César Darío Cadena Lerma

Efficient simultaneous localisation and mapping in large and complex environments

Departamento

Informática e Ingeniería de Sistemas

Director/es

Neira Parra, José

ISSN 2254-7606

Tesis Doctoral

EFFICIENT SIMULTANEOUS LOCALISATION AND MAPPING IN LARGE AND COMPLEX ENVIRONMENTS

Autor

César Darío Cadena Lerma

Director/es

Neira Parra, José

UNIVERSIDAD DE ZARAGOZA
Informática e Ingeniería de Sistemas

2011

THESIS

Ph.D. in Computer Science and Systems Engineering

Efficient Simultaneous Localisation And Mapping in Large and Complex Environments

Cesar Dario Cadena Lerma

Advisor: José Neira Parra

Computer Science and Systems Engineering Department

Centro Politécnico Superior

Universidad de Zaragoza

September 2011

Abstract

Finding its way in the environment in which a robot operates is a basic problem to solve for true autonomy. There are two main aspects to this problem, known as Simultaneous Localization and Mapping (SLAM): (1) the continuous problem of estimating the location of elements of interest for the robot, and (2) the discrete problem of finding correspondences between measurements of the sensor that the robot uses to perceive its environment and the elements already in the map. This problem is usually denominated the data association problem. In this thesis we address these two aspects of SLAM.

The estimation problem is classically solved with a filtering approach with satisfactory solutions for environments of limited size. But as the operating environments grow, basic filtering approaches become no more an option because of high computational cost and loss of precision. There has been great progress in advancing filtering algorithms in this sense. In this thesis we intend to push filtering algorithms further forward. We propose a highly scalable filtering algorithm, also conceptually simple to implement. The Combined Filter SLAM algorithm (CF SLAM) is a judicious combination of Extended Kalman (EKF) and Extended Information Filters (EIF) that can be used to execute highly efficient SLAM in large environments.

CF SLAM is always more efficient than any other EKF or EIF algorithm: filter updates can be executed in as low as $\mathcal{O}(\log n)$ as compared with the linear cost of the currently most efficient filtering algorithms. In the worst cases, updates are executed in linear time for CF SLAM as compared with the quadratic cost for all others. In addition to its computational cost savings, CF SLAM does not include approximations (apart from linearisations) and require only linear memory with respect to the size of the environment.

The estimation process in SLAM is subordinated to the constraints imposed by the detected correspondences between sensor observations, and thus the algorithm that solves the correspondence must be robust. In this thesis, we address a central data association problem: detecting that the robot is revisiting a previously visited place. This problem is known as loop closing or place recognition.

We propose a place recognition algorithm for SLAM systems that works with image and 3D sensors. Our algorithm considers both appearance and geometric information of points of interest in the image and 3D data. Both near and far scene points provide information for the recognition process. Hypotheses about loop closings are generated using a fast appearance technique based on the bag-of-words (BoW) method. Our main contribution is in the verification stage, using conditional random fields (CRFs). We build on CRF-matching with two main novelties: we use both image and 3D geometric information, and we carry out inference on a Minimum Spanning Tree (MST), instead of a densely connected graph. Our results show that MSTs provide an adequate representation of the problem, with the additional advantages that exact inference is possible and the computational cost of the inference process is limited.

We evaluate our proposals in a several publicly available datasets of indoor and outdoor static and dynamic urban environments, using stereo cameras and 3D laser sensors. We compare with other methods in the state of the art to illustrate the improvements in computational cost and robustness of the methods proposed.

Resumen

El conocimiento de autolocalización en el entorno en que un robot opera es un problema básico a solucionar para una verdadera autonomía. Hay dos aspectos principales de este problema, conocido como Simultaneous Localisation And Mapping (SLAM): (1) el problema de estimación continua de la ubicación de los elementos de interés para el robot, y (2) el problema discreto de encontrar correspondencias entre medidas del sensor que el robot utiliza para percibir el entorno y los elementos ya incluidos en el mapa. Este problema generalmente se denomina el problema de asociación de datos. En esta tesis nosotros abordamos estos dos aspectos del SLAM.

Clasicamente el problema de estimación ha sido resuelto con un enfoque de filtrado con soluciones satisfactorias para entornos de tamaño limitado. Pero, a medida que los entornos de operación crecen en escala los enfoques básicos de filtrado no son más una opción debido a su alto costo computacional y pérdida de precisión. Ha habido un gran progreso en mejorar los algoritmos de filtrado, en este sentido. En esta tesis, nuestra inteción es avanzar aún más allá en los enfoque de filtrado. Proponemos un algoritmo de filtrado altamente escalable y conceptualmente simple de implementar. El Combine Filter SLAM (CF SLAM) es una juiciosa combinación de los filtros extendidos de Kalman (EKF) y de Informacion (EIF) que puede ser usado para ejecutar SLAM muy eficientemente en entornos grandes.

CF SLAM es siempre mas eficiente que cualquier otro algoritmo basado en EKF o EIF: las actualizaciones del filtro pueden ser realizadas en un orden tan bajo como $\mathcal{O}(\log n)$ en comparación con el costo lineal de los algoritmos mas eficientes actuales. En el peor de los casos, CF SLAM ejecuta las actualizaciones en tiempo lineal, en comparación con el costo cuadrático de los demás. Adicional a sus ventajas en eficiencia, CF SLAM no incluye aproximaciones (aparte de linealizaciones) y sólo requiere memoria lineal con el tamaño del ambiente modelado.

El proceso de estimación esta supeditado a las restricciones que imponen las corresponcencias entre observaciones del sensor, por lo cual el algoritmo que resuelve las correspondencias debe ser tan robusto como sea posible. En esta tesis hemos abordado un problema central en la asociación de datos: la detección de que el robot está revisitando un lugar, conocido como cerrado de bucles o reconocimiendo de lugares.

Proponemos un algoritmo de reconocimiento de lugares para sistemas de SLAM que funciona con sensores de imagen y 3D. El algoritmo considera ambas, la información de apariencia y la geométrica, de los puntos de interés de la imagen, así como los puntos tanto cercanos como lejanos al sensor. Nos apoyamos en un método basado en bag of words (BoW) para una eficiente generación de candidatos y hacemos nuestro principal aporte en la etapa de verificación de éstos con un método basado en los conditional random fields (CRFs). Nuestro CRF-matching tiene dos novedades principales, el uso de información de imagen y de geometría 3D, y la definición de la estructura del grafo con un minimum spanning tree (MST), lo que nos permite llevar a cabo una inferencia eficiente y exacta.

Evaluamos nuestras propuestas en un gran número de datasets de acceso público de entornos urbanos de interiores y exteriores, y estáticos y dinámicos, usando cámaras estéreo y sensores de 3D con láser. Mostramos comparaciones cuantitativas con otros métodos en el estado del arte e ilustramos las ventajas de cada una de nuestras contribuciones.

Contents

Chapter 1

Introduction

1.1 Motivation

What does the world look like? Where am I? The answer to these two fundamental questions is the first step in building an autonomous robot able to navigate safely and effectively in an unknown environment. Answering these questions at the same time is known as simultaneous localisation and mapping (SLAM): a robot estimates its position with respect to an incrementally mapped environment.

Mapping the environment, the answer to the first question, has a direct and critical influence on basic robotic tasks, such as collision avoidance, path planing, and in non-autonomous applications as teleoperation in unknown and risky zones. The real world is complex, irregular, and dynamic. Furthermore, available sensors commonly provide noisy measurements, and robots must operate robustly in this complex world.

To navigate reliably, mobile robots must know where they are in the environment, which is the answer to the second question. For a robot to decide what to do next it must know where it is located. In order to localise itself, a robot has access to relative and absolute measurements about its movements and the environment surrounding the robot. Given this information, the robot has to determine its location as accurately as possible.

Over the past two decades mobile robotics research in localisation and map building problem has been very active. However, nowadays most of the robots operating in controlled and limited environments are expensive, and there are only a few examples of low-cost robots with very limited assignments in home, like the Roomba vacuum cleaner. This is an evidence of the shortcomings in the state of the art.

Of course, the quality of the mapping and pose estimation depends on the quality

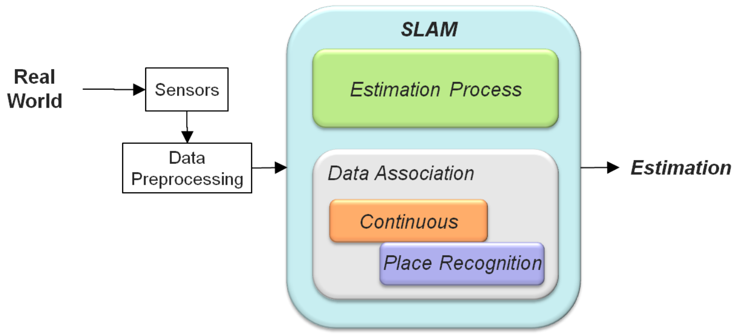


Figure 1.1: A general view of a SLAM system with the sub-blocks often used.

of the sensors, but as the robot operates in larger and more complicated environments, the limiting factor becomes *algorithmic*: the amount of computational effort required to process sensor observations grows quickly with the number of observations. Conventional stochastic solutions to SLAM involve the computation of covariance matrices which are in general of complexity $\mathcal{O}(n^2)$ in the number of features (or landmarks) in the map.

Much effort has been made to reduce the complexity of the *estimation process* (or back-end) in SLAM algorithms. However, that is not enough for solving the problem, it is mandatory to also solve *data association*, the front-end. Association of map features becomes difficult as errors in position estimations increase. Reliable detection and association of landmarks plays a major role in autonomous localisation and mapping.

The challenges increase with cheap sensors due to their limitations, large noise and ambiguity. A lot of current outdoor robotics is based on sensors that provide geometric (range and bearing) information. Although these sensors are accurate, they only provide geometric profiles of objects which are, in general, insufficient for recognition. Imaging sensors provide richer information such as texture, and, in most cases, are less expensive than ranging sensors. Appearance information has the difficulty to interpret it, but it also gives a complementary point of view to the purely geometric information. In addition to the latter, these kind of sensors have unlimited range, which has the inconvenience of increasing the difficulty of managing the information of far-field objects, but also opens the opportunity for using such information.

A major problem when performing localisation and mapping in large outdoor environments is reliable data association. As the uncertainty over the position of landmarks and vehicle grows, correct association can be very difficult. If an incorrect data association hypothesis is accepted and introduced in the estimation process, the estimation may become inconsistent compromising the whole map.

The detection of loop closure in SLAM is also a significant issue when a robot follows an extensive path. Observed landmarks are initialised in the map and have to be recognised and correctly associated when the robot re-observes them in a trajectory with loops. When navigating without closing the loop, the uncertainty of the robot grows unbounded, making further associations more difficult. Furthermore, the number of landmarks used in large-scale SLAM can be significant, increasing the computational complexity of the problem.

In this thesis we are concerned to advance the state of the art by making SLAM systems *more efficient* (so they can handle larger environments) and *more robust* to noise and ambiguity.

1.2 Estimation

Since the mid 80s, SLAM has been widely addressed using different approaches, see Durrant-Whyte and Bailey (2006) for a comprehensive review. Researches have devoted much effort to develop algorithms to improve the computational efficiency of SLAM in large scale environments (Bailey and Durrant-Whyte 2006). We find work in Extended Kalman filters (EKF) and Extended Information filters (EIF) (Dissanayake et al. 2001; Thrun et al. 2005), particle filters (Burgard et al. 1996; Thrun 2002), nonlinear optimization (Grisetti et al. 2007a; Olson et al. 2006), sparse linear algebra, e.g., Dellaert and Kaess (2006), to name a few.

The earliest formulation for SLAM was the EKF of Smith et al. (1988). Although it is very popular, it is known by its limiting computational properties (Thrun et al. 2005). Given a map of n features, the classical EKF SLAM algorithm is known to have a cost of $\mathcal{O}(n^2)$ per update step.

One important contribution has been the idea of splitting the full map into local maps and then putting the pieces back together in some way. Decoupled Stochastic Mapping (Leonard and Feder 2001), Constant Time SLAM (Leonard and Newman 2003) and the ATLAS system of Bosse et al. (2004) are local mapping solutions close to constant time, although through approximations that reduce precision. Map Joining of Tardós et al. (2002) and the Constrained Local Submap Filter of Williams et al. (2002) are exact solutions (up to linearisation) that require periodical $\mathcal{O}(n^2)$ updates. Map Joining builds the map with a sequence of independent local submaps with EKF. Local EKF submaps lead to a more consistent map in addition to computational saving. The Constrained

Local Submap Filter is a similar technique to Map Joining.

Belonging to the exact solutions with submapping, two recent algorithms have provided important reductions in computational cost: D&C SLAM of Paz et al. (2008) has an amortised cost $\mathcal{O}(n)$ per step, and Sparse Local Submap Joining (SLSJF) SLAM of Huang et al. (2008a) reports a cost $\mathcal{O}(n^{1.5})$ per step in the worst cases. D&C SLAM introduces an improved method of managing the submaps which postpones expensive joining operations of large maps. And SLSJF takes advantage of the sparsity of the information form to reduce the computational complexity.

Recently proposed, the CI-Graph (Pinies et al. 2009) is an algorithm in which the local maps are conditionally independent. CI-Graph keeps the submaps separate, but additionally, takes into account that information may shared between submaps, breaking their independence. A spanning tree topology is used to handle environments with many loops. The price for keeping the conditional independence between submaps is some overhead in the size of the submaps: all extra components added in different submaps in order to keep the conditional independence and to propagate the information in updates. However, CI-Graph reduces memory requirements when exploring an environment as it does not need to maintain all covariance matrix entries. The authors empirically obtained a cost per step close to linear in the worst case.

Exact solutions also include incremental Smoothing and Mapping (iSAM) of Kaess et al. (2008) and Tectonic SAM of Ni et al. (2007), and their more recent and efficient versions: iSAM2 (Kaess et al. 2011) and TSAM2 (Ni and Dellaert 2010), respectively. These solutions keep all the trajectory and features in the state vector and use the square root of the information matrix to efficiently solve the SLAM problem. Using also the information form, Frese (2006) proposed the Treemap, which captures the sparse structure of the system by a tree representation. Each leaf in the tree is a local map and the consistency of the estimation is achieved by sending updates to the local maps through the tree. This is a very efficient algorithm but it is not exact: where there are many overlapping features, it uses a weak link breakage policy in order to prune edges and hold the tree structure. Under topological restrictions on the environment, Treemap has a cost $\mathcal{O}(\log n)$ per step. However, if the map has many local connections the size of the local maps can be very large and their updates become computationally expensive as shown by Konolige et al. (2010b).

The SLAM problem has also been addressed using particle filters (Thrun 2002). The particle filter keeps a number k of possible locations of the robot, and computes an alternative map for each. The most efficient SLAM algorithms based on particle filters factorise the problem so that only the vehicle location is represented with a set of particles, and have a cost $\mathcal{O}(nk)$ per step (Eliazar and Parr 2004; Grisetti et al. 2007b; Montemerlo et al. 2003; Törnqvist et al. 2009). In FastSLAM (Montemerlo et al. 2002) each particle represents a distinct hypothesis of the vehicle trajectory and has its own EKF associated with it to estimate the map. The particle are dispersed according to the motion model, to create a proposal distribution. The particles are then weighted and re-sampled according to the measurements model. Montemerlo et al. (2003) refine the algorithm, to FastSLAM 2.0, by modifying the proposal distribution to improve the richness of particles in the most likely regions of the posterior distribution. The DP-SLAM of Eliazar and Parr (2004) is an algorithm analogous to FastSLAM, where evidence grids are used rather than EKFs to represent map information. Particle filters are robust to data association errors. Due to the large number of particles, careful memory management is needed for efficient implementation. The consistency and robustness of these algorithms depends on the number k of particles in the filter. In order not to underestimate the uncertainty, this number must grow with the size of the environment that is expected to map.

When the application requires mainly to have a very accurate localisation of the robot, and a detailed map is of secondary importance, the use of Pose based SLAM (Eustice et al. 2006; Grisetti et al. 2008; Olson et al. 2006) become interesting. Eustice et al. (2006) proposed a exactly sparse information filter keeping all the pose history and marginalising the landmarks. Olson et al. (2006) suggested a gradient descent approach to optimise pose graphs. And Grisetti et al. (2009) extended this approach by applying a tree-based parametrisation that increases the convergence speed. Both approaches assume that the error is uniformly distributed overall pose-graph. Later, Konolige et al. (2010b) present the Sparse Pose Adjustment (SPA), an optimised implementation of the underlying bundle adjustment that solves very large 2D SLAM problems with several thousand poses in short time. It takes the covariance in the constraints into account which leads to more accurate solutions with very fast convergence and fully non-linear.

In this sequences of improvements for pose-graphs the g2o was proposed by Kümmerle et al. (2011), a general framework for performing optimisation of non-linear least squares problems that can be represented as a graph. This toolkit exploits the sparse connectivity of the graph and advanced sparse linear solvers to be more efficient. Another pose-based SLAM solutions is proposed by Ila et al. (2010), the Information-based Pose SLAM. It keeps the efficiency including only some informative poses from the odometry information and only few loops. That idea trades off precision for efficiency in the estimation. Although the authors claim a amortised constant cost updating loop closures until the next will come, it is not possible to know by advance when the next informative loop closure will take place.

All the pose based methods do not compute an optimal structure estimation and instead focus on computing the optimal vehicle trajectory. They keep a state vector containing all or some the poses of the robot, the map must be computed a posteriori.

In the vision community the SLAM problem is known as Structure From Motion (SFM), and it has been classically solved with bundle adjustment (Triggs et al. 2000). In mobile robotics have been applied with success a lot of ideas from computer vision, moreover for solving efficiently the estimation problem. Some solution are the proposals by Klein and Murray (2007); Konolige and Agrawal (2008); Mei et al. (2011); Sibley et al. (2009); Strasdat et al. (2010). Klein and Murray (2007) split tracking and map building into parallel threads in their PTAM system, with different timing requirements. While tracking has to be performed in real-time, the map is extended only by select keyframes and can be more time consuming. Their approach has set a keypoint with respect to accuracy and speed, although it is still restricted to operation in small desktop environments. Konolige and Agrawal (2008) proposed the FrameSLAM reducing the number of camera poses and 3D points. They showed that even with reduced data the same accuracy could be reached after bundle adjustment. FrameSLAM reaches efficiency keeping only some keyframes in a skeleton and performing a local optimisation in exploration mode and over the skeleton in loop closures. Later, Konolige et al. (2010a) extent this work to the View-based Maps including place recognition and weak links for multisession mapping. Sibley et al. (2009) introduced a new representation of map structure and camera poses. Map points and camera poses are described by their relative positions. They have developed a local-consistency optimiser, a adaptive relative bundle adjustment. They show constant-time updates, even in the presence of loop closures, to create a locally consistent system. The same idea is exploit by Mei et al. (2011) with their RSLAM system. RSLAM is again a locally consistent optimiser rather than a exact solution to the SLAM problem. Very recently, Strasdat et al. (2010) use a bundle adjustment in sliding windows in exploration mode for a monocular camera. And, in loop closure event they optimise the relative similarity transform between all the camera poses taking also the scale drift of monocular tracking into account. Their results in terms of accuracy are comparable to standard bundle adjustment.

Other interesting approaches to the estimation problem are the topological maps (Cummins and Newman 2008; Ranganathan and Dellaert 2011). The topological maps solve the problem of in which place in the environment the robot is, and how those places are connected. Cummins and Newman (2008) proposed the FAB-MAP, a very efficient visual appearance based system. In FAB-MAP the places of the topology are disjunct images captured by the vehicle. Ranganathan and Dellaert (2011) use a particle filter to infer the topology of the graph of places. That system is independent of the sensor used by the robotic platform.

In this thesis we aim at further reducing the computational complexity, without any sacrifice of precision or accuracy.

1.3 Data Association

Producing a correspondence between the observed data and quantities to be estimated is known as the *data association* problem (Christensen and Hager 2008). It is an essential step for the estimation process, and it is one of the most difficult problems in SLAM (Thrun and Leonard 2008).

In the second part of this thesis we discuss the crucial, and sometimes omitted, problem of data association in SLAM.

This problem is naturally divided in two problems: continuous data association, or feature tracking, and loop closure or place recognition problem.

1.3.1 Continuous Data Association

Data association has been addressed classically within of the EKF paradigm for SLAM as finding the measurement-to-track correspondence (Bar-Shalom and Fortmann 1988) because there is the advantage of available prior state and covariance ($\mu_{t-1}, \mathbf{\Sigma}_{t-1}$). This allows to predict features and compare with the current observations by carrying out statistical tests to find possible matches based on stochastic geometry. When additional information is available, such as texture or appearance in vision sensors, it is possible obtain matchings based not only on location but also on appearance.

The most basic method is the Nearest Neighbour Standard Filter, or simply Nearest Neighbour (NN). The NN rule (Mahalanobis distance) is used to select the best matchings from the compatibility given by the normalised squared innovation test (Bar-Shalom and

Fortmann 1988). Although, it is a very simple and efficient method and widely used, it is not reliable in cluttered environments or with very noisy sensors, because it overlooks the fact that the predicted features are correlated, and therefore, the compatibility should be ensured jointly and not only individually.

The Joint Compatibility Branch and Bound (JCBB) technique of Neira and Tardós (2001) addresses this issue. It matches features via a deterministic interpretation tree (Grimson 1990) and has been successfully used in different SLAM applications (Clemente et al. 2007; Fenwick et al. 2002; Kaess and Dellaert 2009; Tardós et al. 2002; Williams et al. 2007). JCBB traversal considers a joint Gaussian prior on feature positions and calculates the joint probability that any particular hypothesised set of correspondences is correct. The disadvantage of the JCBB technique is its possible exponential cost in the number of measurements. Paz et al. (2008) proposed the Randomised Joint Compatibility (RJC) to overcome this problem. The RJC randomises the jointly compatible space search in the spirit of random sample consensus RANSAC (Fischler and Bolles 1981). The model is the joint compatibility of a small random set of measurements. The remaining measurements are verified to be jointly compatible with the model. The set with greater support is chosen as the final data association solution.

The RJC was developed for data association between local maps in the D&C SLAM algorithm, due to the growing in the size of the joined maps. But the RJC is also applicable to sequential mapping with a high number of observations. Another efficient algorithm for data association in visual SLAM, the 1-point RANSAC, has been proposed recently by Civera et al. (2009). The 1-point RANSAC is a combination of RANSAC and EKF, that uses the prior probabilistic information from EKF update step in the RANSAC model hypothesise stage. In the visual SLAM context the Active Matching algorithm was developed by Chli and Davison (2008), together with its more efficient version Scalable Active Matching (Handa et al. 2010). This technique searches sequentially for correspondences guided by expected information gain as function of the covariance of the innovation. Both, 1-point RANSAC and Active Matching were conceived to raise the frame rate bound in the sequential matching process in an EKF framework, where the covariance matrices are directly available.

Bibby and Reid (2007) proposed a SLAM solution in dynamic environments (SLAMIDE). The main idea is to carry out the data association, using either NN or JCBB, in sliding windows before to incorporate the information to the whole estimation. If a data association is wrong is expected to realise it inside the sliding window, then to reject it before to incorporate that information to the estimation. However, it is not able to incorporate loop closure events.

In terms of local maps, up to now the best way to find the correspondences between submaps is perhaps the RJC. However, for full overlapping or high uncertainty in two large submaps finding the correspondences could have a prohibitive computational cost.

1.3.2 Place Recognition

Detecting when a mobile robot is in a place already visited is fundamental in the SLAM context, to recover from failures and to select policies of exploration in active SLAM. Since cameras are easily available and provide rich scene detail, place recognition using visual information has been a problem of great interest in robotics for some time.

Most successful methods consider appearance or geometric information, or a combination of both. Williams et al. (2009) compared three loop closure methods representative of each idea: a map-to-map method that considers mainly geometry, an image-to-image method that considers only appearance, and an image-to-map method that considers both. The best results were obtained for the image-to-image and image-to-map methods, although the image-to-map method does not scale well in large environments.

The image-to-image method considered in the work of Williams et. al. was FAB-MAP, the first successful appearance-only method, proposed by Cummins and Newman (2008). FAB-MAP uses the bag-of-words (BoW) representation (Sivic and Zisserman 2003), supported by a probabilistic framework. This system proved very successful in large scale environments. It can run with full precision (no false positives), although at the expense of low recall (the rate of true positives declines). Avoiding false positives is crucial because they result in failure to obtain correct maps, but avoiding false negatives is also important because they limit the quality of the resulting maps, particularly if large loops are not detected. Geometric information has shown to be important in avoiding false positives while sacrificing less true positives. Angeli et al. (2008) proposed an incremental version of BoW, using a discrete Bayes filter to estimate the probability of loop-closure. Since the Bayes filter can still exhibit false positives in cases where the same features are detected, but in a different geometric configuration, the epipolar constraint was used to verify candidate matchings. Valgren and Lilienthal (2007, 2010) also verify topological localisation candidates using the epipolar constraint. Matching of an image is carried out against the complete image database, which can become inefficient for large environments.

Other methods have shown the importance of incorporating geometric constraints to avoid false positives. Olson (2009) tests the *spatial* consistency of a set of candidate matchings by additionally considering their associated pose estimates. This requires odometry or some other source for the priors on the poses. Cummins and Newman (2010) incorporated a simplified constraint check for an omni-directional camera installed on a car in FAB-MAP 2.0. This system was tested using two extremely large (70km and 1000km) datasets. They obtain recalls of 48.4% and 3.1%, respectively, at 100% precision. Recently proposed by Paul and Newman (2010), FAB-MAP 3D additionally includes 3D distance between features provided by a laser scanner. This results in higher recall for the same precision in the first urban experiment of FAB-MAP. However, the system can only make inferences about visual features in the laser range.

The place recognition problem has also been addressed using only 3D range data. Steder et al. (2010) extract features from range images obtained by a 3D laser scanner and query a database in order to detect loop closures. This system has high computational requirements compared with systems based on BoWs, but higher recall is attained for the same precision. An important limitation is that this system cannot distinguish between locations with similar shape but different appearance, for example corridors, or with different background beyond sensor's range.

1.4 Contributions

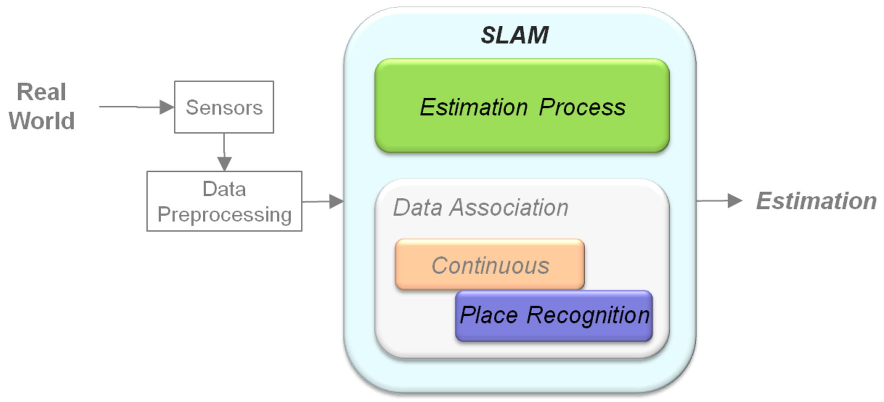


Figure 1.2: Our contributions are focused in the efficiency of the estimation process and in the robustness of the place recognition process.

Our main contributions in this thesis are:

- An **sublinear** algorithm for the state vector estimation, the Combined Filter (CF).

The CF SLAM algorithm is a judicious combination of Extended Kalman and Extended Information Filters, using a divide and conquer local mapping strategy. Chapter 2 describes the CF SLAM details with its properties on complexity and consistency and its performance in real environments.

- A **robust** algorithm for loop closing using stereo cameras that considers both appearance and geometric information of points of interest in the images. Chapter 3 is dedicated to describe the place recognition system evaluating it in real and challenging experiments.

Our two main contributions are assembled in a unified featured-based SLAM system, which is tested in different environments with different sensors in chapter 4. We demonstrate how the whole system successfully operates in larger and more complex environments than previous approaches.

Chapter 2

The Combined Filter SLAM

We devote the first part of the thesis to contextualise our contribution to the estimation problem improving the computational efficiency of SLAM. As we have seen in the introduction of the thesis, there are several approaches to the estimation process. In section 2.1 we focus on featured-based SLAM and we describe the filtering and submapping approaches. We include a detailed description of the improvements that have been reported on the use of Kalman and Information filters for SLAM, leading to the algorithm that we propose in this thesis. Then, we describe our contribution to the estimation problem in SLAM, the Combined Filter SLAM (CF SLAM) algorithm, a highly efficient filtering algorithm in section 2.2; the section 2.3 contains a study of its computational cost and consistency properties. We carry out comparisons with state of the art algorithms using benchmark datasets, like the Victoria Park and DLR datasets, in section 2.6. We also propose data association strategies for CF SLAM in section 2.7. Finally, we summarise the results and draw the fundamental conclusions of our contribution.

2.1 Extended Kalman filters, Extended Information filters and Map Joining filters

In this section we summarise the basic concepts about the basic Kalman Filter and the basic Information Filter, as well as about Map Joining techniques and the state of art SLAM algorithms that use them.

2.1.1 The Extended Kalman Filter

The Extended Kalman Filter (EKF) is one of the main paradigms in SLAM (Thrun et al. 2005; Thrun and Leonard 2008). In EKF SLAM, a map $(\mu, \mathbf{\Sigma})$ includes the state μ to be estimated, which contains the current vehicle location and the location of a set of environment features. The covariance of μ, represented by $\mathbf{\Sigma}$, gives an idea of the precision in the estimation (0 meaning total precision). EKF SLAM is an iterative prediction-sense-update process whose formulation we believe is widely known and is thus summarised in Table 2.1. During exploratory trajectories, and using a sensor of limited range thus providing a constant number of r features by observation, the size of the map n grows linearly. Given that each EKF update step is $\mathcal{O}(n^2)$, the *total* cost of carrying out EKF SLAM is known to be $\mathcal{O}(n^3)$.

	EKF-SLAM	
Prediction	$\mu_{t\|t-1} = g(u_t, \mu_{t-1})$	$\mathcal{O}(1)$
	$\mathbf{\Sigma}_{t\|t-1} = F_t \mathbf{\Sigma}_{t-1} F_t^T + G_t R_{t-1} G_t^T$	$\mathcal{O}(n)$
Innovation	$\nu_t = z_t - h(\mu_{t\|t-1})$	$\mathcal{O}(r)$
	$S_t = H_t \mathbf{\Sigma}_{t\|t-1} H_t^T + Q_t$	$\mathcal{O}(r^3)$
Test χ^2	$D^2 = \nu_t^T S_t^{-1} \nu_t$	$\mathcal{O}(r^3)$
	$K_t = \mathbf{\Sigma}_{t\|t-1} H_t^T / S_t$	$\mathcal{O}(nr^2)$
Update	$\mathbf{\Sigma}_t = (I - K_t H_t) \mathbf{\Sigma}_{t\|t-1}$	$\mathcal{O}(n^2 r)$
	$\mu_t = \mu_{t\|t-1} + K_t \nu_t$	$\mathcal{O}(n)$
Cost per step	$\mathcal{O}(n^2)$	

Table 2.1: Formulations of the computational cost of each of the operations carried out using the Extended Kalman Filter in the SLAM problem. Variable n is the size of the final state μ_t, and r is the size of the measurement vector z_t, **constant** in EKF-SLAM. The test χ^2 is only required for data association.

Jacobians			
F_t	$=$	$\left.\frac{\partial g(u_t,\mu_{t-1})}{\partial \mu_{t-1}}\right\|_{\hat{\mu}_{t-1}}$	$\mathcal{O}(1)$
G_t	$=$	$\left.\frac{\partial g(u_t,\mu_{t-1})}{\partial u_t}\right\|_{\hat{u}_t}$	$\mathcal{O}(1)$
H_t	$=$	$\left.\frac{\partial h(\mu_{t\|t-1})}{\partial \mu_{t\|t-1}}\right\|_{\hat{\mu}_{t\|t-1}}$	$\mathcal{O}(r)$

Table 2.2: Jacobians required for the EKF and EIF. Variable r is the size of the measurement vector z_t.

2.1.2 The Extended Information Filter

In Extended Information Filter (EIF) SLAM, a map $(\xi, \mathbf{\Omega})$ consists of the information state ξ to be estimated and the information matrix $\mathbf{\Omega}$, which gives an idea of the information known about the estimation (0 meaning no information). EIF SLAM is also an iterative prediction-sense-update process, its formulation is summarised in Table 2.3. The Information filter is an algebraic equivalent to the Kalman filter, because the following equivalences hold (Thrun et al. 2005):

$$\mathbf{\Omega} = \mathbf{\Sigma}^{-1} \qquad and \qquad \xi = \mathbf{\Sigma}^{-1}\mu \tag{2.1}$$

For this reason, KF and IF are considered dual filters (Mutambara and Al-Haik 1997). Unfortunately, in the nonlinear case the filters are not completely dual, since both the transition function g and the measurement function h require the state as input (Thrun et al. 2005). For this reason, the initial required computation during the prediction step is to derive the state variables μ_{t-1}. In the general case, the EIF is considered computationally more expensive than the EKF: computing the state μ_{t-1} is $\mathcal{O}(n^3)$ because of the inversion of the the information matrix $\mathbf{\Omega}_{t-1}$. In the SLAM context, the information matrix has special structural properties, thus state vector recovery can be carried out by solving a equation system of size n very efficiently through the Cholesky decomposition. An important insight into reducing the computational cost of EIF SLAM was to observe that the information matrix $\mathbf{\Omega}$ is *approximately* sparse (Thrun et al. 2005), and can be easily sparsified. This Sparse Extended Information Filter (SEIF) allows a computational cost of $\mathcal{O}(n)$ (pure exploration) up to $\mathcal{O}(n^2)$ (repeated traversal), although because of sparsification SEIF SLAM is not an exact algorithm. Another important observation regarding EIF SLAM is that if all vehicle locations are incorporated into the information vector, instead of the current one only, the information matrix becomes *exactly* sparse (Eustice et al. 2006). In this Exactly Sparse Delayed-State Filter (ESDF) SLAM, the reduction in the computational cost is the same as in SEIF. Additionally, since no approximations due to sparsification take place, the results are more precise (Eustice et al. 2005). When the state or the information vector contain only the current vehicle location, we have an *online* SLAM problem; if it contains all vehicle locations along the trajectory, we have *full* SLAM.

The total cost of ESDF is known to be range from $\mathcal{O}(n^2)$ (pure exploration) to $\mathcal{O}(n^3)$

(repeated traversal), as compared with the always cubic cost of EKF SLAM. Note however that the information vector is ever increasing. Even during revisiting, every new vehicle location is incorporated in the state vector, thus increasing n. The term r in Table 2.3 refers to the field of view of the sensor, we note again that in these cases is constant with respect to n.

	EIF-SLAM	
Prediction	$\mu_{t-1} = \mathbf{\Omega}_{t-1} \backslash \xi_{t-1}$	$\mathcal{O}(nr^2)$-$\mathcal{O}(n^2r)$
	$\Phi = F_t^{-T} \mathbf{\Omega}_{t-1} F_t^{-1}$	$\mathcal{O}(1)$
	$\mathbf{\Omega}_{t\|t-1} = \Phi - \Phi G_t (R_{t-1} + G_t^T \Phi G_t)^{-1} G_t^T \Phi$	$\mathcal{O}(n)$
	$\xi_{t\|t-1} = \mathbf{\Omega}_{t\|t-1} g(u_t, \mu_{t-1})$	$\mathcal{O}(nr)$
Innovation	$\nu_t = z_t - h(\mu_{t\|t-1})$	$\mathcal{O}(r)$
	$S_t = H_t(\mathbf{\Omega}_{t\|t-1} \backslash H_t^T) + Q_t$	$\mathcal{O}(nr^2)$
Test χ^2	$D^2 = \nu_t^T S_t^{-1} \nu_t$	$\mathcal{O}(r^3)$
Update	$\mathbf{\Omega}_t = \mathbf{\Omega}_{t\|t-1} + H_t^T Q_t^{-1} H_t$	$\mathcal{O}(r)$
	$\xi_t = \xi_{t\|t-1} + H_t^T Q_t^{-1}(\nu_t + H_t \mu_{t\|t-1})$	$\mathcal{O}(r)$
Cost per step	$\mathcal{O}(n)$ to $\mathcal{O}(n^2)$	

Table 2.3: Formulations of the computational cost of each of the operations carried out using the Extended Information Filter in the SLAM problem. Variable n is the size of the final information vector ξ_t, and r is the size of the measurement vector z_t, **constant** in EIF-SLAM. The test χ^2 is only required for data association.

2.1.3 Map Joining SLAM with EKF

Local mapping (or submapping) algorithms were the next contribution to the reduction of the computational cost of SLAM. In these algorithms, local maps of constant size p are sequentially built (in constant time because of the bounded size) and then put together into a global map in different ways.

One of such solutions, Map Joining SLAM (Tardós et al. 2002), works in the following way: given two consecutive local maps $(\mu_1, \mathbf{\Sigma}_1)$ and $(\mu_2, \mathbf{\Sigma}_2)$, the map $(\mu, \mathbf{\Sigma})$ resulting from joining their information together is computed in three initialisation-innovation-update steps, summarised in Table 2.4. A specialised version of the Extended Kalman Filter is used, where the full state vectors and covariance matrices are simply stacked in the predicted map; correspondences can then be established between features from both maps through a prediction function h, equivalent to considering a perfect measurement $z = 0, Q = 0$. Notice that this is possible because EKF allows the consideration of 0 covariance measurements. In this case, h computes the discrepancy of features from both maps in the same reference. The update step includes a computation using the function

g to first delete duplicate features appearing in both maps, and then transform all map features and vehicle locations to a common base reference, usually the starting vehicle location in the first map.

Map Joining SLAM is constant time most of the time, when working with local maps. However, map joining operations are $\mathcal{O}(n^2)$ on the final size of the map, and although it results in great computational savings (it may slash the cost by a large constant), Map Joining SLAM is still $\mathcal{O}(n^2)$ per step, just as EKF SLAM is.

	Join with EKF	
Initialisation	$\mu^- = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$	$\mathcal{O}(n)$
	$\mathbf{\Sigma}^- = \begin{bmatrix} \mathbf{\Sigma}_1 & 0 \\ 0 & \mathbf{\Sigma}_2 \end{bmatrix}$	$\mathcal{O}(n^2)$
Innovation	$\nu = -h(\mu^-)$	$\mathcal{O}(s)$
	$S = H\mathbf{\Sigma}^- H^T$	$\mathcal{O}(s^2)$
Update	$K = \mathbf{\Sigma}^- H^T / S$	$\mathcal{O}(n^2)$
	$\mathbf{\Sigma}^+ = (I - KH)\mathbf{\Sigma}^-$	$\mathcal{O}(n^2 s)$
	$\mu^+ = \mu^- + K\nu$	$\mathcal{O}(ns)$
	$\mu = g(\mu^+)$	$\mathcal{O}(n_2)$
	$\mathbf{\Sigma} = G\mathbf{\Sigma}^+ G^T$	$\mathcal{O}(n_2^2)$
Cost per join	$\mathcal{O}(n^2 s)$	

Table 2.4: Formulations of the computational cost of each of the operations carried out for Map Joining using the Extended Kalman Filter in the best case. Variables n_1, n_2 and n are the size of the first, second and final state vector respectively, s is the size of the overlap between both submaps, and p is the size of the local map, **constant** with respect to the size of the map.

2.1.4 Map Joining SLAM with EIF

The Extended Information filter can also be used to carry out the map joining operations, as reported in the Sparse Local Submap Joining filter (SLSJF) SLAM (Huang et al. 2008a). Its application is not as straightforward as the Map Joining with EKF for two reasons. First, in Map Joining with EKF, correspondences are established by considering a perfect measurement $z = 0, Q = 0$. In the information form, 0 covariance measurements are not allowed since Q^{-1} is required in the formulation. For this reason, in SLSJF SLAM, having two consecutive local maps $(\mu_1, \mathbf{\Sigma}_1)$ and $(\mu_2, \mathbf{\Sigma}_2)$ to join, the resulting map $(\xi, \mathbf{\Omega})$ is predicted in the information form with the information of the first map, and an initial 0 (no information) from the second map. The innovation is computed considering the second map as a set of measurements for the full map ($z_t = \mu_2$, $Q_t = \mathbf{\Omega}_2^{-1}$), and the final update step computes the information state ξ and information matrix $\mathbf{\Omega}$ using the standard EIF equations. Note in Table 2.5 that g has the same functionality as in the previous section without delete features, and function h transforms the features revisited from the first map to the reference of the second map.

An important observation made in (Huang et al. 2008a) is that the information matrix resulting from the map joining operation using EIF is *exactly* sparse if the vehicle locations coming from each local map are maintained in the final information state. This is a situation very similar to the full SLAM problem, except that not all vehicle locations remain, only a fraction corresponding to the final vehicle locations in each local map. There is an additional final computation of the final state μ, to make it available for potential future map joining operations. This state recovery can be done with a pre-ordering of minimum degree of the information matrix and the sparse Cholesky factorisation to solve the linear system, such as was pointed out by (Huang et al. 2008a). In this thesis we shall see that the cost of this computation can be proportional to n during exploration, and up to $\mathcal{O}(n^2)$ in the worst case.

	Join with EIF	
Initialisation	$\mu^- = g([\mu_1;\mu_2])$	$\mathcal{O}(n_2)$
	$\xi^- = \begin{bmatrix} \xi_1 \\ 0 \end{bmatrix}$	$\mathcal{O}(n)$
	$\mathbf{\Omega}^- = \begin{bmatrix} \mathbf{\Omega}_1 & 0 \\ 0 & 0 \end{bmatrix}$	$\mathcal{O}(np)$
Innovation	$\nu = \mu_2 - h(\mu^-)$	$\mathcal{O}(s)$
	$Q^{-1} = \mathbf{\Omega}_2$	given
Update	$\mathbf{\Omega} = \mathbf{\Omega}^- + H^T \mathbf{\Omega}_2 H$	$\mathcal{O}(n_2 p)$
	$\xi = \xi^- + H^T \mathbf{\Omega}_2(\nu + H\mu^-)$	$\mathcal{O}(n_2 p)$
	$\mu = \mathbf{\Omega} \backslash \xi$	$\mathcal{O}(np^2)$ to $\mathcal{O}(n^2 p)$
Cost per join	$\mathcal{O}(n)$ to $\mathcal{O}(n^2)$	

Table 2.5: Formulations of the computational cost of each of the operations carried out for Map Joining using the Extended Information Filter in the best case. Variables n_1, n_2 and n are the size of the first, second and final information vector respectively, s is the size of the overlap between both submaps, and p is the size of the local map, **constant** with respect to the size of the map.

Jacobians			
G	$=$	$\left.\frac{\partial g(\mu)}{\partial \mu}\right\|_{\hat{\mu}}$	$\mathcal{O}(n_2)$
H	$=$	$\left.\frac{\partial h(\mu^-)}{\partial \mu^-}\right\|_{\hat{\mu}^-}$	$\mathcal{O}(n_2)$

Table 2.6: Jacobians required for the map joining operations with EKF and EIF. Variable n_2 is the size of the second state or information vector.

2.1.5 Divide and Conquer with EKF

In the Divide and Conquer (D&C) SLAM algorithm (Paz et al. 2008) it was pointed out that SLAM can be carried out in *linear* time per step if map joining operations are carried out in a hierarchical binary tree fashion, instead of a sequential fashion. The leafs of the binary tree represent the sequence of l local maps of constant size p, computed with standard EKF SLAM. These maps are joined pairwise to compute $l/2$ local maps of double their size ($2p$), which will in turn be joined pairwise into $l/4$ local maps of size $4p$, until finally two local maps of size $n/2$ will be joined into one full map of size n, the final map. The $\mathcal{O}(n^2)$ updates are not carried out sequentially, but become more distant as the map grows. An $\mathcal{O}(n^2)$ computation can then be amortised in the next n steps, making the amortised version of the algorithm linear with the size of the map. It can also be shown that the total cost of D&C SLAM is $\mathcal{O}(n^2)$, as compared to the total cost of EKF SLAM, $\mathcal{O}(n^3)$.

In this thesis we aim at further reducing the computational complexity, without any sacrifice of precision or accuracy.

2.2 The Combined Filter SLAM

In the remain of this chapter, we describe our contribution to the estimation problem in SLAM, the Combined Filter SLAM (CF SLAM) algorithm, a highly efficient filtering algorithm.

2.2.1 Method

The algorithm proposed here, Combined Filter SLAM, has three main highlights (see algorithm 1):

1. *Local mapping is carried out using standard online EKF SLAM to build a sequence of maps of constant size p.* Each local map $(\mu_i, \mathbf{\Sigma}_i)$ is also stored in information form $(\xi_i, \mathbf{\Omega}_i)$; both the information vector and the information matrix are computed in $\mathcal{O}(p^3)$, constant with respect to the total map size n. Each local map only keeps the final pose of the robot. EKF SLAM in local maps allows robust data association, e.g. with JCBB (Neira and Tardós 2001), and small local maps remain consistent.

2. *Map joining is carried out using EIF, keeping vehicle locations from each local map in the final map.* This allows to exploit the exact sparse structure of the resulting information matrix and the join can be carried out in as low as linear time with the final size of the map. The covariance matrix is not computed after joins at the lower level.

3. *In contrast with the sequential map joining strategy followed by SLSJF, the D&C strategy is followed to decide when map joining takes place.* We will see that this will result in a total computation cost as low as $\mathcal{O}(n \log n)$, as compared with the total cost of SLSJF, $\mathcal{O}(n^2)$. Additionally, the computation per step can be amortised to $\mathcal{O}(\log n)$, as compared with $\mathcal{O}(n)$ for SLSJF. This makes CF SLAM the most efficient SLAM filtering algorithm.

Algorithm 1 The CF SLAM Algorithm

```
$maps \leftarrow \{\}$
while data from sensor do
    $map \leftarrow ekf\_slam$
    if $isempty(maps)$ then
        $maps \leftarrow \{map\}$
    else
        while $size(map) \geq size(maps\{last\})$
        or global map is needed do
            $map \leftarrow eif\_map\_join(maps\{last\}, map)$
            $maps\{last\} \leftarrow \{\}$
        end while
        $maps \leftarrow \{maps, map\}$
    end if
end while
```

2.3 Computational complexity of CF SLAM

We study initially the case of pure exploration, in which the robot is always observing new territory, and thus the sensor measurements have a constant overlap with the map already built. This case is very interesting because is the first situation that any SLAM algorithm will face, and it is also the case where the size of the map increases, and thus also the size of the problem. We discuss several other cases in section 2.5.

In exploratory trajectories, the process of building a map of size n using the proposed CF SLAM follows the divide and conquer strategy: $l = n/p$ maps of size p are produced (not considering overlap), at cost $\mathcal{O}(p^3)$ each, which are joined into $l/2$ maps of size $2p$, at cost $\mathcal{O}(2p)$ each. These in turn are joined into $l/4$ maps of size $4p$, at cost $\mathcal{O}(4p)$ each. This process continues until two local maps of size $n/2$ are joined into one local map of size n, at a cost of $\mathcal{O}(n)$. SLSJF SLAM and our algorithm carry out the same number of map joining operations. The fundamental difference is that in our case the size of the maps involved in map joining increases at a slower rate than in SLSJF SLAM.

The *total* computational complexity of CF SLAM in this case is:

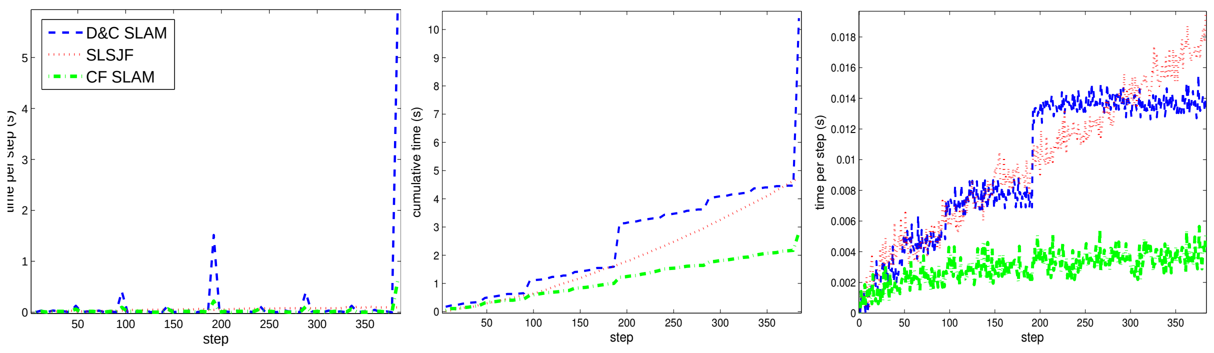
$$\begin{aligned}
C &= \mathcal{O}\left(p^3 l + \sum_{i=1}^{\log_2 l} \frac{l}{2^i}(2^i\,p)\right) \\
&= \mathcal{O}\left(p^3 n/p + \sum_{i=1}^{\log_2 n/p} \frac{n/p}{2^i}(2^i\,p)\right) \\
&= \mathcal{O}\left(p^2 n + \sum_{i=1}^{\log_2 n/p} n\right) \\
&= \mathcal{O}\left(n + n\log n/p\right) \\
&= \mathcal{O}\left(n + n\log n\right) \\
&= \mathcal{O}\left(n\log n\right)
\end{aligned}$$

Therefore, CF SLAM offers a reduction in the total computational cost to $\mathcal{O}\left(n\log n\right)$, as compared with the total cost $O\left(n^3\right)$ for EKF SLAM, and $\mathcal{O}\left(n^2\right)$ for D&C SLAM and SLSJF SLAM. Furthermore, as in D&C SLAM, the map to be generated at step t will not be required for joining until step $2\,t$. This allows us to amortise the cost $\mathcal{O}(t)$ at this step by dividing it up between steps $t+1$ to $2\,t$ in equal $\mathcal{O}(1)$ computations for each step. In this way, our amortised algorithm becomes $\mathcal{O}(\log n)$ *per step*. In the worst cases the cost can grow up to $\mathcal{O}(n)$, but the cost of D&C SLAM and SLSJF SLAM will also grow, see section 2.5.

2.4 CF SLAM vs. other filtering algorithms

2.4.1 Computational cost

To illustrate the computational cost of the algorithm proposed, we conducted a simulated experiment using a simple MATLAB implementation of CF SLAM, D&C SLAM (Paz



(a) (b) (c)

Figure 2.1: Computational time in the simulated execution for D&C SLAM (blue), SLSJF SLAM (red) and our algorithm, CF SLAM (green): Time per step 2.1(a); Total time of execution 2.1(b); Time per step for SLSJF vs. amortised time per step for D&C SLAM and our algorithm, CF SLAM 2.1(c). The final map contains 1093 features from 64 local maps.

et al. 2008), and SLSJF (Huang et al. 2008a) (with reordering using *symmmd* instead of the heuristic criteria proposed there). The simulated environment contains equally spaced point features $1.33m$ apart, and a robot equipped with a range and bearing sensor with a field of view of 180 degrees and a range of $2m$ in a pure exploratory trajectory. Local maps are built containing $p = 30$ features each. All tests are done over 100 MonteCarlo runs. Fig. 2.1 shows the resulting execution costs of the three algorithms. We can see that the total costs of D&C SLAM and the SLSJF tend to be equal, Fig. 2.1(b). This is expected, since both have a total cost of $\mathcal{O}(n^2)$. We can also see that the total cost of our algorithm increases more slowly, it is expected to be $\mathcal{O}(n \log n)$. Finally, we can see that the amortised cost of our algorithm exhibits an $\mathcal{O}(\log n)$ behaviour, always smaller than the cost of the other two algorithms, Fig. 2.1(c).

We also simulated a robot moving in a 4-leaf clover trajectory, Fig. 2.2(a). The robot is equipped with a range and bearing sensor. The features are uniformly distributed with a separation between them of $6m$. There are 1660 features and 9150 steps. 477 submaps were obtained. The cost for each EKF step in mean was $1.6ms$ with a total of $14s$, equal for all the algorithms tested. Fig. 2.2(b) shows the computation cost per step of Map Joining SLAM and SLSJF vs. the amortised cost for the D&C SLAM and CF SLAM. We can see the sublinear cost of CF SLAM in the map updates as expected. Fig. 2.2(c) shows the cumulative costs of map updates. The algorithms based on EKFs did not solve the problem completely because they exceeded the available memory before the end of the experiment.

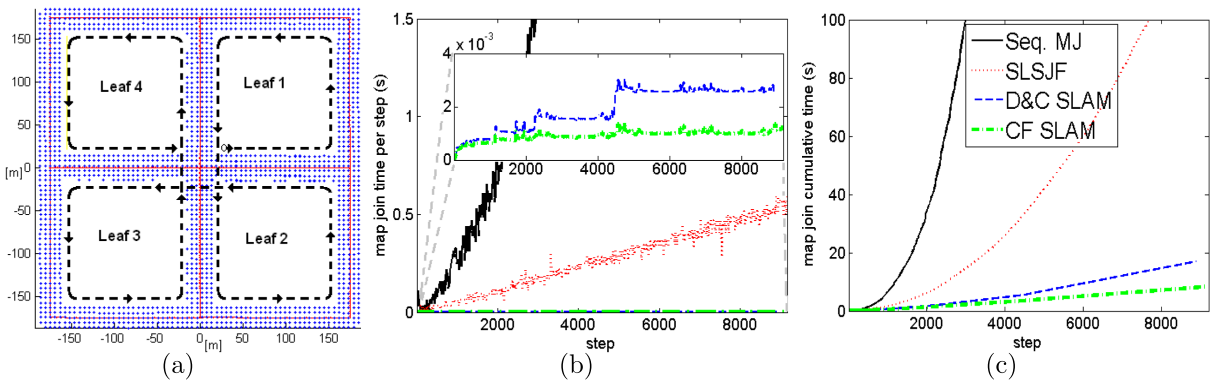


Figure 2.2: Simulated experiment of a 4-leaf clover trajectory (a). In (b) map update time per step, and map update cumulative times for all algorithms in (c).

2.4.2 Consistency

When the ground truth is available as in this simulated experiment, we can carry out a statistical test on the estimation $(\mu, \mathbf{\Sigma})$ for filter consistency. We define the consistency index, CI, as:

$$CI = \frac{D^2}{\chi^2_{n,1-\alpha}} \tag{2.2}$$

where D^2 is the Mahalanobis distance or Normalised Estimation Error Squared (NEES) (Bar-Shalom et al. 2001), $n = dim(\mu)$ and $(1 - \alpha)$ is the confidence level (95% typically). When $CI < 1$, the estimation is consistent with the ground truth, otherwise the estimation is optimistic, or inconsistent.

It is known that local map-based strategies, e.g. SLSJF and D&C SLAM, improve the consistency of SLAM by including less linearisation errors than strategies based on one global map, e.g. Treemap (Huang and Dissanayake 2007). We tested consistency of SLSJF, D&C SLAM and CF SLAM on the simulated experiments with 100 MonteCarlo runs. Fig. 2.3 shows the evolution of the mean consistency index of the vehicle position in x (left) and y (center), and orientation ϕ (right), during the steps of the trajectory. We can see that the performance of the index for D&C SLAM and for our proposal a very similar and clearly better than for SLSJF. This is due to the fact that both D&C SLAM and our proposal follow a tree structure to carry out the map joining process. In contrast, in SLSJF map joining is sequential, thus errors increase faster in the global map. Recently, a more consistent filter based on SLSJF, I-SLSJF, was proposed by Huang et al. (2008b). This filter, in addition to being more computationally expensive than the original SLSJF, requires setting a threshold empirically to decide when it is necessary to

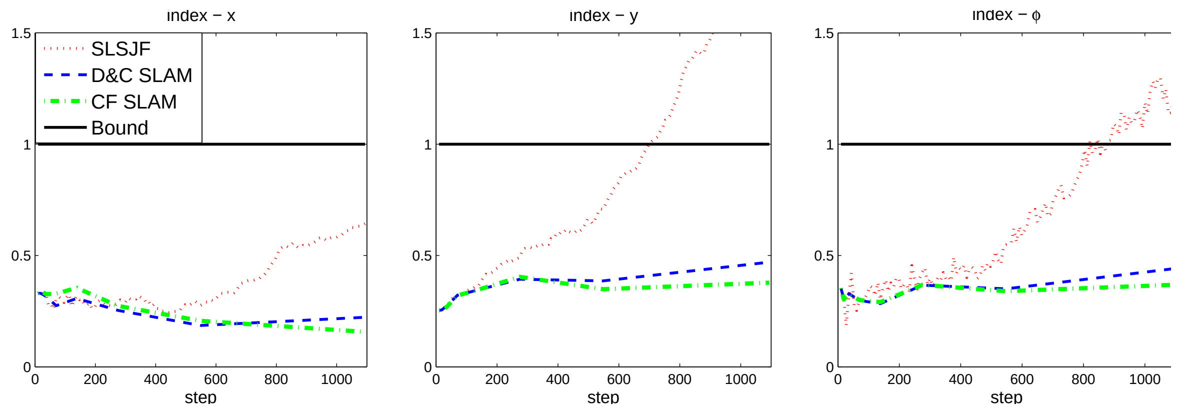


Figure 2.3: Mean consistency index CI in x, y, and ϕ for SLSJF, D&C SLAM and our CF SLAM.

solve the least squares problem and recompute the state vector from all the local maps stored.

2.5 Factors that have influence in the computational cost

2.5.1 Vehicle trajectory

Given that local mapping is a constant time operation, we concentrate on the computational cost of map joining in CF SLAM. The state recovery is the most computationally expensive operation in this case. State recovery is carried out using the Cholesky factorisation, with a previous minimum degree ordering of the information matrix. The cost of this operation depends on the sparsity pattern of the information matrix and the density of non-zero elements (Gilbert et al. 1992; Huang et al. 2008a). This in turn depends on the environment, on the sensor used and more importantly, on the trajectory of the robot.

We have used the simulated experiments to study the effect of the trajectory of the vehicle in the computational cost of the map joining operation. Fig. 2.4 shows the trajectory studied (left), the sparsificacion pattern of the information matrix of the final map (middle), and the mean cost (for the 100 Monte Carlo runs) of state recovery and joining between maps versus the dimension of the state vector after joining (right).

To determine the order of the computational cost, we compute a fit to equation $y = ax^b$ for the state recovery and for map joining costs. The independent variable is the dimension of the state vector resulting from each map joining. The dependent variable is the cost of the operation: `sr` for the state recovery $\mu = \mathbf{\Omega} \backslash \xi$, and `jEIF` is the cost of all operations involved in map joining with EIF, including `sr`. The results of the fit can be seen in Table 2.7. The sum of squared errors (SSE), the coefficient of determination (R^2) and root mean squared errors (RMSE) are reported.

The most important results are:

- In pure *exploratory trajectories*, Fig. 2.4 (top), the fit suggests that the exponent b can be considered equal to 1 in both state recovery and map joining operations, see Table 2.7 (top). Thus, as we said in the previous section, in the case of exploration, the cost of map joining has a linear behaviour with respect to the dimension of the state vector, and can be amortised in CF SLAM to attain $\mathcal{O}(\log n)$.

- We have also studied *lawn mowing*, Fig. 2.4 (upper middle). This type of trajectory is frequent in underwater mosaicing applications (Eustice et al. 2006). In this case, the cost can increase to $\mathcal{O}(n^{1.3})$ (see the exponent from the fit, Table 2.7).

- In another type of trajectory, *outward spiral*, Fig. 2.4 (lower middle), the cost can increase to $\mathcal{O}(n^2)$. Outward spirals are frequent in airborne mapping applications (Bryson and Sukkarieh 2008).

- In the worst case, *repeated traversal* (in this case a loop), Fig. 2.4 (bottom), the cost of map joining is linear most of the time, except in the full overlap, when the operation is quadratic with the dimension of the state vector.

Three things are important to note:

1. Whatever the cost of the map joining operation, it can be amortised in CF SLAM. This means that in the worst case, when map joining is $\mathcal{O}(n^2)$, CF SLAM is $\mathcal{O}(n)$ per step.

2. In these cases, the computational cost of D&C SLAM and SLSJF also increase in the same manner: in the worst case, both will be $\mathcal{O}(n^2)$ per step, so CF SLAM will *always* be better.

3. Worst case situations will probably not very frequent, once a map of the environment of interest is available, the robot can switch to localisation using an *a priori* map, a much less expensive task computationally.

Table 2.8 summarises the computational costs of all algorithms in the best and worst cases.

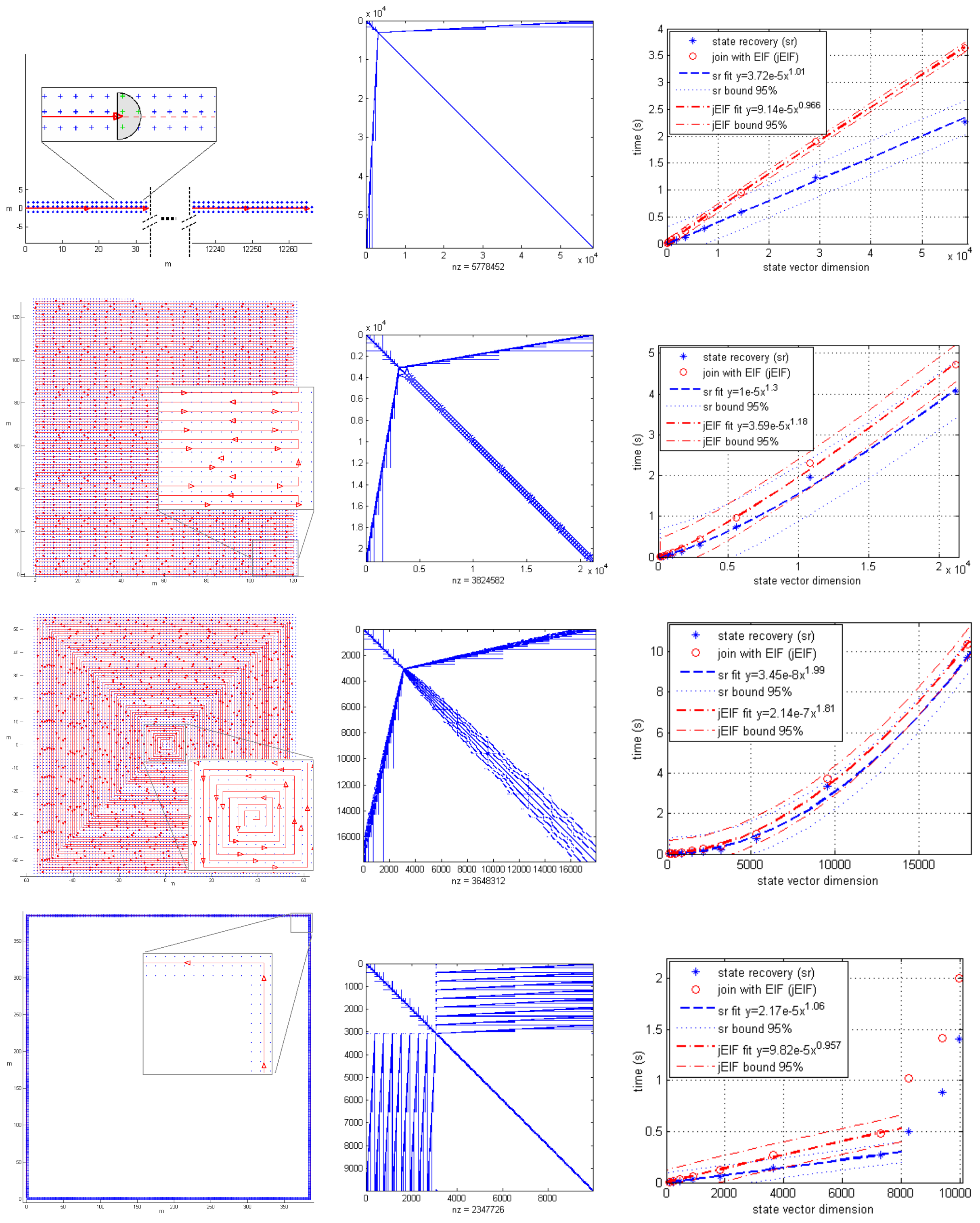


Figure 2.4: Computational cost of state recovery in four cases: exploration with 27651 features (top), lawn mowing with 9056 features (upper middle), out-ward spiral with 7423 features (lower middle), several loops with 3456 features (bottom), all from 1024 local maps. From left to right: ground truth environment and trajectory, sparse information matrix obtained by our algorithm and execution time to do joining and recovery state versus the state vector's dimension with their fit functions. In order to concentrate in studying computational costs these simulations were carried out with noise equal to zero.

Trajectory		b	(95%)	SSE	R^2	RMSE
Exploration	sr	1.01	(0.956,1.058)	0.1385	0.9976	0.1316
	jEIF	0.97	(0.955,0.977)	0.0094	0.9999	0.0342
Lawn mowing	sr	1.30	(1.21,1.39)	0.6849	0.9976	0.2926
	jEIF	1.18	(1.13,1.237)	0.2900	0.9991	0.1904
Out-ward spiral	sr	1.99	(1.816,2.157)	0.9726	0.9993	0.3497
	jEIF	1.81	(1.695,1.92)	0.6629	0.9995	0.2880
Inside the loops	sr	1.06	(0.982,1.139)	0.0072	0.9985	0.0379
	jEIF	0.96	(0.884,1.03)	0.0119	0.9976	0.0487
Exploration with smallest local maps	sr	1.14	(1.097,1.174)	0.1151	0.9990	0.1023
	jEIF	1.08	(1.032,1.123)	0.1958	0.9980	0.1334

Table 2.7: Results of the fit to $y = ax^b$, both the cost of state recovery (`sr`) and the joining with EIF (`jEIF`). We can see the value of the exponent b with 95% confidence bounds, the sum of squared errors (SSE), the coefficient of determination (R^2) and root mean squared errors (RMSE) for the simulations of Fig. 2.4 and Fig. 2.5(right).

	Cost per step		Total cost	
	Best	Worst	Best	Worst
SLSJF	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
D&C SLAM	$\mathcal{O}(n)$ amort.	$\mathcal{O}(n^2)$ amort.	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
CF SLAM	$\mathcal{O}(\log n)$ amort.	$\mathcal{O}(n)$ amort.	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^2)$

Table 2.8: Computational costs for all filtering algorithms in the best case (pure exploration) and in the worst case (repeated traversal). Note that the CF SLAM is the most efficient always.

2.5.2 Local map size

The selection of the local map size p can also influence the computational cost of CF SLAM. Local maps of a large size p (for example $p = 350$) cannot be computed in real time, and also increase the density on non-zero elements in the information matrix (see Fig. 2.5, top left). If on the contrary the local map size is too small ($p = 4$), a large number of robot poses will appear in the state vector (Fig. 2.5, top right). Both situations may result in the cost of map joining not being linear any more (Fig. 2.5, bottom).

In the first case, the density of non-zero elements in $\mathbf{\Omega}$ is 1 in every 12, and thus map joining in the lower levels of the tree (the most frequent) are very expensive, more than $4s$ in the simulation. In the case of small local maps, the exponent from the fit increases to 1.14, see Table 2.7. The density is much lower, 1 in 1070, but the state vector is much larger because we have many more poses. In Fig. 2.5 the number of features is different because the memory requirements of the scenario on the left overflowing the capacity of MATLAB. In our experience, a good rule of thumb is that we should select p to keep the feature variable vs. pose variable ratio between 5 and 20.

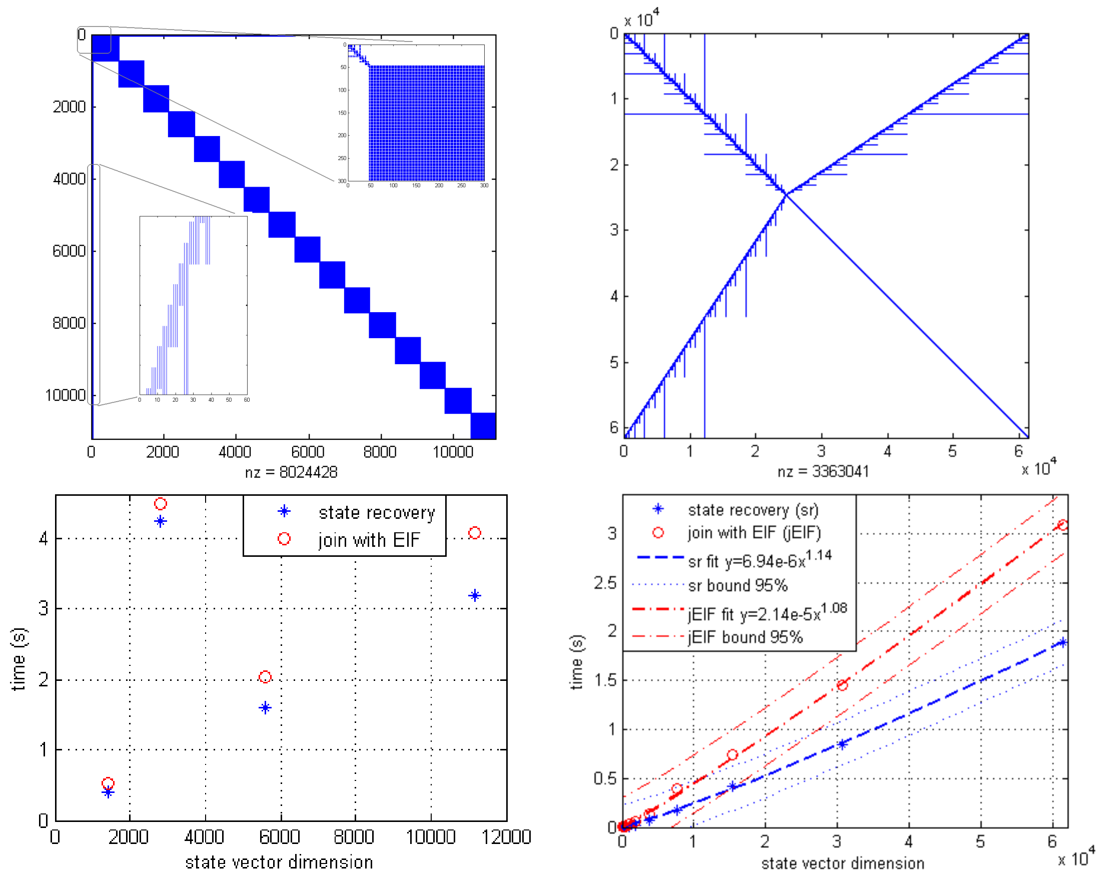


Figure 2.5: Results of different sizes of local maps. (Top) The sparse matrix information and (bottom) execution time of state recovery and map joining, both in exploration trajectory. On the left the local map size is 350 features, note the time needed for a final map of 5564 features from 16 local maps. On the right the local map size is 4 features (field of view of the sensor) and the final map has 18431 features from 8192 local maps.

2.6 Experiments

To compare the performance of the Combined Filter with other popular EKF and EIF based algorithms we use publicly available datasets. All algorithms were implemented in MATLAB and executed on 2.4GHz Intel Core 2 CPU 6600 and 3GB of RAM.

2.6.1 The Victoria Park dataset: Comparison with iSAM, iSAM2, TSAM2 and CI Graph

The Fig. 2.6(left) shows the resulting map obtained by CF SLAM on the Victoria Park dataset. The trajectory of the vehicle explores and revisits frequently. We can see in Fig. 2.6(center and right) that CF SLAM is the most efficient for map updates.

Different setups of Victoria Park are used in the literature, considering a different

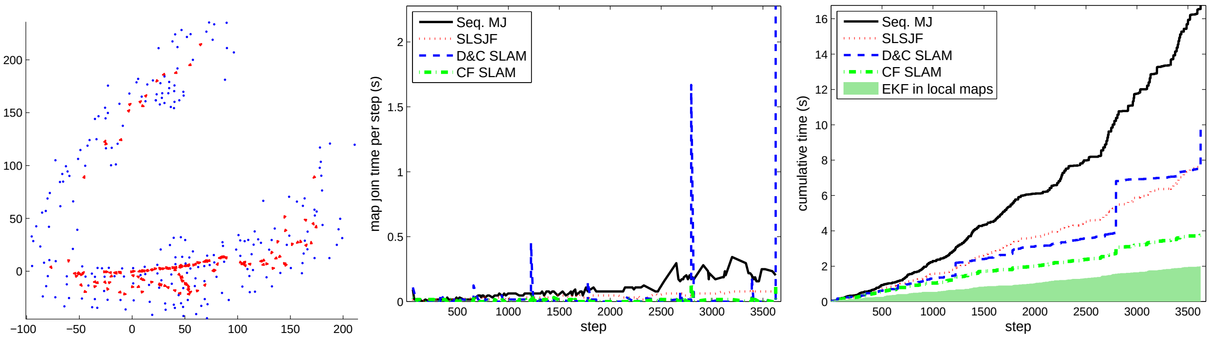


Figure 2.6: Results using the Victoria Park dataset. On the left, the final map with CF SLAM. Computational time per step for map updates (center), and cumulative total cost including each EKF step inside of the local maps (right).

number of odometry steps and also different noise models for the sensor data. We sample one in every two odometry steps, processing a total of 3625. The iSAM algorithm has also used this dataset for its tests, sampling at every odometry step. Kaess et al. (2008) implemented it in OCaml on a 2GHz Pentium M laptop computer. The authors report a cumulative time of 159s with known data association. Kaess et al. (2011) reported 31.3s for a more recent implementation of iSAM (iSAM1) and 27.2s for iSAM2 (an graph-based version of iSAM1 much more efficient), both on laptop with Intel Core 2 Duo 2.4GHz processor. Comparatively, CF SLAM takes 4.15s for solving all the dataset: 2.1s for building local maps, 2.05s for map joins. Although our sampling frequency is one half, this only affects the speed of computation of the local maps, potentially doubling 2.1s of the 4.15s for a total of 6.25s, still much lower. The work of Ni and Dellaert (2010) recently reports results of Tectonic SAM (TSAM2), a batch algorithm of smoothing and mapping with submapping, using also Victoria Park. They report a total time for optimising around of 200 submaps of 4.5s on an Macbook Pro with 2.8GHz CPU, without reporting the time for building the 200 submaps. The particle filter approach also was used over this dataset, the FastSLAM 2.0 (Montemerlo et al. 2003), implemented on 1GHz pentium PC, reported a total time of 54s using 1 particle, and 315s with 50 particles. The very recent CI-Graph algorithm of Pinies et al. (2009) solved the Victoria Park dataset in a total time greater than 50s implemented in MATLAB on a Pentium IV at 2.8GHz.

2.6.2 The DLR dataset: Comparison with Treemap

In the DLR dataset, the robot is equipped with a camera, and carries out a trajectory almost all indoors. Features are white cardboard circles placed on the ground. This

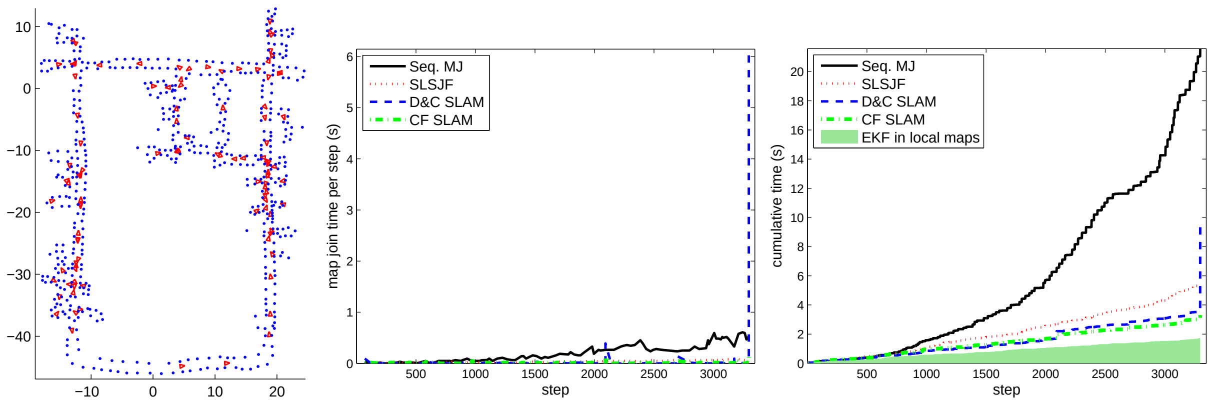


Figure 2.7: Results using the DLR-Spatial-Cognition dataset. Computational cost per step for map updates (center), and cumulative total cost including each EKF step inside of the local maps(right).

dataset has 560 features and 3297 odometry steps. The path consists of a large loop with several smaller loops in the way. The cumulative computational costs are show in Fig. 2.7(right). In this mostly exploratory dataset, both D&C SLAM algorithms are clearly superior than both sequential algorithms.

Frese (2006) reported a total execution time of 2.95s with the Treemap for this dataset, with known data association, implemented in C++ on an Intel Xeon, 2.67GHz. Our algorithm implemented in MATLAB spent 3.4s. We believe that the CF SLAM algorithm is much more simple to implement and use. Furthermore, Treemap is expected to be less consistent in general, it being an absolute map algorithm (Huang and Dissanayake 2007).

2.7 CF SLAM with unknown correspondences

The problem of data association is often ignored when evaluating the efficiency and effectiveness of a SLAM algorithm. In CF SLAM the state covariance is only available in local mapping. When the covariance matrix is not directly available, data association can be usually solved in two ways: (1) recovering the covariance necessary sub-matrices and carrying out statistical tests to find possible matches based on stochastic geometry; (2) if additional information is available, such as texture or appearance in vision sensors, we can obtain matchings based not on location but on appearance. In the following we describe two data association algorithms, one belonging to each category, that can be used in combination with CF SLAM.

2.7.1 Data association using geometrical information

In some situations, only geometrical information, the uncertain location of features relative to the sensor location, is available for data association. Such is the case of 2D points or straight walls obtained with a laser sensor.

As already has been noted by different authors (Huang et al. 2008a; Kaess and Dellaert 2009) recovering the covariance, or parts, is sometimes vital for reliable data association solutions. Kaess and Dellaert (2009) recovered the marginal covariances with dynamic programming in their iSAM framework, and then applied successfully the JCBB technique. Huang et al. (2008a) recovered columns of the covariance matrix by solving the a sparse linear equation efficiently.

In CF SLAM, data association inside a local map is carried out using RJC because the corresponding covariance matrix is available. The data association that remains to be solved is the identification of features that appear in two consecutive maps, lets call them $M_1 = \{\mu_1, \xi_1, \mathbf{\Omega}_1\}$ and $M_2 = \{\mu_2, \xi_2, \mathbf{\Omega}_2\}$, either local maps at the lower level, or maps resulting from previous joins in the D&C map joining process. When the two maps to be joined are local maps, their corresponding covariances are available and data association can be done with the RJC also. If not, we first determine the overlap between the two maps, features that can potentially have pairings, and then we recover the covariance matrix for those features only. We proceed as follows:

1. *Identify the overlap, a set of potential matches*

   This is denominated individual compatibility, IC. Individually compatible features are obtained by tessellating the environment space, as was proposed by Paz et al. (2008), but in our case we represent the grid in polar coordinates, see Fig. 2.8. For each feature in the second local map M_2, we assign an angular window of constant width in angle and height proportional to its distance from the origin, so that more distant features will have a larger region of uncertainty. The features in the first local map M_1 are referenced on M_2 through the last vehicle pose in M_1, μ_{x1}, which is the origin of map M_2. The uncertainty of μ_{x1} is recovered using equation 2.3 (below) and propagated on the these features, transformed to polar coordinates and embedded to angular windows. Features that intersect are considered individually compatible, giving IC.

$$\mathbf{\Omega\Sigma}_{x1} = e_{x1} \tag{2.3}$$

e_{x1} is a column vector with ones in the correspondent components to the pose x_1 and zeros for the rest.

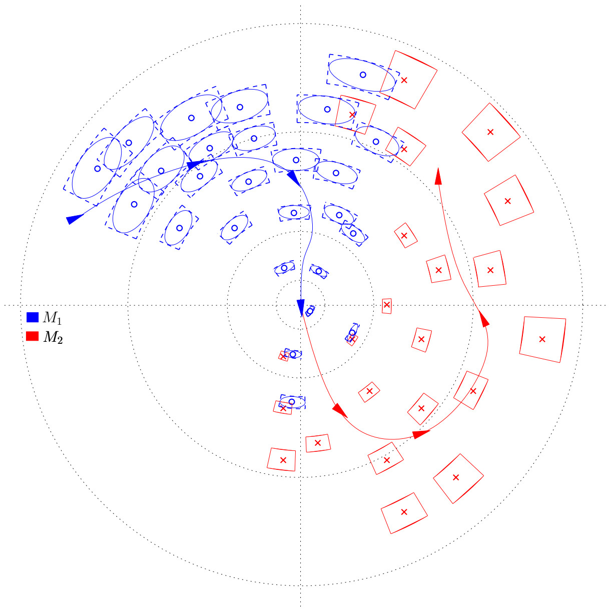


Figure 2.8: Computing the individual compatibility matrix for two local maps using polar coordinates. The angular windows for the features in the M_2 have constant width in angle and height proportional to the distance to the origin. Blue ellipses represent the uncertainties of the predicted features of the first local map with respect to the base reference of the second. The ellipses are approximated by bounding windows.

2. *Partial recovery of covariances*

   For intermediate maps that are not at the lower local level CF SLAM does not compute covariances. As shown by Huang et al. (2008a), we can recover some columns of the covariance matrix by solving the sparse linear equation 2.4. The columns that we require are given by IC. We form a column selection matrix E_{IC} to obtain the columns that are given by IC. If column i of the covariance matrix is required, we include this column vector in E_{IC}:

$$e_i = [\overbrace{0, \cdots, 0, 1}^{i}, 0, \cdots, 0]^T$$

   The sparse linear system to be solved is as follows:

$$\mathbf{\Omega\Sigma}_{IC} = E_{IC} \tag{2.4}$$

This partial recovery of the covariance matrix allows to use robust joint compatibility tests for data association. The efficiency of solving equation 2.4 is the same as for the recovery of the state vector: in the best case, in exploration trajectories, the order is $\mathcal{O}(n)$, amortised $\mathcal{O}(\log n)$. In the worst case, repeated traversal, the overlap will be the full map, and the cost will be $\mathcal{O}(n^3)$, amortised $\mathcal{O}(n^2)$. Here we reuse the Cholesky decomposition used for the state vector recovery.

3. *Prediction and Observation*

   At this point the features of M_1 that have potential matches according to IC, and their covariances, are transformed to be referenced on M_2.

4. *Randomised Joint Compatibility*

   We use the RJC algorithm of Paz et al. (2008), a combination of JCBB and RANSAC that allows to carry out robust data association very efficiently, without traversing the whole solution space.

Algorithm 2 Data association for the Combined Filter using geometrical information only

```
Input: Two maps: $\langle M_1 = \{\mu_1, \xi_1, \mathbf{\Omega}_1\}, M_2 = \{\mu_2, \xi_2, \mathbf{\Omega}_2\}\rangle$
Output: Hypothesis $\mathcal{H}$
  Find the set of potential matches $IC \leftarrow (\mu_1, \mu_2)$
  if covariance matrices are available then
    extract covariances
    $(\mathbf{\Sigma}_{1i}, \mathbf{\Sigma}_{2j}) \leftarrow select(\mathbf{\Sigma}_1, \mathbf{\Sigma}_2, IC)$
  else
    recover partial covariances
    $(\mathbf{\Sigma}_{1i}, \mathbf{\Sigma}_{2j}) \leftarrow recoveryP(\mathbf{\Omega}_1, \mathbf{\Omega}_2, IC)$ eq: 2.4
  end if
  $predictions = (h, H\mathbf{\Sigma}H) \leftarrow predict\_map(\mu_{1i}, \mathbf{\Sigma}_{1i})$
  $observations = (z, R) \leftarrow (\mu_{2j}, \mathbf{\Sigma}_{2j})$
  $\mathcal{H} \leftarrow RJC(predictions, observations, IC)$
```

In the following, we will show that using this algorithm, CF SLAM is computationally as efficient as D&C SLAM but requires less memory because the full covariance matrix is not computed. We use the same experiments that in the previous sections.

The 4-leaf clover simulated experiment

Data association was determined based only on geometric information using algorithm 2. Fig. 2.9 shows the maps obtained with each algorithm on the first row. On the second

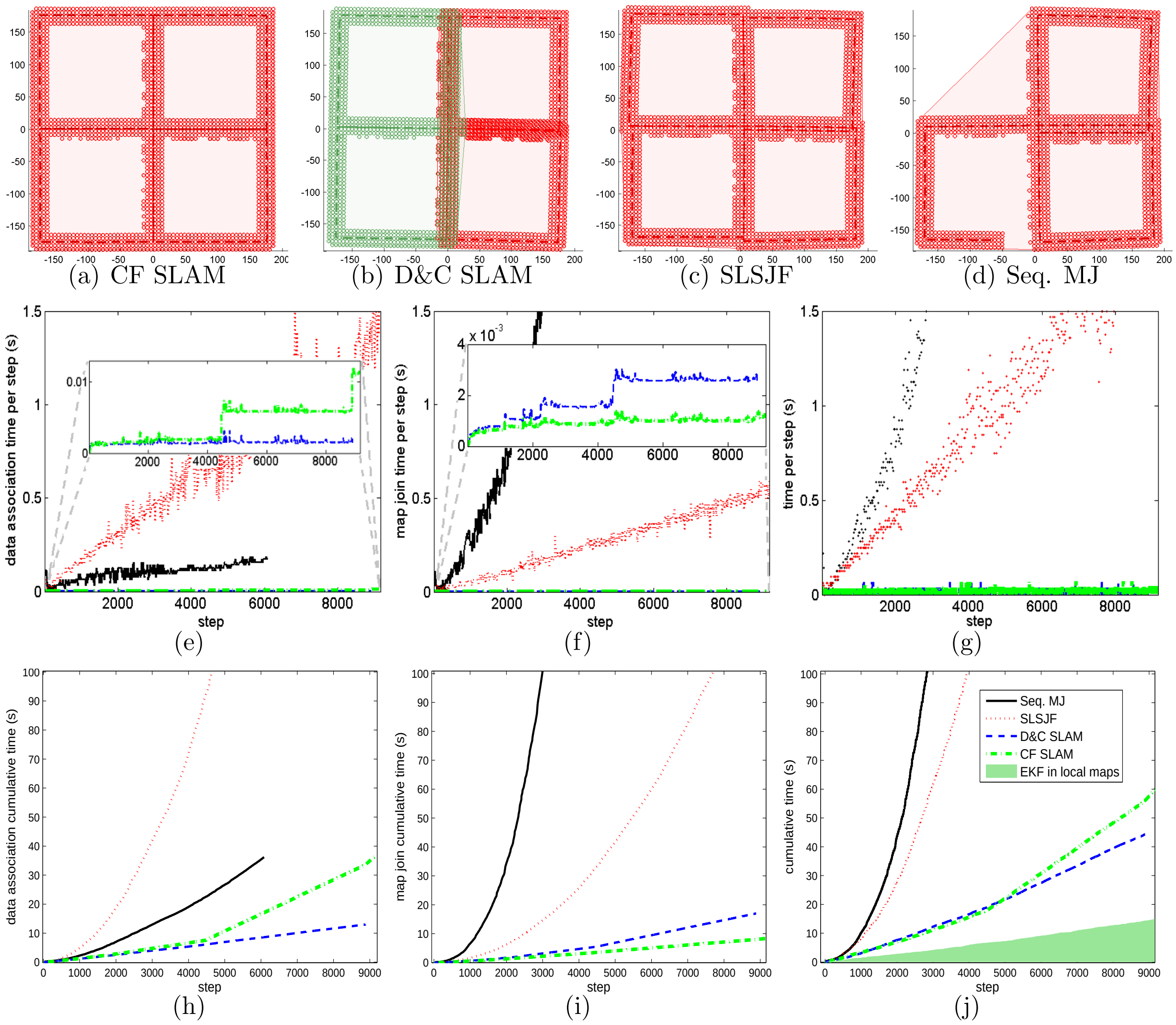


Figure 2.9: Simulated experiment of a 4-leaf clover trajectory. First row, the maps resulting for each algorithm. In the second row the computational times per step, and the cumulatives on the last row. In (e,h) the computational costs for solving data association, in (f,i) for map updates, and the total cost including each EKF step inside of the local maps for all algorithms in (g,j).

row, it shows the computation cost per step of Map Joining SLAM and SLSJF vs. the amortised cost for the D&C SLAM and CF SLAM: left, cost of data association, center: cost of map updates, right: total cost including local map building. On the third, it shows the cumulative costs in same order than before. The algorithms based on EKFs did not solve the problem completely because they exceeded the available memory before the end of the experiment.

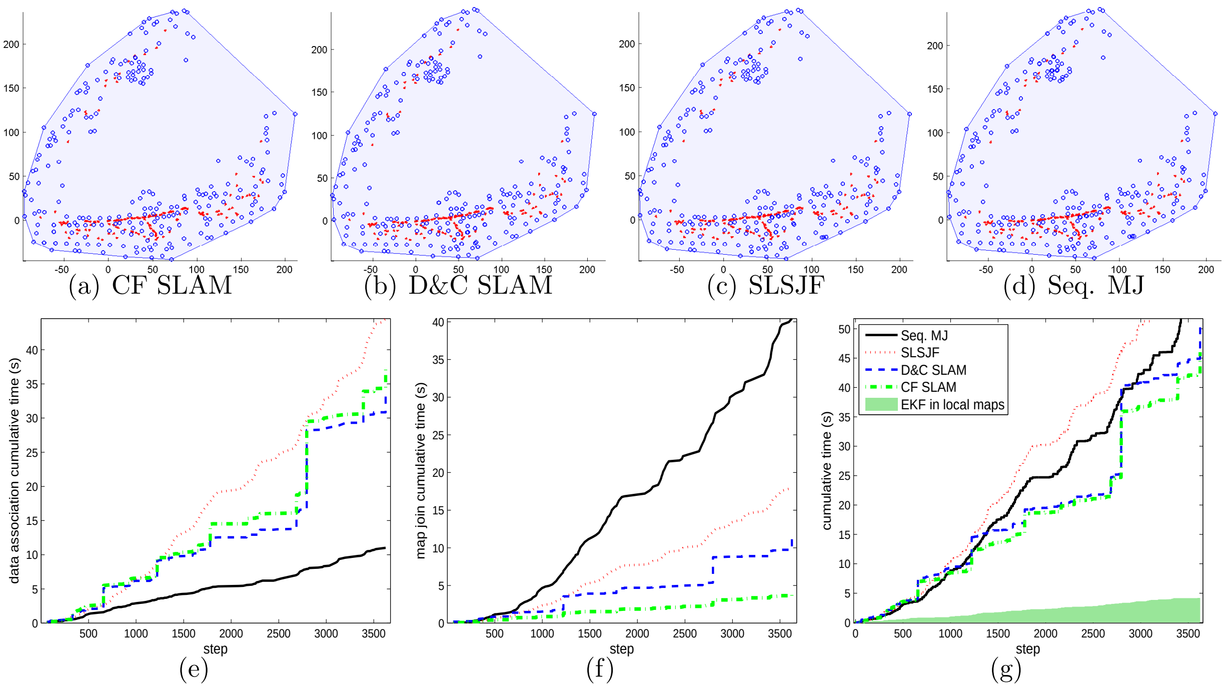


Figure 2.10: Victoria Park experiment. Top, the maps resulting for each algorithm. Bottom, in 2.10(e) the cumulative computational costs for solving data association, in 2.10(f) for the map updates, and the total cost for all algorithms in 2.10(g).

The Victoria Park dataset

The figure 2.6 shows the resulting map obtained by CF SLAM on the Victoria Park dataset. All algorithms solve this dataset correctly. The trajectory of the vehicle explores and revisits frequently, so the uncertainty does not grow much and errors are kept small. The data association was determined with algorithm 2. This dataset is interesting to compare CF SLAM and SLSJF. Both algorithms require the recovery of the covariance sub-matrix for the overlap between the maps to be joined. There are some areas where the overlap is almost complete, thus requiring the recovery of almost the full covariance matrix. The cumulative computational costs are in Fig. 2.10(g). We can see that CF SLAM is the most efficient for map updates, but Map Joining SLAM is most efficient for data association. In total, both algorithms that use the D&C strategy tend to be most efficient.

The DLR dataset

The path consists of a large loop with several smaller loops in the way. Position errors grow enough so that sequential algorithms, Map Joining SLAM and SLSJF, become weak and fail in the data association to close the loop, see Fig. 2.11(c,d). The D&C algorithms, D&C SLAM and CF SLAM have better consistency properties, and both solve the data

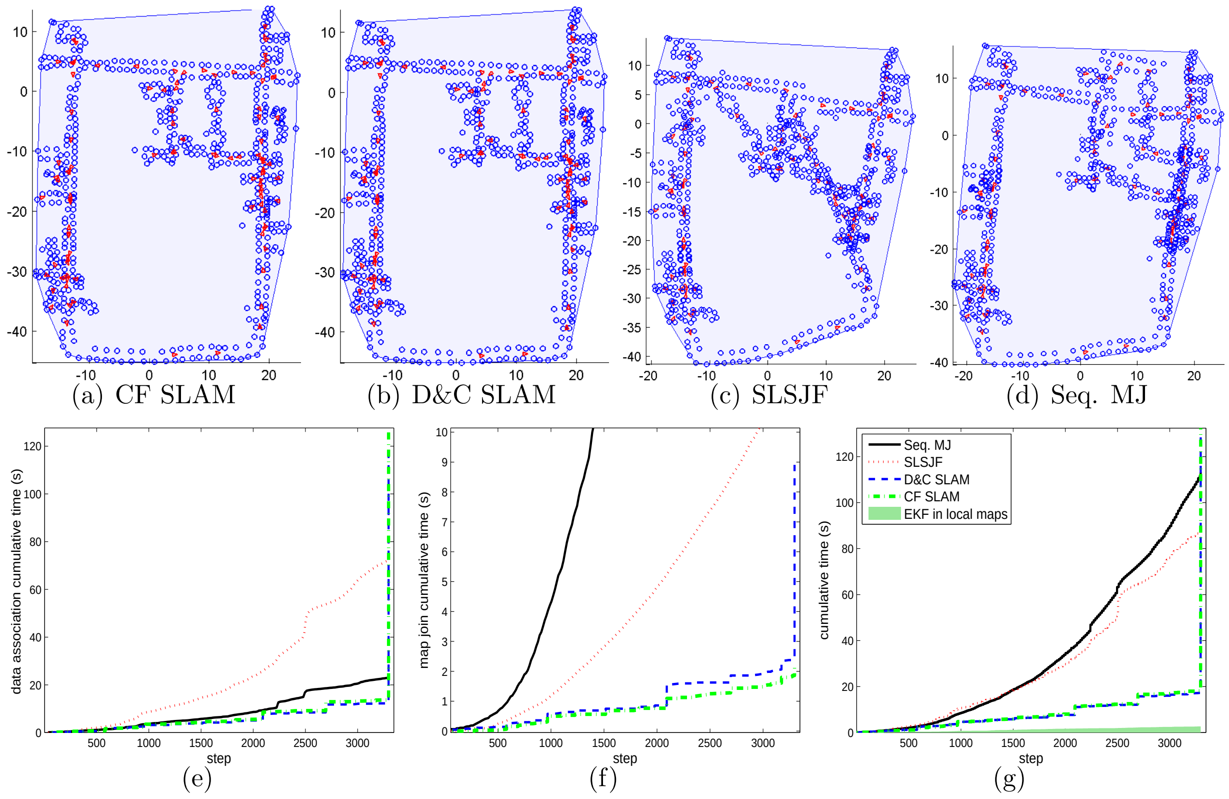


Figure 2.11: Results using the DLR-Spatial-Cognition dataset. Top, the maps resulting for each algorithm. Bottom, in 2.11(e) the computational costs for solving data association. In 2.11(f) the map update time per step, and the cumulative time for all algorithms in 2.11(g).

association for the loop closing in this dataset, see Fig. 2.11(a,b). The cumulative computational costs are shown in Fig. 2.11(e-i). In this mostly exploratory dataset, both D&C SLAM algorithms are clearly superior than both sequential algorithms. During loop closing, the cost of data association for both D&C algorithms is higher than that of both sequential algorithms, for the good reason that data association is computed correctly and the loop can be closed.

Olson (2009) reported a total time of 199s for solving data association using spectrally clustered local matches, implemented on a 2.46GHz Intel proccesor. Our algorithm implemented in MATLAB spent 128s in data association between local maps.

2.7.2 Data association using appearance information

In some cases, features in local maps can have associated appearance information, such as texture coming from vision. In these cases, appearance can be coded using descriptor vectors d from for example SIFT (Lowe 2004) or SURF (Bay et al. 2006), then the map is represented by $M = \{\mu, \xi, \mathbf{\Omega}, d\}$. In these cases, we can proceed as follows in CF SLAM:

1. *Obtain a set of potential matches*

   We find the best possible matches between the descriptors in M_1 and M_2 by searching for the nearest neighbour in the descriptor space. In this way we obtain the individual compatibility matrix, IC.

2. *Obtain a pairwise hypothesis using RANSAC*

   For each pair of minimum local maps, map i belonging to M_1 and map j belonging to M_2, that have a minimum number of matches (5 in our case) in IC, we use RANSAC (Fischler and Bolles 1981) to find the subset $\mathcal{H}_{ij}$ of matches that corresponds to the best rigid-body transform between the two local maps. In Fig. 2.12, the transformations between the pairs $(i = 1, j = 3)$, $(i = 2, j = 3)$ and $(i = 3, j = 2)$ are not evaluated because they do not have sufficient matches.

3. *Obtain the final hypothesis*

   In most cases, the final hypothesis $\mathcal{H}$ is simply the result of joining all $\mathcal{H}_{ij}$. When there is ambiguity (one local map matched with two or more other local maps), we prefer the hypothesis for pairs of maps that have a smallest relative distance, because they have smaller relative errors. In Fig. 2.12 there is ambiguity between $\mathcal{H}_{41}$ and $\mathcal{H}_{42}$. In this case we accept hypothesis $\mathcal{H}_{41}$.

Algorithm 3 Data association using appearance information

```
Input: Two maps: $\langle M_1 = \{\mu_1, \xi_1, \mathbf{\Omega}_1, d_1\}, M_2 = \{\mu_2, \xi_2, \mathbf{\Omega}_2, d_2\}\rangle$
Output: Hypothesis $\mathcal{H}$
  Find set of potential matches $IC \leftarrow (d_1, d_2)$
  for each pair minimum local maps $i, j$ in $IC$ do
    $\mathcal{H}_{ij} \leftarrow RANSAC(\mu_{1i}, \mu_{2j})$
  end for
  $\mathcal{H} \leftarrow select(\mathcal{H}_{ij})$
```

The visual stereo SLAM experiment

Finally, we test CF SLAM in a 3D environment with a high density of features. The sensor is a Triclops camera carried in hand. The path consists of a loop inside the Rose Building at the University of Sydney. We obtain the 3D position of points from the computation of the dense stereo point cloud that corresponds to each SIFT feature. The experiment consists of 132 shoots, with a total of 6064 features. Fig. 2.13(a) shows the map obtained

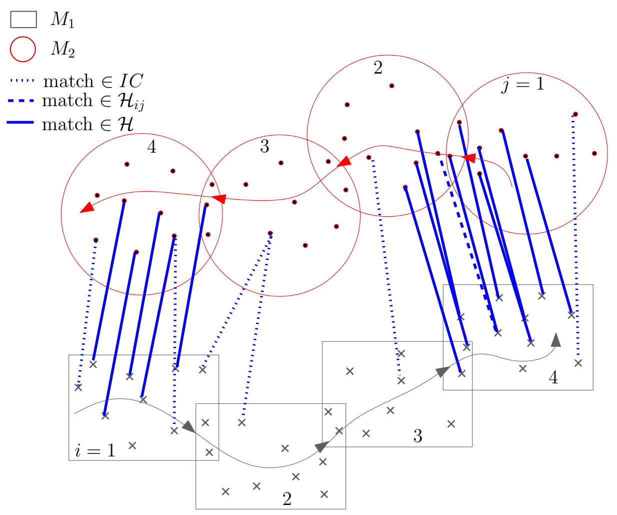


Figure 2.12: Computing the data association hypothesis from two local maps. The rectangles and circles show the minimum local maps of M_1 and M_2 respectively. All lines are potential matches, and form IC. The lines that are not dotted belong to a hypothesis between a pair of the minimum local maps, $\mathcal{H}_{ij}$. The solid lines represent the final hypothesis $\mathcal{H}$.

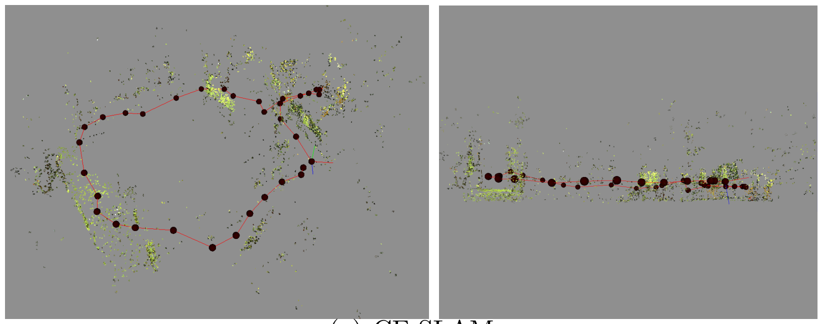
with CF SLAM. SLSJF obtains an incorrect map, Fig. 2.13(b). SLSJF is a sequential map joining algorithm, thus it is expected to provide less consistent results than D&C algorithms. In this experiments, this results in incorrect loop closure.

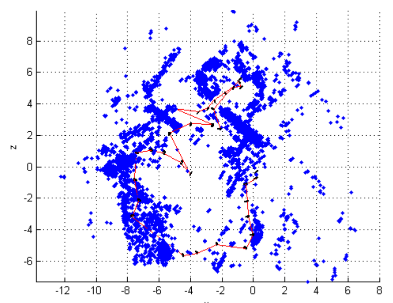
Both Map Joining SLAM and the D&C SLAM exceeded available memory in MATLAB. The data association is obtained with algorithm 3. Cumulative computational costs are shown in Fig. 2.13: for data association (c), for map updates (d) and total cumulative cost (e). CF SLAM clearly outperforms all the other algorithms.

2.8 Discussion

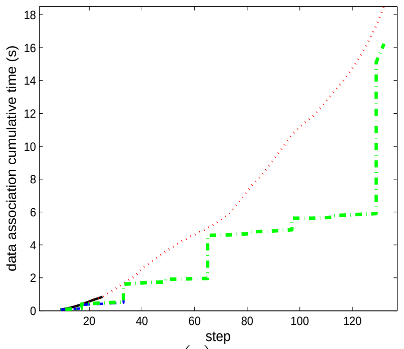
In this chapter we have described the Combined Filter SLAM algorithm, which is always more efficient than any other filtering algorithm for SLAM. It can execute in as low as $\mathcal{O}(\log n)$ per step. CF SLAM brings together the advantages of different methods that have been proposed to optimise EKF and EIF SLAM. There is no loss of information, because the solution is computed without approximations, except for linearisations. It is conceptually simple and easy to implement. There are no restrictions on the topology of the environment and trajectory, although, as it is the case in many SLAM algorithms, the computational efficiency will depend on this.

An important property of a SLAM algorithm is whether it is *on-line*, or provides the full vehicle and map estimation in every step, *delayed*, or providing a suboptimal estima-

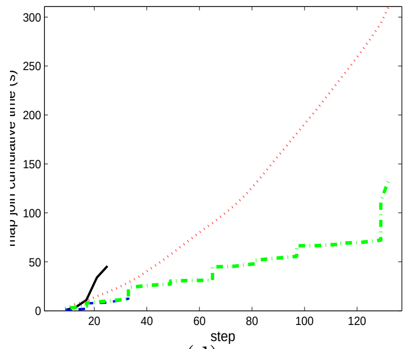

(a) CF SLAM



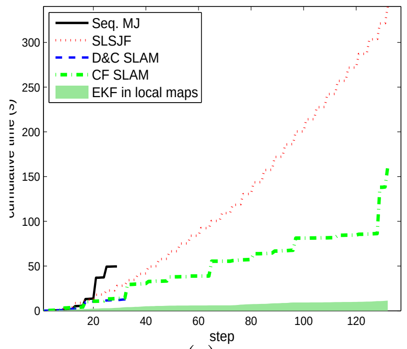
(b) SLSJF



(c)



(d)



(e)

Figure 2.13: Final 3D map using 132 shoots of dense stereo data from a Triclops Pointgrey camera with the CF SLAM (a). We show the cumulative cost for data association (c), map joining (d) and total cost including build local maps, map joining and data association (e). Map Joining SLAM and D&C SLAM exceed available memory capacity in shoots 24 and 32, respectively. The incorrect final map with SLSJF (b).

tion of the map and vehicle states at every step, but requiring additional computation in case the mission requires to have the best estimation, and finally *off-line*, or batch, carrying out the full computation only in case the vehicle and map states are required at a certain step. EKF and EIF SLAM, Map Joining, SLSJF, Treemap, CI-Graph, and iSAM are on-line; D&C SLAM and CF SLAM are delayed, and finally algorithms like Tectonic SAM are off-line. The selection of an appropriate algorithm for a specific mission must take this into account, it can be an overkill to use an on-line algorithm in applications where the map is only required at the end of the mission or very infrequently during the mission.

CF SLAM provides the robot with a local map with all the information needed for local navigation tasks for a very low computational cost. This allows using processor time for other important tasks, such as image processing and data association. This can be essential in situations of limited computational power such as space exploratory rovers. If at a certain moment the robot needs all the environment information with respect to a global reference, for instance to make decisions on global navigation like returning home, CF SLAM can provide the full map carrying out a single additional computation step for a cost as low as $\mathcal{O}(n)$. In these situations, the robot will have a precise map to switch from SLAM to navigation using an *a priori* map. We think that such orders are usually less frequent during a mission in many applications, specially in large scale. For applications requiring global knowledge of the map and vehicle position at every step CF SLAM may be less adequate than sequential algorithms like Map Joining SLAM, SLSJF, CI-Graph or iSAM, given the accumulated delay in computing the global map that CF SLAM incurs in. Finally, in applications where off-line processing is acceptable, we have shown that CF SLAM provides a solution in less time than any other filtering algorithm. Being a local mapping algorithm, the solution provided will have good consistency properties. Algorithms like Tectonic SAM that re-linearise the full problem might provide even more consistent solutions. This is an interesting issue to be investigated.

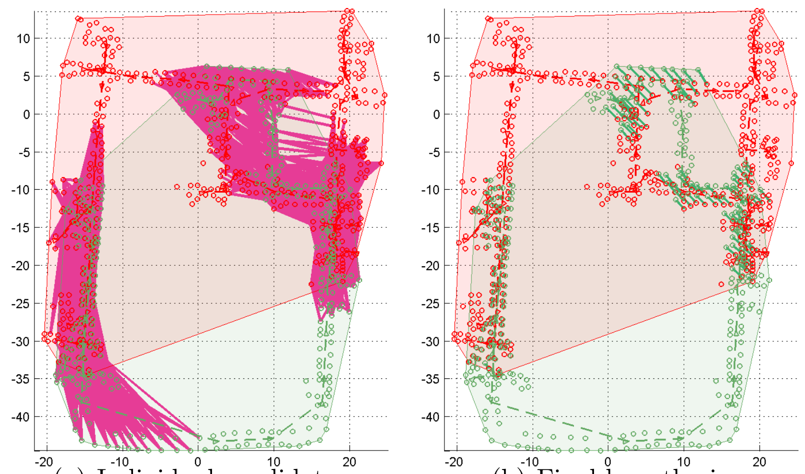
The estimation problem in the SLAM context is crucial and nowadays is well understood in the robotic community. We have provided the most efficient filtering algorithm, CF SLAM, for estimation in large scale up to date. Even more, the CF SLAM is in the same order, or better, of efficiency than other algorithms with different approaches (graphical models and particle filters) in the state of the art. Nevertheless, any estimation algorithm for S4LAM will be not able to work in real applications without a front-end algorithm in order to establish the correct data association. **In the next chapter of this thesis we will address that issue, the importance and the effect of the data association over computational efficiency and precision**.

Also, we have proposed two basic algorithms for data association, one using covariance based statistical tests and other appearance information. Here we have seen that the CF SLAM algorithm remains in the worst of case as computationally effective as D&C SLAM, the fastest to our knowledge that maintains the full covariance matrix and thus allows data association based on stochastic geometry, see Table 2.9. In the order of thousands of features, CF SLAM will outperform D&C SLAM because of reduced memory requirements. If appearance information is available for data association, CF SLAM outperforms all filtering algorithms discussed.

From our experiments it is clear that the greatest computational weight can lie in data association when the robot is revisiting a place after a large exploration period, closing a loops. Sequential SLAM algorithms are weak to solve the data association in those revisited places, see Figs. 2.11(c-d) and 2.13(b). CF SLAM is able to successfully handle those cases, see Fig. 2.14. Although, we obtain a good performance of the continuous data

	Cost per step		Total cost	
	Best	Worst	Best	Worst
seq. MJ	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$	$\mathcal{O}(n^3)$
SLSJF	$\mathcal{O}(n)$	$\mathcal{O}(\cancelto{n^3}{n^2})$	$\mathcal{O}(n^2)$	$\mathcal{O}(\cancelto{n^4}{n^3})$
D&C SLAM	$\mathcal{O}(n)$ amort.	$\mathcal{O}(n^2)$ amort.	$\mathcal{O}(n^2)$	$\mathcal{O}(n^3)$
CF SLAM	$\mathcal{O}(\log n)$ amort.	$\mathcal{O}(\cancelto{n^2}{n})$ amort.	$\mathcal{O}(n \log n)$	$\mathcal{O}(\cancelto{n^3}{n^2})$

Table 2.9: Computational costs for all filtering algorithms recovering columns of the covariance matrix, in the best case (pure exploration) and in the worst case (repeated traversal). Note that the CF SLAM is the most efficient in the best case and it is one of the best in the worst case.



(a) Individual candidates (b) Final hypothesis

Figure 2.14: Data association when there is a loop closure after an exploration for the DLR dataset. In (a) we show all de individual possible associations (magenta links) and in (b) the final association (green links) using algorithm 2.

association in the CF SLAM and D&C SLAM, it demands high computational resources, see high step in Fig. 2.11(e).

However, loops can be large enough so that estimates of vehicle location are not precise enough for these data association algorithms to be useful, or computational too demanding.

In the next chapter we investigate data association algorithms that do not make use of vehicle estimates and are applicable in very large environments.

Chapter 3

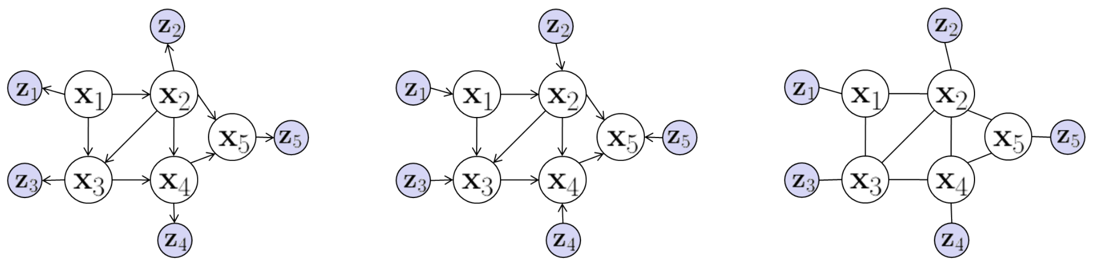
Robust Place Recognition

In this chapter we describe our contribution to one of the data association problem in SLAM known as place recognition. The approach proposed here attempts to detect previously visited places based on a process of inference using probabilistic graphical models. In section 3.1 we explain the technical and intuitive bases of probabilistic graphical models, in section 3.2 we detail how the conditional random fields are incorporated in the place recognition system. We describe the detailed method in section 3.2.1. In section 3.3 we compare the results from our system with state of the art in visual place recognition using benchmark datasets. Finally, we summarise the results and draw the fundamental conclusions of our contribution.

3.1 Preliminaries

Probabilistic graphical models have been used in the machine learning field to solve problems of classification or labelling. The problem of classification is closely related to the data association problem. In the classification process, the goal is to find the correspondences between a set of observed data and a set possible classes or states. In data association, we wish to find the correspondences between two sets of features observed at different times. This idea was first explored by Ramos et al. (2007) using as probabilistic graphical model the conditional random field (CRF) of Lafferty et al. (2001). The work of Ramos et al. (2007) inspire us to pursue in this way.

Probabilistic graphical models use a graph-based representation to model a joint probability distribution over a multi-dimensional space (of random variables) and a graph that is a compact representation of a set of independences that hold in the distribution. In our



(a) HMM: $p(\mathbf{x}, \mathbf{z})$ (b) MEMM: $p(\mathbf{x}|\mathbf{z})$ (c) CRF: $p(\mathbf{x}|\mathbf{z})$

Figure 3.1: Different probabilistic graphical models.

case, each node represents a random variable and each edge represents the probabilistic relationship between the connected variables.

Graphical models can be directed or undirected. In the *directed graph*, also named Bayesian Network, the edges have a particular direction and therefore impose a one-way causality. The *undirected graph*, also named Markov Random Field (MRF), does not have a preferred direction allowing causality both ways.

Graphical models can also classified in generative models and discriminative models. *Generative models* represent the join distribution over the states and the observations. *Discriminative models* describe a conditional distribution over the states given some observations.

In these categories we can find, see Fig. 3.1, Hidden Markov Models (HMM): a generative model with directed graph, Maximum Entropy Markov Models (MEMM): a discriminative model with directed graph, and Conditional Random Fields(CRF): a discriminative model with undirected graph.

These different probabilistic graphical models have their pros and cons (Koller and Friedman 2009). For instance, in terms of computational cost for learning, HMM and MEMM are better than CRF. In the richness of the features, MEMM and CRF are better than HMM. HMM and MEMM suffer from the label bias problem because of one-way causality, unlike CRF.

In the computer vision community, CRFs have been broadly used with success in different tasks: denoising images (Tappen et al. 2007), segmentation (He et al. 2004), object and gesture recognition (Quattoni et al. 2007), and stereo vision (Scharstein and Pal 2007). On the other hand, the robotic community has seen the advantages of this tool and it has been used in tasks like segmentation of 2D and 3D laser scans for mapping (Douillard et al. 2011; Lim and Suter 2007), clustering of dynamic objects (Tipaldi and Ramos 2009), and extracting places and activities from GPS traces (Liao et al. 2007).

3.1.1 Conditional Random Fields

Conditional random fields are probabilistic undirected graphical models first developed by Lafferty et al. (2001) for labelling sequence data. CRFs are a case of Markov Random Fields, and thus satisfy the Markov properties, where there is no need to model the distribution over the observations (Bishop 2006; Koller and Friedman 2009). If the neighbourhood of a node A (i.e. all nodes with edges to A) in the graph is known, the assignment to A is independent of the assignment to another node B outside the neighbourhood of A.

Instead of relying on Bayes' rule to estimate the distribution over hidden states $\mathbf{x}$ from observations $\mathbf{z}$, CRFs directly model $p(\mathbf{x}|\mathbf{z})$, the *conditional* distribution over the hidden variables given observations. Due to this structure, CRFs can handle arbitrary dependencies between the observations. This makes them substantially flexible when using complex and overlapped attributes or observations.

The nodes in a CRF represent hidden states, denoted $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n \rangle$, observations are denoted $\mathbf{z}$. The nodes $\mathbf{x}_i$, along with the connectivity structure represented by the undirected graph, define the conditional distribution $p(\mathbf{x}|\mathbf{z})$ over the hidden states $\mathbf{x}$. Let $\mathcal{C}$ be the set of cliques (fully connected subsets) in the graph of a CRF. Then, a CRF factorises the conditional distribution into a product of *clique potentials* $\phi_c(\mathbf{z}, \mathbf{x}_c)$, where every $c \in \mathcal{C}$ is a clique in the graph, and $\mathbf{z}$ and $\mathbf{x}_c$ are the observed data and the hidden nodes in such clique. Clique potentials are functions that map variable configurations to non-negative numbers. Intuitively, a potential captures the "compatibility" among the variables in the clique: the larger a potential value, the more likely the configuration. Using the clique potential, the conditional distribution over hidden states is written as:

$$p(\mathbf{x}|\mathbf{z}) = \frac{1}{Z(\mathbf{z})} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{z}, \mathbf{x}_c) \tag{3.1}$$

where $Z(\mathbf{z}) = \sum_{\mathbf{x}} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{z}, \mathbf{x}_c)$ is the normalising partition function. The computation of this function can be exponential in the size of $\mathbf{x}$. Hence, exact inference is possible for a limited class of CRF models only, e.g. in tree-structured graphs.

Potentials $\phi_c(\mathbf{z}, \mathbf{x}_c)$ are described by log-linear combinations of *feature functions* $\mathbf{f}_c$, i.e., the conditional distribution (3.1) can be rewritten as:

$$p(\mathbf{x}|\mathbf{z}) = \frac{1}{Z(\mathbf{z})} \exp \left\{ \sum_{c \in \mathcal{C}} \mathbf{w}_c^T \cdot \mathbf{f}_c(\mathbf{z}, \mathbf{x}_c) \right\} \tag{3.2}$$

where $\mathbf{w}_c$ a weight vector, which represents the importance of different features for correctly identifying the hidden states. Weights can be learned from labelled training data.

Inference

The process of *inference* allows answering queries using the graphical model as our model of the problem, and it is required for training (learning) and decoding (Cohn 2007; Koller and Friedman 2009). Inference in a CRF estimates the marginal distribution of each hidden variable $\mathbf{x}_i$, and can thus determine the most likely configuration of the hidden variables $\mathbf{x}$ (i.e., the maximum a posteriori, or MAP, estimation). Both tasks can be solved using *belief propagation* (BP) (Pearl 1988), which works by transmitting messages containing beliefs through the graph structure of the model. Each node sends messages to its neighbours based on messages it receives and the clique potentials. BP generates exact results in graphs with no loops, such as trees or polytrees. For cyclic graphs, approximate inference methods like loopy belief propagation can be used (Murphy et al. 1999).

Parameter learning

The goal of parameter learning is to determine the weights of the feature functions used in the conditional likelihood (3.2) for fitting the model to training data. CRFs learn these weights discriminatively by maximising the conditional likelihood of labelled training data. We resort to maximising the *pseudo-likelihood* of the training data, which is given by the product of all local likelihoods $p(\mathbf{x}_i|\text{MB}(\mathbf{x}_i))$; where $\text{MB}(\mathbf{x}_i)$ is the Markov Blanket of variable $\mathbf{x}_i$, which contains the immediate neighbours of $\mathbf{x}_i$ in the CRF graph. Optimisation of this pseudo-likelihood is performed by minimising the negative of its log, resulting in the following objective function:

$$L(\mathbf{w}) = -\sum_{i=1}^{n} \log p(\mathbf{x}_i|\text{MB}(\mathbf{x}_i), \mathbf{w}) + \frac{\mathbf{w}^T\mathbf{w}}{2\sigma_{\mathbf{w}}^2} \tag{3.3}$$

The rightmost term in (3.3) serves as a zero-mean Gaussian prior, with variance $\sigma_{\mathbf{w}}^2$, on each component of the weight vector, and is used to avoid overfitting.

3.1.2 Data Association with CRFs

In the context of our interest, data association in mobile robots, perhaps the first application of CRFs was introduced by Ramos et al. (2007), called CRF-Matching, for matching

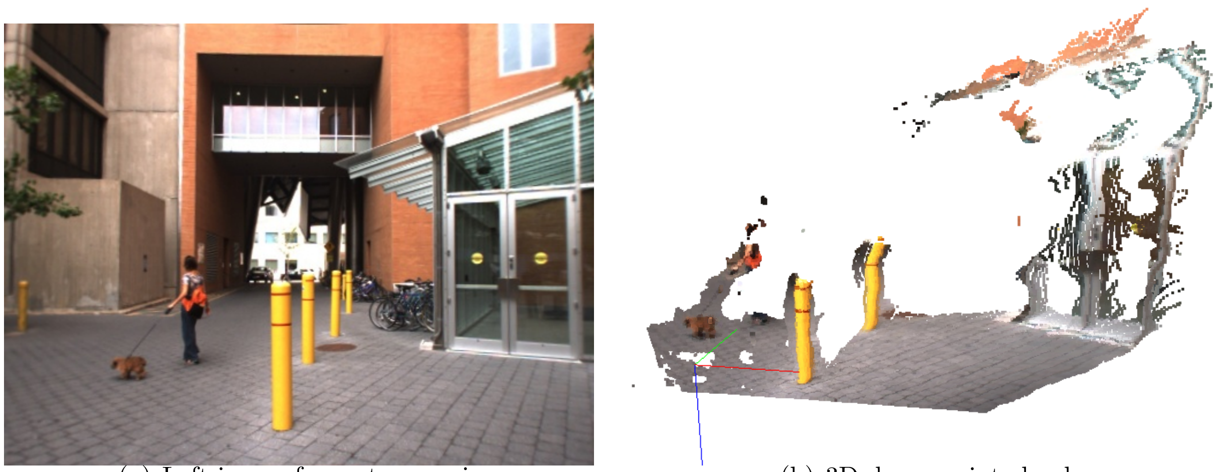

(a) Left image from stereo pair (b) 3D dense point cloud

Figure 3.2: Scene from an outdoor environment. We use only an image from the stereo pair for the appearance information 3.2(a), and the 3D dense point cloud 3.2(b).

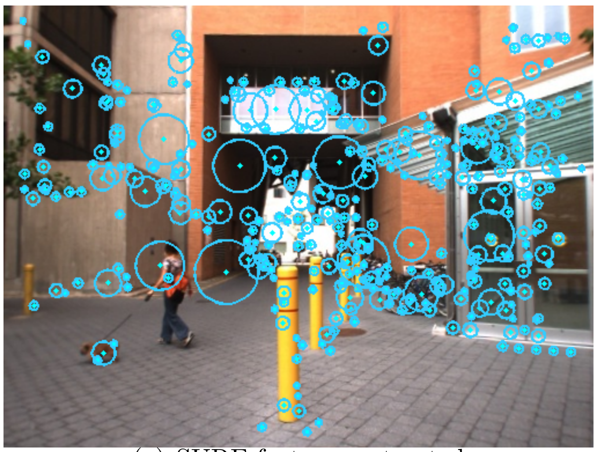
2D laser scans. The idea of CRF-Matching is take two sets of points, from different times of acquisition, and find their point-to-point correspondences. Ramos et al. (2007) show that CRF-Matching is much more robust to high rotational and translational movements of the robot than the classical Iterative Closest Point algorithm. Later, Ramos et al. (2008) extended the idea to associate image features , using a 2D Delaunay triangulation as graph structure, and then a loopy belief propagation method for inference.

From now on, we focus our experimentation with stereo cameras. The use of a stereo camera allows us to combine appearance information with 3D metric information when available, see Fig. 3.2.

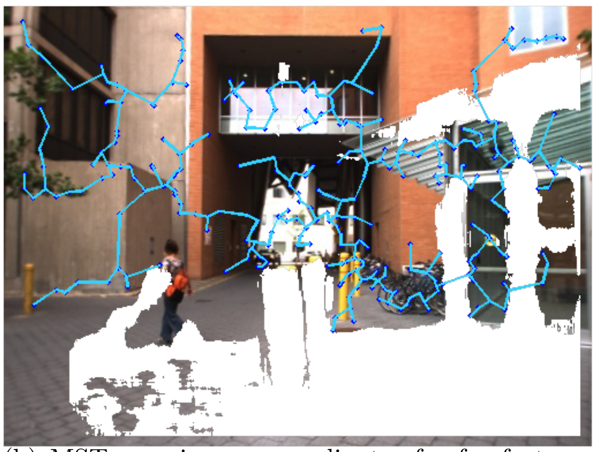
Our contribution here is the extension of CRF-Matching to 3D point clouds. It includes two keys ideas:

1. Include only characteristic points as nodes in the graph and not the thousands provided by the stereo. Lim and Suter (2007) use CRFs and sub-sample the 3D laser data with an adaptive data reduction based on spatial properties in order to reduce both learning and inference times. We take advantage of texture in visual information to sub-sample the 3D dense information and consider only salient visual features and their coverage areas. We use the extractor SURF (Bay et al. 2006) on one of the images for that purpose, see Fig. 3.3(a).

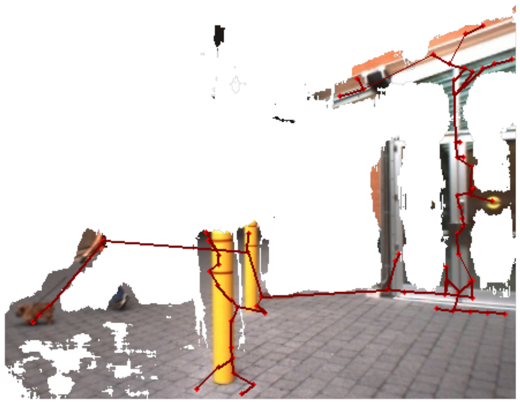
2. Use a Minimum Spanning Tree as graph structure for our CRF model, instead of the dense Delaunay triangulation. This idea was previously used by Quattoni et al. (2007) in the context of object classification in images. In that work, equivalent classification performance was shown for MSTs in comparison with more densely



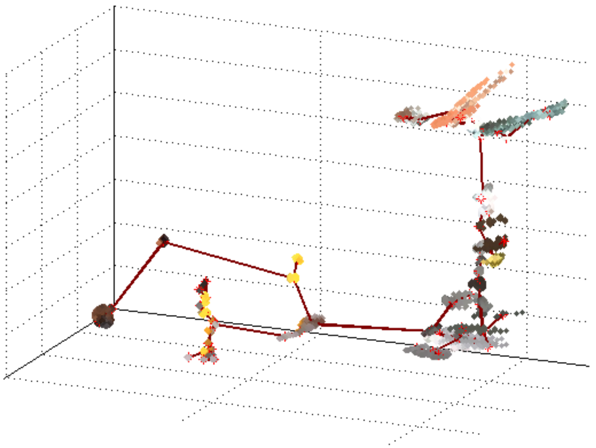
(a) SURF features extracted



(b) MST over images coordinates for far features ($\mathcal{G}_{Im}$)



(c) MST over 3D metric coordinates for near features ($\mathcal{G}_{3D}$)



(d) 3D visualitation of $\mathcal{G}_{3D}$

Figure 3.3: In each scene we get the SURF features over one image of the stereo pair 3.3(a), and compute the two minimum spanning trees: one for features with 3D information (near features), and the other for the remaining ones (far features). On 3.3(b), we show the graph for far features ($\mathcal{G}_{Im}$) in blue, in dark red the graph for near features ($\mathcal{G}_{3D}$) on 3.3(c). We apply the CRF-Matching over both graphs. The minimum spanning tree of $\mathcal{G}_{3D}$ is computed according to the metric coordinates, here projected over the images only for visualisation. On 3.3(d), we show $\mathcal{G}_{3D}$ in metric coordinates with the 3D point cloud (textured) of each vertex in the tree. The MST gives us an idea of the dependencies between features in a scene, and enforce the consistency of the features association between scenes.

connected graphs. By definition, the minimum spanning tree connects points that are close in the measurement space, highlighting intrinsic localities in the scene. This implies: first, the associations are jointly compatible within neighbourhoods, and second, the compatibility is enforced and propagated from neighbourhood to neighbourhood by the edges between them. In addition, trees allow exact inference algorithms, as compared, for instance, with loopy belief propagation for cyclic graphs, which is approximate and more expensive. As Quattoni et al. (2007), our results show that MSTs properly encode connections between the hidden variables and ensures global consistency in both, image and 3D.

In our framework of data association, the hidden states correspond to all the possible associations between the n features in scene A and the m features in scene B, i.e. $\mathbf{x}_i \in \{0, 1, 2, \ldots, m\}$, where the additional state 0 is the outlier state. Observations are provided by the sensors, e.g. 3D point clouds, appearance descriptors, or any combination of them.

The data association process is done by inferring on the matching between SURF-points of the scenes. Here we refer to SURF-point as the pixel in the image where the SURF-feature was detected.

We model the scene as two graphs: the first graph ($\mathcal{G}_{3D}$) models the near objects, i.e. those pixels with sufficient disparity from the stereo, and hence with 3D information, see Fig. 3.3(c) and 3.3(d). And the second graph ($\mathcal{G}_{Im}$) models the far objects from pixels without disparity information, see Fig. 3.3(b). The nodes of the graphs are the SURF-points extracted before, and the edges of the graphs result from computing the minimum spanning tree (MST), according to the Euclidean distances between the pixel coordinates in the case of $\mathcal{G}_{Im}$, and between the 3D metric coordinates in the case of $\mathcal{G}_{3D}$.

Feature description

With the graph structure defined for our CRF models, we have to define $\mathbf{f}_c(\mathbf{z}, \mathbf{x}_c)$ in eq. 3.2. The CRF matcher can employ arbitrary local features to describe shape, image properties, or any particular aspect of the data. Our features describe *differences* between shape (only for $\mathcal{G}_{3D}$) and appearance (for both graphs) of the features. The local features we use are the following:

Shape difference: These features capture how much the local shape of dense stereo data differs for each possible association. We use the geodesic, PCA and curvature distance.

The *geodesic distance*, defined as the sum of Euclidean distances between points in the minimum spanning tree, provides information about the density of the neighbourhood of each node of the graph. It can be calculated for different neighbourhoods representing local or long-term shape information. Given points $z_{A,i}$, $z_{B,j}$ and a neighbourhood k, the

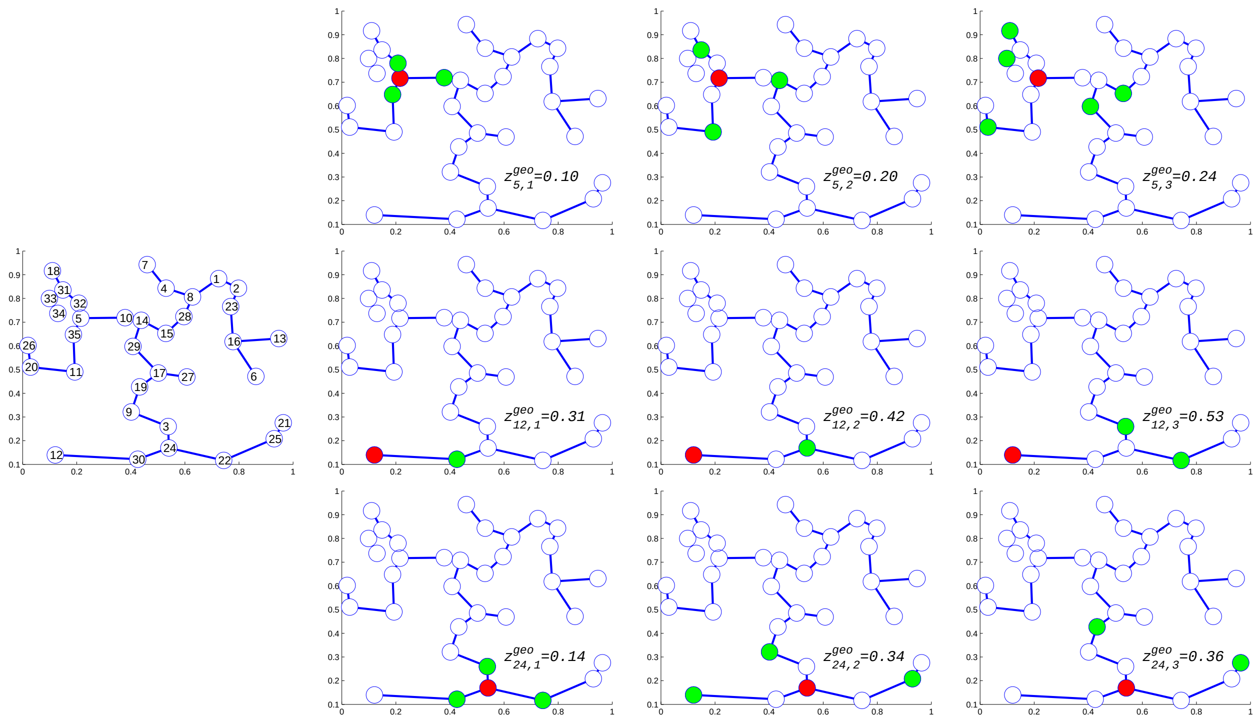


Figure 3.4: Geodesic distances in a randomly generated minimum spanning tree. Columns correspond to neighbourhoods of order 1, 2 and 3, respectively, for three different nodes (rows).

geodesic distance feature is computed as:

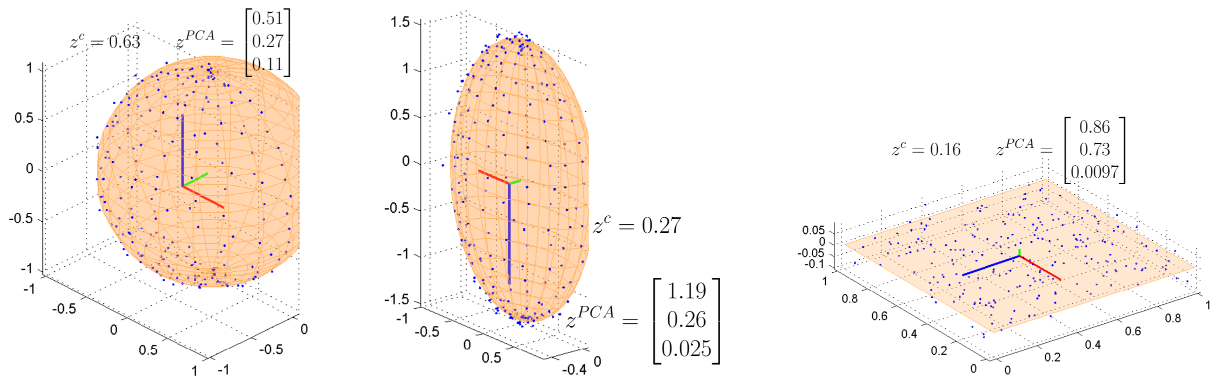
$$\mathbf{f}_{geo}(i,j,k,z_A,z_B) = \left\| z^{geo}_{A,i,k} - z^{geo}_{B,j,k} \right\| = \left\| \sum_{l=i}^{i+k-1} \| z_{A,l+1} - z_{A,l} \| - \sum_{l=j}^{j+k-1} \| z_{B,l+1} - z_{B,l} \| \right\| \tag{3.4}$$

where i and j correspond to the hidden state $\mathbf{x}_i$ that associate the feature i of the scene A with the feature j of the scene B. The neighbourhood k of $\mathbf{x}_i$ in the graph corresponds to all the nodes separated k nodes from $\mathbf{x}_i$. In our implementation, this feature is computed for $k \in \{1,2,3\}$. A similar feature is used to match 3D laser scans in (Anguelov et al. 2005).

We also use *Principal Component Analysis* over the dense 3D point cloud that is contained within a radius given by the scale obtained by the SURF extractor for each node in the graph, textured points in Fig. 3.3(d). The *PCA distance* is computed as the absolute difference between principal components of a dense point cloud $z^{pca}_{A,i}$ in scene A and $z^{pca}_{B,j}$ in scene B:

$$\mathbf{f}_{PCA}(i,j,z^{pca}_A,z^{pca}_B) = \left| z^{pca}_{A,i} - z^{pca}_{B,j} \right| \tag{3.5}$$

Another way to consider local shape is by computing the *difference between the cur-*



(a) Sphere (b) Ellipsoid (c) Plane

Figure 3.5: Synthetic examples of point clouds in a node of the graph. These point clouds are used in the PCA and curvature distances. The synthetic shapes are with additive gaussian noise.

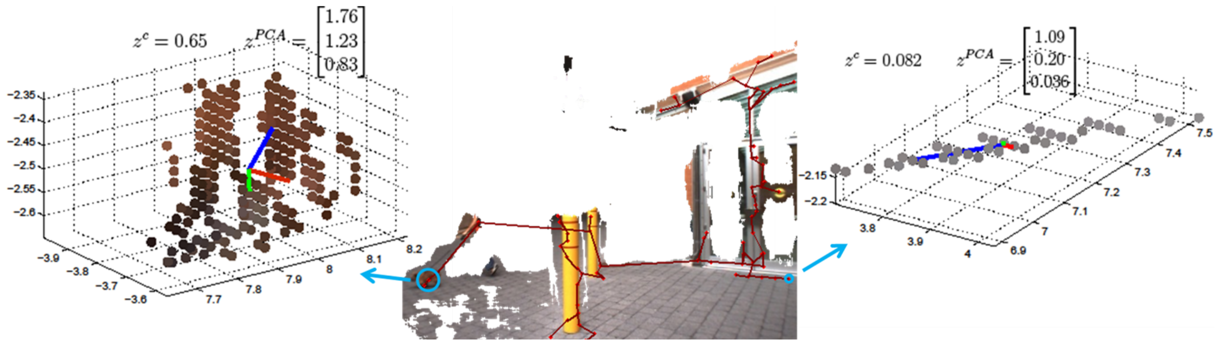


Figure 3.6: Real examples of point clouds in nodes of the graph from the scene in Fig. 3.2.

vatures of the dense point clouds. This feature is computed as:

$$\mathbf{f}_{curv}(i, j, z^c_A, z^c_B) = \left\| z^c_{A,i} - z^c_{B,j} \right\| \tag{3.6}$$

where $z^c = \frac{3s_3}{s_1+s_2+s_3}$, and $s_1 \geq s_2 \geq s_3$ are the *singular values* of the point cloud of each node.

Visual appearance: These features capture how much the local appearance from the points in the image differs for each possible association. We use the *SURF distance*. This feature calculates the Euclidean distance between the descriptor vectors for each possible association:

$$\mathbf{f}_{SURF}(i, j, z^{descr}_A, z^{descr}_B) = \left\| z^{descr}_{A,i} - z^{descr}_{B,j} \right\| \tag{3.7}$$

Ramos et al. (2008) also include as features the distances between the individual dimensions of the descriptor space. In our training and validations data we do not find a significant improvement in the accuracy of the labelling, and this greatly increases the size of the weight vector.

All previous features described are unary, in that they only depend on a single hidden state i in scene A (indices j and k in the features denote nodes in scene B and neighbourhood size). In order to generate mutually *consistent* associations it is necessary to define features, over the cliques, that relate the hidden states in the CRF with each other.

Pairwise distance: This feature measures the consistency between the associations of *two* hidden states $\mathbf{x}_i$ and $\mathbf{x}_j$ and observations $z_{A,i}$, $z_{A,j}$ from scene A and observations $z_{B,k}$ and $z_{B,l}$ in scene B:

$$\mathbf{f}_{pair}(i, j, k, l, z_A, z_B) = \quad \| \| z_{A,i} - z_{A,j} \| - \| z_{B,k} - z_{B,l} \| \| \tag{3.8}$$

The z_A and z_B are in metric coordinates for $\mathcal{G}_{3D}$, and in pixels for $\mathcal{G}_{Im}$.

In addition to the features described, we define two more binary features for the outlier state, just as Ramos et al. (2007), in order to handle features from scene A that do not have a correspondence in scene B, then their hidden sate must be set to "outlier". One feature function is local and one for pairwise. Whenever the outlier feature is true, the value of all corresponding features are set to zero. In the learning process the algorithm learns weights for the binary outlier features.

Learning weights

The learning phase of our CRF-Matching we is as follows:

1. Capture a sequence of stereo images.
2. Randomly choose a subset stereo images, in 200 frames k, uniformly distributed over the experiment.
3. Label each frame k using the scene in the frame $k-1$ with data associations based on RANSAC over the best body rigid transformation for the SURF-points with 3D information, and over the fundamental matrix for the remaining SURF-points in the image.
4. Build in each scene k the graph structures with the MST: one over the euclidean distances in metric space for $\mathcal{G}_{3D}$, and another over pixel coordinates for $\mathcal{G}_{Im}$.
5. Compute the features defined in the previous section for each graph.

6. Learn the weights of the CRF models using the pseudo-likelihood over the 60% of the subset chosen.

7. Validate the learning process with the remaining 40%.

We show the performance of the data association with two different datasets. The first is taken from the RAWSEEDS (2009) Project. This project contains stereo sequences from two different campuses, the campus Bicocca of the University of Milano and the campus Bovisa of the Politecnico di Milano, taken with a Videre stereo camera. The scenes correspond to indoors and outdoors, and a mix of them, in both static and dynamic environments. We choose for the learning and validation process the experiment *Bovisa_2008-09-01_Static* taken on the September first of 2008 in the campus Bovisa with mix of indoors and outdoors in a static environment. In Fig. 3.7 we can see some sample images of the subset for learning. We have used the data corresponding to the Stereo Vision System with an 18cm baseline. Images (640x480 px) are taken at 15 fps.

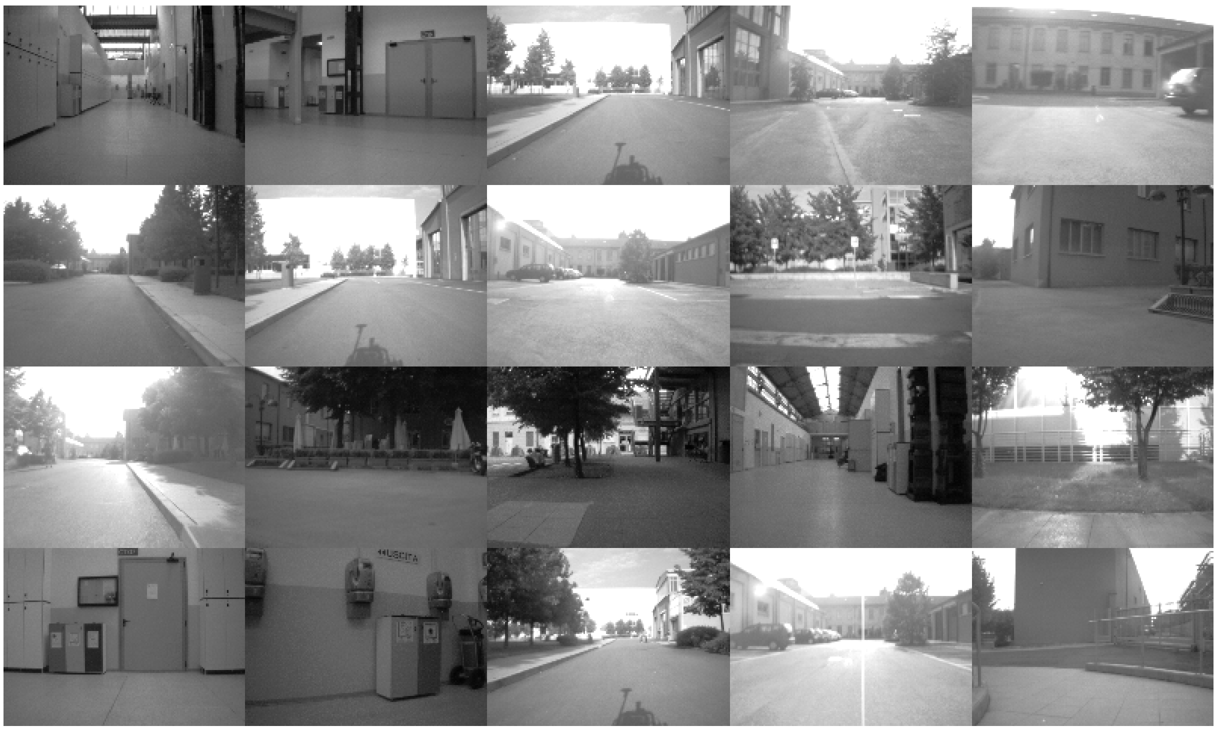


Figure 3.7: Sample of right images from stereo pair of the RAWSEEDS dataset used for learning.

Table 3.1 shows the weights as a result of the learning process with the mixed static dataset. The weights are learned for both graphs.

The accuracy of the learning process is computed in the subsets for training and for validation, see Table 3.2. We remark here that the process of labelling with RANSAC is not perfect, sometimes you can obtain mistakes in the labels. But the learning process has showed be robust against that error, mainly due to the a majority of good labels.

From the Table. 3.2 we can see that the model is not over-fitted to the subset of training.

		SURF	*geo*			*PCA*			*curv*	*pair*	*outlier*	
			k=1	k=2	k=3	1^{st}	2^{nd}	3^{rd}			*local*	*pair*
$\mathcal{G}_{3D}$	**w**	-8.55	0.15	0.80	1.09	0.16	-1.34	0.06	-0.08	-4.60	-1.24	-0.83
$\mathcal{G}_{Im}$	**w**	-15.82								-18.69	-2.27	-0.57

Table 3.1: Weights obtained in the learning process in the RAWSEEDS dataset. A higher value in magnitude has more importance in eq. 3.2

	$\mathcal{G}_{3D}$		$\mathcal{G}_{Im}$	
	Training set	Validation set	Training set	Validation set
Accuracy	86.26%	85.00%	79.25%	80.31%

Table 3.2: Statistics of the learning process in the RAWSEEDS dataset.

The second dataset used was taken in the MIT campus in multiple sessions around of the Stata Center building, with indoor and outdoor routes taken on July of 2010. The stereo images were collected with a BumbleBee2, from PointGrey, with an 8cm baseline. The environment is dynamic, with people and car in movement. We choose for the learning and validation process 200 images (512x384 px) uniformly distributed in time from an indoor session taken on the April 8th of 2010. In Fig. 3.8 we can see some sample images of the chosen subset.

Table 3.3 shows the weights as result of the learning process with the indoor dataset in the Stata Center building. The weights are learned for both graphs.

From the Table. 3.4 we can see that the model is not over-fitted to the subset of training.

		SURF	*geo*			*PCA*			*pair*	*outlier*	
			k=1	k=2	k=3	1^{st}	2^{nd}	3^{rd}		*local*	*pair*
$\mathcal{G}_{3D}$	**w**	-15.28	-1.07	-0.51	-0.81	-0.26	-3.56	-0.03	-17.44	-2.13	-1.11
$\mathcal{G}_{Im}$	**w**	-19.56							-12.63	-1.77	-0.74

Table 3.3: Weights obtained in the learning process in the Stata Center indoors session.

In the Fig. 3.9 we show a typical result of the data association between sequential scenes with the parameter previously learned over a Stata Center outdoors session taken on July 19th of 2010. The process take $\approx 0.3s$ in average for both graphs (including the SURF extraction).

Figure 3.8: Sample of left images from stereo pair of a Stata Center indoors session used for learning.

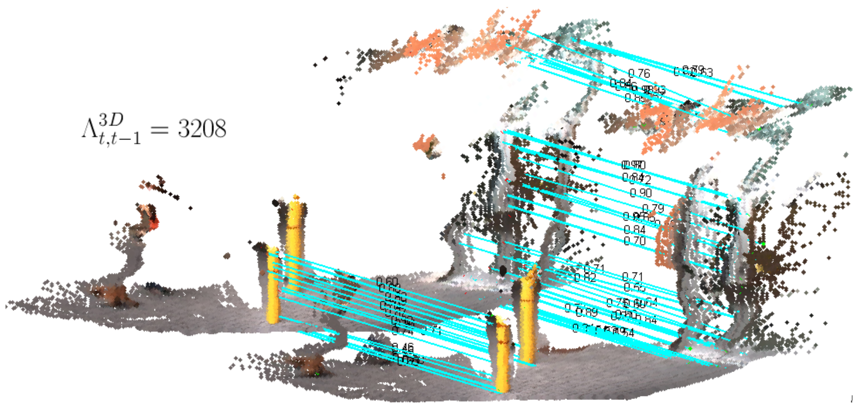
	$\mathcal{G}_{3D}$		$\mathcal{G}_{Im}$	
	Training set	Validation set	Training set	Validation set
Accuracy	85.76%	81.69%	72.17%	74.37%

Table 3.4: Statistics of the learning process in the Stata Center indoors session.

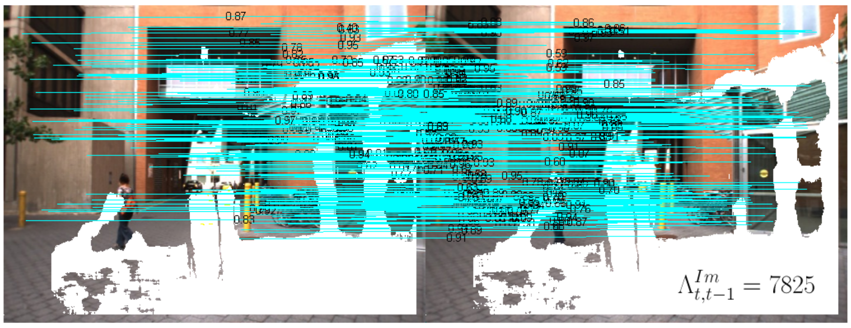
3.2 Using CRFs for Place Recognition

Up to now, we have seen how to find the correspondences between two scenes, in both 3D and image, with CRF-Matching. This tool is very powerful, but has its advantages and limitations. The main advantage is that allows us carry out robust similarity comparisons only using the information provide by the sensor. The limitation is the computational cost: two scenes can be compared in constant time; the cost of finding similar scenes in a sequence with n scenes will be $\mathcal{O}(n)$. But it is very convenient for a robust evaluation of candidates to loop closing.

With this issue in mind, we propose to solve the place recognition problem by using two complementary techniques. The *first* one is based on the bag-of-words method (BoW) of Sivic and Zisserman (2003), which reduces images to sparse numerical vectors by quantising their local features. This enables quick comparisons among a set of images to find



(a) CRF-Matching over 3D information



(b) CRF-Matching over image information

Figure 3.9: Data association results with CRF-Matching for each graph, $\mathcal{G}_{3D}$ in (a) and $\mathcal{G}_{Im}$ in (b). The scenes have one second of separation. In both we show the individual probabilities of associations and the negative log-likelihood for the maximum-a-posteriori Λ.

those which are similar. We rely on the efficient technique of detection of *candidates* to loop closing developed and implemented by Dorian Gálvez-López [1]. Here, we only give a short description of his implementation and its novelties.

After candidates generation, only the unclear loop closure candidates are *verified* by matching the scenes with CRFs. We also propose accepting the loop closure candidates based on a normalised similarity score in terms of the likelihoods of the matched scenes with respect to recent images. Ramos et al. (2007) proposed the possibility of detecting loop closures with CRF by taking the maximum log-likelihood of the match between the current and all previous scans. Comparing the current location against all the previous ones is impractical in real applications. Furthermore, their metric does not provide a way to distinguish between true and false loop closures.

Our basic idea is to exploit the efficiency of BoW for detecting candidate revisited places in real-time, and in unclear cases use the robustness of CRF-Matching to verify that candidate matches are correct.

[1]Available at http://webdiis.unizar.es/˜dorian

3.2.1 Method

Our place recognition system can be summarised in algorithm 4.

Algorithm 4 Place recognition system

```
Input: Scene at time $t$, Database $\langle 1, \dots, t-1 \rangle$
Output: Time $t'$ of the revisited place, or null
  $Output = Null$
  Search the bag-of-words database for the best matching scene at $t'$ with score $\eta_c(t, t')$
  if $[t-1, t'_1], ..., [t-\tau_l, t'_{\tau_l}]$ matched and $|t'_i - t'_j| \leq \tau_d$ then
    {Loop candidate detected}
    if $\eta_c(t, t') \geq \alpha^+$ then
      $Output = t'$        {Accepted}
    else
      if $\eta_c(t, t') \geq \alpha^-$ then
        {Loop candidate verification}
        Build $\mathcal{G}_{3D}$ and $\mathcal{G}_{Im}$
        Infer with CRFs and compute the scores $\eta_{\mathcal{G}}$
        if $\eta_{3D}(t, t') \leq \beta_{3D} \wedge \eta_{Im}(t, t') \leq \beta_{Im}$ then
          $Output = t'$        {Accepted}
        end if
      end if
    end if
  end if
  Add current scene to the Database
```

Loop Closing Detection

The first component is based on the bag-of-words method (BoW) of Sivic and Zisserman (2003) which is implemented in a hierarchical way, thus improving efficiency (Nister and Stewenius 2006). In this implementation we use 64-SURF-features, see Fig. 3.3(a) with the novelties proposed by Gálvez-López in (Cadena et al. 2011a):

Normalised similarity score Representing images as numerical vectors is very convenient since it allows performing really quick comparisons between images. Given two vectors v and w, their similarity is measured as the score $s(v, w)$:

$$s(v, w) = 1 - \frac{1}{2}\left\| \frac{v}{||v||} - \frac{w}{||w||} \right\| \tag{3.9}$$

where $||.||$ stands for the L_1-norm. Note that this score is 0 when there is no similarity at all, and 1 when both vectors are the same.

In the special case where the acquired data are sequential the vectors v and w are associated to instants of time t and t'. It is expected to have lots of very similar images in our problem, since they are collected close in time. This may make the matched vector with highest score s be incorrect in many cases. It is desirable to distinguish those cases, but the range the score s varies is very dependent on the query image. For these reasons, a metric of similarity is defined, the *normalised similarity score*, as:

$$\eta_c(t, t') = \frac{s(v_t, w_{t'})}{s(v_t, v_{t-\gamma})} \tag{3.10}$$

The score obtained is normalised from a match between v_t and $w_{t'}$ with the expected score for the query vector v_t. The expected value for v_t is the score obtained when comparing it against a very similar vector. Here, the vector obtained $\gamma = 1s$ ago. The higher η_c is, the more similar two images are. Unlike score s, values of the normalised score η_c from different images can be directly compared. Note that the normalised similarity score can be defined for any similarity score.

Temporal consistency To detect loops, a temporal constraint is imposed. A loop candidate between images at time t and t_0 is detected if there are matches $< v_t, w_{t_0} >$, $< v_{t-1}, w_{t_1} >$, ..., for a short time interval $\tau_l = 4$ seconds, and the timestamps t_0, t_1, ..., are close (i.e. their difference is within $\tau_d = 2$ seconds). These temporal values were selected according to the frequency of image sequences, and the expected reliability of the method.

Double threshold Finally, the match $< v_t, w_{t_0} >$, with normalised score $\eta_c(t, t_0)$, is checked by a *double threshold* (α^-, α^+) in order to be accepted as a loop closure. Then, there are three possibilities:

1. if $\eta_c(t, t_0) \geq \alpha^+$ the match is considered highly reliable and accepted;

2. if $\alpha^- < \eta_c(t, t_0) < \alpha^+$ the match is checked by CRF-Matching in the next step of verification.

3. if $\eta_c(t, t_0) \leq \alpha^-$ the match is ignored.

Loop Closing Verification

We use the CRF matcher stage over the loop closing candidates provided by the BoW stage. We compute the negative log-likelihood (Λ) from the MAP associations between the scene in time t, against the loop closing candidate in time t', $\Lambda_{t,t'}$, and the scene in $t-\gamma$, $\Lambda_{t,t-\gamma}$, $\gamma = 1s$.

The negative log-likelihood Λ^{3D} of the MAP association for $\mathcal{G}_{3D}$ provides a measure of how similar two scenes are in terms of close range, and Λ^{Im} for $\mathcal{G}_{Im}$ in terms of far range. Thus, in order to compare how similar the current scene is with the scene in t', $\Lambda_{t,t'}$, with respect to how similar the current scene is with the scene in $t-\gamma$, $\Lambda_{t,t-\gamma}$, we use again a normalised similarity score as:

$$\eta_{\mathcal{G}} = \frac{\Lambda^{\mathcal{G}}_{t,t'}}{\Lambda^{\mathcal{G}}_{t,t-\gamma}} \tag{3.11}$$

where $\mathcal{G}$ indicates the graph.

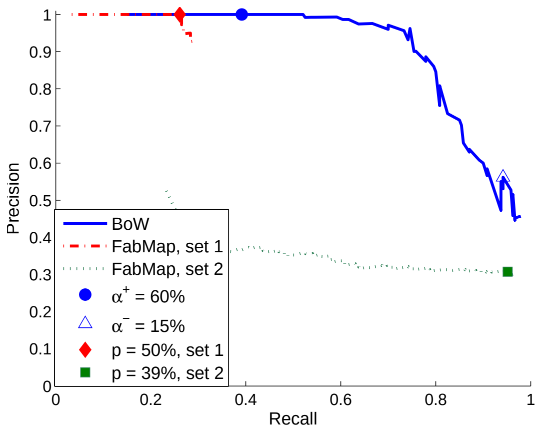
Score $\eta_{\mathcal{G}}$ is compared to $\beta_{\mathcal{G}}$, a control parameter of the level of similarity we demand for $(t, t-\gamma)$, where a smaller β means a higher demand. By choosing different parameters for near and far information we can make a balance between the weight of each in our acceptance.

3.3 Experiments

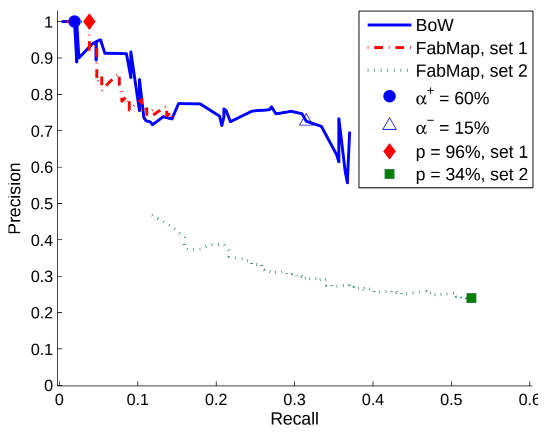
We tested place recognition system in three datasets from the RAWSEEDS Project: static indoors, static outdoors and dynamic mixed. These three and the training datasets were collected on different dates and in two different campuses. Please refer to the RAWSEEDS (2009) Project for more details. We use the weights learned from the training dataset, see table 3.1.

For the first bag-of-words stage, we have to set the minimum confidence expected for a loop closure candidate, α^-, and the minimum confidence for a trusted loop closure, α^+. We selected the working values $\alpha^- = 0.15$ and $\alpha^+ = 0.60$. We have set these parameters by observing their effect on the precision-recall curves achieved by the BoW algorithm on its own in the indoor and outdoor datasets tested (see Fig. 3.10). Since these datasets are fairly heterogeneous, we think these values can work well in many situations. It might depend on the vocabulary size, though.

For the second CRF matcher stage, we work on the set-point given by the α parameters



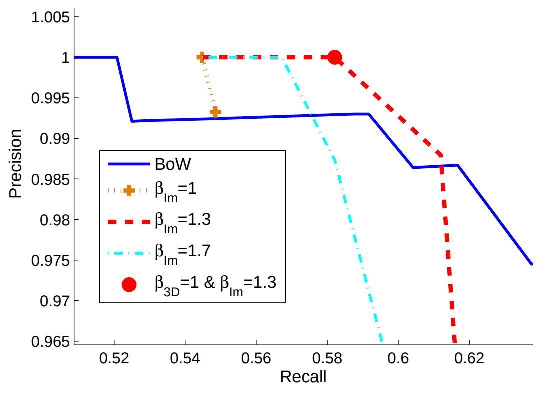
(a) Indoor



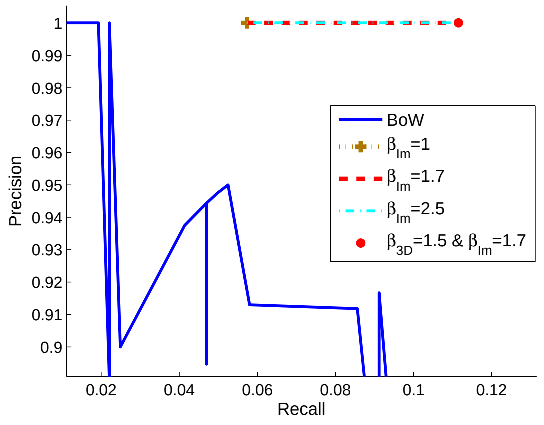
(b) Outdoor

Figure 3.10: Precision and Recall in the (a) indoor dataset and (b) outdoor dataset, without CRF verification or epipolar constraint.

selected. Then, by sweeping the β parameters we obtain the precision-recall curves of the Fig. 3.11. Note the scale and the improvements with respect to the BoW precision-recall curve. We set the β parameters in order to obtain 100% precision, see below. We do this to compare recall with the state of the art. We use the parameters obtained for the outdoor dataset also in the mixed dataset. All the parameters used are shown in table 3.5.



(a) Indoor



(b) Outdoor

Figure 3.11: Precision and Recall in (a) the indoors dataset and (b) outdoors dataset with CRF verification. Each curve is computed for a constant value of β_{Im} and sweeping of β_{3D} parameter between $[0, 3]$. We let the BoW precision-recall curve only for illustrating the improvements with the CRF stage of verification.

We have compared the results from our system against the state-of-the-art technique FAB-MAP 2.0 of Cummins and Newman (2010). The FAB-MAP software[2] provides some predefined vocabularies. We have used the FAB-MAP indoor vocabulary for our indoor

[2]The software and vocabularies were downloaded from `http://www.robots.ox.ac.uk/~mobile/`

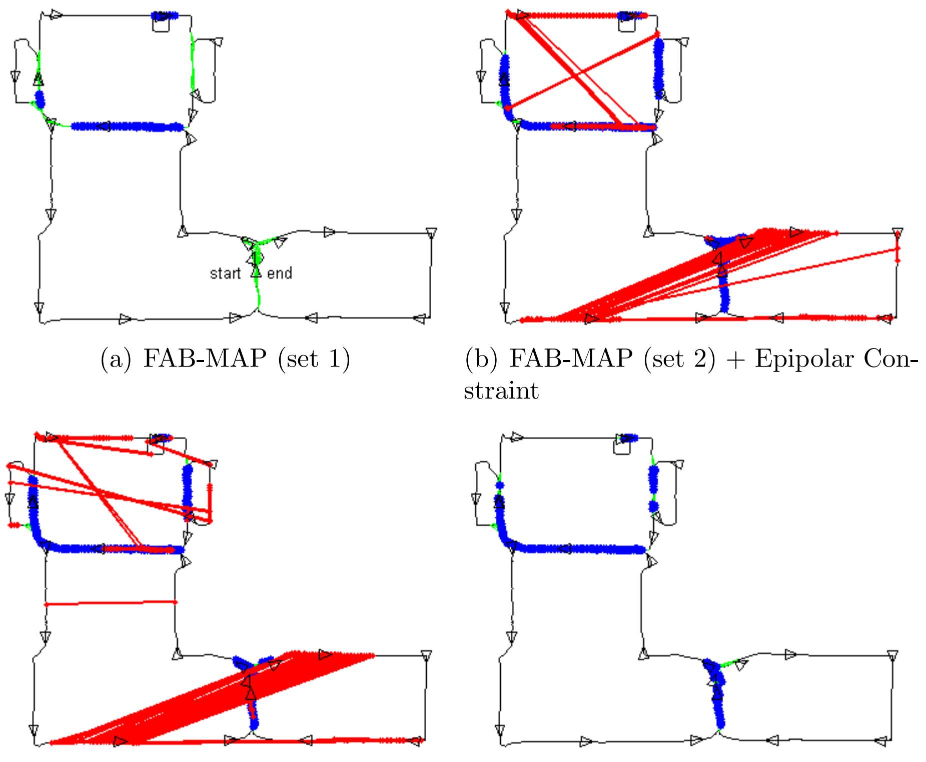
dataset and the FAB-MAP outdoor vocabulary for our mixed and outdoor datasets. This technique has a set of parameters to tune in order to obtain the best performance in each experiment. The parameters that we have modified are the following ones (for further details please see (Cummins and Newman 2008) and (Cummins and Newman 2010)):

- p: Probability threshold. The minimum matching probability required to accept that two images were generated at the same place.
- $P(\text{obs}|\text{exist})$: True positive rate of the sensor. Prior probability of detecting a feature given that it exists in the location.
- $P(\text{obs}|\neg\text{exist})$: False positive rate of the sensor. Prior probability of detecting a feature given that it does not exist in the location.
- $P(\text{newplace})$: Probability for new place. Prior probability that the last image is a new place.
- σ: Likelihood smoothing factor. Factor for smoothing the likelihood values through consecutive places.
- *Motion Model*: Model Motion Prior. This biases the matching probabilities according to the expected motion of the robot. A value of 1.0 means that all the probability mass goes forward, and 0.5, means that probability goes equally forward and backward.
- *Dis. Local*: Disallow N local matches. Set the prior to be zero on the last N places. We use the same parameter in our system during the BoW stage for not producing matches against the last N scenes.

The final values used by us are shown in table 3.5. Figs. 3.10(a,b) show the precision-recall curves varying the probability of acceptance p. We have chosen two parameter sets

Table 3.5: Parameters for the experiments

	FAB-MAP 2.0							Our System		
	Indoor		Outdoor		Mixed			Indoor	Outdoor	Mixed
	set 1	set 2	set 1	set 2	set 1	set 2				
p	50%	39%	96%	53%	62%	29%	α^+	0.6	0.6	0.6
$P(\text{obs}\|\text{exist})$	0.31	0.31	0.39	0.31	0.37	0.31	α^-	0.15	0.15	0.15
$P(\text{obs}\|\neg\text{exist})$	0.05	0.01	0.05	0	0.05	0	β_{3D}	1	1.5	1.5
$P(\text{newplace})$	0.9	0.9	0.9	0.9	0.9	0.9	β_{Im}	1.3	1.7	1.7
σ	1	0.99	1	1	1	1	τ_l	4s		
Motion Model	0.8	0.8	0.8	0.6	0.6	0.6	τ_d	2s		
Dis. Local	20s						Dis. Local	20s		



(a) FAB-MAP (set 1)

(b) FAB-MAP (set 2) + Epipolar Constraint

(c) BoW + Epipolar Constraint

(d) Our System

Figure 3.12: Loops detected by each of the methods in the Indoor dataset. First row: FAB-MAP with the set 1 of parameters (left), FAB-MAP with the set 2 of parameters plus the epipolar constraint (right). Second row: BoW + epipolar constraint (left), and our system (right). Black lines and triangles denote the trajectory of the robot; light green lines, actual loops, deep blue lines denote true loops detected, and light red lines denote false loops detected.

in order to obtain different results. The first one is tuned to obtain the maximum possible recall at one hundred percent precision. The second set is tuned to obtain the maximum possible recall, to improve the precision in a second step with geometrical checking. Since the last available version of the FAB-MAP software does not implement the geometrical checking described by Cummins and Newman (Cummins and Newman 2010), we implemented the geometrical checking based on epipolar geometry. This epipolar constraint consists in computing the fundamental matrix (by using RANSAC and the 8-point algorithm (Hartley 1997)) between the two matched scenes by FAB-MAP. This test is passed if a well conditioned fundamental matrix can be obtained.

The results of FAB-MAP over the datasets are shown in Figs. 3.12(a), 3.14(a) and 3.15(a) for the first set of parameters and in Figs. 3.12(b), 3.14(b) and 3.15(b) for the second set of parameters plus the geometrical checking with the epipolar constraint.

In order to show the improvements of our stage of loop closure verification, we have checked the same candidates with the epipolar constraint. The results are shown in

Figs. 3.12(c),. 3.14(c) and 3.15(c). In all the cases the precision was less than 100%. The results of our system over the datasets are shown in Figs. 3.12(d), 3.14(d) and 3.15(d), and the comparative statistics of all experiments made in the Table 3.7.

The indoor experiment is shown in Fig. 3.12. In Fig. 3.12(a), some loop closures are not detected, including the big area on the beginning of the map, especially important in the experiment because if no loop is detected in that area, a SLAM algorithm can hardly build a correct map after having traversed such a long path (around 300 metres). In Fig. 3.12(b) and 3.12(c) all the loop closures are detected but with too many false positives due to the perceptual aliasing, see Fig. 3.13; this is disastrous for any SLAM algorithm. The result from our system is shown in Fig. 3.12(d). At 100% precision we can detect all the loop closure areas.

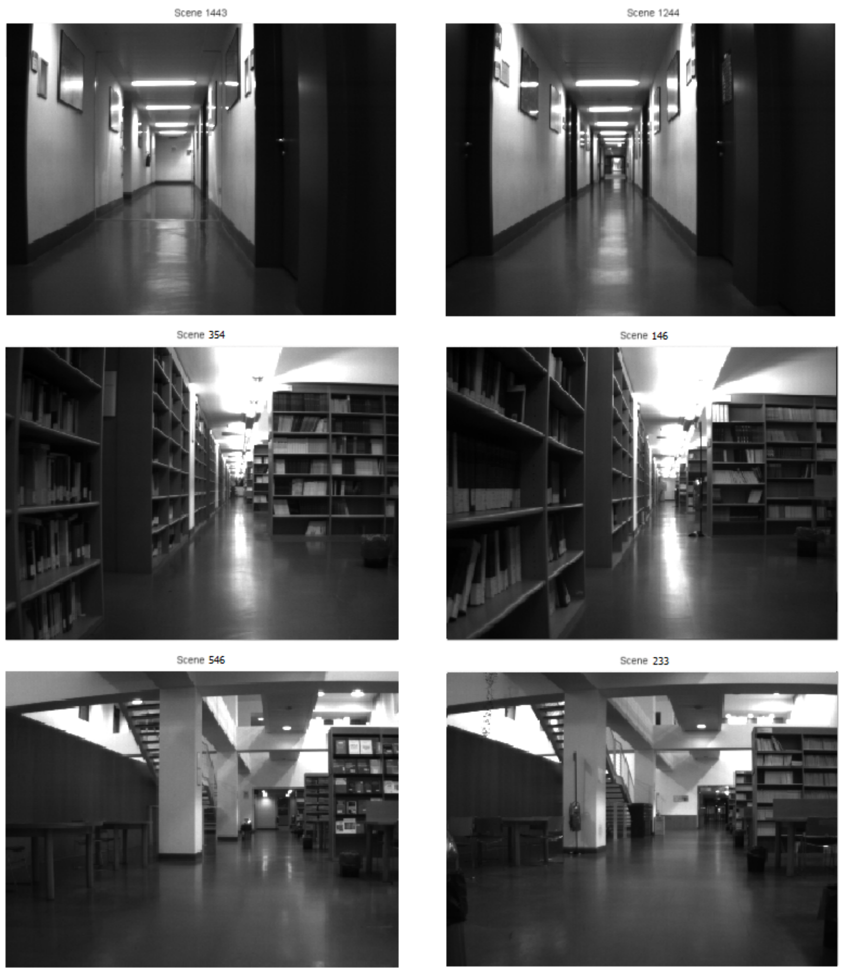
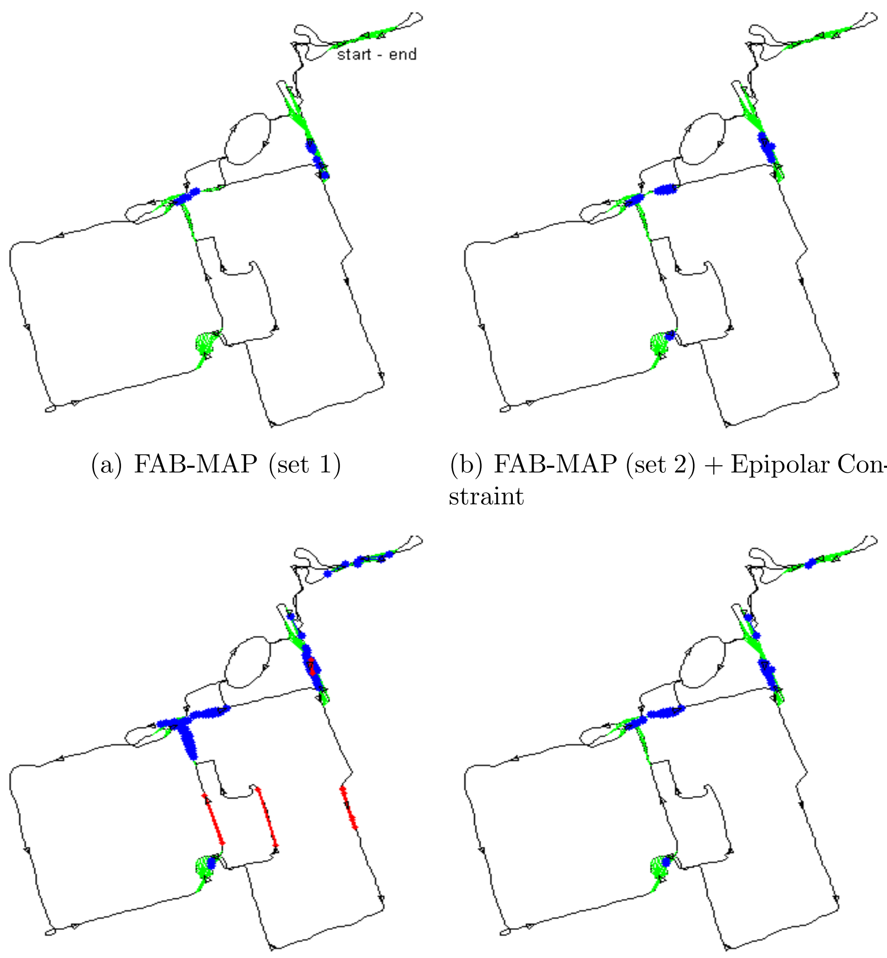


Figure 3.13: Three of the false positive cases that the FAB-MAP and BoW plus epipolar constraint obtain in the indoor dataset.

In the outdoor dataset, FAB-MAP does not detect all the loop closures either, as shown in Fig. 3.14(a). In Fig. 3.14(b) almost all the loop closures are detected at 100%



(a) FAB-MAP (set 1)

(b) FAB-MAP (set 2) + Epipolar Constraint

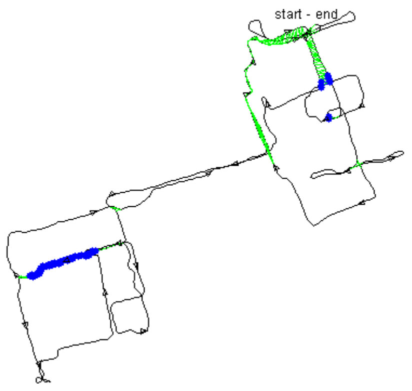
(c) BoW + Epipolar Constraint

(d) Our System

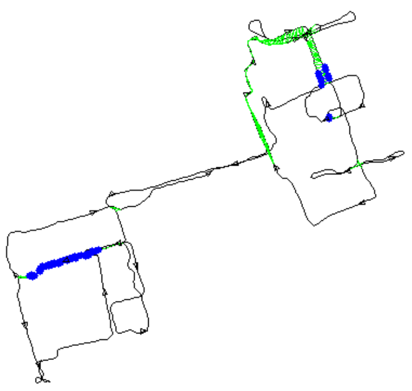
Figure 3.14: Loops detected by each of the methods in the Outdoor dataset. First row: FAB-MAP with the set 1 of parameters (left), FAB-MAP with the set 2 of parameters plus the epipolar constraint (right). Second row: BoW + epipolar constraint (left), and our system (right). Black lines and triangles denote the trajectory of the robot; light green lines, actual loops, deep blue lines denote true loops detected, and light red lines denote false loops detected.

precision, and the recall is close to the one obtained by our system (see second block, second and fourth row, of Table 3.7), but the biggest loop is missed in the starting and final point of the experiment, in the top-right area of the map. An example of a false negative in this area is shown in Fig. 3.16(a). The verification with the epipolar constraint over the candidates of our first stage results in some false positives, see Fig. 3.14(c). The result of our system is shown in Fig. 3.14(d). At 100% of precision we can detect all the loop closure areas.

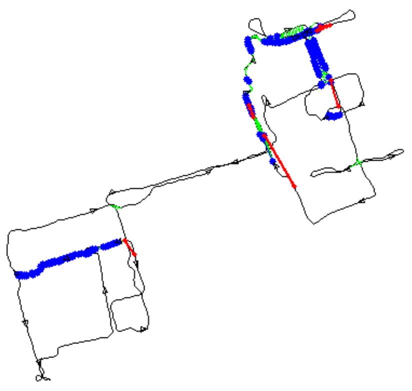
For the experiment in the dynamic mixed environment, both FAB-MAP and our system obtain 100% precision, though our system results in a higher recall level (Table 3.7, third block). Furthermore, FAB-MAP does not detect all the loop closure zones in



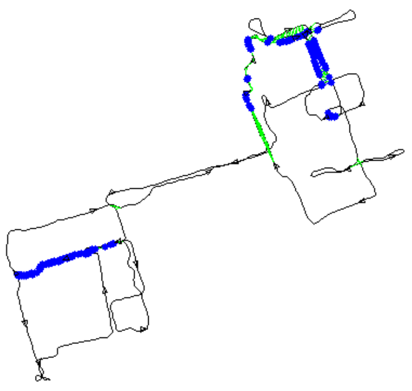
(a) FAB-MAP (set 1)



(b) FAB-MAP (set 2) + Epipolar Constraint



(c) BoW + Epipolar Constraint



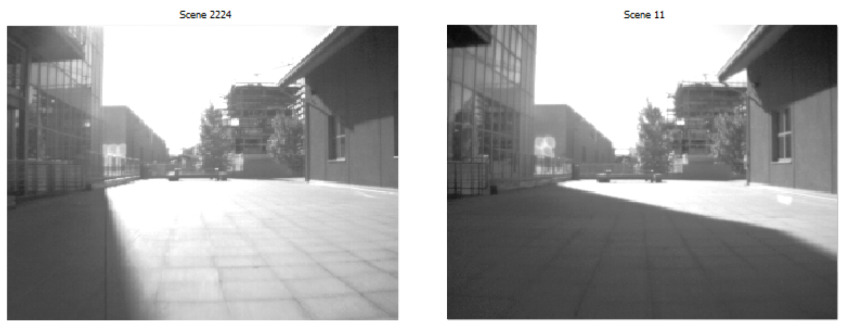
(d) Our System

Figure 3.15: Loops detected by each of the methods in the Mixed dataset. First row: FAB-MAP with the set 1 of parameters (left), FAB-MAP with the set 2 of parameters plus the epipolar constraint(right). Second row: BoW + epipolar constraint (left), and our system (right). Black lines and triangles denote the trajectory of the robot; light green lines, actual loops, deep blue lines denote true loops detected, and light red lines denote false loops detected.

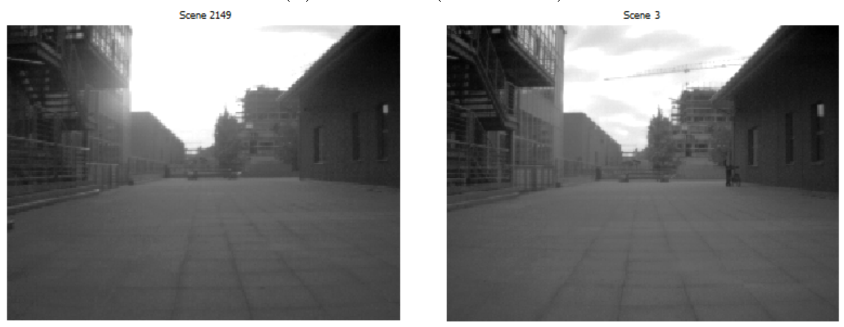
the map (see figures 3.15(a) and 3.15(b)). Examples of false negative cases are shown in Fig. 3.16(b). This cases are correctly matched by our system (see 3.15(d)). In this experiment, the verification with epipolar constraint over the candidates of our first stage result again in some false positives, see Fig. 3.15(c).

The system also was evaluated using a dataset taken in the MIT campus. In the Fig. 3.17 we sketch the trajectories (using Google Maps) and results. Both, our system and FAB-MAP obtain similar results in precision and recall. The weights used for CRF matcher are from table 3.3. And the parameters for both, FAB-MAP 2.0 and our system are showed in table 3.6.

The on-line system runs at 1 fps in all the experiments executed. We have a research implementation in C++ using the OpenCV library. In Table 3.8 we show the average and maximum times for each stage of the system on a 2.3 GHz IntelCore i3 CPU M350



(a) Outdoor (start-final)



(b) Mixed (start-final)

Figure 3.16: False negatives of FAB-MAP with geometrical checking, both with parameter sets 1 and 2, in the outdoor and mixed datasets. These scenes correspond to the biggest loops in the trajectories.

Table 3.6: Parameters for MIT Campus experiments

FAB-MAP 2.0	p	P(obs|exist)	P(obs|!exist)	Motion Model
	33%	0.39	0.05	0.6
Our system	α^+	α^-	β_{3D}	β_{Im}
	60%	15%	1.5	1.7

and 4GB of RAM. For the whole system, the average and the maximum times were computed only when all the stages were executed. Note that the maximums for each stage happen in different scenes. That is more evident in the inference process for $\mathcal{G}_{3D}$ and $\mathcal{G}_{Im}$: when a scene provides us with more 3D points, it leaves us with less background information. For a scene, the number of nodes and hidden states between $\mathcal{G}_{3D}$ and $\mathcal{G}_{Im}$ are complementary. The times reported for the CRFs in the graphs include computing the MSTs, the corresponding features and the inference for each one. The times for the whole system include computing the 3D point cloud from the disparity map and writing and reading the SURF descriptors and point clouds on disk.

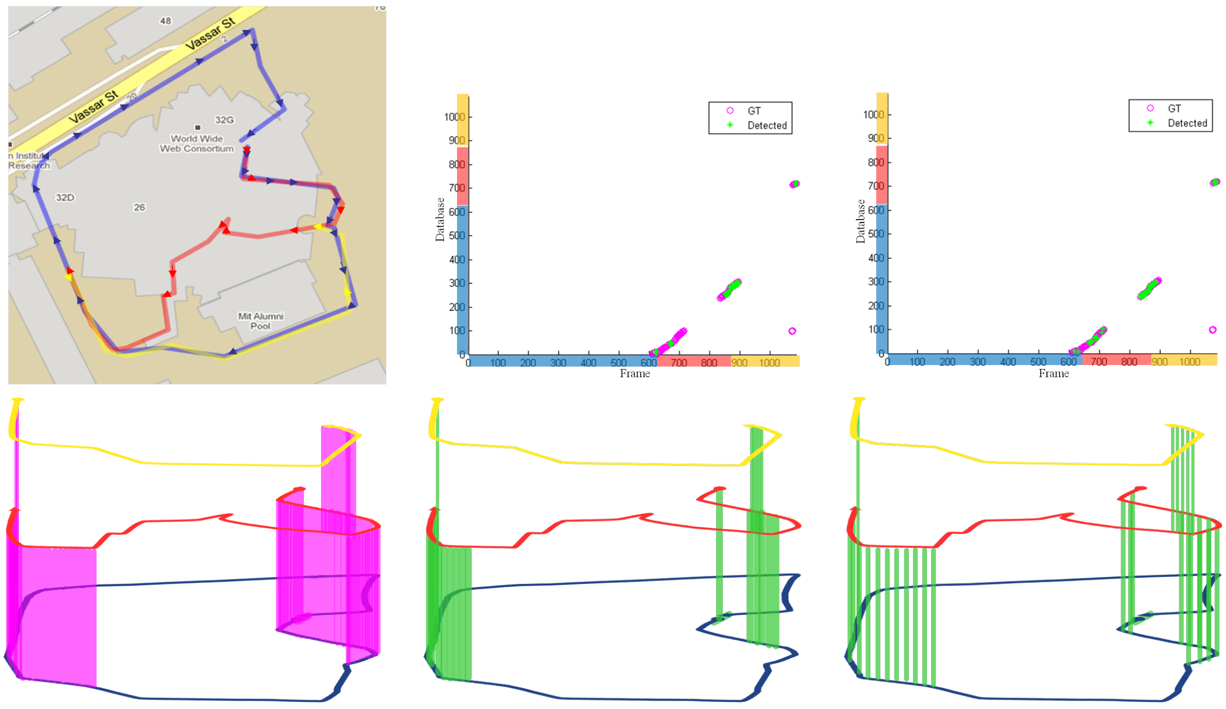


Figure 3.17: Multisession experiment in the MIT campus. Loops closure(green lines and stars) detected in the Stata Center multi-session dataset with FAB-MAP (top-middle and bottom-middle) and our system (top and bottom right). Different colours correspond to different sessions (blue, red and yellow). On the top, the Google map (left) and we show the query of the current frame vs. the database with the frames already added (middle and right). Ground truth (GT) is showed on bottom-left with magenta lines, on top with magenta circles.

Table 3.7: Results for all datasets

	Precision	Recall
RAWSEEDS Indoor		
FAB-MAP set 1	100%	26.12%
FAB-MAP set 2 + EC	52.37%	82.46%
BoW + EC	58.54%	80.6%
Our System	**100%**	**58.21%**
RAWSEEDS Outdoor		
FAB-MAP set 1	100%	3.82%
FAB-MAP set 2 + EC	100%	10.83%
BoW + EC	92.63%	28.03%
Our System	**100%**	**11.15%**
RAWSEEDS Mixed		
FAB-MAP set 1	100%	13.47%
FAB-MAP set 2 + EC	100%	15.27%
BoW + EC	92.9%	47.01%
Our System	**100%**	**35.63%**
Multisession MIT		
FAB-MAP	100%	38.89%
Our System	**100%**	**38.27%**

Table 3.8: Computational times for our system (in s)

	SURF extraction	BoW	CRF Matcher $\mathcal{G}_{3D}$	CRF Matcher $\mathcal{G}_{Im}$	Whole System
Average	0.15	0.01	0.15	0.15	0.47
Maximum	0.30	0.04	0.36	0.65	1.04

3.4 Discussion

In this chapter we have presented a place recognition system that combines two powerful matching algorithms, bag-of-words and conditional random fields, to robustly solve the place recognition problem with stereo cameras. We have evaluated our place recognition system in public datasets and in different environments (indoor, outdoor and mixed). In all cases the system can attain 100% precision (no false positives) with higher recall than the state of the art (less false negatives), and detecting all (especially important) loop closure zones. No false positives means that the environment model will not be corrupted, and less false negatives means that it will be more precise. Our system also is more robust in situations of perceptual aliasing.

As mentioned by Piniés et al. (2010), the effectiveness of FAB-MAP decreases when the camera looks forward, because FAB-MAP models the environment as "a collection of discrete and disjoint locations" (Cummins and Newman 2008). However, in our experiments the stereo camera system faces forward and distant objects (e.g., buildings in outdoor scenes) persist for many frames, thus making scenes overlap and be less discriminative. This causes the matching probability mass of FAB-MAP to be flattened over the scenes. It is easier for our system to overcome those cases because our normalised similarity scores (η_c, η_{3D}, η_{Im}) for matching acceptance are computed at each frame and take into account the similarity between consecutive frames.

By using jointly the CRF-Matching algorithm over visual near 3D information (here provided by stereo vision, but also possible with range scanners, etc.) and far information, we have demonstrated that challenging false loop closures can be rejected. Furthermore, CRF-Matching is also able to fuse any other kind of information, such as image colour, with ease.

Our place recognition system is able to run in real time, processing scenes at one frame per second. In most cases, after extracting the SURF features (max. $300ms$), our system only takes $11ms$ to check if there are possible loop closures.

In our experiments, the β thresholds for acceptance of the CRF matching turned out to be clearly different for indoor and for outdoors scenarios. These parameters will also depend on the velocity of motion, mainly due to the fact that we use images from the previous second as reference in the comparisons. Incorporating the computation of these thresholds as part of the learning stage would also make the system more flexible.

An important line of future work is addressing the place recognition problem over time. Our system performs well in multi-day sessions using parameters learned in different months, and this is also true of alternative systems such as FAB-MAP. The environment can also change during the operation in the same session, see Fig. 3.16. Our algorithm is also able to detect places revisited at different times of day, while alternative systems sometimes reject them in order to maintain high precision.

Several extensions are possible for operation in longer periods of time. The vocabulary for the BoW has shown to be useful in different environments, which suggests that a rich vocabulary does not require frequent updates. The learned parameters in the CRF stage can be re-learned in sliding window mode depending on the duration of the mission. The system would then be able to adjust to changing conditions. In cases of periodical changes, such as times of day or seasons, we will need to maintain several environment models and selecting the most appropriate for a given moment of operation.

Chapter 4

Experiments Using the Combined Filter and Our Place Recognition

In this chapter our two main contributions are assembled in to a unified featured-based SLAM system, which is tested in diverse environments with different sensors. Our purpose is to show that these two components can be very valuable to build robust and efficient SLAM systems.

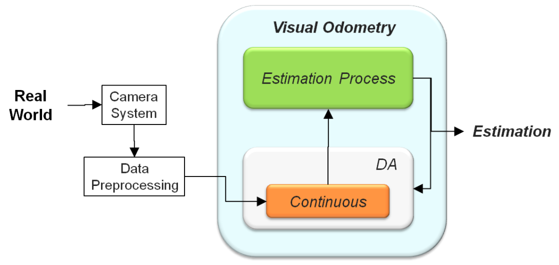
There are several successful SLAM systems proposed by different robotic groups around the world. In our opinion two of the most impressive and complete experimental results are Konolige et al. (2010a) in indoor environments and Newman et al. (2009) in outdoor environments.

View-based Maps proposed by Konolige et al. (2010a) are topological maps that use geometric feature matching in stereo views and maintain a vocabulary tree to check loop-closure candidates. The sketch of the system structure is shown in Fig. 4.1(c). Odometry is computed from the stereo vision. The system has the advantage of scalability: using incremental techniques, new views can be added and the skeleton is optimised online. Skeletons are similar to the pose graphs familiar from laser SLAM work although the latter typical have just 2D implementations. Also, skeletons retain the images that are captured at each node, for future matching via place recognition with a simple bag-of-words approach with a strong geometrical checking. The skeleton is optimised by bundle adjustment and can handle multi-sessions. Recently, McDonald et al. (2011) propose a 6DoF multisession visual SLAM system using also a stereo camera based on the iSAM with anchor nodes (Kim et al. 2010) and our place recognition system.

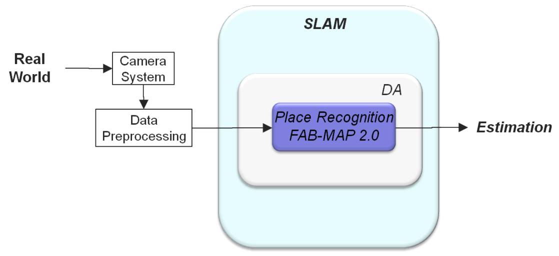
Newman et al. (2009) present an impressive integrated system with laser, stereo and omni vision, using visual odometry for front-end tracking from stereo cameras, and FAB-MAP for on-line loop closure using an omnidirectional camera system. When FAB-MAP gives a loop closure, a metric constraint is computed with ICP using either the laser data or the dense point cloud from the stereo. The constraints are used in a batch optimisation. They also note that maps can be stitched together over non-contiguous runs, although they use batch-mode processing to do so.

Using topological maps, Cummins and Newman (2010) have shown the largest experiment with their FABMAP 2.0, see Fig. 4.1(b). Using an omni-directional camera installed on a car and a soft geometrical constraint check, their system was tested using two extremely large (70km and 1000km) datasets.

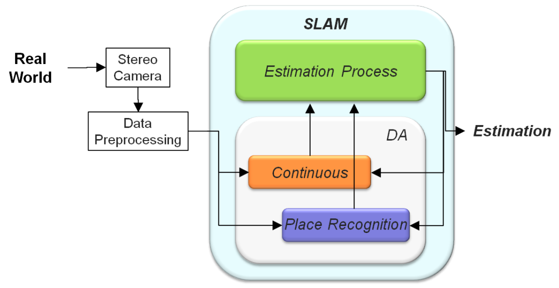
A complete feature-based SLAM system is proposed by Piniés et al. (2010). They implemented the CI-Graph (Pinies et al. 2009) as core for the estimation and a bag-of-words with geometrical checking for solving loop closures. This system was evaluated in two datasets from the RAWSEEDS Project, one indoors and one outdoors. The sensors used for estimation were the trinocular camera system on board the robot and the odometry when it is available. The map contains 3D point features in both cartesian and in inverse depth parametrisation (Montiel et al. 2008). Place recognition works over one of the images of the trinocular system.



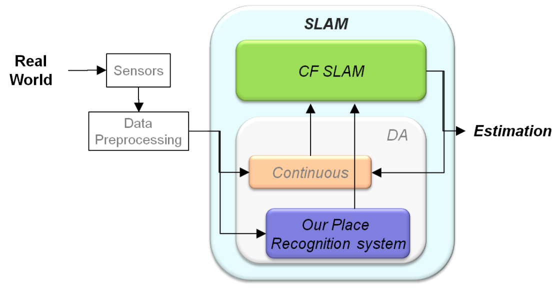
(a) Structure of the visual odometry system of Maimone et al. (2007).



(b) Structure of the topological system of Cummins and Newman (2010).



(c) Structure of the full SLAM system of Konolige et al. (2010a).



(d) Structure of the SLAM system used in this thesis.

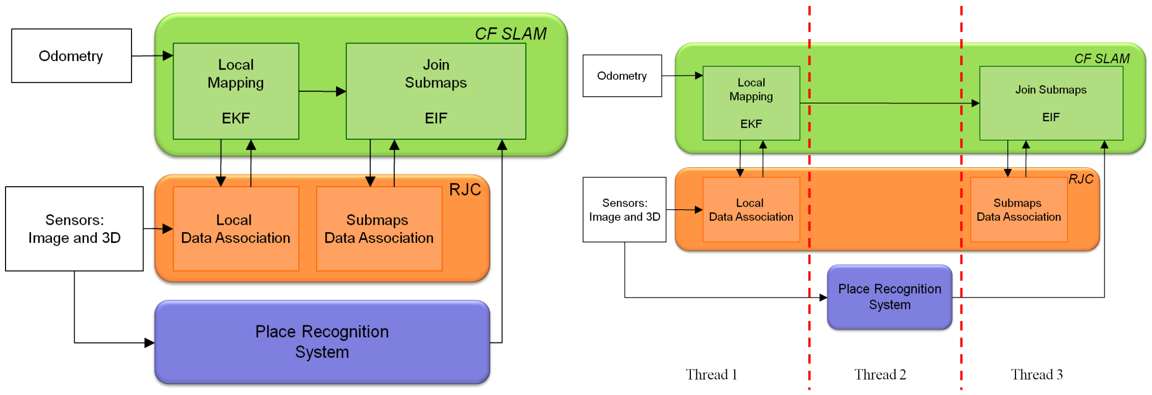
Figure 4.1: Some system structures.

Systems based on visual odometry are very convenient for exploration tasks (Maimone et al. 2007), see Fig. 4.1(a). These systems have the important advantage of constant time execution. In only exploratory trajectories 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. On the Mars Exploration Rovers (Maimone et al. 2007), the visual odometry algorithm uses image feature tracking between stereo image pairs to estimate the translation and rotation between image captures. Unfortunately, visual odometry does not cope with loop closings, and thus, eventual drift in these cases is inevitable.

These systems have a very careful engineering work in each stage, from the selection and management of the features to track, to the use of weak links for multisession SLAM in (Konolige et al. 2010a), or the semantic labeling of the point clouds in (Newman et al. 2009). The intention of the system implemented here, Fig. 4.1(d), is show the good performance in efficiency and robustness of our two contributions working together. Analysis related to the features to track, or the kind of odometry to use (or not) is out of the scope of this chapter.

4.1 System Overview

Here, we integrate our two main contributions in a feature based SLAM system. We show a schematic of the system architecture in Fig. 4.2(a).



(a) Architecture. (b) Different thread that could be parallelised.

Figure 4.2: Our feature based SLAM.

The estimation process is carried out with the CF SLAM algorithm proposed by us in (Cadena and Neira 2009, 2010) and detailed in this thesis in Chapter 2. Data association inside local maps is carried out with randomised joint compatibility (RJC) of Paz et al. (2007). We also use RJC the as data association algorithm between local maps in the limits of the submaps. To detect revisited places, we use our place recognition system proposed by us in (Cadena et al. 2010, 2011b) and explained in this thesis in Chapter 3. In each map joining the CF SLAM asks to the place recognition system if there is loop closing between the two sub-maps.

In the following we detail the system components organised by threads, see Fig. 4.2(b).

4.1.1 Local Mapping - Thread 1

Information coming from the sensors is processed to carry out local mapping with a basic EKF SLAM. Since we have available the state covariance, RJC is used for data association. The modules in this thread work as follows:

1. *Odometry*: Obtains the odometry information if is available in the vehicle to be used in the EKF prediction step. We keep a pose in the state vector with six components $[x, y, z, \phi, \theta, \psi]$, three Euclidean coordinates and the roll-pitch-yaw angles. In the experiments where the odometry is not available, we apply a constant velocity model in which we represent the pose with twelve components in the state vector: three Euclidean coordinates, three angles and six parameters for linear and angular velocities.

2. *Sensors*: Inputs are the images and the 3D information from 3D lidar or computed from the stereo cameras. Here, we extract SURF-points in the image and send them to local data association. If the SURF-points have related 3D information, we also send it.

3. *Local Mapping - EKF*: Runs a basic EKF SLAM. Besides of the state vector and covariance, we also save the timestamps and descriptors of the first time that each feature was added in the map.

4. *Local Data Association*: This module finds the correspondences between the features already in the map with the observations in the current step. The individual compatibility is computed in the descriptor space and the RJC is carried out with the covariance matrix in order to get reliable correspondences.

When a local map reaches a minimum of features f_{min} and a minimum of steps s_{min} we compute its information form and send to the map joining process. A new local map is initialised. We enforce a minimum of steps here to guarantee that the map joining process can be amortised in a known number of steps.

4.1.2 Place Recognition - Thread 2

The place recognition system can process images and 3D information from the sensors at 1 fps with their respective timestamps. As explained in section 3.2, SURF-points are used for the loop candidates detection stage. With the 3D information, we compute the features and the graphs for the inference in the CRF stage of loop closure validation. We use the SURF and 3D extraction already computed by the sensors module.

The revisited places are sent to the map joining process along with their timestamps.

4.1.3 Joining Submaps - Thread 3

We join the local maps with the divide and conquer strategy. The modules in this process are (see Fig. 4.3):

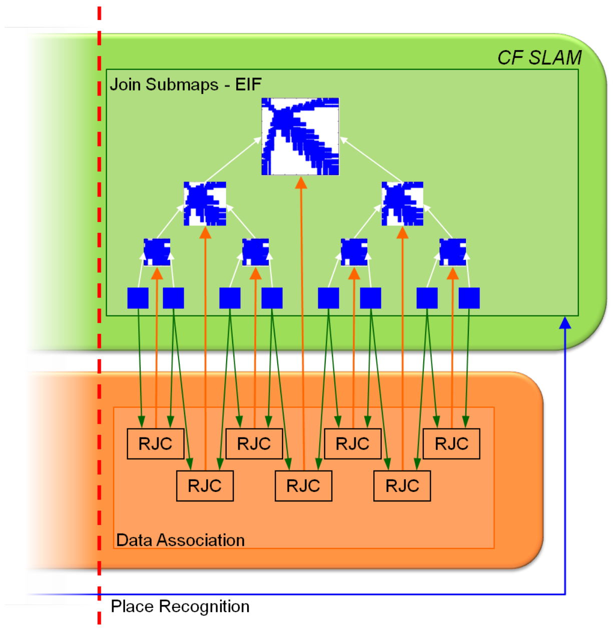


Figure 4.3: A more detailed view of the submap joining process.

1. *Data Association*: Taking advantage of the fact that the local maps have the state covariance available, we execute the RJC algorithm over each consecutive pair of local maps. In that way, the sequential correspondences between each two submaps can be solved in constant time, independent of the size of the submaps to join in

EIF form. When the local maps are not necessary any more, they are eliminated of the memory. Timestamps and descriptors are kept in the computed submaps.

2. *Join submaps - EIF*: This module joins two submaps with EIF. In each join uses the correspondences between submaps from the data association module. In case the two submaps contain timestamps corresponding to revisited places according to the place recognition system, the features in those timestamps are pairing in the descriptor space and also used in the join process.

   Note that the cost of this process could be amortised during $2^l s_{min}$ steps, l equal to the level in the tree of submaps, with the guarantee of finish the joining before the resulting submap is required. In our prototype implementation the amortisation is not yet.

The CF SLAM and the data association processes are implemented in MATLAB. In this prototype we have not implemented local mapping and joining submaps in different threads. This is in our plans for future work. The place recognition process is implemented in C++ using the OpenCV library. The system runs on a 2.3 GHz IntelCore i3 CPU M350 and 4GB of RAM.

4.2 Evaluation

In this section we evaluate the system using publicly available datasets over two different sensors and robotic platforms, using the RAWSEEDS (2009) Project and the Ford Campus Vision and Lidar Dataset of Pandey et al. (2011).

Here we show four experiments, in three different locations: a building of the Bicocca-campus belonging to the Università di Milano-Bicocca in Milan (Italy), another in the Bovisa-campus of the Politecnico di Milano, located in via Durando, Milan (Italy), and in the Ford Research campus located in Dearborn, Michigan (USA), see Fig. 4.4.

The experiments were taken in a wide variety of urban environments and situations: indoors, outdoors, mixed (indoor-outdoor), static, dynamic, and with natural and artificial illumination.

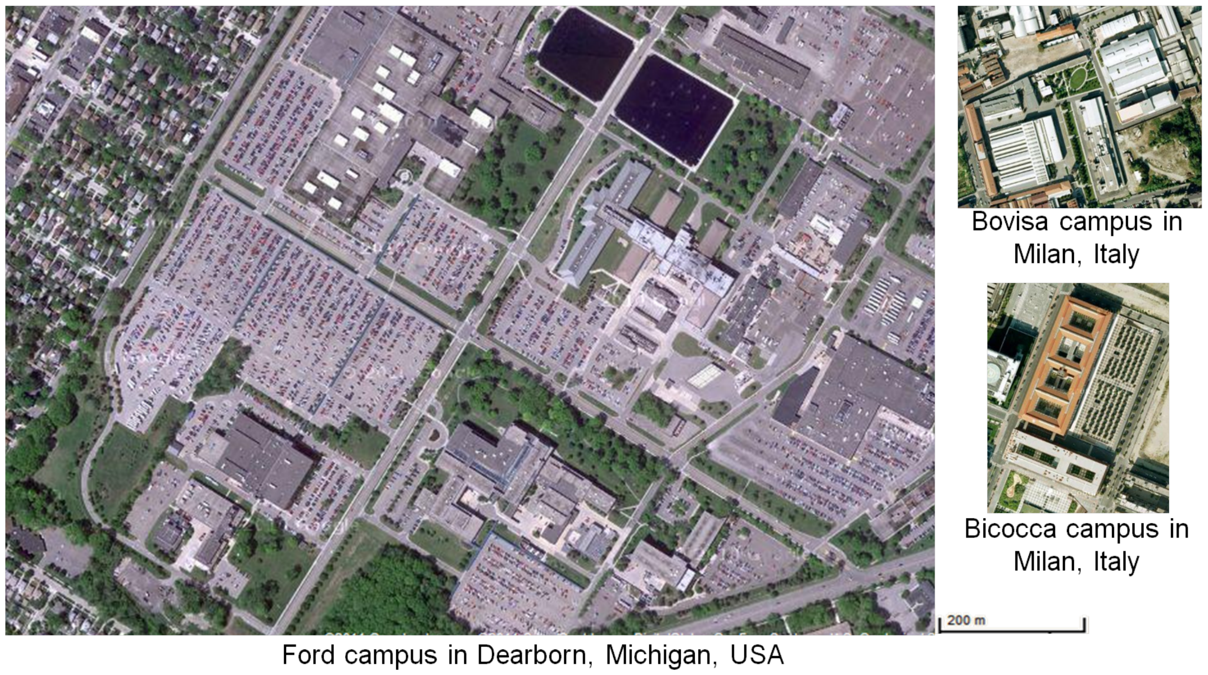

Bovisa campus in Milan, Italy
Bicocca campus in Milan, Italy
200 m
Ford campus in Dearborn, Michigan, USA

Figure 4.4: Locations used for the experimental evaluation of our system. The satellite images are courtesy of Google Maps.

4.2.1 With Stereo Cameras

First we evaluate our SLAM system on a robotic platform with a stereo camera, our place recognition system was first developed for that sensor. We use three datasets from the RAWSEEDS project.

We process the datasets at 5 frames per second in our local mapping and at 1 fps in our place recognition system. In local mapping, we select $f_{min} = 50$ and $s_{min} = 10 * fps = 50$ steps resulting in at least $10s$ per local map.

For local mapping we use features parametrised in 3D Euclidean coordinates when the SURF-points have 3D information from the stereo camera. For the remaining SURF-points, we use the inverse depth parametrisation of Montiel et al. (2008). When the local map is closed and another must be initialised, the inverse depth features with enough parallax index as explained by Civera et al. (2007), are converted to 3D cartesian. The features with insufficient parallax are marginalised from the local map.

Accuracy is evaluated by comparing the estimated trajectory to the extended ground truth solution (only indoors) or GPS data with a precision of 0.9m (outdoors). The Absolute Trajectory Errors (ATE) are calculated for each timestamps in the ground truth (GT). Poses from our final estimation are interpolated to the available timestamps in GT. These are computed using the *Rawseeds Metrics Computation Toolkit* provided by the

RAWSEEDS Project.

PR Learning Stage: Bovisa_2008-09-01_Static

Our place recognition system uses a vocabulary and a vector of weights for the potentials in CRF-matching. Here we uses this information as we explained in Chapter 3 with other dataset different to the used for the SLAM evaluation. Only for remember, we show the parameters used in our place recognition in table 4.1

Table 4.1: Parameters for RAWSEEDS datasets

	α^+	α^-	β_{3D}	β_{Im}
Indoor	60%	15%	1.0	1.3
Mixed	60%	15%	1.5	1.7
Outdoor	60%	15%	1.5	1.7

The place recognition system is executed in parallel with the cost per step as shown again in the table 4.2.

Table 4.2: Computational times for PR (in s)

	SURF	BoW	CRF Matcher		Whole
	extraction		$\mathcal{G}_{3D}$	$\mathcal{G}_{Im}$	System
Average	0.15	0.01	0.15	0.15	0.47
Maximum	0.30	0.04	0.36	0.65	1.04

Indoor: Bicocca_2009-02-25b

This is a dataset in a static environment with artificial illumination. The stereo images were collected during 30min along a path of 760m.

In total 169 local maps are built. The fianl result is a map of 5426 features and 169 poses, a state vector of 17292 components and an information matrix with 4277307 non zero entries, 1.3% of density. The final map can be seen in Fig. 4.5.

We show in Fig. 4.6 (left) the cost per step to carry out each process in the CF SLAM algorithm: local mapping and joining submaps, and data association for submaps. Local mapping includes the cost of its internal data association (solid light-blue line). The joining submap cost (dashed orange line) and data association of submaps (dash-dot pink line) could be amortised during the next $2^l 10$ seconds as we show in Fig. 4.6 (bottom-left). The peak in the last step is because the experiment has finished and the global map is required. Even though, that cost is only 10.8s for all the pending submap joins and the

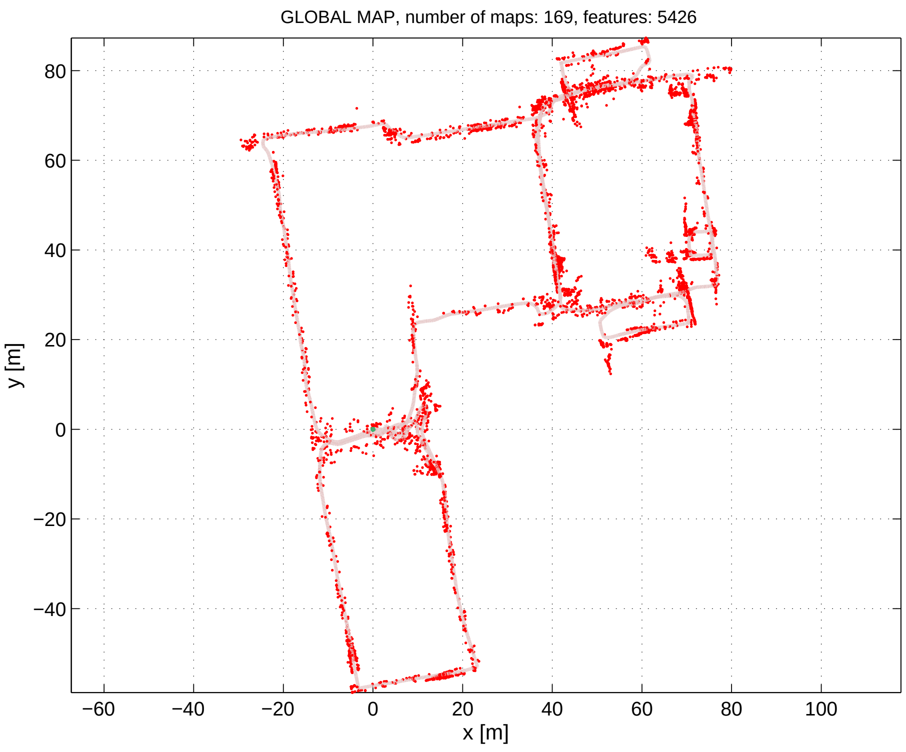


Figure 4.5: Final map for the Bicocca_2009-02-25b dataset after running our SLAM system.

most of the time the cost is dominated by the local mapping that is constant. In Fig. 4.6 (right) we show the cumulative cost for the same variables.

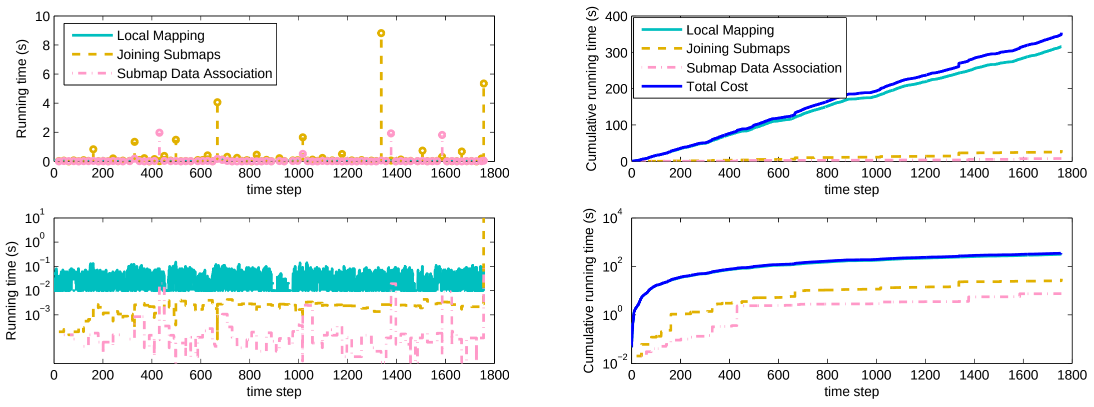
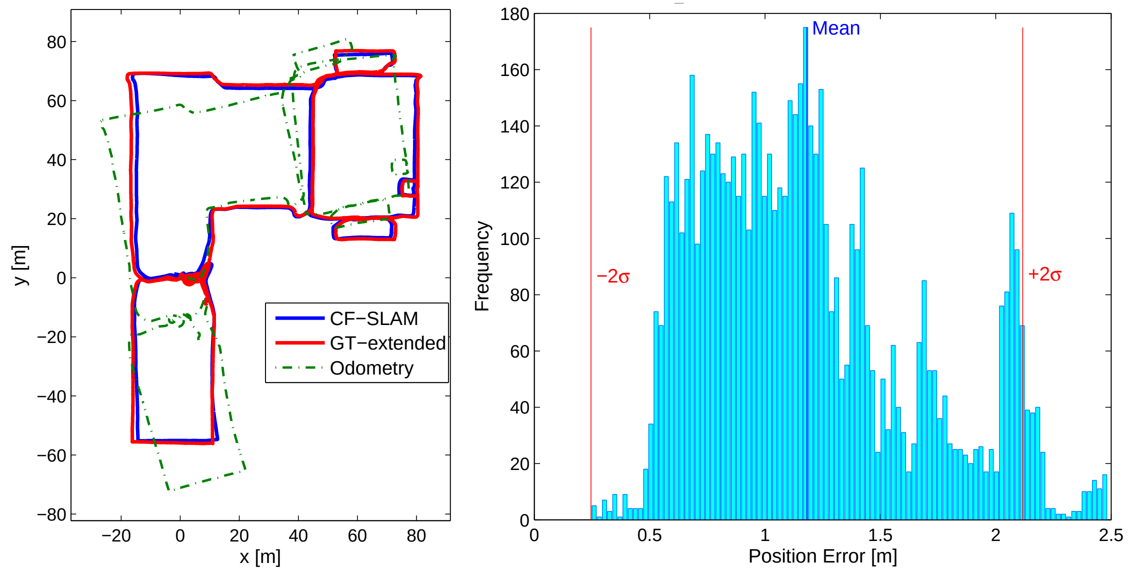


Figure 4.6: Running times for our system decomposed in Local Mapping, Joining submaps and Data Association between submaps. On the left, top: running times per step for our system; bottom in semi-log scale: The same running times but amortising the costs of joining and DA between submaps. On the rigth, top: the cumulative running times including the total cost; bottom: the same cumulative times in semi-log scale.

Fig. 4.7(a) shows the estimated trajectory (solid blue line), the odometry (dashed green line) and the GT (solid red line). They are aligned using the Rawseeds Metrics Computation Toolkit. The histogram of errors in the trajectory is shown in Fig. 4.7(b).

Our solution achieves a mean absolute error of 1.18m with 0.47m of standard deviation. Note that the maximum error is as only 2.5m, 0.33% of the 760m travelled.



(a) Trajectory. (b) Histogram of ATE.

Figure 4.7: Estimated trajectory compared against the extended ground truth and the odometry on the left. Robot pose error histogram with 2σ error bounds on the right.

Mixed: Bovisa_2008-10-06_Dynamic

This dataset was taken in a dynamic environment in a combination of indoors and outdoors with natural illumination. The stereo images were collected during 35 min along a path of 1.89km.

In total 212 local maps are built. The result is one final map of 10056 features and 212 poses, a state vector of 31440 components and an information matrix with 8841177 non zero entries, 0.36% of density. The final map can be seen in Fig. 4.8.

We show in Fig. 4.9 (left) the cost per step to carry out each process in the CF SLAM algorithm: local mapping and joining submaps, and data association for submaps. With partial outdoor trajectories, the local mapping adds more features in inverse depth parametrisation, increased cost due to more features with greater number components in the parametrisation.

Local mapping includes the cost of its internal data association (solid light-blue line). The joining submap cost (dashed orange line) and data association of submaps (dash-dot pink line) could be amortised as we show in Fig. 4.9 (bottom-left). The peak in the last step is 38s for all the pending submap joins and the most of the time the cost is dominated

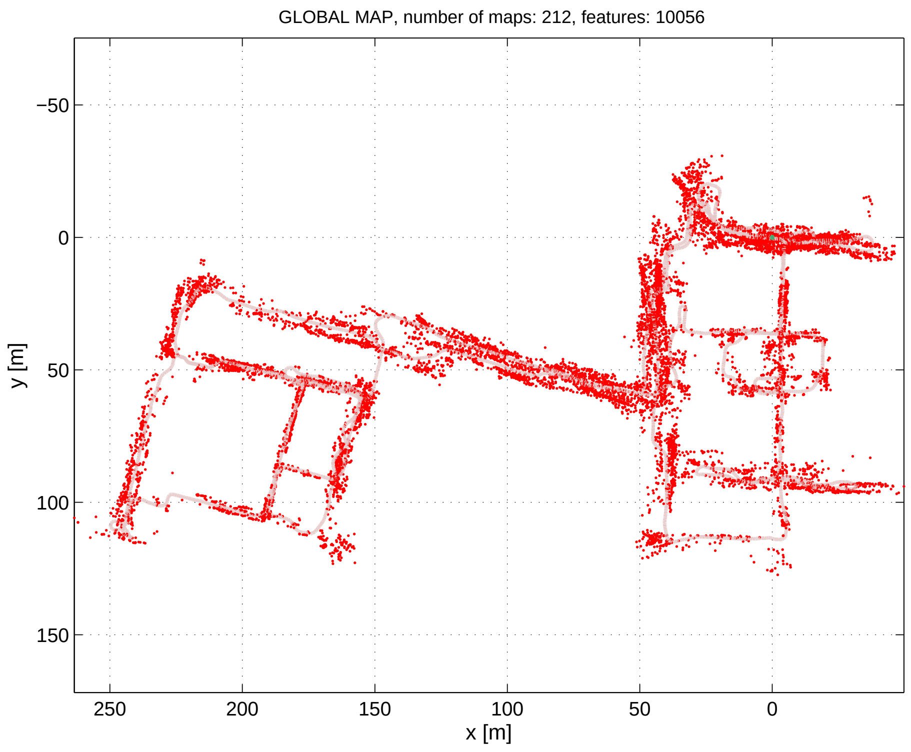


Figure 4.8: Final map for the Bovisa_2008-10-06 dataset after running our SLAM system.

by the local mapping, lesser than 0.1s. The length of the trajectory is more twice than in the indoor experiment, doubling the size of the final map and increased the cost of the joining process. In Fig. 4.9 (right) we show the cumulative cost for the same variables.

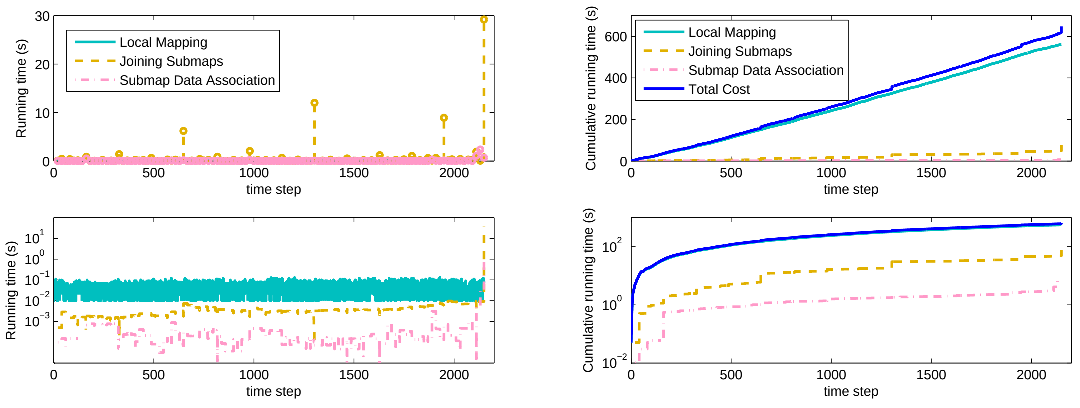
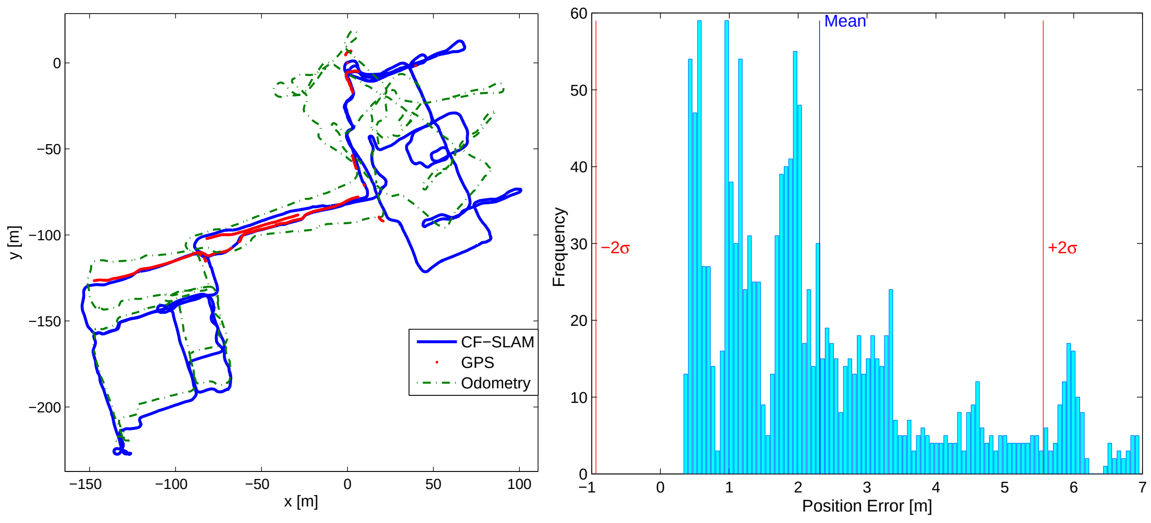


Figure 4.9: Running times for our system decomposed in Local Mapping, Joining submaps and Data Association between submaps. On the left, top: running times per step for our system; bottom in semi-log scale: The same running times but amortising the costs of joining and DA between submaps. On the rigth, top: the cumulative running times including the total cost; bottom: the same cumulative times in semi-log scale.

Fig. 4.10(a) shows the estimated trajectory (solid blue line), the odometry (dashed green line) and the GT (solid red line). They are aligned using the Rawseeds Metrics

Computation Toolkit. The histogram of errors in the trajectory is shown in Fig. 4.10(b). Our solution achieves a mean absolute error of 2.31m with 1.62m of standard deviation.



(a) Trajectory. (b) Histogram of ATE.

Figure 4.10: Estimated trajectory compared against the ground truth given by the GPS and the odometry on the left. Robot pose error histogram with 2σ error bounds on the right.

Outdoor: Bovisa_2008-10-04_Static

This dataset is taken in a static environment with natural illumination. The stereo images were collected during 38 min along a path of 1.72km. For this dataset, a partial visual odometry solution is available as a "Rawseeds Benchmark Solution" covering 1.3km of the trajectory. The visual odometry was obtained and published by Civera et al. (2010). They use the fontral camera (320x240) and their EKF + 1-point RANSAC algorithm, using the wheel odometry to observe the scale. We use this visual odometry in the timestamps where it is available, when it is not, we use the simple wheel odometry, see Fig 4.11.

In total 223 local maps are built. The result is one final map of 10170 features and 223 poses, a state vector of 31848 components and an information matrix with 9876676 non zero entries, 0.32% of density. The final map can be seen in Fig. 4.12.

We show in Fig. 4.13 (left) the cost per step to carry out each process in the CF SLAM algorithm: local mapping and joining submaps, and data association for submaps. In outdoor experiments, the local mapping adds more features in inverse depth parametrisation, increased again the cost due to more features with greater number components in the parametrisation.

Local mapping includes the cost of its internal data association (solid light-blue line).

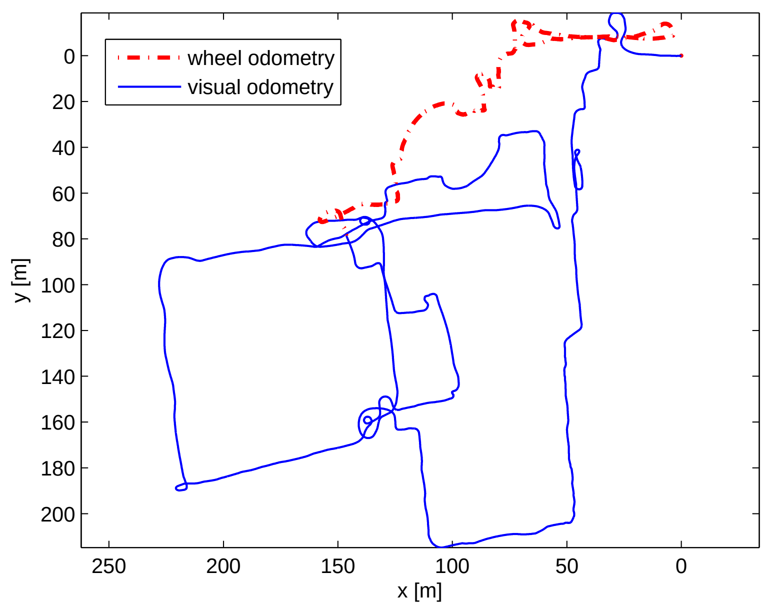


Figure 4.11: Odometry used in this dataset. Visual odometry covers 1.3km and the wheel odometry the remaining 0.4km.

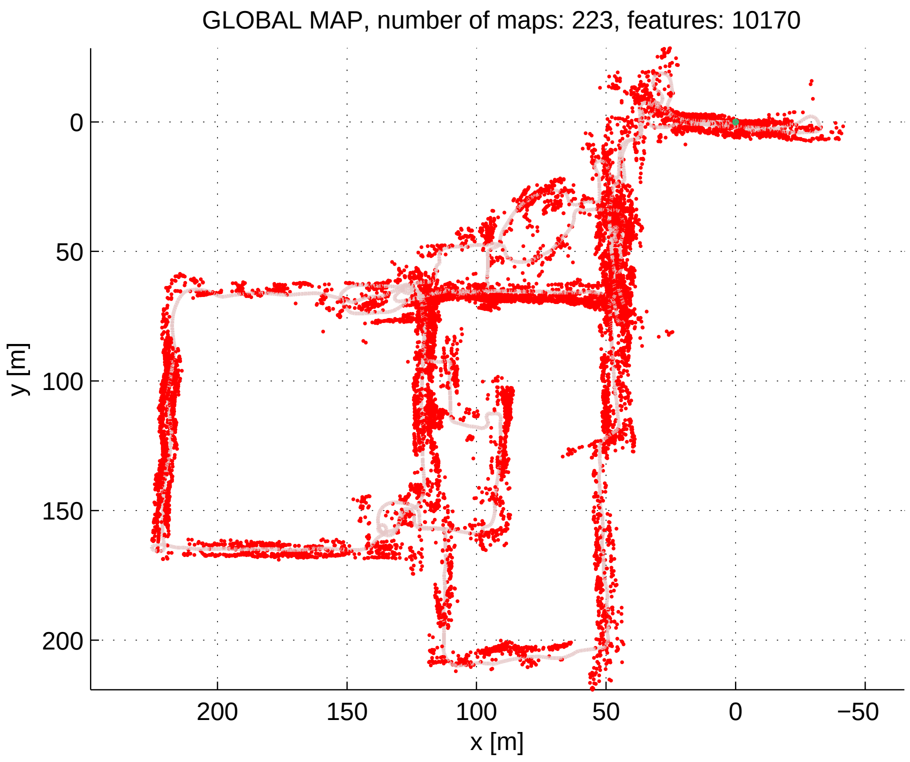


Figure 4.12: Final map for the Bovisa_2008-10-04 dataset after running our SLAM system.

The joining submap cost (dashed orange line) and data association of submaps (dash-dot pink line) could be amortised as we show in Fig. 4.13 (bottom-left). The peak in the last step is 38.5s for all the pending submap joins. The cost is dominated by the local mapping, lesser than 0.1s the most of the time. The length of the trajectory is more twice than in the indoor experiment, doubling the size of the final map and increased the cost of the joining process. In Fig. 4.13 (right) we show the cumulative cost for the same variables.

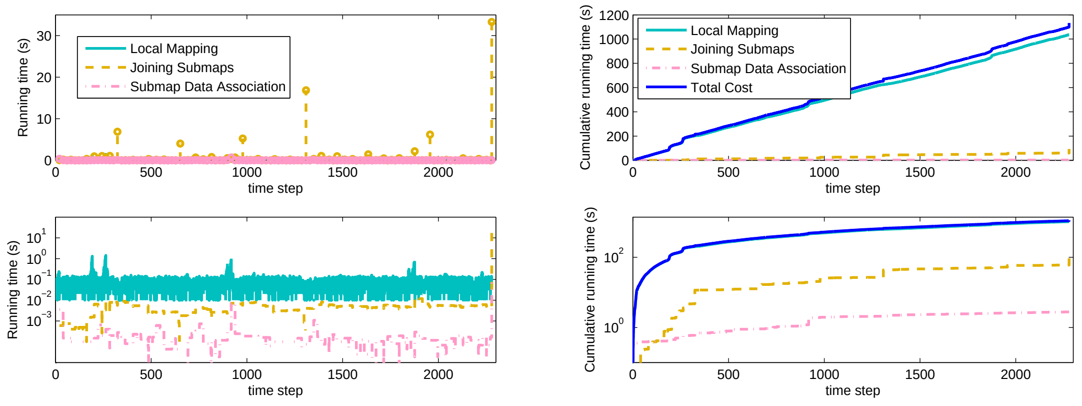


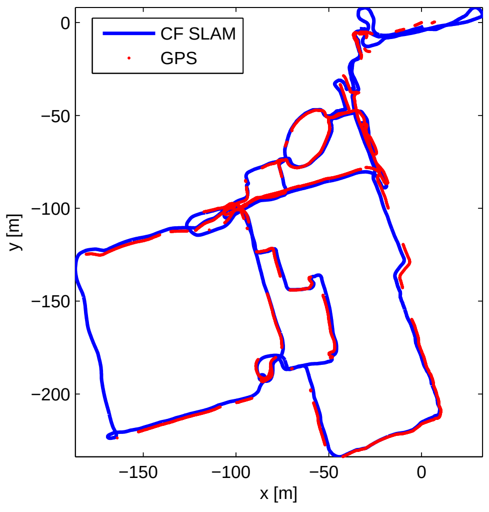
Figure 4.13: Running times for our system decomposed in Local Mapping, Joining submaps and Data Association between submaps. On the left, top: running times per step for our system; bottom in semi-log scale: The same running times but amortising the costs of joining and DA between submaps. On the rigth, top: the cumulative running times including the total cost; bottom: the same cumulative times in semi-log scale.

Fig. 4.14(a) shows the estimated trajectory (solid blue line), the odometry (dashed green line) and the GT (solid red line). They are aligned using the Rawseeds Metrics Computation Toolkit. The histogram of errors in the trajectory is shown in Fig. 4.14(b). Our solution achieves a mean absolute error of 1.58m with 0.66m of standard deviation.

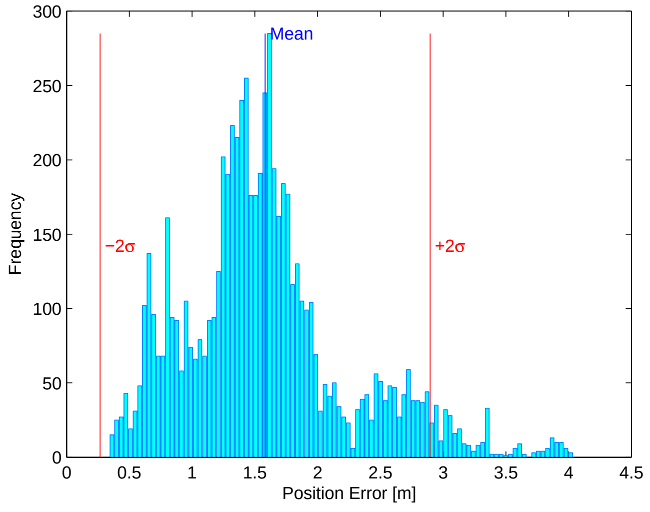
4.2.2 With Omidirectional Cameras + 3D LIDAR

In this experiment, we evaluate our SLAM system on an autonomous ground vehicle, a modified Ford F-250 pickup truck equipped with several sensors. We use the PointGrey Ladybug3 omnidirectional camera system to obtain the images and the Velodyne three-dimensional lidar scanner for 3D information. Pandey et al. (2011) provide the data between these two sensors already calibrated. We use the five calibrated images from the ladybug cropped to 788x1236 each one.

Two datasets are available, the first one (Ford-1) was taken in downtown Dearborn, MI (USA), in a trajectory of 1.44km during $\sim$ 6 min with a single large loop. The second one (Ford-2) was taken in the Ford Research campus, also in Michigan. The trajectory of Ford-2 is a two large loops path of 4km during $\sim$ 10 min



(a) Trajectory.



(b) Histogram of ATE.

Figure 4.14: Estimated trajectory compared against the ground truth given by the GPS on the left. Robot pose error histogram with 2σ error bounds on the right.

PR Learning Stage: Ford-1

Our place recognition system is based on information of texture and 3D, and these dataset provide us with both. The learning process is carried out using Ford-1 in the same way than for the stereo camera datasets, but taking into account the five images for the 360° of field of view and the 3D information directly from the 3D laser. We show the resulting weights in table 4.3 learned from randomly chosen 200 pairs of scenes separated 0.1 seconds between them in Ford-1.

		SURF	*geo*			*PCA*			*pair*	*outlier*	
			k=1	k=2	k=3	1st	2nd	3rd		*local*	*pair*
$\mathcal{G}_{3D}$	**w**	-13.50	0.16	-0.53	-0.41	0.21	-0.85	-0.57	-12.42	-2.78	-1.83
$\mathcal{G}_{Im}$	**w**	-15.98							-34.48	-1.90	-1.38

Table 4.3: Weights obtained in the learning process in the Ford-1 dataset.

The vocabulary for the BoW stage was trained here with all the image of Ford-1 with $k = 10$ and $l = 6$ given us a dictionary of 1 million of words. We carry out our place recognition system first using Ford-1 for tuning the α and β parameters. The final parameters selected are in table 4.4

Table 4.4: Parameters for Ford datasets

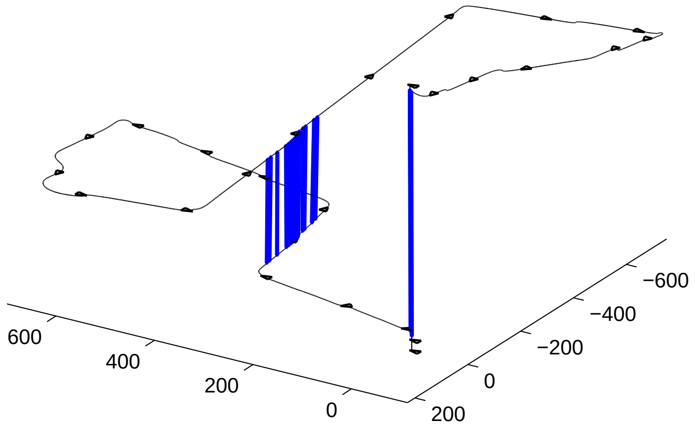
α^+	α^-	β_{3D}	β_{Im}
70%	20%	1.6	2

Evaluation with Ford-2

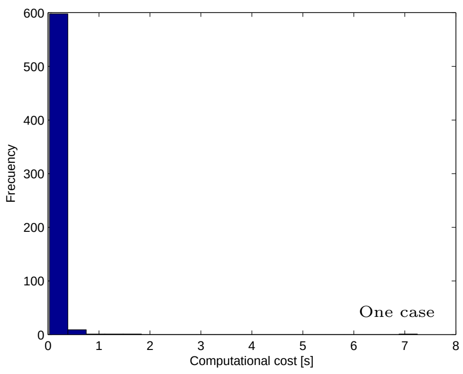
We process the dataset at 10 frames per second in our local mapping and at 1 fps in our place recognition system. In local mapping we select $f_{min} = 100$ and $s_{min} = 5 * fps = 50$ steps resulting in at least $5s$ per local map.

In the local mapping we use features parametrised in 3D Euclidean coordinates when the SURF-points have corresponding 3D information from the 3D lidar. Here we don't use any odometry information. We apply a constant velocity model in which we represent the pose with twelve components in the state vector: three Euclidean coordinates, three Euler angles and six parameters for linear and angular velocities. When the local map is closed and another must be initialised, the velocity components are marginalised from the local map.

With the vocabulary, weights and parameters selected, we run the place recognition over Ford-2 at 1 fps. The loops detected in this process are shown in Fig. 4.15(a). In table 4.5 we can see the average and maximum computational times for this process. Excluding the SURF extraction cost, we plot the histograms of time per step in Fig.4.15(b). We can see that only one place recognition verification consumes more than 1 second.



(a) Revisited places.



(b) Costs.

Figure 4.15: (a) Loops detected by our place recognition system in Ford-2. The ground truth trajectory in black and the loop closures in blue. Axis is in meters and the height represents the time of the experiment. (b) Histogram for the total computational cost without the SURF extraction cost.

Accuracy is evaluated by comparing the estimated trajectory to the position given by an Applanix POS-LV INS with Trimble GPS on board. The absolute trajectory errors were computed in the same way than before with the same toolkit.

In total 122 local maps are built. The result is one final map of 5980 features and 122

Table 4.5: Computational times for our place recognition system (in s) over Ford-2

	SURF extraction	BoW	CRF Matcher $\mathcal{G}_{3D}$	CRF Matcher $\mathcal{G}_{Im}$	Whole System
Mode	1.82	0.07	0.01	0.11	1.92
Average	1.94	0.09	0.19	0.17	2.08
Maximum	3.11	0.32	0.45	6.83	9.39

poses, a state vector of 18672 components and an information matrix with 9466558 non zero entries, 2.7% of density. The final map can be seen in Fig. 4.16.

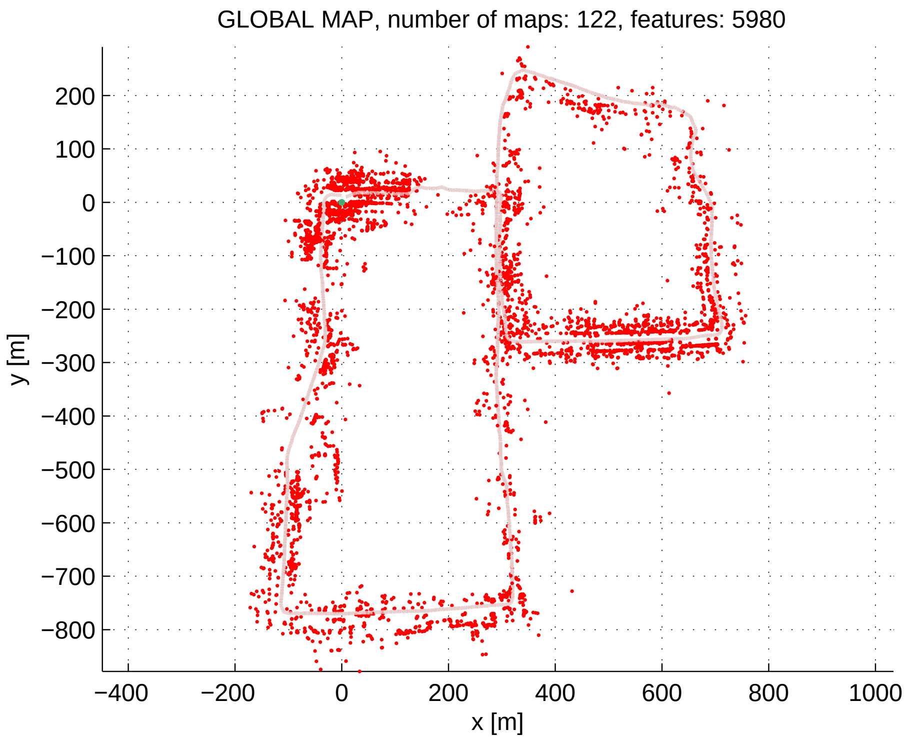


Figure 4.16: Final map for Ford-2 dataset after running our SLAM system.

We show in Fig. 4.17 (left) the cost per step to carry out each process in the CF SLAM algorith: local mapping and joining submaps, and data association for submaps. Local mapping includes the cost of its internal data association (solid light-blue line). The joining submap cost (dashed orange line) and data association of submaps (dash-dot pink line) could be amortised during the next 2^l5 seconds as we show in Fig. 4.17 (bottom-left). The peak in the last step is 39s and 38s for all the pending submap joins and data associations, respectively. The cost is dominated by the local mapping and data association between submaps. That is probably because the lidar has a longer range with many more observations per step increasing the cost of internal data associations and updates in the local mapping. For the same reason, the overlap between adjacent local maps is much larger, then the data association for submaps increase its cost, even though is constant with respect to the size of the experiment. In Fig. 4.17 (right) we show the cumulative cost for the same variables.

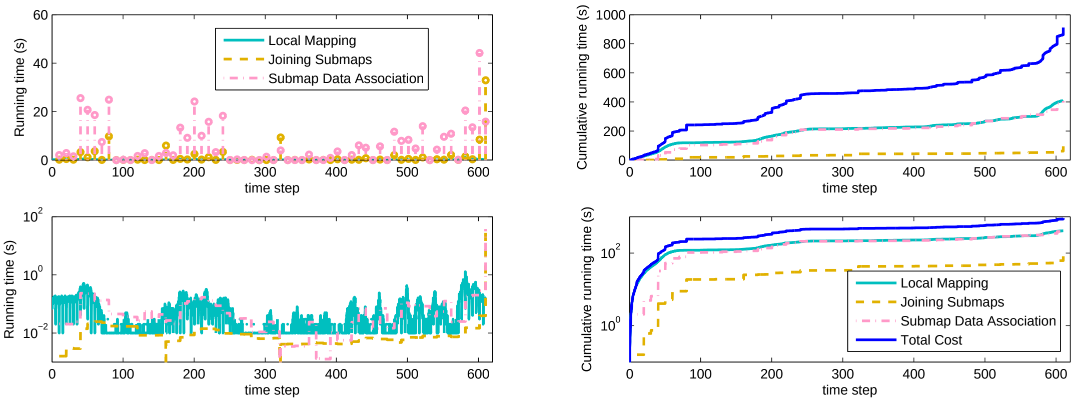


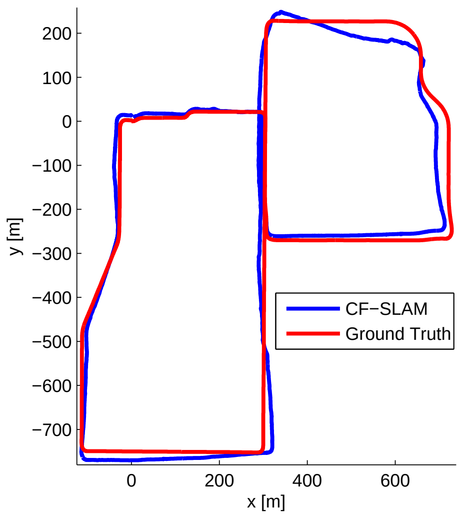
Figure 4.17: Running times for our system decomposed in Local Mapping, Joining submaps and Data Association between submaps. On the left, top: running times per step for our system; bottom in semi-log scale: The same running times but amortising the costs of joining and DA between submaps. On the rigth, top: the cumulative running times including the total cost; bottom: the same cumulative times in semi-log scale.

Fig. 4.18(a) shows the estimated trajectory (solid blue line) and the ground truth (GT, solid red line). They are aligned using the Rawseeds Metrics Computation Toolkit. The histogram of errors in the trajectory are shown in Fig. 4.18(b). Our solution achieves a mean absolute error of 15.89m with 8.85m of standard deviation. A mean precision of 0.4% of the length of the trajectory.

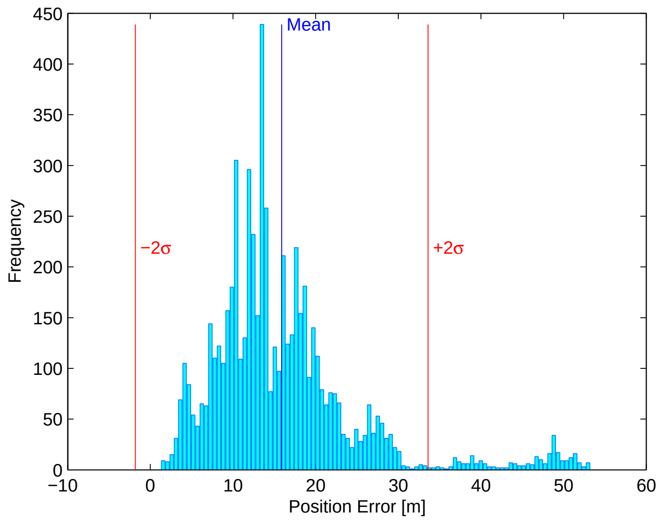
4.3 Discussion

In this chapter we have seen as the CF SLAM and our place recognition system can work together with computational efficiency in the estimation process and robustness in the loop closure events. Thanks to the place recognition system we can compute the data association only in contiguous local maps, bounding its computational cost and making it constant with the size of the final map.

We think that our future implementation in C++ of the system can be run at real time easily in these experiments with the same configurations and even at higher frame rate. Also, this system can be improved with a visual odometry system at higher frame rate.



(a) Trajectory.



(b) Histogram of ATE.

Figure 4.18: Estimated trajectory compared against the ground truth given by the Applanix on the left and robot pose error histogram with 2σ error bounds on the right.

Chapter 5

Conclusions

The ability to understand its environment and know its place within it, simultaneously and automatically, is essential in a robotic platform to adequately navigate, carry out path planning and decision making. The sensor measurements that the robot obtain to perceive its own motion and the environment are inherently imperfect, since both the sensors and actuators on board are noisy. Environments where mobile robots are expected to operate are increasingly large, unstructured and changing. In such environments perceptual ambiguity is highly probable. Some current SLAM systems are able to work in environments up to a certain scale and a tolerance to noise and ambiguity. When that scale or level of ambiguity are overcome, these systems can become inefficient, make mistakes or altogether fail. The contributions of this thesis are focused on extending the boundaries of current SLAM systems in scale and robustness.

Among the many issues to address in SLAM, we have addressed two that stand as crucial, map estimation and place recognition. The estimation should find the most precise configuration for the map elements and robot poses that satisfy the constraints given by data association. Place recognition imposes constraints on the estimation which are usually strong, and greatly informative to achieve higher accuracy. They are so crucial, that an error in one of these constraints can damage the estimation completely.

5.1 Contributions

It is this thesis we have described improvements in these two topics, estimation and place recognition, in efficiency and robustness.

The first contribution of the thesis is focused on the efficiency of the estimation process.

We have proposed a state estimation algorithm with sub-linear computational cost per step, the Combined Filter SLAM algorithm, the most efficient filtering up to now to work on large scale environments. CF SLAM does not sacrifice any information to achieve such efficiency, and its memory requirement remains linear with the size of the environment.

The second contribution is made in the data association problem, specifically in the detection of places revisited by the robot. The system we have proposed is based on the very fast selection of place candidates and very robust verification of these with probabilistic graphical models. Our place recognition system works with image and distance sensors, and makes use of all the information available in a unified framework of inference on probabilistic graphs.

Our contributions have been evaluated in different real scenarios and compared with the state of the art. The efficiency of the CF SLAM has been tested in indoor environments with artificial and natural features, and outdoor environments with natural features. The robustness of our place recognition system has been tested using different stereo cameras on various robotic platforms, and even with hand-held stereo cameras, in diverse environments, and also using omnidirectional cameras in conjunction with a 3D LIDAR. Furthermore, both contributions have been tested together as part of a prototype of a complete SLAM system.

5.2 Future Work

Our contributions allow a robot to operate in larger environments and perceptually more complex. Immediate improvements in our system include taking more advantage of the effort that has already made to compute the graph structures and potentials in the conditional random fields. The currently available information allows to incorporate semantic labeling of the map: the identification of objects such as vehicles, lamp posts, trees, walls, buildings, pedestrians, cyclists, and place classes such as parks, roads, corridors, kitchens. This will allow a better understanding of the environment and also data association via a higher level of abstraction.

Now that the problem of modeling large-scale environments is better understood, there are at least three major technical challenges ahead:

- Maintaining a representation of the environment which is valid over long periods of time. This problem is known as Life-Long Learning or Persistent SLAM.

- The ability to work in dynamic environments. Rather than deal with dynamic objects as outliers, recognize them as active agents of the map with different kinds of dynamics and levels of interaction.
- The ability to detect that past data association constraints were incorrect, and recover the state estimation accuracy.

Efforts along these lines will place us closer to the goal of having safe and useful autonomous robotic systems.

Conclusiones

La localización de un robot y el mapeo del ambiente en el que interactúa de manera simultánea y automática es esencial para el correcto funcionamiento de otras tareas, como los sistemas de navegación, la planificación de trayectorias o la toma de decisión sobre el ambiente a explorar. Los entornos en los que se espera que la robótica móvil sea aplicable son cambiantes, no estructurados y probablemente de gran ambigüedad. Adicional a ello las mediciones que puede hacer el robot sobre su movimiento y su entorno son inherentemente imperfectas, tanto los sensores como los actuadores abordo son ruidosos.

A día de hoy se han propuesto ya sistemas que son capaces de trabajar en este tipo de ambientes hasta una escala y hasta un nivel de tolerancia al ruido y a la ambigüedad de los datos. Cuando dicha escala o nivel de ambigüedad son superados dejan de funcionar, se vuelven ineficientes o cometen errores. Las contribuciones de esta tesis están enfocadas a extender esas fronteras en escala y robustez.

Entre las muchas cuestiones que se deben abordar en el problema del SLAM resaltan dos como las más cruciales, la estimación y el reconocimiento de lugares. La estimación debe encontrar la mejor configuración para el mapa y para las posiciones del robot que satisfagan las restricciones dadas por la asociación de datos. El reconocimiento de lugares impone también restricciones en la estimación y normalmente son restricciones más fuertes, pero así mismo son mas informativas para alcanzar mayor precisión. Son tan cruciales que un error en una sola de estas restricciones puede dañar por completo la estimación.

Contribuciones

Es esta tesis hemos descrito mejoras en estos dos tópicos, estimación y reconocimiento de lugares, en eficiencia y en robustez.

La primera contribución de la tesis está enfocada a la eficiencia del proceso de estimación. Hemos propuesto un algoritmo de estimación de estados con costo computacional sublineal por paso, el Combined Filter SLAM, el algoritmo de filtrado más eficiente a día de hoy para trabajar en entornos a gran escala. CF SLAM no sacrifica ninguna información para lograr tal eficiencia y mantiene un consumo de memoria lineal con el tamaño del entorno.

La segunda contribución esta enmarcada en el problema de asociación de datos, espećficamente en la detección de lugares del entorno ya visitados por el robot. El sistema que hemos propuesto está basado en la muy rápida selección de candidatos y en la muy robusta verificación de éstos con modelos gráficos probabilísticos. Nuestro sistema de reconocimiento trabaja con sensores de imagen y de distancia y hace uso de toda la información en un marco unificado de inferencia sobre grafos probabilistas.

Nuestras contribuciones han sido evaluadas en diferentes escenarios reales y comparadas con el estado del arte. La eficiencia del CF SLAM ha sido probada en entornos de interiores, con características artificiales y naturales, y en entornos exteriores con características naturales extraídas con sensores laser. La robustez de nuestro sistema de reconocimiento de lugares ha sido probada usando diferentes sistemas estéreo sobre diferentes plataformas robóticas, e incluso llevados en la mano, en diferentes y variados entornos; también usando cámaras omnidireccionales en conjunción con un laser 3D.

Aún mas, las dos han sido evaluadas en conjunto como parte de un prototipo de sistema completo de SLAM.

Trabajo Futuro

Nuestras contribuciones permiten a un robot operar en entornos más grandes y difíciles perceptualmente. Mejoras inmediatas en nuestro sistema incluyen aprovechar aún más todo el esfuerzo que ya se ha hecho construyendo y calculando los modelos gráficos y los potenciales en los conditional random fields, con toda la información ya disponible pasaremos a un etiquetado semántico del mapa, detectar objetos como: vehiculos, arboles, paredes, edificios, petaones, ciclistas y clases de lugares como parques, caminos, corredores, cocinas, etc. Esto nos permitirá un mejor entendimiento del entorno e incluso una vía de asociación de datos en un nivel más alto de abstracción.

Pero ahora que el problema de modelar ambientes de gran escala está bien entendido, hay al menos tres grandes retos técnicos por afrontar:

- Ser capaz de mantener una representación del ambiente válida sobre largos periodos de tiempo. Este problema es conocido como Life-Long Learning o Persistent SLAM.

- Ser capaz de trabajar en ambientes dinámicos. Más que tratar los objetos dinámicos como outliers reconocerlos como agentes activos del mapa con diferentes clases de dinámica y de nivel de interacción con él.

- Ser capaz de recuperar la estimación de estados después de restricciones erróneas.

Esfuerzos en este sentido se nos acercan más a la meta de contar con sistemas robticos autnomos seguros y tiles.

Bibliography

Angeli, A., Filliat, D., Doncieux, S., & Meyer, J. (2008). A fast and incremental method for loop-closure detection using bags of visual words. *IEEE Transactions On Robotics, Special Issue on Visual SLAM*, 24:1027–1037.

Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., & Davis, J. (2005). SCAPE: shape completion and animation of people. *ACM Trans. Graph.*, 24(3):408–416.

Bailey, T. & Durrant-Whyte, H. (2006). Simultaneous localization and mapping (SLAM): Part II. *IEEE Robotics & Automation Magazine*, 13(3):108–117.

Bar-Shalom, T. & Fortmann, T. (1988). *Tracking and Data Association.* Academic Press In.

Bar-Shalom, Y., Li, X. R., & Kirubarajan, T. (2001). *Estimation with Applications to Tracking and Navigation.* John Willey and Sons.

Bay, H., Tuytelaars, T., & Gool, L. V. (2006). SURF: Speeded up robust features. In: *Proceedings of the 9th European Conference on Computer Vision*, volume 3951, pages 404–417. Springer LNCS.

Bibby, C. & Reid, I. (2007). Simultaneous localisation and mapping in dynamic environments (slamide) with reversible data associa. In: *Proceedings of Robotics: Science and Systems.*

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer.

Bosse, M., Newman, P. M., Leonard, J. J., & Teller, S. (2004). SLAM in large-scale cyclic environments using the atlas framework. *Int. J. Robotics Research*, 23(12):1113–1139.

Bryson, M. & Sukkarieh, S. (2008). Observability Analysis and Active Control for Airborne SLAM. *IEEE Transactions on Aerospace Electronic Systems*, 44:261–280.

Burgard, W., Fox, D., Hennig, D., & Schmidt, T. (1996). Estimating the absolute position of a mobile robot using position probability grids. In: *Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAi-96).*

Cadena, C., Gálvez-López, D., Ramos, F., Tardós, J., & Neira, J. (2010). Robust place recognition with stereo cameras. In: *IEEE Int. Conf. on Intelligent RObots and Systems.*

Cadena, C., Gálvez-López, D., Tardós, J., & Neira, J. (2011a). Robust place recognition with stereo sequences. *IEEE Trans. Robotics.* submitted.

Cadena, C., McDonald, J., Leonard, J., & Neira, J. (2011b). Place recognition using near and far visual information. In: *18th World Congress of the International Federation of Automatic Control (IFAC).*

Cadena, C. & Neira, J. (2009). SLAM in O(log n) with the Combined Kalman - Information Filter. In: *IEEE Int. Conf. on Intelligent RObots and Systems.*

Cadena, C. & Neira, J. (2010). SLAM in O(log n) with the Combined Kalman-Information Filter. *Robotics and Autonomous Systems*, 58(11):1207–1219.

Chli, M. & Davison, A. (2008). Active matching. In: *European Conference on Computer Vision ECCV 2008*, D. Forsyth, P. Torr, & A. Zisserman, ed., volume 5302 of *Lecture Notes in Computer Science*, pages 72–85. Springer Berlin / Heidelberg.

Christensen, H. & Hager, G. (2008). Sensing and Estimation. In: *Springer Handbook of Robotics*, B. Siciliano & O. Khatib, ed., Springer Handbooks, chapter 4, pages 87–107. Springer.

Civera, J., Davison, A., & Montiel, J. (2007). Inverse Depth to Depth Conversion for Monocular SLAM. In: *Proc. IEEE Int. Conf. Robotics and Automation.*

Civera, J., Grasa, O., Davison, A., & Montiel, J. (2009). 1-point ransac for ekf-based structure from motion. In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 3498 –3504.

Civera, J., Grasa, O. G., Davison, A. J., & Montiel, J. M. M. (2010). 1-point ransac for extended kalman filtering: Application to real-time structure from motion and visual odometry. *J. Field Robot.*, 27:609–631.

Clemente, L., Davison, A. J., Reid, I. D., Neira, J., & Tardós, J. D. (2007). Mapping large loops with a single hand-held camera. In: *Proc. Robotics: Science and Systems.*

Cohn, T. (2007). *Scaling Conditional Random Fields for Natural Language Processing.* PhD thesis, University of Melbourne.

Cummins, M. & Newman, P. (2008). FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. *The International Journal of Robotics Research*, 27(6):647–665.

Cummins, M. & Newman, P. (2010). Appearance-only SLAM at large scale with FAB-MAP 2.0. *The International Journal of Robotics Research.*

Dellaert, F. & Kaess, M. (2006). Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. *Int. J. Robotics Research*, 25(12).

Dissanayake, M. W. M. G., Newman, P., Clark, S., Durrant-Whyte, H. F., & Csorba, M. (2001). A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. Robotics and Automation*, 17(3):229–241.

Douillard, B., Fox, D., Ramos, F., & Durrant-Whyte, H. (2011). Classification and semantic mapping of urban environments. *Int. J. Rob. Res.*, 30:5–32.

Durrant-Whyte, H. & Bailey, T. (2006). Simultaneous localization and mapping: part I. *IEEE Robotics & Automation Magazine*, 13(2):99–110.

Eliazar, A. & Parr, R. (2004). Dp-slam 2.0. In: *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, volume 2, pages 1314–1320 Vol.2.

Eustice, R., Walter, M., & Leonard, J. (2005). Sparse extended information filters: Insights into sparsification. In: *IEEE Int. Conf. on Intelligent RObots and Systems.*

Eustice, R. M., Singh, H., & Leonard, J. J. (2006). Exactly Sparse Delayed-State Filters for View-based SLAM. *IEEE Trans. Robotics*, 22(6):1100–1114.

Fenwick, J., Newman, P., & Leonard, J. (2002). Cooperative concurrent mapping and localization. In: *Robotics and Automation, 2002. Proceedings. ICRA '02. IEEE International Conference on*, volume 2, pages 1810 – 1817 vol.2.

Fischler, M. A. & Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395.

Frese, U. (2006). Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping. *Autonomous Robots*, 21(2):103–122.

Gilbert, J. R., Moler, C., & Schreiber, R. (1992). Sparse matrices in matlab: Design and implementation. *SIAM Journal on Matrix Analysis and Applications*, 13:333–356.

Grimson, W. E. L. (1990). *Object Recognition by Computer: The Role of Geometric Constraints*. The MIT Press.

Grisetti, G., Grzonka, S., Stachniss, C., Pfaff, P., & Burgard, W. (2007a). Efficient Estimation of Accurate Maximum Likelihood Maps in 3d. In: *IROS*.

Grisetti, G., Rizzini, D. L., Stachniss, C., Olson, E., & Burgard, W. (2008). Online Constraint Network Optimization for Efficient Maximum Likelihood Mapping. In: *ICRA*.

Grisetti, G., Stachniss, C., & Burgard, W. (2009). Nonlinear constraint network optimization for efficient map learning. *Intelligent Transportation Systems, IEEE Transactions on*, 10(3):428 –439.

Grisetti, G., Tipaldi, G., Stachniss, C., Burgard, W., & Nardi, D. (2007b). Fast and accurate slam with rao-blackwellized particle filters. *Robotics and Autonomous Systems*, 55(1):30–38.

Handa, A., Chli, M., Strasdat, H., & Davison, A. (2010). Scalable active matching. In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1546 –1553.

Hartley, R. (1997). In defense of the eight-point algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 19(6):580–593.

He, X., Zemel, R. S., & Carreira-Perpiñán, M. Á. (2004). Multiscale conditional random fields for image labeling. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 2, pages 695–702. IEEE Computer Society.

Huang, S. & Dissanayake, G. (2007). Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM. *IEEE Trans. Robotics*, 23(5):1036–1049.

Huang, S., Wang, Z., & Dissanayake, G. (2008a). Sparse local submap joining filters for building large-scale maps. *IEEE Trans. Robotics*, 24:1121–1130.

Huang, S., Wang, Z., Dissanayake, G., & Frese, U. (2008b). Iterated slsjf: A sparse local submap joining algorithm with improved consistency. In: *Proceedings of the Australasian Conference on Robotics and Automation.*

Ila, V., Porta, J., & Andrade-Cetto, J. (2010). Information-based compact pose slam. *Robotics, IEEE Transactions on*, 26(1):78 –93.

Kaess, M. & Dellaert, F. (2009). Covariance recovery from a square root information matrix for data association. *Robotics and Autonomous Systems*, 57(12):1198 – 1210. Inside Data Association.

Kaess, M., Johannsson, H., Roberts, R., Ila, V., Leonard, J., & Dellaert, F. (2011). iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In: *IEEE Intl. Conf. on Robotics and Automation, ICRA.*

Kaess, M., Ranganathan, A., & Dellaert, F. (2008). iSAM: Incremental Smoothing and Mapping. *IEEE Trans. on Robotics, TRO*, 24(6):1365–1378.

Kim, B., Kaess, M., Fletcher, L., Leonard, J., Bachrach, A., Roy, N., & Teller, S. (2010). Multiple relative pose graphs for robust cooperative mapping. In: *IEEE Intl. Conf. on Robotics and Automation, ICRA*, pages 3185–3192.

Klein, G. & Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In: *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, ISMAR '07, pages 1–10. IEEE Computer Society.

Koller, D. & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques.* MIT Press.

Konolige, K. & Agrawal, M. (2008). Frameslam: From bundle adjustment to real-time visual mapping. *Robotics, IEEE Transactions on*, 24(5):1066 –1077.

Konolige, K., Bowman, J., Chen, J., Mihelich, P., Calonder, M., Lepetit, V., & Fua, P. (2010a). View-based maps. *The International Journal of Robotics Research*, 29(8):941–957.

Konolige, K., Grisetti, G., K
"ummerle, R., Burgard, W., Limketkai, B., & Vincent, R. (2010b). Efficient sparse pose adjustment for 2d mapping. In: *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 22 –29.

Kümmerle, R., Grisetti, G., Strasdat, H., Konolige, K., & Burgard, W. (2011). g2o: A general framework for graph optimization. In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In: *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA.

Leonard, J. & Feder, H. (2001). Decoupled stochastic mapping. *IEEE Journal of Oceanic Engineering*, 26(4):561–571.

Leonard, J. & Newman, P. (2003). Consistent, convergent and constant-time SLAM. In: *Int. Joint Conf. Artificial Intelligence*.

Liao, L., Fox, D., & Kautz, H. (2007). Extracting places and activities from gps traces using hierarchical conditional random fields. *The International Journal of Robotics Research*, 26(1):119.

Lim, E. H. & Suter, D. (2007). Conditional random field for 3d point clouds with adaptive data reduction. In: *Cyberworlds, 2007. CW '07. International Conference on*, pages 404 –408.

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110.

Maimone, M., Cheng, Y., & Matthies, L. (2007). Two Years of Visual Odometry on the Mars Exploration Rovers. *Journal of Field Robotics*.

McDonald, J., Kaess, M., Cadena, C., Neira, J., & Leonard, J. (2011). 6-DOF Multi-session Visual SLAM using Anchor Nodes. In: *European Conference on Mobile Robotics, ECMR*.

Mei, C., Sibley, G., Cummins, M., Newman, P., & Reid, I. (2011). Rslam: A system for large-scale mapping in constant-time using stereo. *International Journal of Computer Vision*, 94:198–214. 10.1007/s11263-010-0361-7.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2002). FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. In: *Proceedings of the AAAI National Conference on Artificial Intelligence*. AAAI.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. (2003). FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. In: *Int. Joint Conf. Artificial Intelligence*.

Montiel, J. M. M., Civera, J., & Davison, A. J. (2008). Unified inverse depth parametrization for monocular SLAM. *IEEE Trans. Robotics*. To appear.

Murphy, K. P., Weiss, Y., & Jordan, M. I. (1999). Loopy belief propagation for approximate inference: An empirical study. In: *In Proceedings of Uncertainty in AI*, pages 467–475.

Mutambara, A. & Al-Haik, M. (1997). State and information space estimation: a comparison. *American Control Conference, 1997. Proceedings of the 1997*, 4:2374–2375 vol.4.

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

Newman, P., Sibley, G., Smith, M., Cummins, M., Harrison, A., Mei, C., Posner, I., Shade, R., Schroeter, D., Murphy, L., Churchill, W., Cole, D., & Reid, I. (2009). Navigating, recognizing and describing urban spaces with vision and lasers. *The International Journal of Robotics Research*, 28(11-12):1406–1433.

Ni, K. & Dellaert, F. (2010). Multi-Level Submap Based SLAM Using Nested Dissection. In: *IEEE Int. Conf. on Intelligent RObots and Systems*.

Ni, K., Steedly, D., & Dellaert, F. (2007). Tectonic SAM: Exact, Out-of-Core, Submap-Based SLAM. In: *2007 IEEE Int. Conf. on Robotics and Automation*.

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

Olson, E. (2009). Recognizing places using spectrally clustered local matches. *Robotics and Autonomous Systems*, 57(12):1157–1172.

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

Pandey, G., McBride, J. R., & Eustice, R. M. (2011). Ford campus vision and lidar data set. *The International Journal of Robotics Research*.

Paul, R. & Newman, P. (2010). FAB-MAP 3D: Topological mapping with spatial and visual appearance. In: *Proc. IEEE Int. Conf. Robotics and Automation*, pages 2649 –2656.

Paz, L. M., Guivant, J., Tardós, J. D., & Neira, J. (2007). Data association in *O(n)* for divide and conquer SLAM. In: *Proc. Robotics: Science and Systems*.

Paz, L. M., Tardós, J. D., & Neira, J. (2008). Divide and conquer: Ekf slam in O(n). *IEEE Trans. Robotics*, 24(5):1107–1120.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc.

Piniés, P., Paz, L. M., Gálvez-López, D., & Tardós, J. D. (2010). Ci-graph simultaneous localization and mapping for three-dimensional reconstruction of large and complex environments using a multicamera system. *Journal of Field Robotics*, 27:561–586.

Pinies, P., Paz, L. M., & Tardos, J. D. (2009). Ci-graph: An efficient approach for large scale slam. In: *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 3913 –3920.

Quattoni, A., Wang, S., Morency, L.-P., Collins, M., & Darrell, T. (2007). Hidden conditional random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(10):1848–1852.

Ramos, F., Fox, D., & Durrant-Whyte, H. (2007). CRF-Matching: Conditional Random Fields for Feature-Based Scan Matching. In: *Robotics: Science and Systems (RSS)*.

Ramos, F., Kadous, M. W., & Fox, D. (2008). Learning to associate image features with CRF-Matching. In: *ISER*, pages 505–514.

Ranganathan, A. & Dellaert, F. (2011). Online probabilistic topological mapping. *The International Journal of Robotics Research*, 30(6):755–771.

RAWSEEDS (2009). Robotics advancement through Webpublishing of sensorial and elaborated extensive data sets (project FP6-IST-045144). http://www.rawseeds.org/rs/datasets.

Scharstein, D. & Pal, C. (2007). Learning conditional random fields for stereo. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 0, pages 1–8. IEEE Computer Society.

Sibley, G., Mei, C., Reid, I., & Newman, P. (2009). Adaptive relative bundle adjustment. In: *Robotics Science and Systems (RSS)*.

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

Smith, R., Self, M., & Cheeseman, P. (1988). A Stochastic Map for Uncertain Spatial Relationships. In: *Robotics Research, The Fourth Int. Symposium*, O. Faugeras & G. Giralt, ed., pages 467–474. The MIT Press.

Steder, B., Grisetti, G., & Burgard, W. (2010). Robust place recognition for 3d range data based on point features. In: *Proc. IEEE Int. Conf. Robotics and Automation*, pages 1400 –1405.

Strasdat, H., Montiel, J. M. M., & Davison, A. (2010). Scale drift-aware large scale monocular slam. In: *Proceedings of Robotics: Science and Systems*.

Tappen, M. F., Liu, C., Adelson, E. H., & Freeman, W. T. (2007). Learning gaussian conditional random fields for low-level vision. In: *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 0, pages 1–8. IEEE Computer Society.

Tardós, J. D., Neira, J., Newman, P. M., & Leonard, J. J. (2002). Robust Mapping and Localization in Indoor Environments using Sonar Data. *Int. J. Robotics Research*, 21(4):311–330.

Thrun, S. (2002). Particle Filters in Robotics. In: *Proceedings of Uncertainty in AI (UAI)*.

Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. The MIT Press.

Thrun, S. & Leonard, J. (2008). Simultaneous Localization and Mapping. In: *Springer Handbook of Robotics*, B. Siciliano & O. Khatib, ed., Springer Handbooks, chapter 37, pages 871–889. Springer.

Tipaldi, G. & Ramos, F. (2009). Motion clustering and estimation with conditional random fields. In: *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 872 –877.

Törnqvist, D., Schön, T. B., Karlsson, R., & Gustafsson, F. (2009). Particle filter slam with high dimensional vehicle model. *J. Intell. Robotics Syst.*, 55(4-5):249–266.

Triggs, B., McLauchlan, P., Hartley, R., & andrew Fitzgibbon (2000). Bundle adjustment – A modern synthesis. In: *Vision Algorithms: Theory and Practice*, W. Triggs, A. Zisserman, & R. Szeliski, ed., LNCS, pages 298–375. Springer Verlag.

Valgren, C. & Lilienthal, A. J. (2007). SIFT, SURF and seasons: Long-term outdoor localization using local features. In: *Proceedings of the European Conference on Mobile Robots (ECMR)*, pages 253–258.

Valgren, C. & Lilienthal, A. J. (2010). Sift, surf & seasons: Appearance-based long-term localization in outdoor environments. *Robotics and Autonomous Systems*, 58(2):149 – 156. Selected papers from the 2007 European Conference on Mobile Robots (ECMR '07).

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., & Tardós, J. (2009). A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems.*

Williams, B., Klein, G., & Reid, I. (2007). Real-time SLAM relocalisation. In: *Proc. International Conference on Computer Vision.*

Williams, S. B., Dissanayake, G., & Durrant-Whyte, H. (2002). An efficient approach to the simultaneous localisation and mapping problem. In: *Proc. IEEE Int. Conf. Robotics and Automation*, volume 1, pages 406–411.