequence*

A loop closing sequence offers a challenging benchmark for any SLAM algorithm. In this experiment, a handheld camera was carried by a person walking in small circles within a very large student laboratory, carrying out two complete laps. The raw input image sequence is available at `inverseDepth_loopClosingRawImages.tar.gz`, and a movie showing the mapping process at `inverseDepth_loopClosing.avi`.

Fig. 10 shows a selection of the 737 frames from the sequence, concentrating on the beginning, first loop closure, and end of the sequence. Fig. 11 shows the camera location estimate covariance history, represented by the 95% confidence regions for the six camera DOF and expressed in a reference local to the camera.

We observe the following properties of the evolution of the estimation, focussing, in particular, on the uncertainty in the camera location.

1) After processing the first few images, the uncertainty in the depth of features is huge, with highly nonelliptical confidence regions in the *XYZ* space [Fig. 10(a)].
2) In Fig. 11, the first peak in the *X* and *Z* translation uncertainty corresponds to a camera motion backward along

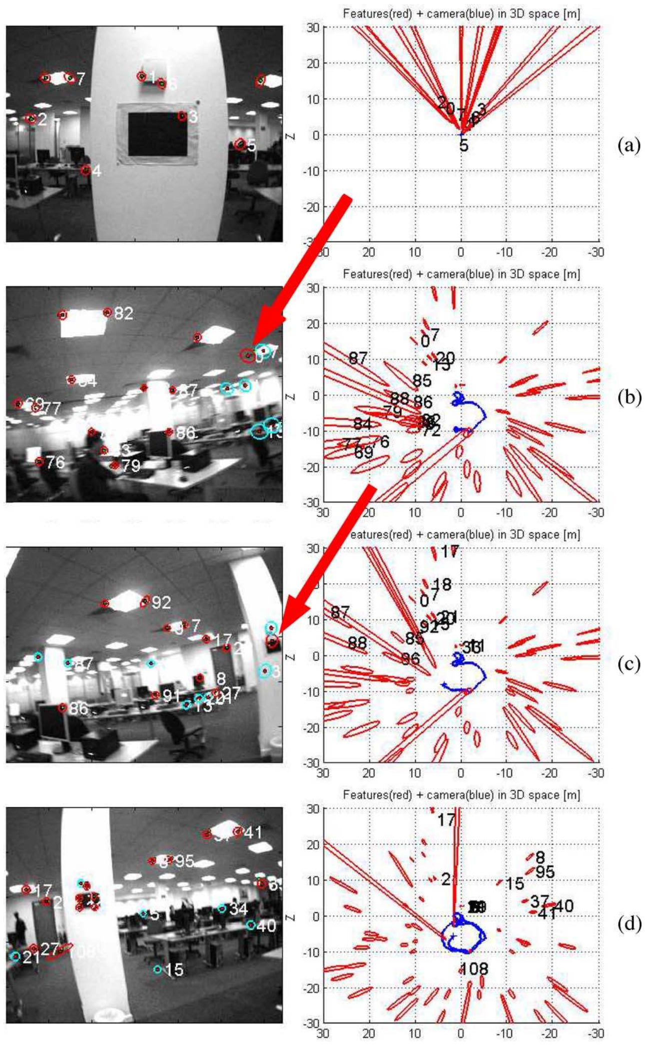


Fig. 10. Selection of frames from the loop closing experiment of Section VII-C. For each frame, we show the current image and reprojected map (left), and a top-down view of the map with 95% confidence regions and camera trajectory (right). Notice that confidence regions for the map features are far from being Gaussian ellipses, especially for newly initialized or distant features. The selected frames are: (a) #11, close to the start of the sequence; (b) #417, where the first loop closing match, corresponding to a distant feature, is detected; the loop closing match is signaled with an arrow; (c) #441, where the first loop closing match corresponding to a close feature is detected; the match is signaled with an arrow; and (d) #737, the last image, in the sequence, after reobserving most of the map features during the second lap around the loop.

the optical axis; this motion produces poor parallax for newly initialized features, and we, therefore, see a reduction in the orientation uncertainty and an increase in the translation uncertainty. After frame #50, the camera again translates in the *X*-direction, parallax is gathered, and the translation uncertainty is reduced.

3) From frame #240, the camera starts a 360° circular motion in the *XZ* plane. The camera explores new scene regions, and the covariance increases steadily as expected (Fig. 11).
4) In frame #417, the first loop closing feature is reobserved. This is a feature that is distant from the camera, and causes an abrupt reduction in the orientation and translation uncertainty (Fig. 11), though a medium level of uncertainty remains.

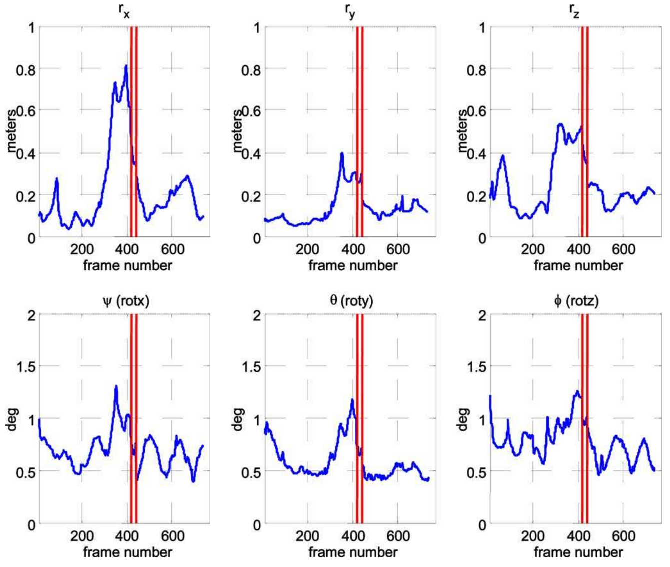


Fig. 11. Camera location estimate covariance along the sequence. The 95% confidence regions for each of the 6 DOF of camera motion are plotted. Note that errors are expressed in a reference local to the camera. The vertical solid lines indicate the loop closing frames #417 and #441.

5) In frame #441, a much closer loop closing feature (mapped with high parallax) is matched. Another abrupt covariance reduction takes place (Fig. 11) with the extra information this provides.
6) After frame #441, as the camera goes on a second lap around the loop, most of the map features are revisited, almost no new features are initalized, and hence, the uncertainty in the map is further reduced. Comparing the map at frame #441 (the beginning of the second lap) and #737, (the end of the second lap), we see a significant reduction in uncertainty. During the second lap, the camera uncertainty is low, and as features are reobserved, their uncertainties are noticeably reduced [Fig. 10(c) and (d)].

Note that these loop closing results with the inverse depth representation show a marked improvement on the experiments on monocular SLAM with a humanoid robot presented in [9], where a gyro was needed in order to reduce angular uncertainty enough to close loops with very similar camera motions.

D. *Simulation Analysis for Inverse Depth to XYZ Switching*

In order to analyze the effect of the parametrization switching proposed in Section VI on the consistency of SLAM estimation, simulation experiments with different switching thresholds were run. In the simulations, a camera completed two laps of a circular trajectory of radius 3 m in the XZ plane, looking out radially at a scene composed of points lying on three concentric spheres of radius 4.3, 10, and 20 m. These points at different depths were intended to produce observations with a range of parallax angles (Fig. 12).

The camera parameters of the simulation correspond with our real image acquisition system: camera 320×240 pixels, frame rate 30 frames/s, image field of view 90°, measurement uncertainty for a point feature in the image, and Gaussian

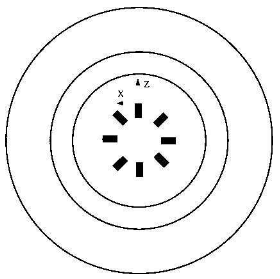


Fig. 12. Simulation configuration for analysis of parametrization switching in Section VII-D, sketching the circular camera trajectory and 3-D scene, composed of three concentric spheres of radius 4.3, 10, and 20 m. The camera completes two circular laps in the (XZ)-plane with radius 3 m, and is orientated radially.

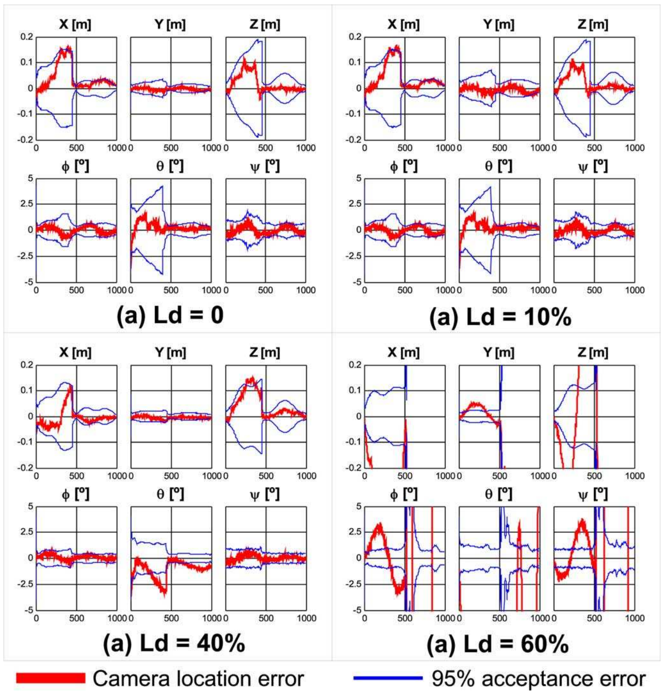


Fig. 13. Details from the parametrization switching experiment. Camera location estimation error history in 6 DOF. (translation in XYZ, and three orientation angles $\psi\theta\phi$) for four switching thresholds: With $L_d = 0\%$, no switching occurs and the features all remain in the inverse depth parametrization. At $L_d = 10\%$, although features from the spheres at 4.3 and 10 m are eventually converted, no degradation with respect to the non-switching case is observed. At $L_d = 40\%$, some features are switched before achieving true Gaussianity, and there is noticeable degradation, especially in θ rotation around the Y axis. At $L_d = 60\%$, the map becomes totally inconsistent and loop closing fails.

$N\left(0, 1 \text{ pixel}^2\right)$. The simulated image sequence contained 1000 frames. Features were selected following the randomized map management algorithm proposed in [8] in order to have 15 features visible in the image at all times. All our simulation experiments work using the same scene features in order to homogenize the comparison.

Four simulation experiments for different thresholds for switching each feature from inverse depth to *XYZ* parametrization were run with $L_d \in \{0\%, 10\%, 40\%, 60\%\}$. Fig. 13 shows the camera trajectory estimation history in 6 DOF (translation in XYZ, and three orientation angles $\psi(\mathrm{Rot}_x), \theta(\mathrm{Rot}_y), \phi(\mathrm{Rot}_z, \mathrm{cyclotorsion})$). The following conclusions can be made.

1) Almost the same performance is achieved with no switching (0%) and with 10% switching. So, it is clearly advantageous to perform 10% switching because there is no penalization in accuracy and the state vector size of each converted feature is halved.
2) Switching too early degrades accuracy, especially in the orientation estimate. Notice how for 40% the orientation estimate is worse and the orientation error covariance is smaller, showing filter inconsistency. For 60%, the estimation is totally inconsistent and the loop closing fails.
3) Since early switching degrades performance, the inverse depth parametrization is mandatory for initialization of every feature and over the long term for low parallax features.

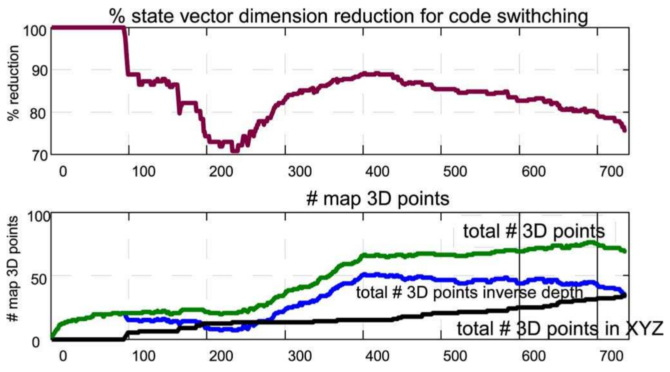


Fig. 14. Parametrization switching on a real sequence (Section VII-E): state vector size history. Top: percentage reduction in state dimension when using switching compared with keeping all points in inverse depth. Bottom: total number of points in the map, showing the number of points in inverse depth and the number of points in *XYZ*.

E. *Parametrization Switching With Real Images*

The loop closing sequence of Section VII-C was processed without any parametrization switching, and with switching at $L_d = 10\%$. A movie showing the results is available at `inverseDepth_loopClosing_ID_to_XYZ_conversion.avi`. As in the simulation experiments, no significant change was noticed in the estimated trajectory or map.

Fig. 14 shows the history of the state size, the number of map features, and how their parametrization evolves. At the last estimation step, about half of the features had been switched; at this step, the state size had reduced from 427 to 322 (34 inverse depth features and 35 *XYZ*), i.e., 75% of the original vector size. Fig. 15 shows four frames from the sequence illustrating feature switching. Up to step 100, the camera has low translation and all the features are in inverse depth form. As the camera translates, nearby features switch to *XYZ*. Around step 420, the loop is closed and features are reobserved, producing a significant reduction in uncertainty that allows switching of more reobserved close features. Our method automatically determines which features should be represented in the inverse depth or *XYZ* forms, optimizing computational efficiency without sacrificing accuracy.

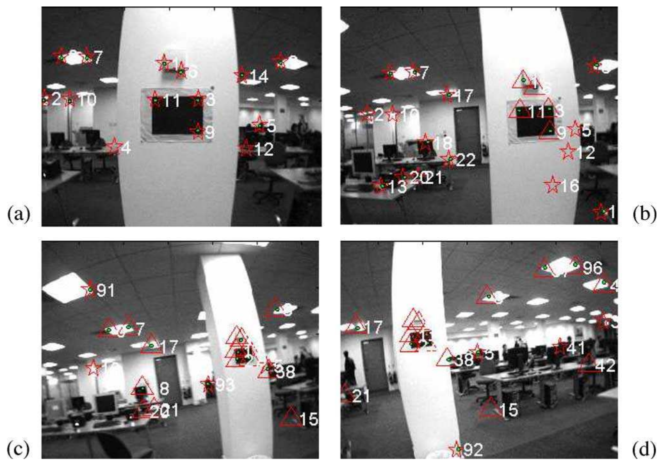


Fig. 15. Parametrization switching seen in image space: points coded in inverse depth ($\star$) and coded in *XYZ* ($\triangle$). (a) First frame, with all features in inverse depth. (b) Frame #100; nearby features start switching. (c) Frame #470, loop closing; most features in *XYZ*. (d) Last image of the sequence.

F. Processing Time

We give some details of the real-time operation of our monocular SLAM system, running on a 1.8 GHz Pentium M processor laptop. A typical EKF iteration would implies the following.

1) A state vector dimension of 300.
2) 12 features observed in the image, a measurement dimension of 24.
3) 30 fps, so 33.3 ms available for processing.

Typical computing time breaks down as follows: image acquisition, 1 ms.; EKF prediction, 2 ms; image matching, 1 ms.; and EKF update, 17 ms. This adds up to a total of 21 ms. The remaining time is used for graphics functions using OpenGL on an NVidia card and scheduled at a low priority.

The quoted state vector size 300 corresponds to a map size of 50 if all features are encoded using inverse depth. In indoor scenes, due to switching maps of up to 60–70, features can be computed in real time. This size is enough to map many typical scenes robustly.

VIII. Conclusion

We have presented a parametrization for monocular SLAM that permits operation based uniquely on the standard EKF prediction-update procedure at every step, unifying initialization with the tracking of mapped features. Our inverse depth parametrization for 3-D points allows unified modeling and processing for any point in the scene, close or distant, or even at "infinity." In fact, close, distant, or just-initialized features are processed within the routine EKF prediction-update loop without making any binary decisions. Due to the undelayed initialization and immediate full use of infinite points, estimates of camera orientation are significantly improved, reducing the camera estimation jitter often reported in previous work. The jitter reduction, in turn, leads to computational benefits in terms of smaller search regions and improved image processing speed.

The key factor is that due to our parametrization of the direction and inverse depth of a point relative to the location from which it was first seen, our measurement equation has low linearization errors at low parallax, and hence, the estimation uncertainty is accurately modeled with a multivariate Gaussian. In Section IV, we presented a model that quantifies linearization error. This provides a theoretical understanding of the impressive outdoor, real-time performance of the EKF with our parametrization.

The inverse depth representation requires a 6-D state vector per feature, compared to three for *XYZ* coding. This doubles the map state vector size, and hence produces a fourfold increase in the computational cost of the EKF. Our experiments show that it is essential to retain the inverse depth parametrization for intialization and distant features, but nearby features can be safely converted to the cheaper *XYZ* representation, meaning that the long-term computational cost need not increase significantly. We have given details on when this conversion should be carried out for each feature to optimize computational efficiency without sacrificing accuracy.

The experiments presented have validated the method with real imagery using a handheld camera as the only sensor, both indoors and outdoors. We have experimentally verified the following the key contributions of our study:

1) real-time performance achieving 30 fps real-time processing for maps up to 60–70 features;
2) real-time loop closing;
3) dealing simultaneously with low and high parallax features;
4) nondelayed initialization;
5) low jitter, full 6-DOF monocular SLAM.

In the experiments, we have focused on a map size around 60–100 features because these map sizes can be dealt with in real time at 30 Hz, and we have focused on the challenging loop closing issue. Useful future work would be a thorough analysis of the limiting factors in EKF inverse depth monocular SLAM in terms of linearity, data association errors, accuracy, map size, and ability to deal with degenerate motion such as pure rotations or a static camera for long-time periods.

Finally, our simulations and experiments have shown that inverse depth monocular SLAM operates well without known patterns in the scene to fix scale. This result points toward further work in understanding the role of scale in monocular SLAM (an avenue that we have begun to investigate in a dimensionless formulation in [6]) and further bridging the gap between sequential SLAM techniques and structure from motion methods from the computer vision literature.

Appendix

To recover the ideal projective undistorted coordinates $\mathbf{h}_u = (u_u, v_u)^\top$ from the actually distorted ones gathered by the camera $\mathbf{h}_d = (u_d, v_d)^\top$, the classical two parameters radial distortion model [17] is applied:

$$\begin{pmatrix} u_u \\ v_u \end{pmatrix} = \mathbf{h}_u \begin{pmatrix} u_d \\ v_d \end{pmatrix} = \begin{pmatrix} u_0 + (u_d - u_0)\left(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4\right) \\ v_0 + (v_d - v_0)\left(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4\right) \end{pmatrix}$$

$$r_d = \sqrt{\left(d_x(u_d - u_0)\right)^2 + \left(d_y(v_d - v_0)\right)^2} \qquad (30)$$

where u_0, v_0 are the image centers and κ_1, κ_2 are the radial distortion coefficients.

To compute the distorted coordinates from the undistorted

$$\begin{pmatrix} u_d \\ v_d \end{pmatrix} = \mathbf{h}_d \begin{pmatrix} u_u \\ v_u \end{pmatrix} = \begin{pmatrix} u_0 + \dfrac{(u_u - u_0)}{(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4)} \\ v_0 + \dfrac{(v_u - v_0)}{(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4)} \end{pmatrix} \tag{31}$$

$$r_u = r_d \left(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4\right) \tag{32}$$

$$r_u = \sqrt{\left(d_x (u_u - u_0)\right)^2 + \left(d_y (v_u - v_0)\right)^2} \tag{33}$$

where r_u is readily computed computed from (33), but r_d has to be numerically solved from (32), e.g, using Newton–Raphson method; hence, (31) can be used to compute the distorted point.

Undistortion Jacobian $\partial \mathbf{h}_u / \partial (u_d, v_d)$ has the following analytical expression:

$$\left(\begin{array}{c|c} \begin{array}{c}\left(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4\right) + \\ 2\left((u_d - u_0) d_x\right)^2 \times \\ \left(\kappa_i + 2\kappa_2 r_d^2\right)\end{array} & \begin{array}{c}2 d_y^2 (u_d - u_0)(v_d - v_0) \times \\ \left(\kappa_1 + 2\kappa_2 r_d^2\right)\end{array} \\ \hline \begin{array}{c}2 d_x^2 (v_d - v_0)(u_d - u_0) \times \\ \left(\kappa_1 + 2\kappa_2 r_d^2\right)\end{array} & \begin{array}{c}\left(1 + \kappa_1 r_d^2 + \kappa_2 r_d^4\right) + \\ 2\left((v_d - v_0) d_y\right)^2 \times \\ \left(\kappa_i + 2\kappa_2 r_d^2\right)\end{array} \end{array}\right) \tag{34}$$

The Jacobian for the distortion is computed by inverting expression (34)

$$\left.\frac{\partial \mathbf{h}_d}{\partial (u_u, v_u)}\right|_{(u_u, v_u)} = \left(\left.\frac{\partial \mathbf{h}_u}{\partial (u_d, v_d)}\right|_{\mathbf{h}_d(u_u, v_u)}\right)^{-1}. \tag{35}$$

Acknowledgment

The authors are very grateful to D. Murray, I. Reid, and other members of Oxford's Active Vision Laboratory for discussions and software collaboration. They also thank the anonymous reviewers for their useful comments.

References

[1] V. J. Aidala and S. E. Hammel, "Utilization of modified polar coordinates for bearing-only tracking," *IEEE Trans. Autom. Control*, vol. 28, no. 3, pp. 283–294, Mar. 1983.

[2] T. Bailey, "Constrained initialisation for bearing-only SLAM," in *Proc. IEEE Int. Conf. Robot. Autom.*, Taipei, Taiwan, R.O.C., Sep. 14–19, 2003, vol. 2, pp. 1966–1971.

[3] M. Bryson and S. Sukkarieh, "Bearing-only SLAM for an airborne vehicle," presented at the Australasian Conf. Robot. Autom. (ACRA 2005), Sydney, Australia.

[4] G. C. Canavos, *Applied Probability and Statistical Methods*. Boston, MA: Little, Brow, 1984.

[5] A. Chowdhury and R. Chellappa, "Stochastic approximation and rate-distortion analysis for robust structure and motion estimation," *Int. J. Comput. Vis.*, vol. 55, no. 1, pp. 27–53, 2003.

[6] J. Civera, A. J. Davison, and J. M. M. Montiel, "Dimensionless monocular SLAM," in *Proc. 3rd Iberian Conf. Pattern Recogn. Image Anal.*, 2007, pp. 412–419.

[7] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth to depth conversion for monocular SLAM," in *Proc. Int. Conf. Robot. Autom.*, 2007, pp. 2778–2783.

[8] A. Davison, "Real-time simultaneous localization and mapping with a single camera," in *Proc. Int. Conf. Comput. Vis.*, Oct. 2003, pp. 1403–1410.

[9] A. J. Davison, I. Reid, N. Molton, and O. Stasse, "Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[10] E. Eade and T. Drummond, "Scalable monocular SLAM," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.* Jun. 17–22, 2006, vol. 1, pp. 469–476.

[11] A. W. Fitzgibbon and A. Zisserman, "Automatic camera recovery for closed or open image sequences," in *Proc. Eur. Conf. Comput. Vis.*, Jun. 1998, pp. 311–326.

[12] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[13] D. Heeger and A. Jepson, "Subspace methods for recovering rigid motion I: Algorithm and implementation," *Int. J. Comput. Vis.*, vol. 7, no. 2, pp. 95–117, Jan. 1992.

[14] J. H. Kim and S. Sukkarieh, "Airborne simultaneous localisation and map building," in *Proc. IEEE Int. Conf. Robot. Autom.* Sep. 14–19, 2003, vol. 1, pp. 406–411.

[15] N. Kwok and G. Dissanayake, "An efficient multiple hypothesis filter for bearing-only SLAM," in *Proc. IROS*, 28 Sep.–2–Oct. 2004, vol. 1, pp. 736–741.

[16] L. Matthies, T. Kanade, and R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *Int. J. Comput. Vis.*, vol. 3, no. 3, pp. 209–238, 1989.

[17] E. Mikhail, J. Bethel, and J. C. McGlone, *Introduction to Modern Photogrammetry*. New York: Wiley, 2001.

[18] J. Montiel, J. Civera, and A. J. Davison, "Unified inverse depth parametrization for monocular SLAM," presented at the Robot. Sci. Syst. Conf., Philadelphia, PA, Aug. 2006.

[19] J. Montiel and A. J. Davison, "A visual compass based on SLAM," in *Proc. Int. Conf. Robot. Autom.*, Orlando, FL, May 15–19, 2006, pp. 1917–1922.

[20] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real-time localization and 3-D reconstruction," in *Proc. IEEE Conf. Comput. Vis. Pattern Recogn.*, 2006, vol. 3, pp. 1027–1031.

[21] D. Nister, O. Naroditsky, and J. Bergen, "Visual odometry for ground vehicle applications," *J. Field Robot.*, vol. 23, no. 1, pp. 3–26, 2006.

[22] M. Okutomi and T. Kanade, "A multiple-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 15, no. 4, pp. 353–363, Apr. 1993.

[23] M. Pollefeys, R. Koch, and L. Van Gool, "Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters," *Int. J. Comput. Vis.*, vol. 32, no. 1, pp. 7–25, 1999.

[24] J. Sola, "Towards visual localization, mapping and moving objects tracking by a mobile robot: A geometric and probabilistic approach," Ph.D. dissertation, LAAS-CNRS, Toulouse, France, 2007.

[25] J. Sola, A. Monin, M. Devy, and T. Lemaire, "Undelayed initialization in bearing only SLAM," in *Proc. 2005 IEEE/RSJ Int. Conf. Intell. Robots Syst.,* 2–6 Aug., 2005, pp. 2499–2504.

[26] N. Trawny and S. I. Roumeliotis, "A unified framework for nearby and distant landmarks in bearing-only SLAM," in *Proc. Int. Conf. Robot. Autom.,* May 15–19, 2006, pp. 1923–1929.

Javier Civera was born in Barcelona, Spain, in 1980. He received the M.S. degree in industrial-electrical engineering in 2004 from the University in Zaragoza, where he is currently working toward the Ph.D. degree with the Robotics, Perception, and Real-Time Group.

He is currently an Assistant Lecturer with the Departamento de Informática, University of Zaragoza, where he teaches courses in automatic control theory. His current research interests include computer vision and mobile robotics.

Andrew J. Davison received the B.A. degree in physics and the D.Phil. degree in computer vision from the University of Oxford, Oxford, U.K., in 1994 and 1998, respectively.

He was with Oxford's Robotics Research Group, where he developed one of the first robot simultaneous localization and mapping (SLAM) systems using vision. He was a European Union (EU) Science and Technology Fellow at the National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan, for two years, where he was engaged in visual robot navigation. In 2000, he returned to the University of Oxford as a Postdoctoral Researcher. In 2005, he joined Imperial College London, London, U.K., where he currently holds the position of Reader with the Department of Computing. His current research interests include advancing the basic technology of real-time localization and mapping using vision while collaborating to apply these techniques in robotics and related areas.

Dr. Davison was awarded a five-year Engineering and Physical Sciences Research Council (EPSRC) Advanced Research Fellowship in 2002. In 2008, he was awarded a European Research Council (ERC) Starting Grant.

J. M. Martínez Montiel was born in Arnedo, Spain. He received the M.S. and Ph.D. degrees in electrical engineering from the University of Zaragoza, Zaragoza, Spain, in 1991 and 1996, respectively.

He is currently an Associate Professor with the Departamento de Informática, University of Zaragoza, where he is in charge of Perception and Computer Vision courses. His current interests include computer vision, real-time vision localization and mapping research, and the transference of this technology to robotic and nonrobotic application domains.

Dr. Montiel is member of the the Robotics, Perception, and Real-Time Group. He has been awarded the Spanish Merrimack Education Center (MEC) grants to fund research at the University of Oxford, Oxford, U.K., and Imperial College London, London, U.K.

Large-Scale 6-DOF SLAM With Stereo-in-Hand

Lina M. Paz, *Member, IEEE*, Pedro Piniés, *Member, IEEE*, Juan D. Tardós, *Member, IEEE*, and José Neira, *Member, IEEE*

***Abstract*—In this paper, we describe a system that can carry out simultaneous localization and mapping (SLAM) in large indoor and outdoor environments using a stereo pair moving with 6 DOF as the only sensor. Unlike current visual SLAM systems that use either bearing-only monocular information or 3-D stereo information, our system accommodates both monocular and stereo. Textured point features are extracted from the images and stored as 3-D points if seen in both images with sufficient disparity, or stored as inverse depth points otherwise. This allows the system to map both near and far features: the first provide distance and orientation, and the second provide orientation information. Unlike other vision-only SLAM systems, stereo does not suffer from "scale drift" because of unobservability problems, and thus, no other information such as gyroscopes or accelerometers is required in our system. Our SLAM algorithm generates sequences of conditionally independent local maps that can share information related to the camera motion and common features being tracked. The system computes the full map using the novel conditionally independent divide and conquer algorithm, which allows constant time operation most of the time, with linear time updates to compute the full map. To demonstrate the robustness and scalability of our system, we show experimental results in indoor and outdoor urban environments of 210 m and 140 m loop trajectories, with the stereo camera being carried in hand by a person walking at normal walking speeds of 4–5 km/h.**

***Index Terms*—Linear time, scalability, stereo vision, visual SLAM.**

Manuscript received December 17, 2007; revised July 3, 2008. First published September 26, 2008; current version published October 31, 2008. This paper was recommended for publication by Associate Editor J. Leonard and Editor L. Parker upon evaluation of the reviewers' comments. This work was supported in part by the European Union under Project RAWSEEDS FP6-IST-045144 and in part by the Dirección General de Investigación of Spain under Project SLAM6DOF DPI2006-13578. A preliminary version of this work was presented at the *Visual SLAM: An Emerging Technology* workshop held at the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Diego, CA, 2008.

The authors are with the Department of Computer Science and Systems Engineering, Universidad de Zaragoza, Zaragoza E-50018, Spain (e-mail: linapaz@unizar.es; ppinies@unizar.es; tardos@unizar.es; jneira@unizar.es).

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the author. This material includes the following video files. *VSLAM_stereo_distribution.avi:* Suppose that a point is located in front of a stereo camera at different distances ranging from 1 to 20 m. The video shows the uncertainty distributions estimated by different methods when the point position is reconstructed from a noisy stereo pair observation. Cloud points are samples obtained from the real distribution, red ellipses represent uncertainties estimated using depth point parameterization, and the interior of the green regions represent the uncertainties estimated using inverse depth parameterization. It is clear from the video that the depth point parameterization is overconfident when the original point is far from the stereo. *VSLAM_local_map.avi:* In this video, the process of building a local map is shown. The video reproduces the sequence of images captured by the stereo as well as a top and lateral perspectives of the local map built. *VSLAM_outdoor.avi*, *VSLAM_indoor.avi:* To demonstrate the robustness and scalability of our system, we show experimental results in indoor and outdoor urban environments of 210- and 140-m loop trajectories, with the stereo camera being carried in hand by a person walking at normal walking speeds of 4–5 km/h. The colors used in the camera images are green for features predicted and matched, blue for predicted but not matched, and red for matchings rejected by JCBB. *Note:* The videos VSLAM_local_map.avi, VSLAM_outdoor.avi, VSLAM_indoor.avi, and VSLAM_stereo_distribution.avi have been tested on VLC media plater: http://www.videolan.org; Windows Media Player with xvid codec from: http//www.xvid.org/

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TRO.2008.2004637

I. Introduction: State-of-the Art in Visual SLAM

THE INTEREST in using cameras in simultaneous localization and mapping (SLAM) has grown tremendously in recent times. Cameras have become much more inexpensive than lasers, and also provide texture rich information about scene elements at practically any distance from the camera. 6-DOF SLAM systems based on 3-D laser scanners plus odometry have been demonstrated feasible both indoors and outdoors [2], [3], as well as vision aided by laser without odometry [4] and vision aided by an inertial navigation system [5], [6]. But in applications where it is not practical to carry heavy and bulky sensors, such as egomotion for people tracking and environment modeling in rescue operations, cameras seem the only light weight sensors that can be easily adapted to helmets used by rescuers, or simply worn.

Current visual SLAM research has been focused on the use of either monocular or stereo vision to obtain 3-D information from the environment. Quite a few monocular visual SLAM systems have been demonstrated to be viable for small environments [7]–[16]. Most are essentially standard extended Kalman filter (EKF) SLAM systems, and vary in the technique used to initialize a feature, given the partiality of the bearing only information provided by one camera, or in the type of interest points extracted from the images (be it Harris corners, Shi–Tomasi corners, scale-invariant feature transform (SIFT) features, or some combination). Some works have also considered segment features [17], [18]. Larger environments have been tackled in hierarchical visual SLAM [19].

A single camera is used in all of these systems, and although very distant features are potentially detectable, scale unobservability is a fundamental limitation. Either the scale is fixed in some way (for example, by observing a known object [16]), or drift in scale can occur as is reported in the hierarchical visual SLAM system [19]. Panoramic cameras are also being used in visual SLAM [20], [21]. Here, the limitation of scale unobservability is overcome using an additional stereo vision bench for motion estimation between consecutive frames. In the work of Royer *et at.* [22], only monocular images are used. Mapping is achieved using a batch hierarchical bundle adjustment algorithm to compute all camera as well as interest points locations. The scale is introduced in the system by manually entering the length of the path.

Stereo visual systems provide scale through the baseline between the cameras, known from calibration. Davison and Murray demonstrated the first active stereo visual SLAM system [23]–[25]. It is based on standard EKF, and thus, has low scalability also. Under restrictive planar environment assumptions, Iocchi *et al.* built an environment map using stereo [26]. Se *et al.* demonstrated a visual stereo SLAM system using SIFT features in a small laboratory environment [27]. This system is also unlikely to scale adequately to large environments or work in more challenging outdoor scenarios as cross-correlations were neglected for computational reasons. In [28] and [29], the authors demonstrate an autonomous blimp system for terrain mapping using stereo as the only sensor, also using a standard EKF SLAM algorithm. Saez *et al.* [30] presented a 6-DOF stereo visual SLAM system, where egomotion estimation is done by a 3-D point matching algorithm, and mapping through a global entropy minimization algorithm in indoor orthogonal scenarios, with difficult extension to more complex nonorthogonal environments.

In [31] and [32], Sim *et al.* describe a dense visual SLAM system using Rao–Blackwellized particle filters and SIFT features (a similar effort in using Rao–Blackwellized particle filters and SIFT features for visual SLAM was reported in [15]). Visual odometry [structure from motion (SFM)] is used to generate proposals for the sensor motion and global pose estimation algorithms for loop closing. This system works in either monocular or stereo mode, with cameras mounted on a robot moving in 2-D; sensor trajectories with 6 DOF will require large amounts of particles for their representation. In [33], the authors also compare the advantages of separate monocular and stereo approaches in traditional SLAM frameworks.

In this paper, we show the advantages of being able to accommodate both monocular and stereo information in carrying out a 6-DOF SLAM with a handheld camera. In the works of Sola *et al.* [34] and Lemaire *et al.* [20], it is also pointed out that combining visual information at close range as well as at infinity should improve the performance of visual SLAM.

Since the initial results of [35], great progress has been made in the related problem of visual odometry [36]–[39]. Visual odometry systems have the important advantage of constant time execution. Furthermore, during exploratory trajectories, in which an environment feature is seen for a certain window of time and never more, visual odometry can obtain the same precision in the estimation of the sensor location as a SLAM system, with a great reduction in cost. Unfortunately, visual odometry does not cope with loop closings, and thus, eventual drift in these cases is inevitable. Stereo visual odometry combined with GPS can result in a mapping system that avoids long-term drift [40], [41], but unfortunately GPS is not always available. Improving the precision in sensor location through loop closing is one of the main advantages of SLAM.

An important limitation of current SLAM systems that use the standard EKF algorithm is that when mapping large environments very soon, they face computational as well as consistency problems [42]. Many efforts have been invested in reducing the $O(n^2)$ cost of the EKF updates. In [43], an information filter, the dual of the Kalman filter, was used, allowing constant time updates irrespective of the size of the map. An approximation is carried out to sparsify the information matrix, which may lead to map divergency [44]. The treemap algorithm [45] performs updates in $O(\log n)$ also by forcing information matrix sparseness by weak link breakage. In more complicated trajectories, such as lawn mowing, the cost can be more than log linear [46]. In the smoothing and mapping method [47], the authors observed that the information matrix is exactly sparse when all vehicle locations are considered in the stochastic map, and thus, very efficient techniques can be used to compute the batch solution (a recent incremental version is described in [48]).

All of these algorithms use the information form, and thus, the state and covariance are not readily available. There are alternatives that work on the covariance form, such as the map joining algorithm [49]. It works on a sequence of local maps of limited size, and thus, it can cut down the cost of EKF SLAM considerably, although remaining $O(n^2)$. It has the additional advantage of improving the consistency of the resulting estimation [42]. The divide and conquer algorithm [50] is able to compute the covariance form of the stochastic map in an amortized time linear with the size of the map, improving further the consistency of the solution. However, in these systems, local maps are required to be statistically independent. This requires creating a new local map from scratch every time the current local map size limit has been reached. Consequently, no sharing of valuable information is possible in a 6-DOF visual SLAM, such as the camera velocity, or information about features currently being tracked. This issue has been tackled in a recent work [51] by using the *conditional independence* property.

In this paper, we describe a robust and scalable 6-DOF visual SLAM system that can be carried in hand at normal walking speeds of 4–5 km/h, and used to map large indoor and outdoor environments. In Section II, we summarize the main characteristics of our system. In Section III, we describe the details of the visual SLAM system that provides the sequence of conditionally independent (CI) local maps, the basic building blocks of our mapping algorithm. This algorithm, CI divide and conquer (D&C) SLAM, is explained in Section IV. In Section V, we describe the two experiments carried out to test the system, an indoor 200 m loop and an outdoor 140 m loop. In Section VI, we discuss the results obtained, and finally, in Section VII, we draw the main conclusions of our work.

II. Our Proposal

The fundamental characteristics of the system that we describe in this paper are as follows.

1) Unlike any other visual SLAM system, we consider information from features, both close and far from the cameras. A stereo provides 3-D information from nearby scene points, and each camera can also provide bearing only information from distant scene points. Both types of information are incorporated into the map and used to improve the estimation of both the camera pose and velocity, as well as the map.

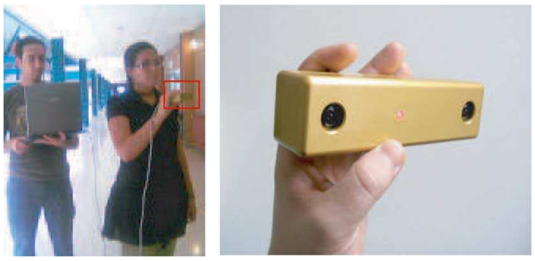


Fig. 1. Stereo vision system used to acquire the image sequences. (Left) Experimental setup during the data acquisition for the indoor experiment.

2) Nearby scene points provide scale information through the stereo baseline, eliminating the intrinsic scale unobservability problem of monocular systems.
3) We use Conditionally Independent Divide and Conquer SLAM algorithm that allows the system to maintain 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. Being a D&C algorithm, it also allows linear time execution, enabling the system to be used for large-scale indoor/outdoor SLAM.

Our 6-DOF hardware system consists of a stereo camera carried in hand and a laptop to record and process a sequence of images (see Fig. 1). Since the camera moves in 6 DOF, we define the camera state using 12 variables: camera position in 3-D Cartesian coordinates, camera orientation in Euler angles, and linear and angular velocities. It is known that a stereo camera can provide depth estimation of points up to a certain distance determined by the baseline between left and right cameras. Therefore, two regions can be differentiated: a region close to the cameras and visible by both, in which the stereo behaves as a range and bearing sensor. The second is the region of features far from the cameras or seen by only one, in which the stereo becomes a monocular camera, providing bearing only measurements of such points. To take advantage of both types of information, we combine 3-D points and inverse depth (ID) points (introduced in [52]) in the state vector in order to build a map and estimate the camera trajectory. The system produces sequences of local maps of limited size containing both types of features using an EKF SLAM algorithm. As we detail in Section IV, these local maps are joined into a full map using the CI D&C SLAM algorithm, obtaining as final result a full stochastic map containing all tracked features, and the final and intermediate camera states from each local map. This system is highly scalable: local maps are built in constant time, regardless of the size of the environment, and the CI D&C algorithm requires amortized linear time.

During the feature tracking process, the right image is chosen as reference to initialize new features. Interest points are detected and classified according to their disparity with the left image. Those points whose disparity reveals a close distance are initialized as 3-D features, otherwise they are modeled as ID points and initialized using the bearing information obtained from the right image. When the camera moves, these features are tracked in order to update the filter and produce the corresponding corrections. To track a feature, its position is predicted in both images inside a bounded region given by the uncertainty in the camera motion and the corresponding uncertainty of the feature.

The process to select, initialize, and manage these features is detailed in the next section.

III. Visual SLAM System

A. State Representation

The state vector that represents a local submap $\mathbf{x}_B$ contains the final camera location $\mathbf{x}_c$ and the location of all features $\mathbf{x}_{f_{1:n}}$ with respect to the map base reference B, the initial camera location. Some features are codified using the *ID parametrization* that model points that are at infinity in $\mathbf{x}_{\mathrm{ID}}$. Additionally, Cartesian *3-D parametrization* is used to represent depth points in $\mathbf{x}_{3\mathrm{D}}$:

$$\mathbf{x}_B = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{x}_{f_{1:n}} \end{bmatrix} = \begin{bmatrix} \mathbf{x}_c \\ \mathbf{x}_{ID} \\ \mathbf{x}_{3D} \end{bmatrix}. \tag{1}$$

The camera is described by the position of its optical center in Cartesian coordinates $\mathbf{r}$, its orientation in Euler angles Ψ, its linear velocity $\mathbf{v}$, and its angular velocity $\mathbf{w}$. In order to carry out the prediction process, the camera motion follows a constant velocity model with zero mean Gaussian noise in the linear and angular accelerations

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{r} \\ \Psi \\ \mathbf{v} \\ \mathbf{w} \end{bmatrix}. \tag{2}$$

Image corners classified as depth points are transformed to 3-D points, given the disparity information provided by the stereo pair. Section III-D describes the criterion adopted to select points as depth points. Since the stereo camera provides rectified images, the backprojection equations to obtain a 3-D point are based on a pinhole camera model that relates image points and 3-D points using the following transformation function:

$$\begin{aligned} \mathbf{x}_{3\mathrm{D}} &= f(u_r, v_r, u_l, v_l) \\ &= [x, y, z]^T \\ &= \left[\frac{b(u_r - u_0)}{d}, \frac{b(v_r - v_0)}{d}, \frac{fb}{d}\right]^T \end{aligned} \tag{3}$$

where (u_r, v_r) and (u_l, v_l) are the pixels on the right and left images, and $d = (u_l - u_r)$ is the horizontal disparity. The remainder terms in the equations are the calibrated parameters of the camera, i.e., the central pixel of the image (u_0, v_0), the baseline b, and the focal length f.

Given the camera location $\mathbf{x}_{c_i}$, an ID point is defined in [52] as

$$\mathbf{x}_{\mathrm{ID}} = \begin{bmatrix} \mathbf{r}_i \\ \theta_i \\ \phi_i \\ \rho_i \end{bmatrix}. \tag{4}$$

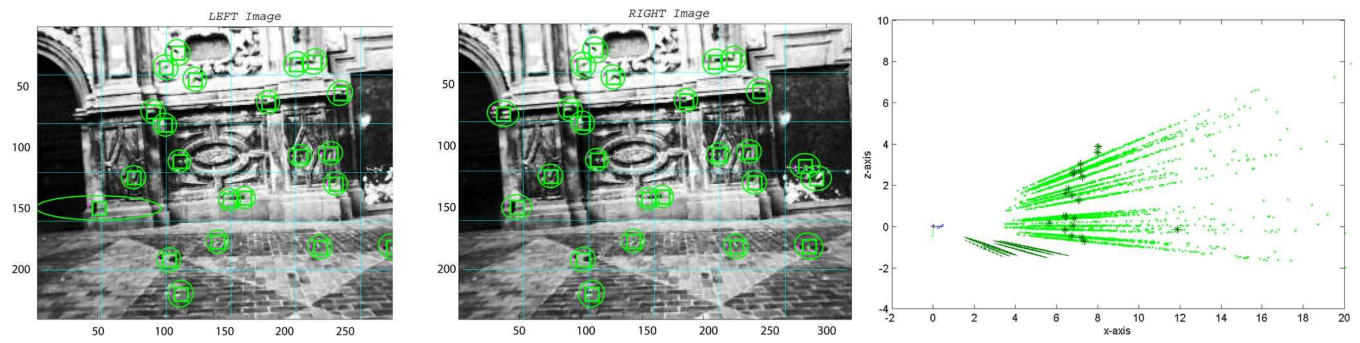


Fig. 2. Points detected using a stereo camera. Projection of map features on both left and middle images. (Right) We show feature uncertainties from a lateral perspective. 3-D feature uncertainties are drawn using darker ellipses whereas we use samples to show the ID feature uncertainties. The accompanying video `VSLAM_local_map.avi` illustrates the process of building a single local submap.

This vector depends on the optical center $\mathbf{r}_i$ of the camera from which the feature was first observed, the direction of the ray passing through the image point (i.e., azimuth θ_i, elevation ϕ_i), and the inverse of its depth, $\rho_i = 1/d_i$.

B. Selection and Management of Trackable Points

To ensure tracking stability of map features, distinctive points have to be selected. Following a similar idea as the one presented in [53], we use the Shi–Tomasi variation of the Harris corner detector to select good trackable image points and their corresponding 11×11 surrounding patch.

From the first step, the right image is split using a regular grid; the point with the best detector response per grid cell is selected (see Fig. 2). At each step, we use only those features that fall in the field of view (FOV) of the camera when they are projected along with their uncertainties on right and left images. Using the patch associated with each feature, a matching search based on normalized cross-correlation is performed inside the projected uncertainty region, as introduced in [24]. During the following steps, those cells that become and remain empty for a given time are monitorized to initialize a new feature when a good point is detected. In this way, features can be uniformly distributed in the image, improving the amount of information gathered from the scene, and therefore the map estimate. The approach is accompanied by a feature management strategy so that nonpersistent features are deleted from the state vector to avoid an unnecessary growth in population.

C. Measurement Equation

At each step, we apply the active search process described before such that, for each projected feature in the stereo image, a match is found after performing normalized cross-correlation. Thus, a new observation $\mathbf{z}$ given by the matched pixel is used to update the state of the camera and the map.

In the right camera, the equation that defines the relation between the ith ID feature $\mathbf{x}_{\mathrm{ID}}^i$ and its observation $\mathbf{z}_{\mathrm{ID}}^{r_i}$ is given by the following measurement equation:

$$\begin{aligned}\mathbf{z}_{\mathrm{ID}}^{r_i} &= h_{\mathrm{ID}}^r(\mathbf{x}_c, \mathbf{x}_{\mathrm{ID}}^i) + \upsilon \\ &= \mathrm{projection}(\ominus \mathbf{x}_c \oplus \mathbf{x}_{\mathrm{ID}}^i) + \upsilon \end{aligned} \tag{5}$$

where h_{ID}^r is the function that projects the ID feature to the right camera and υ is a zero mean Gaussian noise with σ_p standard deviation that represents the projection error in pixels. Alternatively, we can define the measurement equation that relates the inverse point observation on the left image by

$$\begin{aligned}\mathbf{z}_{\mathrm{ID}}^{l_i} &= h_{\mathrm{ID}}^l(\mathbf{x}_c, \mathbf{x}_{\mathrm{ID}}^i) + \upsilon \\ &= \mathrm{projection}(\ominus \mathbf{x}_c \oplus \mathbf{x}_{c_r c_l} \oplus \mathbf{x}_{\mathrm{ID}}^i) + \upsilon \end{aligned} \tag{6}$$

where the displacement of the left camera optical center with respect to the right camera is given by the rigid transformation $\mathbf{x}_{c_r c_l} = [0\; b\; 0]^T$.

In a similar way, we describe observations corresponding to 3-D map features in the right and left cameras as

$$\begin{aligned}\mathbf{z}_{3\mathrm{D}}^{r_i} &= h_{3\mathrm{D}}^r(\mathbf{x}_c, \mathbf{x}_{3\mathrm{D}}^i) + \upsilon \\ &= \mathrm{projection}(\ominus \mathbf{x}_c \oplus \mathbf{x}_{3\mathrm{D}}^i) + \upsilon \\ \mathbf{z}_{3\mathrm{D}}^{l_i} &= h_{3\mathrm{D}}^l(\mathbf{x}_c, \mathbf{x}_{3\mathrm{D}}^i) \\ &= \mathrm{projection}(\ominus \mathbf{x}_c \oplus \mathbf{x}_{c_r c_l} \oplus \mathbf{x}_{\mathrm{ID}}^i) + \upsilon. \end{aligned}$$

Note that we use $\oplus$ and $\ominus$ operators in order to denote the corresponding compositions and inversions of transformations. They represent different transformations depending on the kind of parametrization used to express a feature. In [49], the definitions for 2-D transformations were introduced, dealing mainly with point features and line features. In [54], the operations have been extended for 3-D ID and depth points. Details of the calculation of the corresponding Jacobians to propagate the uncertainties correctly can also be found in [54].

Fig. 2 shows the prediction of these 3-D ID features that fall inside the FOV of each of the cameras. A good advantage of using a stereo camera is that although a feature can disappear from the FOV of one camera, information to update the state is available if the feature can be still found in the other. As it will be shown in the experiments, this fact is of extreme importance when the camera rotates or turns around a corner, since features escape very fast from the FOV of a single camera, making the estimation of the camera location in these moments very weak.

D. Depth Points Versus ID Points

Current research on monocular SLAM has shown that the ID parametrization is suitable to represent the distribution of

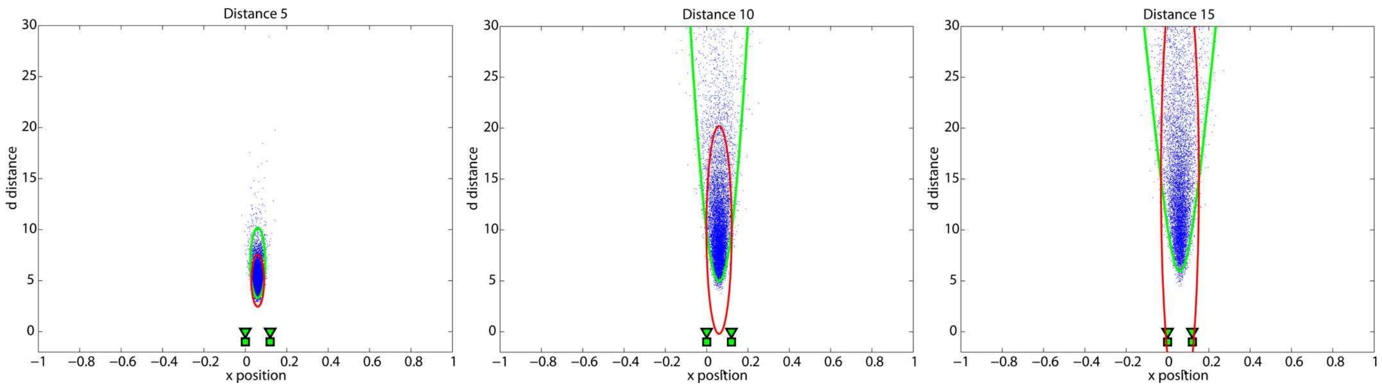


Fig. 3. Simulated experiment of a point reconstruction from a stereo pair observation for a point at (left) 5-, (middle) 10-, and (right) 15-m distance. The point clouds are samples from the real distribution of the point location, given that the pixel noise in the images is Gaussian. Dark red ellipses represent the uncertainty region for the point location when the back projection equations of a depth point are linearized. Light green regions represent the uncertainty in the point using the ID parametrization. The accompanying video `VSLAM_stereo_distribution.avi` shows the real and approximate uncertainties.

features at infinity as well as close points, allowing to perform an undelayed initialization of features. Despite its properties, each ID point needs an overparametrization of six values instead of a simpler three coordinates spatial representation [55]. This produces a computational overhead in the EKF. Working with a stereo camera, which can estimate the depth of points close to the camera, raises the subtle question of when a feature should be initialized using a 3-D or an ID representation.

In order to clarify this issue, we have designed a simulated experiment to study the effect of the linearization in both representations when a point is initialized using the stereo information. In this simulated experiment, the variance of the pixel noise ($\sigma_p = 1$ pixel) and the actual intrinsic parameters of the stereo camera used, such as the baseline, are taken into account to implement the simulation. The experimental setup consists of a stereo pair where the left camera is located at the origin of the reference frame, with its principal axis pointing along Z and the X axes pointing to the right. The right camera is at $b = 12$ cm in X. We consider a point that is in the middle between both cameras at different distances in Z. Given a noisy pixel observation, the uncertainty region of a reconstructed point is sampled and plotted in Fig. 3 for three different point distances: 5,10, and 15 m. The uncertainty region of the 3-D representation, which is calculated using a linearization of (3) and evaluated in the ground truth, is represented by the dark red ellipse. The corresponding uncertainty region of the linearized ID representation is bounded by the light gray lines in the plot. Notice that the ID parametrization models very accurately the real uncertainty for the studied distances. However, although the dark ellipse covers the real distribution at 5 m quite accurately, for longer distances, the ellipse overestimates the uncertainty in the region close to the cameras and is overconfident for far distances.

This empirical analysis suggests choosing a threshold of 5 m. A point closer than 5 m is initialized using a 3-D representation, a more distant point is parameterized as an ID point.

ID features can be transitioned to 3-D points, reducing significantly the number of DOF. Conversion requires an analysis of the linearity of the functions that model both depth point and ID point distributions. In [55], this issue is considered by using a linearity index. Such analysis makes it possible to decide when an inverse point distribution is well approximated with the overparameterized coding. Switching from ID to depth depends on a linearity threshold derived from the analysis.

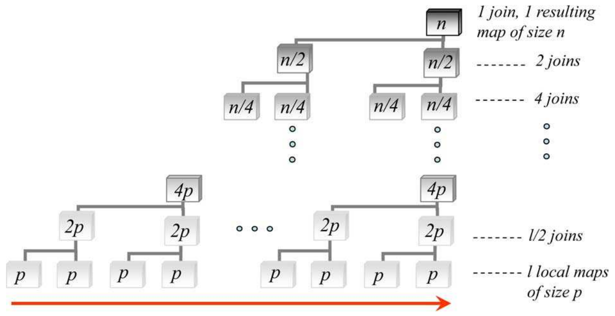


Fig. 4. Binary tree representing the hierarchy of maps that are created and joined in D&C SLAM. The red line shows the sequence in which maps are created and joined.

IV. CI D&C SLAM

D&C SLAM has proved to be a good algorithm in minimizing the computational complexity of EKF-based SLAM and improving consistency of the resulting estimate [50]. The algorithm allows us to efficiently join several local maps into a single state vector using map joining in a hierarchical tree structure (see Fig. 4). Local maps can be obtained in constant time, regardless of the size of the environment, and the map joining operations can be performed in an amortized linear time. The D&C SLAM algorithm was, however, conceived for statistically independent sequences of local maps. This requires creating a new local map from scratch every time the current local map size limit has been reached. Consequently, it is not possible to share valuable information in a 6-DOF visual SLAM, such as the camera velocity, or information about features currently being tracked.

In this section, we describe the *CI* D&C SLAM algorithm, which is able to work with maps that are not statistically independent, but rather *conditionally independent*, and thus, allow sharing of the valuable information with no increment in computational cost or loss of precision whatsoever.

A. CI Local Maps

In visual SLAM, it can be very useful to share some state vector components between consecutive submaps: some camera states, such as linear and angular velocities, as well as features that are in the transition region between adjacent submaps and are currently being tracked. This allows us to improve the estimate of relative location between the submaps and continue tracking the observed features with no interruptions. Nevertheless, special care is needed to join the submaps in a single map since their estimates are not independent anymore.

The novel technique to achieve these requirements is based on the concept of CI local maps presented in [51]. Here, we present a brief summary of the technique.

Suppose that a local map 1 has been built and we want to start a new submap 2 not from scratch, but sharing some elements in common with 1. Submap 1 is described by the following probability density function:

$$p(\mathbf{x}_A, \mathbf{x}_C | \mathbf{z}_a) = \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{A_a} \\ \hat{\mathbf{x}}_{C_a} \end{bmatrix}, \begin{bmatrix} P_{A_a} & P_{AC_a} \\ P_{CA_a} & P_{C_a} \end{bmatrix} \right) \tag{7}$$

where $\mathbf{x}_A$ are the components of the current submap that only belong to map 1, $\mathbf{x}_C$ are the elements that will be shared with map 2, and $\mathbf{z}_a$ the observations gathered during the map construction. Notice that upper case subindices are for state vector components whereas lower case subindices describe which observations $\mathbf{z}$ have been used to obtain the estimate.

Submap 2 is then initialized with the result of marginalizing out the noncommon elements from submap 1:

$$p(\mathbf{x}_C | \mathbf{z}_a) = \int p(\mathbf{x}_A, \mathbf{x}_C | \mathbf{z}_a)\, d\mathbf{x}_A = \mathcal{N}(\hat{\mathbf{x}}_{C_a}, P_{C_a}). \tag{8}$$

During the trajectory along map 2, new observations $\mathbf{z}_b$ are gathered about the common components $\mathbf{x}_C$ as well as observations of new elements $\mathbf{x}_B$ that are incorporated into the map. When map 2 is finished, its estimate is finally described by

$$p(\mathbf{x}_C, \mathbf{x}_B | \mathbf{z}_a, \mathbf{z}_b) = \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{C_{ab}} \\ \hat{\mathbf{x}}_{B_{ab}} \end{bmatrix}, \begin{bmatrix} P_{C_{ab}} & P_{CB_{ab}} \\ P_{BC_{ab}} & P_{B_{ab}} \end{bmatrix} \right) \tag{9}$$

where the subindices in the estimates $\hat{\mathbf{x}}_{C_{ab}}$ and $\hat{\mathbf{x}}_{B_{ab}}$ reveal that both sets of observations $\mathbf{z}_a$ and $\mathbf{z}_b$ have been used in the estimation process. This means that submap 2 is updated with all the information gathered by the sensor. But observe that map 1 in (7) has been updated with the observation $\mathbf{z}_a$ but not with the more recent observations $\mathbf{z}_b$.

Fig. 5 shows a Bayesian network that describes the probabilistic dependencies between elements of submaps 1 and 2. As it can be seen, the only connection between the set of nodes $(\mathbf{x}_A, \mathbf{z}_a)$ and $(\mathbf{x}_B, \mathbf{z}_b)$ is through node $\mathbf{x}_C$, i.e., both subgraphs are *d-separated* given $\mathbf{x}_C$ [56]. This implies that nodes $\mathbf{x}_A$ and $\mathbf{z}_a$ are *CI* of nodes $\mathbf{x}_B$ and $\mathbf{z}_b$ given node $\mathbf{x}_C$. Intuitively, this means that if $\mathbf{x}_C$ is known, submaps 1 and 2 do not carry any additional information about each other.

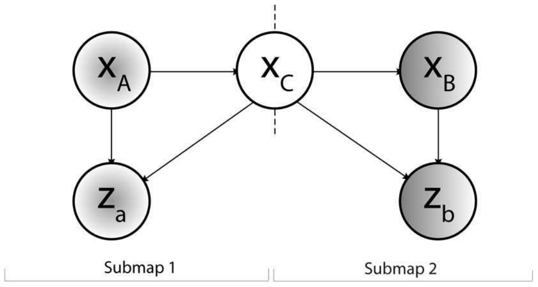


Fig. 5. Bayesian network that describes the relations between two consecutive submaps.

B. CI Map Joining

Consider two consecutive CI local maps. We are interested in joining the maps into a single stochastic map described by

$$p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C | \mathbf{z}_a, \mathbf{z}_b) = \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{A_{ab}} \\ \hat{\mathbf{x}}_{C_{ab}} \\ \hat{\mathbf{x}}_{B_{ab}} \end{bmatrix}, \begin{bmatrix} P_{A_{ab}} & P_{AC_{ab}} & P_{AB_{ab}} \\ P_{CA_{ab}} & P_{C_{ab}} & P_{CB_{ab}} \\ P_{BA_{ab}} & P_{BC_{ab}} & P_{B_{ab}} \end{bmatrix} \right). \tag{10}$$

Taking into account the submap conditional independence property, it can be demonstrated [51] that the optimal map result of the joining can be computed using

$$\begin{aligned} K &= P_{AC_a} P_{C_a}^{-1} \\ &= P_{AC_{ab}} P_{C_{ab}}^{-1} && (11) \\ \hat{\mathbf{x}}_{A_{ab}} &= \hat{\mathbf{x}}_{A_a} + K(\hat{\mathbf{x}}_{C_{ab}} - \hat{\mathbf{x}}_{C_a}) && (12) \\ P_{A_{ab}} &= P_{A_a} + K(P_{CA_{ab}} - P_{CA_a}) && (13) \\ P_{AC_{ab}} &= K P_{C_{ab}} && (14) \\ P_{AB_{ab}} &= K P_{CB_{ab}}. && (15) \end{aligned}$$

Using this technique, we can build local maps that have elements in common, and then retrieve the global information in a consistent manner. After the joining, the elements belonging to the second map are transformed to the base reference of the first map.

C. Actual Implementation for Stereo

The D&C SLAM algorithm of [50] can be adapted to work with conditional independent local maps simply by using the CI map joining operation described before. As we mentioned before, since the camera moves in 6 DOF, the camera state is composed of its position using 3-D Cartesian coordinates, the orientation in Euler angles, and its linear and angular velocities. 3-D points and ID points are included as features in the state vector. When a local map $\mathbf{m}_i$ is finished, the final map estimate is given by

$$\mathbf{m}_i.\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_{R_i R_j} \\ \hat{\mathbf{v}}_{R_i R_j} \\ \hat{\mathbf{x}}_{R_i F_{1:m}} \\ \hat{\mathbf{x}}_{R_i F_{m+1:n}} \end{bmatrix} \tag{16}$$

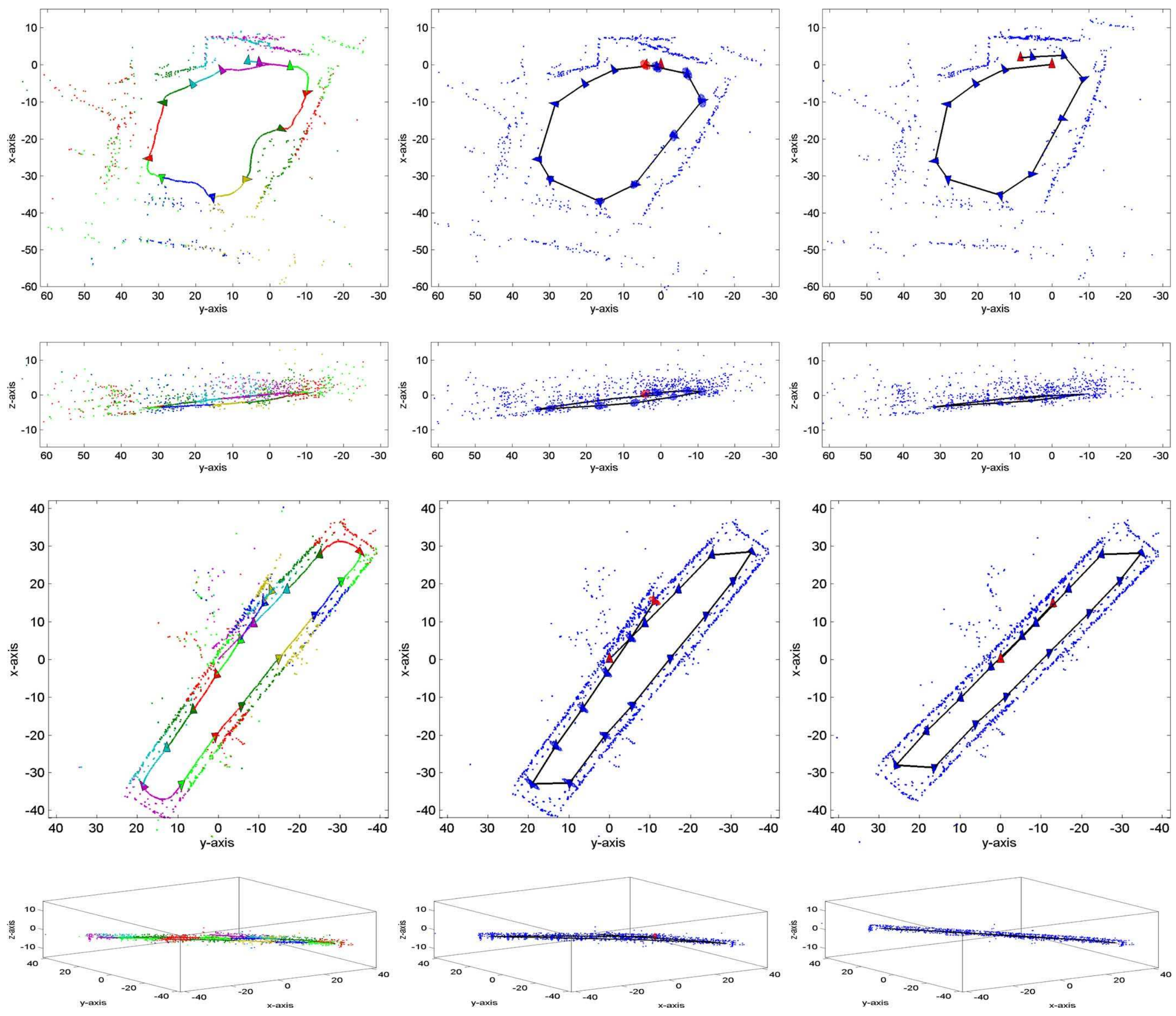


Fig. 6. (a) Outdoors experiment: 6-DOF stereo SLAM on a public square. (Top row) Both XY projection and (top-middle row) YZ projection are shown in order to illustrate the precision obtained. (b) Indoor experiment along a building environment. (Bottom-middle row) XY projection and (bottom row) YZ projection. (Left column) The sequence of CI local maps is represented with respect to the initial reference; (middle column) results obtained after running the D&C algorithm that joins and corrects the estimates; (right column) final map obtained when the loop closing constraint is imposed. The scale factor and camera positions are well recovered due to the combined observations of 3-D points and ID points. The accompanying videos `VSLAM_video_outdoor.avi` and `VSLAM_video_indoor.avi` show the full execution of the outdoor and indoor experiments.

where $\hat{\mathbf{x}}_{R_i R_j}$ is the final camera location R_j with respect to the initial one, R_i and $\hat{\mathbf{v}}_{R_i R_j}$ are the linear and angular velocities, $\hat{\mathbf{x}}_{R_i F_{1:m}}$ are 3-D and ID features that will only remain in the current map, and $\hat{\mathbf{x}}_{R_i F_{m+1:n}}$ are 3-D and ID features that will be shared with the next submap $\mathbf{m}_j$.

Since the current camera velocity $\hat{\mathbf{v}}_{R_i R_j}$ and some features $\hat{\mathbf{x}}_{R_i F_{m+1:n}}$ are used to initialize the next local map, these elements have to be computed with respect to the base reference of the second map R_j:

$$\mathbf{m}_i.\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_{R_i R_j} \\ \hat{\mathbf{v}}_{R_i R_j} \\ \hat{\mathbf{x}}_{R_i F_{1:m}} \\ \hat{\mathbf{x}}_{R_i F_{m+1:n}} \\ \cdots \\ \ominus\hat{\mathbf{x}}_{R_i R_j} \oplus \hat{\mathbf{v}}_{R_i R_j} \\ \ominus\hat{\mathbf{x}}_{R_i R_j} \oplus \hat{\mathbf{x}}_{R_i F_{m+1:n}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{A_a} \\ \cdots \\ \hat{\mathbf{x}}_{C_a} \end{bmatrix} \tag{17}$$

where the new elements define the common part $\hat{\mathbf{x}}_{C_a}$ and the original map defines $\hat{\mathbf{x}}_{A_a}$. Notice that the appropriate composition operation has to be applied for each transformed component and that the corresponding covariance elements have to be added to the map.

In local mapping, a base reference has to be identified to start a new map. This common reference is represented by the final vehicle position, which is the case of R_j between $\mathbf{m}_i$ and $\mathbf{m}_j$.

The initial state vector of the next submap is then given by

$$\mathbf{m}_j.\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_{R_j R_j} \\ \ominus\hat{\mathbf{x}}_{R_i R_j} \oplus \hat{\mathbf{v}}_{R_i R_j} \\ \ominus\hat{\mathbf{x}}_{R_i R_j} \oplus \hat{\mathbf{v}}_{R_i R_j} \\ \ominus\hat{\mathbf{x}}_{R_i R_j} \oplus \hat{\mathbf{x}}_{R_i F_{m+1:n}} \end{bmatrix} \tag{18}$$

where $\hat{\mathbf{x}}_{R_j R_j}$ represents the location of the camera in the new reference frame with initial zero uncertainty and zero correlation with the rest of the elements of the initial map. Notice that the initial velocity brought from the previous map has been replicated twice. One of the copies will change as the camera moves through the new map carrying the current camera velocity. The other copy will remain fixed and, together with the transformed features, will be the common elements with the previous map. The same process is successively repeated with all local maps.

D. Continuous Data Association in Each Local Map

Recent work on large environments [19] has shown that the joint compatibility test [57] helps avoiding map corruption in the visual SLAM by rejecting measurements that come from moving objects. This framework is suitable in environments with a limited number of observations. However, a branch and bound algorithm implementation of (*JCBB*) has limited use when the number of observations per step is large. In this paper, we have obtained more efficient results using the *randomized joint compatibility* version *RJC* proposed in [50], in which, in the spirit of RANSAC, a *joint compatibility (JC)* test is run with a fixed set of p randomly selected measurements. In this case, correlation between patches and individual χ^2 tests is used to obtain candidate matches. If all p measurements and their matches are jointly compatible, we apply the nearest neighbor rule to match the remaining measurements. Once a full hypothesis H is obtained, we check *JC* to avoid false positives. The process is repeated t times with adaptive RANSAC, limiting the probability of missing a correct association.

E. Map Matching

The property of sharing common elements solves the data association problem between consecutive local maps [50]. This requires us to solve data association only in loop closing situations. We use the map matching algorithm of [19] in order to detect a previously visited area. The algorithm finds correspondences between features in different local maps, taking into account the texture and the relative geometry between the features. If sufficient corresponding features are found, an ideal measurement equation that imposes the loop closing constraint is applied in the final map.

V. Experiments in Urban Outdoor and Indoor Environments

In order to demonstrate the robustness and scalability of the visual SLAM system that we propose, we have gathered two 320×240 image sequences with a stereo system (see Fig. 1). The system provides a 65×50 degree FOV per camera, and has a baseline of 12 cm, limiting the 3-D point features initialization up to a distance close to 5 m.

An indoor loop (at 48 fps) and an urban outdoor (at 25 fps) loop sequences were captured carrying the camera in hand, at normal walking speeds of 4–5 km/h. Both sequences were processed in MATLAB with the proposed algorithms on a desktop computer with an Intel 4 processor at 2.4 GHz. The higher frame rate for the indoor experiment helps in reducing the probability

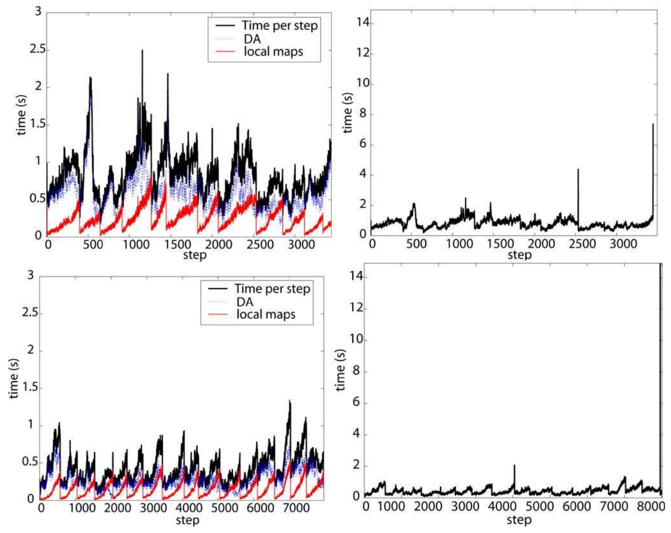


Fig. 7. Running time per step of all associated processes; a detailed analysis of (left) the features extraction, local mapping (labeled as local maps), and data association (DA) times; (right) total time per step where the peaks represent the joins performed by the CI D&C algorithm. (Top) Outdoor environment: the public square. (Bottom) Indoor environment.

of mismatches given that the environment includes brick walls providing ambiguous texture information.

The outdoor sequence is composed of 3441 stereo pairs gathered in a public square of our home town (see Fig. 6 top row). The full trajectory is approximately 140 m long from the initial camera position. Fig. 6, left column, shows the sequence of conditional independent local maps obtained with the technique described in Section IV-A. Each map contains 100 features combining ID and 3-D points. The total number of maps built during the stereo sequence is 11. The result of D&C without applying the loop closing constraint is shown in Fig. 6, middle column. As it can be observed, the precision of the map obtained is good enough to almost align the first and last submaps after all the trajectory has been traversed, even without applying loop closing constraints. Fig. 6, right column, presents the final result after closing the loop.

The second experiment was carried out inside one of our campus buildings in a walk of approximately 210 m (see Fig. 6, bottom row). The same process was run in order to obtain a full map from 8135 stereo pairs. This environment has a particular degree of difficulty due to ambiguous texture and the presence of extensive zones of glass windows such as offices, corridors, and cafeterias. This can be noticed in the long distance points estimated in some of the maps, which are actually inside offices and the cafeteria (see Fig. 6, left column). The result of CI D&C is shown in Fig. 6, middle column, and the final result after loop closing is shown in Fig. 6, right column.

Our 6-DOF SLAM system, even implemented in MATLAB, does not exceed 2 s per step, which is the worst case when building CI local maps. Fig. 7 shows how the system running time remains constant in most of the steps. Moreover, time peaks that appear when CI D&C takes place are below 8 s for the square experiment and 14 s for the indoor experiment, which are the maximum times required in the last step.

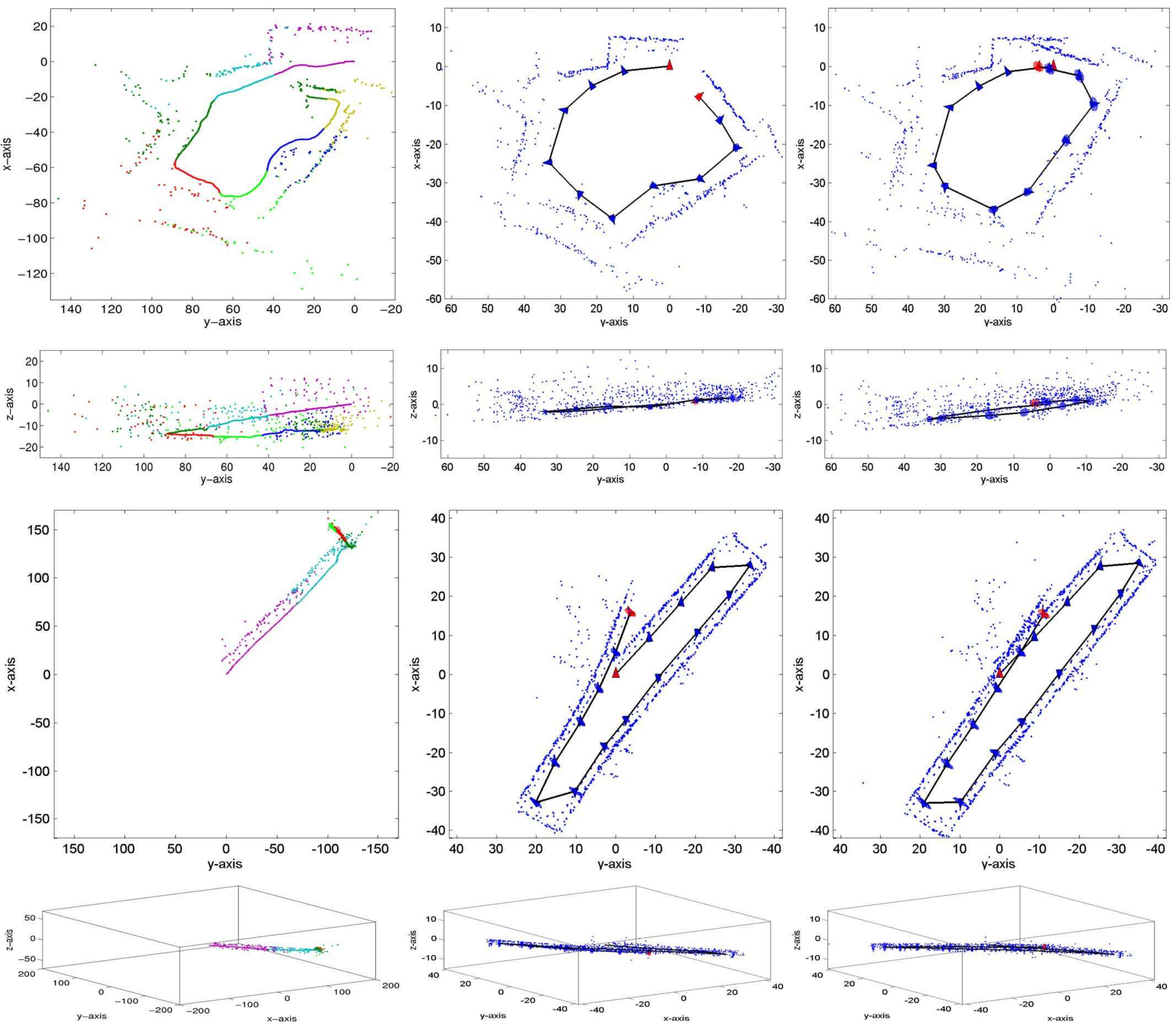


Fig. 8. Comparison of the outdoor and indoor maps obtained before the loop closure using three different techniques. (Left) Monocular SLAM with ID points, (middle) stereo SLAM with 3-D points, and (right) the proposed stereo SLAM with 3-D points and ID points.

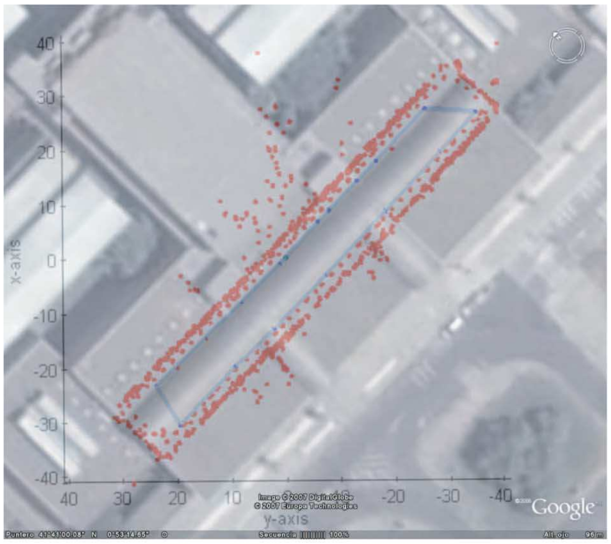
Using the Google Earth tool, we can see that the map scale obtained and the trajectory followed by the camera is very close to the real scale. Fig. 9 illustrates comparative results. We loaded the MATLAB figure in Google Earth and set the scale parameter to the real scale. Given that we had neither GPS nor compass measurements for the initial locations of the camera that are the base reference of each map, the position and orientation of the figure over the map were adjusted by hand. It can be noticed that angles between the square sides and the shape of the walls of the surrounding environment have been captured with precision.

VI. Discussion

As presented in Section I, several works have demonstrated successful visual SLAM systems in small environments using monocular or stereo cameras. There are several important factors that limit the extension of these results to large-scale environments.

First, the computational complexity and consistency of the underlying SLAM technique. In this paper, we have presented a novel algorithm that builds CI local maps in constant time and combines them in an optimal way in amortized linear time. Although the experiments presented here were processed in MATLAB, we expect that the extension to stereo of our current real-time implementation [19] will be able to build local maps up to 100 features in real time, with updates at 25 Hz. The D&C map joining, loop detection, and loop closing can be implemented on a separate thread, taking advantage of current multiple core processors.

In the case of monocular SLAM, another important limiting factor is the intrinsic unobservability of the scale. This problem can be addressed using additional sensors such as the vehicle



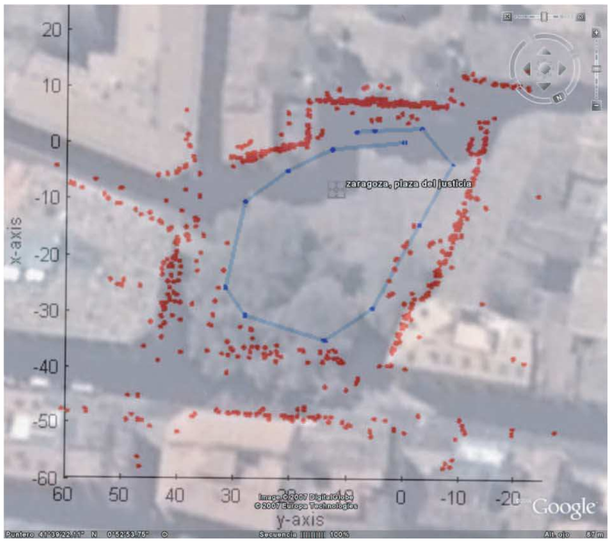


Fig. 9. Stereo visual SLAM recovers the true scale. (Top) Building environment and (bottom) the public square overlapping Google Earth.

odometry, GPS, or inertial units. When they are not available, the scale can be initialized using some *a priori* knowledge about the environment such as the size of a known object visible at the start [16] or the initial speed of the camera. However, in large environments, unless scale information is injected on the system periodically, the scale of the map can slowly drift (see, for example, the experiments in [19]). Another critical issue appears when the scene is mostly planar and perpendicular to the optical axis. In this situation, with a monocular camera, it is very difficult to distinguish between camera translation and rotation, unless a wide FOV is used.

To illustrate these difficulties, we have processed our indoor and outdoor experiments using only the information from the right camera. As we are now using a bearing only system, all the features are initialized using the ID representation. To bootstrap the system, we have introduced a initial estimated speed for the camera of 1 m/s. Apart from that, our visual SLAM algorithm remains unchanged. The resulting maps are represented in the left column of Fig. 8. As it can be seen, the scale obtained by the system drifts (compare the beginning of the loop with the end). Also, in the outdoor experiment, at a certain point, the system misinterprets the camera translation as a rotation, and the map gets corrupted. Here, we are using a camera with FOV of 65°. The results obtained in the same environment with an FOV of 90° are significantly more robust [51]. In the indoor experiment with a monocular camera, as the objects are much closer to the camera, most of the features disappear fast from the FOV when the camera turns, leading to a bad estimation of its position and consequently divergence in the map estimate.

We have also processed the sequences with our SLAM algorithm using conventional stereo, i.e., changed to initialize all the features whose disparity is larger than one pixel as 3-D points. Features without disparity are discarded because its depth cannot be computed by stereo. The immediate benefit is that the true environment scale is observable and the map corruption disappears (Fig. 8, middle column). However, for points that are more than 10 m away from the camera, a Gaussian in *xyz* is a bad approximation for its true uncertainty. This is the reason for the map deformation that is clearly visible in the lower part of the outdoor experiment, where many features are at about 20 m from the camera.

The proposed system (Fig. 8, right column) combines the advantages of stereo and bearing only vision. On the one hand, the true scale is precisely obtained due to the 3-D information obtained by the stereo camera from close point features. On the other hand, the region with useful point features extends up to infinity due to the ID representation developed for bearing-only SLAM. The depth of the features that are far from the camera can be precisely recovered by the system if they are seen from viewpoints that are separated enough. In that case, they can be upgraded to 3-D points for better efficiency [55]. Otherwise, they remain as ID points and still provide very valuable orientation information that improves map precision and keeps the SLAM system stable when few close features are observed.

VII. Conclusion

In this paper, we have shown that 6-DOF visual mapping of large environments can be efficiently and accurately carried out using a stereo camera as the only sensor. One of the contributions of the paper is that information from features nearby and far from the cameras can be simultaneously incorporated to represent the 3-D structure more precisely. Using close points provides scale information through the stereo baseline avoiding "scale-drift," while ID points are useful to obtain angular information from distant scene points.

Another contribution of the paper is the combination of two recent local mapping techniques to improve consistency and reduce complexity in the SLAM process. Using CI local maps [51], our system is able to properly share information related to the camera motion model and common features between consecutive maps. Smoother transitions from map to map are achieved as well as better relative locations between local maps. By means of the simplicity and efficiency of the CI D&C SLAM algorithm, we can recover the full map very efficiently. The combination of both techniques adds robustness to the process without sacrificing precision.

In [50], we describe the performance of D&C SLAM when the vehicle carries out different types of trajectories. For some trajectories, the cost of map joining can increase at some steps, depending of the size of the overlap between the maps to be joined: doing exploration, the overlap is constant and the cost of map joining is small, when completing a loop traversal for a second time the overlap between the maps is total and the cost of joining will be much higher. Although we are able to close large indoor and outdoor loops, the algorithm used for loop closing strongly depends on detecting sets of features already stored in the map when the same area is revisited. It would be interesting to analyze other types of algorithms for loop closing, for instance, the image to map algorithm proposed in [58].

Moreover, as we assume smooth motions, the relocation algorithm presented in [58] would enable the system to avoid failures in case of jitter.

There is also a restriction of the system to estimate pitch orientation due to the use of Euler angles. A combined solution using quaternions can mitigate the problem. This will be part of our future research.

Apart from upward looking cameras and jitter, there are no limitations to manoeuver the camera freely: it can be used in environments that include stairs and other terrain accidents. This kind of experiment will be part of the evaluation process for future work.

We will also focus on comparing our system with other stereo vision techniques such as visual odometry. We are very interested in studying the fusion of the stereo camera with other sensors like GPS or inertial systems in order to compare the precision obtained. We will consider other types of feature detectors as well, and their effect in the final result.

Acknowledgment

The authors would like to thank J. M. M. Montiel and J. Civera for the fruitful discussions.

References

[1] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira, "6DOF SLAM with stereo camera in hand," presented at the Conf. Vis. SLAM—Emerging Technol., IROS 2008, San Diego, CA, 2008.

[2] J. Folkesson and H. Christensen, "Graphical SLAM for outdoor applications," *J. Field Robot.*, vol. 23, no. 1, pp. 51–70, 2006.

[3] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6-D SLAM-3D mapping outdoor environments: Research Articles," *J. Field Robot.*, vol. 24, no. 8/9, pp. 699–722, 2007.

[4] L. Ellekilde, "Dense 3-D map construction for indoor search and rescue," *J. Field Robot.*, vol. 24, no. 1/2, pp. 71–89, Feb. 2007.

[5] J. Kim and S. Sukkarieh, "Airborne simultaneous localisation and map building," in *Proc. IEEE Int. Conf. Robot. Autom., (ICRA 2003)*, Sep. 14–19, vol. 1, pp. 406–411.

[6] M. Bryson and S. Sukkarieh, "Building a robust implementation of bearing-only inertial SLAM for a UAV," *J. Field Robot.*, vol. 24, no. 1/2, pp. 113–143, 2007.

[7] M. Deans and M. Hebert, "Experimental comparison of techniques for localization and mapping using a bearing-only sensor," in *Proc. Int. Symp. Exp. Robot., (ISER 2000).*, Lecture Notes in Control and Information Science, vol. 271, S. S. D. Rus, Ed. Honolulu, HI: Springer-Verlag, 2000, pp. 395–404.

[8] T. Fitzgibbons and E. Nebot, "Bearing only SLAM using colour-based feature tracking," presented at the 2002 Aust. Conf. Robot. Autom., Auckland, New Zealand, 2002.

[9] T. Bailey, "Constrained initialisation for bearing-only SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA 2003)*, Sep. 14–19, vol. 2, pp. 1966–1971.

[10] N. Kwok and G. Dissanayake, "An efficient multiple hypothesis filter for bearing-only SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., (IROS 2004)*, Sep. 28–Oct. 2, vol. 1, pp. 736–741.

[11] N. Kwok, G. Dissanayake, and Q. Ha, "Bearing-only SLAM using a SPRT based Gaussian sum filter," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA 2005)*, Apr. 12–22, 2005, pp. 1109–1114.

[12] J. Sola, A. Monin, M. Devy, and T. Lemaire, "Undelayed initialization in bearing only SLAM," in *Proc. IEEE/RSJ Int. Con. Intell. Robots Syst. (IROS 2005)*, Aug. 21–26, 2005, pp. 2499–2504.

[13] T. Lemaire, S. Lacroix, and J. Sola, "A practical 3D bearing-only SLAM algorithm," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS 2005)*, Aug. 2–6, 2005, pp. 2449–2454.

[14] P. Jensfelt, D. Kragic, J. Folkesson, and M. Bjorkman, "A framework for vision based bearing only 3-D SLAM," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA 2006)*, May 15–19, 2006, pp. 1944–1950.

[15] A. Gil, O. Reinoso, O. Martínez-Mozos, C. Stachniss, and W. Burgard, "Improving data association in vision-based SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS 2006)*, Beijing, China, Oct. 2006, pp. 2076–2081.

[16] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 6, pp. 1052–1067, Jun. 2007.

[17] J. Folkesson, P. Jensfelt, and H. Christensen, "Vision SLAM in the measurement subspace," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA 2005)*, Apr. 18–22, 2005, pp. 30–35.

[18] P. Smith, I. Reid, and A. Davison, "Real-time monocular SLAM with straight lines," in *Proc. Brit. Mach. Vis. Conf.*, 2006, vol. 1, pp. 17–26.

[19] L. Clemente, A. J. Davison, I. D. Reid, J. Neira, and J. D. Tardós, "Mapping large loops with a single hand-held camera," in *Proc. Robotics: Sci. Syst.*, Atlanta, GA, Jun. 2007.

[20] T. Lemaire and S. Lacroix, "SLAM with panoramic vision," *J. Field Robot.*, vol. 24, no. 1/2, pp. 91–111, 2007.

[21] T. Goedemé, M. Nuttin, T. Tuytelaars, and L. Van Gool, "Omnidirectional vision based topological navigation," *Int. J. Comput. Vis.*, vol. 74, no. 3, pp. 219–236, 2007.

[22] E. Royer, M. Lhuillier, M. Dhome, and J. Lavest, "Monocular vision for mobile robot localization and autonomous navigation," *Int. J. Comput. Vis.*, vol. 74, no. 3, pp. 237–260, 2007.

[23] A. Davison, "Mobile robot navigation using active vision," Ph.D. dissertation, Univ. Oxford, Oxford, U.K., 1998.

[24] A. J. Davison and D. W. Murray, "Simultaneous localization and map-building using active vision," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 865–880, Jul. 2002.

[25] A. Davison and N. Kita, "3-D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recogn. (CVPR 2001)*, vol. 1, pp. I-384–I-391.

[26] L. Iocchi, K. Konolige, and M. Bajracharya, "Visually realistic mapping of a planar environment with stereo," in *Proc. Int. Symp. Exp. Robot. (ISER'00)*, pp. 521–532.

[27] S. Se, D. Lowe, and J. Little, "Mobile robot localization and mapping with uncertainty using scale-invariant visual landmarks," *Int. J. Robot. Res.*, vol. 21, no. 8, pp. 735–758, 2002.

[28] I. Jung and S. Lacroix, "High resolution terrain mapping using low altitude aerial stereo imagery," in *Proc. 9th Int. Conf. Comput. Vis.*, Nice, France, Oct. 13–16, 2003, vol. 2, pp. 946–951.

[29] E. Hygounenc, I. Jung, P. Soueres, and S. Lacroix, "The autonomous blimp project of LAAS-CNRS: Achievements in flight control and terrain mapping," *Int. J. Robot. Res.*, vol. 23, no. 4, pp. 473–511, 2004.

[30] J. Saez, F. Escolano, and A. Penalver, "First Steps towards Stereo-based 6DOF SLAM for the visually impaired," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recogn. (CVPR'05)-Workshops*, vol. 3, Washington, DC: IEEE Computer Society, Jun. 20–26, 2005, pp. 23–23.

[31] R. Sim, P. Elinas, M. Griffin, and J. Little, "Vision-based SLAM using the Rao–Blackwellised particle filter," in *Proc. IJCAI Workshop Reason. Uncertainty Robot. (RUR)*, Edinburgh, U.K., 2005, pp. 9–16.

[32] R. Sim, P. Elinas, and J. Little, "A study of the Rao–Blackwellised particle filter for efficient and accurate vision-based SLAM," *Int. J. Comput. Vis.*, vol. 74, no. 3, pp. 303–318, 2007.

[33] T. Lemaire, C. Berger, I. Jung, and S. Lacroix, “Vision-based SLAM: Stereo and monocular approaches,” *Int. J. Comput. Vis.*, vol. 74, no. 3, pp. 343–364, 2007.

[34] J. Sola, A. Monin, and M. Devy, “BiCamSLAM: Two times mono is more than stereo,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA 2007)*, Rome, Italy, Apr. 10–14, 2007, pp. 4795–4800.

[35] Z. Zhang and O. Faugueras, “Three-dimensional motion computation and object segmentation in a long sequence of stereo frames,” *Int. J. Comput. Vis.*, vol. 7, no. 3, pp. 211–241, 1992.

[36] N. Simond and P. Rives, “Trajectography of an uncalibrated stereo rig in urban environments,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS 2004).*, Sep. 28–Oct. 2, 2004, vol. 4, pp. 3381–3386.

[37] D. Nister, O. Naroditsky, and J. Bergen, “Visual odometry for ground vehicle applications,” *J. Field Robot.*, vol. 23, no. 1, pp. 3–20, 2006.

[38] A. Comport, E. Malis, and P. Rives, “Accurate quadri-focal tracking for robust 3-D visual odometry,” in *Proc. IEEE Int. Conf. Robot. Autom., (ICRA 2007)*, Roma, Italy, Apr. 10–14, 2007, pp. 40–45.

[39] M. Maimone, Y. Cheng, and L. Matthies, “Two years of visual odometry on the mars exploration rovers,” *J. Field Robot.*, vol. 24, no. 3, pp. 169–186, 2007.

[40] M. Agrawal and K. Konolige, “Real-time localization in outdoor environments using stereo vision and inexpensive GPS,” in *Proc. Int. Conf. Pattern Recogn. (ICPR 2006)*, Hong Kong, vol. 3, pp. 1063–1068.

[41] K. Konolige, M. Agrawal, R. Bolles, C. Cowan, M. Fischler, and B. Gerkey, “Outdoor mapping and navigation using stereo vision,” presented at the Int. Symp. Exp. Robot., Rio de Janeiro, Brazil, Jul. 2006.

[42] J. Castellanos, R. Martinez-Cantin, J. Tardós, and J. Neira, “Robocentric map joining: Improving the consistency of EKF-SLAM,” *Robot. Auton. Syst.*, vol. 55, no. 1, pp. 21–29, Jan. 2007.

[43] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, “Simultaneous localization and mapping with sparse extended information filters,” *Int. J. Robot. Res.*, vol. 23, no. 7/8, pp. 693–716, 2004.

[44] R. Eustice, M. Walter, and J. Leonard, “Sparse extended information filters: Insights into sparsification,” in *Proc. IEEE/RJS Int. Conf. Intell. Robots Syst.*, Edmonton, AB, Canada, 2–6, Aug. 2005, pp. 3281–3288.

[45] U. Frese, *Treemap: An o(logn) Algorithm for Simultaneous Localization and Mapping*. New York: Springer-Verlag, 2005, pp. 455–476, ch. Spatial Cognition IV.

[46] U. Frese, “Efficient 6DOF SLAM with treemap as a generic backend,” in *Proc. IEEE Int. Conf. Robot. Autom.* Apr. 10–14, 2007, pp. 4814–4819.

[47] F. Dellaert and M. Kaess, “Square root SAM: Simultaneous localization and mapping via square root information smoothing,” *Int. J. Robot. Res.*, vol. 25, no. 12, pp. 1181–1203, Dec. 2006.

[48] M. Kaess, A. Ranganathan, and F. Dellaert, “iSAM: Fast incremental smoothing and mapping with efficient data association,” in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA 2007)*, Roma, Italy, Apr. 10–14, 2007, pp. 1670–1677.

[49] J. D. Tardós, J. Neira, P. M. Newman, and J. J. Leonard, “Robust mapping and localization in indoor environments using sonar data,” *Int. J. Robot. Res.*, vol. 21, no. 4, pp. 311–330, 2002.

[50] L. M. Paz, J. D. Tardós, and J. Neira, “Divide and conquer: EKF SLAM in $O(n)$,” *IEEE Trans. Robot.*, vol. 24, no. 5, Oct. 2008.

[51] P. Piniés and J. D. Tardós, “Large scale SLAM building conditionally independent local maps: Application to monocular vision,” *IEEE Trans. Robot.*, vol. 24, no. 5, Oct. 2008.

[52] J. Civera, A. J. Davison, and J. M. M. Montiel, “Inverse depth parametrization for monocular SLAM,” *IEEE Trans. Robot.*, vol. 24, no. 5, Oct. 2008.

[53] A. J. Davison, “Real-time simultaneous localisation and mapping with a single camera,” in *Proc. Int. Conf. Comput. Vis.*, Nice, France, Oct. 13–16, 2003, vol. 2, pp. 1403–1410.

[54] L. M. Paz, P. Piniés, J. D. Tardós, and J. Neira, “Measurement equation for inverse depth points and depth points,” Dept. Inf. e Ingeniería de Sistemas, Univ. de Zaragoza, Zaragoza, Spain, Internal Rep. RR-08-06, 2008.

[55] J. Civera, A. Davison, and J. Montiel, “Inverse depth to depth conversion for monocular SLAM,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Roma, Italy, Apr. 10–14, 2007, pp. 2778–2783.

[56] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York: Springer-Verlag, 2006.

[57] J. Neira and J. D. Tardós, “Data association in stochastic mapping using the joint compatibility test,” *IEEE Trans. Robot. Autom.*, vol. 17, no. 6, pp. 890–897, Dec. 2001.

[58] B. Williams, P. Smith, and I. Reid, “Automatic relocalisation for a single-camera simultaneous localisation and mapping system,” in *Proc. IEEE Int. Conf. Robot. Autom.*, Roma, Italy, Apr. 10–14, 2007, pp. 2784–2790.

Lina M. Paz (M’08) was born in Cali, Colombia, in 1980. She received the M.S. degree in electronic engineering from the Universidad del Valle, Cali, in 2003. Since 2004, she has been working toward the Ph.D. degree in computer science with the Department of Computer Science and Systems Engineering, University of Zaragoza, Zaragoza, Spain.

Her current research interests include mobile robotics, computer vision for environment modeling, and simultaneous localization and mapping (SLAM).

Pedro Piniés (M’08) was born in Bilbao, Spain, in 1979. He received the M.S. degree in telecommunication engineering in 2004 from the University of Zaragoza, Zaragoza, Spain, where he is currently working toward the Ph.D. degree with the Robotics, Perception, and Real Time Group.

His current research interests include simultaneous localization and mapping (SLAM), mobile robotics, computer vision, and probabilistic inference.

Juan D. Tardós (M’05) was born in Huesca, Spain, in 1961. He received the M.S. and Ph.D. degrees in electrical engineering from the University of Zaragoza, Zaragoza, Spain, in 1985 and 1991, respectively.

He is currently a Full Professor with the Department of Computer Science and Systems Engineering, University of Zaragoza, where he is in charge of courses in robotics, computer vision, and artificial intelligence. His current research interests include simultaneous localization and mapping (SLAM), perception, and mobile robotics.

José Neira (M’07) was born in Bogotà, Colombia, in 1963. He received the M.S. degree from the Universidad de los Andes, Bogotá, in 1986 and the Ph.D. degree from the University of Zaragoza, Zaragoza, Spain, in 1993, both in computer science.

He is currently an Associate Professor with the Department of Computer Science and Systems Engineering, University of Zaragoza, where he teaches courses in compiler theory, computer vision, and mobile robotics. His current research interests include autonomous robots, data association, and environment modeling.

CI-Graph: An efficient approach for Large Scale SLAM

Pedro Piniés, Lina M. Paz, Juan D. Tardós

Abstract— **When solving the Simultaneous Localization and Mapping (SLAM) problem, submapping and graphical methods have shown to be valuable approaches that provide significant advantages over the standard EKF solution: they are faster and can produce more consistent estimates when using *local coordinates*. In this paper we present *CI-Graph*, a submapping method for SLAM that uses a graph structure to efficiently solve complex trajectories reducing the computational cost. Unlike other submapping SLAM approaches, we are able to transmit and share information through maps in the graph in a consistent manner by using *conditionally independent* submaps. In addition, the current submap always summarizes, without further computations, all information available making *CI-Graph* be an intrinsically "up to date" algorithm. Moreover, the technique is also efficient in memory requirements since it does not need to recover the full covariance matrix. To evaluate *CI-Graph* performance, the method has been tested using a synthetic Manhattan world and Victoria Park data set.**

I. INTRODUCTION

Essential tasks in mobile robotics strongly rely, not only on a precise estimation of the robot location, but also, on an accurate map estimate of the surrounding environment. Simultaneous Localization and Mapping algorithms (SLAM) confront both problems in a single estimation process. The first consistent solution proposed was based on the Extended Kalman Filter (EKF) [1], [2]. However, an standard implementation of the algorithm suffers from memory and time complexities of $O(n^2)$ per step, where n is the total number of features stored in the map. To reduce the computational cost, new algorithms take profit of the fact that SLAM is a sparse problem, i.e., from a given robot position only a limited number of features is visible. If all features were always visible then no algorithm could overcome the computational complexity of the EKF solution since the linearized system to be solved would be completely full.

Submapping strategies have become interesting approaches since they work in small regions of the environment reducing the computational cost of EKF and improving consistency. Under the assumption of white noise and if no information is shared between maps, submaps are statistically independent. This allows submaps to be consistently joined using Map Joining algorithm [3] or equivalent Constrained Local Submap Filter (CLSF) [4] with joining cost $O(n^2)$. More recently, Divide and Conquer SLAM [5] has shown to provide a more efficient strategy to join local maps with amortized linear cost in exploration, outperforming past sequential methods. Despite its high scalability, the main limitations of these techniques are their inability to share information between maps and a memory cost of $O(n^2)$.

Pedro Piniés, Lina M. Paz, Juan D. Tardós are with the Departamento de Informática e Ingenieria de Sistemas, Centro Politécnico Superior, Universidad de Zaragoza, Zaragoza, Spain {ppinies, linapaz, tardos}@unizar.es

There are submapping techniques that work on approximations trading off precision for complexity properties [6]. Some of these techniques combine submaps with a graph structure that represents adjacency relations between maps. In ATLAS [7] and CTS [8] for example, nodes of the graph correspond to submaps and links between nodes represent relative locations between adjacent submaps. However, in order to achieve high efficiency, they do not impose loop constraints to update the graph estimation. Hierarchical SLAM [9] outperforms these approaches by introducing an optimization step along the cycles of the graph. Nevertheless, it still remains as an approximate algorithm since optimized information is not transmitted to submaps.

In contrast to EKF-based approaches, there is a family of algorithms that considers the full SLAM problem in a Smoothing and Mapping (SAM) sense. Graph SLAM and Square Root SLAM [10], [11], report that the intrinsic structure of the problem can be modeled as a *sparse graph* (obtained from the sparse information matrix) when the state vector is augmented with the total trajectory. The main problem of these techniques is that they continuously grow with the number of robot poses.

Based on EKF, Graphical SLAM [12], builds a compressed graph of all robot and features poses as nodes. However, special cases as loop closings need particular manipulations. Treemap [13], is based on creating a *balanced binary tree* structure of the map. The technique uses a detailed graph granularity to construct the tree, where leaf nodes represent each map entity (current robot and feature locations) resulting in a very complete but complex graph algorithm.

In this work we are interested in methods that do not use any approximations. We consider the SLAM problem as a Gaussian Graph model that evolves over time. We propose *CI-Graph* SLAM based on [14], a submapping method that performs EKF updates efficiently reducing its quadratic cost. Unlike other non-approximated submapping approaches [3], [4], *CI-Graph* SLAM builds a spanning tree of *conditionally independent* submaps, that allows us to transmit information between submaps in a consistent manner. Compared to batch algorithms [10], [11], *CI-Graph* does not require to augment the state vector with the full trajectory. Instead, only robot poses corresponding to map transitions are considered. In *CI-Graph*, the nodes do not represent each element of the map but the CI-submaps. This results in a graph with high level abstraction of the map that allows a simpler implementation.

Section II is devoted to review conditionally independent submaps and describes their advantages. CI-Graph approach is presented in section III. We evaluate the method using a synthetic Manhattan world and Victoria Park data set. Their results are described in section IV. In section V we discuss concerns related to the construction of the map spanning tree. Finally, we summarize the main advantages of our method in section VI and draw future lines of work.

II. CONDITIONALLY INDEPENDENT (CI) SUBMAPS

In submapping algorithms, instead of dealing with a single total map of an environment, the whole map is divided into groups of state vector entities (features and/or vehicle poses) that are processed separately. We call *absolute submap*, to a map that is expressed in a global coordinate frame while a *local submap* is a submap whose elements are represented with respect to a local reference frame. Most recent submapping techniques are based on building local maps of limited size that are statistically *independent* [3], [4], [15], [5]. This requirement imposes important constraints to the submaps structure. Valuable information present in a submap cannot be used to improve other submap estimates since, otherwise, the independence property could not be preserved. In addition, same environment features observed in different maps have independent estimations in each map.

Instead of using independent submaps, our *CI-Graph* SLAM approach is based on building *conditionally independent* CI-submaps. The previous technique was presented in [14] and allows CI-submaps to share submap components and information in a consistent manner. Using absolute submaps, the final map obtained is the same as with the classical EKF-SLAM algorithm. If local submaps are used better consistency properties than the EKF can be achieved. While techniques based on independent submaps preclude the use of inertial sensors or sensors that give absolute measurements such as GPS and compass, our method can easily use these devices by propagating the information through CI-submaps without approximations. At the same time, CI-submaps inherit the computational efficiency of submapping techniques that, taking into account a subgroup of the map elements, can work with covariance/information submatrices of limited size.

The technique presented in [14] is restricted to sequences of maps forming simple topologies such as single loops. Even though it has been successfully tested in large environments, the generalization to more complex topologies in which the CI property between maps still hold is not trivial. The purpose of this paper is to develop a new algorithm that extents the properties of the CI-submaps to more complicated trajectories. In order to facilitate the explanation, we will work for the rest of the paper with *absolute submaps*, although *local submaps* can be used as well with the technique.

A. Brief CI-submaps review

Figure 1 shows a Bayesian Network that represents the stochastic dependencies between a pair of CI-submaps, $\mathbf{x}_1$ and $\mathbf{x}_2$, that have been built sequentially.

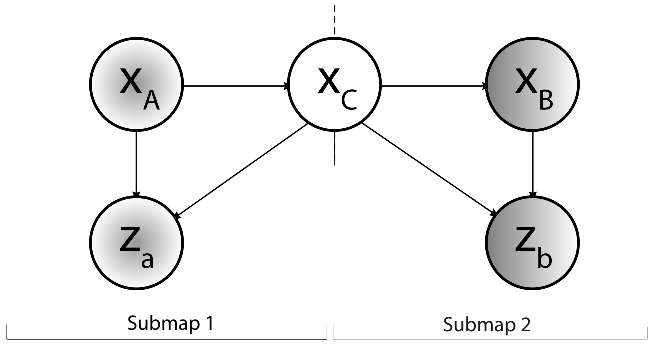


Fig. 1. Bayesian Network that describes the probabilistic dependencies between CI-submaps.

We define the state vectors of the submaps by:

$$\mathbf{x}_1 = \begin{bmatrix} \mathbf{x}_A \\ \mathbf{x}_C \end{bmatrix} \qquad \mathbf{x}_2 = \begin{bmatrix} \mathbf{x}_B \\ \mathbf{x}_C \end{bmatrix} \tag{1}$$

where $\mathbf{x}_A$ represents state components that only belong to the first map, $\mathbf{x}_B$ is for elements exclusively included in the second submap and $\mathbf{x}_C$ represents features and vehicle states that are shared in common between both. Notice that common elements $\mathbf{x}_C$ are replicated in $\mathbf{x}_1$ and $\mathbf{x}_2$. This division of the stochastic state variables can be done in SLAM because it is a sparse problem (features are locally observable).

In figure 1, we can also observe that the only connection between the set of nodes $(\mathbf{x}_A, \mathbf{z}_a)$ and $(\mathbf{x}_B, \mathbf{z}_b)$ is through node $\mathbf{x}_C$ that, according to Bayesian Network theory [16], means that both subgraphs are *d-separated* given $\mathbf{x}_C$. This in turn implies that the components of the submaps are *Conditionally Independent (CI)* when $\mathbf{x}_C$ is known:

$$\begin{aligned} p(\mathbf{x}_A|\mathbf{x}_B, \mathbf{x}_C, \mathbf{z}_a, \mathbf{z}_b) &= p(\mathbf{x}_A|\mathbf{x}_C, \mathbf{z}_a) \\ p(\mathbf{x}_B|\mathbf{x}_A, \mathbf{x}_C, \mathbf{z}_a, \mathbf{z}_b) &= p(\mathbf{x}_B|\mathbf{x}_C, \mathbf{z}_b) \end{aligned} \tag{2}$$

It is precisely by means of the common elements between maps and the CI Property that we can easily transmit information between map pairs *at any time* with no approximations. Suppose now that during the sequential creation of submaps $\mathbf{x}_1$ and $\mathbf{x}_2$, the first submap $\mathbf{x}_1$ was built using observations $\mathbf{z}_a$ whereas for $\mathbf{x}_2$, in addition to $\mathbf{z}_a$ measurements, new observations $\mathbf{z}_b$ were taken into account. Assuming Gaussian distributions for the map states we have:

$$p(\mathbf{x}_A, \mathbf{x}_C|\mathbf{z}_a) \sim \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{A_a} \\ \hat{\mathbf{x}}_{C_a} \end{bmatrix}, \begin{bmatrix} P_{A_a} & P_{AC_a} \\ P_{CA_a} & P_{C_a} \end{bmatrix} \right) \tag{3}$$

$$p(\mathbf{x}_C, \mathbf{x}_B|\mathbf{z}_a, \mathbf{z}_b) \sim \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{C_{ab}} \\ \hat{\mathbf{x}}_{B_{ab}} \end{bmatrix}, \begin{bmatrix} P_{C_{ab}} & P_{CB_{ab}} \\ P_{BC_{ab}} & P_{B_{ab}} \end{bmatrix} \right) \tag{4}$$

where the lowercase subindexes in the estimates $\mathbf{x}_{B_{ab}}$ and $\mathbf{x}_{C_{ab}}$ reveal that both sets of observations $\mathbf{z}_a$ and $\mathbf{z}_b$ have been used in the estimation process.

Notice that map $\mathbf{x}_1$ is 'out of date' with respect to map $\mathbf{x}_2$ since the influence of new observations $\mathbf{z}_b$ is not included in the estimate. In order to update submap $\mathbf{x}_1$ information from $\mathbf{x}_2$ has to be transmitted. The operation of transmitting information is called *back-propagation* where 'back' means propagation from the updated to the out of date map. To update submap $\mathbf{x}_1$ we only need to recalculate the state vector and covariance matrix of those elements related to $\mathbf{x}_A$. The *back-propagation* equations are given by:

$$\begin{aligned} K &= P_{AC_a} P_{C_a}^{-1} \\ &= P_{AC_{ab}} P_{C_{ab}}^{-1} && (5) \\ P_{AC_{ab}} &= K P_{C_{ab}} && (6) \\ P_{A_{ab}} &= P_{A_a} + K(P_{CA_{ab}} - P_{CA_a}) && (7) \\ \hat{\mathbf{x}}_{A_{ab}} &= \hat{\mathbf{x}}_{A_a} + K(\hat{\mathbf{x}}_{C_{ab}} - \hat{\mathbf{x}}_{C_a}) && (8) \end{aligned}$$

Observe that to update the first submap we only need the mean and covariance of the common elements $\hat{\mathbf{x}}_{C_{ab}}$ and $P_{C_{ab}}$ from the second submap. We can also calculate the correlation between non-common elements of both maps $\mathbf{x}_A$ and $\mathbf{x}_B$ by:

$$P_{AB_{ab}} = K P_{CB_{ab}} \qquad (9)$$

III. 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* [17]. 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 as shown in figure 1, 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 all graph figures of the paper, 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.

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 1.

Algorithm 1 : `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 III-C}
    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 III-A}
    $addNode(\mathbf{m}_j, \mathcal{N})$
    $addEdge(\langle \mathbf{m}_i, \mathbf{m}_j \rangle, \mathcal{E}_\mathcal{T})$
    $copyRobot(\mathbf{m}_i, \mathbf{m}_j)$
    $copyActiveFeat(\mathbf{m}_i, \mathbf{m}_j)$
    $i = j$ {Map change}
  end if
  if reobserved $\mathbf{f} \notin \mathbf{m}_i$ & $\mathbf{f} \in \mathbf{m}_j$ then
    {Subsection III-B}
    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 III-D}
$updateAllMaps(\mathbf{m}_i, \mathcal{T})$ {Updates $\mathcal{T}$ starting from $\mathbf{m}_i$}
```

A. 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 2. 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

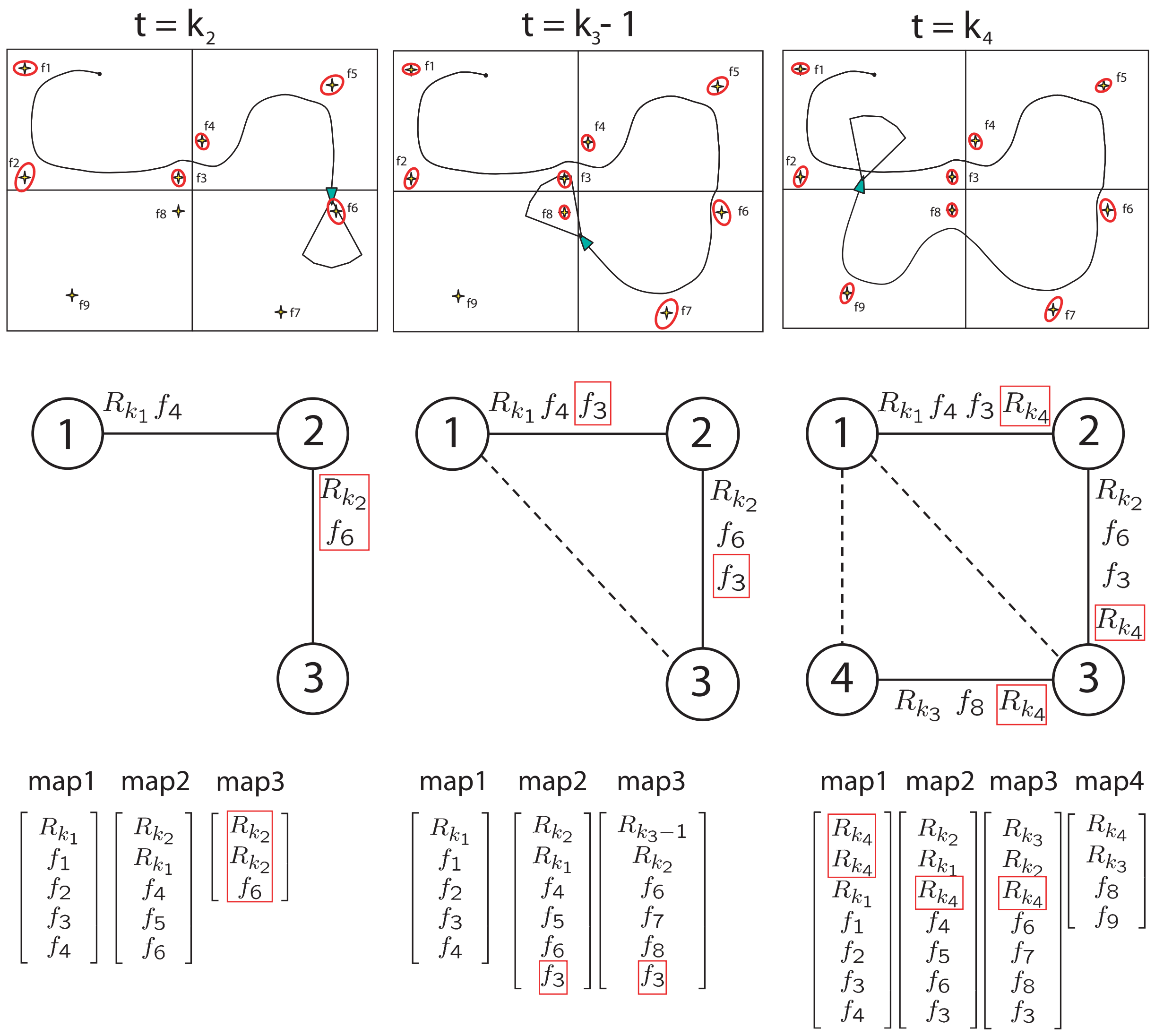


Fig. 2. Example using CI-Graph SLAM. The figure is divided in three rows that show information about the state of a simulated experiment at three different instants of time (columns). In the first row, the map of the simulated environment with the current robot position is shown. In the second row, the 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.

with robot R_{k_2} and feature f_6 from submap 2.

B. 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 (5-8) and the correlations with the elements of $\mathbf{m}_l$ are calculated with equation (9).

Figure 2 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 $\mathcal{E}_\mathcal{G} \backslash \mathcal{E}_\mathcal{T}$.

C. 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 (5-8) and to copy the current robot pose, correlations with submaps elements are calculated with equation (9).

Figure 2 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.

D. Updating all maps from the current submap

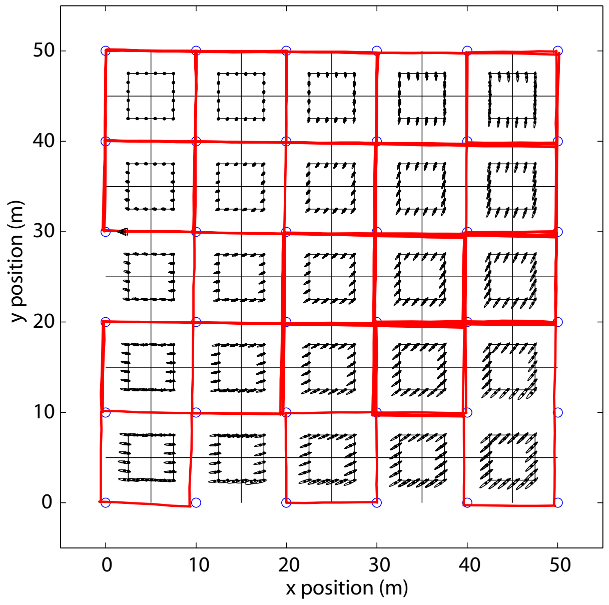
Using the Graph operations just described, we can assure that the current submap is always updated with all available information. In addition, the CI property between submaps is preserved. An interesting property of the back-propagation equations is that they can be applied at any moment. They work correctly even if we back-propagate twice the same information: the terms inside the parentheses in equations (7,8) will be zero and the maps will remain unchanged. 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 $\mathcal{T}$.

IV. EXPERIMENTS AND RESULTS

CI-Graph SLAM has been tested using a simulated environment that emules a Manhattan World, as the one proposed in [11], with 2420 point features lying on the walls of a 11×11 matrix of building blocks. For this $2D$ example, the total space is divided in submaps using a grid cell. When the robot crosses the border between two cells for the first time a new submap is initialized. If the arriving cell has already been traversed we consider that a previous submap is revisited. The actual size of each submap, is not limited to the number of features content in a cell but to the number of features that are *observed* from it. For more general situations, $3D$ environments with complex topologies and different kind of sensors such as cameras, the decision to start a new submap can be based on the maximum number of features allowed in a a map, to limit computational complexity, or on a maximum value for the uncertainty of the robot position, to improve the consistency of the result.

In the Manhattan environment, the vehicle performs a randomly chosen trajectory of 1600 steps of $1m$. Fig. 3 top, shows a smaller 5×5 example to give the reader an idea of the experiment. Each time the vehicle reaches a block corner (circles), a control input is applied randomly. This kind of motions allows the robot to perform any trajectory: the robot can move from one map to its neighbor and performs any large loops. The motion model noises are assumed to be gaussian with respectively $\sigma_{xy} = 0.05m$ and $\sigma_\theta = 0.3deg$ standard deviations in position and orientation. As the robot moves, the graph of CI-submaps is created on the fly. Fig. 3 bottom, shows an example of the resulting spanning tree with nodes numerated in the order they were created.

In this simulated experiment, Monte Carlo runs are particularly suitable to evaluate the CI-Graph SLAM efficiency.



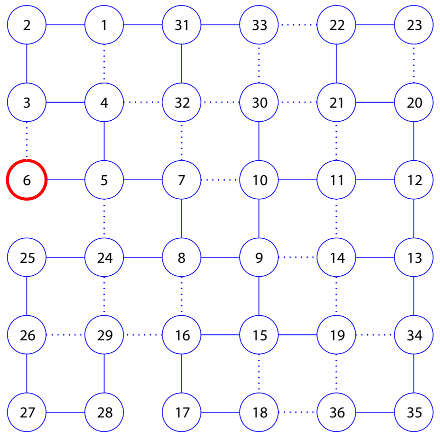


Fig. 3. CI-Graph SLAM execution on a Simulated environment of a 5×5 matrix of building blocks of a Manhattan World (top). The environment is divided up into 36 submaps (nodes) using a cell grid. The darker-red line represents the estimated trajectory. Ellipses also shows the estimated feature uncertainties. The final CI-Graph contains direct links (continuos lines) that forms the spanning tree between nodes (bottom). Indirect links are shown in dashed lines such that their represents mutual information seen between adjacent nodes. Thus, there exist a path formed by direct links through which the information can be transmitted.

We ran 300 samples of our algorithm implemented in MATLAB on a Pentium IV at $2.8GHz$. Note that each sample represents a different random trajectory and so, a different spanning tree. For the same reason, the number of total mapped features varies although the environment remains unmodified. Fig. 4 shows the mean running time per step. We can see that, for this kind of environment, the algorithm presents a close to linear running time.

We have also tested *CI-Graph* SLAM on Victoria Park data set as it is considered a benchmark for most of the relevant approaches previously mentioned in section I. Additionally, Victoria Park data set suits well due to its complex trajectory topology. As in the Manhattan World we use a grid cell to divide the space in submaps. Fig. 5

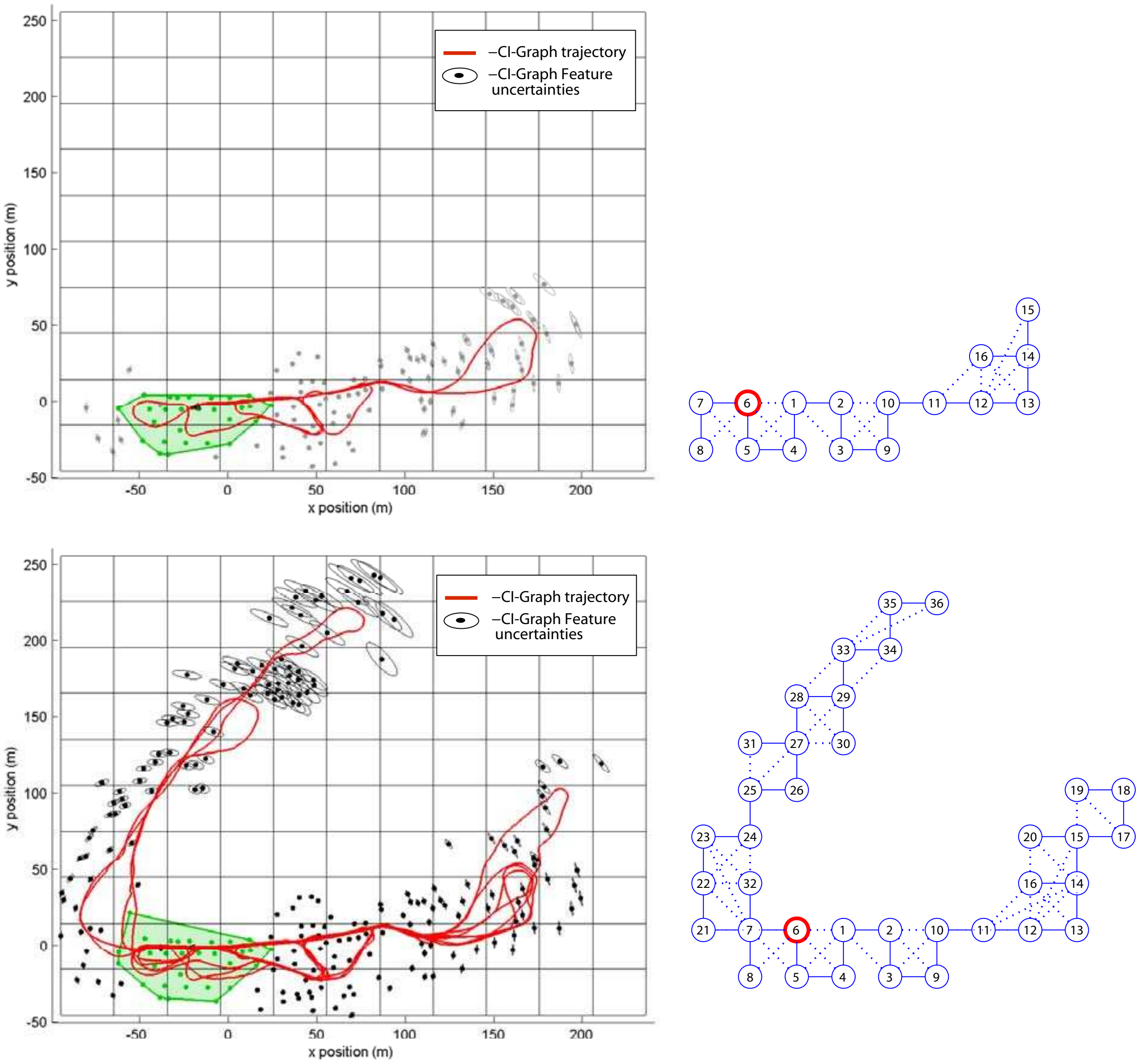


Fig. 5. *CI-Graph* SLAM execution on Victoria Park data set. The robot traverses local regions across the environment (left). During exploration, only the last submap is updated with all information available (top). At the same time, the spanning tree is being created with darker node as the current visited submap (right). The final result is obtained after building 36 nodes (submaps) using $30m$ of resolution in $x - y$ dimensions for our cell-grid (bottom). The accompanying video `video_CIGraph_xvid.mpg` (high quality version available http://webdiis.unizar.es/∼ppinies/video_CIGraph_xvid.avi) shows a slow execution of *CI-Graph* SLAM.

top, shows a partial result when the vehicle explores the environment with known data association. The accompanying video `video_CIGraph_xvid.avi` shows an execution in slow motion to visualize the CI-submaps building process. At the same time, it is posible to see that information is transmitted through direct links of the spanning tree when a node is discovered or when any two nodes share information. In addition, we ran EKF SLAM for comparison purposes. Figure 7 top, shows the running time per step, pointing out a constant behavior for the *CI-Graph* approach. Small peaks in the plot (up to $0.16sec$) represents the steps in which *CI-Graph* performs propagation. At final step, *CI-Graph* computes all optimum submaps propagating the information in $0.12sec$, one third of the time required for EKF SLAM ($0.37sec$). The map obtained is exactly the same as EKF SLAM with an absolute error difference of order 10^{-10} between vector estimates and of order 10^{-11} between estimated diagonal covariances. The differences are mainly due to numerical errors rather than estimation errors. The total cost plot in fig. 7 bottom, shows evidence about the efficient performance of *CI-Graph* SLAM as it is 6.5 times faster than standard EKF SLAM. From fig. 7 top, we could point out a constant behavior of the time per step. However it is difficult to establish this as an insight since the environment presents a disperse feature distribution. The accompanying video `video_CIGraph_EKF_xvid.avi` shows both algorithm executions with same data association both running at their corresponding execution times. A convex hull is

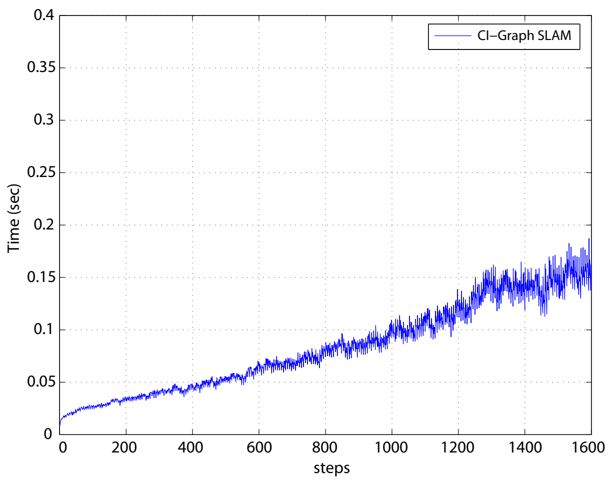


Fig. 4. Average running time per step for 300 random runs in 11×11 Manhattan World experiment. The environment is mapped with total size in a interval of $n = 947$ to $n = 2199$ point features. Also, for each run, the submap size does not exceeds 200 features in average.

drawn around the current submap considering its vehicle and features estimates while submaps already built are drawn in soft color to see the covered total map. The final updated result coincides with the final EKF map. Google Earth tool is used to show map precision on a real image of the environment (see fig. 6).

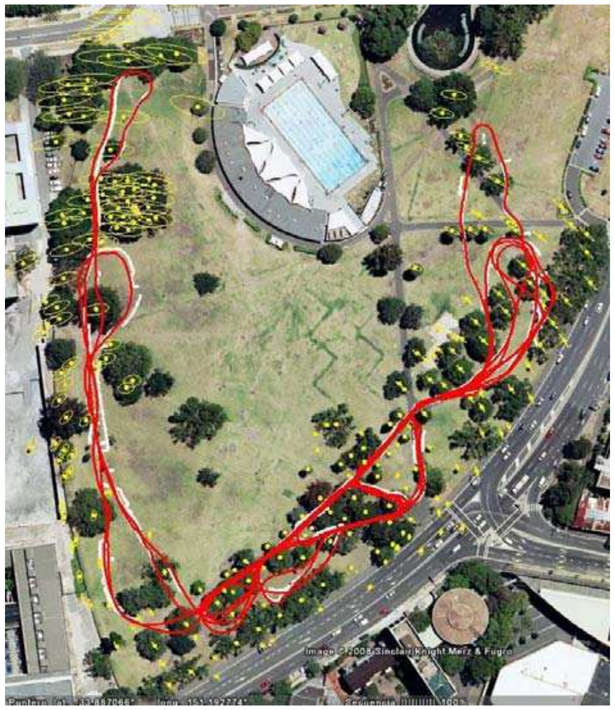


Fig. 6. Final Map obtained after running *CI-Graph* SLAM on Victoria Park data set. The map is projected using google Earth tool together with GPS data (white points).

V. DISCUSSION

The price paid to maintain the conditional independence between submaps is some overhead in the size of the maps. We call overhead to all those elements of a submap that cannot be observed from it, i.e., robot positions corresponding to the transitions between submaps and features included in the current submap because its node is in a path in $\mathcal{T}$ between two nodes that share the features. For example, in figure 2 at $t = k_4$, robot position R_{k_4} is an overhead element for submaps 2 and 3. However features f_3 and f_4 are considered as intrinsic features of submaps 1, 2 and 3 since they can be observed from them. To reduce the overhead due to the robot positions, relocation methods could be applied once the submaps are well estimated. Regarding replicated features, the overhead is clearly dependent on the spanning tree used to transmit information through the graph.

Figure 8 on the left shows a graph with six submaps and its corresponding spanning tree represented with a continuous line linking nodes $5 - 4 - 1 - 2 - 3 - 6$. Suppose now that we are in submap 5 and we observe a feature in submap 6 closing the loop. In order to maintain the CI property between submaps, instead of directly closing the loop introducing a continuous link between nodes 5 and 6 we indirectly close the loop by replicating the observed feature along the submaps in the path between both nodes, as was explained in subsection III-B. This has the drawback of increasing the size of all the intermediate submaps increasing the computational cost. A better alternative would be to change the spanning tree to one with shorter loops as the example shown in Fig. 8 right. The new spanning tree has better properties because common elements between nodes linked with a dashed edge are only locally replicated. For example, common information between nodes 3 and 6 is just replicated in nodes 4 and 5. Therefore, in order to reduce the overhead, it is more convenient to generate a spanning tree with small loops between nodes connected with dashed edges. It is important to point out that the estimated solution obtained with any of the possible spanning trees is exactly the same and, in case of using absolute submaps, identical to the solution obtained with the EKF. The only difference is that the overhead introduced by the replicated features can be reduced and therefore we can improve the computational behavior of the algorithm. A method to find a good spanning tree for a given SLAM graph or how to online change a given spanning tree to a better one is left as future work.

VI. CONCLUSIONS

In this paper we have presented *CI-Graph*, an extension of the CI-submaps that allows us to efficiently solve complex map/trajectory topologies reducing the computational cost without approximations. *CI-Graph* models the SLAM process as a Gaussian Graph that evolves over time. Nodes of the graph correspond to CI-submaps and links between nodes reveal submap relations due to either robot transitions or covisible features. This results in a high level abstraction graph that allows a simple implementation. By building a spanning tree of the graph we have shown that information can be shared and transmitted from map to map without loosing the conditional independence property between submaps.

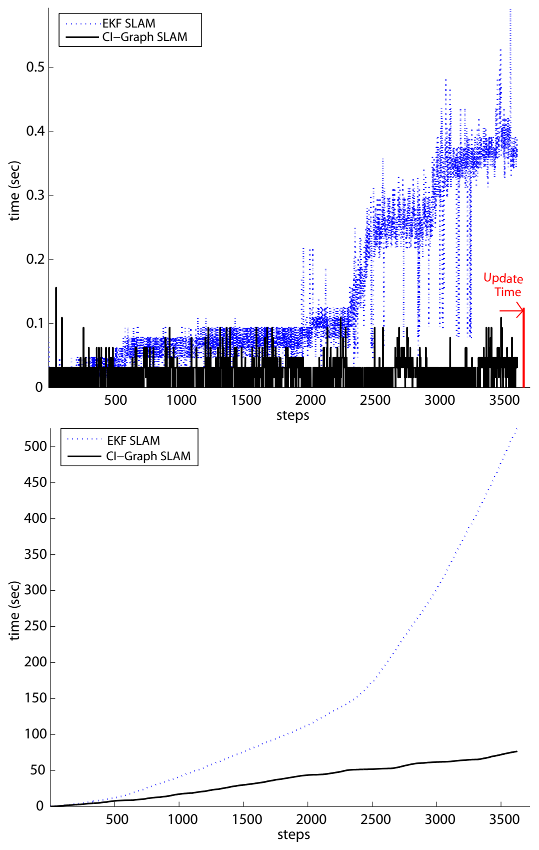


Fig. 7. *CI-Graph* SLAM vs. EKF SLAM running times. Time per step (top). Last time value corresponds to the time due to updating all submaps. Total execution time of EKF SLAM vs. *CI-Graph* SLAM(bottom). The accompanying video `video_CIGraph_EKF_xvid.mpg` (high quality version available in http://webdiis.unizar.es/∼ppinies/video_CIGraph_EKF_xvid.avi) shows that *CI-Graph* SLAM outperforms 6.5 times EKF SLAM. Additionally, *CI-Graph* only requires one third of the time required for EKF SLAM to compute the optimum map.

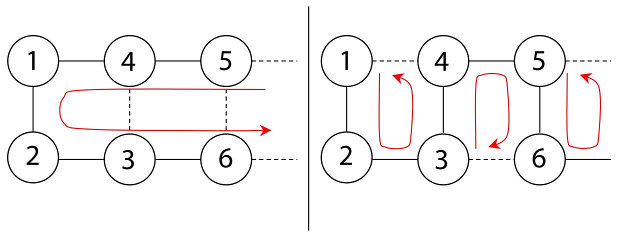


Fig. 8. A bad (left) and a good (right) spanning trees for the same graph.

One of the advantages of using *CI-Graph* with respect to other approaches, is its ability to reduce memory requirements when exploring an environment as it does not need to maintain all covariance matrix entries (correlation terms in EKF SLAM). We have also shown empirically the efficiency of *CI-Graph* SLAM to perform updates. In the presented experiments, we have obtained a cost per step close to linear time in the worst case. However, mathematical proofs about computational cost bounds will be analyzed in future work. A strategy to choose a good spanning tree to efficiently transmit information will also be addressed in future research.

CI-submaps have already shown to be very suitable for applications that involve the use of cameras in large environments. Sharing well localized features and camera states between CI-submaps gives much better results than starting each new submap from scratch. In future work we expect to proof that *CI-Graph* is a very powerful technique to solve Visual SLAM in large and complex environments.

VII. ACKNOWLEDGMENTS

This research has been funded in part by the European Union under project RAWSEEDS FP6-IST-045144 and the Dirección General de Investigación of Spain under project SLAM6DOF DPI2006-13578.

REFERENCES

[1] R. Smith, M. Self, and P. Cheeseman, “A stochastic map for uncertain spatial relationships,” in *Robotics Research, The Fourth Int. Symposium*, O. Faugeras and G. Giralt, Eds. The MIT Press, 1988, pp. 467–474.

[2] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: part I,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[3] J. D. Tardós, J. Neira, P. M. Newman, and J. J. Leonard, “Robust mapping and localization in indoor environments using sonar data,” *Int. J. Robotics Research*, vol. 21, no. 4, pp. 311–330, 2002.

[4] S. B. Williams, G. Dissanayake, and H. Durrant-Whyte, “An efficient approach to the simultaneous localisation and mapping problem,” in *Proc. IEEE Int. Conf. Robotics and Automation*, vol. 1, Washington DC, 2002, pp. 406–411.

[5] L. M. Paz, J. D. Tardós, and J. Neira, “Divide and Conquer: EKF SLAM in $O(n)$,” *Aceepted in Transactions on Robotics (in Print)*, vol. 24, no. 5, October 2008. [Online]. Available: http://webdiis.unizar.es/ lpaz/publications_archivos/papers/dcslam.pdf

[6] J. J. Leonard and H. J. S. Feder, “A computationally efficient method for large-scale concurrent mapping and localization,” in *Robotics Research: The Ninth International Symposium*, D. Koditschek and J. Hollerbach, Eds. Snowbird, Utah: Springer Verlag, 2000, pp. 169–176.

[7] M. Bosse, P. M. Newman, J. J. Leonard, M. Soika, W. Feiten, and S. Teller, “An atlas framework for scalable mapping,” in *Proc. IEEE Int. Conf. Robotics and Automation*, Taipei, Taiwan, 2003, pp. 1899–1906.

[8] J. Leonard and P. Newman, “Consistent, convergent and constant-time SLAM,” in *Int. Joint Conf. Artificial Intelligence*, Acapulco, Mexico, August 2003.

[9] C. Estrada, J. Neira, and J. D. Tardós, “Hierarchical SLAM: real-time accurate mapping of large environments,” *IEEE Trans. Robotics*, vol. 21, no. 4, pp. 588–596, August 2005.

[10] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, September 2005.

[11] F. Dellaert and M. Kaess, “Square root SAM: Simultaneous localization and mapping via square root information smoothing,” *Int. J. Robotics Research*, vol. 25, no. 12, December 2006.

[12] J. Folkesson and H. Christensen, “Graphical SLAM for outdoor applications,” *Journal of Field Robotics*, vol. 23, no. 1, pp. 51–70, 2006.

[13] U. Frese, “Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping,” *Autonomous Robots*, vol. 21, no. 2, pp. 103–122, September 2006.

[14] P. Piniés and J. D. Tardós, “Large Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision,” *Accepted in Transactions on Robotics (in print).*, vol. 24, no. 5, October 2008. [Online]. Available: http://webdiis.unizar.es/ jdtardos/papers/2008_IEEE_TRO_Pinies_Tardos.pdf

[15] S. Huang, Z. Wang, and G. Dissanayake, “Sparse Local Submap Joining Filters for building large-scale maps,” *Accepted for publication in IEEE Transactions on Robotics*, 2008.

[16] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[17] T. H. Cormen, C. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, 2nd ed., T. M. Press, Ed., Cambridge, Massashusetts, 2001.

Visual SLAM with 3D Segments from Stereo

[Regular Paper]

I. Introduction

We present a visual SLAM system that builds a geometric map of a scene. The scene is observed, i.e., measured, in terms of 3D segments generated by stereo systems. This systems reconstructs the scene with the stable geometric primitive 3D segment. We can say shortly that 3D segments from stereo constitute a synthetic and representative measure of a human-built scene. The work extends the known hierarchical SLAM algorithm to deal with 3D segments from stereo, in a full 6DoF observer pose, which we believe to be appropriate for properly handling also the small imperfections present in indoor conditions, beside being necessary for outdoor applications. We start with a brief description of the sensor system and then we will present our 3D Segment Visual SLAM approach.

II. Trinocular Sensor System

The perception system we used is a system that reconstructs the scene in terms of 3D segments. In order to give out such data the system has to deal with segments since the very first (image) processing step, this is the intended meaning of the term segment-based 3D reconstruction system here used. Segments are represented by the 3D coordinates of their extrema. This choice is in agreement with realizing the intrinsic 3D nature of indoor environment. Our system has been implemented to provide 3D extrema approximated using mean and covariance by Jacobian-based uncertainty propagation of Gaussian noise, both in the pixel detection and in the projection parameters. It bases on the trinocular approach [1], [2], [3], [4], [5], [6]. The three cameras are calibrated with respect to a common reference system, located on the robot. We use a DLT technique to determine the three projection matrices. Figure 1 presents a schematic representation of a trinocular system, where: $\mathbf{D}$ is the 3D scene segment, $\mathbf{C}_i$ and $\mathbf{d}_i$ are respectively the projection center and the projection of $\mathbf{D}$ for image i.

Cameras are calibrated altogether with their covariance matrix so that 3D extrema can be estimated altogether with an associated covariance matrix, to represent the measurement uncertainty as a first order, i.e., normal, probability distribution. The projection matrices are necessary for the following steps. A features extraction process generates a set of elements (2D segments in this case) from the images. These segments are described by the coordinates of their extrema ([x1 , y1 , x2 , y2]). Then the trinocular approach is applied; it bases on trinocular constraint ([5]), to speed up the search for the corresponding segments. The last stage is the computation of the 3D segments parameters. The 3D support line is first calculated from triplet of corresponding segments. Then the 3D extrema are computed using the interpretation lines from each 2D segment; the 3D segment given in out is the intersection of all such 3D reprojected segments. In Figure 2 we show three images captured from our trinocular sensor; in particular, (c) represents (superimposed to the image from the right camera) the result after the image processing and stereo-matching step (the green 2D segments are correctly associated while, for the red ones, three corresponding 2D segments, in all images, have not been found). Figure 2(d) reports the 3D scene reconstruction from these three images by using the trinocular approach.[1]

In the following we will refer to a *view* as to the result of one activation of the perception system. A view is composed by a set of observed features and by the movement of the robot in the space w.r.t. the previous position. In our system, it

[1]Such systems are quite widespread in the computer vision and robotics communities. Our implementation differs from the original only in the use of the Fast Line Finder algorithm [7]

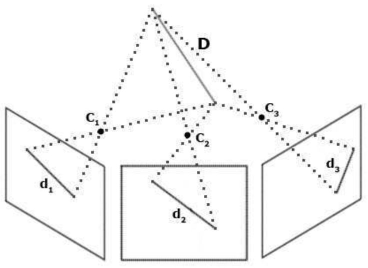


Fig. 1. 3D segment-based reconstruction for a trinocular stereoscopic system.

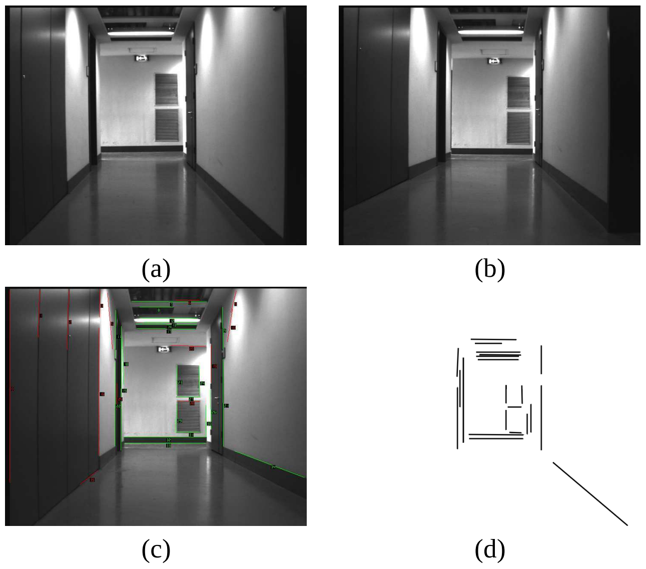


Fig. 2. Images captured from: (a): Camera left. (b): Camera top. (c): Camera right (with stereo matched segments). (d): 3D segments extraced by the sensor.

is constituted by the set of 3D segments provided by trinocular system. This segments are identified using: three parameters for each extremum of the segment (the x,y,z coordinates of the extremum in the space) and the $(6, 6)$ covariance matrix, to represent the uncertainty of the segment.

Each segment extremum is represented by a point in the 3D space w.r.t. a cartesian reference frame that is fixed w.r.t. the robot.

III. 3D-6DoF Hierarchical SLAM

In our system, we use 3D data from the perception system mentioned above and the pose is modelled as a full rigid-body transformation, so it is a 6DoF pose, turning the whole system into a 3D-6DoF Slam system, like the one using 3D points from a 3D LRF [8] or the original MonoSLAM system by Davison [9]. In this work we present a method based on hierarchical decomposition of the map [10]. In hierarchical SLAM, the main idea is to subdivide the whole map in two levels: a local level and a global level. The local level is composed by on a set of submaps, represented through Extended Kalman Filters (EKF hereafter), that have a limited size and they are treated, from a probabilistic point of view, as if they were independent. This approach allows the generation of blocks (i.e., the submaps) on which to build the global level. The global level is represented by a graph where each node describes one submap and the edges represent the existing relations between them (i.e., the relative position and its uncertainty). These two levels are the two abstraction level through which it is possible to observe the world. This subdivision, aims at limiting the influence of the errors due to the linearizzation in the EKF process; moreover it aims at reducing the computational cost. The intent of the following sections is to present this technique in the 3D6DoF case, when dealing with trinocular segment data.

A. Submap Local Structure

Each submap is generated by the integration of views through an EKF, whose state is composed by the robot pose and the location of the observed features w.r.t. the submap reference frame. Each submap has its own local reference frame, called submap base reference; all measurements refer to it (features and robot pose). Moreover, each submap is assumed to be a stochastically independent representation of a region of the space; therefore, the submap is unique and independent of all others, and, at the local level, no relation exists, to links one submap to the other. This property, as we will see in the following, has an important effect on the reconstruction process; although we might replicate the same feature in more than a map, this allows us not to calculate any correlation value between features belonging to the current local submap and the features of every other submap.

When we *initialize* a new submap, we first set its base reference on the last robot pose. This means that the current robot position becomes the origin of the new coordinate system, while its direction is used for the orientation of the y axes. The only relationship we maintain between the new submap and the previous one is represented by the estimate of the last movement of the robot in the previous submap. This information is recorded inside the global map structure as a constraint between the two submaps (the edge between the two nodes that identify the submaps). After the displacement, the perception system is activated again and a new view is generated. These data are subsequently integrated into the submap without any need for data association, since the submap does not contain previously observed feature. The initial position of the robot, when starting the whole algorithm, is initialized to the origin ($[0, 0, 0, 0, 0, 0]^T$) with a null covariance matrix, since it can be assumed to be perfectly known.

When the *robot moves*, odometry provides an estimate of the rototraslation $x_{R_k}^{R_{k-1}}$ between the previous and the actual position:

$$x_{R_k}^{R_{k-1}} = \hat{x}_{R_k}^{R_{k-1}} + \nu_k \tag{1}$$

$$E[\nu_k] = 0, \qquad E[\nu_k \nu_j^T] = \delta_{kj} Q_k \tag{2}$$

being ν the additive white zero mean odometric process error, and $\hat{x}_{R_k}^{R_{k-1}}$ the displacement estimate. We use this estimate to compute the rototraslation between the base reference of the actual submap B and the new robot pose (i.e., the prediction step in the Kalman Filter)

$$x_{R_k}^{B} = x_{R_{k-1}}^{B} \oplus x_{R_k}^{R_{k-1}} \tag{3}$$

with $\oplus$ representing the composition of transformations, and $x_{R_{k-1}}^{B}$ being the estimate of the robot pose at the previous step, w.r.t. the base reference of submap B. The new view is then used to compute the integration of the observed features with the elements in the current submap, as described in the next section.

When the integration phase is ended, we return to the exploration phase, with the consequent movement of the robot. This procedure continues until some submap closure constraint is not satisfied: then the current submap is closed and a new submap is initialized at the current robot pose. The constraints used to decide whether a submap should be closed and a new submap created, are chosen w.r.t. various factors such as attainment of a maximum number of features pertaining to the submap, overtaking of an a priori limit on the uncertainly of robot pose with respect to the base reference of the submap, nonexistence of reliable matches between last view features and current submap features, etc. In our implementation we choose to close a submap using a maximum number of features that could pertain to a submap. In this way we can keep the size of the filter state as well as the computational complexity limited, being the computational complexity of the EKF $O(n^2)$ (where n is the state dimension). It should be noticed that the parameters describing submaps and their relationships have an uncertainty matrix associated. This uncertainty arise from the observation errors during the acquisition process, this is in turn due to both the measurement noise in the image processing and the drift problem in the odometry measure.

B. Global Graph Structure

The global graph represents the highest abstraction level. This graph allows to maintain the topological relations between the various submaps that constitute the environment model. The global map is an oriented graph where each node represents an independent submap, at the local level, while the edges are the spatial relationships (rototraslations) between the base reference of one submap with respect to the base reference of the different and stocastically independent previous (or next) submap. The graph data are modeled using vectors to represent the graph edges, matrices to represent the covariance associated to the edges

$$\hat{\mathbf{x}} = \left[\begin{array}{cccccc} \hat{\mathbf{x}}_{B_1}^{B_0} & \hat{\mathbf{x}}_{B_2}^{B_1} & \ldots & \hat{\mathbf{x}}_{B_i}^{B_{i-1}} & \hat{\mathbf{x}}_{B_{i+1}}^{B_i} & \ldots \end{array} \right]^T \tag{4}$$

$$\mathbf{P} = diag(\mathbf{P}_{B_1}^{B_0}, \mathbf{P}_{B_2}^{B_1}, \cdots, \mathbf{P}_{B_{i+1}}^{B_i}, \cdots) \tag{5}$$

being $\hat{\mathbf{x}}_{B_i}^{B_{i-1}}$ the estimate of the rototraslation between the base references B_{i-1} and B_i, and $\mathbf{P}_{B_i}^{B_{i-1}}$ its covariance matrix.

IV. Data Association with 3D Segments

Before getting into the details of the local map integration process, it is worth to explain the algorithm used to perform data association. Suppose we have measured a set of features $\mathcal{V} = \{v_i\}$ from a certain robot position. Each feature is represented by its coordinates and its uncertainty. We have also the set of features $\mathcal{M} = \{m_j\}$ observed and integrated in the map up to that moment. The purpose of a data-association algorithm is to find a proper hypothesis of correspondences between them (list of couples v_i and m_j) $H_k = \{< v_1, m_{j_1} >, < v_2, m_{j_2} >, \ldots, < v_V, m_{j_V} >\}$. To generate this hypothesis we must explore an interpretation tree where each node represents a couple of corresponding features $< v_i, m_j >$. Usually the point-to-point distance is considered an appropriate criterium for single segment matching and most of the effort is devoted in finding a good association strategy for dealing with the exponential complexity of finding the best matches for the whole set of segments in the view, i.e., the best strategy for association tree traversal. In [11] we have shown that defining a better criterium for 3D segment matching will result in a better data association, and this is almost independent from the algorithm for interpretation tree traversal (i.e., data association). The approach we propose is based on a multi-criteria evaluation for associating segments in the View with segments belonging to the Submap; the reason for this change from the point-to-point criterium is mainly due to the problem of the moving-field-of-view in the sensing system, which turns in a moving window on the world feature, see Figure 3. More precisely the segments extrema are introduced by the reduced field of view of the cameras and are not

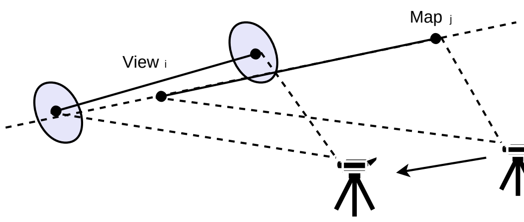


Fig. 3. The “moving window” problem.

always related to real extrema in the world; having a sensing system that moves introduce new extrema, and thus new possible segments, at each step and this can easily become a problem for the classical point-to-point distance.

Initially, we transform the measures of all features belonging to the view taken from the actual robot position $\mathbf{x}_{R_k}$ using the rototraslation between the base reference of the submap $\mathbf{x}_B$ and it.

Now we can perform data association using Mahalanobis distance, being all the measurements referred to the common submap base reference and having covariance matrixes for both view and submap elements. In particular there are three criterium to validate a hypothesis of match between the view segment F_i and the submap segment F_j.

The first criterium considers the support line of segment F_j

$$\begin{cases} x = x'_j + (x''_j - x'_j)t \\ y = y'_j + (y''_j - y'_j)t \\ z = z'_j + (z''_j - z'_j)t \end{cases} \tag{6}$$

where $[x'_j, y'_j, z'_j]^T$ are the coordinates of an extremum of segment F_j and $[x''_j, y''_j, z''_j]^T$ are the coordinates of the other. The distance of a point $\mathbf{P} = [x_i, y_i, z_i]^T$, e.g., an extremum of a F_i segment, from the line is obtained from:

$$D(\mathbf{P}, F_j) = \sqrt{\frac{A^2 + B^2 + C^2}{(x''_j - x'_j)^2 + (y''_j - y'_j)^2 + (z''_j - z'_j)^2}}, \tag{7}$$

having defined $A = det \begin{bmatrix} x_i - x''_j & y_i - y'_j \\ x''_j - x'_j & y''_j - y'_j \end{bmatrix}$, $B = det \begin{bmatrix} x_i - x''_j & z_i - z'_j \\ x''_j - x'_j & z''_j - z'_j \end{bmatrix}$, $C = det \begin{bmatrix} y_i - y''_j & z_i - z'_j \\ y''_j - y'_j & z''_j - z'_j \end{bmatrix}$. Having to deal with the two extrema of segment F_i, the considered distance is:

$$\mathbf{h}(F_i, F_j) = \begin{bmatrix} D(F'_i, F_j) \\ D(F''_i, F_j) \end{bmatrix} \tag{8}$$

where: F'_i is the first extremum of the segment F_i, F''_i is the second extremum of the segment F_i. It is now possible to compute the Mahalanobis distance between the two segments and compare it with a threshold by Chi-square test:

$$D^2_{\mathbf{h}} = \mathbf{h}^T \mathbf{C}_{\mathbf{h}}^{-1} \mathbf{h} < \chi^2_{d,\alpha} \tag{9}$$

$$\mathbf{C_h} = \mathbf{H} \mathbf{P}_{F_j} \mathbf{H}^T + \mathbf{G} \mathbf{P}_{F_i} \mathbf{G}^T \tag{10}$$

$$\mathbf{H}_{(6,2)} = \begin{bmatrix} \frac{\partial D(F'_i, F_j)}{\partial F_j} \\ \frac{\partial D(F''_i, F_j)}{\partial F_j} \end{bmatrix}, \mathbf{G}_{(6,2)} = \begin{bmatrix} \frac{\partial D(F'_i, F_j)}{\partial F_j} \\ \frac{\partial D(F''_i, F_j)}{\partial F_i} \end{bmatrix} \tag{11}$$

The second criterium checks the angle between the support lines of the two segments: this constraint complements the previous, to avoid wrong associations of a short segment to another one perpendicular to it. We can write the two support lines as:

$$\begin{cases} x = a_1 t + x'_i & a_1 = (x''_i - x'_i) \\ y = b_1 t + y'_i & b_1 = (y''_i - y'_i) \\ z = c_1 t + z'_i & c_1 = (z''_i - z'_i) \end{cases} \tag{12}$$

and

$$\begin{cases} x = a_2 t + x'_j & a_2 = (x''_j - x'_j) \\ y = b_2 t + y'_j & b_2 = (y''_j - y'_j) \\ z = c_2 t + z'_j & c_2 = (z''_j - z'_j) \end{cases} \tag{13}$$

the angle between them is:

$$angle_{ji} = cos^{-1}\left(\frac{a_1a_2 + b_1b_2 + c_1c_2}{\sqrt{a_1^2 + b_1^2 + c_1^2}\sqrt{a_2^2 + b_2^2 + c_2^2}}\right), \tag{14}$$

and for it we can use again the Chi-square test

$$D_{ji}^2 = angle_{ji}\mathbf{C}_{angle_{ji}}^{-1} angle_{ji} < \chi_{d,\alpha}^2 \tag{15}$$

$$\mathbf{C} = \mathbf{F}\mathbf{P}_{F_j}^{B}\mathbf{F}^T + \mathbf{G}\mathbf{P}_{F_i}^{R_k}\mathbf{G}^T \tag{16}$$

$$\mathbf{F} = \frac{\partial angle_{ji}}{\partial \mathbf{x}_{F_j}^{B}}, \mathbf{G} = \frac{\partial angle_{ji}}{\partial \mathbf{x}_{F_i}^{R_k}}. \tag{17}$$

The third matching criterion is not a probabilistic one; instead it is related to the projections of the segment extrema on the other segment; one of the following two conditions has to hold to get an association:

1) at least one of the extrema of the view segment has to project onto the map segment;
2) both the extrema of the view segment project outside the map segment, but the extrema of the map segment project inside the view segment.

Once defined this set of probabilistic criteria, we can use the *joint compatibility branch and bound* association algorithm to consider, in each data-association iteration, all feature association choices computed until that time and the correlation between them (see [12] for details). Indeed, the measurements are always correlated by the uncertainly on the robot position (this information is lost when we use a classical nearest neighbor approach). In this way it is possible to reconsider the previous couples of features to limit the possibility of accepting false matching. This is obtained by traversing the interpretation tree using a branch and bound method in conjunction with a joint compatibility test. The quality of a node is represented by the number of non-null association executed to reach that node.

Given a scoring function $\mathbf{f}_{H_k}$, the joint compatibility of a hypothesis H_k is determined by:

$$D_{H_k}^2 = \mathbf{f}_{H_k}^T\mathbf{C}_{H_k}^{-1}\mathbf{f}_{H_k} < \chi_{d,\alpha}^2 \tag{18}$$

$$\mathbf{C}_{H_k} = \mathbf{H}_{H_k}\mathbf{P}\mathbf{H}_{H_k}^T + \mathbf{G}_{H_k}\mathbf{S}\mathbf{G}_{H_k}^T \tag{19}$$

where $\mathbf{P}$ is the covariance of the state (robot and features pose), $\mathbf{S}$ is the covariance of the observations and:

$$\mathbf{H}_{H_k} = \frac{\partial \mathbf{f}_{H_k}}{\partial \mathcal{M}}, \mathbf{G}_{H_k} = \frac{\partial \mathbf{f}_{H_k}}{\partial \mathcal{V}}. \tag{20}$$

We integrate the three criteria in the joint compatibility branch and bound technique in the following manner. To establish the correctness of this hypothesis we use the joint compatibility function $\mathbf{f}_{H_k}$:

$$\mathbf{f}_{H_k} = \begin{pmatrix} D(v_1', m_{j_1}) \\ D(v_1'', m_{j_1}) \\ \vdots \\ D(v_i', m_{j_i}) \\ D(v_i'', m_{j_i}) \end{pmatrix}$$

being D the distance between the support line of the submap segment and the extrema of the view segment. To reduce the amount of computational time required to explore one branch, we, initially, check the third criterion on the couple $< v_i, m_{j_i} >$ because it is very selective and fast; subsequently we verify the angle criterium on the same pair, to decrease the possibility to compute in vain the next step; finally, we use the distance criteria in the joint compatibility test.

V. Data Fusion and Kalman Filtering

The fusion process is the basic component of the integration phase. Robustness and efficiency are the characteristics that allow to generate valid and consistent submaps. The purpose of this process is to integrate, in the world model, the observations obtained by the sensors (stored in the view structure).

To fuse observed data into a local submap, we use and Extended Kalman Filter. The well-known approximation problems, caused by the linearization of the non-linear equations of the fusion computation, are minimized by the use of a world model constituted by stochastically independent and limited in size submaps. This allows to reduce the errors as the errors are propagated only inside each single submap; when we start a new submap the uncertainty on the robot pose is re-initialized. The state of the filter is the union of the vectors representing the features, and the 6DoF robot pose represented by ϕ, γ, θ, i.e., the robot orientation (Euler angle III), and x, y, z, i.e., the robot cartesian coordinates, w.r.t. the submap base reference.

During the prediction, the estimate of the robot pose $\hat{x}^B_{R_{k-1}}$ at time $k-1$ is updated with the new motion value $x^{R_{k-1}}_{R_k}$. This operation is obtained by the composition of the old robot pose estimate with the odometric data.

$$\hat{\mathbf{x}}^B_{F_{k|k-1}} = \begin{bmatrix} \hat{\mathbf{x}}^B_{R_{k-1}} \oplus \hat{\mathbf{x}}^{R_{k-1}}_{R_k} & \hat{\mathbf{x}}^B_{F_{1,k-1}} & \dots & \hat{\mathbf{x}}^B_{F_{m,k-1}} \end{bmatrix}^T . \tag{21}$$

We need the state prediction as well as its uncertainty, especially for the map features in the state; this is obtained by means of the classical uncertainty propagation [13].

$$\mathbf{P}^B_{F_{k|k-1}} = \mathbf{F}_k \mathbf{P}^B_{F_{k-1}} \mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T \tag{22}$$

where $\mathbf{P}^B_{F_{k-1}}$ is the state covariance before prediction, $\mathbf{Q}_k$ is the odometric covariance,

$$\mathbf{F}_k = \begin{bmatrix} \mathbf{J_B} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix}, \mathbf{G}_k = \begin{bmatrix} \mathbf{J_R} \\ \mathbf{0} \end{bmatrix}, \tag{23}$$

having defined $\mathbf{J}_B$ as the Jacobian of the $\hat{\mathbf{x}}^B_{R_{k-1}} \oplus \hat{\mathbf{x}}^{R_{k-1}}_{R_k}$ rototraslation w.r.t. $\hat{\mathbf{x}}^B_{R_{k-1}}$ and $\mathbf{J}_R$ as the Jacobian of the $\hat{\mathbf{x}}^B_{R_{k-1}} \oplus \hat{\mathbf{x}}^{R_{k-1}}_{R_k}$ rototraslation w.r.t. $\hat{\mathbf{x}}^{R_{k-1}}_{R_k}$,

Associations, available from the previous phase, are then used to update the predicted state. We move the view data back to the submap base reference and exploit, as innovation given by the new data, the distance of the endpoints of the view segment from the support line of the associated segment in the submap. The robot pose, being part of the state, is updated according to the new view data; in long motion sequences this helps limiting to some extent the cumulative odometric errors, see Section VII.

For each pair of corresponding features (observed feature $\mathbf{F}_i$ and submap feature $\mathbf{F}_{j_i}$), the distance between the observation and the feature estimate in the state filter, is a non-linear measurement function h_{ij_i} taking as arguments the robot pose and the features $\mathbf{x}^B_F$. First of all, we need to rototraslate the measurements to refer all data to the same submap base reference:

$$\mathbf{x}^B_{F_i} = \mathbf{x}^B_{R_{k|k-1}} \oplus \mathbf{x}^R_{F_i}. \tag{24}$$

Now we can compute the distance between 3D segments (see Section IV), obtaining:

$$\mathbf{h}(\mathbf{x}^R_{F_i}, \mathbf{x}^B_{F_{j_i,k-1}}, \mathbf{x}^B_{R_{k|k-1}}) = \begin{bmatrix} D(\mathbf{x'}^B_{F_i}, \mathbf{x}^B_{F_{j_i,k-1}}) \\ D(\mathbf{x''}^B_{F_i}, \mathbf{x}^B_{F_{j_i,k-1}}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{25}$$

This is going to be used as measurement function in the Extended Kalman Filter.

Observed features that have no associated map feature, can be added to the filter state, without any fusion. They are moved into the submap, by using the estimate of the robot pose, and then became part of the state for the next view integration. The new integration process is subdivided into two steps. In the first phase, we rototraslate the new observations basing on the estimate of the robot pose (which is in state filter $\hat{\mathbf{x}}^B_{R_{k|k}}$), then we extend the filter state and its covariance matrix accordingly.

Beside the specificity of state representation and measurement equations, the described process is nothing more than a standard EKF algorithm; however a note should be raised about the 3D extrema of integrated segments. The extrema computation is executed independently on every feature; this is necessary for the reliability of the integration process because the uncertainty on the segment extrema is greater than the uncertainty on the segment direction.

When we update the estimate of the segment parameters, we need to recalculate the estimate of its 3D extrema. First of all, we compute the orthogonal projection of the view segment extrema on the support line, the one built on the submap segment (its estimate is updated before the use); the view segment is rototraslated with respect to the current robot pose, then we calculate the orthogonal projection of the view segment extrema on the 3D submap segment:

$$\hat{\mathbf{x}}'_{proj} = \begin{bmatrix} x_0 - a(\frac{a(x_0-x_p)+b(y_0-y_p)+c(z_0-z_p)}{a^2+b^2+c^2}) \\ y_0 - b(\frac{a(x_0-x_p)+b(y_0-y_p)+c(z_0-z_p)}{a^2+b^2+c^2}) \\ z_0 - c(\frac{a(x_0-x_p)+b(y_0-y_p)+c(z_0-z_p)}{a^2+b^2+c^2}) \end{bmatrix}, \tag{26}$$

where $a = (x_1 - x_0)$, $b = (y_1 - y_0)$, $c = (z_1 - z_0)$, $\mathbf{x}'_p = \begin{bmatrix} x_p & y_p & z_p \end{bmatrix}^T$ is the view segment ($\hat{\mathbf{x}}^R_{F_i}$) extremum rototraslated w.r.t. $\hat{\mathbf{x}}^B_R$ and $\hat{\mathbf{x}}^B_{F_{j_i}} = \begin{bmatrix} x_0 & y_0 & z_0 & x_1 & y_1 & z_1 \end{bmatrix}^T$ is the submap segment.

Next, for each submap segment extremum, we compute two distances: the distance between the center of mass of the segment and its extrema, and between the center of mass of the segment and the previously computed orthogonal projection. The estimate of the extrema position is updated only if the distance between the center of mass and the orthogonal projection is greater than the distance between the center of mass and the extremum. By doing so we obtain that the segment length, after the computation, will be always equal or longer then before the integration process.

The reason for this is that we do not want to loose newly and partially seen segments.

Moreover, we select the orthogonal projection to update extrema, because the sensor observations are, usually, more accurate on the measurement of the segment direction w.r.t. the segment endpoints. The orthogonal projection allows to give to the first

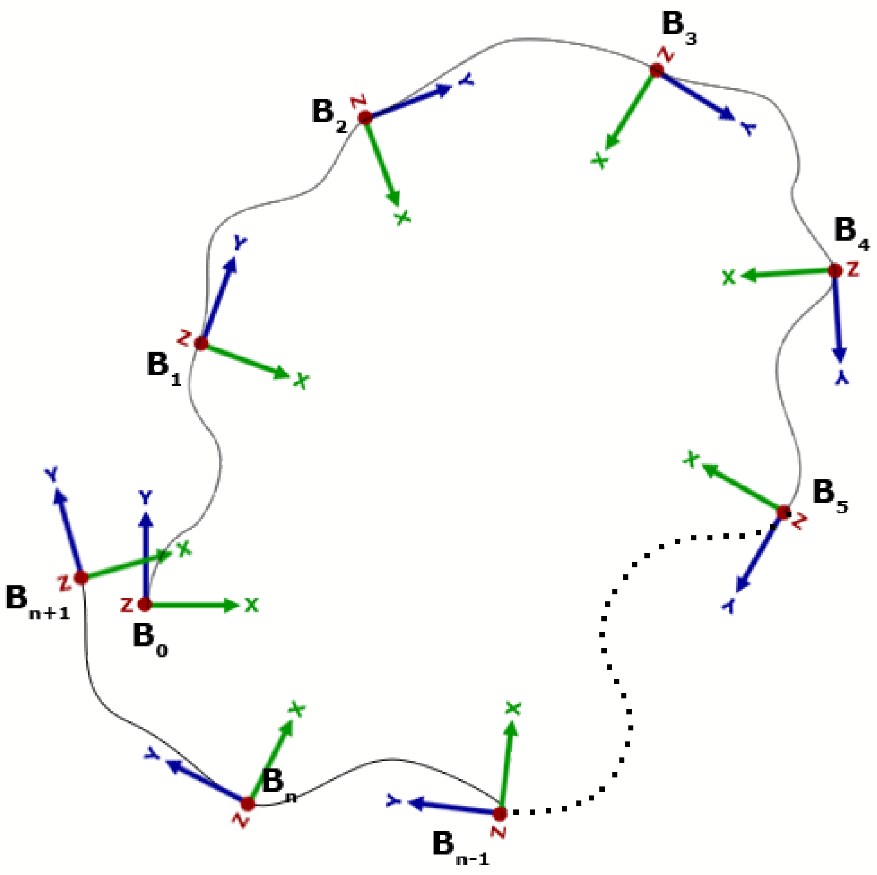


Fig. 4. Submap overlapping

piece of information a major weight with respect to the second one, in order to define the extrema. In addition we compute the new extrema covariance by means of uncertainty propagation; first we propagate the rototranslation uncertainty,

$$\mathbf{P}_p = \mathbf{F}\mathbf{P}_R^B\mathbf{F}^T + \mathbf{G}\mathbf{P}_{F_i}^R\mathbf{G}^T \tag{27}$$

where $\mathbf{F}$ is the Jacobian of the rototraslation $\hat{\mathbf{x}}_R^B \oplus \hat{\mathbf{x}}_{F_i}^R$ w.r.t. $\hat{\mathbf{x}}_R^B$ and $\mathbf{G}$ is the Jacobian of the rototraslation $\hat{\mathbf{x}}_R^B \oplus \hat{\mathbf{x}}_{F_i}^R$ w.r.t. $\hat{\mathbf{x}}_{F_i}^R$; then we propagate the uncertainty in the orthogonal projection:

$$\mathbf{P}_{proj} = \mathbf{K}\mathbf{P}_{F_{j_i}}^B\mathbf{K}^T + \mathbf{H}\mathbf{P}_p\mathbf{H}^T \tag{28}$$

where $\mathbf{K}$ is the Jacobian of the orthogonal projection w.r.t. $\hat{\mathbf{x}}_{F_{j_i}}^B$ and $\mathbf{H}$ is the Jacobian of the same orthogonal projection w.r.t. $\hat{\mathbf{x}}'_p$.

VI. Large Loop Closure

The detection of a large loop closure allows to largely improve the global level representation of the world. By merging two compatible submaps into a single one, i.e., by imposing a new constraint to the model, we are able to improve the estimate of all the submap base references involved in the loop.

Loop closure is a complex activity that involves many steps. The first step is obviously *loop Detection*, i.e., detecting a submap close to the one just closed, that could be involved in a loop closure. This is obtained by looking for a submap overlapping to some extent with the one just closed, among the submaps in the neighborhood of the last pose.

The size of the neighborhood considered is a function of submap dimension (i.e., the bounding box of its features) and of pose uncertainty after propagating the uncertainties in roto-traslations between submaps, from the first base reference:

$$\begin{aligned}
&\hat{\mathbf{x}}_{B_j}^{B_0} = \hat{\mathbf{x}}_{B_1}^{B_0} \oplus \hat{\mathbf{x}}_{B_2}^{B_1} \oplus \cdots \oplus \hat{\mathbf{x}}_{B_j}^{B_{j-1}} \\
&\mathbf{P}_{B_2}^{B_0} = \mathbf{F}\mathbf{P}_{B_1}^{B_0}\mathbf{F}^T + \mathbf{G}\mathbf{P}_{B_2}^{B_1}\mathbf{G}^T \\
&\mathbf{P}_{B_3}^{B_0} = \mathbf{H}\mathbf{P}_{B_2}^{B_0}\mathbf{H}^T + \mathbf{K}\mathbf{P}_{B_3}^{B_2}\mathbf{K}^T \\
&\ldots \\
&\mathbf{P}_{B_j}^{B_0} = \mathbf{W}\mathbf{P}_{B_{j-1}}^{B_0}\mathbf{W}^T + \mathbf{D}\mathbf{P}_{B_j}^{B_{j-1}}\mathbf{D}^T
\end{aligned} \tag{29}$$

where $\mathbf{F}$ is the Jacobian of the rototraslation $\hat{\mathbf{x}}_{B_1}^{B_0} \oplus \hat{\mathbf{x}}_{B_2}^{B_1}$ w.r.t. $\hat{\mathbf{x}}_{B_1}^{B_0}$, $\mathbf{G}$ is the Jacobian of the rototraslation $\hat{\mathbf{x}}_{B_1}^{B_0} \oplus \hat{\mathbf{x}}_{B_2}^{B_1}$ w.r.t. $\hat{\mathbf{x}}_{B_2}^{B_1}$, $\mathbf{H}$ is the Jacobian of the rototraslation $\hat{\mathbf{x}}_{B_2}^{B_0} \oplus \hat{\mathbf{x}}_{B_3}^{B_2}$ w.r.t. $\hat{\mathbf{x}}_{B_2}^{B_0}$, $\mathbf{K}$ is the Jacobian of the rototraslation $\hat{\mathbf{x}}_{B_2}^{B_0} \oplus \hat{\mathbf{x}}_{B_3}^{B_2}$ w.r.t. $\hat{\mathbf{x}}_{B_3}^{B_2}$, $\mathbf{W}$ is the Jacobian of the rototraslation $\hat{\mathbf{x}}_{B_{j-1}}^{B_0} \oplus \hat{\mathbf{x}}_{B_j}^{B_{j-1}}$ w.r.t. $\hat{\mathbf{x}}_{B_{j-1}}^{B_0}$, $\mathbf{D}$ is the Jacobian of the rototraslation $\hat{\mathbf{x}}_{B_{j-1}}^{B_0} \oplus \hat{\mathbf{x}}_{B_j}^{B_{j-1}}$ respect to $\hat{\mathbf{x}}_{B_j}^{B_{j-1}}$. The bounding box of the features is computed analyzing their extrema.

Whenever a possible loop has been detected, it is necessary to perform again *Data Association*, to verify that the two submaps match in a significant way and, at the same time, extract the common features. This is obtained by using, again, a data association procedure that seeks an new hypothesis H that connects each feature in the first submap to the corresponding feature in the second submap. This hypothesis will be used to estimate both robot pose and feature position in the submaps.

There are different approaches to find H, in this work we use an interpretation tree, see e.g. [14] and [10], with hypothesis

$$H = \begin{bmatrix} E_1 & F_{j_1} \\ E_2 & F_{j_2} \\ \vdots & \vdots \\ E_m & F_{j_m} \end{bmatrix} \tag{30}$$

where E is the first submap, F is the second submap, j_i is the index of the feature F_{j_i} associated to the feature E_i, and $j_i = 0$ when the feature E_i hasn't feature F associated.

Nodes, in this tree, represent an association between features (one pertaining to the first submap E_i and one pertaining to the second submap F_{j_i}) and each level describes all possible matching between a feature and the other ones. This structure has some important properties: the number of levels is equal to the number of features of $E(m)$; the number of branches that are coming out from each node is equal to the number of features of $F(n)$+1, called star-branch, which is used to define the non-existence of one match; the number of possible solutions is exponential in the number of features of E: $N = (n+1)^m$.

Finding a proper H turns into finding a path connecting a root of the tree to one of the leaves; to do this, we use an adapted RANSAC algorithm [15] to perform data association with the interpretation tree between submaps having respectively n and $m \leq n$ features. We generate a valid hypothesis that respects unary and binary constraints, as described below, using p out of m features randomly selected, and validate it on the remaining $m - p$ using a joint compatibility test adapted to the 3D-6DoF problem from [12], also described in the following.

Being P_g the probability of a feature, randomly selected from E submap, to have a corresponding feature in the other submap F and being P_{fail} the probability of not finding a proper H when it exist (i.e., how much we accept to fail in data association to avoid exhaustive $O\left((n-1)^m\right)$ search in the tree) the number of trials is bounded by $\lceil \frac{\log(P_{fail})}{\log(1-P_g)} \rceil$ and independent on the number of features. The number of required iterations is recalculated when a new hypothesis is generated. In this way, the algorithm behavior is adapted to the best current hypothesis.

We use two geometrical constraints to reduce the computational cost during the tree exploration, pruning and trimming some paths. The advantage in the using of strong constraints is that we can prune a larger number of branches, i.e., we can decrease the computational complexity of the creation of one hypothesis. The first geometrical constraint is named *unary constraint*. Suppose we have a pair of segments $p_{ij} = (E_i, F_j)$, pertaining to the interpretation tree,

we can impose a geometrical constraint on the length of these segments:

$$\hat{\mathbf{u}}_i = L_i = \sqrt{(x_i' - x_i'')^2 + (y_i' - y_i'')^2 + (z_i' - z_i'')^2} \tag{31}$$

$$\hat{\mathbf{u}}_j = L_j = \sqrt{(x_j' - x_j'')^2 + (y_j' - y_j'')^2 + (z_j' - z_j'')^2} \tag{32}$$

$$\mathbf{P}_i = P_{L_i} = (\frac{\partial L_i}{\partial \mathbf{x}_i'})^2 P_i' + (\frac{\partial L_i}{\partial \mathbf{x}_i''})^2 P_i'' \tag{33}$$

$$\mathbf{P}_j = P_{L_j} = (\frac{\partial L_j}{\partial \mathbf{x}_j'})^2 P_j' + (\frac{\partial L_j}{\partial \mathbf{x}_j''})^2 P_j'' \tag{34}$$

by using a Mahalanobis distance to compare the two values:

$$(L_i - L_j)^T (P_{L_i} + P_{L_j})^{-1} (L_i - L_j) < \chi^2_{1,\alpha}. \tag{35}$$

The second geometrical constraint is named *binary constraint*. Suppose we have two pair of segments $p_{ij} = (E_i, F_j)$ and $p_{kl} = (E_k, F_l)$ with the firsts pertaining to the E submap $((E_i, E_k))$ and the seconds to the F submap $((F_j, F_l))$. A binary constraint is a geometric relation between the measures estimate of the features (E_i, E_k) that it is satisfied also by the corresponding features (F_j, F_l); in this case we consider the angle between them

$$A_{jl} = \cos^{-1}\left(\frac{a_1a_2 + b_1b_2 + c_1c_2}{\sqrt{a_1^2 + b_1^2 + c_1^2}\sqrt{a_2^2 + b_2^2 + c_2^2}}\right) \tag{36}$$

being $B = B_F, a_1 = (x''^B_{F_j} - x'^B_{F_j})$, $b_1 = (y''^B_{F_j} - y'^B_{F_j})$, $c_1 = (z''^B_{F_j} - z'^B_{F_j})$, $a_2 = (x''^B_{F_l} - x'^B_{F_l})$, $b_2 = (y''^B_{F_l} - y'^B_{F_l})$, $c_2 = (z''^B_{F_l} - z'^B_{F_l})$, and its covariance

$$P_{A_{jl}} = \mathbf{F}_j\mathbf{P}^{B_F}_{F_j}\mathbf{F}_j^T + \mathbf{G}_l\mathbf{P}^{B_F}_{F_l}\mathbf{G}_l^T + \mathbf{F}_j\mathbf{P}^{B_F}_{F_jF_l}\mathbf{F}_j^T + \mathbf{G}_l\mathbf{P}^{B_F}_{F_lF_j}\mathbf{G}_l^T \tag{37}$$

with $\mathbf{F}_j = \frac{\partial A_{jl}}{\partial \mathbf{x}^{B_F}_{F_j}}$ and $\mathbf{G}_l = \frac{\partial A_{jl}}{\partial \mathbf{x}^{B_F}_{F_l}}$. Here the Mahalanobis constraint becomes:

$$(A_{ik} - A_{jl})^T(P_{A_{ik}} + P_{A_{jl}})(A_{ik} - A_{jl}) < \chi^2_{1,\alpha}. \tag{38}$$

Unary and binary tests, executing only independent comparisons between pairs of segments, do not guarantee the global consistency of the generated hypothesis by themselves so we apply the joint compatibility test as well. Suppose we have an hypothesis H_i with m couples of corresponding segments (if $j_i = 0$ then the corresponding segment does not exist)

$$H = \begin{bmatrix} E_1 & F_{j_1} \\ E_2 & F_{j_2} \\ \vdots & \vdots \\ E_m & F_{j_m} \end{bmatrix}, \tag{39}$$

we use as compatibility test the distance between the extrema of one segment (e.g., pertaining to submap E) and the straight line built using the extrema of the other segment (e.g., pertaining to submap F).

Out of this distance, we build a function D composed by i functions, such as for each couple in H_i:

$$\mathbf{D}_{H_i}(\mathbf{x}^{B_E}_E, \mathbf{x}^{B_F}_F) = \begin{pmatrix} \mathbf{D}_{H_{i-1}}(\mathbf{x}^{B_E}_E, \mathbf{x}^{B_F}_F) \\ \mathbf{D}_{ij_i}(\mathbf{x}^{B_E}_{E_i}, \mathbf{x}^{B_F}_{F_{j_i}}) \end{pmatrix} \tag{40}$$

$$= \begin{pmatrix} D(\mathbf{x}'^{B_E}_{E_1}, \mathbf{x}^{B_F}_{F_{j_1}}) \\ D(\mathbf{x}''^{B_E}_{E_1}, \mathbf{x}^{B_F}_{F_{j_1}}) \\ \vdots \\ D(\mathbf{x}'^{B_E}_{E_i}, \mathbf{x}^{B_F}_{F_{j_i}}) \\ D(\mathbf{x}''^{B_E}_{E_i}, \mathbf{x}^{B_F}_{F_{j_i}}) \end{pmatrix} = 0 \tag{41}$$

where $\mathbf{x}^{B_E}_{E_i}$ is the coordinate vector for E_i, $\mathbf{x}^{B_F}_{F_{j_i}}$ is the coordinate vector for F_{j_i}, and D is the point-line distance function as from Equation 7. The joint-compatibility test on hypothesis H_i is thus:

$$D^2_{H_i} = D^T_{H_i}\mathbf{C}^{-1}D_{H_i} < \chi^2_{2i,\alpha} \tag{42}$$

$$\mathbf{C}_{H_i} = \mathbf{H}_{H_i}\mathbf{P}^{B_E}_E\mathbf{H}^T_{H_i} + \mathbf{G}_{H_i}\mathbf{P}^{B_F}_F\mathbf{G}^T_{H_i} \tag{43}$$

where $\mathbf{H}_{H_i} = \frac{\partial D_{H_i}}{\partial \mathbf{X}^{B_E}_E}$ and $\mathbf{G}_{H_i} = \frac{\partial D_{H_i}}{\partial \mathbf{X}^{B_F}_F}$.

Once data-association has been found it is possible to perform *Robot Relocation* and *Local Map Joining*, i.e., to estimate the spatial relationship between the robot pose and its position in the associated submap and thus join the two submaps (in this example E and F) w.r.t. a common reference frame. This is obtained by using the selected hypothesis H_i to estimate the transformation between the two reference frame and its covariancematrix.

In doing this, we use an Extended Kalman Filter. For each pair of observations in the hypothesis, we have

$$\hat{\mathbf{z}}^{B_F}_F = \hat{\mathbf{x}}^{B_F}_F = \begin{bmatrix} \hat{\mathbf{x}}^{B_F}_R & \hat{\mathbf{x}}^{B_F}_{F_1} & \dots & \hat{\mathbf{x}}^{B_F}_{F_n} \end{bmatrix}^T \tag{44}$$

$$\mathbf{R}^{B_F}_F = \mathbf{P}^{B_F}_F = diag(\mathbf{P}^{B_F}_R, \mathbf{P}^{B_F}_{F_1}, \dots, \mathbf{P}^{B_F}_{F_n}) \tag{45}$$

and

$$\hat{\mathbf{z}}^{B_E}_E = \hat{\mathbf{x}}^{B_E}_E = \begin{bmatrix} \hat{\mathbf{x}}^{B_E}_R & \hat{\mathbf{x}}^{B_E}_{E_1} & \dots & \hat{\mathbf{x}}^{B_E}_{E_n} \end{bmatrix}^T \tag{46}$$

$$\mathbf{R}^{B_E}_E = \mathbf{P}^{B_E}_E = diag(\mathbf{P}^{B_E}_R, \mathbf{P}^{B_E}_{E_1}, \dots, \mathbf{P}^{B_E}_{E_n}). \tag{47}$$

describing in the filter state the rototraslation to estimate

$$\hat{\mathbf{x}}_{\mathbf{B_F}}^{\mathbf{B_E}} = [\hat{\phi}, \hat{\gamma}, \hat{\theta}, \hat{x}, \hat{y}, \hat{z}]^T \tag{48}$$

we can use the geometrical relationships between the state and the observations described by equations:

$$h_i(\mathbf{z}_{E_i}^{B_E}, \mathbf{z}_{F_{j_i}}^{B_F}, \mathbf{x}_{B_F}^{B_E}) = \left[\begin{array}{c} D(\mathbf{z}'^{B_E}_{E_i}, \mathbf{x}_{B_F}^{B_E} \oplus \mathbf{z}_{F_{j_i}}^{B_F}) \\ D(\mathbf{z}''^{B_E}_{E_i}, \mathbf{x}_{B_F}^{B_E} \oplus \mathbf{z}_{F_{j_i}}^{B_F}) \end{array} \right] = \left[\begin{array}{c} 0 \\ 0 \end{array} \right] \tag{49}$$

Since the observations are uncertain, the equation will not provide null value, however the filter state will converge to their least square solution.

Using the so estimated transformation, it is possible to change the base reference of one of the map into the other, by using two pairs of features, therefore creating a single submap and moving the robot pose to the new reference frame.

The change base reference process begins with the selection of two couples of segments from the set of all corresponding features in the two submaps (hypothesis H_i).

Then we calculate the rototraslation between the base reference of one submap (e.g., F submap) and the features selected previously. Once we obtain the rototraslation parameters, we can proceed to apply this transformation to all submap features. he first step is to compose the rototraslation with the robot pose:

$$\hat{\mathbf{x}}_R^{F_i} = \hat{\mathbf{x}}_B^{F_i} \oplus \hat{\mathbf{x}}_R^{B} \tag{50}$$

$$\mathbf{P}_R^{F_i} = \mathbf{F}\mathbf{P}_B^{F_i}\mathbf{F}^T + \mathbf{G}\mathbf{P}_R^{B}\mathbf{G}^T \tag{51}$$

where $\hat{\mathbf{x}}_B^{F_i} = -\hat{\mathbf{x}}_{F_i}^{B}$, $\mathbf{F}$ is the Jacobian of $\hat{\mathbf{x}}_B^{F_i} \oplus \hat{\mathbf{x}}_R^{B}$ w.r.t. $\hat{\mathbf{x}}_B^{F_i}$ and $\mathbf{G}$ is the Jacobian of $\hat{\mathbf{x}}_B^{F_i} \oplus \hat{\mathbf{x}}_R^{B}$ w.r.t. $\hat{\mathbf{x}}_R^{B}$. The same process is applied to all submap features.

$$\hat{\mathbf{x}}_k = \left[\begin{array}{cccc} \hat{\mathbf{x}}_B^{F_i} \oplus \hat{\mathbf{x}}_R^{B} & \hat{\mathbf{x}}_B^{F_i} \oplus \hat{\mathbf{x}}_{F_1}^{B} & \dots & \hat{\mathbf{x}}_B^{F_i} \oplus \hat{\mathbf{x}}_{F_n}^{B} \end{array} \right]^T \tag{52}$$

$$\mathbf{P}_F^{F_i} = \mathbf{F}\mathbf{P}_B^{F_i}\mathbf{F}^T + \mathbf{G}\mathbf{P}_F^{B}\mathbf{G}^T \tag{53}$$

where $\mathbf{F}$ is the Jacobian of $\hat{\mathbf{x}}_B^{F_i} \oplus \hat{\mathbf{x}}_F^{B}$ w.r.t. $\hat{\mathbf{x}}_B^{F_i}$ and $\mathbf{G}$ is the Jacobian of $\hat{\mathbf{x}}_B^{F_i} \oplus \hat{\mathbf{x}}_F^{B}$ w.r.t. $\hat{\mathbf{x}}_F^{B}$.

As a result, all submap features will have their coordinates described w.r.t. to the base reference placed on the chosen feature pair; thus we can add all the features $\mathbf{x}_F^{B_F}$ to the submap E. In this way we create a single submap with the features pertaining to both submaps. The computation of the rototraslation is similar to the previous algorithm (the one used to obtain the rototraslation between features and base reference).

We can now proceed to the fusion phase. This operation is necessary because we have features that are different estimates of the same world segment. The submap fusion is analogous to the view segment integration process shown in before; however, in this process we use a different state equation of the Extended Kalman Filter. In fact, in this case we do not have a set of views to integrate, but we have only 3D segments with the same base reference (we have not odometric data).

When the process ends, the corresponding features will be totally correlated with the same mean and covariance. Thereby one of the two estimates can be removed from the state. At the end we execute a final integration process to recalculate the 3D extrema of the updated segments.

In the very last phase of loop closure we can reduce the uncertainty on the spatial relationships among features, by forcing the consistency of the links between the base references involved in the loop (topological and metrical closure). Consider the case of loop closure in Figure VI where we highlight the two submaps B_0 and B_{n+1}. Suppose we have to close this single loop constituted by the following transformations: $\mathbf{x}_1^0 \oplus \mathbf{x}_2^1 \oplus \cdots \oplus \mathbf{x}_n^{n-1} \oplus \mathbf{x}_{n+1}^n$ where $\mathbf{x}_1^0$ has enough overlapping with $\mathbf{x}_{n+1}^n$.

We fuse at the local level $\mathbf{x}_{n+1}^n$ with $\mathbf{x}_1^0$ as shown so far. Thus we obtain a loop sets up by the transformations: $\mathbf{x}_1^0 \oplus \mathbf{x}_2^1 \oplus \cdots \oplus \mathbf{x}_n^{n-1} \oplus \mathbf{x}_0^n$. Now we can impose that:

$$h(\mathbf{x}) = \mathbf{x}_1^0 \oplus \mathbf{x}_2^1 \oplus \cdots \oplus \mathbf{x}_n^{n-1} \oplus \mathbf{x}_0^n = 0 \tag{54}$$

since submap $\mathbf{x}_0^n$ is in spatial relation with the submap $\mathbf{x}_1^0$. When we have large loops and composed by various rototraslations, the errors, due to the linearization processes, become pronounced. The techniques introduced during the fusion phase at the local level don't allow to compute, with enough accuracy, the transformations $\mathbf{x}_j^i$. For this reason, we formulate the problem as the estimate of the maximum posterior probability on the relative positions at the global level given the $h(\mathbf{x}) = \mathbf{0}$ loop closure constraint

$$\min_x f(\mathbf{x}) = \min_x \frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^T \mathbf{P}^{-1} (\mathbf{x} - \hat{\mathbf{x}}) \tag{55}$$

To solve this type of problems we use the SQP (Sequential Quadratic Programming) technique [16] derived from the Kuhn-Tucker equations. The SQP estimate of the transformations $\mathbf{x}$ and their covariance $\mathbf{P}$ can be obtained by the iteration of the following two equations:

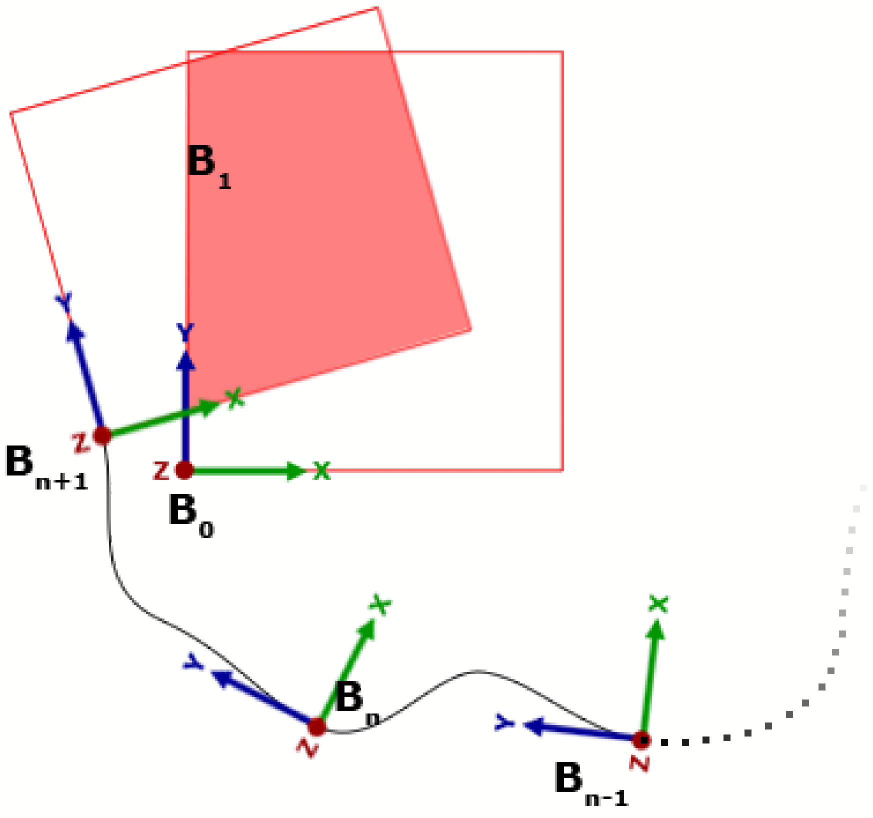


Fig. 5. Loop Closure

$$\mathbf{P}_i = \mathbf{P}_0 - \mathbf{P}_0\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_0\mathbf{H}_i^T)^{-1}\mathbf{H}_i\mathbf{P}_0 \tag{56}$$

$$\hat{\mathbf{x}}_{i+1} = \hat{\mathbf{x}}_i - \mathbf{P}_i\mathbf{P}_0^{-1}(\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_0) - \mathbf{P}_0\mathbf{H}_i^T(\mathbf{H}_i\mathbf{P}_0\mathbf{H}_i^T)^{-1}\hat{h}_i \tag{57}$$

where: $\hat{\mathbf{x}}_0$ is the best estimate so far, $\hat{h}_i = h(\hat{\mathbf{x}}_i)$, $\mathbf{H}_i$ is the Jacobian of $h(\mathbf{x})$ evaluated in $\hat{\mathbf{x}}_i$. The same process can be used to close multiple loops at the same time, increasing the number of constraints in the minimization problem.

VII. Experimental Activity

For the experimental activity we used a Robuter mobile robot from Robosoft which computes odometry as a 3DoF pose; this datum reaches a PC via serial line. On the PC we have an Eltec frame grabber capable to grab three 704x558x8 pixels images at the same time. Each channel of the frame grabber is connected to a Sony XC75CE camera. Cameras have been calibrated with a standard DLT approach like in [17]. The robot has been moved inside the 4th floor of building U7, Univ. Milano - Bicocca, Milano, Italy. Distances between consecutive robot poses, i.e. views, was about 0.05m. The overall distance travelled has been about 200m.

In Figure 6a the odometric travel is shown, altogether with the environment planimetry. The odometric error is modelled as zero mean Gaussian, and the propagation of this error is shown in Figure 6b with the usual 99% ellipses; notice that the actual, i.e. first, poses are in good agreement with the uncertainty propagated up to the last ones; this confirms the correctness of the probabilistic model of the odometric error. Submap termination is currently set on the cardinality of the features in the submap; the value used in the reported experiment is 50. In Figure 6c the odometric travel is superimposed to the base references of the submaps, i.e. what could be considered as the overall result of the integration of views processing. This figure shows that the integration of views gives a large increase in accuracy, even though this is not enough for obtaining a geometrically consistent map. When a submap is closed loop-detection is activated; in Figure 6d the bounding boxes of the first and last submaps, i.e. the ones for which a loop is detected, are shown. On these two submaps Robot Relocation and Local Map Joining are applied. At the end the two submaps are fused together in a single submap. The geometric consistency is still not attained at this stage. For this reason we apply Loop Closure, in order to distribute the errors along the whole set of submaps, i.e. relative poses of submaps. The result of such iterative non-linear optimization (graph relaxation) is shown, in terms of base references, in Figure 6e.

In order to show the effect of an appropriate modelling of the reality, in particular about the number of pose DoF, we used the same data-set to run the system with a 3DoF pose. Figure 7a presents a bird-eye view of the reconstruction, before Loop

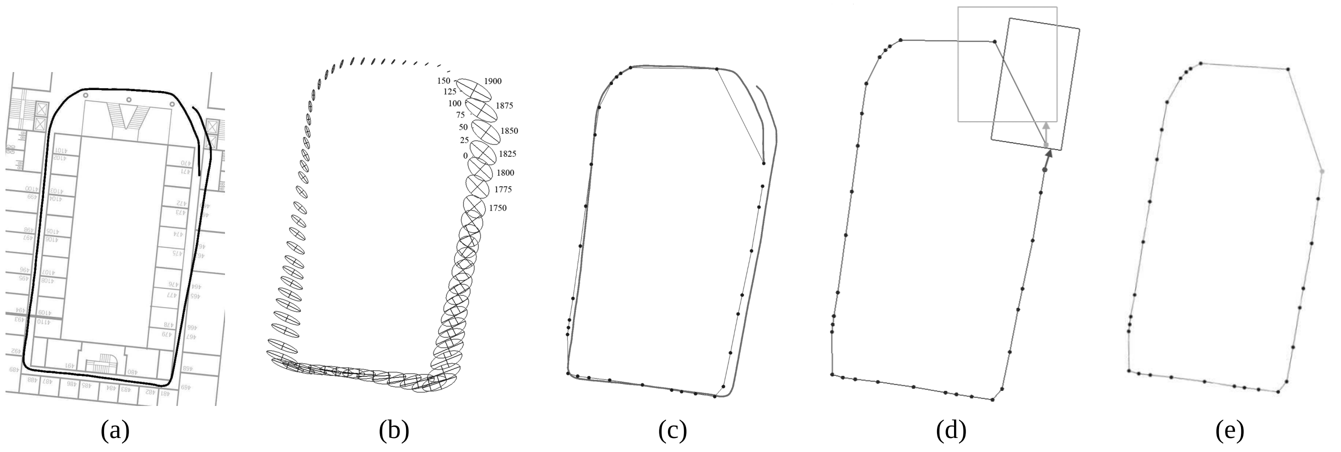


Fig. 6. (a): Odometric travel superimposed to the environment planimetry. (b): Odometric error ellipses ($\pm 3\sigma$), with view_id. (c): Odometric travel (full) superimposed to the base references of the submaps (circles connected by lines). (d): Bounding boxes of last (darker) and first (lighter) submaps. (e): The base references of the submaps after graph relaxation, compare with the base references in (c) or (d).

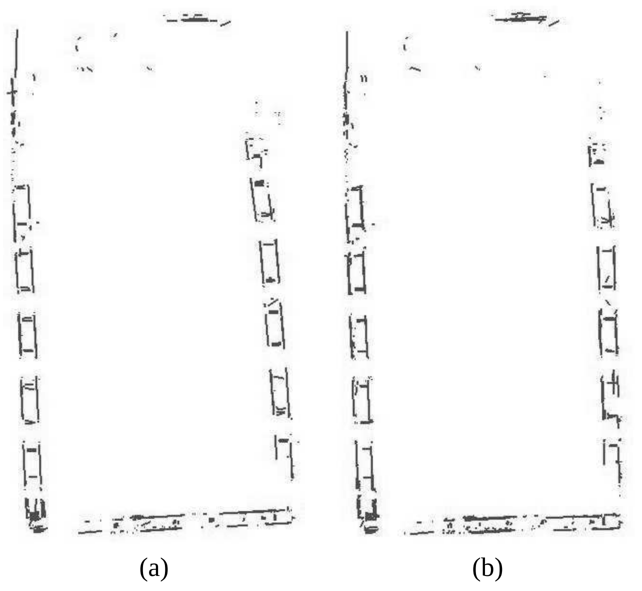


Fig. 7. Reconstructions before graph relaxation; (a): 3DoF-pose. (b): 6DoF-pose.

Closure, obtained with such setting. On the other hand, the reconstruction, before Loop Closure, obtained with a 6DoF pose is shown in Figure 7b. Another effect, beside the higher accuracy, which is not perceivable in the pictures, is related to the view-to-submap data associations. In the 6DoF-pose reconstruction there are about 15% less missed matches than in the 3DoF reconstruction. Notice that, even though the overall geometric consistency is obtained in both cases, the final world model is affected by these errors, i.e. it presents these isolated features. Therefore the higher accuracy of the 6DoF-pose presents also this relevant effect at the global level, after Loop Closure. A larger accuracy could improve performance in SLAM systems other than the Hierarchical one.

The 6DoF-pose final reconstruction, projected on the floor, is shown in Figure 8, superimposed to the environment planimetry for checking the absolute accuracy; a 3D view of it is presented in Figure 9.

REFERENCES

[1] M. Yachida, “3-d data acquisition by multiple views,” in *Robotics Research: Third Int. Symp.*, O. D. Faugeras and G. Giralt, Eds., 1986, pp. 11–18.

[2] M. Yachida, Y. Kitamura, and M. Kimachi, “Trinocular vision: New approach for correspondence problem,” in *Proc. IEEE ICPR*, Oct 1986, pp. 1041–1044.

[3] N. Ayache, *Artificial Vision for Mobile Robots*. The MIT Press, 1991.

[4] O. D. Faugeras, *Three Dimensional Computer Vision - A geometric viewpoint*. The MIT Press, 1993.

[5] N. Ayache and F. Lustman, “Trinocular stereo vision for robotics,” *IEEE Trans. on PAMI*, vol. 12, no. 1, 1991.

[6] N. J. Ayache and O. D. Faugeras, “Maintaining representations of the environment of a mobile robot,” *IEEE Trans. on R&A*, vol. 5, no. 6, pp. 804–819, 1989.

[7] P. Kahn, L. Kitchen, and E. M. Riseman, “A fast line finder for vision-guided robot navigation,” *IEEE Trans. on PAMI*, vol. 12, no. 11, Nov 1990.

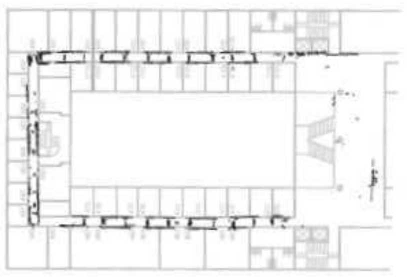


Fig. 8. Bird-eye view of the 6DoF-pose final reconstruction, after graph relaxation, superimposed to planimetry.

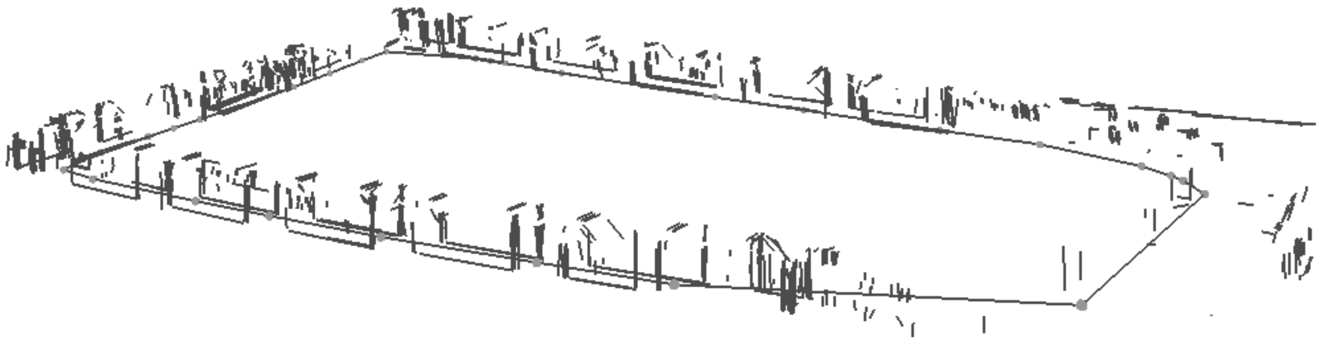


Fig. 9. A 3D view of the 6DoF-pose final reconstruction; the solid-circle line is the same as in Figure 6e.

[8] H. Surmann, A. Nüchter, K. Lingemann, and J. Hertzberg, “6d slam: Preliminary report on closing the loop in six dimensions,” in *5th IFAC Symp. IAV*, June 2004.
[9] A. Davison, “Real-Time Simultaneous Localisation and Mapping with a Single Camera,” in *IEEE International Conference on Computer Vision*, October 2003, pp. 1403–1410.
[10] C. Estrada, J. Neira, and J. D. Tardós, “Hierarchical slam: real-time accurate mapping of large environments,” *IEEE Trans. on Robotics*, to appear.
[11] D. Marzorati, D. G. Sorrenti, and M. Matteucci, “Multi-criteria data association in 3d-6dof hierarchical slam with 3d segments,” in *Proc. of ASER06 Conf.*, 2006.
[12] J. Neira and J. D. Tardós, “Data association in stochastic mapping using the joint compatibility test,” *IEEE Trans. R&A*, vol. 17, no. 6, pp. 890–897, 2001.
[13] L. Matthies and S. Shafer, “Error modelling for stero navigation,” vol. 3, pp. 239 – 250, Jun. 1987.
[14] E. Trucco and A. Verri, *Introductory Techniques for 3D Computer Vision*. Prentice-Hall, 1998.
[15] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Comm. ACM*, vol. 24, no. 6, pp. 381–395, 1981.
[16] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, “Bundle adjustment – a modern synthesis,” in *Vision Algorithms: Theory and Practice*, ser. Lecture Notes in Computer Science, B. Triggs, A. Zisserman, and R. Szeliski, Eds., vol. 1883. Springer-Verlag, 2000, pp. 298–372.
[17] O. Faugeras and G. Toscani, “The calibration problem for stereo,” in *Proceedings of IEEE CVPR*, 1986.