Divide and Conquer:
EKF SLAM in $O(n)$

Lina María Paz Pérez

Doctoral thesis supervised by José Neira Parra

UNIVERSIDAD DE ZARAGOZA
CENTRO POLITÉCNICO SUPERIOR DE INGENIEROS
DEPARTAMENTO DE INFORMÁTICA E INGENIERíA DE SISTEMAS
MARIA DE LUNA, 3, ADA BYRON BUILDING
SEPTEMBER 2008

Resumen

Esta tesis ha sido desarrollada, en parte, dentro el contexto del proyecto *"EXPRES: Técnicas de Exploración Automatizada para Aplicaciones de Rescate"* [1]. En este proyecto se ha abordado el desarrollo de técnicas de percepción y control implementadas sobre robots de rescate para la exploración de entornos con alta probabilidad de riesgo o en difíciles condiciones de navegación. Tareas tales como el manejo e integración de información obtenida por sensores, la exploración y planificación de movimientos y el control seguro de robots en entornos no estructurados han sido formulados como objetivos específicos. Todas estas tareas confían en gran medida en una estimación precisa de la posición del robot y un mapa del entorno construido con exactitud. Los algoritmos de Localización y Construcción de Mapas Simultánea, (SLAM, de sus siglas en inglés), tratan estos dos problemas en un solo proceso de estimación.

Una de las soluciones más populares propuestas en la literatura considera el problema de SLAM como un proceso de filtrado incremental en el cual el estado del sistema, formado por la(s) posición(es) actual(es) del robot y de elementos que conforman la estructura del entorno (mapa), es representado en tiempo discreto mediante un vector $\mathbf{x}_k$ y una matriz de covarianza $\mathbf{P}_k$ como medida de incertidumbre. Basándose en una representación del mapa con elementos geométricos característicos, el *Filtro de Kalman Extendido (EKF)* ha sido la herramienta usada para estimar el vector de estado. Su objetivo es integrar incrementalmente medidas parciales afectadas por ruido, las cuales son obtenidas del entorno mediante un sensor, y estimar las cantidades representadas por $\mathbf{x}_k$ y $\mathbf{P}_k$. A primera vista, cualquier algoritmo de SLAM desarrollado en las últimas dos décadas y basado en el Filtro de Kalman podría ser aplicado para obtener una buena estimación. Desafortunadamente, los entornos a gran escala pueden ser considerados en tareas de rescate incluyendo trayectorias con presencia de bucles largos. Este problema acompañado del crecimiento cuadrático de la covarianza de la estimación, ($O(n^2)$ en el número de características n), hace que el filtro de Kalman requiera de costosas actualizaciones cada vez que nueva información parcial deba ser integrada. Por otra parte, SLAM es un problema no lineal, en cuyo caso aplicar algoritmos como el EKF tienen la limitación de reducir la precisión y consistencia debido al efecto que las linealizaciones producen sobre las estimaciones del robot y el mapa.

[1] Proyecto DPI2003-07986 fundado en parte por la Dirección General de Investigación de España.

Este trabajo de tesis se enfoca en el desarrollo de una nueva familia de algoritmos, capaces de resolver las principales limitaciones del SLAM: su complejidad computacional y su consistencia. Con este propósito, los procesos de actualización de la matriz de covarianza, asociación de datos, localización global y cerrado de bucles han sido abordados con un extenso análisis permitiendo formular estrategias robustas.

La primera parte de este libro hace una revisión de las técnicas recientes que han impulsado un importante progreso en el contexto de complejidad computacional en SLAM. El Filtro de Kalman Extendido es introducido abordando sus limitaciones mediante respectivos análisis de coste computacional y consistencia.

Los métodos avanzados de *Local Mapping* basados en la construcción y fusión de mapas locales, son el principal objeto de estudio de esta tesis, ya que permiten direccionar las limitaciones de complejidad en SLAM. Una de sus mayores ventajas radica en la disponibilidad para obtener covarianzas exactas sin llevar a cabo ningún tipo de aproximación para su estimación (excepto la linealizacin). Este tema es de gran importancia para la solución de los problemas de asociación de datos y localización global de forma robusta. El análisis propuesto en esta tesis describe principalmente el algoritmo *Map Joining*, como un método avanzado de fusión y discute su dependencia sobre el tamaño de los mapas locales para lograr optimizar el costo computacional y la consistencia.

Como resultado de este análisis, proponemos *Divide and Conquer (D&C) SLAM*, una nueva estrategia que fusiona mapas locales en forma jerárquica llevando a cabo actualizaciones en $O(n)$. Esta estrategia, basada en técnicas de Fusión de Mapas (Map Joining), reduce el costo cuadrático del Filtro de Kalman y reporta ventajas de consistencia en el cálculo de la estimación. En esta tesis se ilustran las ventajas de usar D&C SLAM cuando diferentes tipos de trayectorias son consideradas. En un análisis detallado se discuten las limitaciones del algoritmo debido a la presencia de solapes o regiones de asociación de tamaño variable entre mapas locales del mismo tamaño.

Abordando las limitaciones anteriormente expuestas, se presenta *Randomized Joint Compatibility* (RJC), un algoritmo implementado para resolver el problema de la asociación de datos en D&C en tiempo lineal. El estudio realizado distingue entre una asociación de datos secuencial, esta es la forma convencional usada en EKF-SLAM y adoptada para construir mapas locales, y la asociación de datos requerida para fusionar mapas locales del mismo tamaño con técnicas como Map Joining. El método propuesto se basa en el empleo de técnicas de división en celdas para la representación del espacio comprendido por el mapa (técnicas de *Tessellation*), y técnicas de consenso sobre muestreos aleatorios ($RANSAC$). Ambos permiten que características asociadas entre dos mapas sean obtenidas en $O(n)$. El desempeño de los métodos ha sido validado sobre entornos no estructurados a gran escala usando sensores de laser y odometria.

El problema de la Localización Global, conocido como la determinación de la posición del robot después de la ocurrencia de fallos en el sistema de navegación (problema del robot secuestrado) es también abordado. De forma similar, las técnicas de división por celdas

han sido adaptadas de forma tal que, dado un conjunto de medidas parciales del entorno, la posición del robot es obtenida en tiempo lineal.

Adicionalmente, el trabajo presentado en esta tesis recibe una segunda motivación claramente relacionada con el proyecto de investigación *"SLAM6DOF: Sistemas portátiles de Localización y Construcción de mapas para entornos complejos a gran escala"* [2]. El interés en desarrollar soluciones basadas en técnicas de Visión por Computador está experimentando un gran crecimiento debido a que la utilización de cámaras permite abaratar costos por compararción con sensores laser tradicionalmente usados en robótica. Las cámaras además proporcionan importante información del entorno tal como texturas, las cuales pueden ser utilizadas exitosamente en tareas de reconocimiento o asociación de datos. Particularmente, las aplicaciones con una sola cámara conocidas en la literatura como *Monocular SLAM* han recibido más atención gracias a los avances demostrados en visión activa y su desempeño en tiempo real. Desafortunadamente, el factor de escala real constituye su principal limitación debido a los problemas de no-observabilidad registrados en investigaciones previas. La desventaja viene del hecho de que con una sola cámara no es posible determinar el tamaño real de los elementos de un entorno. Sensores como las cámaras estero constituyen una segunda opción ya que proporcionan la información necesaria para recuperar dicho parámetro. Sin embargo, a pesar de la ventaja de proporcionar información de distancia de corto alcance, las cámaras binoculares comerciales poseen una separación pequeña entre los dos ejes ópticos y por lo tanto, la profundidad de objetos lejanos no puede ser estimada. Esta dificultad aparece también cuando una de las cámaras que conforman el par estéreo falla o cuando los objetos se encuentran realmente lejos del sensor. Recientes investigaciones sugieren el uso de técnicas complementarias que puedan manejar información de profundidad cuando esta esté disponible sin desperdiciar ningún otro tipo de información que pueda usarse para la estimación de orientación.

Como contribución en las aplicaciones de SLAM, esta tesis aborda el uso de sensores de visión, particularmente cámaras binoculares para proporcionar un sistema confiable, automático y preciso de modelado tridimensional de un entorno. Esta tesis describe las características de un sistema visual de SLAM en 6 grados de libertad para la construcción de mapas en entornos de interior y exterior a gran escala. El sistema calcula el mapa completo de un entorno a partir de la construcción de mapas locales usando Divide and Conquer SLAM, de tal forma que es posible operar regularmente en tiempo constante con actualizaciones periódicas lineales. Paralelamente la información de profundidad y orientación es considerada de acuerdo a su disponibilidad combinando la parametrización del *inverso de la profundidad* junto con una parametrización convencional en 3D en el mismo vector de estado.

En la parte final de este libro se discuten las principales conclusiones del trabajo presentado en esta tesis, se detallan las publicaciones resultantes, y se trazan líneas de trabajo futuro.

[2]Proyecto DPI2006-13578 fundado en parte por la Dirección General de Investigación de España.

Abstract

This thesis work has been developed, in part, in the context of the project *"EXPRES: Automated Exploration Techniques for Rescue Applications"*[3]. The project involves the development of perception-control techniques implemented on rescue robots in environments with high probability of risk or under poorly navegable conditions. Tasks such as the management and integration of sensor information, motion planning and exploration, and robot control safety in non-structured scenarios, have been considered as specific aims. Most of these tasks rely strongly, not only on a precise estimation of the robot location, but also on an accurate map of the surrounding environment. Simultaneous Localization and Mapping (SLAM) algorithms deal directly with both problems merged into one single estimation process.

The most popular solution proposed in the literature considers the SLAM problem as an incremental filtering process where the state of the system is represented in discrete time by a vector containing both the vehicle location and the map structure. In a feature-based framework for map representation, the Extended Kalman Filter (EKF) is used to compute an estimation of a state vector $\mathbf{x}$ by integrating noisy sensor measurements obtained from the environment structure, together with a covariance matrix $\mathbf{P}$ that represents a measurement of precision of the estimates. At first sight, almost any EKF-based SLAM algorithm developed in 80's and 90's decades could be used to provide a good estimated solution. Unfortunately, hundreds of meters in loops trajectories can be considered in the task of searching for victims in rescue applications. This fact, together with the quadratic growth of the estimated covariance with the number of features n, makes the EKF algorithm perform expensive updates when new information is integrated. In addition, SLAM is a non-linear problem, thus applying the EKF method has the limitation of reducing precision and consistency due to the effect that linearizations yield on both vehicle and map estimates.

This thesis focuses primarily on the development of a new family of algorithms capable of overcoming the main limitations of SLAM: *computational complexity and consistency*. The issues of map updates, global localization, data association and loop closure have been addressed with extensive analysis in order to formulate robust strategies.

This book makes a review of recent techniques that drive important progress in the context

[3]Project DPI2003-07986 funded in part by the Dirección General de Investigación of Spain.

of SLAM. EKF SLAM is introduced in this thesis, pointing out its limitations by means of computational complexity and consistency analysis.

Advanced methods based on Local Mapping are mainly studied as they allow us to address the computational complexity limitation of SLAM. As an advantage, they have the feasibility of providing covariances to deal robustly with data association and global localization problems. The proposed analysis mainly describes *Map Joining SLAM* and discusses its dependency on the local map size in order to optimize the cost and maximize consistency.

Based on the promising results in computational cost and consistency obtained for the Map Joining algorithm, we formulate *Divide and Conquer (D&C) SLAM*, a novel strategy that uses hierarchical Map Joining to produce local maps such that updates are performed in $O(n)$, reducing the quadratic cost of standard EKF. We illustrate the advantages of using D&C SLAM when a robot carries out different types of trajectories, as well as its complications to solve data association due to the presence of overlaps of variable size between local maps.

To address the data association problem, we present *Randomized Joint Compatibility* (RJC), an algorithm implemented to address the data association problem of D&C SLAM also in linear time. The analysis realized distinguishes between sequential data association, the conventional framework adopted to build local maps, and Map Joining data association, a potentially different problem that must be solved in order to join two local maps of the same size. The proposed method is based on spatial tessellation and RANSAC techniques so that associated features from one map to another are obtained in $O(n)$.

We also tackle the Global Localization problem, the determination of the robot position after system failures (also known as the kidnapping problem). Similarly, tessellation techniques that use grid sampling structures are considered in such a manner that the robot position is obtained in linear time given a set of partial measurements.

Additionally, this thesis is closely related to the project *"SLAM6DOF: Portable Simultaneous Localization and Mapping Systems for Large and Complex Environments"*[4]. The interest in Computer vision solutions is undergoing a high growth in SLAM since cameras are cheaper than lasers. Furthermore, cameras provide texture rich information of the environment. Monocular applications have received particular attention in recent years thanks to the advances in active vision and real time applications. Unfortunately, the scale factor issue has been a big limitation due to the known unobservability problems. Sensors such as stereo cameras become a second option in the sense that they can provide the information required to recover the scale factor. If so, no other information such as gyroscopes or accelerometers is required. Comercial binocular sensors provide a small separation between cameras. Then, depth for distant objects cannot be extracted. The latter case arises either when one of the cameras cannot see the objects or when they are really far away. Recent research suggests the use of complementary techniques that can manage depth information when available, without throwing out orientation information that

[4]Project DPI2006-13578 funded by the Dirección General de Investigación of Spain.

this can be considered to refine the cameras' angular estimation. As part of the research evolution on SLAM applications, this thesis concentrates in the use of vision sensors, in particular stereo cameras to provide with a reliable, automatic and precise 3D modeling of the environment.

Part of this thesis is devoted to demonstrate the robustness and scalability of a 6DOF Visual SLAM system in indoor and outdoor urban environments. The proposed system computes the full map from a sequence of local maps using the Divide and Conquer algorithm, allowing constant time operation most of the time, with linear time updates to compute the full map. As contribution, depth and orientation information are both considered when available by combining inverse depth and 3D parameterizations in the same state vector. Also, the concept of conditionally independent local maps is studied as a valuable tool to share information related to the camera motion and common features that are tracked. The use of this strategy will allow the system to share scale information between local maps. Also, it will enable the system to create new local maps consistently when a constant velocity model is implemented. In this way the vehicle maintains information of the cameras' prior state.

Finally, we discuss the main conclusions related to the work presented in this thesis; we list the main publications resulting from this research and draw lines of possible future work.

Table of Contents

List of Figures

Chapter 1

Introduction: The EKF Solution to SLAM

1.1 Introduction

The problem of Simultaneous Localization and Mapping (SLAM) has received special attention in mobile robotics since it allows a robot to navigate autonomously in unknown environments; more over, it can provide knowledge about the spatial relationships among objects combining uncertain spatial information. From this point of view, SLAM can be seen as the problem of estimating the locations of the robot at the same time the environment structure is being modeled [Durrant-Whyte 06; Bailey 06]. Only partial noisy measurements of the environment are available and, as it frequently occurs in most applications, no prior information of the vehicle location exists. Typically, the environment consists of a set of stationary landmarks (natural features that are used for identification purposes), like those extracted from data gathered with a range finder or a vision sensor. As an example, one can consider lines, planes segmented from lasers or just visual corners corresponding to very salient features in an image scene.

Focusing on feature-based SLAM, we find that a family of methods and applications developed in the last decades, since the seminal works [Chatila 85; Smith 86; Smith 88; Leonard 91], are based on the popular Extended Kalman Filter (EKF) [Castellanos 99a; Leonard 92]. The EKF recursively (or incrementally) estimates the *joint probability distribution* over a set of random variables representing the current vehicle pose $\mathbf{x}_{R_k}$ and n feature locations $\mathbf{x}_{F_{k1\ldots n}}$ at time k which are expressed in the state vector $\mathbf{x}$. The final estimation is a Multivariable Gaussian distribution constituted by its mean $\hat{\mathbf{x}}$ together with the covariance matrix $\mathbf{P}$ related to the error in the estimation (see algorithm 1).

Alternatively, sampling based methods have reported considerable success in challenging environments [Montemerlo 02; Thrun 02; Montemerlo 03; Grisetti 05; Stachniss 05]. They have been generally implemented using a combination of grid-based environment modelling and particle filtering techniques. Both Extended Kalman Filter based methods (EKF-SLAM) and sampling based methods can robustly and efficiently deal with

environments of *limited size*. In such circumstances, SLAM is largely considered solved.

1.2 Motivation

1.2.1 The Computational Complexity problem of SLAM

In recent years, the interest of researchers has focused on reducing the computational cost of performing SLAM in large environments. Some processes associated to the move-sense-update cycle of EKF SLAM are linear in the number of map features n: vehicle prediction and inclusion of new features [Castellanos 99b; Guivant 01]. Because of the correlations between all pair of map features represented by the covariance $\mathbf{P}$, the size of its entries grows as $O(n^2)$. Consequently, EKF-SLAM techniques have a *first* important limitation: the computational cost of updating the map at each step is of order $O(n^2)$. On the other hand, in sampling based methods the number of required samples scales exponentially in the dimension of the vehicle location uncertainty. Thus the application of any MonteCarlo method to SLAM, such as FastSLAM [Montemerlo 02] and Rao-Blackwellised Particle Filtering [Murphy 01], scales with difficulty.

The impractical use of the EKF solution to SLAM in large scale environments due to its quadratic computational complexity has become a subject of much interest in research. Postponement [Knight 01], the Compressed EKF filter [Guivant 01], the Sparse Weight Kalman Filter [Julier 01b] and Map Joining SLAM [Tardós 02] are alternatives that work on local areas alternatives that work on local areas of the stochastic map and are essentially constant most of the time, although they require periodical $O(n^2)$ updates. Recently, researchers have pointed out the approximate sparseness of the Information matrix $\mathbf{Y}$, the inverse of the full covariance matrix $\mathbf{P}$. This suggests using the Extended Information Filter, the dual of the Extended Kalman Extended Information Filter (SEIF) algorithm [Thrun 04] (SEIF) algorithm [Thrun 04] approximates the Information matrix by a sparse form that allows $O(1)$ updates on the information vector. Nonetheless, data association becomes more difficult when the state and covariance matrix are not available, and the approximation produces overconfident estimations of the state [Eustice 05]. This overconfidence is overcome by the Exactly Sparse Extended Information Filter (ESEIF) [Walter 04] with a strategy that produces an exactly sparse Information matrix with no introduction of inaccuracies through sparsification, but discarding odometry information.

The Thin Junction Tree Filter algorithm [Paskin 03] works on the Gaussian graphical model represented by the Information matrix, and achieves high scalability by working on an *approximation*: a reduction of the width of a given Junction Tree representing the map distribution is carried out by choosing a *maximum likelihood projection* or variable contraction so that weak links are broken. The Treemap algorithm [Frese 06] is a closely related technique: in cases where there are many overlapping features, an optional optimization uses a weak link breakage policy. Recently, the Exactly Sparse Delayed State Filter [Eustice 06] and Square Root Smoothing and Mapping ($\sqrt{SAM}$) [Dellaert 06], provided the insight that the full SLAM problem, the complete vehicle trajectory plus the

map, is sparse in information form (although ever increasing) allowing the use of sparse linear algebra techniques. The last work provides an intuitive analysis based on graph theory to solve SLAM when this is seen as a Least Squares problem. In this way, the use of well known Factorization techniques together with reordering graph methods, allows us to find an efficient solution for the non-linear system, although in batch form.

In the same research line of Smoothing and Mapping, the Tectonic SAM algorithm [Ni 07] provides a local mapping version to reduce the computational cost. However, the method remains a batch algorithm and covariance is not available to solve data association. The Fast Incremental Smoothing and Mapping (iSAM) algorithm [Kaess 07] addresses the data association problem by recovering the exact marginal covariance (just the current vehicle position and the map which is the same for EKF SLAM) using QR-factorization on the general Jacobian matrix (Jacobians for the full trajectory are included). Only when the vehicle does not revisit previously mapped areas (closing a loop for example), the factor R allows us to recover the state as well as the marginal covariance in $O(n)$ using back-substitution, being n the number of trajectory positions and map feature locations.

1.2.2 The Consistency problem of SLAM

A second important limitation of standard EKF SLAM is the effect that linearizations have in the precision and consistency of the final vehicle and feature estimates. Early studies and counter examples point out that linearizations introduce errors in the estimation process that reduce precision and can render the result inconsistent, in the sense that the computed state covariance does not represent the real error in the estimation [Julier 01a]. Among other things, this complicates data association, which is based on contrasting predicted feature locations with observations made by the sensor. Thus, important processes in SLAM like loop closing are complicated. All algorithms for EKF SLAM based on efficiently computing an approximation of the EKF solution will inevitably suffer from this consistency problem [Paskin 03; Frese 06]. The Unscented Kalman Filter [Julier 97] avoids linearizations via a parametrization of means and covariances through selected points to which the nonlinear transformation is applied. Unscented SLAM has been shown to have improved consistency properties [Martinez-Cantin 05]. These solutions however ignore the computational complexity problem.

In a recent paper [Huang 07] the authors prove that convergence properties known to hold for linear EKF SLAM [Dissanayake 01] hold for nonlinear EKF SLAM only when Jacobians (and thus linearizations) are computed around the ground truth solution, which is in general unfeasible. It is also proven that the use of Jacobians computed around the estimated value may result in overconfident estimates. In particular, when the robot orientation uncertainty is large, the extent of inconsistency is significant. Likewise, when the robot orientation uncertainty is small, the extent of inconsistency is insignificant. The authors point out that since the robot orientation error is the main source of inconsistency, algorithms that use local submapping, where the robot orientation error remains small, have the potential to improve consistency. In this thesis we confirm the consistency improvement offered by one of such local submapping algorithms.

Algorithm 1 : m = `ekf_slam(steps)`

$$\begin{aligned} \mathbf{z}_0, \mathbf{R}_0 &= get\_measurements \\ \hat{\mathbf{x}}_0, \mathbf{P}_0 &= new\_map(\mathbf{z}_0, \mathbf{R}_0) \end{aligned}$$

for $k = 1$ to steps **do**

$$\begin{aligned} & \{EKF\ Movement\ and\ Prediction\ step\} \\ \hat{\mathbf{x}}_{R_k}^{R_{k-1}}, \mathbf{Q}_k &= get\_odometry \\ \hat{\mathbf{x}}_{k|k-1}, \mathbf{F}_k, \mathbf{G}_k &= prediction(\hat{\mathbf{x}}_{k-1}, \hat{\mathbf{x}}_{R_k}^{R_{k-1}}) \\ \mathbf{P}_{k|k-1} &= \mathbf{F}_k \mathbf{P}_{k-1} \mathbf{F}_k^T + \mathbf{G}_k \mathbf{Q}_k \mathbf{G}_k^T \end{aligned} \tag{1.1}$$

$$\begin{aligned} & \{Observing\ the\ environment\} \\ \mathbf{z}_k, \mathbf{R}_k &= get\_measurements \\ \mathcal{H}_k, \mathbf{H}_{\mathcal{H}_k} &= data\_assoc(\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}, \mathbf{z}_k, \mathbf{R}_k) \end{aligned}$$

$$\begin{aligned} & \{EKF\ Update\ step\} \\ \mathbf{S}_{\mathcal{H}_k} &= \mathbf{H}_{\mathcal{H}_k} \mathbf{P}_{k|k-1} \mathbf{H}_{\mathcal{H}_k}^T + \mathbf{R}_{\mathcal{H}_k} && (1.2) \\ \mathbf{K}_{\mathcal{H}_k} &= \mathbf{P}_{k|k-1} \mathbf{H}_{\mathcal{H}_k}^T / \mathbf{S}_{\mathcal{H}_k} && (1.3) \\ \mathbf{P}_k &= (\mathbf{I} - \mathbf{K}_{\mathcal{H}_k} \mathbf{H}_{\mathcal{H}_k}) \mathbf{P}_{k|k-1} && (1.4) \\ \nu_{\mathcal{H}_k} &= \mathbf{z}_k - \mathbf{h}_{\mathcal{H}_k}(\hat{\mathbf{x}}_{k|k-1}) && (1.5) \\ \hat{\mathbf{x}}_k &= \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_{\mathcal{H}_k} \nu_{\mathcal{H}_k} && (1.6) \\ & \{Adding\ new\ elements\} \\ \hat{\mathbf{x}}_k, \mathbf{P}_k &= add\_feat(\hat{\mathbf{x}}, \mathbf{P}_k, \mathbf{z}_k, \mathbf{R}_k, \mathcal{H}_k) \end{aligned}$$

end for
`return` $\mathbf{m} = (\mathbf{x}_k, \mathbf{P}_k)$

The goal of this thesis is to propose algorithms that do not sacrifice precision in order to limit computational cost. Thus, a significant part of this thesis contains a detailed explanation of EKF-based SLAM methods that use submapping techniques [Tardós 02]. A detailed analysis of the Extended Kalman Filter is useful because many of the new approaches developed for real time applications in large environments are based on EKF, and more benefit can be extracted from it.

1.3 The EKF SLAM algorithm

The EKF SLAM algorithm (see alg. 1) has been widely used for mapping and localization. Several authors have described the computational complexity of this algorithm

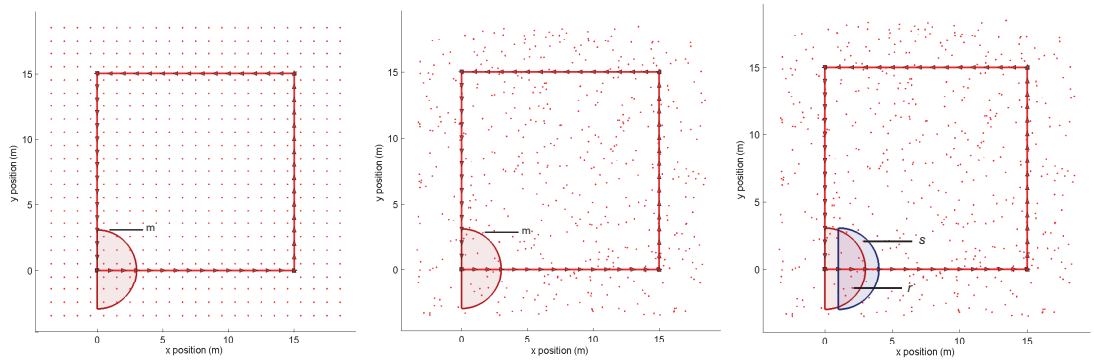


Figure 1.1: Simulated exploration experiment: the environment to be mapped consists of n features $\{\mathbf{f}_1 \cdots \mathbf{f}_n\}$ distributed uniformly either: systematically (left); or randomly (middle). The vehicle is provided with a sensor whose range allows us to obtain a subset of m measurements of the environment features at a time k. Also, an exploration trajectory is considered (it could include loop closing situations) in which the vehicle stop to obtain measurements at equally spaced intervals. This allows us to reobserve r features seen in the previous step, i.e., at each k step there will be $s = m - r$ measurements corresponding to features previously unseen.

[Castellanos 99b; Guivant 01]. With the purpose of comparing EKF SLAM with advanced SLAM algorithms, in this section we briefly analyze its main operations and computational complexity.

1.4 Computational complexity of EKF SLAM

For simplicity, assume that in the environment being mapped features are distributed more or less uniformly (Fig. 1.1). If the vehicle is equipped with a sensor of limited range and bearing, the amount of measurements obtained at any location will be more or less constant. Assume that at some step k:

- the map contains n features,
- the sensor provides m measurements,
- r of which correspond to re-observed features,
- $s = m - r$ which correspond to new features.

This corresponds to an exploratory trajectory, where the size of the map is proportional to the number of steps that have been carried out.

1.4.1 Computational cost per step

The computational complexity of carrying out the move-sense-update cycle of algorithm `ekf_slam` at step k involves the computation of the *predicted map* $\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}$, which

requires obtaining also the computation of the corresponding Jacobians $\mathbf{F}_k, \mathbf{G}_k$, and the *updated map* $\mathbf{x}_k, \mathbf{P}_k$, which requires the computation of the corresponding Jacobian $\mathbf{H}_k$, the Kalman gain matrix $\mathbf{K}_k$, as well as the innovation ν_k, and its covariance $\mathbf{S}_k$ (the complexity of data association is analyzed in chapter 4).

The fundamental fact regarding computational complexity in standard EKF SLAM is that, given a sensor of limited range and bearing, Jacobians matrices $\mathbf{F}_k, \mathbf{G}_k$ and $\mathbf{H}_k$ are sparse [Castellanos 99b; Guivant 01; Dellaert 06]. Their computation is $O(1)$. But more importantly, since they take part in the computation of both the predicted and updated map, the computational cost of eqs. (1.1) to (1.6) can also be reduced. Appendix A summarizes the main EKF matrix operations and establish the order in which they must be implemented so that its actual cost minimized.

The use of sparse matrices is well justified, for example, in the case of the innovation covariance matrix $\mathbf{S}_k$ in eq. (2) (see fig. 1.2 top). Without considering sparseness, the computation of this $r \times r$ matrix would require $rn^2 + r^2 n$ multiplications and $rn^2 + r^2 n + r^2$ sums, that is, $O(n^2)$ operations. But given that matrix $\mathbf{H}_k$ is sparse, with an effective size of $r \times c$, the computation requires $rcn + r^2 c$ multiplications and $rcn + r^2 c + r^2$ sums, that is, $O(n)$ operations. Similar analysis leads to the conclusion that the cost of computing both the predicted covariance $\mathbf{P}_{k|k-1}$ and the Kalman gain matrix $\mathbf{K}_k$ is $O(n)$, and that the greatest cost in an EKF SLAM update is the computation of the covariance matrix $\mathbf{P}_k$, which is $O(n^2)$. Thus, the computational cost per step of EKF SLAM is quadratic on the size of the map:

$$C_{EFK,k} = O(n^2) \tag{1.7}$$

1.4.2 Total computational cost

Considering the simulated experiment previously described, during an exploratory trajectory a constant number of s new features are added to the map at each step. Thus to map an environment of size n, n/s steps are required. The total cost of EKF SLAM will then be:

$$\begin{aligned} C_{EKF} &= \sum_{k=1}^{n/s} O\left((ks)^2\right) \\ &= \sum_{k=1}^{n/s} O\left(k^2\right) \end{aligned}$$

The square power summation is known to be:

$$\sum_{k=1}^{n} k^2 = (1/6)n(n+1)(2n+1)$$

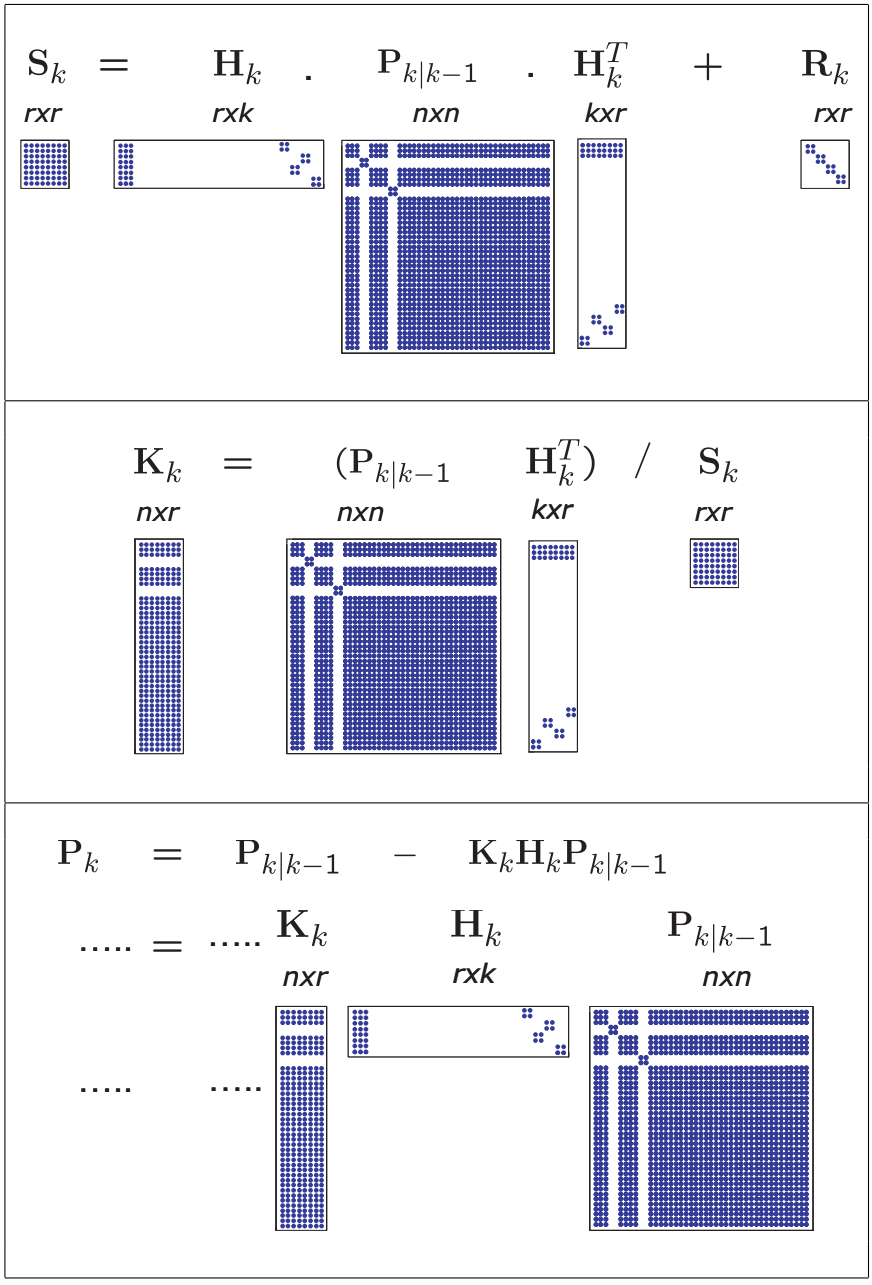


Figure 1.2: Computation of the innovation covariance $\mathbf{S}_k$ matrix (top): given that the effective size of the Jacobian matrix $\mathbf{H}_k$ is $r \times c$, the computation requires $O(n)$ operations ($rcn + r^2c$ multiplications and $rcn + r^2c + r^2$ sums). In a similar way, the computations of the Kalman gain $\mathbf{K}_k$ matrix (middle) requires $O(n)$ operations, and the covariance Matrix $\mathbf{P}_k$ (bottom) requires $O(n^2)$ operations.

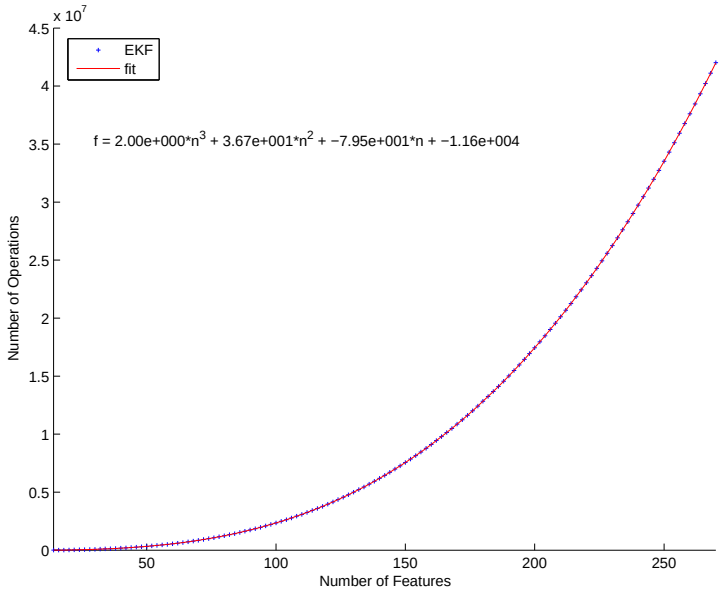


Figure 1.3: Computational cost of standard EFK SLAM approach implemented in MATLAB over $n = 127$ total steps.

Thus, the total cost of EKF SLAM is cubic:

$$\begin{aligned} C_{EKF} &= O\left(\frac{(n/s)(n/s+1)(2n/s+1)}{6}\right) \\ &= O\left(\frac{n^3}{s^3}\right) \\ &= O(n^3) \end{aligned} \tag{1.8}$$

Figure 1.3 shows in a simple MATLAB implementation of the computational cost when the vehicle traverses an environment with $n = 270$ total point features, 135 on each side of the straight path trajectory. Features are $2m$ apart. The range and bearing sensor has a range such that $m = 14$ features are perceived from a certain location. The robot moves $2m$ at each step. The vehicle thus leaves $s = 2$ features behind at each step and thus reobserves $r = 12$. It can be seen how the function increases in order of $O(n^3)$ as the polynomial curve fitting shows on the crosses marks obtained after the algorithm execution.

1.5 Consistency of EKF-SLAM

In the problem of estimating a parameter, an estimator is said to be *consistent* if the estimate converges to the true value in a stochastic sense [Bar-Shalom 01]. This implies

that there is an increasing amount of information about the parameter that asymptotically reduces to zero the uncertainty about its true value. When working with the EKF estimator, the uncertainty of the current estimate of the state $\hat{\mathbf{x}_k}$ at k step is associated to the covariance matrix $\mathbf{P}_k$. Thus, consistency determines how well the $\mathbf{P}_k$ represents the error in the estimate: $\tilde{\mathbf{x}_k} = \mathbf{x}_k - \hat{\mathbf{x}}_k$. For practical evaluation proposes, a state estimator is called consistent if its state estimation error satisfies the following two properties:

$$\mathbf{E}\left[\mathbf{x_k} - \hat{\mathbf{x}}_\mathbf{k}\right] = \mathbf{E}\left[\tilde{\mathbf{x_k}}\right] = \mathbf{0} \tag{1.9}$$

$$\mathbf{E}\left[(\mathbf{x_k} - \hat{\mathbf{x}}_\mathbf{k})(\mathbf{x_k} - \hat{\mathbf{x}}_\mathbf{k})^\mathbf{T}\right] = \mathbf{P}_k \tag{1.10}$$

where eq. (1.9) is a condition for the estimator of *unbiasedness* required to guarantee a zero-mean estimation error. Eq. (1.10) means that the actual *Mean Square Error* matches the filter calculated covariances. When estimating the state of non linear dynamic system, no convergence of its estimate can be guaranteed. Given that SLAM is a nonlinear problem, consistency checking is of paramount importance [Castellanos 04]. When the ground true solution $\mathbf{x}$ for the state variables is available, a statistical test for filter consistency can be carried out based on the Normalized Estimation Error Squared (NEES), defined as:

$$D^2 = (\hat{\mathbf{x}} - \hat{\mathbf{x}})^T \, \mathbf{P}^{-1} \, (\mathbf{x} - \hat{\mathbf{x}}) \tag{1.11}$$

The above quadratic form allows us to verify simultaneously both consistency properties. The left-hand side of eq. (1.11) is also known as the *Mahalanobis distance*. The results of applying this test point out when the estimation error is out of the probability concentration ellipsoid, obtained by cutting the tails of the error multivariate Gaussian density $\tilde{\mathbf{x}_k}$. We also can perform checking consistency with a derived chi-squared test:

$$D^2 \leq \chi^2_{r,1-\alpha} \tag{1.12}$$

where $r = dim(\mathbf{x})$ and α is the desired significance level (usually 0.05). If we define the consistency index of a given estimation ($\hat{\mathbf{x}}$, $\mathbf{P}$) with respect to its true value $\mathbf{x}$ as:

$$CI = \frac{D^2}{\chi^2_{r,1-\alpha}} \tag{1.13}$$

when $CI < 1$, the estimation is consistent with ground truth, and when $CI > 1$, the estimation is inconsistent (optimistic) with respect to ground truth.

Notice that in EKF-SLAM, the computational cost depends neither on sensor nor vehicle error. On the other hand, if they were large, data association would produce more expensive operations. But *map consistency* greatly depends on sensor and vehicle error.

To illustrate the EKF SLAM consistency, we run algorithm 1 such that the robot travels in a loop trajectory. Figure 1.4 illustrates the evolution of the estimated robot covariance. For one such runs for this particular example, the final vehicle covariance does not enclose

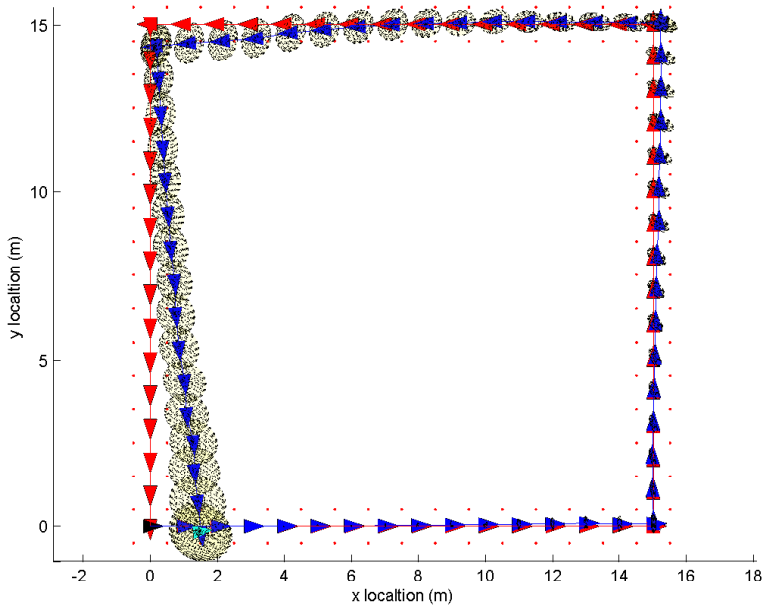


Figure 1.4: Consistency divergency on a loop trajectory.

the starting position. For the EKF SLAM iterative process this means that no loop closing constrains will be deleted and can not be applied to perform a map correction. The result mainly shows the divergency of the EKF SLAM algorithm.

1.6 Thesis Contributions

The main contribution of this thesis is to provide a family of algorithms capable of overcoming the main limitations of SLAM: *computational complexity and consistency*. The issues of map updates, global localization, data association and loop closure have been addressed in order to formulate robust strategies. The results obtained have been presented to the research community in high quality conferences and journals. This section presents a review of each chapter as follows:

Chapter 2 Local Mapping techniques have shown to be a valuable option to address the complexity SLAM problem without sacrificing precision. This chapter describes *Map Joining* SLAM, a known algorithm that allows the sequential fusion of local maps of a given size to produce a full map represented in a global reference. In order to analyze the complexity and consistency properties of Map joining SLAM, we study its cost dependency with the local map size. The derivation of its total cost, knowing the total environment size and the characteristics of the sensors, is used as a tool to determine the optimal size of the local maps required to minimize the total computational cost and maximize map consistency.

Chapter 3 In this chapter *Divide and Conquer* SLAM (D&C SLAM) algorithm, a new proposal to perform Simultaneous Localization and Mapping using the Extended Kalman Filter, is presented. D&C SLAM overcomes the two fundamental limitations of standard EKF SLAM: **1-** the computational cost per step is reduced from $O(n^2)$ to $O(n)$ (consequently the total SLAM is reduced from $O(n^3)$ to $O(n^2)$); **2-** the resulting vehicle and map estimates have better consistency properties than standard EKF SLAM in the sense that the computed state covariance adequately represents the real error in the estimation. Unlike many current large scale EKF SLAM techniques, this algorithm computes an exact solution, without relying on approximations or simplifications to reduce computational complexity. Also, estimates and covariances are available when needed by data association without any further computation. Both cost and consistency characteristics allow to extend the range of environments that can be mapped in real time using EKF.

Chapter 4 Given that D&C SLAM relies on Map Joining techniques to merge local maps, the problem of data association has to be analyzed, not just from the point of view of sequentially building a local map (e.g. the conventional framework adopted in EKF SLAM), but also from the point of view of finding correspondences when the joining process is carried out. Classical algorithms such as *Individual Compatibility* **IC** and *Join Compatibility* **JCBB** [Neira 01], may be too costly for the joining process. One can use **JCBB** in the internal building process of each local map. However, its exponential behavior when traversing the association hypothesis, even in a *Branch and Bound* framework, might yield catastrophic costs for D&C SLAM process. To ensure the linear cost of the data association at any level of the SLAM algorithm, a tesellation-based algorithm is introduced in this chapter to test *Individual compatibility* efficiently, and a *Randomized* version of the *Join Compatibility* **RJC** is derived. These two algorithms allow to carry out data association in linear time for D&C SLAM.

Chapter 5 Global localization, the determination of the vehicle location in a previously mapped environment with no other prior information is another important issue in SLAM studied here. It is shown that, using a grid sampling representation of the configuration space (mainly used in MonteCarlo methods), it is possible to evaluate all vehicle location hypotheses in the environment (up to a certain resolution) with a computational cost that is bilinear: linear both in the number of map features and in the number of sensor measurements. A *pairing-driven* algorithm that considers only individual measurement-feature pairings is proposed. In contrast with current correspondence space algorithms, it avoids searching in the exponential correspondence space. It uses a voting strategy that accumulates evidence for each vehicle location hypothesis, assuring robustness to noise in the sensor measurements and environment models. The general nature of the proposed strategy allows the consideration of different types of features and sensor measurements. Its performance is also compared to *location-driven* algorithms where the solution space is usually randomly sampled. It is shown that the proposed pairing-driven technique is computationally

more efficient in proportion to the density of features in the environment.

Chapter 6 This chapter is devoted to demonstrate the robustness and scalability of a Visual SLAM system with D&C in indoor and outdoor urban environments. In order to evaluate the D&C SLAM algorithm in visual applications, this chapter describes the feature extraction process using stereo cameras and the sensor model used with the corresponding jacobians derivation for filter updating. Depth and orientation information are both considered when available by combining inverse depth and 3D parameterizations in the same state vector. The concept of conditionally independent local maps is studied as a tool available to share information related to the camera motion and common features that are tracked. The use of this strategy will allow the system to share scale information between local maps. Also, it will enable the system to create new local maps consistently when a constant velocity model is implemented. In this way the vehicle maintains information of its prior state. The system computes the full map from a sequence of local maps using the Divide and Conquer algorithm, allowing constant time operation most of the time, with periodical linear time updates.

Chapter 7 This chapter discusses the conclusions of this thesis work and addresses points for future work.

Chapter 2

Advanced EKF SLAM: Local Mapping Algorithms

In this chapter we show how optimize the computational cost and maximize consistency in EKF-based SLAM for large environments. We analyze the combined solution of Local Mapping with Map Joining in a way that the total cost of computing the final map is minimized compared to standard EKF-SLAM. For a given environment size and sensor range, we can determine the optimal size of the local maps required to minimize the total computational cost and maximize map consistency. The motivation of this analysis is described in a map building experiment in our lab, and the statistical significance of the proposed method is validated using Monte Carlo simulations.

2.1 Introduction

Local mapping techniques have been proposed as computationally efficient alternatives to EKF SLAM. Instead of working on a full global map all the time as EKF SLAM does, this techniques build a sequence of local maps of limited size. When the size of the current local map reaches some limit size, the map is closed and stored, and a new local map is created. This allows us to maintain the computational cost constant most of the time, while working on the current local map. Local maps are then joined together into a global map that should be equivalent to the map obtained by the standard EKF-SLAM approach [Knight 01; Guivant 01; Williams 02; Tardós 02; Guivant 03]. There are also a number of methods that use local maps for SLAM [Leonard 03; Bosse 04; Estrada 05] with alternative joining steps. Some of these algorithms include suboptimal or approximate solutions such as Decoupled Stochastic Mapping (DSM) [Leonard 01], Constant Time SLAM (CTS) [Leonard 03], the ATLAS framework [Bosse 04] and Hierarchical SLAM [Estrada 05]. These algorithms sacrifice precision in the resulting estimation of the map in order to maintain the computational cost linear in the size of the map at worst.

There are also alternative solutions that do not carry out approximations, such as Map

Joining SLAM [Tardós 02] and the Constrained Local Submap Filter [Williams 01]. How to decide the size of each local map remains unanswered in these methods. Actually, in none of the Local Mapping methods is this question answered. However, we are interested in algorithms that do not sacrifice precision in order to limit computational cost, so we concentrate on these two.

To compare local map sequencing with standard EKF-SLAM, we performed a map building experiment, using a robotized wheelchair equipped with a SICK laser scanner. The vehicle was hand-driven along a loop of about $250m$ in a populated indoor/outdoor environment in the Ada Byron building of our campus. The laser scans were segmented to obtain lines using the RANSAC technique. The global map obtained using the classical EKF-SLAM algorithm is shown in figure 2.1, top. At this point, the vehicle was very close to the initial starting position, closing the loop. The figure shows that the vehicle estimated location has some $10m$ error and the corresponding 95% uncertainty ellipses are ridiculously small, giving an inconsistent estimation. Due to these small uncertainty, the data association algorithm was unable to properly detect the loop closure. This corroborates the results obtained with simulations in [Castellanos 04]: in large environments the map obtained by EKF-SLAM quickly becomes inconsistent, due to linearization errors.

What happens if we divide the full map in N smaller local maps and then join them together? In order to try to answer this question, we divided the environment in 25 smaller maps, and so the same dataset was processed to obtain independent local maps at fixed trajectory intervals of about $10m$. The local maps were joined and fused obtaining the global map shown in figure 2.1, bottom. In this case the loop was correctly detected by data association and the map obtained seems to be consistent. Furthermore, the computational time was about 50 times smaller that the standard EKF approach. Thus, there is a strong reduction in computational cost and increase in consistency of the resulting map.

This experiment raises the following questions: given a certain environment size and sensor characteristics (1) is there a *computationally optimal* local map size, such that the total computational cost is minimal? (2) is there a *consistency optimal* local map size, such that the consistency of the final map is maximal? (3) do these optimals coincide?

In this chapter we try to answer these questions. We believe that any local mapping method can also benefit in terms of computation cost as well as map consistency from the analysis proposed here. In section 2.2 we describes Map Joining algorithm. Then, in section 2.3 we describe the local map sequencing problem for a given environment size and feature density, sensor characteristics and size of local map. In section 2.4 we detail its related computational cost and derive the optimal solution. We validate this solution using a simulated experiment. In section 2.5, we study the obtained solution with respect to map consistency and error. Monte Carlo simulations are used to show that the solution of minimal computational cost is also the one that provides the map of maximal consistency. Finally, in section 2.6 we draw a discussion of Sequential Local Supmapping.

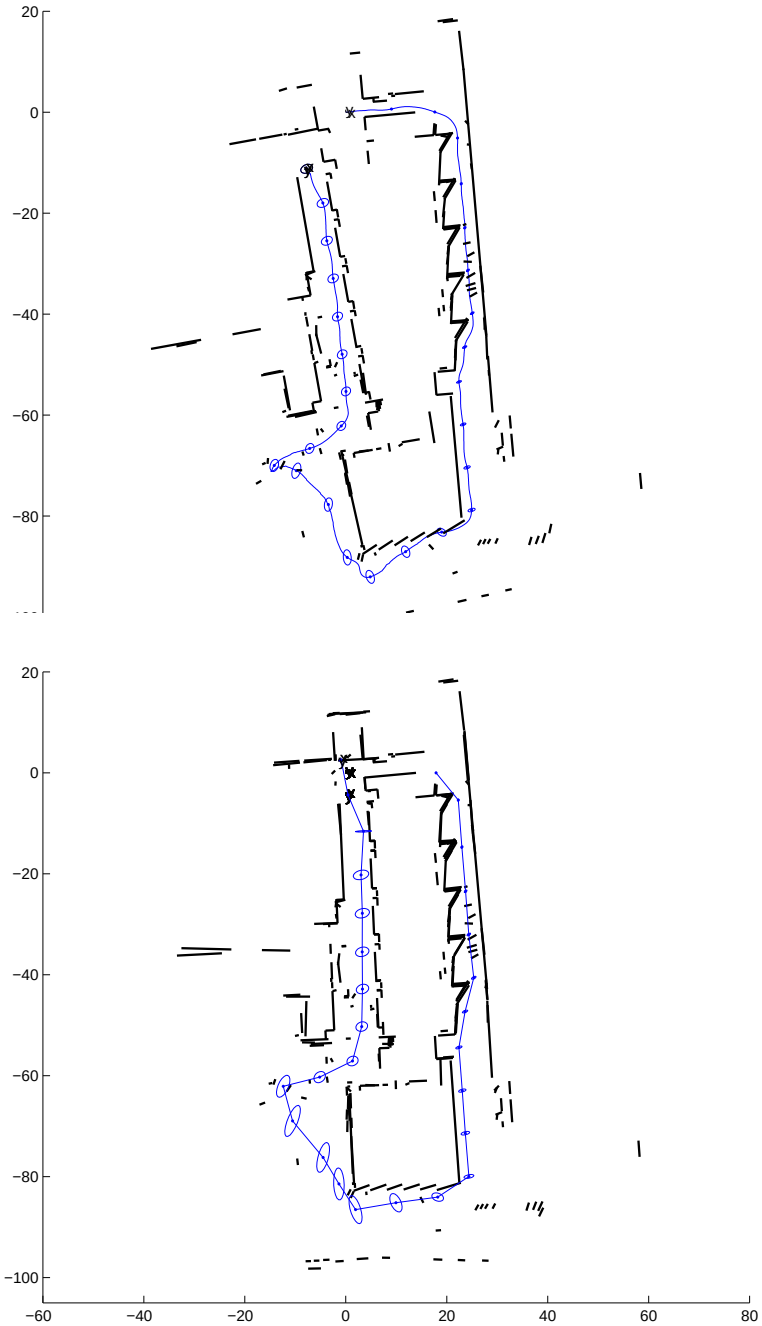


Figure 2.1: Global maps obtained using the standard EKF-SLAM algorithm (top) and local map joining for $N = 25$ local maps(bottom).

2.2 Local Map Sequencing with Map Joining SLAM algorithm

Map Joining SLAM (and similarly, the Constrained Local Submap Filter) is an EKF-based algorithm in which a sequence of independent local maps of a limited size p is produced using the standard EKF SLAM algorithm. Map independence is guaranteed by construction: once the current local map is closed, a new local map is initialized with zero covariance using the current vehicle pose as the base reference for the new map. These local maps are later fused using a map joining procedure to produce a single final stochastic map. Algorithm 2 details the map joining procedure. For two consecutive local maps $\mathbf{m}_{i\ldots j}$ and $\mathbf{m}_{j\ldots k}$, computed in steps $i \ldots j$ and $j \ldots k$ respectively, map joining computes the resulting map $\mathbf{m}_{i\ldots k}$ for all steps $i \ldots k$ in the following way:

- Both maps are simply stacked together, with the features of each in each local base reference. Thanks to map independence, the cross-covariance between both maps is zero.
- Data association is carried out to find common features between both maps.
- Using a modified version of the EKF update equations the map is optimized by fusing common features.
- All features are transformed to the base reference of the first map.

A more detailed explanation of this procedure and its notation can be found in [Paz 07a] as well as in the Appendix B. It is worth noting that Map Joining SLAM never revisits a prior local map. Instead, a new local map is created, which will be joined later with the previous map to obtain a global map which includes all available information.

An analysis similar to that of fig 1.2 in chapter 1 shows that for local maps of limited size, the computational cost of Map Joining SLAM is $O(n^2)$ on the size of the resulting global map, again being the most expensive operation the update of the covariance matrix $\mathbf{P}$ (see fig. 2.2). However, map joining takes place only in few steps when a given local map has reached its limit size. In all other steps, only a local map is being updated, with a computational cost of $O(1)$.

Algorithm 3 carries out Map Joining SLAM: `ekf_slam` is used to compute a local map $\mathbf{m}_k$ of a given limit size (or for a given limit number of steps). This local map is joined to a global map $\mathbf{m}$ by means of the `join` function in a sequential fashion. The process continues until the environment is completely covered. The reader can find a similarity between the Map Joining process and the standard move-sense-update cycle of EKF SLAM, in which a local map behaves as a set of observations. The difference lies in that the local map elements are not independent, and therefore correlations between the current vehicle location and local features are involved.

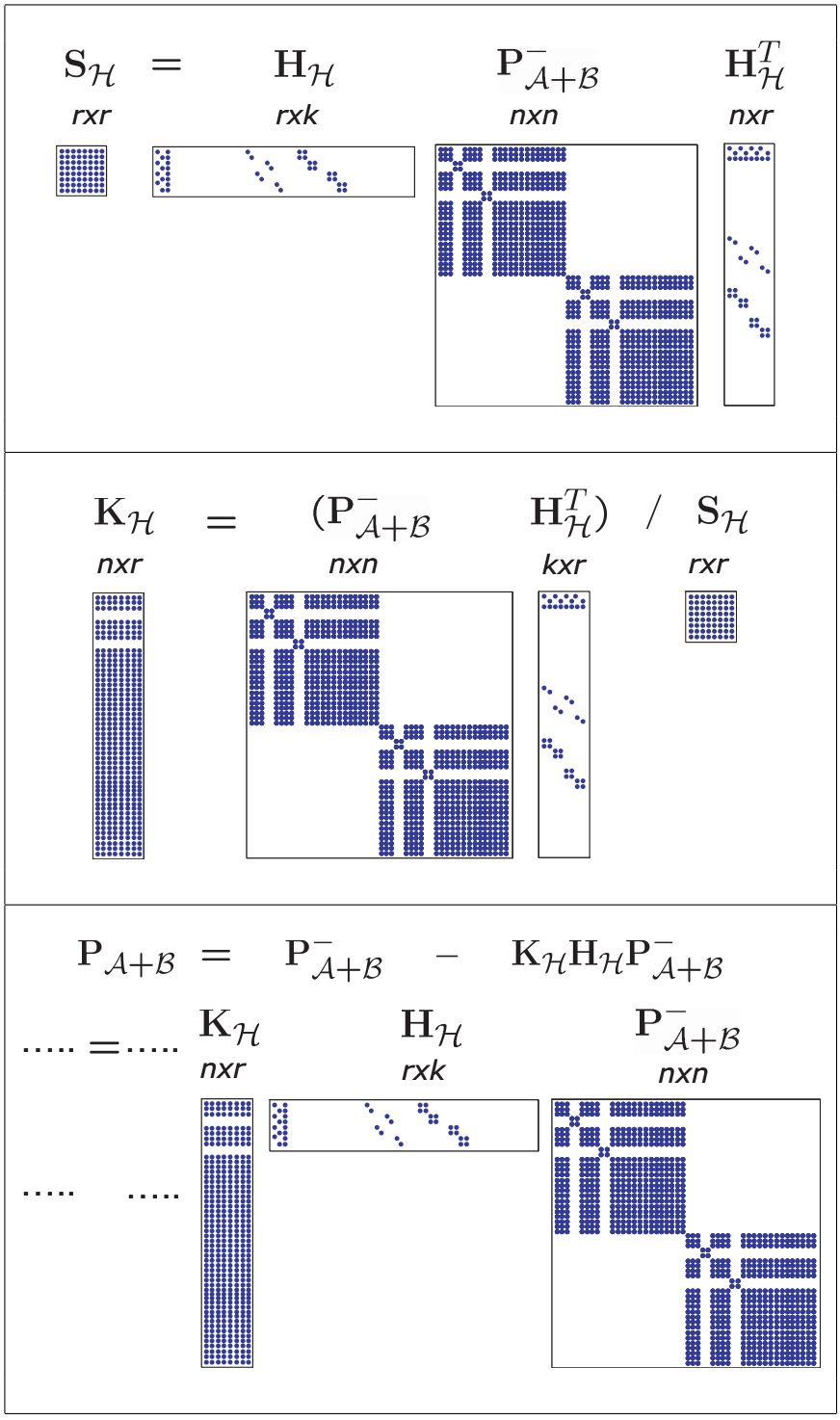


Figure 2.2: Map Joining Computational complexity. Computation of the innovation covariance $\mathbf{S}_{\mathcal{H}}$ matrix (top): the computation requires $O(n)$ operations ($rn^2 + r^2n$ multiplications and $rn^2 + r^2n + r^2$ sums). The Gain Filter $\mathbf{K}_{\mathcal{H}}$ matrix is $O(n)$ as well (middle) and the Covariance Matrix $\mathbf{P}$, requires $O(n^2)$ (bottom).

Algorithm 2 : $\mathbf{m}_{i...k} =$ `join_function`$(\mathbf{m}_{i...j}, \mathbf{m}_{j...k})$

$$
\begin{aligned}
& \{joining\ maps\ step\} \\
\hat{\mathbf{x}}^-_{i...k} &= map\_joining(\hat{\mathbf{x}}_{i...j}, \hat{\mathbf{x}}_k) \\
\hat{\mathbf{P}}^-_{i...k} &= blkdiag(\hat{\mathbf{P}}_{i...j}, \hat{\mathbf{P}}_{j...k}) \\
& \{data\ association\ step\} \\
\mathcal{H}, \mathbf{H}_{\mathcal{H}} &= data\_assoc(\hat{\mathbf{x}}_{i...j}, \hat{\mathbf{x}}_{j...k}, \hat{\mathbf{P}}_{i...j}, \hat{\mathbf{P}}_{j...k})
\end{aligned}
$$

$$
\begin{aligned}
& \{Update\ step\} \\
\mathbf{S}_{\mathcal{H}} &= \mathbf{H}_{\mathcal{H}} \mathbf{P}^-_{i...k} \mathbf{H}^T_{\mathcal{H}} && (2.1) \\
\mathbf{K}_{\mathcal{H}} &= \mathbf{P}^-_{i...k} \mathbf{H}^T_{\mathcal{H}} \left(\mathbf{S}_{\mathcal{H}}\right)^{-1} && (2.2) \\
\hat{\mathbf{x}}^+_{i...k} &= \hat{\mathbf{x}}^-_{i...k} - \mathbf{K}_{\mathcal{H}} \mathbf{H}_{\mathcal{H}} (\hat{\mathbf{x}}^-_{i...k}) && (2.3) \\
\mathbf{P}^+_{i...k} &= (\mathbf{I} - \mathbf{K}_{\mathcal{H}} \mathbf{H}_{\mathcal{H}}) \mathbf{P}^-_{i...k} && (2.4)
\end{aligned}
$$

$$
\begin{aligned}
& \{Global\ transformation\ step\} \\
\hat{\mathbf{x}}_{i...k} &= transformation\_step(\hat{\mathbf{x}}^+_{i...k}) \\
\mathbf{P}_{i...k} &= \frac{\partial \hat{\mathbf{x}}_{i...k}}{\partial \hat{\mathbf{x}}^+_{i...k}} \mathbf{P}^+_{i...k} \left(\frac{\partial \hat{\mathbf{x}}_{i...k}}{\partial \hat{\mathbf{x}}^+_{i...k}} \right)^T
\end{aligned}
\qquad (2.5)
$$

`return` $\mathbf{m}_{i...k} = (\mathbf{x}_{i...k}, \mathbf{P}_{i...k})$

2.3 The Local Map size dependency

In order to analyze the computational cost involved in performing SLAM using local map sequencing, we consider the simulated exploration experiment described in the previous chapter: the environment to be mapped consists in n equally spaced features $\{\mathbf{f}_1 \cdots \mathbf{f}_n\}$. We are provided with a sensor whose range allows us to obtain measurements of m of the environment features at a time in this environment. We carry out a straight forward exploration trajectory (with no loop closing) in which we stop to obtain measurements at equally spaced intervals. This allows us to reobserve r features seen in the previous step, i.e., at each step there will be $s = m - r$ measurements corresponding to features previously unseen. At step 1 measurements in the first measurement set $\mathbf{z}_1$ will correspond to features $1 \dots m$, measurements $\mathbf{z}_2$ at the second step will correspond to features $s + 1 \dots s + m$, and thus at step i, the last measurement of $\mathbf{z}_i$ will correspond to feature $\mathbf{f}_{f(i)}$, where:

Algorithm 3 : $\mathbf{m}$ = `map_joining_slam`
`sequential implementation of map joining.`

```
m = ekf_slam(steps)
{
Main loop
}
repeat
  m_k = ekf_slam(steps)
  m = join(m, m_k)
until end_of_map

return (m)
```

$$\begin{aligned} f(i) &= (i-1)s + m \\ &= (i-1)(m-r) + m \end{aligned} \tag{2.6}$$

If we decide to make a local maps of size p features, we will reach feature $\mathbf{f}_p$ at step i, where:

$$p = (i-1)(m-r) + m$$

and thus:

$$i = \frac{p-m}{m-r} + 1 \tag{2.7}$$

This means that in order to have p features in local map $\mathbf{m}_1$, it should be built using steps $1, \ldots, \frac{p-m}{m-r} + 1$, the second local map $\mathbf{m}_2$ should be built using measurements from steps $\frac{p-m}{m-r} + 2, \ldots, 2\frac{p-m}{m-r} + 2$, and in the same way, map $\mathbf{m}_k$ will be built using measurements at step $t(k)$, $\mathbf{z}_{t(k)}$, where:

$$t(k) = k\left(\frac{p-m}{m-r}\right) + k \tag{2.8}$$

Thus, to determine which feature we will reach in local map $\mathbf{m}_k$, we substitute i for $k\left(\frac{p-m}{m-r}\right) + k$ in eq. (2.6):

$$\begin{aligned} f(t(k)) &= \left(k\left(\frac{p-m}{m-r}\right) + k - 1\right)(m-r) + m \\ &= kp - (k-1)r \end{aligned} \tag{2.9}$$

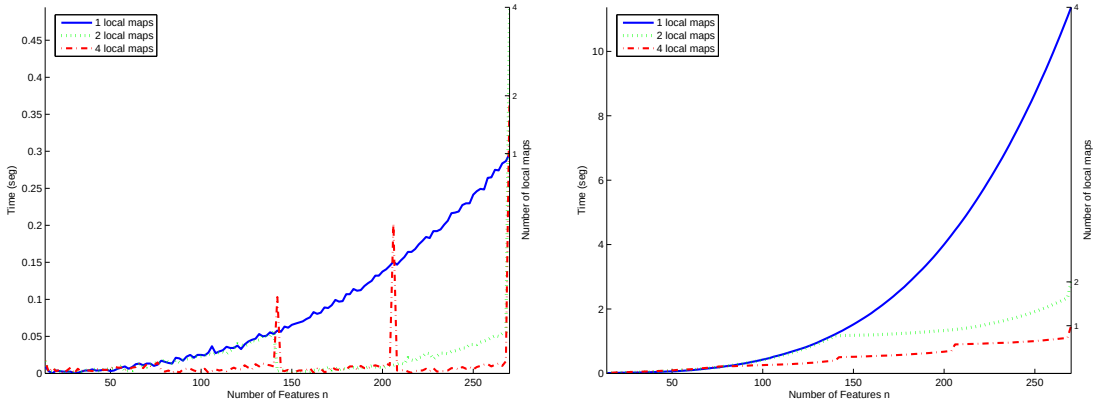


Figure 2.3: Map Joining SLAM execution for three different local map size. Cost per step (left). Accumulated cost (right).

If we want to reach the last environment feature $\mathbf{f}_n$ in the last step of map $\mathbf{m}_N$ and thus conclude mapping, according to eq. (2.9) we will have:

$$n = Np - (N-1)r$$

and thus the number of local maps N will be:

$$N = \frac{n-r}{p-r} \tag{2.10}$$

Consider as an example an experiment in an environment with $n = 270$ total point features, 135 on each side of the straight path that the vehicle traverses. Features are $2m$ apart. The range and bearing sensor has a range such that $m = 14$ features are perceived from a certain location. The robot moves $2m$ at each step. The vehicle thus leaves $s = 2$ features behind at each step and thus reobserves $r = 12$. If we decide to have local maps of size $p = 270$ then obviously $N = 1$; if we choose the smallest local map size of $p = m = 14$, then we should build $N = 128$ local maps.

We programmed the Local Map Sequencing algorithms of section 2.2 in Matlab and executed the mapping process for different sizes of local maps. Figure 2.3 shows the executions of algorithm 3 on our simulated experiment. We divide the total environment using three different local maps sizes $p = 270, 141, 77$ corresponding to $N = 1, 2, 4$ with $N = 1$ equal to standard EKF SLAM. Figure 2.3 top, shows the cost per step with peaks in those steps where Map joining process is performed. For local map size of $p = 77$ both the cost per step and the total cost (see fig. bottom) shows a significant reduction.

In order to find the local map size that produces the minimum cost, we extended the analysis from the smallest value of $p = 14$ requiring $N = 128$ local maps, to building one local map of size $p = 270$ covering the whole environment. Figure 2.4 shows the resulting

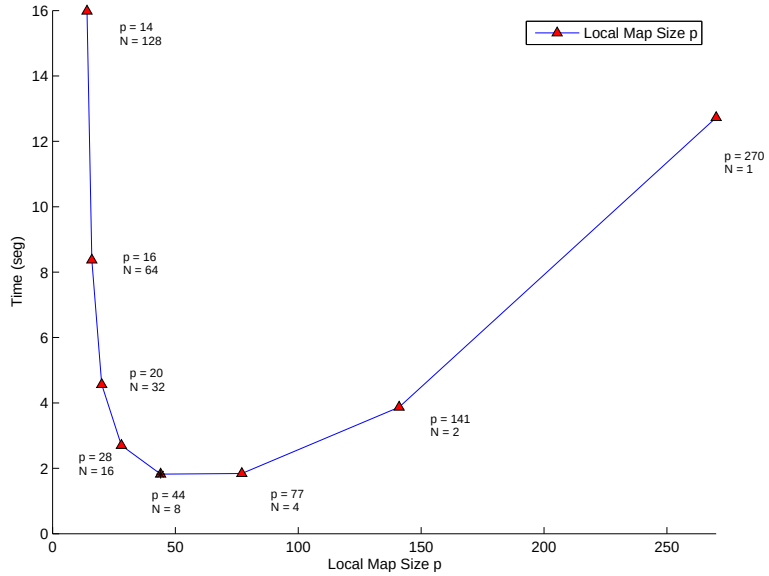


Figure 2.4: Execution time in seconds in Matlab of Local Maps Sequencing with Map Joining for local maps from $p = 14$ to $p = 270$. Results for N=1,2,4,8,16,32,64 and 128 local maps are shown (correspondingly, p=270,141,77,44,28,20,16,14).

execution times for the different values of p from 14 to 270. We can see that there is an *optimal* value for the local map size p (in this case close to $p = 44$) that minimizes the total cost of SLAM. Using this optimal value we can greatly reduce the cost of computation compared to standard EKF-SLAM (dividing by 7 in this small example). In the next section we will derive the expression for the computational cost of Local Map Sequencing that allows us to determine the optimal value of p for a given mapping problem.

2.4 Computational cost of Map Joining SLAM

2.4.1 Cost of building Local maps with Standard EKF-SLAM

Consider building a map $\mathbf{m} = (\hat{\mathbf{x}}, \mathbf{P})$ containing p features for the experiment proposed above. At the beginning of step k, when the vehicle moves from position $k-1$ to k, the state vector $\hat{\mathbf{x}}_{k-1}$ and its corresponding covariance matrix $\mathbf{P}_{k-1}$ contain $n_{k-1} = f(k-1)$ features. According to eq. (2.6):

$$n_{k-1} = (k-2)(m-r) + m$$

During the *prediction step*, the fundamental cost involves updating the correlation between the predicted vehicle location and the n_{k-1} map features. Hence, we consider the cost of the prediction step in number of operations as:

$$C_{Pred_k} = n_{k-1} \tag{2.11}$$

During the *estimation step* at k, the sensor obtains a set of measurements of m environment features. We summarize the main matrix operations and the cost necessary to update the estimate according to the number of reobserved features r:

$$\begin{aligned}
&\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k && rn_{k-1} \ ops \\
&\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T (\mathbf{S}_k)^{-1} && r^2 n_{k-1} \ ops \\
&\hat{\mathbf{x}}_k = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - \mathbf{h}_k(\hat{\mathbf{x}}_{k|k-1})) && rn_{k-1} \ ops \\
&\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} && rn_{k-1}^2 \ ops
\end{aligned}$$

Thus, the total cost of the update step is:

$$C_{Up_k} = rn_{k-1} + r^2 n_{k-1} + rn_{k-1} + rn_{k-1}^2$$

During the *addition step*, n new features are directly added to the current stochastic state vector $\hat{\mathbf{x}}_k$ by using the relative transformation between the robot R_k and the observed features, which requires n operations. Similarly, the covariance matrix $\mathbf{P}_k$ will require n^2 operations to add $\mathbf{R}_k$, and $n_{k-1}n$ to update correlations among the robot and the new features. The main cost of adding new features is summarized as follows:

$$C_{Ad_k} = n_{k-1}n + n^2$$

Carrying out EKF to build a map of p features requires that we process steps 1 to $\frac{p-m}{m-r}+1$. The total cost associated will be:

$$\begin{aligned}
C_{1map} &= C_{Pre_k} + C_{Up_k} + C_{Ad_k} \\
&= \sum_{k=1}^{\frac{p-m}{m-r}+1} (n_{k-1} + rn_{k-1} + r^2 n_{k-1} \\
&+ rn_{k-1} + rn_{k-1}^2 + sn_{k-1} + s^2)
\end{aligned}$$

Solving the arithmetic progression for all steps we obtain the total cost of building a local map of p features as (see the appendix for the values of k_1 to k_4):

$$C_{1map} = k_1 p^3 + k_2 p^2 + k_3 p + k_4 \tag{2.12}$$

2.4.2 Cost of joining all Local Maps

The standard EKF-SLAM technique presented above is used in Local Map Sequencing to divide the full map in N local maps $\mathbf{m}^1 \ldots \mathbf{m}^N$ containing p features each. In order to

map the environment up to the last feature $\mathbf{f}_P$, $N = \frac{P-r}{p-r}$ local maps should be built. The cost of building those N maps is given by (see appendix for the values of k_5 to k_8):

$$\begin{aligned} C_{Nmap} &= \left(\frac{n-r}{p-r}\right) C_{1map} \\ C_{Nmap} &= k_5 p^2 + k_6 p + k_7 + k_8 \frac{1}{(p-r)} \end{aligned} \tag{2.13}$$

The N local maps are joined into global map $\mathcal{M} = \left(\hat{\mathcal{X}}, \mathcal{P}\right)$ in $j = 1 \ldots N-1$ join and optimize steps, similar to those of EKF (please refer to Appendix B for the complete formulation). At global step j, we join global map $\mathcal{M}_j = \left(\hat{\mathcal{X}}_j, \mathcal{P}_j\right)$ with local map $\mathbf{m}^{j+1} = \left(\hat{\mathbf{x}}^{j+1}, \mathbf{P}^{j+1}\right)$. According to eq. (2.9), the size of global map $\mathcal{M}_j$ is:

$$\begin{aligned} n_j &= f(s(j)) \\ &= jp - (j-1)r \end{aligned}$$

The *joining step* consists in obtaining the state vector $\hat{\mathcal{X}}_{j+1/j}$ and covariance matrix $\mathcal{P}_{j+1/j}$. In the joined state vector $\hat{\mathcal{X}}_{j+1/j}$, all p features in local map $\mathbf{m}_j$ must be referenced to the global frame; this requires p operations. The covariance matrix $\mathcal{P}_{j+1/j}$ update requires computing the correlations between the $jp + (j-1)r$ features in the global map and the p features in the local map, for a cost of:

$$C_{MJ_j} = n_j\, p \tag{2.14}$$

We apply the specialized EKF update equations to obtain a new global map $\mathcal{M}_{j+1} = \left(\hat{\mathcal{X}}_{j+1}, \mathcal{P}_{j+1}\right)$ as follows:

$$\begin{aligned} &\mathcal{S}_{j+1} = \mathcal{H}_{j+1}\mathcal{P}_{j+1/j}\mathcal{H}_{j+1}^T && r(n_j+p)\ ops \\ &\mathcal{K}_{j+1} = \mathcal{P}_{j+1/j}\mathcal{H}_{j+1}^T(\mathcal{S}_{j+1})^{-1} && r^2(n_j+p)\ ops \\ &\hat{\mathcal{X}}_{j+1} = \hat{\mathcal{X}}_{j+1/j} - \mathcal{K}_{j+1}\mathbf{h}_{\mathbf{j+1}}(\hat{\mathcal{X}}_{\mathbf{j+1/j}}) && r(n_j+p)\ ops \\ &\mathcal{P}_{j+1} = (\mathbf{I} - \mathcal{K}_{\mathbf{j+1}}\mathbf{H}_{\mathcal{H}})\mathcal{P}_{\mathbf{j+1/j}} && r(n_j+p)^2\ ops \end{aligned}$$

Thus, the total cost of an update is:

$$\begin{aligned} C_{MJU_j} &= r(n_j+p) + r^2(n_j+p) + \\ &\quad r(n_j+p) + r(n_j+p)^2 \end{aligned}$$

Finally, the total cost of Local Map Sequencing is

$$\begin{aligned} C_{TOTAL} &= C_{Nmap} + \sum_{j=2}^{N} C_{MJU_j} \\ &= \frac{k_9 p^4}{(p-r)^2} + \frac{k_1 0 p^3}{(p-r)^2} + \\ &+ \frac{k_{11} p^2}{(p-r)^2} + \frac{k_{12}\, p}{(p-r)^2} + \\ &+ \frac{k_{13}}{(p-r)^2} + \frac{k_{14}\, p^3}{(p-r)} + \frac{k_{15}\, p^2}{(p-r)} + \\ &+ \frac{k_{16}\, p}{(p-r)} + \frac{(k_{17}+k_8)}{(p-r)} + (k_{18}+k_5)p^2 + \\ &+ (k_{19}+k_6)p + (k_{20}+k_7) \end{aligned} \tag{2.15}$$

See the appendix C for the values of k_9 to k_{20}.

2.4.3 Optimizing the Total Computational Cost

Using the values of our experiment, equation 2.15 becomes:

$$\begin{aligned} C_{TOTAL} &= \frac{-25p^4}{(p-r)^2} + \frac{3369p^3}{(p-r)^2} + \frac{865290p^2}{(p-r)^2} + \\ &+ \frac{13521924p}{(p-r)^2} + \frac{-182337480}{(p-r)^2} + \frac{-12p^3}{(p-r)} + \\ &+ \frac{948p^2}{(p-r)} + \frac{381816p}{(p-r)} + \frac{63996488}{(p-r)} + \\ &+ 528p^2 + 16142p - 247935 \end{aligned}$$

The optimal size of the local maps can be obtained by solving the following equation:

$$\frac{\partial C_{TOTAL}}{\partial p} = 0$$

$$982p^4 - 15638p^3 - 285498p^2 - 97445780p + 9.974e8 = 0$$

Fig. 2.5 shows the corresponding function C_{TOTAL} for the considered example. The analytical solution for this case can be easily obtained by means of the MATLAB `solve` *function*. According to `solve`, the predicted optimal size is attained when the local map size is $p = 52$ features. $N = 7$ to $N = 8$ local maps should be built and joined. It can be easily seen that C_{TOTAL} resembles very closely the actual execution times of the program

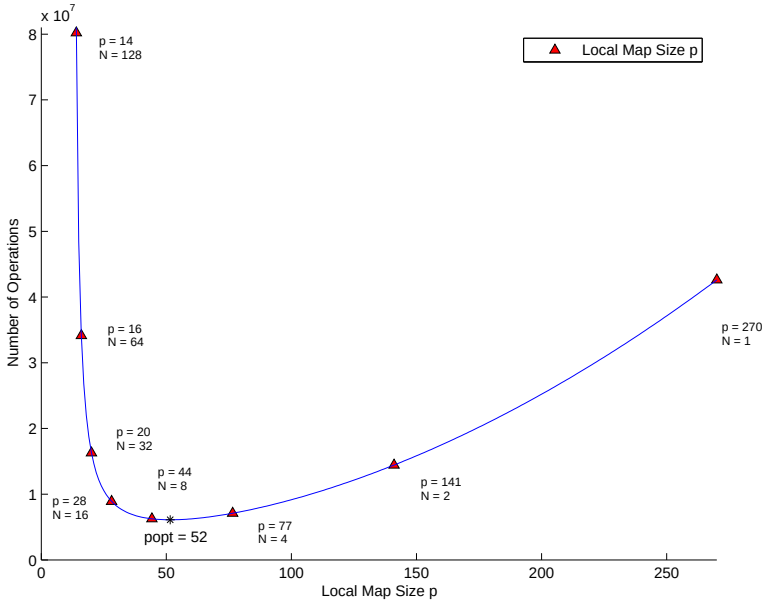


Figure 2.5: Computational cost versus local map size p in number of operations according to C_{TOTAL}; results for N=1,2,4,8,16,32,64 and 128 local maps are shown (correspondingly, p=270,141,77,44,28,20).

in fig. 2.4, and that the minimum $p = 44$ is very close to the predicted minimum of $p = 52$, coinciding in the optimal number of $N = 8$ local maps. Differences in both plots can be due to memory management and other implementation issues in Matlab.

It is also interesting to study the expected behaviour of the optimal computational cost as a function of sensor characteristics and total map size. For this purpose, we computed the optimal size p as a function of the sensor range m (fig. 2.6, left), of the number of new features per step n (center), and of the size of the environment P (right). The optimal local map size shows an apparent linear growth with m, and sublinear growth with n and P.

2.5 Evaluating Consistency

Consistency analysis have been carried out for both standard EKF SLAM and Map Joining SLAM. We evaluate consistency by using the index introduced in section 1.5 by eq. (1.13).

In order to study the effect of the local map size in the consistency of the resulting map estimate, we carry out Monte Carlo runs on the simulated experiment with the given standard deviation error of 5% of the distance in range, and $1deg$ in bearing for the sensor, and standard deviation error of $20cm$ frontal, $10cm$ lateral, and $2deg$ angular in the vehicle motions. The algorithm was run 20 times for each possible value of p, from 14 (equal to the sensor range m) to 270 (equal to the environment size n). Figure 2.7 shows the mean consistency index of the final map for the different local map sizes. Surprisingly,

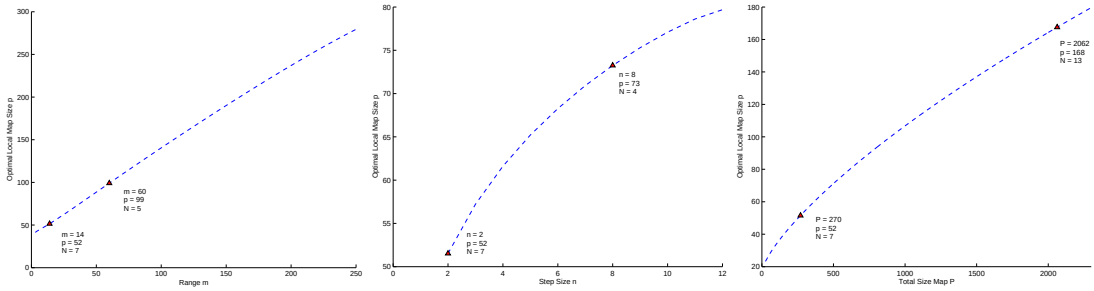


Figure 2.6: Optimal size of local maps p: apparent linear growth with the range sensor m (left); sublinear growth with the number of new features per step n (center); and also with the total environment size P (right).

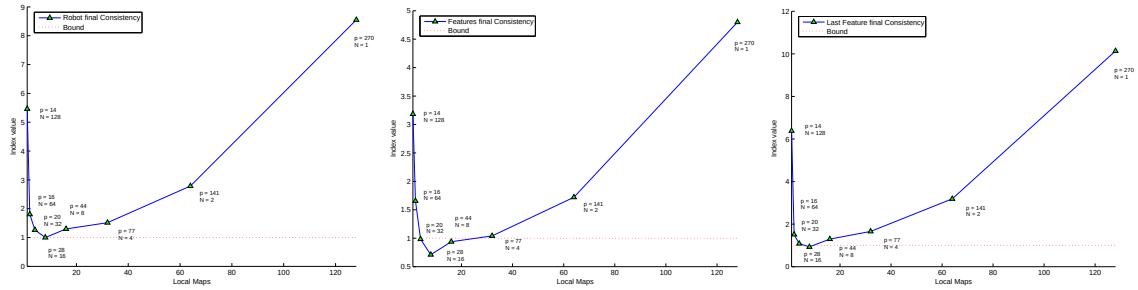


Figure 2.7: Figures on the right shows the final Mean Consistency Index with respect to local map size p for the Robot location (left), for all features in the map (middle) and also for the last features added (right).

the most consistent resulting map on average coincides with the local map size that is optimal from the computational point of view. The large and small linearizations have a least significant effect when the global and local map updates are optimally combined.

The analysis has been extended to the robot and the features separately. Figure 2.7 shows the individual consistency index for each. In contrast with the map consistency index, the index exceed the consistency bound in almost all cases, but a minimum value is reached by building around 8 to 16 local maps.

We also studied the resulting map quality in terms of vehicle and feature error. For effects of comparison, we focuses on the maximal local map size ($N = 1$) and on the optimal local map size ($N = 8$). Fig. 2.8, left, shows the evolution of the mean absolute error of the vehicle orientation and $x - y$ position for both the total EKF-SLAM algorithm and the Local Map Sequencing algorithm. The figure also shows the theoretical uncertainty (2σ bounds with no measurement or motion error) of both algorithms. For the mean angular error (top), it can be seen that in the case of the full EKF, the real uncertainty quickly falls both below the theoretical uncertainty and the angular vehicle error estimation grows very quickly. The real uncertainty of the Local Map Sequencing also falls below the theorical uncertainty, but at a much slower rate rendering better, and is always over the real angular error for this simulated experiment. Fig. 2.8, right, shows the evolution of the mean

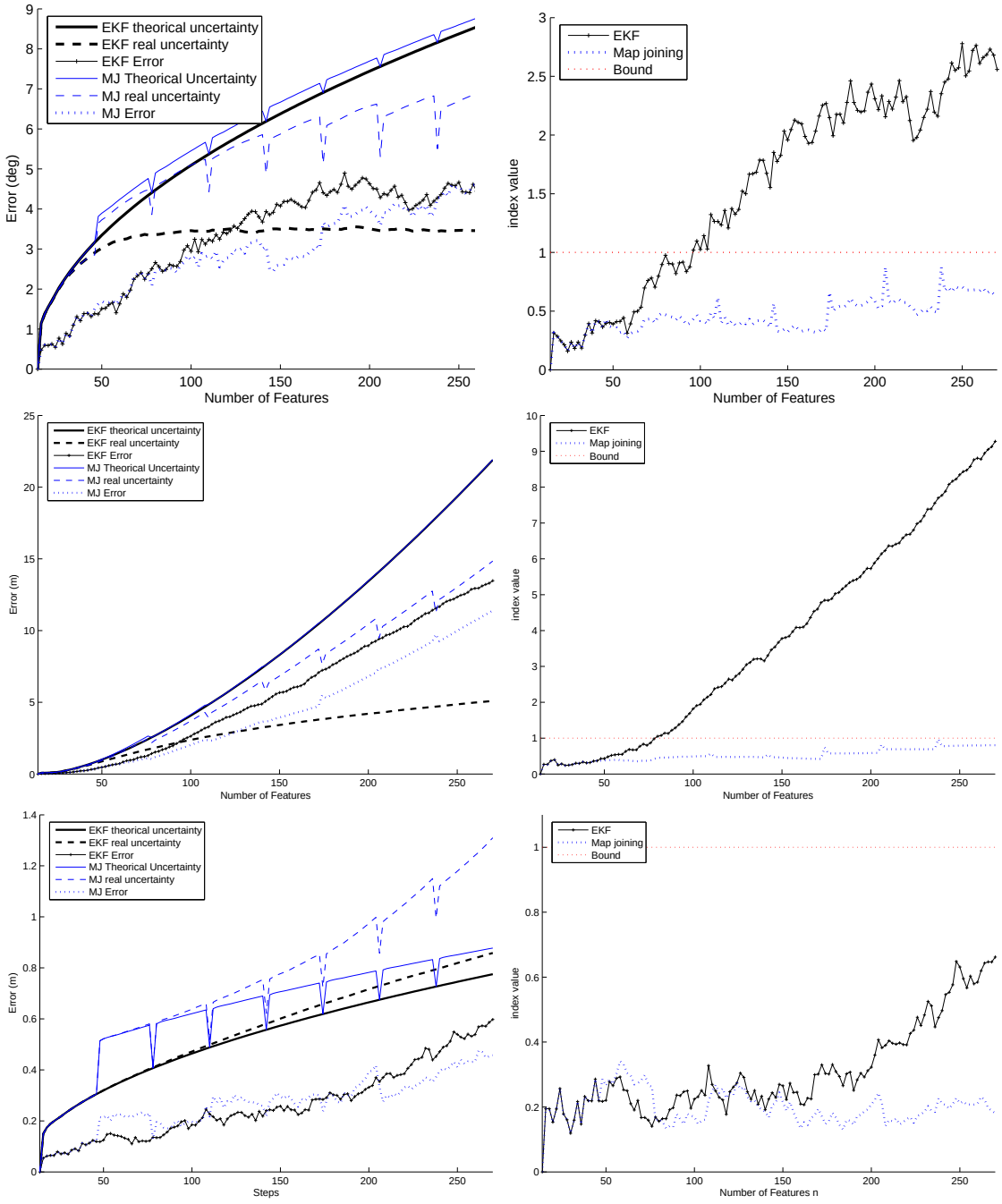


Figure 2.8: Vehicle orientation and position Mean Absolute Errors along the vehicle trajectory for the approaches discussed: a full EKF and a map joining of 8 local maps. Error and uncertainty bounds (right); Consistency index (left). Angular mean absolute error (top); x mean absolute error (middle); y mean absolute error (bottom).

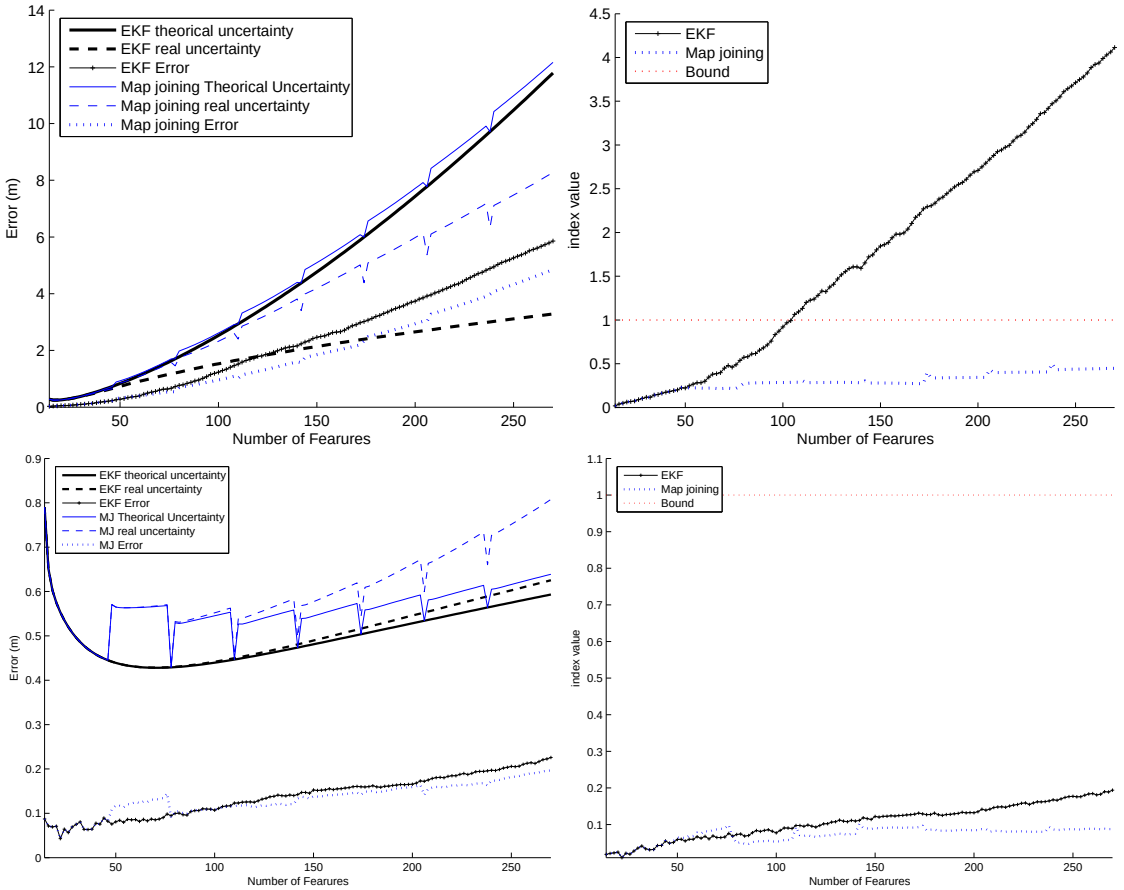


Figure 2.9: Comparison of position Mean Absolute Errors for all features in the map.

consistency index of eq. (1.13) of the angular and position error. We can see that in the case of the full EKF SLAM algorithm, the vehicle estimated orientation quickly becomes inconsistent, while in the case of the Local Map Sequencing Algorithm, the estimation of vehicle orientation remains consistent for the full length of the experiment. The same analysis is extended to the $x - y$ vehicle location where similar results are registered.

Similar results are also observed in the case of map feature error. Figures 2.9, and 2.10, left, show the comparative mean error in the x and y coordinates (orthogonal to the vehicle motion) of all map features and the last feature, for full EKF-SLAM and Local Map Sequencing. The figures also shows the mean 2σ bounds of the theoretical (with no measurement or motion error) and real uncertainty (with error) of both algorithms. It can be seen that the mean feature error of Local Map Sequencing is always smaller than that of full EKF-SLAM. The real uncertainty of both algorithms falls below the theoretical uncertainty, but at a much slower rate in the case of Local Map Sequencing. Fig. 2.9, right, shows the evolution of the mean value of the consistency index for all features as the map grows. Full EKF-SLAM quickly becomes inconsistent, while the estimation obatined

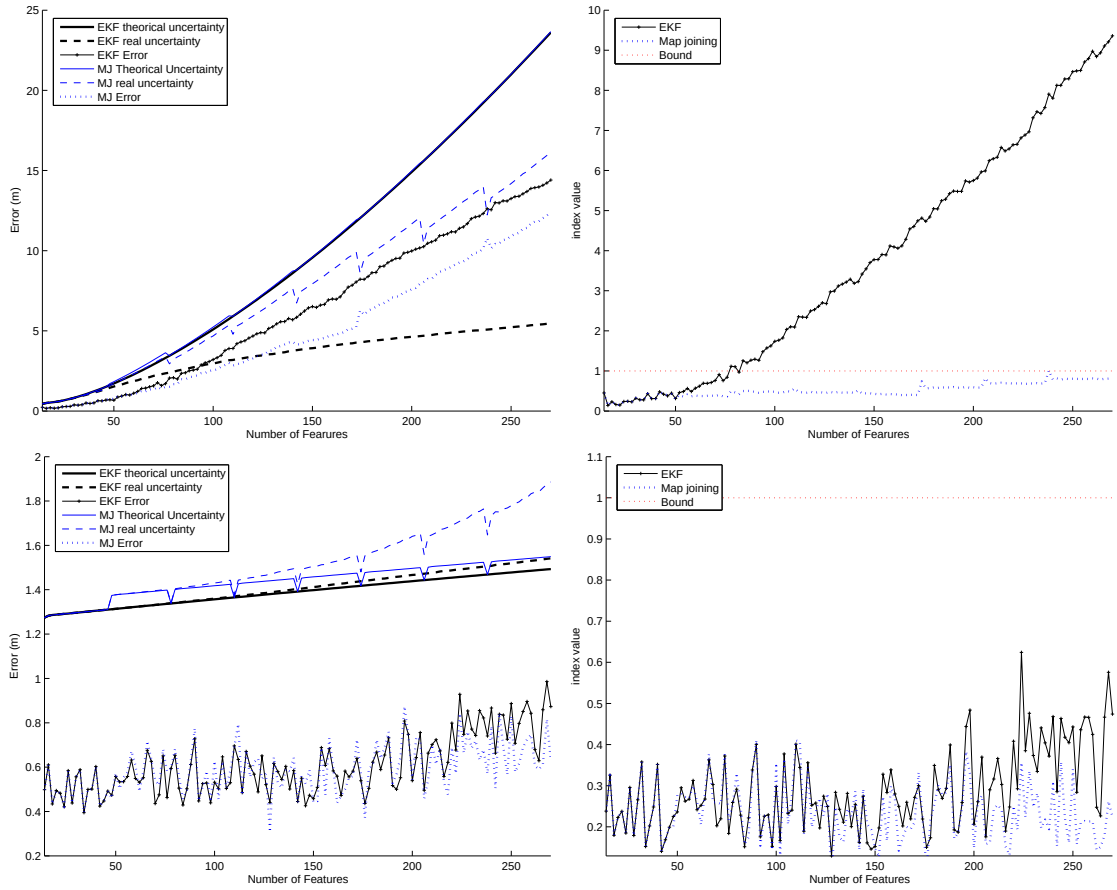


Figure 2.10: Comparison of position Mean Absolute Errors for the last feature in the map. y position (top). x position (bottom).

by Local Map Sequencing remains consistent for the whole map building process.

2.6 Discussion

The computational analysis for EKF-SLAM using local mapping and map joining allows us to determine the optimal size of local maps that minimize the total computational cost for a given sensor and vehicle characteristics. Unfortunately, the size of the environment to be mapped is seldom known a priori. This fact makes impossible to realize the cost analysis. To assure the autonomous performance of robots, we have to propose new strategies that avoid the calculation of the optimal local map size. This is the subject of the next chapter, in which we describe a new local supmapping algorithm based on Map Joining to overcome complexity and consistency problems.

Chapter 3

The Divide and Conquer SLAM algorithm

This chapter presents Divide and Conquer SLAM (D&C SLAM), a novel algorithm that exploits the advantages of the Map Joining method to overcome the two fundamental limitations of the standard EKF SLAM:

- *It reduces the computational cost per step from $O(n^2)$ to $O(n)$ (the total cost of SLAM is reduced from $O(n^3)$ to $O(n^2)$);*
- *It improves the consistency of the resulting estimates.*

The first property is a consequence of the Local Submapping strategies combined with a hierarchical structure that allows to join maps of the same size. The second implies that the state covariance obtained, represents more adequately the real error in the estimation. Unlike many current large scale EKF SLAM techniques, this algorithm computes an exact solution, without relying on approximations or simplifications to reduce computational complexity. Also, estimates and covariances are available when needed to solve data association without any further computation. Furthermore, as the method works most of the time in local maps, where angular errors remain small, the effect of linearization errors is limited. The resulting vehicle and map estimates are more precise than those obtained with standard EKF SLAM: the errors with respect to the true value are smaller, and the computed state covariance is consistent with the real error in the estimation. This chapter describes the algorithm and studies its computational cost and consistency properties.

3.1 Introduction. Related Work

Many potential methods have been described in the SLAM literature by which it is posible to regulate the growth of complexity imposed by the EKF covariance update step. Some of these methods have focused on to the effect of the map representation in the computational

complexity as well as on the estimation accuracy. Among the first is the work developed in [Csorba 98], where a Bounded Region Filter and a Relative Filter were presented. The latter algorithm, modifies the definition of the states to be estimated: instead of estimating the absolute state of single features, the relative filter uses relative locations between them in the state vector, such as relative distances and angles. However, it was shown that the Relative filter can result in inconsistent estimates [Newman 03].

In recent years, various mechanisms for partitioning the map state and covariance have shown promising results to manage the complexity problem. Early algorithms addressed the computational issue by splitting the global map into a number of globally referenced local maps, each with their own vehicle track. [Leonard 01] presented the Decoupled Stochastic Mapping, a computationally efficient approximation to address the transition between maps conservatively under vehicle relocation. Since this work, many other SLAM approaches based on local maps (with local reference) have been formulated: the Compressed Filter [Guivant 01], the Constrained Local Supmap [Williams 01] and Map Joining SLAM [Tardós 02] are the most representative solutions requiring periodically global map updates with cost $O(n^2)$. Other approaches relay on approximations to perform SLAM in constant time, paying a loss of precision price: Constant Time SLAM [Leonard 03], ATLAS framework [Bosse 04] and Hierarchical SLAM [Estrada 05] are classified into this set.

This chapter describes D&C SLAM, an algorithm that achieves linear time updates and recovers feature location estimates together with their covariances without approximations other than linearizations. Moreover, D&C SLAM inherits the consistency improvement property of Map Joining SLAM. The chapter is organized as follows: section 3.2 describes the structure of the D&C SLAM algorithm. In section 3.2.1 the total cost of performing D&C SLAM is derived. In section 3.2.2 it is detailed how D&C SLAM can carry out covariance updates in $O(n)$ per step. The simulations results of section 3.3 support the complexity and consistency properties of D&C SLAM under several scenarios, showing its applicability for large scale environments. In section 3.3.1 D&C SLAM is evaluated accordingly its consistency index and map precision. Section 3.4 discuses the limitations of D&C SLAM.

3.2 The Divide and Conquer algorithm

Instead of joining each new local map to a global map sequentially, as Map Joining SLAM does, the algorithm proposed in this chapter, Divide and Conquer SLAM, carries out map joining in a hierarchical fashion, as depicted in fig. 3.1. In order to compute a map of n features a sequence of l local maps of minimal size p are computed with standard EKF SLAM. The lower nodes of the hierarchy represent a sequence of l local maps of minimal size p. 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 size. D&C is implemented using algorithm 4, which uses a stack to save intermediate

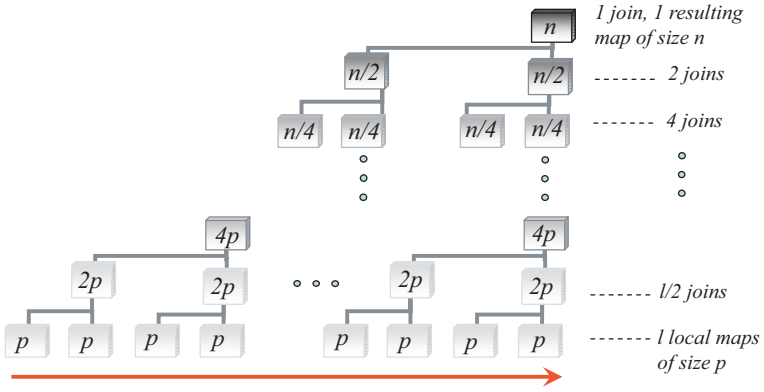


Figure 3.1: Hierarchy of maps that are created and joined in D&C SLAM. The lower level is the sequence of l local maps of size p, computed with standard EKF SLAM as the arrow suggests. The intermediate levels represent intermediate joins during the process. The top level represents the final map of size n resulting from the join of two local maps os size $n/2$.

maps. Whenever two maps of around the same size are at the top of the stack, they are replaced in the stack by their join. This allows a sequential execution of D&C SLAM. Fig. 3.2 illustrates the process for which an absolute map of $2D$ point features is created.

The tree strategy of D&C SLAM allows to obtain optimal (without approximations) and globally referenced estimates, when it is required, in contrast with the graph structures bound to [Bosse 04] and [Estrada 05], where sub-optimal updates are carried out ignoring cross-correlations.

3.2.1 Total computational complexity of D&C SLAM

In section 2.3 the influence of the local map size on the total cost of Map Joining SLAM was evaluated. In Map Joining SLAM, the total cost depends on the square of the growing size of the absolute map obtained. In contrast, D&C total cost also depends on the joining structure used, which produces local maps of different levels during the SLAM process.

During exploration in D&C SLAM, the process of building a map of size n requires the creation of $l = n/p$ maps of size p without considering overlap (common features), at cost $O(p^3)$ each (see eq. (2.12), chapter 2). These l maps form the first level of the structure and they are posteriorly joined to obtain $l/2$ local maps of size $2p$ corresponding to the second level, at cost $O((2p)^2)$ each. These in turn are joined into $l/4$ maps of size $4p$, at cost $O((4p)^2)$ 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 $O(n^2)$. Fig. 3.2 exemplifies this process. The total computational complexity of D&C SLAM is:

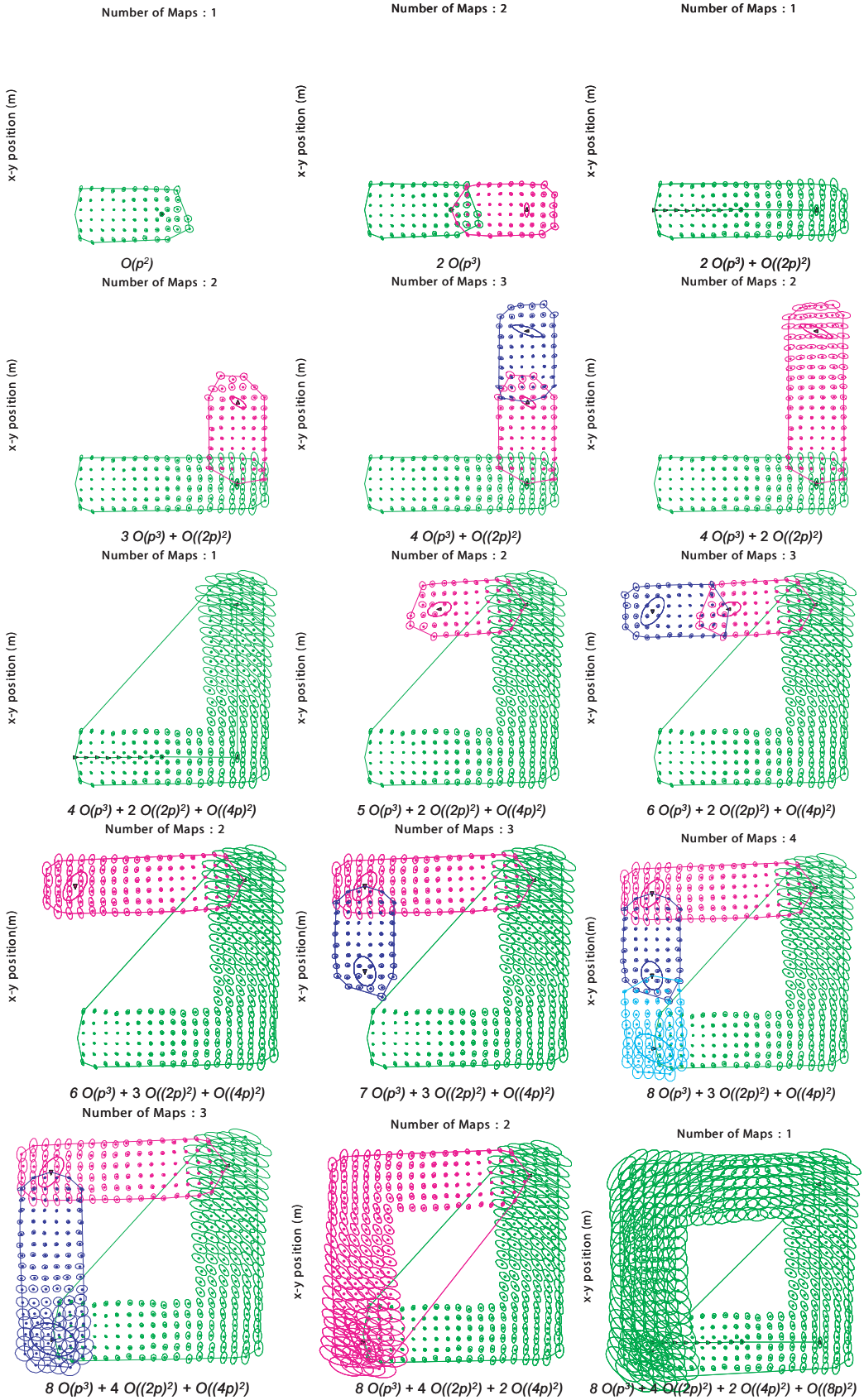


Figure 3.2: A sequence of l local maps are built and subsequently joined as the hierarchical structure suggests. The cost per step is described progressively: it considers the cost of building local maps of size p and adds the quadratic cost due to carrying out Map Joining when there are 2 local maps of the same size. In some steps of execution, specifically those that are a power of 2, a global map is obtained.

Algorithm 4 : `dc_slam`
`sequential implementation using a stack.`

```
stack = new()
m_0 = ekf_slam()
stack = push(m_0, stack) {
Main loop: postorder traversing of the map tree.
}
repeat
  m_k = ekf_slam()
  while ¬ empty(stack) and then
  size(m_k) ≥ size(top(stack)) do
    m = top(stack)
    stack = pop(stack)
    m_k = join(m, m_k)
  end while
  stack = push(m_k, stack)
until end_of_map {
Wrap up: join all maps in stack for full map recovery.
}
while ¬ empty(stack) do
  m = top(stack)
  stack = pop(stack)
  m_k = join(m, m_k)
end while
return (m_k)
```

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

Note that the sum represents all costs associated to map joining. This corresponds to the sum of a geometric progression of the type:

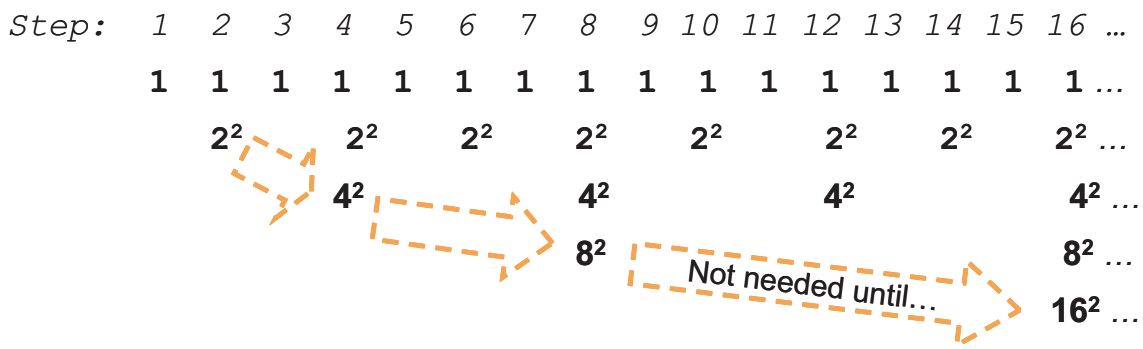


Figure 3.3: Instead of overloading a single step to join two maps (for example step 4), Map joining operations can be carried out during the next steps until a map of the same size is available (at step 8). In such a way, the cost per step is amortized. In this example, local maps of size $p = 1$ are considered.

$$\sum_{i=1}^{k} r^i = \frac{r - r^{k+1}}{1 - r}$$

Thus, in this case:

$$\begin{aligned} C_{DC} &= O\left(p^2 n + p\, n \frac{2^{\log_2 n/p+1} - 2}{2 - 1}\right) \\ &= O\left(p^2 n + p\, n(2\, n/p - 2)\right) \\ &= O\left(p^2 n + 2n^2 - 2pn\right) \\ &= O(n^2) \end{aligned} \tag{3.1}$$

This means that D&C SLAM performs SLAM with a total cost quadratic with the size of the environment, as compared with the cubic cost of standard EKF-SLAM. Map joining SLAM and D&C SLAM carry out the same number of map joining operations. The fundamental difference is that in D&C SLAM the size of the maps involved increases at a slower rate than in Map Joining SLAM. This allows the total cost to remain quadratic with n.

3.2.2 Computational complexity of D&C SLAM per step

In D&C SLAM, in steps that are a power of 2 (when $k = 2^t$) $i = 1 \cdots t$ joins will take place, at a cost $O(2^2), O(4^2) \cdots O(k^2)$ respectively. An analysis similar to that of eq. (3.1) shows that the total cost of such steps is $O(k^2)$, of the same order as a standard EKF SLAM step. However, in D&C SLAM the map to be generated at step k will not be required for joining until step $2\,k$ (see fig. 3.3). This allows us to amortize the cost $O(k^2)$ at this step by dividing it up between steps $k + 1$ to $2\,k$ in equal $O(k)$ computations for each step. In this way, amortized D&C SLAM becomes a *linear time* SLAM algorithm.

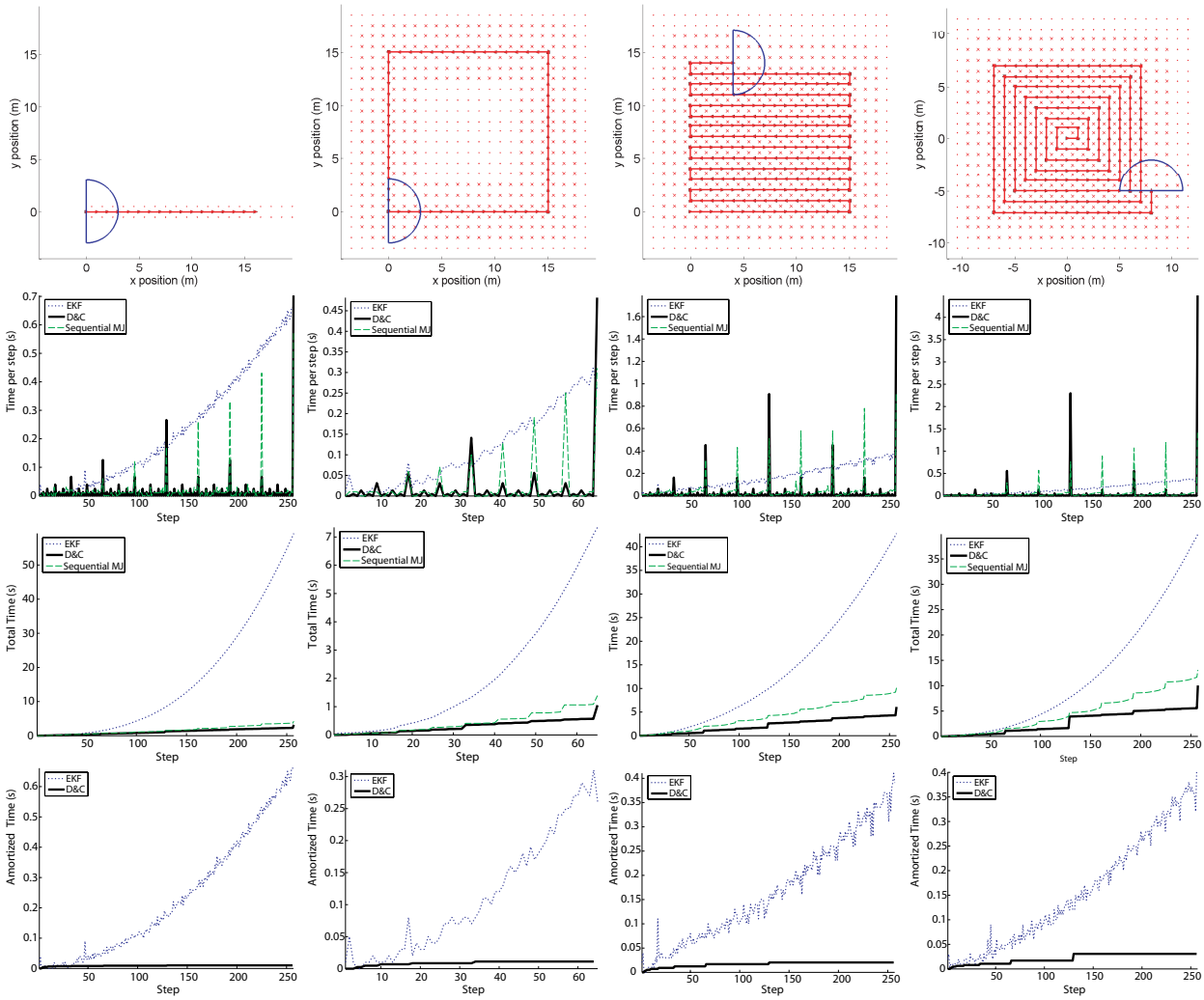


Figure 3.4: Four simulated experiments for comparing the EKF, Sequential Map Joining and D&C SLAM algorithms: (first column) straight forward trajectory; (second column) loop closing; (third column) lawn mowing; (fourth column) outward spiral. Ground truth environment and trajectory (top row); execution time per step of EKF vs. Map Joining vs. D&C SLAM (second row); total execution time (third row); execution time per step of EKF SLAM vs. amortized execution time per step of D&C SLAM (bottom row).

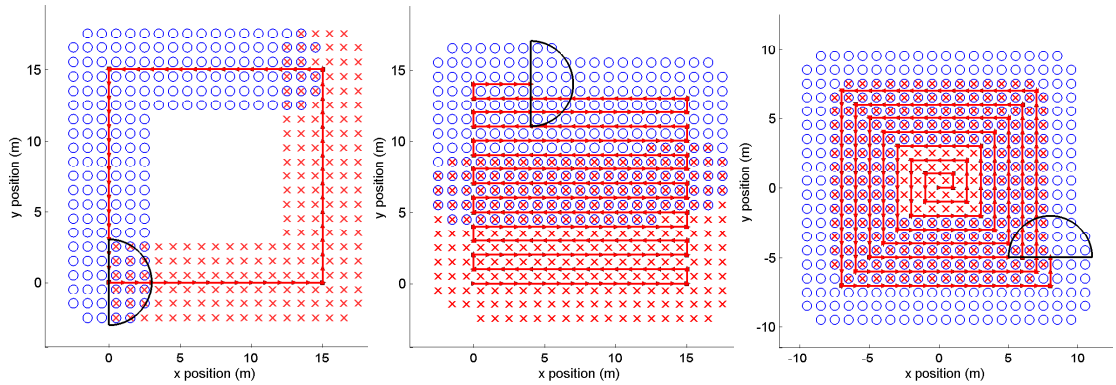


Figure 3.5: Size of the overlap between two final local maps, i.e. common features in both maps: crosses correspond to the first map and circles to the second map. Square loop trajectory (left), lawn mowers trajectory (middle), and outward spiral (rigth).

An amortized version of D&C SLAM can be implemented in a simple way using two threads: one high priority thread executes `ekf_slam` (alg. 1) to update the current local map, and the other low priority thread executes `dc_slam` (alg. 4). In this way, all otherwise idle time in the processor will be used for the more costly map joining operations, but high priority is given to local mapping, allowing for real time execution of D&C SLAM. As we will see in Monte Carlo simulations and experiments, the amortized cost of D&C SLAM is always lower than that of EKF SLAM. D&C SLAM is an anytime algorithm, if at any moment during the map building process the full map is required for another task, it can be computed in a single $O(n^2)$ step.

3.3 Simulated Experiments

We use four simulated scenarios (fig. 3.4) to illustrate the properties of the algorithm discussed in this chapter. In an environment which consists of equally spaced point features, a vehicle equipped with a range and bearing sensor carries out four different trajectories: straight forward, a square loop, lawn mowing and outward spiral (first, second, third and fourth column, respectively). The simulated experiments were carried out with known data association for the evaluated algorithms, in order to discard mismatching effects in the resulting performance of the estimators. The first row shows the environment and each of the trajectories. The second and third rows show the execution time per step and the total execution time, respectively. It can be seen that the total cost of D&C SLAM quickly separates from the total cost of EKF SLAM, and also from Map Joining SLAM. The reason is that the computational cost per step of D&C SLAM is lower than that of EKF SLAM most of the time. EKF SLAM works with a map of non-decreasing size, while D&C SLAM works on local maps of small size most of the time. Map Joining SLAM is computationally equivalent to D&C SLAM when working on local maps. In some steps (in the simulation those which are a multiple of 2), the computational cost of D&C is

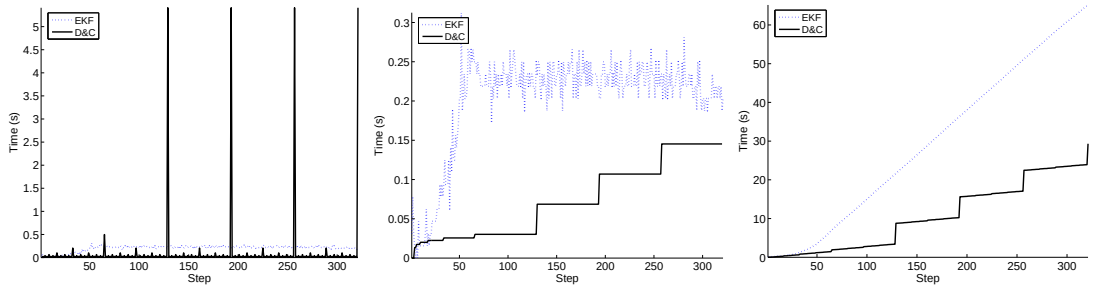


Figure 3.6: The cost per step (left) shows peaks around $5sec$ that corresponds to steps where the vehicle closes the loop. In those steps, Map Joining is performed between maps of size n each. In this case, the amortized version of cost per step (middle) becomes quadratic, the same that EKF but in a slower rate. Then the total cost (right) will be cubic.

higher. In those steps, one or more map joining operations take place (in those that are a power of 2, 2^t, t map joining operations take place). The actual running times of the algorithms where obtained from our `Matlab` implementation.

Fig. 3.4 (bottom row) shows the amortized cost per step for the four simulated experiments. We can see that the amortized cost of D&C SLAM is always lower than that of EKF SLAM.

We also evaluate the overlap dependency of the D&C SLAM cost. As it is described in algorithm `dc_slam`, D&C SLAM carries out a `join` step based on algorithm 2 described in chapter 2. The overlap represents the region of common features between local maps of the same size that allows to update the joint state vector. It basically depends on the environment and type of trajectory. Consider the simulated examples of fig. 3.5 where two $n/2$ maps are shown. In the first case, the square loop, the region of overlap between two maps will be of constant size, basically dependent on the sensor range. In the case of the lawn mowers trajectory, the overlap will be proportional to the length of the trajectory before the vehicle turns back, basically the square root of n. In the third case, the outward spiral, the region of overlap between the inner map and the encircling map is proportional to the square root of n as well. In some cases, like traversing a loop for a second time, the size of the overlap is the entire environment whose size is n. The latter corresponds to the worst case scenario where the most expensive operation is the inversion of the term $\mathbf{H}_{\mathcal{H}}\mathbf{P}^{-}_{i...k}\mathbf{H}^{T}_{\mathcal{H}}$, making the cost per step cubic and the amortized cost per step quadratic (see fig. 3.6).

3.3.1 Consistency and precision in Divide and Conquer SLAM

Apart from computational complexity, another important aspect of the solution computed by the EKF has gained attention recently: map consistency and precision.

We tested consistency of both standard EKF and D&C SLAM algorithms by carrying 100 Monte Carlo runs on the simulated experiments because simulations allow to have ground

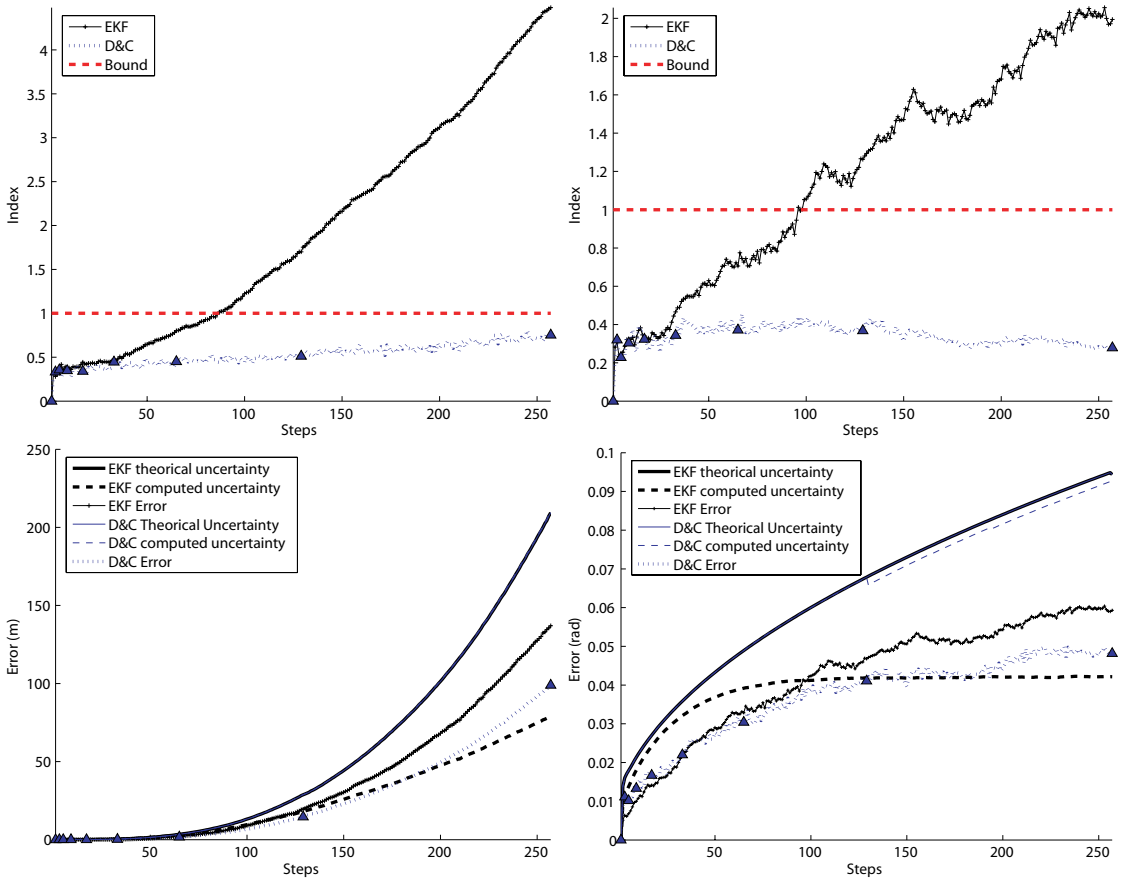


Figure 3.7: Mean consistency index CI of eq. 1.13 (top) and Root Mean Squared Error (bottom) for the robot x-y position (left) and orientation (right) for standard EKF and D&C SLAM. The root mean square error is always smaller for D&C SLAM; also the computation of the variance of the error is more precise, and thus the estimation remains consistent.

truth available. Additionally, Monte Carlo runs allow to obtain statistically significant evidence about the consistency of the algorithms being compared.

Figure 3.7 (top) shows the evolution of the mean consistency index defined in chapter 2 of the vehicle position (left) and orientation (right) during all steps of the straight forward trajectory simulation. We can see that the D&C estimate on vehicle location is always more consistent than the standard EKF estimate; in less than 100 steps EKF falls out of consistency while D&C remains consistent. Figure 3.7 (bottom) shows the evolution of the root mean square error on the vehicle position and orientation. The 2σ bounds for the *theoretical* uncertainty are computed by running the simulated experiment without measurement and robot noise so that linearizations take place in the true state values, and thus introduce no errors. The computed uncertainty of both standard EKF and D&C SLAM are also drawn. We can see how the RMS error increases more slowly in the case of D&C SLAM. We can also see the fast rate at which the uncertainty computed by standard

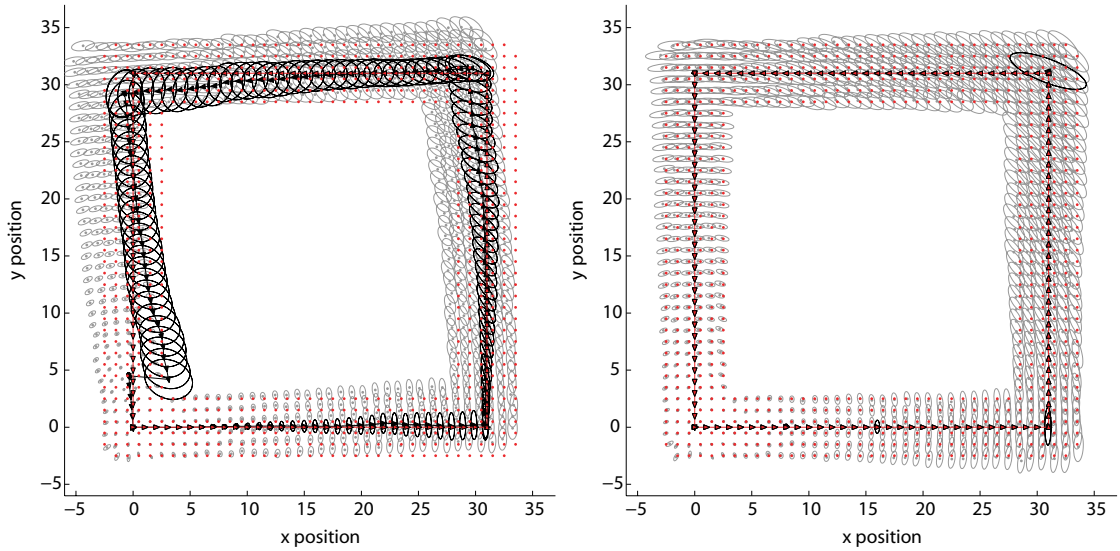


Figure 3.8: A typical result when running EKF SLAM (left) and D&C SLAM (right) on the same data. The vehicle and map features tend to accumulate more error during exploration with EKF SLAM. Even if data association is known, the final result after closing a loop is less precise. The absolute vehicle location estimates are shown when available from each algorithm. Ground truth environment and trajectory are shown in red.

EKF SLAM falls below its theoretical value.

Monte carlo runs show that *Divide and Conquer* SLAM is less subject to linearization errors than EKF SLAM. Fig. 3.8 shows a typical situation: the two algorithms run on exactly the same data of a loop closure. Because of less accumulated error and thus better linearizations, the final result is much more precise for D&C SLAM (data association is known and used by both algorithms).

3.4 Discussion

In this chapter D&C, a fast algorithm for SLAM, is proposed and examined. Its amortized computational cost is shown to be linear with map size. In several simulated scenarios D&C SLAM shows to be an adequate method to scale large map environments. Also, it is empirically shown that, when using the consistency index as measurement of convergence, D&C SLAM represents more exactly the covariance, closer to the theoretical value. Moreover, simple experiments with known data association gave suggest that the estimates provided are more precise when compare to EKF SLAM.

In last years, there have been few analytic results in the context of the consistency for EKF-based SLAM approaches. Recently, in [Huang 07] the authors give the insight for which it is posible to explain the good properties of D&C SLAM. They prove that convergence properties known to hold for linear EKF SLAM [Dissanayake 01] hold for nonlinear EKF

SLAM only when Jacobians (and thus linearizations) are computed around the ground truth solution, which is in general unfeasible. In the general case, the inconsistency is produced since the use of Jacobians evaluated at the current state estimate violates fundamental constraints inherent into the same Jacobians. In particular, when the robot orientation uncertainty is large, the extent of inconsistency is significant. Likewise, when the robot orientation uncertainty is small, the extent of inconsistency is insignificant. The authors point out that since the robot orientation error is the main source of inconsistency, algorithms that use local submapping, where the robot orientation error remains small, have the potential to improve consistency. Here we confirm the consistency improvement offered by one of such local submapping algorithms.

A notable problem arises when Map Joining process takes place: now correspondences between local maps of the same size have to be found. This is true all levels of the D&C structure, and corresponds to a more generic formulation about the data association problem regarding the overlap dependency of D&C SLAM. In next chapter, the problem of data association is addressed for continuos standard SLAM and D&C SLAM. The results achieved will show the feasibility of performing data association also in $O(n)$.

Chapter 4

Linear time data association for D&C SLAM

In D&C SLAM the problem of data association has to be analyzed, not just at the local map building level (e.g. the conventional framework adopted in EKF SLAM), but also at the map joining level: finding correspondences between local submaps. In D&C as local map size increases, the number of common elements between two local maps may not be constant, making the classical algorithms such as Individual Compatibility **IC** *and* Joint Compatibility **JC**, *unsuitable for real time execution. For consistent map joining, correct matching between estimated features and observations are essential since any single correspondence can corrupt the estimation process. One can use* **JCBB** *in the internal building process of each local map with the smallest size. However, its exponential behavior when traversing the association hypothesis, even in a* Branch and Bound *framework, might yield enormous costs for the D&C SLAM process. To ensure a linear cost of the data association at any level of the SLAM algorithm, a tesellation-based algorithm is introduced in this chapter to test* Individual compatibility *efficiently, and a* Randomized *version of the joint compatibility* **RJC** *is proposed.*

4.1 Introduction

The problem of Simultaneous Localization and Mapping must provide a full estimated solution for both the robot and the map fearures locations. With this propose, odometry and landmarks observations should be incorporated at each step. Additionally, the SLAM solution must be obtained in real time including data association. In the recent race of proposing strategies to manage SLAM complexity, we can find many attempts that address the problem from the graph SLAM point of view. Unfortunately, although both batch and incremental approaches have been studied in this sense [Frese 05; Dellaert 06; Ni 07; Olson 07; Olson 06; Grisetti 07b; Grisetti 07a; Grisetti 08], they do not consider

data association. Indeed, these works are the proof that data association is still a critical issue for a real time SLAM implementation. Some of their authors directly agree in that batch approaches are suitable to deal better with linearizations and improve the estimated relations in the map, but incremental ones have more possibilities for being implemented for real time requirements. Some claim that to solve data association covariances are not a mandatory factor since appearance based approaches have achieved success in the visual SLAM literature. Quickly, the authors realized that reliable data associations have to be solve incrementally based on the map posterior density function [Kaess 07; Grisetti 07a]. Once more, the recovery of covariances becomes useful for researchers in order to guarantee that match reliability [Ni 07; Clemente 07].

Probabilistic approaches as EKF SLAM allow to perform data association by using statistical validation tests. Under this assumption many algorithms have been designed. In principle, an *Individual Compatibility* test *(IC)*, which makes use of the Mahalanobis distance to contrast map features with a limited set of partial observations, can deal with the problem of finding associations between map features and partial observations. However, in cases where the robot uncertainty is large, or features density is high *IC* fails, producing multiple unreliable matches. Hence, the use of *IC* is limited and can only be applied as a candidates pre-selector for the final algorithm solution. Data association has evolved to more sophisticated approaches allowing to evaluate the geometric consistency between features *jointly* [Neira 01].

In the previous chapter, the advantages of D&C SLAM were described considering a simulated environment. Several situations were considered comprising complicated trajectories as spirals or those preformed by a lawn mower. Parallel to the extensive analysis of computational complexity, the results obtained from the experiments demonstrate the high performance of the linear D&C strategy when it is compared with a standard EKF implementation. We also mentioned the issue of overlap size in D&C with respect to the map size.

This chapter is devoted to solving data association in a way that the proposed method is able of finding the overlap maintaining the cost per step linear. Tessellation techniques to perform Individual Compatibility in linear time and a *randomized* version of the *Jointly Compatible* approach, *RJC*, are the core of the solution proposed. In section 4.2 the standard framework to solve data association is summarized. This is the framework adopted to build local maps. The new method for data association between D&C maps is introduced in section 4.3. In order to show the feasibility of implementing the full D&C method, experiments on Victoria park data set were performed 4.4.

4.2 Data association for standard EKF SLAM

The data association problem in continuous EKF SLAM consists in producing a hypothesis $\mathcal{H} = [j_1 j_2 \cdots j_i \cdots j_m]$ where correspondences are established between each of $i = 1 \ldots m$ sensor measurements and one (or none) of $j = 1 \ldots n$ map features. The space

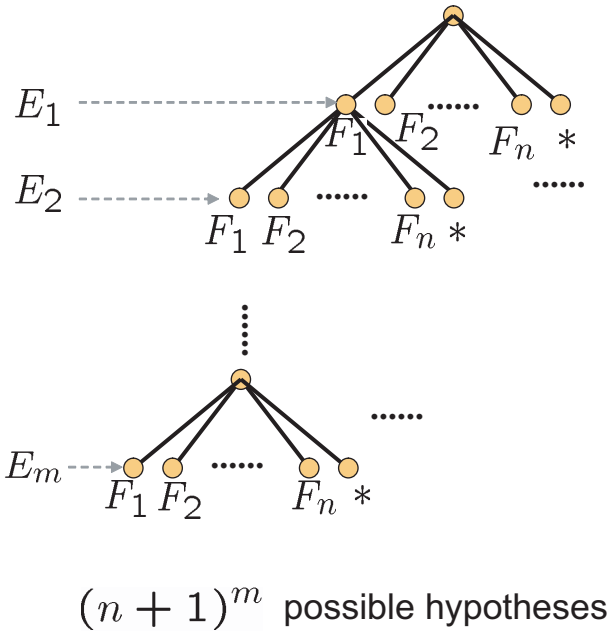


Figure 4.1: The *interpretation tree.*

of measurement-feature correspondences can be represented by an *interpretation tree* of m levels as introduced by [Grimson 90a] (see fig. 4.1). Each node of the tree at level i has $n+1$ branches, corresponding to the n alternative feature pairings for measurement i, and an extra node (star-branch) to account for the possibility of the measurement being spurious or a new feature. The size of this correspondence space, (i.e. the number of alternative hypotheses) in which data association must be solved is exponential with the number of measurements: $(n+1)^m$.

Fortunately, the availability of a stochastic model for both the map and the measurements allows us to check each measurement-feature correspondence for *individual compatibility* by predicting the location of the map features relative to the sensor reference, and determining compatibility using a hypothesis test on the innovation and covariance of each possible pairing. The procedure is illustrated in fig. 4.2, left. The discrepancy between the observation i and the predicted observation of map feature j is measured using the innovation term of eq. (1.5) and covariance (1.2). The measurement can be considered corresponding to the feature if the Mahalanobis distance $D^2_{k,ij}$ satisfies [Bar-Shalom 88]:

$$D^2_{k,ij} = \nu^T_{k,ij} \mathbf{S}^{-1}_{k,ij} \nu_{k,ij} < \chi^2_{d,1-\alpha} \tag{4.1}$$

where $d = \dim(\mathbf{h}_{k,j})$ and $1-\alpha$ is the desired confidence level, usually 95%. $\nu_{k,ij}$ represents the difference between the predicted map feature $\mathbf{h}_j$ and partial observation $\mathbf{m}_i$ at step k with corresponding covariance $\mathbf{S}_{k,ij}$.
In standard EKF SLAM, and for a sensor of limited range and bearing, the number of

Algorithm 5 : $\texttt{IC}(\mathbf{Z}_{1\cdots m}, \mathbf{h}_{1\cdots n})$

$IC = []$
for $i = 1$ to m **do**
  for $j = 1$ to n **do**
    $\mathbf{S}_{\mathcal{H}_k ij} = (\mathcal{H}_{kj}\mathbf{P}_{kj}\mathcal{H}_{kj}) + \mathbf{R}_{ki}$
    $D^2 = (\mathbf{z}_{ki} - \mathcal{H}_{kj})^T \mathbf{S}^{-1}_{\mathcal{H}_k ij} (\mathbf{z}_{ki} - \mathcal{H}_{kj})$
    $IC_{ij} = D^2 \leq \chi^2_{2,1-\alpha}$
  end for
end for

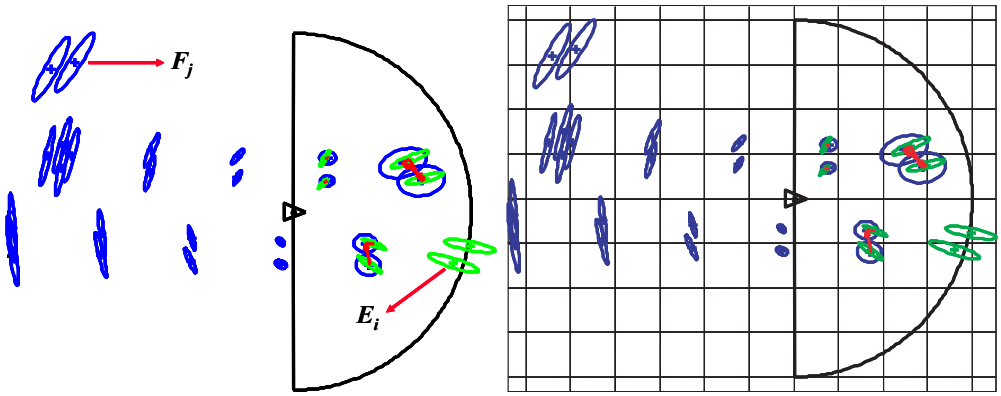


Figure 4.2: Computing the Individual Compatibility. The blue map is predicted at current time k where green observations are measured (left). This framework requires the evaluation of $O(mn)$ pairings to find c total candidates. Tesellation can reduce the cost to $O(1)$ by checking matched cells (right).

measurements m is constant and thus individual compatibility is $O(nm) = O(n)$, linear on the size of the map. Algorithm 5 shows a naive implementation to perform IC validation. This cost can be easily reduced to $O(m)$, a constant, by a simple tessellation or grid of the map computed during map building, which allows us to determine candidates for a measurement in constant time simply by checking the grid element and nearby grid elements in which its predicted location falls. In 2D problems, the cost of computing and updating the tessellation would be constant per step (a constant amount of new features are included in the map per step), while the space required would be proportional to the total area covered by the map and the resolution of the tessellation (see fig. 4.2, right).

In cases where clutter or vehicle error are high, there may be more than one possible correspondence for each measurement. More elaborate algorithms are required to disambiguate in these cases. Nevertheless, the overlap between the measurements and the map is the size of the sensor range plus the vehicle uncertainty, and thus more or less constant. In local mapping, after individual compatibility is sorted out, we use the **Joint Compatibility Branch and Bound (JCBB)** algorithm by [Neira 01] (see alg. 6) to disambiguate

Algorithm 6 Joint Compatibility Branch and Bound

```
Continuous_JCBB (E_{1...m}, F_{1...n}):
Best = []
JCBB ([], 1)
return Best

JCBB (H, i): {find pairings for observation E_i}
if i > m then {leaf node?}
   if pairings(H) > pairings(Best) then
      Best ← H
   end if
else
   for j = 1 to n do
      if individual_compatibility(i, j) and then
      joint_compatibility(H, i, j) then
         JCBB([H j], i + 1) {pairing (E_i, F_j) accepted}
      end if
   end for
   if pairings(H) + m − i > pairings(Best) then {can do better?}
      JCBB([H 0], i + 1) {star node, E_i not paired}
   end if
end if
```

between the possible associations for the m measurements. It has been shown that algorithms such as JCBB, based on a probabilistic model (feature estimates and covariances), can greatly increase the robustness of data association, even when other measurement properties are available. For instance, in the case of monocular SLAM, the combined use of texture-based matching and stochastic compatibility tests has proved critical to reject outliers, especially from repetitive patterns and dynamic objects [Clemente 07]. **JCBB** performs branch and bound search on the interpretation tree looking for *jointly compatible* correspondences, but only in the overlap determined by individual compatibility. Given that this is a region of the map of constant size, each measurement will have a more or less constant number of feature candidates, say c, and thus the solution space is constant: $(c+1)^m$. In this way, **JCBB** will execute in constant time.

4.3 Data association for Divide and Conquer SLAM

Data association in D&C SLAM is a very particular problem because it involves finding correspondences between two local maps of similar size whenever joining is to take place. For instance, before obtaining a final map of size n, the data association problem has to be solved between two maps of size $n/2$ and so computing *individual compatibility* would be $O(n^2)$ instead of $O(n)$. Fortunately, as in the case of individual compatibility for standard EKF SLAM, finding potential matches for one feature in another map can be done in constant time using a simple tessellation or grid in the map where the search is

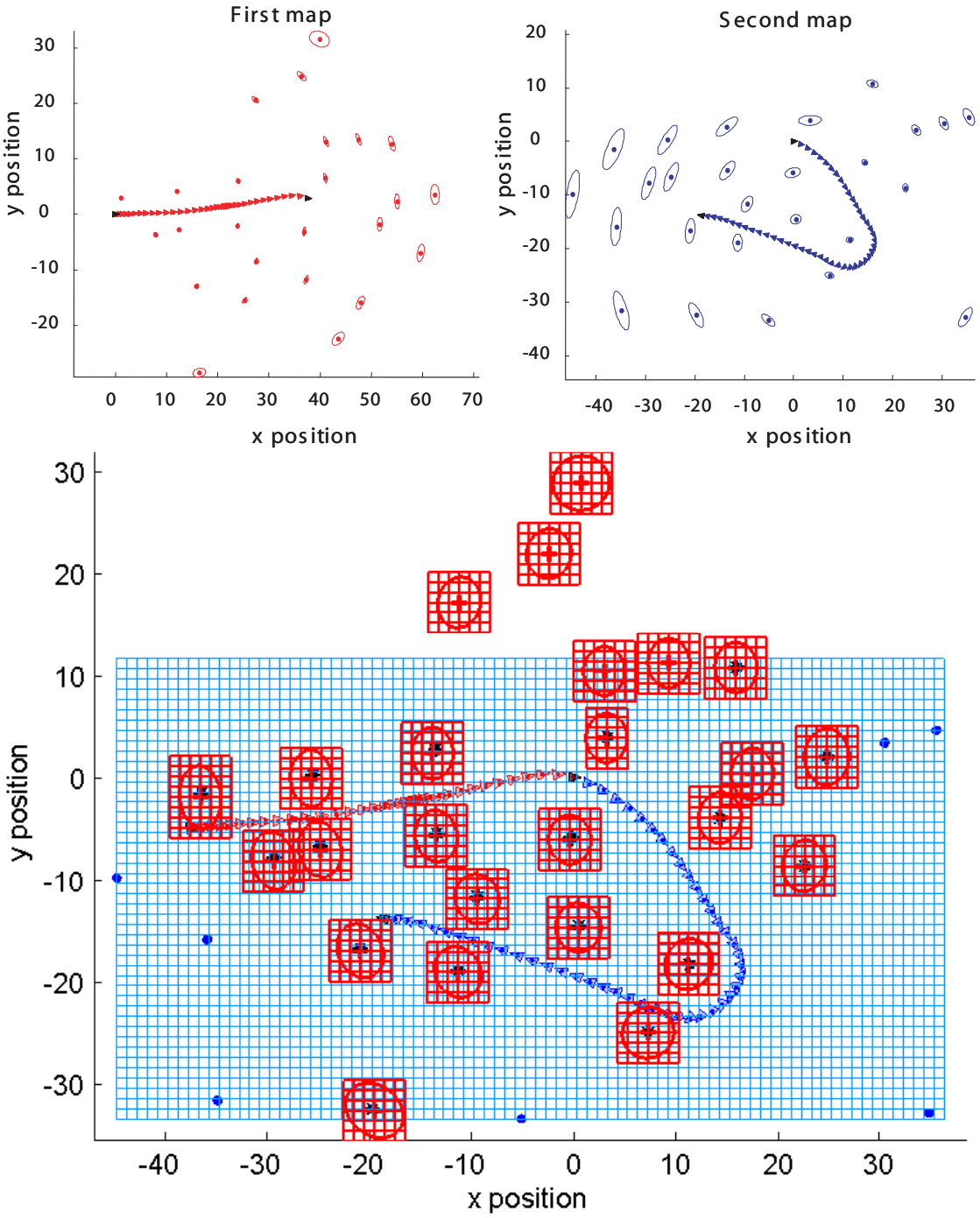


Figure 4.3: Tessellation to compute individual compatibility between two local maps of similar size (left). The second (blue) map is tessellated using a grid. Red ellipses represent the uncertainties of the predicted features of the first local map with respect to the base reference of the second (right). The ellipses are approximated by windows, and in this way possible candidates (asterisks) for each red feature can be found in constant time. The robot trajectory is also shown for each local map with the corresponding color.

Algorithm 7 :RJC

$\mathbf{P}_{fail} = 0.01$, $\mathbf{P}_{good} = 0.8$, $b = 4$
$i = 1$, $Best = []$, $\mathcal{H} = []$
while $(i \leq t)$ **do**
  $\mathbf{m}_2^* = \text{random\_select}(\mathbf{m}_2, b)$
  $\mathcal{H} = \text{JCBB*}(\mathcal{H}, 1, \mathbf{m}_1, \mathbf{m}_2^*)$
  if pairings($\mathcal{H}$) > pairings($Best$) **then**
    $Best = \mathcal{H}$
  end if
  $\mathbf{P}_{good} = max(\mathbf{P}_{good}, pairings(Best)/m)$
  $t = \log \mathbf{P}_{fail} / \log(1 - \mathbf{P}_{good}^b)$
  $i = i + 1$
end while

{
JCBB: testing the joint compatibility for b pairings.*
}
`procedure JCBB*(`$\mathcal{H}$`,i,`$\mathbf{m}_1$`,`$\mathbf{m}_2^*$`)`
if pairings($\mathcal{H}$)==b **then**
  $\mathcal{H} = \text{NN}(\mathcal{H}, i + 1, \mathbf{m}_1, \mathbf{m}_2^*)$
else
  for $j = 1$ to $length(\mathbf{m}_1)$ **do**
    if `individually_compatible(i,j)` $\wedge$ `jointly_compatible(`$[\mathcal{H}\ j]$`)` **then**
      `JCBB*(`$[\mathcal{H}\ j]$`,`$i + 1$`,`$\mathbf{m}_1$`,`$\mathbf{m}_2^*$`)`
    end if
  end for
end if

done. Consider the example in fig. 4.3, top. The red trajectory and features correspond to the first local map built, and the blue trajectory and features correspond to the second local map. Individual compatibility may be done in a way similar to standard EKF SLAM: we predict the location of features in the first (red) map relative to the base reference of the second (blue) map, and check for possible correspondences with blue features. If the blue map is tessellated (fig. 4.3, bottom) we can find potential matches for a red feature in constant time, and for the whole red map in linear time. The cost of computing and updating the tessellation is constant per step, while the space required is proportional to the total area covered by each map. Alternative solutions, such as 2D priority kd-trees [Uhlmann 01], can be used to make the storage space required be $O(n \log n)$ on the number of map features, instead of dependent on the covered area as the tessellation is. There is however a higher cost involved in finding a potential match per feature, $O(\log n)$, and an update cost of $O(\log n)$ per feature to be included.

A second issue of importance is the size of the region of overlap between two local maps. While in standard EKF SLAM this region is constant, and thus data association algorithms like **JCBB** will execute in constant time, in D&C SLAM the size of the overlap is not always constant.

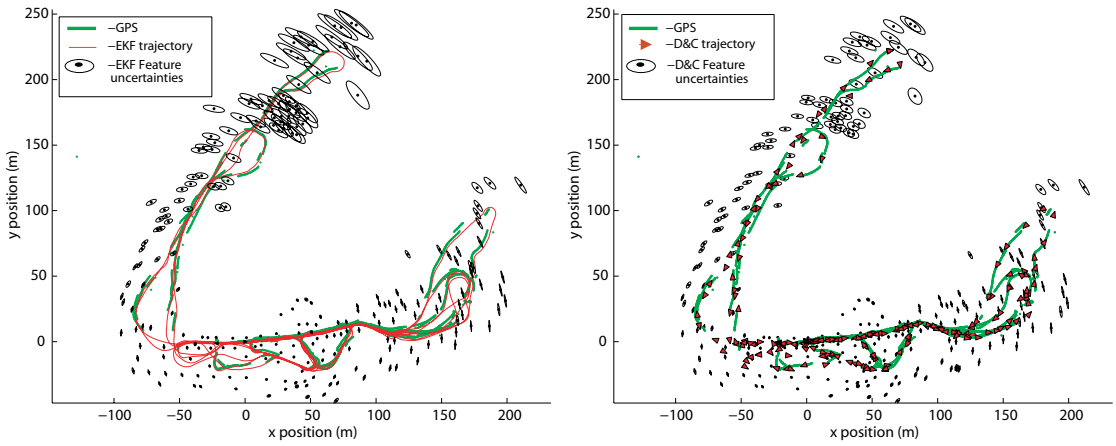


Figure 4.4: Map for Victoria Park dataset according to the EKF SLAM (left) and to D&C SLAM (right). The results are essentially equivalent; there are some minor differences due to missed associations in the case of EKF. The estimated position along the whole trajectory is shown as a red line for EKF SLAM, and the vehicle locations are drawn as red triangles when available in D&C SLAM. Green points are GPS readings in both cases, and are not used in either case.

In order to limit the computational cost of data association between local maps in D&C SLAM, we use a *randomized joint compatibility* algorithm. Our **RJC** approach (see algorithm 7) is a variant of the linear **RS** algorithm described in [Neira 03] used for global localization. Consider two consecutive maps $\mathbf{m}_1$ and $\mathbf{m}_2$, of size n_1 and n_2 respectively, to be joined. First, the overlap between the two maps is identified using individual compatibility. Second, instead of performing branch and bound interpretation tree search in the whole overlap as in **JCBB**, we randomly select b features in the overlapped area of the second map and use **JCBB***. This algorithm is a version of **JCBB** where all b features are expected to be found in the second map (no star branch). This produces a hypothesis $\mathcal{H}$ of b jointly compatible features in the first map. Associations for the remaining features in the overlap are obtained using the simple nearest neighbor rule given hypothesis $\mathcal{H}$, that is, finding pairings that are compatible with the first b features. In the spirit of adaptive **RANSAC** [Hartley 00], we repeat this process t times, so that the probability of missing a correct association is limited to P_{fail}.

The RJC algorithm successfully detects the overlap between two local maps in either continuous data association or loop closing. The only requirement is that the stochastic maps remain consistent, a condition which is enforced by the D&C algorithm.

Since **JCBB*** is executed using a fixed number of features, its cost remains constant. Finding the nearest neighbor for each remaining feature among the ones that are individually compatible with it, a constant number, will be constant. The cost of each try is thus $O(n)$. The number of tries depends on b, the number of features randomly selected, on the probability that a selected feature in the overlap can be actually found in the first map P_{good}, and on the acceptable probability of failure in this probabilistic algorithm, P_{fail}. It does not depend on the size of either map. In this way, we can maintain data association

in D&C SLAM linear with the size of the joined map.

4.4 Experiments

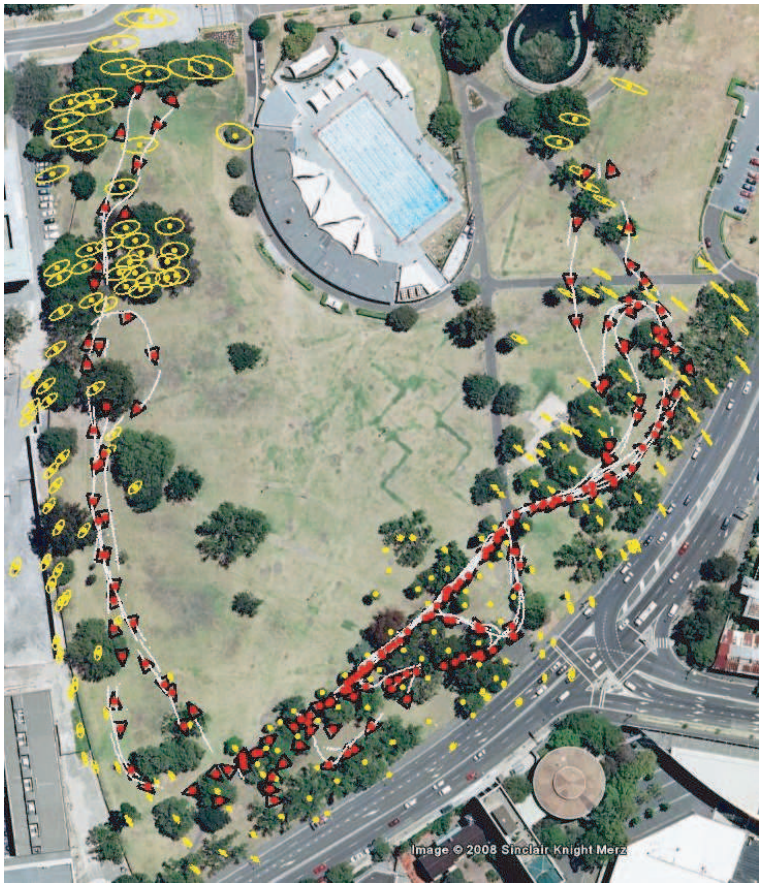


Figure 4.5: The results were projected on Google Earth in order to compare the precision obtained.

We have used the well known Victoria Park data set to validate the algorithms D&C SLAM and **RJC**. This experiment is particulary adequate for testing SLAM due to its large scale and the significant level of spurious measurements. The experiment also provides critical loops in absence of reliable features.

For **RJC**, we chose $b = 4$ as the number of map features to be randomly selected as seed for hypothesis generation. Two features are sufficient in theory to fix the relative location between the maps, but we have found 4 to adequately disambiguate. The probability that a selected feature in the overlap is not spurious, $\mathbf{P}_{good}$ is set to 0.8, and the probability of not finding a good solution when one exists, $\mathbf{P}_{fail}$ is set to 0.01. These parameters make the data association algorithm carry out 9 random tries.

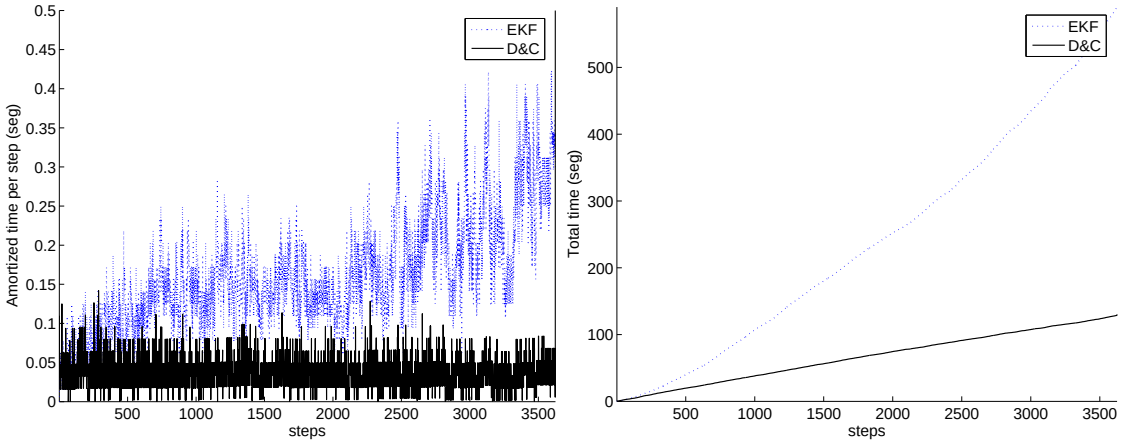


Figure 4.6: Time per step of EKF SLAM vs. amortized time per step of D&C SLAM (left); accumulated time of EKF SLAM vs. D&C SLAM (right).

Figure 4.4 shows the resulting maps from standard EKF SLAM vs. D&C SLAM. Each algorithm solves data association on its own. This allows seeing when the estimator falls out of consistency precisely because data association starts to fail; there are some minor differences due to missed associations in the case of EKF. Figure 4.6, left, shows the amortized cost of D&C SLAM. We can see that in this experiment an EKF step can take 0.5 seconds, while the amortized D&C SLAM step will take around 0.05 seconds. The main source of the noise visible in the EKF timing values is the variable number of observations gathered when the vehicle traverses the environment.

In real experiments like Victoria Park it is not generally possible to predict at which step the size of the current local map will reach its limit size. This depends on the trajectory that the vehicle follows and on the density of features in the environment. Some features can be initialized in the map and latter removed if they are not re-observed. In Victoria Park, the vehicle sometimes carries out exploratory trajectories and sometimes it revisits previously mapped regions of the park. When two local maps are joined, the size of the resulting map will depend on the overlap. For these reasons, the total map size does not increase linearly with the number of steps. This makes the total cost of standard EKF SLAM to not be cubic with the number of steps (fig. 4.6, right). For the same reasons the total cost of D&C SLAM seems to grow linearly, instead of quadratically, with the step number. In any case, the benefits of using D&C SLAM can be clearly seen. In this experiment, the total cost of D&C SLAM is one fifth of the total cost of standard EKF, (130.24s on a 2.8 GHz Pentium IV, compared to 590.48s for EKF SLAM). This result is also comparatively better than the reported by iSAM algorithm [Kaess 07] (464s on a 2 GHz Pentium M) and comparable to FastSLAM 2.0 [Montemerlo 03] (140s on a 1 GHz Pentium IV). Plotting the results of D&C SLAM on ©Google Earth (fig. 4.5) reveals a superior performance when comparing with the precision obtained by the mentioned algorithms [Kaess 07; Montemerlo 03].

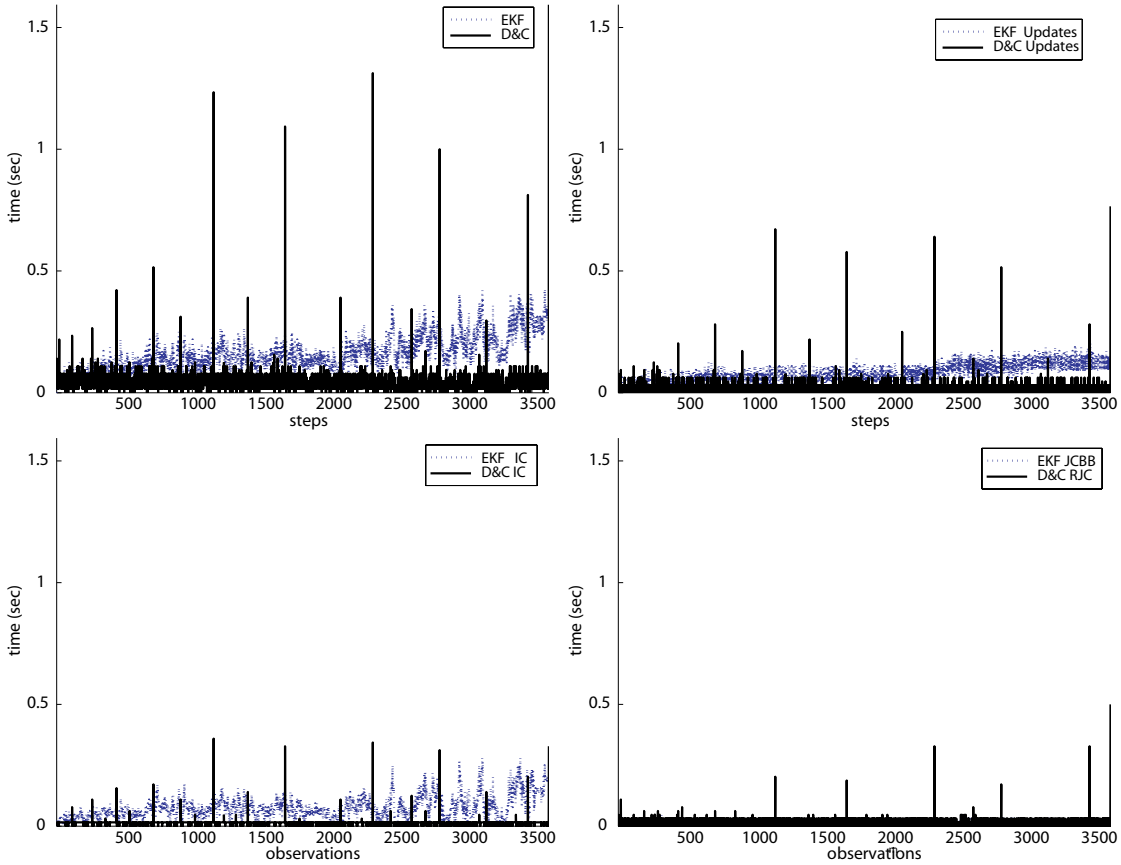


Figure 4.7: Analysis of the Victoria Park cost per step (top-left). D&C updates (top-right), Tessellation for Individual (bottom-left) Compatibility and Randomized Joint Compatibility (bottom-right) were counted in the cost per step.

The cost per step has been evaluated so that the contributions of the data association can be appreciated separately. Fig. 4.7 shows in detail the comparative running times of both standard EKF SLAM and D&C SLAM. In all cases, the D&C dominant peaks make reference to the Map Joining cost when the process is not amortized. Higher peaks are mainly due to the D&C update step (top-right). Figures (bottom,left-right) also demonstrate the dependency of the Individual Compatibility and Randomized Joint Compatibility costs on the overlap size.

4.5 Discussion

We have discussed new data association strategies that can be applied in many SLAM applications. Nonetheless, some techniques as tessellation are mainly suitable for range and bearing sensors. In cases where the scale is not observable as in monocular SLAM applications (i.e. depth information is not straightly available from observations), the

search for individual compatible pairings may involve dependency on more parameters and would imply tessellation on unknown dimensions which definitely could increase the cost.

In this sense, we believe that data association based on geometry jointly with vision can complement each other to provide reliable solutions. Vision sensors provide rich information as shape, color and texture, which can be used to find a correspondence between two data sets. SLAM can benefit from appearance signatures useful to predict a possible association in tasks such as closing a loop. Also, visual information is valuable for assisting conventional compatible hypothesis with additional discrimination information [Clemente 07]. In computer vision, appearance signatures and image similarity metrics have been developed for indexing image databases [Rubner 98] and for recognizing places in topological mapping [Ulrich 00]. More recently, appearance measures have been applied to detecting loops in SLAM [Gutmann 99], [Newman 06], [Clemente 07]. The work on visual appearance methods for loop detection by Newman et al. [Newman 06] introduces two notable innovations. First, a similarity metric is computed over a sequence of images, rather than a single image and, second, an eigenvalue technique is employed to remove common-mode similarity. This approach considerably reduces the occurrence of false positives by considering only matches that are interesting or uncommon. Other hybrid approaches based on learning map features have shown to be strong to outline the problem of relocating a portable 6DOF system after failures [Williams 07a].

In this thesis, the idea of tessellating the space seems also feasible to address the global localization problem in SLAM: the determination of the vehicle location in a previously mapped environment with no other prior information. As preliminary work, in the next chapter it will be shown that, using a grid sampling representation of the configuration space, it is possible to evaluate all vehicle location hypotheses in the environment (up to a certain resolution) with a computational cost that is bilinear: linear both in the number of map features and in the number of sensor measurements.

Chapter 5

Linear time relocation in SLAM

This chapter describes a method to solve the global localization problem in SLAM: in absence of prior information about its location, a vehicle must relocate using an environment map which has been previously built. This situation is common when the vehicle is kidnapped, in presence of large position uncertainties or after system failures. This last one can occur, for example, in visual SLAM applications after tracking failure due to occlusions, motion blur or unmodelled rapid motions. We show that, using a grid sampling representation of the configuration space, it is possible to evaluate all alternative vehicle location hypotheses in the environment (up to a certain resolution) with a computational cost that is bilinear: linear both in the number of map features and in the number of sensor measurements. It is proposed a pairing-driven *algorithm that considers measurement-feature pairings individually and thus, in contrast with current correspondence space algorithms, it avoids searching in the exponential correspondence space. It uses a voting strategy that accumulates evidence for each vehicle location hypothesis, assuring robustness to noise in the sensor measurements and environment models. The general nature of the proposed strategy allows the consideration of different types of features and sensor measurements. Using the popular Victoria Park dataset, we compare the performance of this algorithm with* location-driven *algorithms where the solution space is usually randomly sampled. The results prove that the proposed pairing-driven technique is computationally more efficient than alternative* location-driven *algorithms in proportion to the density of features in the environment.*

5.1 Introduction

A problem of considerable attention in SLAM is the determination of the location of a vehicle in an environment, given a set of m local measurements and a map of the environment with n features, but no prior information about the vehicle location. A

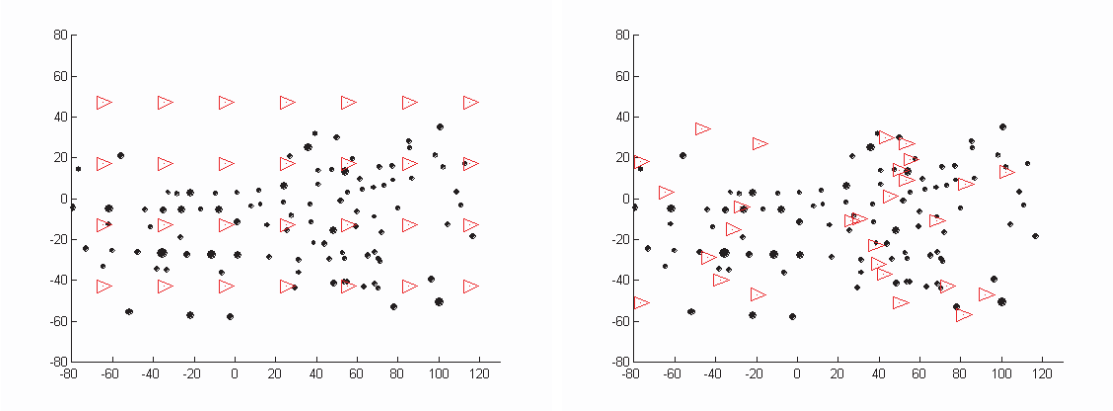


Figure 5.1: Location driven techniques

solution to this *global localization* problem allows us to restart the SLAM algorithm when the vehicle is lost or when it arrives to a previously mapped area with no odometry or high odometry error. In environments or situations in which no GPS fix is possible, this constitutes the only alternative.

Most current global localization algorithms are based on some form of consensus, and therefore are *robust* to spurious sensor measurements, or the detection of features not present in the map. In current applications of interest, because large environments are being considered, the issue of global localization has shifted towards *efficiency*. Since sensors are typically local, and therefore the number of measurements will normally be bound, attention is drawn to how well these algorithms scale with the size of the map.

Global localization algorithms can be classified as *location-driven* or *pairing-driven*. Location-driven strategies explore the *configuration space*, where a set of s different vehicle localization hypotheses, or location samples, are considered. Figure 5.1 shows this idea. Each is ranked according to how well the local measurements match the environment map at the hypothesized location. Algorithms that use random sampling of the configuration space, such as Monte Carlo localization [Fox 99], and those that use grid sampling, such as Markov Localization [Fox 98; Burgard 96], belong to this category. If the environment map is represented as an occupancy grid, the computational complexity of these algorithms, called here `Loc_driven`, is $O(s \cdot m)$. At each of the s locations each of the m measurements is compared with the map (usually a grid) to find correspondences.

In contrast to location-driven techniques, pairing-driven techniques explore the *correspondence space*, where data association hypotheses are produced considering consistent combinations of measurement-feature pairings. For the most promising hypotheses (in terms of the number of pairings), the vehicle location can be computed and the hypothesis can be verified. Algorithms that belong to this category traverse the exponential Interpretation Tree [Grimson 90a] using techniques such as: hypothesize and test [Lim 00], branch and bound [Neira 01], maximum clique [Bailey 00], or random sampling [Neira 03]. In this

last work it is shown that introducing the concept of *locality*, consider only those features covisible with the involved sensor from the hypothesized any pairing-driven algorithm can be made linear with the size of the map, but there remains in the Interpretation Tree search a component exponential in the number of measurements.

In this chapter a *pairing-driven* algorithm, `Pair_driven` is proposed, that uses a voting strategy, in the spirit of the Generalized Hough Transform [Ballard 81]. In contrast to current correspondence space algorithms, it evaluates all location samples considering only individual measurement-feature pairings. This avoids the exponential computational cost of interpretation tree traversal, resulting in an algorithm that is $O(n \cdot m)$, linear with both the number of environment features and of sensor measurements. We show that this pairing-driven algorithm is computationally faster than location-driven algorithms in proportion to the density of features in the environment. The main reason is that our algorithm exploits the lattice structure offered by a grid sampled configuration space; another problem in which a lattice structure offers advantages is motion planning [Lindemann 04]. Markov Localization [Fox 98; Burgard 96] uses grid sampling, but it does not take advantage of the lattice structure that the grid offers.

This chapter is organized as follows: section 5.2 describes both location-driven algorithms that work in the configuration space, and pairing-driven algorithms that traverse the correspondence space. In section 5.3 the alternatives in the representation of both the environment and the configuration space, and their effect in the computational complexity of these algorithms are analyzed. This section also contains a probabilistic analysis of the robustness of these voting algorithms. Section 5.4 contains comparative experimental results of running both algorithms in a large outdoor environment, using the popular Victoria Park dataset obtained by Guivant and Nebot [Guivant 02].

5.2 Algorithms for global localization

Assume that the environment map consists of n features distributed in a 2D environment.

$$\mathbf{F} = \{\mathbf{f}_1, \ldots, \mathbf{f}_n\}$$

Without loss of generality, we will consider 2D point features, so that

$$\mathbf{f}_j = (x_j,\ y_j)^t$$

Assume a sensor mounted on the vehicle obtains observations of m features gathered from the vehicle location to be determined.

$$\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_m\}$$

The information available from each measurement depends on the type of sensor. In the case we are considering, sensors can give range-only (sonar), bearing-only (a camera), or range and bearing (laser) measurements of 2D points.

The configuration space is the space of possible vehicle location hypotheses of the form

$$\mathbf{x} = (x,\, y,\, \phi)^t$$

where

$$x \in [x_{min}, x_{max}], y \in [y_{min}, y_{max}] \; and \; \phi \in [\phi_{min}, \phi_{max}]$$

Given that the configuration space is continuous although bound, the number of alternative vehicle location hypotheses is infinite. A limited number of hypotheses, or location samples, within the bounds, must then be considered. Monte Carlo methods [Fox 99] work by drawing random samples in this space, while Markov Localization [Fox 98; Burgard 96] draw grid samples. Both random sampling and grid sampling methods can be tuned to adequately represent the solution space for a given resolution.

Assume that a set $\mathbf{X}$ of s samples are obtained from the configuration space. We can either uniformly divide the configuration space in n_x, n_y and n_ϕ grid elements in x, y and ϕ, respectively, or equivalently, we can randomly sample the configuration space in

$$s = n_x \cdot n_y \cdot n_\phi$$

location samples. The appropriate resolution depends on the distribution of features in the environment and on the precision of the local sensor.

Global localization algorithms work by evaluating the compatibility between sensor measurements and environment features considering their individual properties, such as length, radius, color, etc. (unary constraints), but mainly analyze their relative geometry and the compatibility between the measurements predicted from the map and the actual measurements. Both Monte Carlo methods and Markov Localization methods compute the likelihood of each location sample given an occupancy grid map and the measurements.

In the case of feature-based maps, the correspondence space techniques that are usually applied try to maximize the number of observations that successfully match a feature in the map. The implicit assumption is that the likelihood of a location hypothesis increases with the number of matchings. In section 5.3.4 we perform a probabilistic analysis that validates this assumption.

Thus we consider global localization algorithms that evaluate each location sample by computing its corresponding number of observation-feature matches. There are two basic alternatives with respect to how to proceed in the evaluation of each sample in the set $\mathbf{X}$ of alternative vehicle locations: *location-driven* strategies, and *pairing-driven* strategies.

- In a *location-driven* strategy, each vehicle location hypothesis $\mathbf{x}$ is considered in turn. Each of the m sensor measurements will agree with the location hypothesis if its hypothesized absolute location matches one or more of the n environment features. Algorithm 8, `Loc_driven`, implements this alternative. $\mathbf{F}_i$ is the set of hypothesized feature locations that would produce the measurement. In the case of range and bearing, it will be one 2D point, in the case of range, it will be a set of 2D points in

Algorithm 8 `Loc_driven`:
consider each alternative location hypothesis in turn

```
votes = 0
for each hypothesis x ∈ X do
  for each measurement z_i ∈ Z do
    F_i = predict_features(x, z_i)
    if any_compatible_feature(F_i, F) then
      votes(x) = votes(x) + 1
    end if
  end for
end for
```

Algorithm 9 `Pair_driven`:
consider each measurement-feature matching in turn

```
votes = 0
for each measurement z_i ∈ Z do
  for each feature f_j ∈ F do
    X_ij = hypothesize_locations(f_j, z_i)
    X_v = compute_compatible_locations(X_ij, X)
    votes(X_v) = votes(X_v) + 1
  end for
end for
```

a circle around the vehicle location sample at the measured range distance, etc. The validity of a measurement-feature association can be determined using error bounds; statistical validations using χ^2 tests can also be carried out. In this algorithm, each location hypothesis is ranked according to the number of measurements for which a feature association can be established. This strategy amounts to considering each candidate location and counting the number of votes cast by measurements that agree with that location.

- In a *pairing-driven* strategy, each measurement $\mathbf{z}_i$ is considered in turn. For every feature $\mathbf{f}_j$ in map $\mathbf{F}$, we compute the vehicle location(s) from where the feature could have produced the measurement. We then find the locations in $\mathbf{X}$ compatible with these hypothesized locations, and update their ranking (see algorithm 9, `Pair_driven`). This is a voting strategy in the spirit of the Hough transform [Ballard 81]: each measurement $\mathbf{z}_i$ will determine the set of vehicle locations $\mathbf{X}_{ij}$ from which it is feasible to detect the environment feature $\mathbf{f}_j$ in order to produce the given measurement, and vote for location samples $\mathbf{X}_v$ sufficiently close to those feasible locations (see fig. 5.2). It amounts to consulting each voter (the measurements) in turn and adding a vote to each of the candidates (the vehicle location

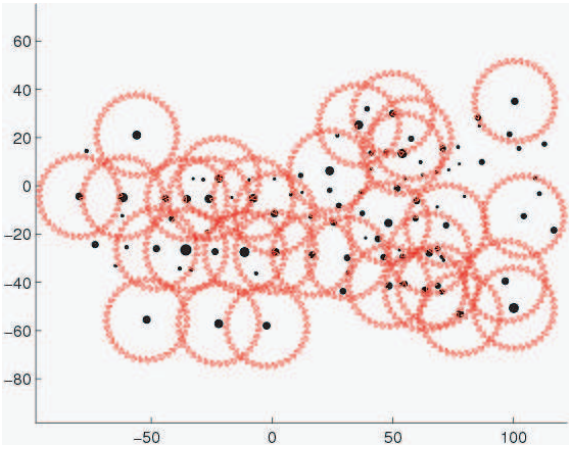


Figure 5.2: Pairing driven techniques

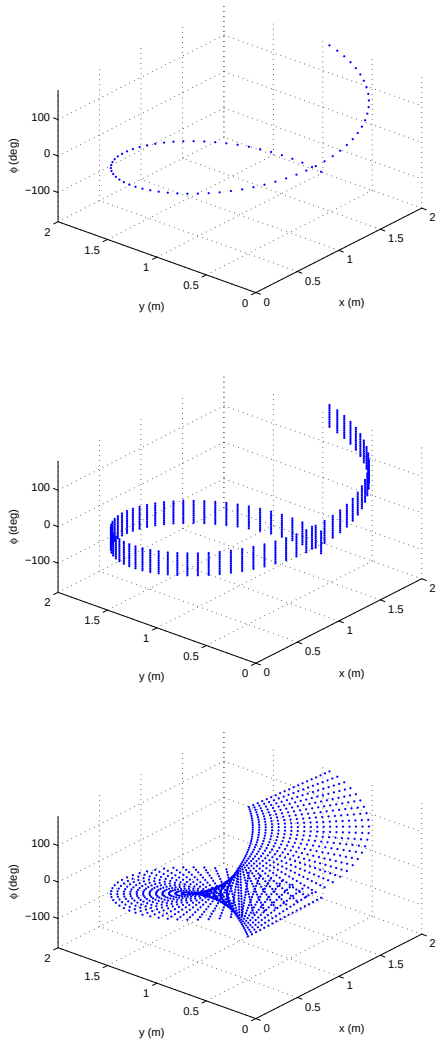
samples) it points out. The amount of votes the measurement casts depends on the type of map features and sensor measurements. Fig. 5.3 shows the votes that a measurement would cast when detecting a 2D point feature at coordinates $(1.0, 0.0)$ with respect to the vehicle, when the feature is at absolute coordinates $(1.0, 1.0)$, and the sensor is range and bearing (for example a laser scanner), range-only (sonar), and bearing-only (a camera).

Both strategies are bound to obtain roughly the same results, and so from the point of view of robustness, `Loc_driven` and `Pair_driven` can be considered basically equivalent. Unary constraints can be easily incorporated to the algorithms to improve their efficiency. In the next section we discuss how the representation that we choose, both for the environment map and for the configuration space, will play a crucial role in the computational cost of these algorithms.

5.3 Computational complexity and robustness of global localization algorithms

5.3.1 Computational complexity of `Loc_driven`

The representation of the environment map has a very important effect in the complexity of `Loc_driven`. This algorithm traverses the whole configuration space, computing the support for each of the $s = n_x \cdot n_y \cdot n_\phi$ candidate locations. It gathers evidence by predicting the absolute location of each of the m measurements in trying to find any compatible feature. If the absolute map feature coordinates are maintained *explicitly*, in a location vector $(\mathbf{f}_1 \cdots \mathbf{f}_n)^t$, a naive implementation of `any_compatible_feature` would sequentially consider all the n alternative features in $\mathbf{F}$. The computational complexity



$\mathbf{X}_{ij} = \texttt{hypothesize\_locations}(\mathbf{f}_j,\ \mathbf{z}_i)$:
; $\mathbf{f}_j = (x_j,\ y_j)$

; For Range and Bearing:
; $\mathbf{z}_i = (\rho_i,\ \theta_i)$
;
$\mathbf{X}_{ij} = [\,]$
for $\phi = \phi_{min}$ to ϕ_{max} step ϕ_{step} **do**
  $(x_i,\ y_i) = \texttt{pol2cart}(\theta_i + \phi,\ \rho_i)$
  $\mathbf{X}_{ij} = [\mathbf{X}_{ij}; [(x_j - x_i)\ (y_j - y_i)\ (\phi)]]$
end for
; For Range Only:
; $\mathbf{z}_i = \rho_i$
;
$\mathbf{X}_{ij} = [\,]$
for $\phi = \phi_{min}$ to ϕ_{max} step ϕ_{step} **do**
  for $\theta = \theta_{min}$ to θ_{max} step θ_{step} **do**
    $(x_i,\ y_i) = \texttt{pol2cart}(\theta + \phi,\ \rho_i)$
    $\mathbf{X}_{ij} = [\mathbf{X}_{ij}; [(x_j - x_i)\ (y_j - y_i)\ (\phi)]]$
  end for
end for
; For Bearing Only:
; $\mathbf{z}_i = \theta_i$
;
$\mathbf{X}_{ij} = [\,]$
for $\phi = \phi_{min}$ to ϕ_{max} step ϕ_{step} **do**
  for $\rho = \rho_{min}$ to ρ_{max} step ρ_{step} **do**
    $(x_i,\ y_i) = \texttt{pol2cart}(\theta_i + \phi,\ \rho)$
    $\mathbf{X}_{ij} = [\mathbf{X}_{ij}; [(x_j - x_i)\ (y_j - y_i)\ (\phi)]]$
  end for
end for

Figure 5.3: Hypothesized vehicle locations in the X space, generated by a measurement of a 2D point feature at $(1.0, 1.0)$, whose location relative to the vehicle is at $(1.0, 0.0)$, when the measurement gives range and bearing (top), range only, such as polaroid sonars (center), and bearing only (bottom). Algorithms on the right show the actual implementation of function $\mathbf{X}_{ij} = \texttt{hypothesize\_locations}(\mathbf{f}_j,\ \mathbf{z}_i)$ for each case.

Sensor	`Loc_driven`	`Pair_driven`	`Pair_driven`/`Loc_driven`
Range and bearing	$O(n_x \cdot n_y \cdot n_\phi \cdot m)$	$O(n \cdot m \cdot n_\phi)$	$n/(n_x \cdot n_y)$
Range-only	$O(n_x \cdot n_y \cdot n_\phi \cdot m \cdot n_\theta)$	$O(n \cdot m \cdot n_\phi \cdot n_\theta)$	$n/(n_x \cdot n_y)$
Bearing-only	$O(n_x \cdot n_y \cdot n_\phi \cdot m \cdot n_r)$	$O(n \cdot m \cdot n_\phi \cdot n_r)$	$n/(n_x \cdot n_y)$

Figure 5.4: Comparative computational cost of `Loc_driven` and `Pair_driven`

would then be

$$O(n_x \cdot n_y \cdot n_\phi \cdot m \cdot n)$$

.

In some cases, such as when features are 2D points and the sensor gives range and bearing measurements, the map can be preprocessed into a tree, and then finding a match for a predicted measurement at a hypothesized location can be computed in $O(\log n)$ instead of $O(n)$.

Alternatively, an *implicit, indexed* representation of the absolute feature locations can be maintained, for example using an occupancy grid or binary image. Using such a representation, determining whether a predicted measurement is present in the map amounts to consulting the corresponding element(s) in the occupancy grid or image. If we have a range and bearing sensor, instead of incurring in a $O(n)$ cost, we incur in $O(1)$. For a range-only sensor, we have to consult elements of the occupancy grid in an arc of a circle around the hypothesized vehicle location at the given range. Discretizing this arc in n_θ angles, the cost will be $O(n_\theta)$[1]. In the case of bearing-only sensors, and discretizing the viewing direction in n_r steps, we will have to consult $O(n_r)$ elements. Table 5.4 contains a summary of the resulting computational complexities.

5.3.2 Computational complexity of `Pair_driven`

Algorithm 9, `Pair_driven`, considers each of the m measurements in turn, and then each of the n map features in turn. It hypothesizes the set $\mathbf{X}_{ij}$ of vehicle locations from where feature $\mathbf{f}_j$ would produce measurement $\mathbf{z}_i$. If the s location samples are randomly drawn, $\mathbf{X}$ will be represented a set of location hypotheses with *explicit* coordinates. Again, a first implementation of function `compute_compatible_locations`, would cost $O(s)$; it would sequentially consider each alternative location sample $\mathbf{x}$ in $\mathbf{X}$ to decide whether it is compatible with any of $\mathbf{X}_{ij}$. The total computational complexity of `Pair_driven` would then be

$$O(m \cdot n \cdot s) = O(m \cdot n \cdot n_x \cdot n_y \cdot n_\phi)$$

.

[1] In the case of indoor sonar, the span of θ is around $30deg$. In the pure range-only case, the span of θ will be $360deg$. In this case there is no need to sample ϕ in neither algorithm because the vehicle orientation cannot be recovered from the observations.

Alternatively, if grid sampling is used, samples of the configuration space are drawn in a systematic way, evenly placed in the center of equally-sized *tiles* that fully cover the configuration space and exhibit a lattice structure, and can easily be represented by a location grid. This allows us to compute the closest sample to a given location hypothesis, required by function `compute_compatible_locations` in constant time. Thus, using a grid sampling method, `Pair_driven` can be bilinear in both the number of features and the number of measurements, $O(n \cdot m)$. The number of hypothesized locations will be $O(n_\phi)$ for range and bearing sensors, in $O(n_\phi \cdot n_\theta)$ for range-only sensors, and in $O(n_\phi \cdot n_r)$ for bearing-only sensors (see fig. 5.3). Since sensors are local, n_ϕ and n_r are constants.

5.3.3 `Loc_driven` .vs. `Pair_driven`

How do both algorithms compare? Consider the case of range and bearing sensors: the computational cost of `Loc_driven` will be $O(n_x \cdot n_y \cdot n_\phi \cdot m)$, while `Pair_driven` will be $O(n \cdot m \cdot n_\phi)$. Their ratio then will be equal to ρ, the density of features per grid cell unit:

$$\frac{\texttt{Pair\_driven}}{\texttt{Loc\_driven}} = \frac{n \cdot m \cdot n_\phi}{n_x \cdot n_y \cdot n_\phi \cdot m} = \frac{n}{n_x \cdot n_y} = \rho$$

This will be the case for any type of sensor (see table 5.4). This means that the computational cost of algorithm `Pair_driven` will be a fraction of the cost of `Loc_driven` proportional to feature density. This makes it especially efficient in sparse environments, as we will see in section 5.4 Victoria Park, Sydney, considering an area of around $197m \times 93m$ there are 99 trees. If you decide to discretize the configuration space every $1.5m$, you will obtain 132×63, 8316 grid elements in position. If $n = 99$ then you can expect `Pair_driven` to run about 82 times faster.

5.3.4 A probabilistic analysis of the robustness of voting strategies

Global localization problems in fact pose a twofold question: (1) is the vehicle in the map? (2) if so, where? Both `Loc_driven` and `Pair_driven` compute the number of observation-feature pairings for each element in the set of location samples, so we further need to decide when the hypothesis with the highest number of votes can be accepted and thus the vehicle can be considered to be in the map. In [Neira 03], an empirical threshold $t = 6$ of six matchings was used to prevent false positives for the Victoria Park dataset.

In the following, we carry out a probabilistic analysis of voting strategies, that allows us to compute the probability of accepting a location hypothesis formed with matchings occurring at random.

Consider the `Pair_driven` algorithm in the case of range and bearing. Each observation casts $n \cdot n_\phi$ votes, that are to be distributed among some of the $n_x \cdot n_y \cdot n_\phi$ grid cells, or candidate locations. Assuming that features are randomly distributed in the environment, the probability that a given candidate randomly gets a vote from an observation is equal to the density of features in the map:

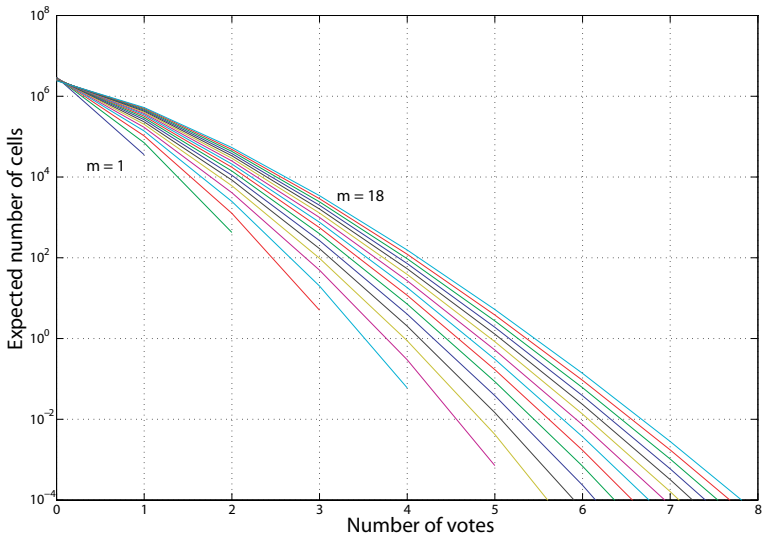


Figure 5.5: Number of location hypotheses expected to get a certain number of votes, for $m = 1$ to $m = 18$ observations.

$$\frac{n \cdot n_\phi}{n_x \cdot n_y \cdot n_\phi} = \rho$$

Thus, the probability that a given candidate is randomly voted by k of the m observations follows the binomial distribution [Papoulis 02] as follows:

$$p_m(k) = \frac{k}{k!(m-k)!} \rho^k \ (1-\rho)^{m-k}$$

As a result, the expected number of candidates in the configuration space having k votes will be:

$$r_{k,m} = n_x \cdot n_y \cdot n_\phi \cdot p_m(k)$$

Similar results can be obtained by considering the `Loc_driven` algorithm, as well as considering other sensors.

This analysis can be used to derive an adequate threshold for the acceptance of a location hypothesis. For the Victoria Park dataset, figure 5.5 shows, for different values of m, the number of cells expected to get k of the m votes. We can see that a fixed $t = 6$ pairings criteria is not uniformly restrictive: for a small number of observations, say $m = 6$, the expected number of cells with 6 random pairings is $8.5220e{-}6$, but for $m = 18$, this expected number climbs up to 0.1370.

We can limit the probability of accepting random false positives by setting a fix bound to $r_{k,m}$. For example, in order to have $r_{k,m} \leq 10^{-2}$ random false positives, for $m = 5$ the threshold t should be set to 5 , while for $m = 18$, t should be set to 7.

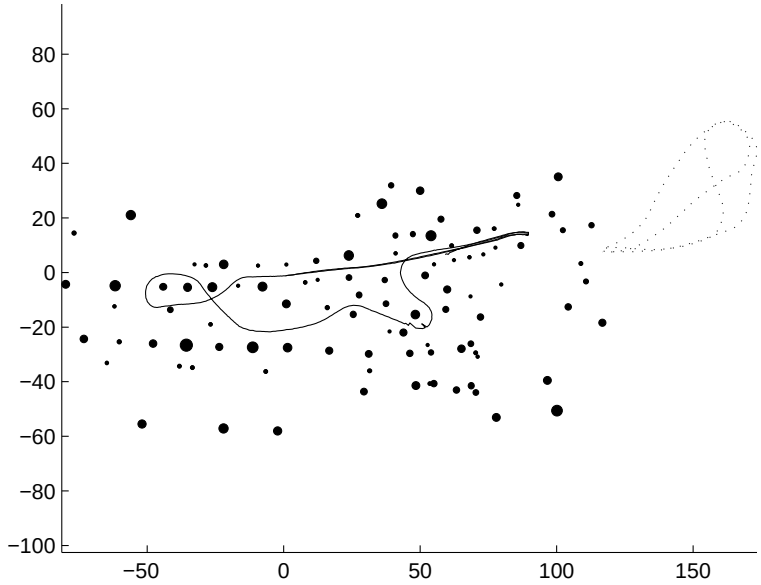


Figure 5.6: Stochastic map of 2D points built until step 1000 with n = 99 features. Reference vehicle trajectory inside the map (solid line) and outside of map (dashed line). Tree radiuses × 5; distances are in meters.

5.4 Experiments

An experimental comparison of both algorithms previously detailed is carried out using the Victoria park dataset obtained by Guivant and Nebot [Guivant 02]. We used 2500 steps of the trajectory of an outdoor vehicle equipped with a laser sensor along Victoria Park, Sydney. Point features, corresponding to trees are segmented from the scan using the `find_trees` algorithm [Guivant 02]. A stochastic map of $n = 99$ point features was generated with the first 1000 steps (fig. 5.6). The remaining steps (1001 to 2500) were used in the relocation algorithms, `Loc_driven` and `Pair_driven`. In this way, the statistical independence between the scans and the stochastic map is guaranteed. To verify the vehicle locations calculated by our algorithms, we obtained a reference solution running continuous SLAM until step 2500. The number of observations gathered from each position of the vehicle ranges between $m = 3$ and $m = 18$. Figure 5.7 shows the segmented trees for a scan of $m = 16$ measurements corresponding to step 1888.

Both algorithms are executed using a grid sampled configuration space of resolution $1.5\,m$ for x and y, and 1^o for ϕ. Fig. 5.8 shows a summarized table of votes corresponding to the result of executing `Pair_driven` at step 1888. For each grid cell in position, the corresponding image pixel depicts the maximum number of votes of all orientations corresponding to that position. For this example there is only one cell solution with 16 votes, represented by a black pixel in the image, and only one cell with 7 votes, a neighbor of the most voted cell (a predictable tessellation effect). The remaining cells contain less than 6 votes, as predicted by our probabilistic analysis (fig. 5.5).

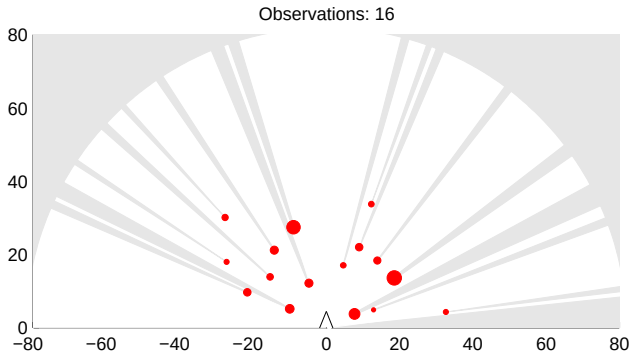


Figure 5.7: Segmented trees by the algorithm at step 1888. Tree radiuses × 5; distances are in meters.

In order to compare the computational complexity of `Loc_driven` and `Pair_driven` we have executed the algorithms for all the test steps. `Loc_driven` has been implemented representing the map using an occupancy grid. Figure 5.9 shows the mean running time of each algorithm versus the number of measurements. Both algorithms were implemented in MATLAB, and executed on a Pentium IV, at 2.8GHz. Algorithm `Pair_driven` is faster than `Loc_driven` by a factor of around 60. The differences of this result with a predicted 80 times gain in efficiency for this data are probably due to slightly different implementations of the operations in each algorithm, as well as memory management issues in MATLAB array operations.

In [Neira 03] it was shown that, for the same Victoria Park dataset, with less than 6 pairings there is insufficient evidence to assert that the vehicle is within map limits without having false positives. In order to compare the results of this work with the `RS` algorithm in [Neira 03], we first consider the criteria of $t = 6$ pairings as a threshold for accepting the solution of the algorithms.

Of the 1500 test steps, we consider 737 steps within map limits (see fig. 5.6, continuous line). In 74 steps of those (10.1%), the number of segmented trees was less than six. In 98 steps (13.3%) there are six or more measurements, but our algorithm finds less than six pairings. Thus, there is a total of 172 false negatives (23.4%). In the 565 remaining cases (76.6%), the algorithm found six or more pairings, and the solution obtained was always consistent with the reference solution, without false positives (fig. 5.10). These results show that the `Pair_driven` algorithm is slightly more prone to *false negatives* than the `RS` algorithm reported in [Neira 03]. In that work, in 604 cases (82%), RS finds six or more pairings, in 39 more cases. The difference is expected, since it is well known that voting

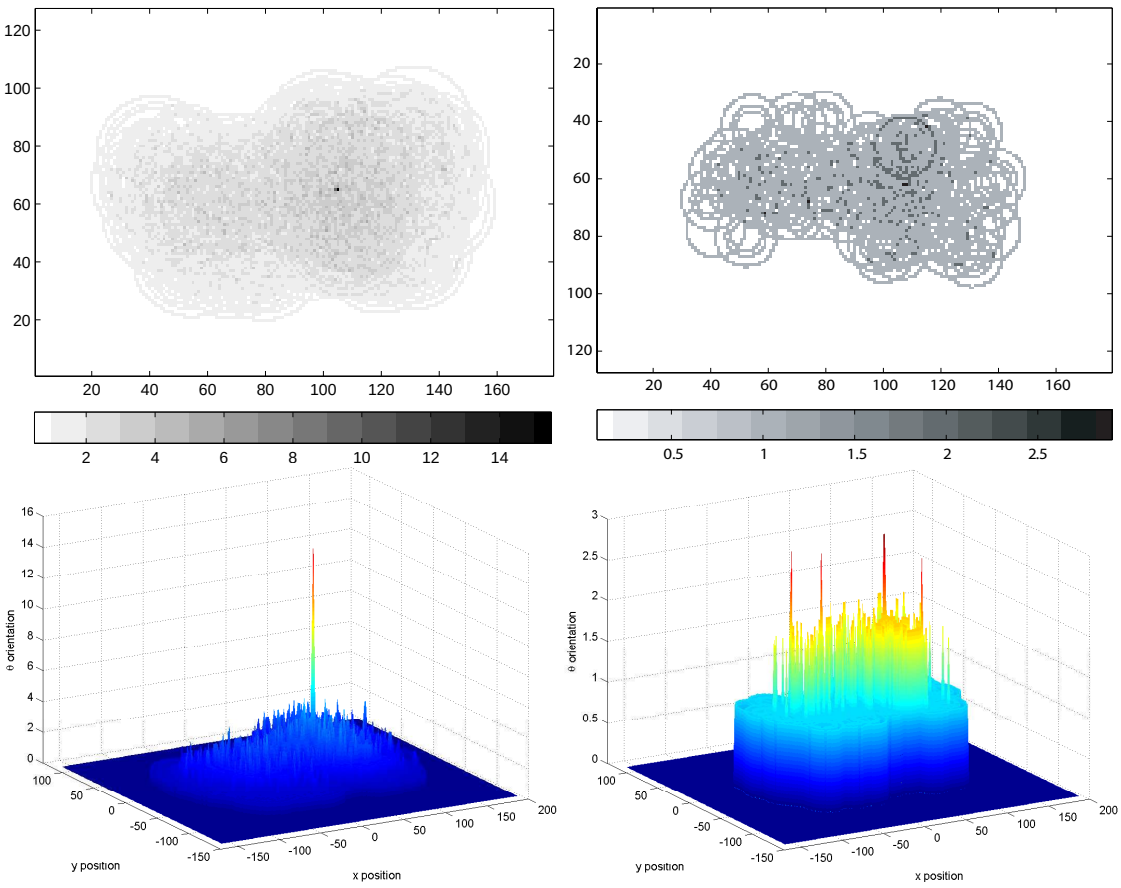


Figure 5.8: Cumulative voting table for step 1888 (m = 16) with dark pixel representing an unique hypothesis solution with votes = 16 (left). In step 1870 the sensor obtains m = 3 measurements of the environment such as the algorithm produces multiple hypothesis (right).

algorithms using strategies of the type of the Hough Transform are sensitive to tessellation effects [Grimson 90b]. In some cases, some of the votes for the correct solution may fall in bordering cells if the solution is close to the border. Further work will be necessary to refine the determination of the resolution of the grid.

In these results, the false negative rate is rather high. The reason is that, in steps in which there are less than 6 observations, the answer is always negative, because of insufficient data. We can improve the results if we use the variable threshold described in subsection 5.3.4. For steps with $m = 4$ observations, we consider a threshold $t_4 = 4$; the expected number of random hypotheses will be 0.0601. For larger values of m, we will set t_m so that $r_{k,m}$, the expected number of false cells, will be less than 10^{-2}: $t_m = 5$ for $m = 5 \ldots 6$, $t_m = 6$ for $m = 7 \ldots 12$, and $t_m = 7$ for $m = 13 \ldots 18$. Using this threshold, we obtain true positive solutions in 609 cases, around 82.6%. We still consider steps with $m = 1 \ldots 3$ observations as having insufficient information, 22 cases. Once a solution is found (fig.

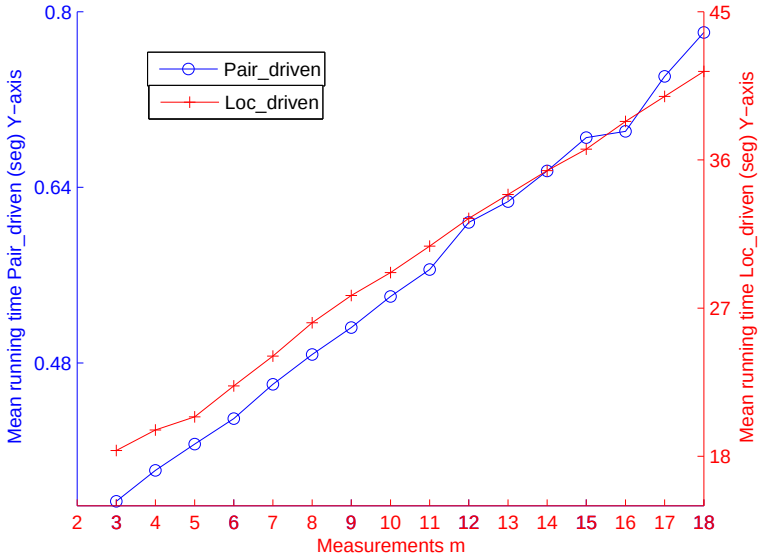


Figure 5.9: Mean running time of each algorithm versus the number of measurements.

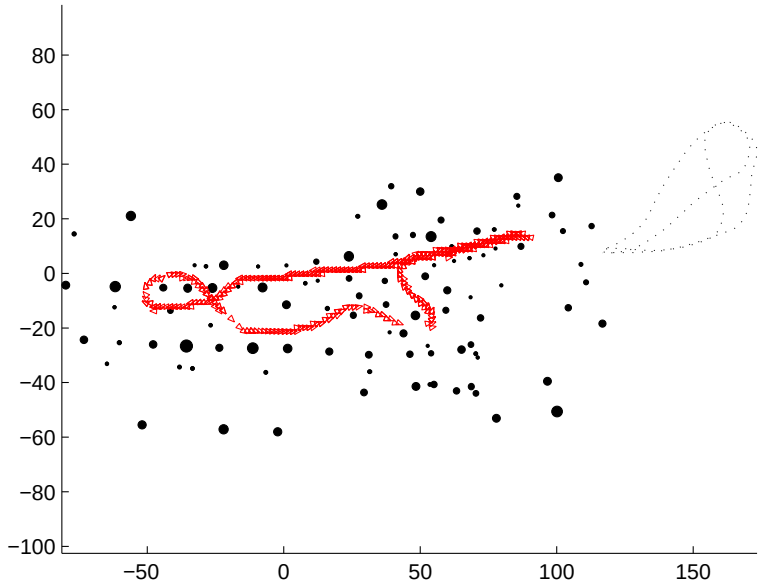


Figure 5.10: True positive solutions: 565 out of 737 steps. Missing steps (false negatives) are due to insufficient number of measurements, or insufficient number of pairings.

5.10), a Kalman filter is used to refine the vehicle location.

We also tested the 580 steps in which the vehicle is outside the map limits (fig. 5.6, dashed line), although in a similar environment. Thus, the sensor detects features that are not to be found in the map. Using the new threshold, there are no false positive answers.

5.5 Discussion

Surprisingly, in contrast to most of the global localization literature, in our experiments the system is able to localize the vehicle in *one-shot*. The reasons are that the environment is sparse, without symmetries, and the laser scanner is very precise. More complex scenarios will probably require to move the vehicle in order to acquire sufficient information for localization.

Incorporating vehicle motion to consider measurements obtained at more than one location can be easily considered in the algorithm. Additionally, the method can be combined with incremental sampling strategies [Lindemann 04] to allow incorporating the decision on the grid resolution into the voting algorithm. This would allow the algorithm to quickly identify the most promising regions of the configuration space using low resolution, and then concentrating on these regions using high resolution, further improving the computational cost of global localization. A generalization to different types of features is feasible, along with the incorporation of a probabilistic framework to include the uncertainty in map and the measurements.

This technique may also be useful as a bootstrapping step for Monte Carlo methods. With no prior information on vehicle location in a very large environment, the *pairing-driven* algorithm could be used to determine promising areas within the environment. A Monte Carlo algorithm could then take control focusing on those areas with a smaller number of random particles.

Chapter 6

Applications: 6DOF SLAM with Stereo-In-Hand

The growing interest in developing 3D Visual SLAM systems to explore large environments, requires the use of scalable techniques. This complementary chapter describes a system that can carry out SLAM in large indoor and outdoor environments using a stereo pair moving with 6DOF as the only sensor. The proposed SLAM system generates sequences of conditionally independent local maps, a new adopted framework that allows us to share information related to the camera motion and common features being tracked. The system computes the full map using a variant of the D&C algorithm of chapter 3, which allows constant time operation most of the time, with linear time updates to compute the full map. Unlike current visual SLAM systems that use either bearing-only monocular information or 3D stereo information, the system accommodates both monocular and stereo. Textured point features are extracted from the images and stored as 3D points if seen in both images with sufficient disparity, or stored as inverse points otherwise. This allows us to map both near and far features: the first provide distance and orientation, and the second orientation information. Unlike other vision only SLAM systems, stereo does not suffer from 'scale drift' because of unobservability problems, and thus no other information such as gyroscopes or accelerometers is required in our system. To demonstrate the robustness and scalability of our system, we show experimental results in indoor and outdoor urban environments of 210m and 140m loop trajectories, with the stereo camera being carried in hand by a person walking at normal walking speeds of 4-5km/hour.

6.1 Introduction. State of the art in visual SLAM

The interest in using cameras in SLAM has grown tremendously in recent times. Cameras have become much more inexpensive than lasers, and also provide texture rich information

about scene elements at practically any distance from the camera. 6DOF SLAM systems based on 3D laser scanners plus odometry have been demonstrated feasible both indoors and outdoors [Folkesson 06; Nüchter 07], as well as vision aided by laser without odometry [Ellekilde 07] and vision aided by an inertial navigation system [Kim 03; Bryson 07]. But in applications where it is not practical to carry heavy and bulky sensors, such as egomotion for people tracking and environment modeling in rescue operations, cameras seem the only light weight sensors that can be easily adapted to helmets used by rescuers, or simply worn.

Current Visual SLAM research has been focused on the use of either monocular or stereo vision to obtain 3D information from the environment. Quite a few monocular visual SLAM systems have been demonstrated to be viable for small environments [Deans 00; Fitzgibbons 02; Bailey 03; Kwok 04; Kwok 05; Sola 05; Lemaire 05; Jensfelt 06; Gil 06; Davison 07]. Most are essentially standard EKF SLAM systems, and vary in the technique used to initialize a feature, given the partiality of the bearing only information provided by one camera, or in the type of interest points extracted from the images (be it Harris corners, Shi-Tomasi corners, SIFT features, or some combination). Some works have also considered segment features [Folkesson 05; Smith 06]. Larger environments have been tackled in Hierarchical Visual SLAM [Clemente 07].

A single camera is used in all of these systems, and although very distant features are potentially detectable, scale unobservability is a fundamental limitation. Either the scale is fixed in some way (for example by observing a known object [Davison 07]), or drift in scale can occur as is reported in the Hierarchical Visual SLAM system [Clemente 07]. Panoramic cameras are also being used in visual SLAM [Lemaire 07b; Goedemé 07]. Here the limitation of scale unobservability is overcome using an additional stereo vision bench for motion estimation between consecutive frames. In the work of [Royer 07] only monocular images are used. Mapping is achieved using a batch hierarchical bundle adjustment algorithm to compute all camera as well as interest points locations. The scale is introduced in the system by manually entering the length of the path.

Stereo visual systems provide scale through the baseline between the cameras, known from calibration. Davison and Murray demonstrated the first active stereo visual SLAM system [Davison 98; Davison 02; Davison 01]. It is based on standard EKF and thus also has low scalability. Under restrictive planar environment assumptions, [Iocchi 00] built an environment map using stereo. [Se 02] demonstrated a visual stereo SLAM system using SIFT features in a small laboratory environment. This system is also unlikely to scale adequately to large environments or work in more challenging outdoor scenarios as cross-correlations were neglected for computational reasons. In [Jung 03; Hygounenc 04] the authors demonstrate an autonomous blimp system for terrain mapping using stereo as the only sensor, also using a standard EKF SLAM algorithm. [Saez 05] presented a 6DOF stereo visual SLAM system where egomotion estimation is done by a 3D point matching algorithm, and mapping through a global entropy minimization algorithm in indoor orthogonal scenarios, with difficult extension to more complex non-orthogonal environments.

[Sim 05; Sim 07] describe a dense visual SLAM system using Rao-Blackwellized Particle Filters and SIFT features; a similar effort in using Rao-Blackwellized Particle Filters and SIFT features for visual SLAM was reported in [Gil 06]. Visual odometry (SFM) is used to generate proposals for the sensor motion and global pose estimation algorithms for loop closing. This system works in either monocular or stereo mode, with cameras mounted on a robot moving in 2D; sensor trajectories with 6DOF will require large amounts of particles for their representation. In [Lemaire 07a] authors also compare the advantages of separate monocular and stereo approaches in traditional SLAM frameworks. In this chapter we show the advantages of being able to accommodate both monocular and stereo information in carrying out 6DOF SLAM with a hand-held camera. In the works of [Sola 07] and [Lemaire 07b] it is also pointed out that combining visual information at close range as well as at infinity should improve the performance of visual SLAM.

Since the initial results presented in [Zhang 92] great progress has been made in the related problem of visual odometry [Simond 04; Nister 06; Comport 07; Maimone 07]. Visual odometry systems have the important advantage of constant time execution. Furthermore, during exploratory trajectories, in which an environment feature is seen for a certain window of time and never more, visual odometry can obtain the same precision in the estimation of the sensor location as a SLAM system, with a great reduction in cost. Unfortunately, visual odometry does not cope with loop closings, and thus eventual drift in these cases is inevitable. Stereo visual odometry combined with GPS can result in a mapping system that avoids long term drift [Agrawal 06; Konolige 06], but unfortunately GPS is not always available. Improving the precision in sensor location through loop closing is one of the main advantages of SLAM.

As mentioned in this thesis, the important limitation of current SLAM systems that use the standard EKF algorithm is that when mapping large environments, very soon they face computational as well as consistency problems [Castellanos 04]. Many efforts have been invested in reducing the $O(n^2)$ cost of the EKF updates. Algorithms that use the information form and thus the state and covariance are not readily available [Thrun 04; Eustice 05; Frese 05; Frese 07; Dellaert 06], except in the recent version of iSAM presented by [Kaess 07]. The Divide and Conquer algorithm first presented in [Paz 07a; Paz 07b; Paz 08] and in this thesis in chapter 3 is able to compute the covariance form of the stochastic map in amortized time linear with the size of the map, improving further the consistency of the solution. However, in these systems, local maps are required to be statistically independent. This requires creating a new local map from scratch every time the current local map size limit has been reached. Consequently, no sharing is possible of valuable information in 6DOF visual SLAM, such as the camera velocity, or information about features being currently tracked. This drawback is also common to the Hierarchical SLAM algorithm of [Estrada 05].

Conditional Independent local maps were conceived currently and applied to the Visual SLAM problem in [Piniés 08]. They solves the problem of sharing common elements making a more stable joining between maps. Their major contribution is the use of a clearer covariance space structure for local maps in diagonal form, which allows evident linear

updates whereas memory grows linearly as well. The correlations recovering is formulated without any approximations using the back-propagating concept: when common elements are previously seen, the information is back propagated in order to obtain the exact solution. Even so, it was restricted to move in loop trajectories given the complexity it reaches when back-propagating information through all maps.

In this chapter we describe a robust and scalable 6DOF visual SLAM system that can be carried in hand at normal walking speeds of 4-5km/hour, and used to map large indoor and outdoor environments. In section 6.2 we summarize the main charateristics of the system. Section 6.3 describes the details of the visual SLAM system that provides the sequence of conditionally independent local maps; the basic building blocks of our mapping algorithm. This algorithm, Conditionally Independent Divide and Conquer SLAM, is explained in section 6.4. Section 6.5 shows the two experiments carried out to test the system, an indoor 200m loop and an outdoor 140m loop. In section 6.6, the results obtained are discussed.

6.2 The proposal

The fundamental characteristics of the system described in this chapter are:

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

2. Nearby scene points provide scale information through the stereo baseline, eliminating the intrinsic scale unobservability problem of monocular systems.

3. Conditionally Independent Divide and Conquer SLAM is used, a novel SLAM algorithm that allows us to maintain both camera velocity information and current feature information during local map initialization. This adds robustness to the system without sacrificing precision or consistency in any way. Being a Divide and Conquer algorithm, it also allows linear time execution, enabling the system to be used for large scale indoor/outdoor SLAM.

The 6DOF hardware system consists of a stereo camera carried in hand and a laptop to record and process a sequence of images (fig. 6.1). Since the camera moves in 6DOF, the camera state is defined using 12 variables: camera position in 3D cartesian coordinates, camera orientation in Euler angles, and linear and angular velocities. It is known that a stereo camera can provide depth estimation of points up to a certain distance, determined by the baseline between left and right cameras. Therefore, two regions can be differentiated: a region close to the cameras and visible by both, in which stereo behaves as a range and bearing sensor. The second is the region of features far from the cameras

Figure 6.1: Stereo vision system used to acquire the image sequences. Picture on the left shows the experimental setup during the data acquisition for the indoor experiment.

or seen by only one, in which the stereo becomes a monocular camera, only providing bearing measurements of such points. To take advantage of both types of information, 3D points and inverse depth points (introduced in [Montiel 06]) are combined in the state vector in order to build a map and estimate the camera trajectory. The system produces sequences of local maps of limited size containing both types of features using an EKF SLAM algorithm. As we detail in section 6.4, these local maps are joined into a full map using the Conditionally Independent Divide and Conquer SLAM algorithm, obtaining as final result a full stochastic map containing all tracked features and the final and intermediate camera states from each local map. This system is highly scalable: local maps are built in constant time, regardless of the size of the environment, and the Conditionally Independent Divide and Conquer algorithm requires amortized linear time.

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

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

6.3 The Visual SLAM System

6.3.1 State Representation

The state vector that represents a local submap $\mathbf{x}_B$ contains the final camera location $\mathbf{x}_c$ and the location of all features $\mathbf{x}_{f_{1:n}}$ with respect to the map base reference B, the initial camera location. Some features are codified using the *Inverse Depth (ID) parametrization*

[Civera 08] that model points that are at the infinity in $\mathbf{x}_{ID}$. Additionally, cartesian $3D$ *parametrization* is used to represent depth points in $\mathbf{x}_{3D}$:

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

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

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{r} \\ \Psi \\ \mathbf{v} \\ \mathbf{w} \end{bmatrix} \tag{6.1}$$

Image corners classified as depth points are transformed to 3D points, given the disparity information provided by the stereo pair. Subsection 6.3.4 describes the criterion adopted to select points as depth points. Since the stereo camera provides rectified images, the back-projection equations to obtain a 3D point are based on a pinhole camera model which relates image points and 3D points using the following transformation function:

$$\begin{aligned} \mathbf{x}_{3D} &= f(u_r, v_r, u_l, v_l) \\ &= [x, y, z]^T \\ &= \left[\frac{b(u_r - u_0)}{d}, \frac{b(v_r - v_0)}{d}, \frac{fb}{d}\right]^T \end{aligned} \tag{6.2}$$

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

Given the camera location $\mathbf{x}_{c_i}$, an inverse depth point is defined as in [Montiel 06]:

$$\mathbf{x}_{ID} = \begin{bmatrix} \mathbf{r}_i \\ \theta_i \\ \phi_i \\ \rho_i \end{bmatrix} \tag{6.3}$$

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

6.3.2 Selection and Management of Trackable points

To ensure tracking stability of map features, distinctive points have to be selected. Following a similar idea as the one presented in [Davison 03], we use the Shi-Tomasi variation of the Harris corner detector to select good trackable image points and their corresponding 11×11 surrounding patch.

From the first step, the right image is split using a regular grid; the point with the best detector response per cell is selected, see fig. 6.2. At each step, we use only those features that fall in the FOV of the camera when they are projected along with their uncertainties on right and left images. Using the patch associated with each feature, a matching search based on normalized cross-correlation is performed inside the projected uncertainty region. This is the outline of an active search as introduced in [Davison 02]. During the following steps, those cells that become and remain empty for a given time are monitorized to initialize a new feature when a good point is detected. In this way features can be uniformly distributed in the image, improving the amount of information gathered from the scene and therefore the map estimate. The approach is accompanied by a feature management strategy so that non-persistent features are deleted from the state vector to avoid an unnecessary growth in population.

6.3.3 Measurement Equation

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

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

$$\begin{aligned} \mathbf{z}^{r_i}_{ID} &= h^r_{ID}(\mathbf{x}_c, \mathbf{x}^i_{ID}) + \upsilon \\ &= projection(\ominus \mathbf{x}_c \oplus \mathbf{x}^i_{ID}) + \upsilon \end{aligned} \tag{6.4}$$

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

$$\begin{aligned} \mathbf{z}^{l_i}_{ID} &= h^l_{ID}(\mathbf{x}_c, \mathbf{x}^i_{ID}) + \upsilon \\ &= projection(\ominus \mathbf{x}_c \oplus \mathbf{x}_{c_r c_l} \oplus \mathbf{x}^i_{ID}) + \upsilon \end{aligned} \tag{6.5}$$

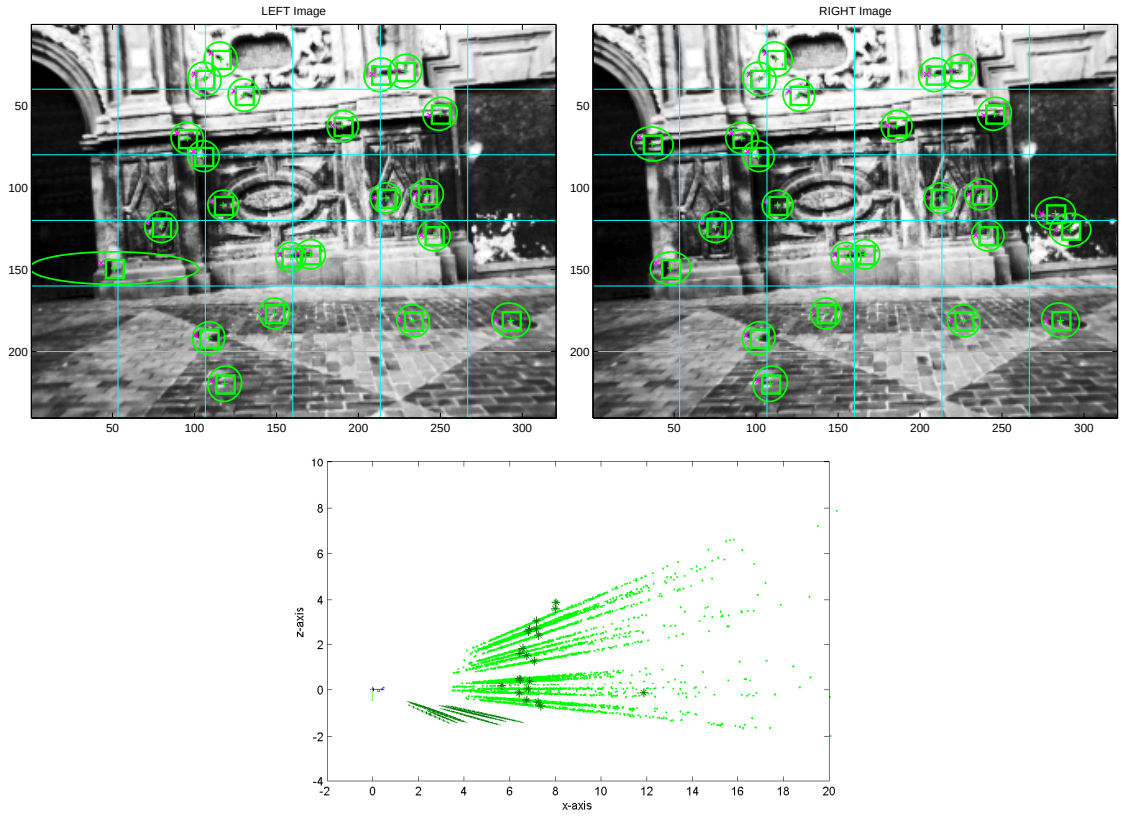


Figure 6.2: Points detected using a stereo camera. Projection of map features on both left (top left) and right (top right) images. We show feature uncertainties from a lateral perspective (bottom): 3D feature uncertainties are drawn using darker ellipses whereas we use samples to show the inverse depth feature uncertainties.

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

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

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

Note that we use $\oplus$ and $\ominus$ operators in order to carry out the corresponding compositions and inversions of transformations. They represent different transformations depending on the kind of parametrization used to express a feature. In [Tardós 02], the definitions for

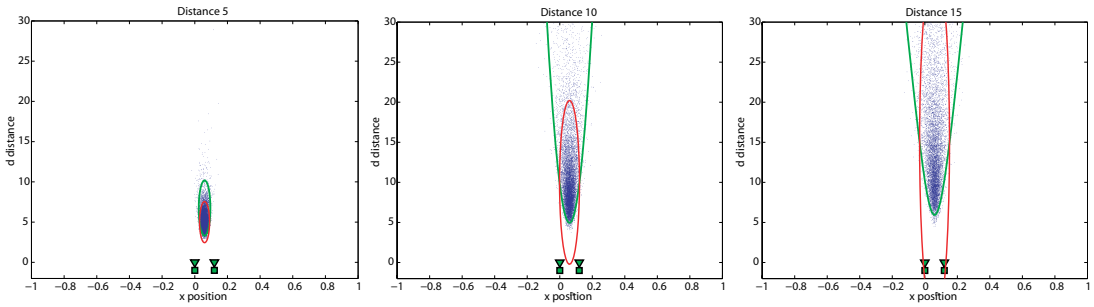


Figure 6.3: Simulated experiment of a point reconstruction from a stereo pair observation, for a point at $5m$ distance (left), $10m$ (middle) and $15m$ (right). The point clouds are samples from the real distribution of the point location, given that the pixel noise in the images is Gaussian. Dark red ellipses represent the uncertainty region for the point location when the back projection equations of a depth point are linearized. Light green regions represent the uncertainty in the point using the inverse depth parametrization.

2D transformations were introduced, dealing mainly with point features and line features. Appendix D describes an extension of these operations for 3D inverse depth and depth points. It also details the calculation of the corresponding Jacobians to propagate the uncertainties correctly.

Fig. 6.2 shows the prediction of those 3D and inverse depth features that fall inside the field of view of each of the cameras. A good advantage of using a stereo camera is that although a feature can disappear from the field of view of one camera, information to update the state is available if the feature can still be found in the other. As it will be shown in the experiments, this fact is of extreme importance when the camera rotates or turns around a corner, since features escape very fast from the FOV of a single camera making the estimation of the camera location in those moments very weak.

6.3.4 Depth points Vs. Inverse Depth points

Current research on Monocular SLAM has shown that the inverse-depth parametrization is suitable to represent the distribution of features at infinity as well as close points, allowing to perform an undelayed initialization of features. Despite its properties, each inverse depth point needs an over-parametrization of six values instead of a simpler three coordinates spatial representation [Civera 07]. This produces a computational overhead in the EKF. Working with a stereo camera, which can estimate the depth of points close to the camera, raises the subtle question of when a feature should be initialized using a $3D$ or an ID representation.

In order to clarify this issue we have designed a simulated experiment to study the effect of the linearization in both representations when a point is initialized using the stereo information. In this simulated experiment the variance of the pixel noise ($\sigma_p = 1pixel$) and the actual intrinsic parameters of the stereo camera used, such as the baseline, are taken into account to implement the simulation. The experimental setup consists of a stereo pair where the left camera is located at the origin of the reference frame, with its principal axis

pointing along Z and the X axis pointing to the right. The right camera is at $b = 12cm$ in X. We consider a point that is in the middle between both cameras at different distances in Z. Given a noisy pixel observation the uncertainty region of a reconstructed point is sampled and plotted in fig.6.3 for three different point distances: 5,10 and 15 meters. The uncertainty region of the 3D representation which is calculated using a linearization of eq. (6.2) and evaluated in the ground truth, is represented by the dark red ellipse. The corresponding uncertainty region of the linearized inverse depth representation is bounded by the light green lines in the plot. Notice that the inverse depth parametrization models very accurately the real uncertainty for the studied distances. However, although the dark ellipse covers the real distribution at 5 meters quite accurately, for longer distances the ellipse overestimates the uncertainty in the region close to the cameras and is overconfident for far distances.

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

Inverse depth features can be transitioned to 3D points reducing significantly the number of DOF. Conversion requires an analysis of the linearity of the functions that model both depth point and inverse depth point distributions. In [Civera 07] this issue is considered by using a linearity index. Such analysis makes it possible to decide when an inverse point distribution is well approximated with the over parameterized coding. Switching from inverse depth to depth depends on a linearity threshold derived from the analysis.

6.4 Conditionally Independent Divide and Conquer SLAM

Divide and Conquer SLAM (D&C) described in chapter 3 has proved to be a good algorithm in minimizing the computational complexity of EKF-based SLAM and improving consistency of the resulting estimate [Paz 07b]. The algorithm allows us to efficiently join several local maps in a single state vector using Map Joining in a Hierarchical tree structure. Local maps can be obtained in constant time, regardless of the size of the environment, and the map joining operations can be performed in amortized linear time. The D&C SLAM algorithm was however conceived for statistically independent sequences of local maps. This requires creating a new local map from scratch every time the current local map size limit has been reached. Consequently, it is not possible to share valuable information in 6DOF visual SLAM, such as the camera velocity, or information about features currently being tracked.

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

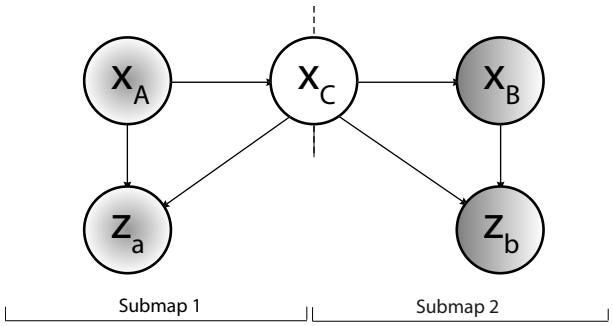


Figure 6.4: Bayesian network that describes the relations between two consecutive submaps

6.4.1 Conditionally Independent Local Maps

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

The novel technique to achieve these requirements is based on the concept of Conditionally Independent Local Maps (CI) presented by [Piniés 08]. A brief summary of the technique is presented here for completeness.

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

$$p(\mathbf{x}_A, \mathbf{x}_C|\mathbf{z}_a) = \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{A_a} \\ \hat{\mathbf{x}}_{C_a} \end{bmatrix}, \begin{bmatrix} P_{A_a} & P_{AC_a} \\ P_{CA_a} & P_{C_a} \end{bmatrix}\right) \tag{6.6}$$

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

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

$$p(\mathbf{x}_C|\mathbf{z}_a) = \int p(\mathbf{x}_A, \mathbf{x}_C|\mathbf{z}_a)\, d\mathbf{x}_A = \mathcal{N}(\hat{\mathbf{x}}_{C_a}, P_{C_a}) \tag{6.7}$$

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

$$p(\mathbf{x}_C, \mathbf{x}_B | \mathbf{z}_a, \mathbf{z}_b) = \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{C_{ab}} \\ \hat{\mathbf{x}}_{B_{ab}} \end{bmatrix}, \begin{bmatrix} P_{C_{ab}} & P_{CB_{ab}} \\ P_{BC_{ab}} & P_{B_{ab}} \end{bmatrix} \right) \tag{6.8}$$

where the subindexes in the estimates $\hat{\mathbf{x}}_{C_{ab}}$ and $\hat{\mathbf{x}}_{B_{ab}}$ reveal that both sets of observations $\mathbf{z}_a$ and $\mathbf{z}_b$ have been used in the estimation process. This means that submap 2 is updated with all the information gathered by the sensor. But observe that map 1 in eq. (6.6) has been updated with the observation $\mathbf{z}_a$ but not with the more recent observations $\mathbf{z}_b$.

Figure 6.4 shows a Bayesian network that describes the probabilistic dependencies between elements of submaps 1 and 2. As it can be seen, the only connection between the set of nodes ($\mathbf{x}_A$, $\mathbf{z}_a$) and ($\mathbf{x}_B$, $\mathbf{z}_b$) is through node $\mathbf{x}_C$, i.e. both subgraphs are *d-separated* given $\mathbf{x}_C$ [Bishop 06]. This implies that nodes $\mathbf{x}_A$ and $\mathbf{z}_a$ are *conditionally independent* of nodes $\mathbf{x}_B$ and $\mathbf{z}_b$ given node $\mathbf{x}_C$. Intuitively this means that if $\mathbf{x}_C$ is known, submaps 1 and 2 do not carry any additional information about each other.

6.4.2 Conditionally Independent Map Joining

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

$$\begin{aligned} p(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C | \mathbf{z}_a, \mathbf{z}_b) = & \qquad (6.9) \\ = \mathcal{N}\left(\begin{bmatrix} \hat{\mathbf{x}}_{A_{ab}} \\ \hat{\mathbf{x}}_{C_{ab}} \\ \hat{\mathbf{x}}_{B_{ab}} \end{bmatrix}, \begin{bmatrix} P_{A_{ab}} & P_{AC_{ab}} & P_{AB_{ab}} \\ P_{CA_{ab}} & P_{C_{ab}} & P_{CB_{ab}} \\ P_{BA_{ab}} & P_{BC_{ab}} & P_{B_{ab}} \end{bmatrix} \right) & \end{aligned}$$

Taking into account the submap conditional independence property, it can be demonstrated [Piniés 08] that the optimal map result of the joining can be computed using:

$$\begin{aligned} K &= P_{AC_a} P_{C_a}^{-1} \\ &= P_{AC_{ab}} P_{C_{ab}}^{-1} && (6.10) \\ \hat{\mathbf{x}}_{A_{ab}} &= \hat{\mathbf{x}}_{A_a} + K(\hat{\mathbf{x}}_{C_{ab}} - \hat{\mathbf{x}}_{C_a}) && (6.11) \\ P_{A_{ab}} &= P_{A_a} + K(P_{CA_{ab}} - P_{CA_a}) && (6.12) \\ P_{AC_{ab}} &= K P_{C_{ab}} && (6.13) \\ P_{AB_{ab}} &= K P_{CB_{ab}} && (6.14) \end{aligned}$$

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

6.4.3 Actual implementation for stereo

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

$$\mathbf{m}_i.\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_{R_iR_j} \\ \hat{\mathbf{v}}_{R_iR_j} \\ \hat{\mathbf{x}}_{R_iF_{1:m}} \\ \hat{\mathbf{x}}_{R_iF_{m+1:n}} \end{bmatrix} \tag{6.15}$$

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

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

$$\mathbf{m}_i.\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_{R_iR_j} \\ \hat{\mathbf{v}}_{R_iR_j} \\ \hat{\mathbf{x}}_{R_iF_{1:m}} \\ \hat{\mathbf{x}}_{R_iF_{m+1:n}} \\ \cdots \\ \ominus\hat{\mathbf{x}}_{R_iR_j} \oplus \hat{\mathbf{v}}_{R_iR_j} \\ \ominus\hat{\mathbf{x}}_{R_iR_j} \oplus \hat{\mathbf{x}}_{R_iF_{m+1:n}} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}_{A_a} \\ \cdots \\ \hat{\mathbf{x}}_{C_a} \end{bmatrix} \tag{6.16}$$

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

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

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

$$\mathbf{m}_j.\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{x}}_{R_jR_j} \\ \ominus\hat{\mathbf{x}}_{R_iR_j} \oplus \hat{\mathbf{v}}_{R_iR_j} \\ \ominus\hat{\mathbf{x}}_{R_iR_j} \oplus \hat{\mathbf{v}}_{R_iR_j} \\ \ominus\hat{\mathbf{x}}_{R_iR_j} \oplus \hat{\mathbf{x}}_{R_iF_{m+1:n}} \end{bmatrix} \tag{6.17}$$

where $\hat{\mathbf{x}}_{R_jR_j}$ represents the location of the camera in the new reference frame with initial zero uncertainty and zero correlation with the rest of the elements of the initial map.

Notice that the initial velocity brought from the previous map has been replicated twice. One of the copies will change as the camera moves through the new map carrying the current camera velocity. The other copy will remain fixed and, together with the transformed features, will be the common elements with the previous map. The same process is successively repeated with all local maps.

6.4.4 Continuous data association in each local map

Recent work on large environments [Clemente 07] has shown that the Joint Compatibility Test [Neira 01] helps avoiding map corruption in visual SLAM by rejecting measurements that come from moving objects. This framework turns out to be suitable in environments with a small number of observations. However, even though impressive results were registered, a Branch and Bound algorithm implementation of (**JCBB**) that uses the joint compatibility test has limited use when the number of observations per step is large. In chapter 4 we shown that, more efficient results can be obtained using the *Randomized Joint Compatibility* version **RJC** proposed in [Paz 07b], in which, in the spirit of RANSAC, a *Joint Compatibility* **JC** test is run with a fixed set of p randomly selected measurements. In this case, correlation between patches and individual χ^2 tests are used to obtain candidate matches. If all p measurements and their matches are jointly compatible, the Nearest Neighbor rule is applied to match the remaining measurements. Once a full hypothesis H is obtained, **JC** is checked to avoid false positives. The process is repeated t times with adaptive RANSAC, limiting the probability of missing a correct association.

6.4.5 Map matching

The property of sharing common elements solves the data association problem between consecutive local maps [Paz 07b]. This requires to solve data association only in loop closing situations. The Maximum Clique Algorithm of [Clemente 07] is used in order to detect a previously visited area. The algorithm finds correspondences between features in different local maps, taking into account the texture and the relative geometry between the features. If sufficient corresponding features are found, an ideal measurement equation that imposes the loop closing constraint is applied in the final map.

6.5 Experiments in Urban Outdoor and Indoor Environments

In order to demonstrate the robustness and scalability of the proposed visual SLAM system, two 320x240 image sequences were gathered with a Point Grey Bumblebee stereo system (see fig. 6.1). The system provides a 65 x 50 degree field of view per camera, has a baseline of 12cm, limiting the 3D point features initialization up to a distance close to $5m$.

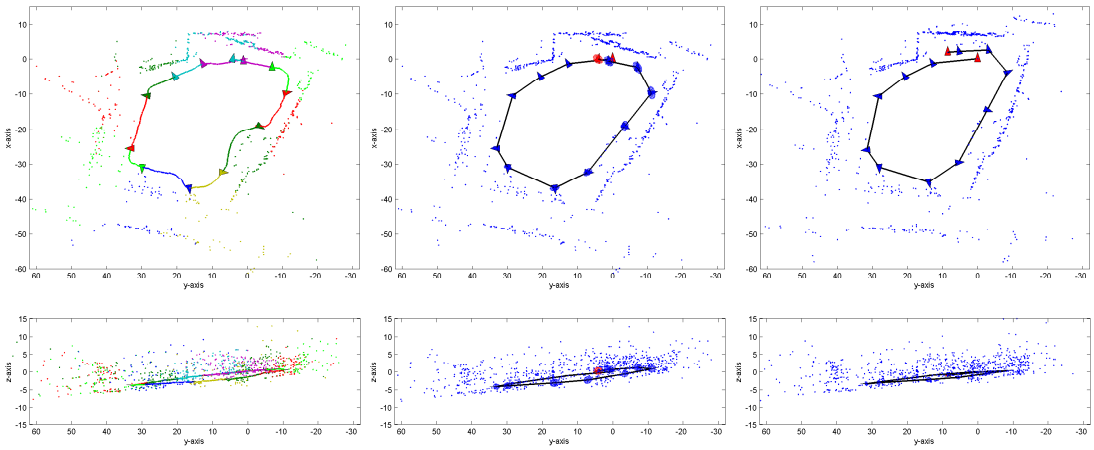


Figure 6.5: Outdoors experiment: 6DOF stereo SLAM on a public square. The sequence of CI Local maps is represented with respect to the initial reference (left); results obtained after running the D&C algorithm that joins and corrects the estimates (middle); final map obtained when the loop closing constraint is imposed (right). The scale factor and camera positions are well recovered thanks to the combined observations of 3D points and inverse depth points. Both a XY projection (first row) and a YZ projection (second row) are shown in order to illustrate the precision obtained.

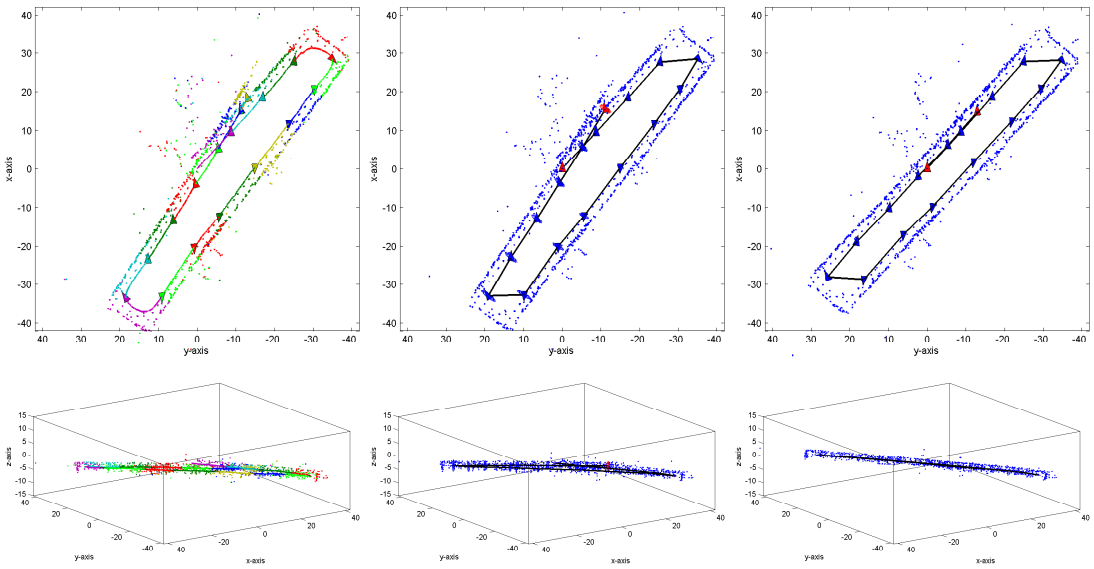


Figure 6.6: Indoor experiment: 6DOF visual SLAM along a building environment. Sequence of CI local maps with 100 features each (left), result after CI D&C joins and updates (middle) and final map estimate after loop closing (right).

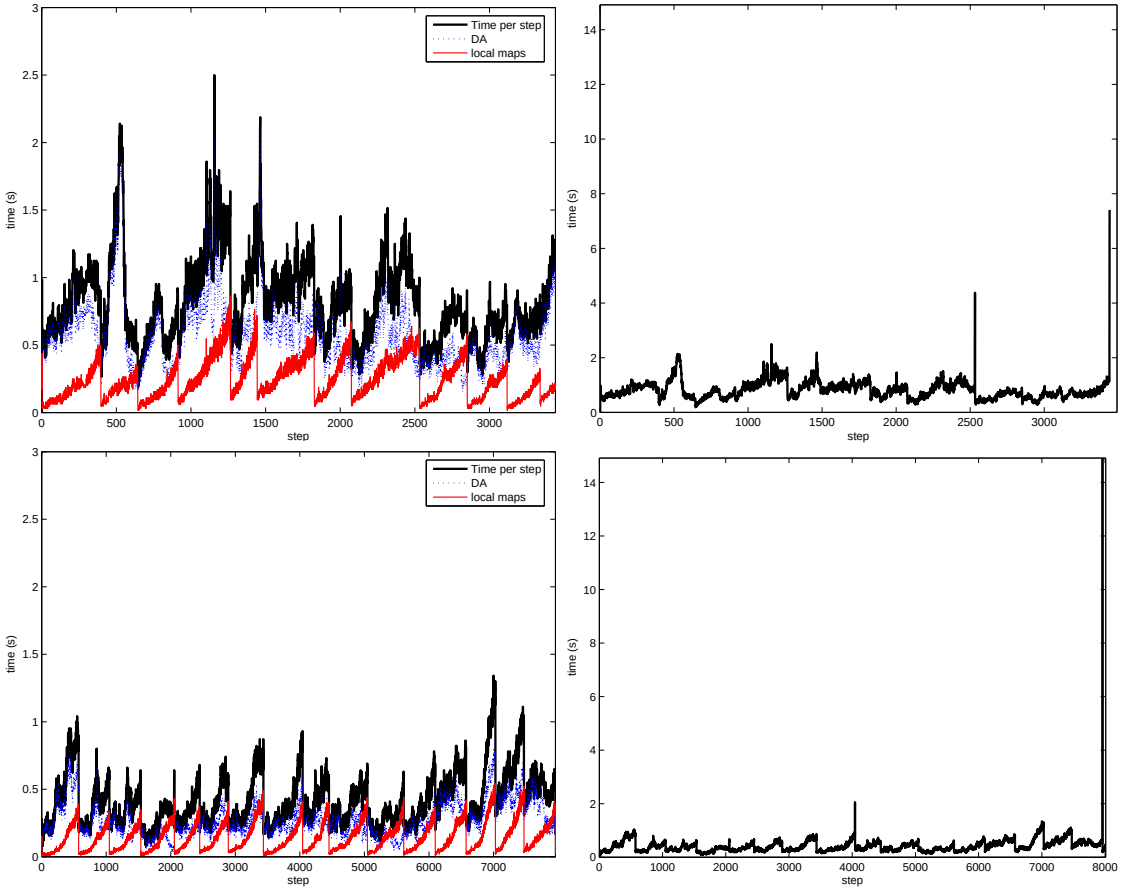


Figure 6.7: Running time per step of all associated processes: a detailed analysis of the Features extraction, Local Mapping (labeled as local maps) and Data Association (DA) times (left); total time per step where the peaks represent the joins performed by the CI D&C algorithm (right). Outdoor environment: the Public square (top). Indoor environment (bottom).

An indoor loop (at 48 fps) and an urban outdoor (at 25 fps) loop sequences were captured carrying the camera in hand, at normal walking speeds of 4-5km/hour. Both sequences were processed with the proposed algorithms on a desktop computer with an Intel 4 processor at 2,4GHz. The higher frame rate for the indoor experiment helps reducing the probability of mismatches given that the environment includes brick walls providing ambiguous texture information.

The outdoor sequence is composed of 3441 stereo pairs gathered in a public square of our home town. The full trajectory is approximately 140 meters long from the initial camera position. Figure 6.5 (left) shows the sequence of conditional independent local maps obtained with the technique described in section 6.4.1. Each map contains 100 features combining inverse depth and 3D points. The total number of maps built during the stereo sequence is 11. The result of D&C without applying the loop closing constraint

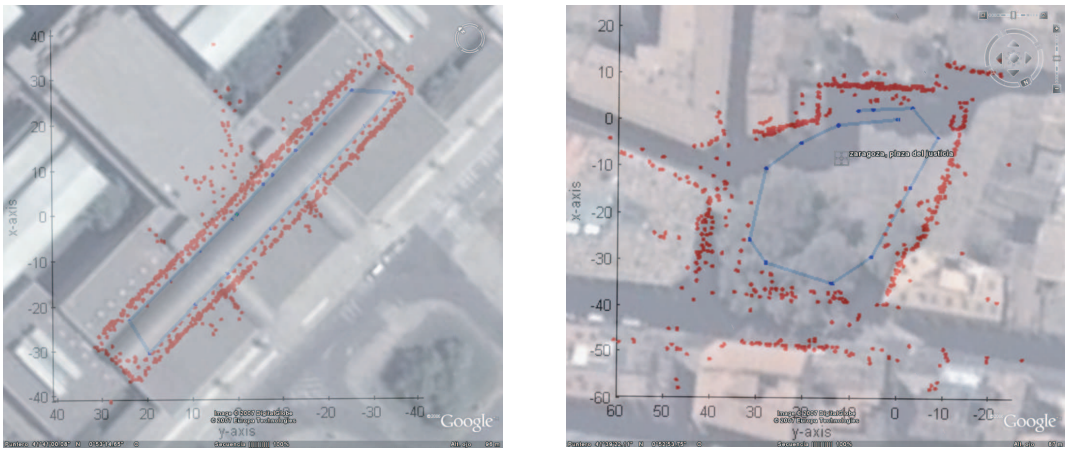


Figure 6.8: Stereo visual SLAM recovers the true scale: the building environment (top) and the Public square (bottom) overlapping Google Earth.

is shown in Fig. 6.5 (middle). As it can be observed, the precision of the map obtained is good enough to almost align the first and last submaps after all the trajectory has been traversed, even without applying loop closing constraints. Fig. 6.5 right presents the final result after closing the loop.

The second experiment was carried out inside one of our campus buildings in a walk of approximately 210 meters. The same process was run in order to obtain a full map from 8135 stereo pairs. This environment has a particular degree of difficulty due to ambiguous texture and the presence of extend zones of glass windows such as offices, corridors and cafeterias. This can be noticed in the long distance points estimated in some of the maps, which are actually inside offices and the cafeteria (fig.6.6, left). The result of CI D&C is shown in fig. 6.6, middle, and the final result after loop closing is shown in fig.6.6, right.

Our 6DOF SLAM system, even implemented in MATLAB, does not exceed 2 seconds per step, which is the worst case when building CI local maps. Fig. 6.7 shows how the running time system remains constant in most of the steps. Moreover, time peaks that appear when CI D&C takes place are below 8 seconds for the square experiment and 14 seconds for the indoor experiment, which are the maximum times required in the last step.

Using the Google Earth tool we can see that the map scale obtained and the trajectory followed by the camera is very close to the real scale. Fig. 6.8 illustrates comparative results. We loaded the MATLAB figure in Google Earth and set the scale parameter to the real scale. Given that we had no GPS nor compass measurements for the initial locations of the camera which are the base reference of each map, the position and orientation of the figure over the map were adjusted by hand. It can be noticed that angles between the square sides and the shape of the walls of the surrounding environment have been captured with precision.

6.6 Discussion

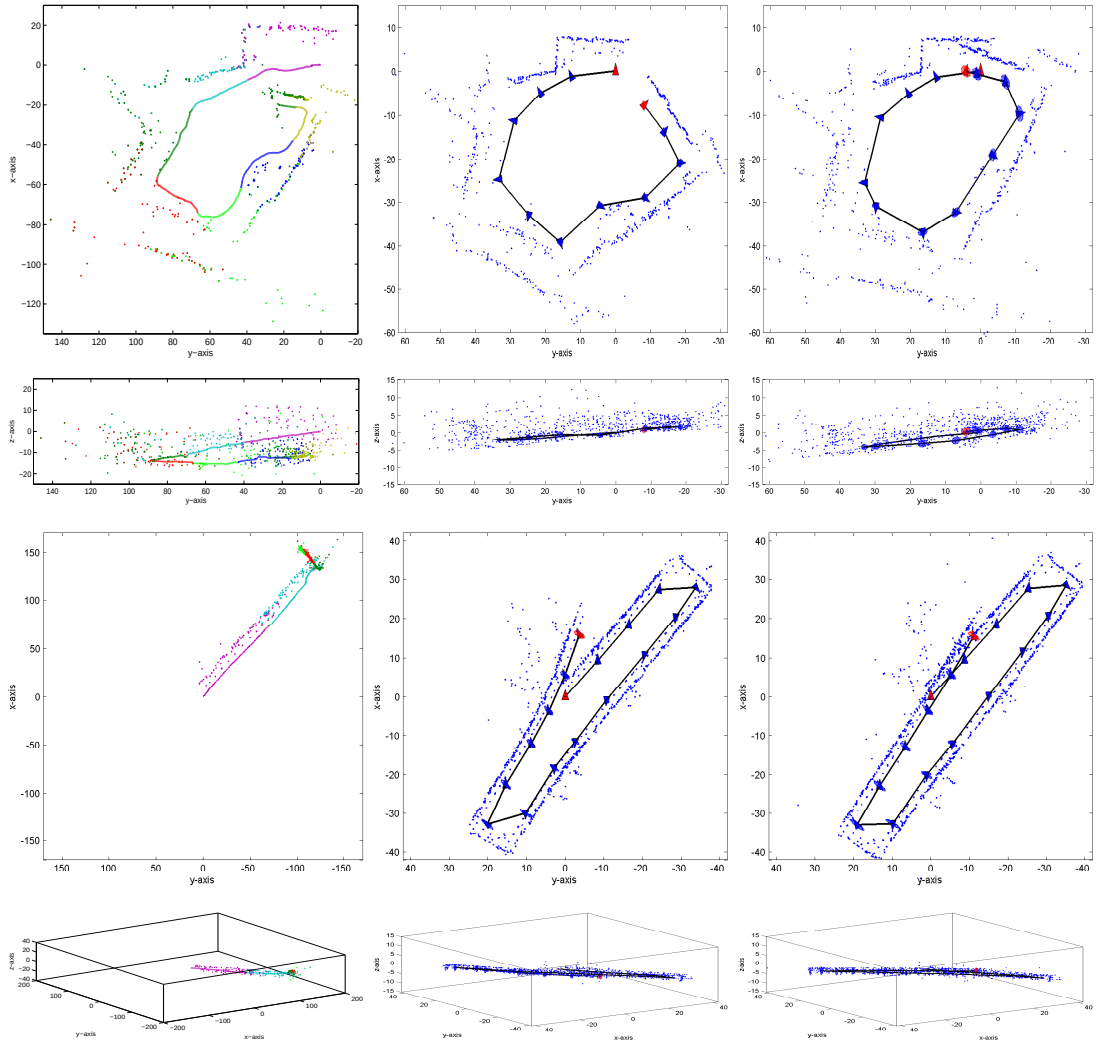


Figure 6.9: Comparison of the outdoor and indoor maps obtained before the loop closure using three different techniques: monocular SLAM with inverse depth points (left), stereo SLAM with 3D points (middle) and the proposed stereo SLAM with 3D points and inverse depth points (right)

Several works have demonstrated successful Visual SLAM systems in small environments using monocular or stereo cameras. There are several important factors that limit the extension of these results to large scale environments.

First, the computational complexity and consistency of the underlying SLAM technique. Here we have presented a novel algorithm that builds conditionally independent local maps in constant time and combines them in an optimal way in amortized linear time. Although the experiments presented here were processed in MATLAB, we expect that the extension to stereo of our current real-time implementation [Clemente 07] will be able to

build local maps up to 100 features in real time, with updates at 25Hz. The D&C map joining, loop detection and loop closing can be implemented on a separate thread, taking advantage of current multiple core processors.

In the case of monocular SLAM, another important limiting factor is the intrinsic unobservability of the scale. This problem can be addressed using additional sensors such as the vehicle odometry, GPS or inertial units. When they are not available, the scale can be initialized using some a priori knowledge about the environment such as the size of a known object visible at the start [Davison 07] or the initial speed of the camera [Pinies 07]. However, in large environments, unless scale information is injected on the system periodically, the scale of the map can slowly drift (see for example, the experiments in [Clemente 07]). Another critical issue appears when the scene is mostly planar and perpendicular to the optical axis. In this situation, with a monocular camera it is very difficult to distinguish between camera translation and rotation, unless a wide field of view (FOV) is used.

To illustrate these difficulties, we have processed our indoor and outdoor experiments using *only* the information from the right camera. As we are now using a bearing-only system, all the features are initialized using the inverse depth representation. To bootstrap the system, we have introduced an initial estimated speed for the camera of 1m/s. Apart from that, our Visual SLAM algorithm remains unchanged. The resulting maps are represented in the left column of figure 6.9. As it can be seen, the scale obtained by the system is arbitrary. Also in the outdoor experiment, at a certain point, the system misinterprets the camera translation as a rotation, and the map gets corrupted. Here we are using a camera with FOV of 65 degrees. The results obtained in the same environment with a FOV of 90 degrees are significantly more robust [Piniés 08]. In the indoor experiment with a monocular camera, as the objects are much closer to the camera, most of the features disappear fast from the FOV when the camera turns leading to a bad estimation of its position and consequently divergence in the map estimate.

We have also processed the sequences with our SLAM algorithm using conventional stereo, i.e. changed to initialize all the features whose disparity is bigger than one pixel as 3D points. Features without disparity are discarded because its depth cannot be computed by stereo. The immediate benefit is that the true environment scale is observable and the map corruption disappears (fig. 6.9, middle column). However, for points that are more than 10m away from the camera, a Gaussian in xyz is a bad approximation for its true uncertainty. This is the reason for the map deformation that is clearly visible in the lower part of the outdoor experiment, where many features are at about 20m from the camera.

The proposed visual SLAM system (fig. 6.9, right column) combines the advantages stereo and bearing only vision. On the one hand, the true scale is precisely obtained thanks to the 3D information obtained by the stereo camera from close point features. On the other hand, the region with useful point features extends up to infinity, thanks to the inverse depth representation developed for bearing-only SLAM. The depth of the features that are far from the camera can be precisely recovered by the system if they are seen form

viewpoints that are separated enough. In that case, they can be upgraded to 3D points for better efficiency as explained in [Civera 08]. Otherwise, they remain as inverse depth points and still provide very valuable orientation information that improves map precision and keeps the SLAM system stable when few close features are observed.

Chapter 7

Conclusions

In this thesis we concentrate our efforts to overcome some of the main EKF-SLAM limitations reported in recent years. First of all, we successfully reduce the complexity problem by proposing strategies for carrying out updates, data association and global localization processes in linear time. At the same time, we have shown a consistency improvement on the map estimate. As final result, we develop a fast visual 3D mapping system combining these methods together. This chapter summarizes the main conclusions of our work.

In chapter 2, we developed a computational analysis for EKF-SLAM using Local Mapping and Map Joining techniques. Monte Carlo simulated experiments show that building a global map via local maps and map joining also renders an improvement in linearization errors and thus improving consistency. In our simulated experiments, the maximum computational saving achieved with map joining coincides with the maximun of map consistency. Results provide the insight that other methods based on local maps, besides map joining, can profit from this result to reduce computational cost and overcome nonlinearlity problems.

In chapter 3, we show that EKF SLAM can be carried out in time *linear* with map size. We describe an EKF SLAM variant: *Divide and Conquer* SLAM, an algorithm that can be easily implemented. In contrast with many current efficient SLAM algorithms, all information required for data association is available when needed with no further processing. D&C SLAM computes the EKF SLAM solution, both the state *and* its covariance, with no approximations, and with the additional advantage of providing always a more precise and consistent vehicle and map estimate.

Consecutively, in chapter 4 we describe **RJC**, a data association algorithm that also executes in linear time per step when using spatial tessellation and RANSAC-based techniques. The advantage in the use of grid sampling to check individual compatibility in linear time makes this framework adaptable to the global localization problem in SLAM.

In chapter 5 we prove that the global localization problem in SLAM can be also solved in time linear with both the size of the map and the number of sensor measurements. The representation of the vehicle configuration space via grid sampling allows to use a

more efficient pairing-driven voting strategy than the location-driven voting strategy that Monte Carlo style algorithms use. Recent works [LaValle 04] suggest that, for motion planning, there is no advantage in the use of random sampling techniques.

In chapter 6 we show that D&C SLAM is the algorithm to use in all applications in which the Extended Kalman Filter solution is to be used. We have shown that 6DOF visual mapping of large environments can be efficiently and accurately carried out using a stereo camera as the only sensor. One of the contributions of this work is that information from features nearby and far from the cameras can be simultaneously incorporated to represent the 3D structure more precisely. Using close points provides scale information through the stereo baseline avoiding 'scale-drift', while inverse depth points are useful to obtain angular information from distant scene points.

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

Although we are able to close large indoor and outdoor loops, the algorithm used for loop closing strongly depends on detecting sets of features already stored in the map when the same area is revisited. Future work should concentrate in analyzing other types of loop closing algorithms, for instance the image to map algorithm proposed in [Williams 07b].

Also as future work, we will focus on comparing our system with other stereo vision techniques such as visual odometry. We are also interested in studying the fusion of the stereo camera with other sensors like GPS or inertial systems in order to compare the precision obtained. We will also consider other types of feature detectors, and their effect in the final result. We believe that the D&C map hierarchical splitting strategy can also be incorporated into other algorithms based on local submaps and similar strategies to improve the results.

We enumerate a the list of relevant publications covered in this thesis:

- Large Scale 6DOF SLAM with a stereo camera in hand L.M. Paz, P. Pinis, J.D. Tards and J. Neira *Accepted in Transactions on Robotics, 2008.*
- Divide and Conquer: EKF SLAM in $O(n)$ L.M. Paz, J.D. Tards and J. Neira *Accepted in Transactions on Robotics, 2008.*
- 6DOF SLAM with a stereo camera in hand L.M. Paz, P. Pinis, J.D. Tards and J. Neira 2007. *Visual SLAM Workshop. IEEE/RSJ Int.Conf.on Intelligent Robots and Systems, IROS 2007, San Diego, USA, Octubre, 2007.*

- Data association in O(n) for Divide and Conquer SLAM L.M. Paz, J. Guivant, J.D. Tards and J. Neira. *2007 Robotics: Science and Systems, RSS 2007, June 27-30, Atlanta, USA.*

- EKF SLAM updates in O(n) with Divide and Conquer SLAM L.M. Paz, P. Jensfelt, J.D. Tards, J. Neira. *2007 IEEE Intl. Conference on Robotics and Automation, ICRA 2007, Rome, Italy, April, 2007.*

- Optimal Local Map Size for EKF-based SLAM L.M. Paz, J. Neira. *2006 IEEE/RSJ Int.Conf.on Intelligent Robots and Systems, IROS 2006, Beijing, China, October, 2006.*

- Global Localization in SLAM in Bilinear Time L.M. Paz, P. Pines, J. Neira, J.D. Tards 2005. *IEEE/RSJ Int.Conf.on Intelligent Robots and Systems, IROS 2005, Edmonton, Canada, pp 655-661. August, 2005.*

Collaborations

Part of the research presented in this thesis have been carry out thanks to the grant BES-2004-5431 founded by the Dirección General de Investigación de España in the following international centers:

- Royal Institute of Technology (KTH), SE, 2006 (three months): during this stay the algorithm steps of *D&C* SLAM were carried out. Its early evaluation under simulations were performed in joint work with Patric Jensfelt.

- Australian Center for Field Robotics (ACFR), AU, 2006 (one month): fruitful discussions about *Randomized Joint Compatibility* to overcome Map Joining data association were held with José Guivant.

We also collaborate actively with Pedro Piniés. His research about *Conditionally Independent Local Maps* was applied to conceive the Large Scale 6DOF SLAM System.

Appendix A

EKF SLAM operations

This appendix provides a guide to perform the main EKF SLAM operations efficiently by using sparse matrices. The power of a sparse representation relays on its compact form which may speed up the computations when is used carefully. A matrix can be represented in sparse form when most of its entries are zero. This means that only those non-zero values are saved in memory. Real implementations for sparse matrices requires the use of special structures. The `CSpace` package developed by Timothy A. Davis [Davis 06] has been considered as the most complete tool to create and operate with matrices corresponding to both full and sparse form. The package consists of a set of functions to perform basic operations such as multiplications. As well, it provides an extension for those common packages that solve linear algebra systems. Although it allows for all functions to be called in C++ routines, the current SLAM implementation is completely run in MATLAB. Here, some tips are given as help find the most efficient way of solving a sparse matrix problem in MATLAB. However, it is convenient to review the full explanation provided in [Davis 06].

A.1 Efficient EKF Calculations

Notice that, even thought efficient calculations can be derived under the use of sparse matrices they can be avoided as well by computing just the parts which really correspond to changes.

- Consider the case, for instance, of the *EFK prediction* process (see eq. 1.1 in chapter 1), where given a map $\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1}$ at the step $k-1$ the predicted covariance $\mathbf{P}_{k|k-1}$ calculation involves the following matrices:

$$\mathbf{P}_{k-1|k-1} = \begin{pmatrix} \mathbf{P}_R & \mathbf{P}_{RF_1} & \dots & \mathbf{P}_{RF_n} \\ \mathbf{P}_{RF_1}^T & \mathbf{P}_{F_1} & \dots & \mathbf{P}_{F_1F_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{P}_{RF_n}^T & \mathbf{P}_{F_1F_n}^T & \dots & \mathbf{P}_{F_n} \end{pmatrix}$$

$$\mathbf{F}_k = \frac{\partial \mathbf{x}_{k|k-1}}{\partial \mathbf{x}_{k-1}} = \begin{pmatrix} \mathbf{J}_{1\oplus}\{\mathbf{x}_{R_{k-1}}, \mathbf{x}_{R_{k|k-1}}\} & 0 & \dots & 0 \\ 0 & \mathbf{I} & & \vdots \\ \vdots & & \ddots & \\ 0 & \dots & & \mathbf{I} \end{pmatrix}$$

$$\mathbf{G}_k = \frac{\partial \mathbf{x}_{k|k-1}}{\partial \mathbf{x}_{R_{k|k-1}}} = \begin{pmatrix} \mathbf{J}_{1\oplus}\{\mathbf{x}_{R_{k-1}}, \mathbf{x}_{R_{k|k-1}}\} \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

Here $\mathbf{x}_{R_{k-1}} = (x, y, \theta)^T$ is the current robot position estimate and $\mathbf{x}_{R_{k|k-1}}$ represents the motion model with associated zero-mean gaussian noise and error covariance Q_k. $\mathbf{J}_{1\oplus}$ and $\mathbf{J}_{2\oplus}$ are the Jacobians of the transformation composition which allow to propagate the uncertainty under linealizarion (see [Castellanos 99b]). A naive implementation using these matrices either in sparse form, (i.e. just non-zero elements are maintained in memory), or in full form (zero-elements are considered) could produce an overcost on the operations when these are carried out as it is illustrated in Fig. A.1, top. Only those parts of the resulting matrix related to the robot, i.e. the robot covariance $\mathbf{P}_R$ and its correlation with map features $\mathbf{P}_{RF_n}$, should be computed while the features covariance part remains unmodified (see Fig. A.1, bottom):

$$\mathbf{P}_{k|k-1} = \begin{pmatrix} \mathbf{J}_{1\oplus}\mathbf{P}_R\mathbf{J}_{1\oplus}^T + \mathbf{J}_{2\oplus}\mathbf{Q}_k\mathbf{J}_{2\oplus}^T & \mathbf{J}_{1\oplus}\mathbf{P}_{RF_1} & \dots & \mathbf{J}_{1\oplus}\mathbf{P}_{RF_n} \\ \mathbf{J}_{1\oplus}^T\mathbf{P}_{RF_1}^T & \mathbf{P}_{F_1} & \dots & \mathbf{P}_{F_1F_n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{J}_{1\oplus}^T\mathbf{P}_{RF_n}^T & \mathbf{P}_{F_1F_n}^T & \dots & \mathbf{P}_{F_n} \end{pmatrix} \quad \text{(A.1)}$$

$$= \begin{pmatrix} \mathbf{P}_{R_{k|k-1}} & \mathbf{J}_1\mathbf{P}_{RF_{k-1}} \\ \mathbf{J}_1^T\mathbf{P}_{RF_{k-1}}^T & \mathbf{P}_{F_{k-1}} \end{pmatrix} \quad \text{(A.2)}$$

- Consider the update step in the `ekf_slam` algorithm, which requieres the computation of the innovation covariance $\mathbf{S}_k$ at step k before performing a correction over the map estimate $\hat{\mathbf{x}}$ (see eq. 1.2). The matrices involved in this computation are the jacobian matrix $\mathbf{H}_k$ resulting after linearization, once the observations are predicted by means a measurement equation; it is followed by the covariance matrix $P_{k|k-1}$

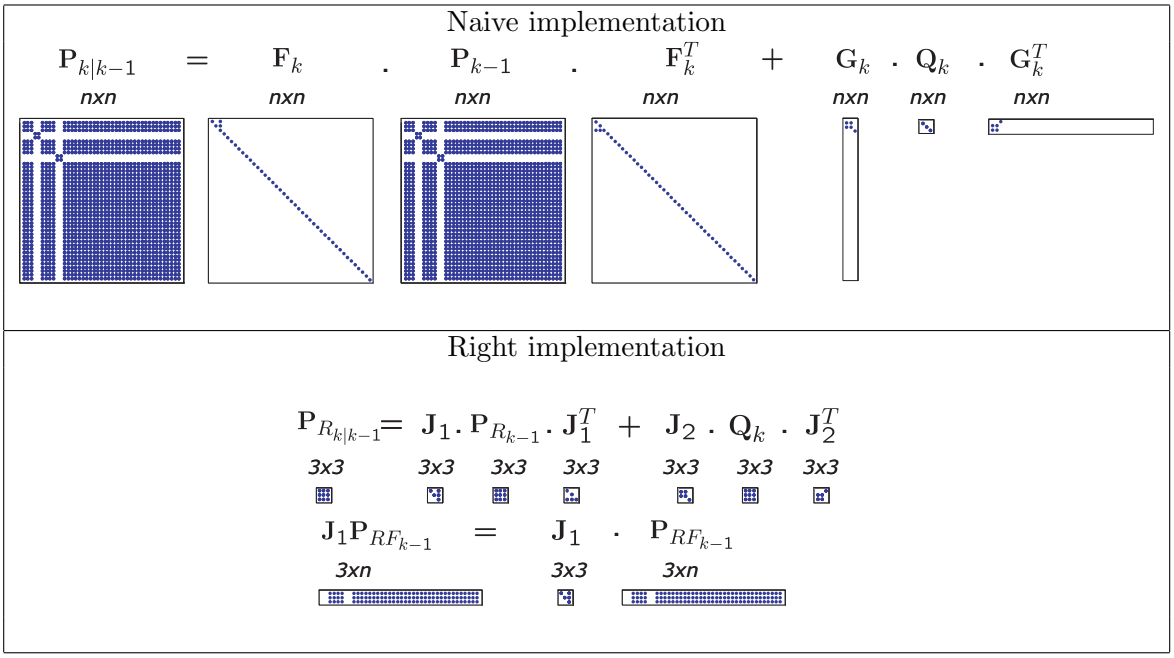


Figure A.1: Predicted covariance matrix calculation. A navive implementation could yield a cubic cost in the number of features in the map (top). To the prediction step, performing an efficient implementation results in an $O(n)$ cost.

resulting after prediction step and the observations covariance R that represents the sensor model.

The first operation that should be performed is the product $\mathbf{H}_k\mathbf{P}_{k|k-1}$, as it is shown in fig. A.2, top. For this propose, matrix $\mathbf{H}_k$ can be initialized as sparse before computing its entries. Thus, besides of the involved sums and multiplications costs, an indexing cost must be associated to the matrices product cost. Notice that to implement the product operation, MATLAB must to examine every column of $\mathbf{H}_k$, looking for row index i in each column. The MATLAB statement $r = \mathbf{H}_k(i,:)$ for a scalar i access a row of $\mathbf{H}_k$. This is costly compared with accessing a column due tot the internal representation of sparse matrices. Access to a column of this matrix $\mathbf{H}_k$ is simple, equivalent to $c = (:, j)$ in MATLAB, where j is a scalar. This short analysis suggest that transposing a sparse matrix and accessing its columns is better than repeatedly accessing its rows. Accordingly with this statement, it is advisable to shift for $(\mathbf{P}_{k|k-1}\mathbf{H}_k^T)^T$. This new operation produces the same result than $\mathbf{H}_k\mathbf{P}_{k|k-1}$ and requires only the creation of $\mathbf{H}_k^T$ instead of $\mathbf{H}_k$(see fig. A.2, bottom).

- Accordingly to eq. (1.5) in algorithm `ekf_slam`, chapter 1, the computation of the EKF gain matrix $\mathbf{K}_k$ can be carried out as $(\mathbf{P}_{k|k-1}\mathbf{H}_k^T)(\mathbf{S}_k)^{(-1)}$. In MATLAB, the matrix inversion is executed directly by the slash operator or matrix right division

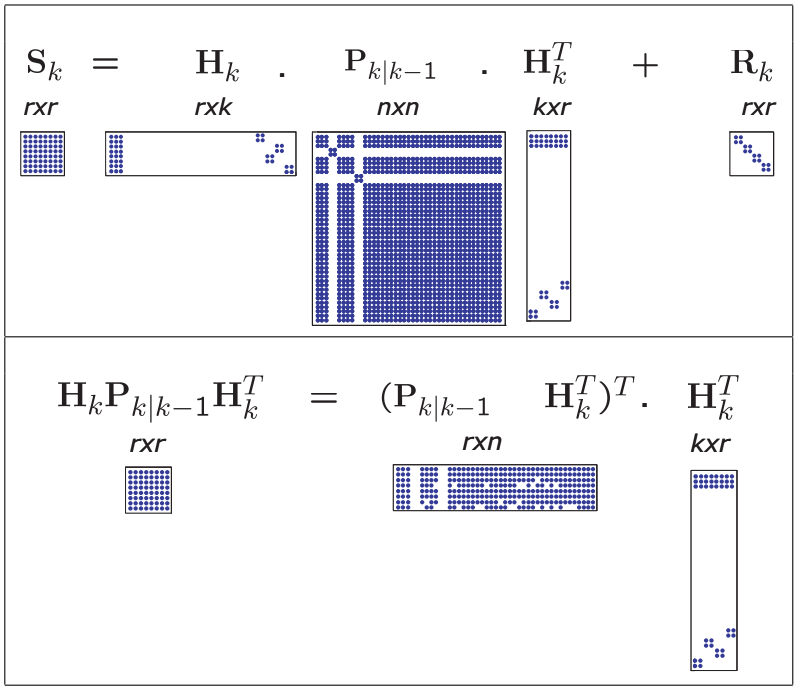


Figure A.2: Computation of S_k: involving access to rows of sparse H_k which is costly (top). There is a more efficient implementation that avoids pre-multiplying by a sparse matrix (bottom).

(see figure A.3). Two possible ways of solving for $\mathbf{K}_k$ are allowed; fig. A.3 shows both options. On the other hand, the computations can also be considered as solving for the linear system $(A^T \backslash B^T)^T$ by using Gaussian elimination which is roughly the same as B/A, where A is a $r \times r$ square matrix representing the innovation covariance $\mathbf{S}_k$, and B is a r columns matrix given by the product $\mathbf{P}_{k|k-1}\mathbf{H}_k^T$.

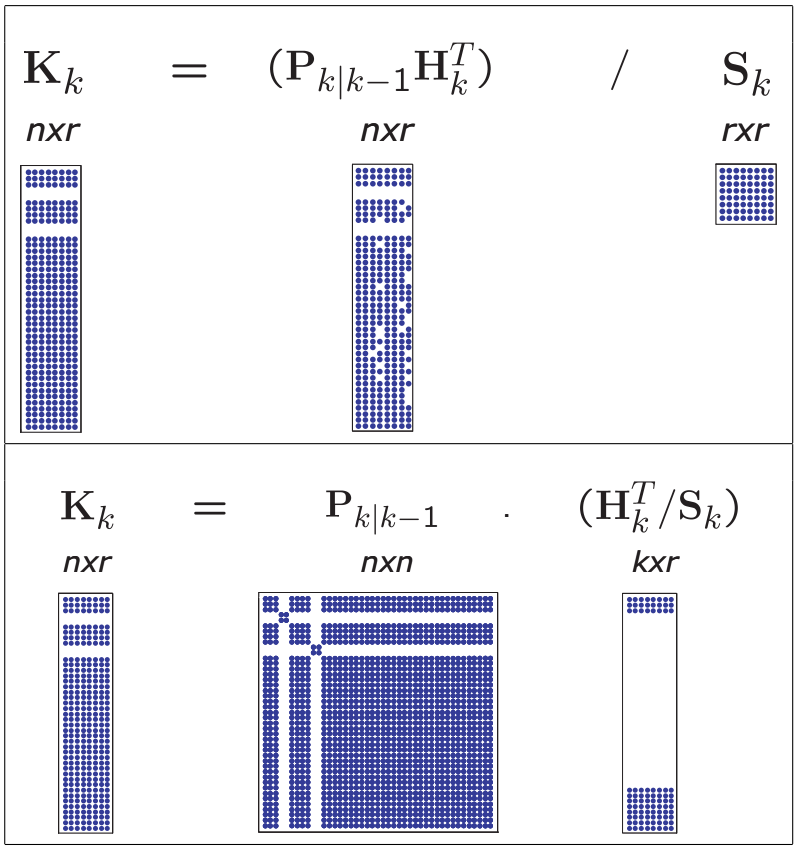


Figure A.3: The EKF Update Step: the gain matrix calculation.

Appendix B

Map Joining 2.0

This appendix describes the map joining process used in D&C SLAM, an improved version with respect to the original map joining 1.0 in [Tardós 02]. The general idea is the same: in a sequential move-sense-update cycle, a local map is initialized at some moment i using the current vehicle location R_i as base reference, and thus the initial vehicle location in the map is $\mathbf{x}_{R_iR_i} = 0$ an also the initial vehicle uncertainty $\mathbf{P}_{R_i} = 0$. Standard EKF SLAM is carried out in a this move-sense-update fashion, until the map reaches a certain size of n features $F_1 \dots F_n$ at step j. In this moment the state vector $\hat{\mathbf{x}}_{i\dots j}$ will be:

$$\hat{\mathbf{x}}_{i\dots j} = \begin{bmatrix} \hat{\mathbf{x}}_{R_iR_j} \\ \hat{\mathbf{x}}_{R_iF_1} \\ \vdots \\ \hat{\mathbf{x}}_{R_iF_n} \end{bmatrix}$$

with corresponding covariance matrix $\mathbf{P}_{i\dots j}$. This map is then closed, and a new local map $\mathbf{m}_{j\dots k} = (\hat{\mathbf{x}}_{j\dots k}, \mathbf{P}_{j\dots k})$ is initialized in the same way (for simplicity, assume the sensor measurements at step j are used to update the first map, and the vehicle motion from R_j to R_{j+1} is carried out in the second map). This results in having the last vehicle location in the first map, R_j, be the base reference of the second map, which allows maps to be joined into a full map in a three step process of (1) joining; (2) update; and (3) transformation, as it is explained next.

B.1 The Map Joining step

Consider two sequential local maps $\mathbf{m}_{i\dots j} = (\hat{\mathbf{x}}_{i\dots j}, \mathbf{P}_{i\dots j})$, $\mathbf{m}_{j\dots k} = (\hat{\mathbf{x}}_{j\dots k}, \mathbf{P}_{j\dots k})$, with n features $F_1 \dots F_n$ and m features $G_1 \dots G_m$ each:

$$\hat{\mathbf{x}}_{i\dots j} = \begin{bmatrix} \mathbf{x}_{R_iR_j} \\ \mathbf{x}_{R_iF_1} \\ \vdots \\ \mathbf{x}_{R_iF_n} \end{bmatrix}; \hat{\mathbf{x}}_{j\dots k} = \begin{bmatrix} \mathbf{x}_{R_jR_k} \\ \mathbf{x}_{R_jG_1} \\ \vdots \\ \mathbf{x}_{R_jG_m} \end{bmatrix} \tag{B.1}$$

In this approach, the *joining step* allows to obtain a stochastic map $\mathbf{m}^-_{i\ldots k} = \left(\hat{\mathbf{x}}^-_{i\ldots k}, \mathbf{P}^-_{i\ldots k}\right)$ in the following simple way:

$$\hat{\mathbf{x}}^-_{i\ldots k} = \begin{bmatrix} \hat{\mathbf{x}}_{i\ldots j} \\ \hat{\mathbf{x}}_{j\ldots k} \end{bmatrix} \tag{B.2}$$

$$\hat{\mathbf{P}}^-_{i\ldots k} = \begin{bmatrix} \mathbf{P}_{i..j} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{j..k} \end{bmatrix} \tag{B.3}$$

Note that the elements in the second map are kept in their own reference R_j instead of being referenced to reference frame R_i as in map joining 1.0. This has the effect of *delaying* the linearization process of converting all features to base reference R_i until the update step has taken place, and thus an improved estimation is used for this linearization. This is the fundamental difference between map joining 1.0 and map joining 2.0

B.2 The update step

Data association is carried out to determine correspondences between features coming from the first and second map. This allows to refine the vehicle and environment feature locations by the EKF update step on the state vector. Let $\mathcal{H}$ be a hypothesis that pairs r features $F_{f_1} \ldots F_{f_r}$ coming from local map $\mathbf{m}_{i\ldots j}$ with features $G_{g_1} \ldots G_{g_r}$ coming from map $\mathbf{m}_{j\ldots k}$. A modified ideal measurement equation for r re-observed features expresses this coincidence:

$$\mathbf{h}_{\mathcal{H}}(\hat{\mathbf{x}}^-_{i\ldots k}) = \begin{bmatrix} \mathbf{h}_{f_1,g_1} \\ \vdots \\ \mathbf{h}_{f_r,g_r} \end{bmatrix} = \mathbf{0} \tag{B.4}$$

where for each pairing:

$$\mathbf{h}_{f_r,g_r} = \mathbf{x}_{R_i F_{f_r}} - \mathbf{x}_{R_i R_j} \oplus \mathbf{x}_{R_j G_{g_r}}.$$

Linearization yields:

$$\mathbf{h}_{\mathcal{H}}(\hat{\mathbf{x}}^-_{i\ldots k}) \simeq \mathbf{h}_{\mathcal{H}}(\hat{\mathbf{x}}^-_{i\ldots k}) + \mathbf{H}_{\mathcal{H}}(\mathbf{x}^-_{i\ldots k} - \hat{\mathbf{x}}^-_{i\ldots k}) \tag{B.5}$$

where:

$$\begin{aligned} \mathbf{H}_{\mathcal{H}} &= \frac{\partial \mathbf{h}_{\mathcal{H}}}{\partial \mathbf{x}^-_{i\ldots k}}\Big|_{(\hat{\mathbf{x}}^-_{i\ldots k})} \\ &= \begin{bmatrix} \frac{\partial \mathbf{h}_{f_1 g_1}}{\partial \mathbf{x}_{R_i R_j}} & \mathbf{0} & \cdots & \mathbf{I} & \mathbf{0} & \frac{\partial \mathbf{h}_{f_1 g_1}}{\partial \mathbf{x}_{R_j G_{g_1}}} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{h}_{f_r g_r}}{\partial \mathbf{x}_{R_i R_j}} & \mathbf{0} & \mathbf{I} & \cdots & \mathbf{0} & \cdots & \frac{\partial \mathbf{h}_{f_r g_r}}{\partial \mathbf{x}_{R_j G_{g_r}}} \end{bmatrix} \end{aligned} \tag{B.6}$$

The update step allows to obtain a new estimate $\mathbf{m}^+_{i\ldots k} = \left(\hat{\mathbf{x}}^+_{i\ldots k}, \mathbf{P}^+_{i\ldots k}\right)$ by applying modified EKF update equations:

$$\begin{aligned} \hat{\mathbf{x}}^+_{i\ldots k} &= \hat{\mathbf{x}}^-_{i\ldots k} - \mathbf{K}\mathbf{h}_{\mathcal{H}}(\hat{\mathbf{x}}^-_{i\ldots k}) \\ \mathbf{P}^+_{i\ldots k} &= (\mathbf{I} - \mathbf{K}\mathbf{H}_{\mathcal{H}})\mathbf{P}^-_{i\ldots k} \end{aligned}$$

where:

$$\mathbf{K} = \mathbf{P}^-_{i\ldots k}\mathbf{H}^T_{\mathcal{H}}\left(\mathbf{H}_{\mathcal{H}}\mathbf{P}^-_{i\ldots k}\mathbf{H}^T_{\mathcal{H}}\right)^{-1}$$

B.3 The transformation step

A final step is carried out to transform all the elements of $\hat{\mathbf{x}}^+_{i\ldots k}$ to the same base reference R_i and obtain the final joined map $\mathbf{m}_{i\ldots k} = (\hat{\mathbf{x}}_{i\ldots k}, \mathbf{P}_{i\ldots k})$:

$$\begin{aligned} \hat{\mathbf{x}}_{i\ldots k} &= \begin{bmatrix} \hat{\mathbf{x}}_{R_iR_k} \\ \hat{\mathbf{x}}_{R_iF_1} \\ \vdots \\ \hat{\mathbf{x}}_{R_iF_n} \\ \hat{\mathbf{x}}_{R_iG_1} \\ \vdots \\ \hat{\mathbf{x}}_{R_iG_m} \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{x}}^+_{R_iR_j} \oplus \hat{\mathbf{x}}^+_{R_jR_k} \\ \hat{\mathbf{x}}^+_{R_iF_1} \\ \vdots \\ \hat{\mathbf{x}}^+_{R_iF_n} \\ \hat{\mathbf{x}}^+_{R_iR_j} \oplus \hat{\mathbf{x}}^+_{R_jG_1} \\ \vdots \\ \hat{\mathbf{x}}^+_{R_iR_j} \oplus \hat{\mathbf{x}}^+_{R_jG_m} \end{bmatrix} \\ \mathbf{P}_{i\ldots k} &= \frac{\partial \hat{\mathbf{x}}_{i\ldots k}}{\partial \hat{\mathbf{x}}^+_{i\ldots k}} \mathbf{P}^+_{i\ldots k} \left(\frac{\partial \hat{\mathbf{x}}_{i\ldots k}}{\partial \hat{\mathbf{x}}^+_{i\ldots k}}\right)^T \\ \frac{\partial \hat{\mathbf{x}}_{i\ldots k}}{\partial \hat{\mathbf{x}}^+_{i\ldots k}} &= \begin{bmatrix} \frac{\partial \mathbf{x}_{R_iR_k}}{\partial \mathbf{x}_{R_iR_j}} & \mathbf{0} & \frac{\partial \mathbf{x}_{R_iR_k}}{\partial \mathbf{x}_{R_jR_k}} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ \frac{\partial \mathbf{x}_{R_iE}}{\partial \mathbf{x}_{R_iR_j}} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{x}_{R_iE}}{\partial \mathbf{x}_{R_jE}} \end{bmatrix} \end{aligned} \tag{B.7}$$

Note again that this linearization is carried out once the map has been refined in the previous update step, thus using a better estimate.

Appendix C

Derived Constants for the Optimal Local Map Size

Constants for Local mapping

$$
\begin{aligned}
c_1 &= r(m-r)^2 \\
c_2 &= (m-r)\left[(r+1)^2 + (m-r) - 2r(m-2r)\right] \\
c_3 &= \left[(r+1)^2(2r-m) + r(m-r) + r(2r-m)^2\right] \\
k_1 &= \frac{c_1}{3(m-r)^3} \\
k_2 &= \left[\frac{c_1(m^2+3m-9r)}{6(m-r)} + \frac{c_2}{2}\right]\frac{1}{(m-r)^2} \\
k_3 &= \left[\frac{c_1(5r^2-8rm)}{6(m-r)} + \frac{c_2(m-3r)}{2}\right]\frac{1}{(m-r)^2} \\
k_4 &= \left[\frac{c_1(rm(r-m)+6r^3)}{6(m-r)} + \frac{c_2(2r^2-rm)}{2}\right]\frac{1}{(m-r)^2}
\end{aligned}
$$

Constants for Map joining

$$
\begin{aligned}
k_5 &= \frac{c_1(P-r)}{3(m-r)^2} \\
k_6 &= \left[\frac{c_1(3m-7r)}{6(m-1)} + \frac{c_2}{2}\right]\frac{(P-r)}{(m-r)^2} \\
k_7 &= \left[\frac{c_1(6r^2-6mr+m^2)}{6(m-r)} + \frac{c_2(m-2r)}{2}\right]\frac{(P-r)}{(m-r)^2} \\
k_8 &= c_3(P-r)
\end{aligned}
$$

$$
\begin{aligned}
k_{\ 9} &= -(2r+1) \\
k_{10} &= (1/2)(2Pr+P-r) \\
k_{11} &= (1/2)(2P^2r-3Pr^2+P^2-Pr+15r^3+ \\
&\quad +14r^2+2r) \\
k_{12} &= (1/2)(2P^2r-10Pr^2+3P^2r^2-12Pr^3- \\
&\quad -Pr-9r^4-12r^3-3r^2) \\
k_{13} &= (-1/2)(4P^2r^2-12Pr^3+3P^2r^3- \\
&\quad -9Pr^4+P^2r-3Pr^2) \\
k_{14} &= (-r) \\
k_{15} &= (1/6)(Pr+17r^2) \\
k_{16} &= (1/6)(3P^2r-8Pr^2-13r^3) \\
k_{17} &= (1/6)(2P^3r-9P^2r^2+13Pr^3) \\
k_{18} &= r \\
k_{19} &= 3r^2+3r \\
k_{20} &= 2r^3+2r^2
\end{aligned}
$$

Appendix D

Measurement equations

Here we describe the main operations involved when predicting the map features in order to carry out an active search and a state vector updating when using a 6DOF SLAM system, combining depth points and inverse depth points. The system uses images gathered with a stereo camera which moves following a constant velocity model with zero mean Gaussian noise in the linear and angular accelerations. We describe the state vector representation and derive the measurement equation, and the Jacobians necessary to predict and project the map features uncertainties on both right and left images. Consecutively, we compute the innovation covariance that allows us to correct the estimate.

D.1 State Representation

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

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

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

$$\mathbf{x}_c = \begin{bmatrix} \mathbf{r} \\ \Psi \\ \mathbf{v} \\ \mathbf{w} \end{bmatrix} \tag{D.1}$$

Image corners classified as depth points are transformed to 3D points, given the disparity information provided by the stereo pair. Since the stereo camera provides rectified images, the back-projection equations to obtain a 3D point are based on a pinhole camera model which relates image points and 3D points using the following transformation function:

$$\begin{aligned} \mathbf{x}_{3D} &= f(u_r, v_r, u_l, v_l) \\ &= [x, y, z]^T \\ &= \left[\frac{b(u_r - u_0)}{d}, \frac{b(v_r - v_0)}{d}, \frac{fb}{d}\right]^T \end{aligned} \tag{D.2}$$

where (u_r, v_r) and (u_l, v_l) are the pixels on the right and left images, and $d = (u_l - u_r)$ is the horizontal disparity. The remainder terms in the equation are the calibrated parameters of the camera, i.e., the central pixel of the image (u_0, v_0), the baseline b and the focal length f.
Given the camera location $\mathbf{x}_{c_i}$, an inverse depth point is defined as in [Montiel 06]:

$$\mathbf{x}_{ID} = \begin{bmatrix} \mathbf{r}_i \\ \theta_i \\ \phi_i \\ \rho_i \end{bmatrix} \tag{D.3}$$

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

D.2 Measurement Equation

At each step, we filter features using only those that fall in the FOV of the camera when they are projected along with their uncertainties on both right and left images. Using the patch associated with each feature, a matching search based on normalized cross-correlation is performed inside the projected uncertainty region [1]. When a match is found, a new observation $\mathbf{z}$ given by the matched pixel is used to compute the innovation and then update the state of the camera and the map.
In the right camera, the measurement equation that defines the relation between the *ith* inverse depth feature $\mathbf{x}^i_{ID}$ and its observation $\mathbf{z}^{r_i}_{ID}$ is given by:

$$\begin{aligned} \mathbf{z}^{r_i}_{ID} &= h^r_{ID}(\mathbf{x}_c, \mathbf{x}^i_{ID}) + \upsilon \\ &= projection(\ominus \mathbf{x}_c \oplus \mathbf{x}^i_{ID}) + \upsilon \end{aligned} \tag{D.4}$$

where h^r_{ID} is the function that projects the inverse depth feature to the right camera and υ is a zero mean gaussian noise with σ_p standard deviation that represents the projection

[1] This is the outline of an active search as introduced in [Davison 02]

error in pixels. Alternatively, we can define the measurement equation that relates the inverse point observation on the left image by:

$$\begin{aligned} \mathbf{z}^{l_i}_{ID} &= h^l_{ID}(\mathbf{x}_c, \mathbf{x}^i_{ID}) + \upsilon \\ &= projection(\ominus \mathbf{x}_c \oplus \mathbf{x}_{c_r c_l} \oplus \mathbf{x}^i_{ID}) + \upsilon \end{aligned} \tag{D.5}$$

where the displacement of the left camera optical center with respect to the right camera is given by the rigid transformation $\mathbf{x}_{c_r c_l} = [0\ b\ 0]^T$.
In a similar way, we describe observations corresponding to 3D map features in the right and left cameras:

$$\begin{aligned} \mathbf{z}^{r_i}_{3D} &= h^r_{3D}(\mathbf{x}_c, \mathbf{x}^i_{3D}) + \upsilon \\ &= projection(\ominus \mathbf{x}_c \oplus \mathbf{x}^i_{3D}) + \upsilon \end{aligned} \tag{D.6}$$

$$\begin{aligned} \mathbf{z}^{l_i}_{3D} &= h^l_{3D}(\mathbf{x}_c, \mathbf{x}^i_{3D}) \\ &= projection(\ominus \mathbf{x}_c \oplus \mathbf{x}_{c_r c_l} \oplus \mathbf{x}^i_{3D}) + \upsilon \end{aligned} \tag{D.7}$$

Note that we make use of $\oplus$ and $\ominus$ operators in order to carry out transformations by means of compositions and inversions operations [Smith 86]. Nonetheless, the operations differ from the original description in [Smith 86] due to the kind of parametrization used to express a feature. How to develop those operations for inverse depth and depth points and how to compute the corresponding Jacobians to propagate the error uncertainties is detailed in next section.

D.2.1 $\oplus$ and $\ominus$ operators for the Inverse Depth points measurement equation

We first develop all operations involved to calculate the innovation covariance when a inverse point feature is projected on right image. The predicted inverse point h^r_{ID} is computed in Eq. D.4 by transforming the inverse feature to the current camera reference. For such a propose, the inner composition is divided by applying a translation and then a rotation to the inverse depth ray,

$$\ominus \mathbf{x}_c \oplus \mathbf{x}^i_{ID} = \mathbf{h}^r_{c_{rot}}(\mathbf{h}^r_{c_{transl}}(\mathbf{x}_c, \mathbf{x}^i_{ID})) \tag{D.8}$$

$$\mathbf{h}^r_{c_{transl}} = (\mathbf{r}_i - \mathbf{r})\,\rho_i + \mathbf{m} \tag{D.9}$$

$$\mathbf{h}^r_{c_{rot}} = Rot(\Psi)^T \cdot \mathbf{h}^r_{c_{transl}} \tag{D.10}$$

Here $Rot(\Psi)$ is a 3×3 rotation matrix corresponding to the camera orientation. Both translation $\mathbf{h}_{c_{transl}}$ and rotation $\mathbf{h}_{c_{rot}}$ are represented from the point of view of vector operations. These equations transform the inverse-depth feature vector $\mathbf{x}_{ID}$ expressed in the base map reference B to a vector in the current camera reference $\mathbf{x}_c$ being $\mathbf{m}$ the unitary ray directional vector given by:

$$\mathbf{m}(\theta_i, \phi_i) = \begin{bmatrix} \cos(\theta_i)\cos(\phi_i) \\ \sin(\theta_i)\cos(\phi_i) \\ \sin(\phi_i) \end{bmatrix} \tag{D.11}$$

After the transformation is done, the ray is projected on the image as follows:

$$\begin{aligned} \mathbf{h}^r_{ID} &= \mathbf{M} \begin{pmatrix} \mathbf{h}^r_{c_{rot2}}/\mathbf{h}^r_{c_{rot1}} \\ \mathbf{h}^r_{c_{rot3}}/\mathbf{h}^r_{c_{rot1}} \\ 1 \end{pmatrix} \\ &= \begin{pmatrix} u_0 - f\mathbf{h}^r_{c_{rot2}}/\mathbf{h}^r_{c_{rot1}} \\ v_0 - f\mathbf{h}^r_{c_{rot3}}/\mathbf{h}^r_{c_{rot1}} \\ 1 \end{pmatrix} \end{aligned} \tag{D.12}$$

where $\mathbf{M}$ is the projection matrix defined as:

$$\mathbf{M} = \begin{pmatrix} -f & 0 & u_0 \\ 0 & -f & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{D.13}$$

D.2.2 Linearization of the measurement equation for Inverse Depth points

We now detail the Jacobian required to propagate the *ith* inverse depth point uncertainty to the image under linearization. Therefore, the measurement equation is derived respect to the state vector $\mathbf{x}_B$

$$\mathbf{H}_{ID^r_i} = \left[\frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{x}_c}, \mathbf{0}, \dots \mathbf{0}, \frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{x}^i_{ID}}, \mathbf{0} \right] \tag{D.14}$$

The function in Eq. D.4 is derived with respect to both the camera position $\mathbf{x}_c$ and the inverse point feature $\mathbf{x}^i_{ID}$. We use the chain rule to solve for each term going backwards from Eq. D.12:

$$\begin{aligned} \frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{x}_c} &= \frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{h}^r_{c_{rot}}} \cdot \frac{\partial \mathbf{h}^r_{c_{rot}}}{\partial \mathbf{x}_c} + \frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{h}^r_{c_{rot}}} \cdot \frac{\partial \mathbf{h}^r_{c_{rot}}}{\partial \mathbf{h}^r_{c_{transl}}} \cdot \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{x}_c} \\ &= \frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{h}^r_{c_{rot}}} \cdot \left(\frac{\partial \mathbf{h}^r_{c_{rot}}}{\partial \mathbf{x}_c} + \frac{\partial \mathbf{h}^r_{c_{rot}}}{\partial \mathbf{h}^r_{c_{transl}}} \cdot \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{x}_c} \right) \end{aligned} \tag{D.15}$$

$$\frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{x}^i_{ID}} = \frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{h}^r_{c_{rot}}} \cdot \frac{\mathbf{h}^r_{c_{rot}}}{\partial \mathbf{h}^r_{c_{transl}}} \cdot \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{x}^i_{ID}} \tag{D.16}$$

We provide each value to the reader so that all partial derivatives can be substituted:

$$\begin{aligned}\frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{h}^r_{c_{rot}}} &= \left[\frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{h}^r_{c_{rot1}}}, \frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{h}^r_{c_{rot2}}}, \frac{\partial \mathbf{h}^r_{ID}}{\partial \mathbf{h}^r_{c_{rot3}}}\right] \\ &= \begin{bmatrix} f\mathbf{h}^r_{c_{rot2}}/(\mathbf{h}^r_{c_{rot1}})^2 & -f/\mathbf{h}^r_{c_{rot1}} & 0 \\ f\mathbf{h}^r_{c_{rot3}}/(\mathbf{h}^r_{c_{rot1}})^2 & 0 & -f/\mathbf{h}^r_{c_{rot1}} \end{bmatrix}\end{aligned} \tag{D.17}$$

$$\frac{\partial \mathbf{h}^r_{c_{rot}}}{\partial \mathbf{x}_c} = \left[\mathbf{0}_{3x3}, \frac{\partial Rot^T}{\partial \Psi_1}\cdot \mathbf{h}^r_{c_{transl}}, \frac{\partial Rot^T}{\partial \Psi_2}\cdot \mathbf{h}^r_{c_{transl}}, \frac{\partial Rot^T}{\partial \Psi_3}\cdot \mathbf{h}^r_{c_{transl}}\right] \tag{D.18}$$

$$\frac{\partial \mathbf{h}_{c_{rot}}}{\partial \mathbf{h}_{c_{transl}}} = Rot(\psi)^T \tag{D.19}$$

$$\begin{aligned}\frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{x}_c} &= \left[\frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{r}}, \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \Psi}\right] \\ &= [-\mathbf{I}_{3x3}\cdot\rho_i, \mathbf{0}_{3x3}]\end{aligned} \tag{D.20}$$

$$\frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{x}^i_{ID}} = \left[\frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{r}_i}, \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \theta_i}, \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \phi_i}, \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \rho_i}\right] \tag{D.21}$$

$$\begin{aligned}\frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{r}_i} &= \mathbf{I}_{3x3}\cdot\rho_i \\ \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \theta_i} &= \begin{pmatrix} -\sin\phi_i\cdot\cos\theta_i \\ \cos\phi_i\cdot\cos\theta_i \\ 0 \end{pmatrix} \\ \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \phi_i} &= \begin{pmatrix} -\cos\phi_i\cdot\sin\theta_i \\ \sin\phi_i\cdot\sin\theta_i \\ \cos\theta_i \end{pmatrix} \\ \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \rho_i} &= (\mathbf{r}_i - \mathbf{r})\end{aligned}$$

Once the Jacobians have been derived from the measurement equation of the inverse depth features on the right image, the extension to the left image is immediate.

$$\begin{aligned}\ominus(\mathbf{x}_c \oplus \mathbf{x}_{c_r c_l}) \oplus \mathbf{x}^i_{ID} &= \ominus\mathbf{x}_c(\ominus\mathbf{x}_{c_r c_l} \oplus \mathbf{x}^i_{ID}) \\ &= \mathbf{h}^l_{c_{transl}}(\mathbf{h}^r_{c_{rot}}(\mathbf{h}^r_{c_{transl}}(\mathbf{x}_c, \mathbf{x}^i_{ID})))\end{aligned} \tag{D.22}$$

$$\mathbf{h}^l_{c_{transl}} = \mathbf{h}^r_{c_{rot}} - \mathbf{x}_{c_r c_l}\cdot\rho_i \tag{D.23}$$

Note that we can describe the prediction on the left image $\mathbf{h}^l_{c_{transl}}$ with respect to the prediction on the right image already given by $\mathbf{h}^r_{c_{transl}}$ and $\mathbf{h}^l_{c_{rot}}$ in Eq. D.8. Thus,

the derivatives are very similar and few changes are produced due to the transformation between both cameras $\mathbf{x}_{c_r c_l}$:

$$\mathbf{H}_{ID_i^l} = \left[\frac{\partial \mathbf{h}_{ID}^l}{\partial \mathbf{x}_c}, \mathbf{0}, \ldots \mathbf{0}, \frac{\partial \mathbf{h}_{ID}^l}{\partial \mathbf{x}_{ID}^i}, \mathbf{0}\right] \tag{D.24}$$

$$\frac{\partial \mathbf{h}_{ID}^l}{\partial \mathbf{x}_c} = \frac{\partial \mathbf{h}_{ID}^l}{\partial \mathbf{h}_{c_{transl}}^l} \cdot \frac{\partial \mathbf{h}_{c_{transl}}^l}{\partial \mathbf{h}_{c_{rot}}^r} \cdot \left(\frac{\partial \mathbf{h}_{c_{rot}}^r}{\partial \mathbf{x}_c} + \frac{\partial \mathbf{h}_{c_{rot}}^r}{\partial \mathbf{h}_{c_{transl}}^r} \cdot \frac{\partial \mathbf{h}_{c_{transl}}^r}{\partial \mathbf{x}_c}\right) \tag{D.25}$$

$$\frac{\partial \mathbf{h}_{ID}^l}{\partial \mathbf{x}_{ID}^i} = \frac{\partial \mathbf{h}_{ID}^l}{\partial \mathbf{h}_{c_{transl}}^l} \cdot \left(\frac{\mathbf{h}_{c_{transl}}^l}{\partial \mathbf{h}_{c_{rot}}^r} \cdot \frac{\mathbf{h}_{c_{rot}}^r}{\partial \mathbf{h}_{c_{transl}}^r} \cdot \frac{\partial \mathbf{h}_{c_{transl}}^r}{\partial \mathbf{x}_{ID}^i} + \frac{\partial \mathbf{h}_{c_{transl}}^l}{\partial \mathbf{x}_{ID}^i}\right) \tag{D.26}$$

$$\frac{\partial \mathbf{h}_{c_{transl}}^l}{\partial \mathbf{h}_{c_{rot}}^r} = \mathbf{I}_{3\times 3} \tag{D.27}$$

$$\frac{\partial \mathbf{h}_{c_{transl}}^r}{\partial \mathbf{x}_{ID}^i} = \left[\frac{\partial \mathbf{h}_{c_{transl}}^l}{\partial \mathbf{r}_i}, \frac{\partial \mathbf{h}_{c_{transl}}^l}{\partial \theta_i}, \frac{\partial \mathbf{h}_{c_{transl}}^l}{\partial \phi_i}, \frac{\partial \mathbf{h}_{c_{transl}}^l}{\partial \rho_i}\right] \tag{D.28}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -b \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

D.2.3 $\oplus$ and $\ominus$ for the Depth points measurement equation

The measurement equation D.6 for depth points is also linearized by carrying out the same process as it was shown above. We define first the composition that transforms the depth point from the base reference to the camera reference.

$$\ominus \mathbf{x}_c \oplus \mathbf{x}_{3D}^i = \mathbf{h}_{c_{rot}}^r(\mathbf{h}_{c_{transl}}^r(\mathbf{x}_c, \mathbf{x}_{3D}^i)) \tag{D.29}$$

$$\mathbf{h}_{c_{transl}}^r = \mathbf{x}_{3D} - \mathbf{r} \tag{D.30}$$

$$\mathbf{h}_{c_{rot}}^r = Rot(\Psi)^T \cdot \mathbf{h}_{c_{transl}}^r \tag{D.31}$$

Again, the projection of the predicted 3D feature on the right image $\mathbf{h}_{3D}^r$ is obtained by means of the projection matrix M.

$$\mathbf{h}_{3D}^r = \mathbf{M} \begin{pmatrix} \mathbf{h}_{c_{rot2}}^r / \mathbf{h}_{c_{rot1}}^r \\ \mathbf{h}_{c_{rot3}}^r / \mathbf{h}_{c_{rot1}}^r \\ 1 \end{pmatrix} = \begin{pmatrix} u_0 - f\mathbf{h}_{c_{rot2}}^r / \mathbf{h}_{c_{rot1}}^r \\ v_0 - f\mathbf{h}_{c_{rot3}}^r / \mathbf{h}_{c_{rot1}}^r \\ 1 \end{pmatrix} \tag{D.32}$$

D.2.4 Linearization of the measurement equation for Depth points

We summarized all derivatives which form the Jacobian matrix for the ith 3D predicted feature:

$$\mathbf{H}_{3D_i^r} = \left[\frac{\partial h^r_{3D}}{\partial \mathbf{x}_c}, \mathbf{0}, \dots \mathbf{0}, \frac{\partial h^r_{3D}}{\mathbf{x}^i_{3D}}, \mathbf{0}\right] \tag{D.33}$$

$$\frac{\partial h^r_{3D}}{\partial \mathbf{x}_c} = \frac{\partial h^r_{3D}}{\partial \mathbf{h}^r_{c_{rot}}} \cdot \left(\frac{\partial \mathbf{h}^r_{c_{rot}}}{\partial \mathbf{x}_c} + \frac{\partial \mathbf{h}^r_{c_{rot}}}{\partial \mathbf{h}^r_{c_{transl}}} \cdot \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{x}_c}\right) \tag{D.34}$$

$$\frac{\partial h^r_{3D}}{\mathbf{x}^i_{3D}} = \frac{\partial h^r_{3D}}{\partial \mathbf{h}^r_{c_{rot}}} \cdot \frac{\partial \mathbf{h}^r_{c_{rot}}}{\partial \mathbf{h}^r_{c_{transl}}} \cdot \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{x}^i_{3D}} \tag{D.35}$$

$$\begin{aligned}\frac{\partial \mathbf{h}^r_{3D}}{\partial \mathbf{h}^r_{c_{rot}}} &= \left[\frac{\partial \mathbf{h}^r_{3D}}{\partial \mathbf{h}^r_{c_{rot1}}}, \frac{\partial \mathbf{h}^r_{3D}}{\partial \mathbf{h}^r_{c_{rot2}}}, \frac{\partial \mathbf{h}^r_{3D}}{\partial \mathbf{h}^r_{c_{rot3}}}\right] \\ &= \begin{bmatrix} f\mathbf{h}^r_{c_{rot2}}/(\mathbf{h}^r_{c_{rot1}})^2 & -f/\mathbf{h}^r_{c_{rot1}} & 0 \\ f\mathbf{h}^r_{c_{rot3}}/(\mathbf{h}^r_{c_{rot1}})^2 & 0 & -f/\mathbf{h}^r_{c_{rot1}} \end{bmatrix}\end{aligned} \tag{D.36}$$

$$\frac{\partial \mathbf{h}^r_{c_{rot}}}{\partial \mathbf{x}_c} = \left[\mathbf{0}_{3x3}, \frac{\partial Rot^T}{\partial \Psi_1} \cdot \mathbf{h}^r_{c_{transl}}, \frac{\partial Rot^T}{\partial \Psi_2} \cdot \mathbf{h}^r_{c_{transl}}, \frac{\partial Rot^T}{\partial \Psi_3} \cdot \mathbf{h}^r_{c_{transl}}\right] \tag{D.37}$$

$$\begin{aligned}\frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{x}_c} &= \left[\frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{r}}, \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \Psi}\right] \\ &= [-\mathbf{I}_{3x3}, \mathbf{0}_{3x3}]\end{aligned} \tag{D.38}$$

$$\frac{\partial \mathbf{h}^r_{c_{rot}}}{\partial \mathbf{h}^r_{c_{transl}}} = Rot(\psi)^T \tag{D.39}$$

$$\begin{aligned}\frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial \mathbf{x}^i_{3D}} &= \left[\frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial x}, \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial y}, \frac{\partial \mathbf{h}^r_{c_{transl}}}{\partial z}\right] \\ &= \mathbf{I}_{3\times 3}\end{aligned} \tag{D.40}$$

All this process can be extended to predict and project the 3D feature on the left image. As for inverse depth features, it presents some differences due to the relation between both cameras. We use the right prediction to describe the composition, so that:

$$\ominus(\mathbf{x}_c \oplus \mathbf{x}_{c_r c_l}) \oplus \mathbf{x}^i_{3D} = \ominus\mathbf{x}_c(\ominus\mathbf{x}_{c_r c_l} \oplus \mathbf{x}^i_{3D}) \tag{D.41}$$

$$= \mathbf{h}^l_{c_{transl}}(\mathbf{h}^r_{c_{rot}}(\mathbf{h}^r_{c_{transl}}(\mathbf{x}_c, \mathbf{x}^i_{3D}))) \tag{D.42}$$

$$\mathbf{h}^r_{c_{transl}} = \mathbf{h}^r_{c_{rot}} - \mathbf{x}_{c_r c_l} \tag{D.43}$$

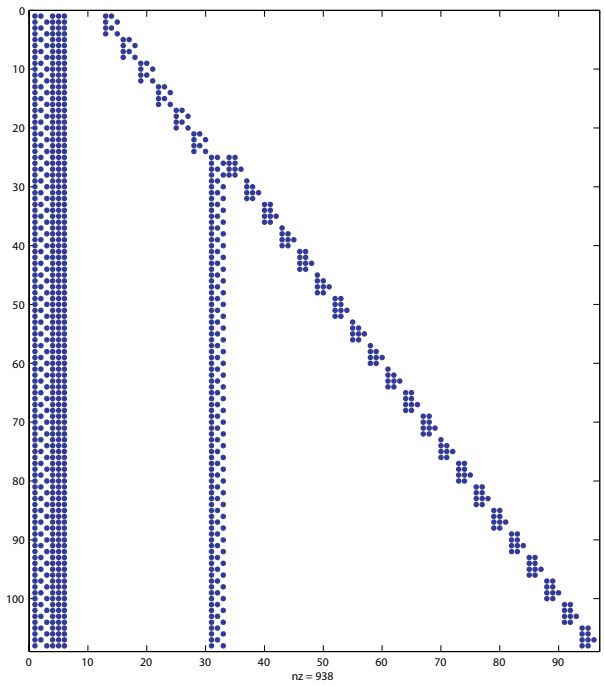


Figure D.1: Jacobian Matrix to linearize the measurement equations. The first six features corresponds to 3D predicted features while the next features represents the prediction of inverse depth features.

The final Jacobian matrix is formed by the Jacobians of the depth features projections $\mathbf{H}_{3D_i} = [\mathbf{H}_{3D_i^r}^T \ \mathbf{H}_{3D_i^l}^T]^T$ and inverse depth features projections $\mathbf{H}_{ID_i} = [\mathbf{H}_{ID_i^r}^T \ \mathbf{H}_{ID_i^l}^T]^T$ as follows:

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{3D_1} \\ \vdots \\ \mathbf{H}_{3D_i} \\ \vdots \\ \mathbf{H}_{3D_m} \\ \mathbf{H}_{ID_1} \\ \vdots \\ \mathbf{H}_{ID_i} \\ \vdots \\ \mathbf{H}_{ID_s} \end{pmatrix} \tag{D.44}$$

Where m is the number of 3D features and s the number of inverse depth features, so that the total number of predicted features is $m + s$. Fig. D.1 shows the way in which this matrix looks when map features are predicted. The zero-elements correspond to the remaining state vector elements.

We can calculate the innovation covariance by:

$$\mathbf{S} = \mathbf{H}\mathbf{P}_B\mathbf{H}^T + \mathbf{R}$$

Where $\mathbf{P}_B$ is the covariance of the current state vector and $\mathbf{R}$ is the sensor covariance used to initialize depth and inverse depth points defined by $diag(\mathbf{R}_{3D}, \mathbf{R}_{ID})$ correspondingly.

Bibliography

[Agrawal 06] M. Agrawal & K. Konolige. *Real-time Localization in Outdoor Environments using Stereo Vision and Inexpensive GPS*. In International Conference on Pattern Recognition (ICPR), 2006.

[Bailey 00] T. Bailey, E. M. Nebot, J. K. Rosenblatt & H. F. Durrant-Whyte. *Data Association for Mobile Robot Navigation: A Graph Theoretic Approach*. In Proc. IEEE Int. Conf. Robotics and Automation, pages 2512–2517, San Francisco, California, 2000.

[Bailey 03] T. Bailey. *Constrained initialisation for bearing-only SLAM*. In Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on, volume 2, 2003.

[Bailey 06] T. Bailey & H. Durrant-Whyte. *Simultaneous Localization and Mapping (SLAM): Part II*. IEEE Robotics & Automation Magazine, vol. 13, no. 3, pages 108–117, 2006.

[Ballard 81] D. H. Ballard. *Generalizing the Hough transform to detect arbitrary shapes*. Pattern Recognition, vol. 13, no. 2, pages 111–122, 1981.

[Bar-Shalom 88] T. Bar-Shalom & T.E. Fortmann. Tracking and data association. Academic Press In., 1988.

[Bar-Shalom 01] Y. Bar-Shalom, X. R. Li & T. Kirubarajan. Estimation with applications to tracking and navigation. John Willey and Sons, New York, 2001.

[Bishop 06] Christopher M. Bishop. Pattern recognition and machine learning. Springer, 2006.

[Bosse 04] M. Bosse, P. M. Newman, J. J. Leonard & S. Teller. *SLAM in large-scale cyclic environments using the atlas framework*. Int. J. Robotics Research, vol. 23, no. 12, pages 1113–1139, December 2004.

[Bryson 07] M. Bryson & S. Sukkarieh. *Building a robust implementation of bearing-only inertial slam for a uav*. Journal of Field Robotics, vol. 24, no. 1-2, pages 113–143, 2007.

[Burgard 96] W. Burgard, D. Fox, D. Hennig & T. Schmidt. *Estimating the absolute position of a Mobile Robot Using Position Probability Grids.* In Proc. of the Fourteenth National Conference on Artificial Intelligence (AAAi-96), 1996.

[Castellanos 99a] J. A. Castellanos, J. M. M. Montiel, J. Neira & J. D. Tardós. *The SPmap: A Probabilistic Framework for Simultaneous Localization and Map Building.* IEEE Trans. Robotics and Automation, vol. 15, no. 5, pages 948–953, 1999.

[Castellanos 99b] J. A. Castellanos & J. D. Tardós. Mobile robot localization and map building: A multisensor fusion approach. Kluwer Academic Publishers, Boston, Mass., 1999.

[Castellanos 04] J. A. Castellanos, J. Neira & J. D. Tardós. *Limits to the Consistency of EKF-based SLAM.* In 5th IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, 2004.

[Chatila 85] R. Chatila & J. Laumond. *Position referencing and consistent world modeling for mobile robots.* In Proc. IEEE Int. Conf. Robotics and Automation, volume 2, 1985.

[Civera 07] J. Civera, A.J Davison & J.M.M Montiel. *Inverse Depth to Depth Conversion for Monocular SLAM.* In Proc. IEEE Int. Conf. Robotics and Automation, Roma, Italy, April 2007.

[Civera 08] J. Civera, A. J. Davison & J. M. M. Montiel. *Inverse Depth Parametrization for Monocular SLAM.* IEEE Trans. Robotics, vol. 24, no. 5, October 2008.

[Clemente 07] Laura Clemente, Andrew J. Davison, Ian D. Reid, José Neira & J. D. Tardós. *Mapping Large Loops with a Single Hand-Held Camera.* In Proc. Robotics: Science and Systems, Atlanta, GA, USA, June 2007.

[Comport 07] A.I. Comport, E. Malis & P. Rives. *Accurate Quadri-focal Tracking for Robust 3D Visual Odometry.* In IEEE Int. Conf. on Robotics and Automation, ICRA, April, 2007.

[Csorba 98] M. Csorba. *Simultaneous localisation and map building.* PhD thesis, University of Oxford, 1998.

[Davis 06] Timothy A. Davis. Direct Methods for Sparse Linear Systems. Siam, 2006.

[Davison 98] A.J. Davison. *Mobile Robot Navigation using Active Vision.* PhD thesis, University of Oxford, 1998.

[Davison 01] A.J. Davison & N. Kita. *3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain.* In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Kauai, 2001.

[Davison 02] Andrew J. Davison & David W. Murray. *Simultaneous Localization and Map-Building Using Active Vision.* IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pages 865–880, 2002.

[Davison 03] A. J. Davison. *Real-Time Simultaneous Localisation and Mapping with a Single Camera.* In Proc. Int. Conf. Computer Vision, Nice, Oct 2003.

[Davison 07] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton & Oliver Stasse. *MonoSLAM: Real-Time Single Camera SLAM.* IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 29, no. 6, pages 1052–1067, June 2007.

[Deans 00] M. Deans & M. Hebert. *Experimental Comparison of Techniques for Localization and Mapping Using a Bearing-Only Sensor.* In S. Singh D. Rus, editor, Int. Symp. on Experimental Robotics, ISER 2000. Lecture Notes in Control and Information Science, volume 271, pages 395–404, Waikiki, Hawaii, USA, 2000. Springer.

[Dellaert 06] F. Dellaert & M. Kaess. *Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing.* Int. J. Robotics Research, vol. 25, no. 12, December 2006.

[Dissanayake 01] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. F. Durrant-Whyte & M. Csorba. *A Solution to the Simultaneous Localization and Map Building (SLAM) Problem.* IEEE Trans. Robotics and Automation, vol. 17, no. 3, pages 229–241, 2001.

[Durrant-Whyte 06] H. Durrant-Whyte & T. Bailey. *Simultaneous localization and mapping: part I.* IEEE Robotics & Automation Magazine, vol. 13, no. 2, pages 99–110, 2006.

[Ellekilde 07] L.P. Ellekilde. *Dense 3D Map Construction for Indoor Search and Rescue.* JOURNAL OF FIELD ROBOTICS, vol. 24, no. 1-2, page 71, 2007.

[Estrada 05] C. Estrada, J. Neira & J. D. Tardós. *Hierarchical SLAM: real-time accurate mapping of large environments.* IEEE Trans. Robotics, vol. 21, no. 4, pages 588–596, August 2005.

[Eustice 05] R. Eustice, M. Walter & J. Leonard. *Sparse extended information filters: Insights into sparsification.* In Proc. IEEE/RJS Int.

Conference on Intelligent Robots and Systems, Edmonton, Alberta, Canada, August 2005.

[Eustice 06] R. M. Eustice, H. Singh & J. J. Leonard. *Exactly Sparse Delayed-State Filters for View-Based SLAM.* IEEE Trans. Robotics, vol. 22, no. 6, pages 1100–1114, Dec 2006.

[Fitzgibbons 02] T. Fitzgibbons & E. Nebot. *Bearing Only SLAM using Colour-based Feature Tracking.* In 2002 Australasian Conference on Robotics and Automation, Auckland, New Zealand, 2002.

[Folkesson 05] J. Folkesson, P. Jensfelt & H.I. Christensen. *Vision SLAM in the Measurement Subspace.* In Proc. IEEE Int. Conf. Robotics and Automation, pages 30–35, 2005.

[Folkesson 06] J. Folkesson & HI Christensen. *Graphical SLAM for outdoor applications.* Journal of Field Robotics, vol. 23, no. 1, pages 51–70, 2006.

[Fox 98] D. Fox, W. Burgard & S. Thrun. *Active Markov Localization for Mobile Robots.* Robotics and Autonomous Systems, vol. 25, no. 3-4, pages 195–207, 1998.

[Fox 99] D. Fox, W. Burgard, Dellaert F. & S. Thrun. *Monte Carlo Localization: Efficient Position Estimation for Mobile Robots.* In Proc. of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), 1999.

[Frese 05] U. Frese. Treemap: An o(logn) algorithm for simultaneous localization and mapping, chapter Spatial Cognition IV, page 455476. Springer Verlag, 2005.

[Frese 06] U. Frese. *Treemap: An O(log n) algorithm for indoor simultaneous localization and mapping.* Autonomous Robots, vol. 21, no. 2, pages 103–122, September 2006.

[Frese 07] U. Frese. *Efficient 6-DOF SLAM with Treemap as a Generic Backend.* In IEEE Int. Conference on Robotics and Automation (ICRA), Rome, 2007.

[Gil 06] A. Gil, O. Reinoso, O. Martínez-Mozos, C. Stachniss & W. Burgard. *Improving Data Association in Vision-based SLAM.* In Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Beijing, China, 2006.

[Goedemé 07] T. Goedemé, M. Nuttin, T. Tuytelaars & L. Van Gool. *Omnidirectional Vision Based Topological Navigation.* International Journal of Computer Vision, vol. 74, no. 3, pages 219–236, 2007.

[Grimson 90a] W. E. L. Grimson. Object recognition by computer: The role of geometric constraints. The MIT Press, Cambridge, Mass., 1990.

[Grimson 90b] W. E. L. Grimson & D. P. Huttenlocher. *On the Sensitivity of the Hough Transform for Object Recognition.* IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 12, no. 3, pages 255–274, 1990.

[Grisetti 05] Giorgio Grisetti, Cyrill Stachniss & Wolfram Burgard. *Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling.* In Proc. of the IEEE International Conference on Robotics and Automation (ICRA), pages 2443–2448, Barcelona, Spain., 2005.

[Grisetti 07a] Giorgio Grisetti, Slawomir Grzonka, Cyrill Stachniss, Patrik Pfaff & Wolfram Burgard. *Efficient Estimation of Accurate Maximum Likelihood Maps in 3D.* In IROS, 2007.

[Grisetti 07b] Giorgio Grisetti, Cyrill Stachniss, Slawomir Grzonka & Wolfram Burgard. *A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent.* In Robotics: Science and Systems (RSS), Atlanta, GA, USA, 2007.

[Grisetti 08] Giorgio Grisetti, Dario Lodi Rizzini, Cyrill Stachniss, Edwin Olson & Wolfram Burgard. *Online Constraint Network Optimization for Efficient Maximum Likelihood Mapping.* In ICRA, San Diego, USA, October 2008.

[Guivant 01] J. E. Guivant & E. M. Nebot. *Optimization of the Simultaneous Localization and Map-Building Algorithm for Real-Time Implementation.* IEEE Trans. Robotics and Automation, vol. 17, no. 3, pages 242–257, 2001.

[Guivant 02] J. E. Guivant, F. R. Masson & E. M. Nebot. *Simultaneous localization and map building using natural features and absolute information.* Robotics and Autonomous Systems, vol. 40, pages 79–90, 2002.

[Guivant 03] J. E. Guivant & E. M. Nebot. *Solving Computational and Memory Requirements of Feature-Based Simultaneous Localization and Mapping Algorithm.* IEEE Trans. Robotics and Automation, vol. 19, no. 4, pages 749–755, August 2003.

[Gutmann 99] J. S. Gutmann & K. Konolige. *Incremental Mapping of Large Cyclic Environments.* In IEEE Int. Symp. on Computational Intelligence in Robotics and Automation CIRA, pages 318–325, 1999.

[Hartley 00] R. Hartley & A. Zisserman. Multiple view geometry in computer vision. Cambridge University Press, Cambridge, U. K., 2000.

[Huang 07] Shoudong Huang & Gamini Dissanayake. *Convergence and Consistency Analysis for Extended Kalman Filter Based SLAM.* IEEE Trans. Robotics, vol. 23, no. 5, pages 1036–1049, October 2007.

[Hygounenc 04] E. Hygounenc, I.K. Jung, P. Soueres & S. Lacroix. *The autonomous blimp project of LAAS-CNRS: Achievements in flight control and terrain mapping.* International Journal of Robotics Research, vol. 23, no. 4, pages 473–511, 2004.

[Iocchi 00] L. Iocchi, K. Konolige & M. Bajracharya. *Visually realistic mapping of a planar environment with stereo.* In Proceesings of the 2000 International Symposium on Experimental Robotics, 2000.

[Jensfelt 06] P. Jensfelt, D. Kragic, J. Folkesson & M. Bjorkman. *A framework for vision based bearing only 3D SLAM.* In Proc. of the IEEE International Conference on Robotics and Automation (ICRA06), 2006.

[Julier 97] S. Julier & J. Uhlmann. *A new extension of the Kalman Filter to nonlinear systems.* In International Symposium on Aerospace/Defense Sensing, Simulate and Controls, Orlando, FL, 1997.

[Julier 01a] S. J. Julier & J. K. Uhlmann. *A Counter Example to the Theory of Simultaneous Localization and Map Building.* In 2001 IEEE Int. Conf. on Robotics and Automation, pages 4238–4243, Seoul, Korea, 2001.

[Julier 01b] Simon J. Julier. *A Sparse Weight Kalman Filter Approach to Simultaneous Localisation and Map Building.* In Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems, volume 1, pages 1251–1256, Hawaii, October 2001.

[Jung 03] I.K. Jung & S. Lacroix. *High resolution terrain mapping using low altitude aerial stereo imagery.* In Proceedings of the 9th International Conference on Computer Vision, Nice, pages 946–951, 2003.

[Kaess 07] M. Kaess, A. Ranganathan & F. Dellaert. *iSAM: Fast Incremental Smoothing and Mapping with Efficient Data Association.* In IEEE Intl. Conf. on Robotics and Automation, ICRA, Rome, Italy, Apr 2007.

[Kim 03] J.H. Kim & S. Sukkarieh. *Airborne Simultaneous Localisation and Map Building.* In Proc. IEEE Int. Conf. Robotics and Automation, Taipei, Taiwan, September 2003.

[Knight 01] J. Knight, A. Davison & I. Reid. *Towards Constant Time SLAM using Postponement*. 2001.

[Konolige 06] K. Konolige, M. Agrawal, R.C. Bolles, C. Cowan, M. Fischler & BP Gerkey. *Outdoor mapping and navigation using stereo vision*. In Intl. Symp. on Experimental Robotics, 2006.

[Kwok 04] NM Kwok & G. Dissanayake. *An efficient multiple hypothesis filter for bearing-only SLAM*. In Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, volume 1, 2004.

[Kwok 05] NM Kwok, G. Dissanayake & QP Ha. *Bearing-only SLAM Using a SPRT Based Gaussian Sum Filter*. In Robotics and Automation, 2005. Proceedings of the 2005 IEEE International Conference on, pages 1109–1114, 2005.

[LaValle 04] S. M. LaValle, M. S. Braincky & S. R. Lindemann. *On the Relationship Between Classical Grid search and Probabilistic Roadmaps*. Int. J. Robotics Research, vol. 23, no. 7-8, pages 673–692, 2004.

[Lemaire 05] T. Lemaire, S. Lacroix & J. Sola. *A practical 3D bearing-only SLAM algorithm*. In Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on, pages 2449–2454, 2005.

[Lemaire 07a] T. Lemaire, C. Berger, I.K. Jung & S. Lacroix. *Vision-Based SLAM: Stereo and Monocular Approaches*. International Journal of Computer Vision, vol. 74, no. 3, pages 343–364, 2007.

[Lemaire 07b] T. Lemaire & S. Lacroix. *SLAM with panoramic vision*. Journal of Field Robotics, vol. 24, no. 1-2, pages 91–111, 2007.

[Leonard 91] J.J. Leonard & H.F. Durrant-Whyte. *Simultaneous Map Building and Localization for an Autonomous Mobile Robot*. In Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems, pages 1442–1447, Osaka, Japan, 1991.

[Leonard 92] J.J. Leonard, H.F. Durrant-Whyte & I.J. Cox. *Dynamic Map Building for an Autonomous Mobile Robot*. Int. J. Robotics Research, vol. 11, no. 4, pages 286–298, 1992.

[Leonard 01] J. Leonard & H.S.J. Feder. *Decoupled Stochastic Mapping*. IEEE Journal of Oceanic Engineering, vol. 26, no. 4, pages 561–571, 2001.

[Leonard 03] J.J. Leonard & P.M. Newman. *Consistent, Convergent and Constant-Time SLAM*. In Proc. Int. Joint Conf. Artificial Intelligence, Acapulco, Mexico, August 2003.

[Lim 00] J. H. Lim & J. J. Leonard. *Mobile robot relocation from echolocation constraints*. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 22, no. 9, pages 1035–1041, September 2000.

[Lindemann 04] S. R. Lindemann, A. Yershova & S. M. LaValle. *Incremental Grid Sampling Strategies in Robotics*. In Workshop on Algorithmic Foundations of Robotics (WAFR-2004), 2004.

[Maimone 07] M. Maimone, Y. Cheng & L. Matthies. *Two Years of Visual Odometry on the Mars Exploration Rovers*. Journal of Field Robotics, 2007.

[Martinez-Cantin 05] R. Martinez-Cantin & J. A. Castellanos. *Unscented SLAM for large-scale outdoor environments*. In Proc. IEEE/RJS Int. Conference on Intelligent Robots and Systems, pages pp. 328–333, Edmonton, Alberta, Canada, 2005.

[Montemerlo 02] M. Montemerlo, S. Thrun, D. Koller & B. Wegbreit. *FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem*. In Proceedings of the AAAI National Conference on Artificial Intelligence, Edmonton, Canada, 2002. AAAI.

[Montemerlo 03] Michael Montemerlo, Sebastian Thrun, Daphne Koller & Ben Wegbreit. *FastSLAM* 2.0*: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges*. In Proc. Int. Joint Conf. Artificial Intelligence, 2003.

[Montiel 06] J. M. M. Montiel, Javier Civera & Andrew J. Davison. *Unified Inverse Depth Parametrization for Monocular SLAM*. In Proc. Robotics: Science and Systems, Philadelphia, USA, August 2006.

[Murphy 01] K. Murphy & S. Russell. *Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks*. In in A. Doucet ed. : Sequential Monte Carlo Methods in Practice, 2001.

[Neira 01] J. Neira & J. D. Tardós. *Data Association in Stochastic Mapping Using the Joint Compatibility Test*. IEEE Trans. Robotics and Automation, vol. 17, no. 6, pages 890–897, 2001.

[Neira 03] J. Neira, J. D. Tardós & J. A. Castellanos. *Linear time vehicle relocation in SLAM*. In IEEE Int. Conf. on Robotics and Automation, pages 427–433, Taipei, Taiwan, September 2003.

[Newman 03] P. Newman, J. Leonard & R. Rikoski. *Towards Constant-Time SLAM on an Autonomous Underwater Vehicle Using Synthetic Aperture Sonar*. In 11th International Symposium on Robotics Research, Sienna, Italy, 2003.

[Newman 06] P. Newman, D. Cole & K. Ho. *Outdoor SLAM using visual appearance and laser ranging.* In Proc. of the IEEE International Conference on Robotics and Automation, 2006.

[Ni 07] K. Ni, D. Steedly & F. Dellaert. *Tectonic SAM: Exact, Out-of-Core, Submap-Based SLAM.* In 2007 IEEE Int. Conf. on Robotics and Automation, Rome, Italy, April 2007.

[Nister 06] D. Nister, O. Naroditsky & J. Bergen. *Visual odometry for ground vehicle applications.* Journal of Field Robotics, vol. 23, no. 1, pages 3–20, 2006.

[Nüchter 07] A. Nüchter, K. Lingemann, J. Hertzberg & H. Surmann. *6D SLAM3D mapping outdoor environments: Research Articles.* Journal of Field Robotics, vol. 24, no. 8-9, pages 699–722, 2007.

[Olson 06] Edwin Olson, John Leonard & Seth Teller. *Fast Iterative Optimization of Pose Graphs with Poor Initial Estimates.* pages 2262–2269, 2006.

[Olson 07] E. Olson, J. Leonard & S. Teller. *Spatially-Adaptive Learning Rates for Online Incremental SLAM.* In Proceedings of Robotics: Science and Systems, Atlanta, GA, USA, June 2007.

[Papoulis 02] A. Papoulis & S.U. Pillai. Probability, Random Variables and Stochastic Processes. McGraw-Hill, 2002.

[Paskin 03] M. A. Paskin. *Thin Junction Tree Filters for Simultaneous Localization and Mapping.* In Proc. Int. Joint Conf. Artificial Intelligence, pages 1157–1164, San Francisco, CA., 2003.

[Paz 07a] Lina M. Paz, Patric Jensfelt, J. D. Tardós & J. Neira. *EKF SLAM updates in* O(n) *with Divide and Conquer SLAM.* In Proc. IEEE Int. Conf. Robotics and Automation, Rome, Italy, April 2007.

[Paz 07b] Lina M. Paz, J.Guivant, J. D. Tardós & J. Neira. *Data Association in* O(n) *for Divide and Conquer SLAM.* In Proc. Robotics: Science and Systems, Atlanta, GA, USA, June 2007.

[Paz 08] Lina M. Paz, Juan D. Tardós & José Neira. *Divide and Conquer: EKF SLAM in $O(n)$.* Transactions on Robotics, vol. 24, no. 5, October 2008.

[Pinies 07] P. Pinies & J. D. Tardós. *Scalable SLAM building Conditionally Independent Local Maps.* In IEEE/RSJ International Conference on Intelligent Robots and Systems IROS, November, 2007.

[Piniés 08] Pedro Piniés & Juan D. Tardós. *Large Scale SLAM Building Conditionally Independent Local Maps: Application to Monocular Vision.* Transactions on Robotics, vol. 24, no. 5, October 2008.

[Royer 07] E. Royer, M. Lhuillier, M. Dhome & J.M. Lavest. *Monocular Vision for Mobile Robot Localization and Autonomous Navigation.* International Journal of Computer Vision, vol. 74, no. 3, pages 237–260, 2007.

[Rubner 98] Y. Rubner, C. Tomasi & L.J. Guibas. *A metric for distributions with applications to image databases.* In Proc. fo the IEEE International Conference on Computer Vision, 1998.

[Saez 05] J.M. Saez, F. Escolano & A. Penalver. *First Steps towards Stereo-based 6DOF SLAM for the Visually Impaired.* In Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)-Workshops-Volume 03. IEEE Computer Society Washington, DC, USA, 2005.

[Se 02] S. Se, D. Lowe & J. Little. *Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks.* Int. J. Robotics Research, vol. 21, no. 8, pages 735–758, 2002.

[Sim 05] R. Sim, P. Elinas, M. Griffin & J.J. Little. *Vision-based SLAM using the rao-blackwellised particle filter.* In IJCAI Workshop on Reasoning with Uncertainty in Robotics (RUR), 2005.

[Sim 07] R. Sim, P. Elinas & J.J. Little. *A Study of the Rao-Blackwellised Particle Filter for Efficient and Accurate Vision-Based SLAM.* International Journal of Computer Vision, vol. 74, no. 3, pages 303–318, 2007.

[Simond 04] N. Simond & P. Rives. *Trajectography of an uncalibrated stereo rig in urban environments.* In Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on, volume 4, 2004.

[Smith 86] R. C. Smith & P. Cheeseman. *On the Representation and Estimation of Spatial Uncertainty.* Int. J. Robotics Research, vol. 5, no. 4, pages 56–68, 1986.

[Smith 88] R. Smith, M. Self & P. Cheeseman. *A Stochastic Map for Uncertain Spatial Relationships.* In O. Faugeras & G. Giralt, editors, Robotics Research, The Fourth Int. Symposium, pages 467–474. The MIT Press, 1988.

[Smith 06] P. Smith, I. Reid & A. Davison. *Real-Time Monocular SLAM with Straight Lines.* In Proceedings of the British Machine Vision Conference, volume 1, pages 17–26, 2006.

[Sola 05] J. Sola, A. Monin, M. Devy & T. Lemaire. *Undelayed initialization in bearing only SLAM.* In Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on, pages 2499–2504, 2005.

[Sola 07] J. Sola, A. Monin & M. Devy. *BiCamSLAM: Two times mono is more than stereo.* In IEEE Int.Conference on Robotics and Automation (ICRA), Rome, 2007.

[Stachniss 05] Cyrill Stachniss, Dirk Haehnel, Wolfram Burgard & Giorgio Grisetti. *On Actively Closing Loops in Grid-based FastSLAM.* Advanced Robotics. The Int. Journal of the Robotics Society of Japan (RSJ), vol. 19, no. 10, pages 1059–1080, 2005.

[Tardós 02] J. D. Tardós, J. Neira, P. M. Newman & J. J. Leonard. *Robust Mapping and Localization in Indoor Environments using Sonar Data.* Int. J. Robotics Research, vol. 21, no. 4, pages 311–330, 2002.

[Thrun 02] S. Thrun. *Particle Filters in Robotics.* In Proceedings of Uncertainty in AI (UAI), 2002.

[Thrun 04] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y. Ng, Zoubin Ghahramani & Hugh Durrant-Whyte. *Simultaneous Localization and Mapping with Sparse Extended Information Filters.* Int. J. Robotics Research, vol. 23, no. 7-8, pages 693–716, 2004.

[Uhlmann 01] J.K. Uhlmann. *Introduction to the Algorithmics of Data Association in Multiple-Target Tracking.* In Handbook of Multisensor Data Fusion. CRC Press, Boca Raton, FL, 2001.

[Ulrich 00] I. Ulrich & I. Nourbakhsh. *Appearance-based place recognition for topological localization.* In Proc. of the IEEE International Conference on Robotics and Automation, pages 1023–1029, 2000.

[Walter 04] M. Walter, R. Eustice & J. Leonard. *A Provably Consistent Method for Imposing Sparsity in Feature-based SLAM Information Filters.* In Proc. Int. Symp. Robotics Research, 2004.

[Williams 01] S. B. Williams. *Efficient Solutions to Autonomous Mapping and Navigation Problems.* PhD thesis, Australian Centre for Field Robotics, University of Sydney, September 2001. Available at http://www.acfr.usyd.edu.au/.

[Williams 02] S. B. Williams, G. Dissanayake & H. Durrant-Whyte. *An efficient approach to the simultaneous localisation and mapping problem.* In Proc. IEEE Int. Conf. Robotics and Automation, volume 1, pages 406–411, Washington DC, 2002.

[Williams 07a] B. Williams, G. Klein & I. Reid. *Real-time SLAM relocalisation.* In Proc. International Conference on Computer Vision, 2007.

[Williams 07b] Brian Williams, Paul Smith & Ian Reid. *Automatic Relocalisation for a Single-Camera Simultaneous Localisation and Mapping System.* In Proc. IEEE Int. Conf. Robotics and Automation, pages 2784–2790, Roma, Italy, April 2007.

[Zhang 92] Z. Zhang & O.Faugueras. *Three-dimensional motion computation and object segmentation in a long sequence of stereo frames.* International Journal of Computer Vision, vol. 7, no. 3, pages 211–241, 1992.