UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

GUILHERME SCHVARCZ FRANCO

S4FE: Sequential Feature Frequency Filter – Front-End for SLAM

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientador: Prof. Dr. Edson Prestes e Silva Júnior

Porto Alegre
2016

CIP – CATALOGING-IN-PUBLICATION

Franco, Guilherme Schvarcz

S4FE: Sequential Feature Frequency Filter – Front-End for SLAM / Guilherme Schvarcz Franco. – 2016.

15 f.:il.

Advisor: Edson Prestes e Silva Junior;

Thesis (Master) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2015.

1. VisualSLAM. 2. Front-End 3. Bag of Words. I. Silva Júnior, Edson Prestes.

No man is an island,

Entire of itself,

Every man is a piece of the continent,

A part of the main.

If a clod be washed away by the sea,

Europe is the less.

As well as if a promontory were.

As well as if a manor of thy friend's

Or of thine own were:

Any man's death diminishes me,

Because I am involved in mankind,

And therefore never send to know for whom the bell tolls;

It tolls for thee. – *John Donne*

AGRADECIMENTOS

Tenho que agradecer a muitas pessoas por essa etapa concluída. A tantos e por tantos motivos diferentes, que certamente não caberia em apenas uma folha de minha dissertação. Em suma, o célebre poema de John Donne nunca me foi tão evidentemente verdadeiro quanto nesses últimos anos que se passaram, nessa longa aventura que foi o mestrado.

Gostaria de agradecer primeiramente a professora Dra. Simone Ceolin, por ter posto fé em meu esforço, ainda lá em Santa Maria/RS, e ter começado a me moldar como pesquisador dando a base necessária à minha formação científica. Agradeço também ao professor Dr. Edson Prestes, por dado continuidade a minha formação e ter me mostrado os passos mais avançados desse universo acadêmico.

Aos meus amigos de longa data, que ainda estão próximo e também aos que infelizmente não estão mais, por terem me motivado e dado forças para que eu pudesse chegar até este ponto. Especialmente ao Cleber Otto Pedrozo, que muito me ensinou sobre a vida desde muito cedo, a longo dos meus tempos mais turbulentos e dos seus também, até seus últimos dias. Aos amigos mais recentes, que espero que se eternizem, por fazerem de cada um dos meus dias livres uma jornada revigorante explorando a capital gaúcha e região. Por fim, aos colegas de laboratório, que me auxiliaram em minhas dúvidas.

Por fim, mas não menos importante, a minha família e especialmente a minha mãe Gislei Terezinha Schvarcz. Por ter me dado a base necessária que ninguém mais poderia ter feito e a força que precisava quando nem mesmo eu a tinha mais.

S4FE - Sequential Feature Frequency Filter – Front-End for SLAM

RESUMO

Fechamento de loops é um dos principais processos das estratégias de SLAM baseadas em grafos, usadas para estimar o erro de deslocamento acumulado à ser minimizado pela técnica. Neste sentido, boas correspondências de cenas permitem criar uma conexão entre dois nós do grafo que está sendo construído para representar o ambiente. Contudo, falsas correspondências podem levar essas estratégias a um estado irreversível de falsa representação do ambiente.

Neste trabalho, um método robusto baseado em *features* que usa sequências de imagens para reconhecer áreas revisitadas é apresentado. Este método usa a abordagem de Bag-of-Words para reduzir efeitos de iluminação e uma ponderação TF-IDF para ressaltar as principais *features* que descrevem cada cena. Além disso, um algoritmo baseado na técnica de Mean Shift é usado sobre uma matriz de similaridade para identificar a possível trajetória seguida pelo robô e melhorar a detecção de fechamento de loop. O método apresentado foi testado em um ambiente aberto usando sequências de imagens coletadas com usando uma câmera de mão e um drone modelo Parrot ArDrone 2.0.

Palavras-chave: VisualSLAM, Mean Shift, Bag of Words.

ABSTRACT

Loop closure recognition is one of the main processes of graph-based SLAM strategies, used to estimate the accumulated motion error to be minimized by the technique. Good scene correspondences allow to create constraints between two nodes in the graph that is currently being built to represent the environment that the robot is immersed. However, false correspondences can lead these strategies to an irreversible wrong environment representation.

In this work, we present a robust feature-based loop closure approach that uses image sequence matching to recognize revisited areas. This approach uses Bag-of-Words to reduce the effects of lightning changes and a TF-IDF weighting to enhance the main features that describe each scene. Besides, an algorithm based on Mean Shift is used over a similarity matrix to identify the possible trajectory followed by the robot and improve the loop closure detection. Our method is tested in a GPS-denied outdoor environment using image sequences collected using a handheld camera and a Parrot ArDrone 2.0.

Keywords: VisualSLAM, Mean Shift, Bag of Words.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BoW	Bag of Words
FAB-MAP	Fast Appearance-Based Mapping
ML	Maximum Likelihood
MSA	Mean Shift Algorithm
ROI	Region of Interest
S4FE	Sequential Feature Frequency Filter – Front-End
SAD	Sum of Absolute Differences
SLAM	Simultaneous Localization and Mapping
SSD	Sum of Squared Differences
SURF	Speeded Up Robust Features
TF-IDF	Term Frequency – Inverse Document Frequency
VisualSLAM	Visual Simultaneous Localization and Mapping
VPR	Visual Place Recognition

SUMÁRIO

1 INTRODUCTION

Humans have always dreamed of going further than their physical capacities can take them without put their lives in dangerous. Travel to outside space to explore the unknown, visit inhabitable territories searching for basic resources to support life or go to deep waters looking for what else the earth has to offer. There are many jobs that may expose human lives to risk in their own habitat. Remove mines of an ancient battlefield to allow the peaceful population of that region lives without afraid; safely defuse bombs from terrorist attacks; search for remaining life on wreckage of buildings after an environmental disaster or digging tunnels under the soil are some of them. Alike these scenarios, an industrial job can expose an employee life to danger. On several cases, a repetitive work, usually done in this kind of labor, can cause injuries to the worker. Moreover, this job can be described as a tiresome employment where the human cognitive abilities are not required.

For all these dull, dirty and dangerous issues, the possibility to use autonomous robots looks promising. A mechanical instrument with certain intelligence could safely perform the necessary work without costing the health or life of humans. However, developing a robot capable to replace a human on these scenarios is a very complex and challenging task. To build robots with such abilities some requirements have to be fulfilled. For instance, for a robot that should move in an environment, three basic abilities are required: (i) path planning, (ii) mapping and (iii) localization (Thrun; Burgard; Fox, 2005). As can be seen in Figure 1.1.

Planning a path (i) is understood as the ability of a robot to determine and follow a path in a known environment. It is not an easy mission to accomplish. The route must be free of obstacles and be the shortest path between the robot's current position and the desired goal. Moreover, it should avoid regions with difficult traversing, such as narrow spaces, grass or sand (Maffei; Jorge; Prestes; Kolberg, 2014; E Prestes; Trevisan; Idiart; Engel, 2003; Edson Prestes; Idiart, 2009). Mapping (ii) is the task of registering an unknown environment, where the robot is traversing, in a representation that can be manipulated. The type of representation map is strongly related to the navigation and localization strategies. This choice impacts directly the computational cost and the precision of the robot's approach (Siegwart; Nourbakhsh, 2004).

Finally, localization (iii) is the process where the robot identifies its current position in an already mapped area. Many approaches use a localization system based on beacons (Neuland et al., 2014; Yol; Delabarre; Dame; Dartois; Marchand, 2014). An example of these systems is the positioning system using orbiting satellites, such as GLONASS, GALILEO or GPS[1], that localize any receiver device around the globe. The drawback of this kind of system is that they require the presence of pre-localized emitter equipment, which only can exist on a beforehand manipulated environment, limiting the application. Besides that, these beacon systems can fail in several situations. In outdoor environments where the robot is surrounded by buildings or in a dense forest, for instance, a navigation based on GPS is not allowed due to satellite's poor signal and reflection problems.

In these cases, the robot needs to rely on usually not precise sensors such as speed or odometer sensors. The continuum integration of nonsystematic errors from these sensors eventually guide the robot to belief that it is in a position in the environment that actually it is not. To address this problem, a usual approach is to try to recognize regions where the robot already passed. Trusting that it has visited the same area, it can estimate those errors accumulated from the last time that it was there.

This work proposes a loop detection strategy based only on cameras. Every image gathered during the robot's traversing is storage in a computational representation of the world known by the robot. Then, new scenes registered by the robot are compared and a confidence measurement is computed to each one of the already images learnt.

1.1 Motivation

Performing the three main necessary tasks to an autonomous mobile robot is already a challenge. Nevertheless, in most robotics applications, these problems cannot be solved independently. Figure 1.1 shows all possible combinations of these tasks (Makarenko; Williams; Bourgault; Durrant-Whyte, 2002).

[1] More information at https://glonass-iac.ru/ (GLONASS), http://www.gsa.europa.eu/ (GALILEO) and http://www.schriever.af.mil/GPS/ (GPS).

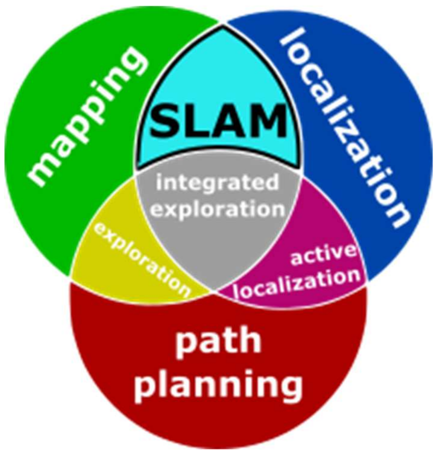


Figure 1.1 The three main tasks of an autonomous robot and their combinations, emphasizing the SLAM problem (focus of this work). Figure adapted from (Makarenko et al., 2002)

The issue of *Active Localization* is given when the robot, based on a known map, tries to recalculate the best path to follow during its traversing. This ability is important to a robot that suffers with an imprecise odometry source. To be able to update its current route towards the goal position, the robot must have an estimate about its localization. This task assumes that the robot has an updated representation of the environment which is immutable. However, this is unpractical on a real scenario, where the robot in operating in a world under constant changes.

A more challenging problem is when the robot should navigate in a region where it does not have any knowledge in advance to create a computational representation of the area. On this problem, usually called *exploration of an unknown environment*, the robot must read its sensors in each step to update its environment representation. Then, the robot calculates a path towards the direction of the boundaries between the mapped and unmapped regions. This task has the assumption that the robot has a perfect odometer, what is unrealistic to be acquired in real scenarios. The robot usually slips due to oil, water or other wastes that can be over ground. Sometimes, due to its own physical features, the robot tends to move not respecting the planned route.

The third combination of the three main skills which an autonomous robot should master, is the ability of performing *Simultaneous Localization and Mapping* (SLAM) (Grisetti; Kummerle; Stachniss; Burgard, 2010; Thrun; Montemerlo, 2006). Unlike the before combined tasks, this is a mandatory expertise that a robot must have in order to navigate in the real world. These mapping strategies need to be robust enough to deal with possible divergence of routes. Without assuming a perfect odometry, after each movement, the SLAM process compares the robot's knowledge about the world, acquired so far, to what its sensors are collecting from the environment. This is made to reduce eventual odometry errors that could lead it to build a wrong representation of the world.

In contrast to the two before introduced abilities, which are hard to implement in a real world, many scenarios require a teleoperated robot capable of mapping a new region as it's explored. For instance, this ability could be assigned with a search and rescue task in a disaster region, where a hurricane or landslide made a significantly change on the landscape. Without knowing the situation of the terrain affected by the natural disaster, sending a rescue group of humans to seek for remaining lives could put more people in risk. As a map of the environment is not available yet, a rescue group of robot should have the SLAM ability to properly accomplish the mission.

Finally, the composition of these three previously tasks is known as *Integrated Exploration*. This problem is given when the robot must plan its motion to explore an unknown area, creating the very first registration of the region, at the same time that it is trying to localize itself over this partial environment map. The navigation decision must have a balance between a path that adds more knowledge about the environment and a path that improves the robot's localization. This is a complex problem which has in the core the SLAM and Exploration strategies. Only after satisfactory solving the SLAM problem, the robot is able to choose a navigation path that could help the localization and mapping approaches.

Despite the efforts made by researchers developing new SLAM algorithms, a path divergence still can happen. Commonly, many strategies employ the process of loop closure detection to deal with these unhandled deviations. When the robot recognizes a revisited region, in other words, when a loop is closed, the accumulated error along the path can be measured by the difference of its expected and estimated positions over the learnt map. Usually, in robots equipped with rangefinders, a loop closure is detected when the robot measures a scan reading similar to a structure already stored in the environment representation, according to a likelihood metric, next to its localization belief. However, a single scan reading is poor of information and liable to false positives matches.

In this sense, place recognition using cameras, commonly called *Visual Place Recognition* (VPR), has received considerable attention in the latest years as a loop closure detection strategy for SLAM techniques (Cummins; Newman, 2007, 2010; Milford; Schill; Corke; Mahony; Wyeth, 2011; Milford; Wyeth, 2012). These algorithms, that attempt to solve the SLAM problem using visual information, are known in the literature as Visual SLAM (Davison; Kita, 2001). Cameras are low cost sensor, have small size and low power consumption (Milford; Turner; Corke, 2013) compared to other sensors used for place recognition (Bosse et al., 2003; Lee; Song, 2010; Milford; Wyeth, 2012).

The main idea of VPR is to match two images or two sets of images in a sequence collected during the robot motion to detect revisits and, consequently, close loop. While some strategies prefer to rely on feature-based techniques to compare images, such as SIFT (Lowe, 2004) or SURF (Bay; Ess; Tuytelaars; Gool, 2008; Bay; Tuytelaars; Van Gool, 2006), others prefer to correlate images using appearance-based approaches such as SSD or Mutual Information (Viola; Wells, W.M., 1995).

1.2 Objectives

This work proposes a new strategy for loop closure detection that searches for a matching between two sequences of images to determine loop closures, as part of a SLAM strategy. For our approach, each frame is described by a histogram of their features which are weighted by the relevance of their information to distinguish the scene.

Some prominent algorithms in the Visual SLAM literature (Davison; Kita, 2001) that use features scheme to detect a revisited area have demonstrated encouraging results. FAB-MAP (Cummins; Newman, 2007, 2010) is a state-of-the-art technique that uses *Bag-of-Words* (BoW) (Sivic; Zisserman, 2003) combined with a *Bayesian Network* to identify loop closures through a single frame matching. Others, as SeqSLAM (Milford et al., 2013; Milford; Wyeth, 2012; Pepperell; Corke; Milford, 2014) and Outdoor SLAM (Ho; Newman, 2007) indicate that the detection of sequences of frames along a path tends to decrease incorrect place matching occurrences when compared to a single frame matching. Previous results show evidence that SeqSLAM significantly outperforms FAB-MAP (Milford; Wyeth, 2012).

This work proposes the use of BoW combined with a modified version of *Mean Shift Algorithm* (MSA) (Fukunaga; Hostetler, 1975) to locally track matching sequences, that may represent loops, in a co-occurrence matrix. Our proposal performs the normalization of similarity values in every region of interest (ROI) in the co-occurrence matrix to highlight distinguishable line patterns. If the mean normalized similarity is above a certain threshold, then the corresponding ROI has a possible line that represents a loop. In this situation, MSA is performed.

1.3 Organization

This dissertation is divided as follow: Chapter 2 reviews the fundamentals of SLAM and Visual SLAM presenting two loop closure methods: FAB-MAP (Cummins; Newman, 2007) and SeqSLAM (Milford; Wyeth, 2012). Chapter 3 introduces the core ideas of our method comparing them to the state-of-art works. Then, an overview of the proposed algorithm is presented followed by its formal derivation to clarify in details each process of our approach. Chapter 4 presents all tested environments used to appraise the present work along with a discussion about the challenges faced. Later, the results obtained by our approach are compared to the FAB-MAP and SeqSLAM. The differences between the tested techniques are highlighted and discussed. Finally, Chapter 5 discusses the results obtained and presents the conclusion of this work with a consideration about possible enhancement to our method as a future work.

2 SLAM FOUNDATION

This chapter starts with a review of the foundations of Simultaneous Localization and Mapping techniques, describing the problem in details and formulating a basic mathematical model. Then, a division of the SLAM approaches is introduced and the Visual SLAM strategies are presented since their origin at Graph-based SLAM. Finally, we introduce two main loop closure methods. First, we introduce the FAB-MAP (Cummins; Newman, 2007) approach and the Bayesian Network tree used to deal with noise on the features' descriptors. Then, SeqSLAM (Milford; Wyeth, 2012) and the use of sequence matching are presented.

2.1 SLAM

Simultaneous Localization and Mapping (SLAM) is a fundamental problem in robotics. As introduced in Section 1.1, SLAM problem arises when a robot tries to navigate in a region without having a previous map and any knowledge about its own pose on the environment. SLAM is a complex problem where the robot needs to simultaneously estimate two interdependent variables: its pose on the space and an accurate map representation of the environment. It is more difficult than Localization problem, where the robot has the map and does not need to estimate it. Also more difficult than mapping problem, considering that the poses are unknown and need to be estimated during the mapping process.

Figure 2.1 shows the robot's SLAM scheme. At time t two observations $Z_{1,t}$ and $Z_{2,t}$ are made from landmarks m_1 and m_2. Considering the robot has a perfect knowledge about its pose X_t, the position of the landmarks m_1 and m_2 can be estimated applying the respective observations to the current pose of the robot. After performing an input control u_t, the robot estimates its new pose X_{t+1} and get two new observations: $Z_{2,t+1}$ and $Z_{3,t+1}$ from the landmarks m_2 and m_3. In the sequence, poses X_{t+2} and X_{t+3} are estimated after the control inputs u_{t+1} and u_{t+2}, respectively, and the landmarks position m_3 and m_4 are estimated using the observations $Z_{3,t+2}$ and $Z_{4,t+2}$.

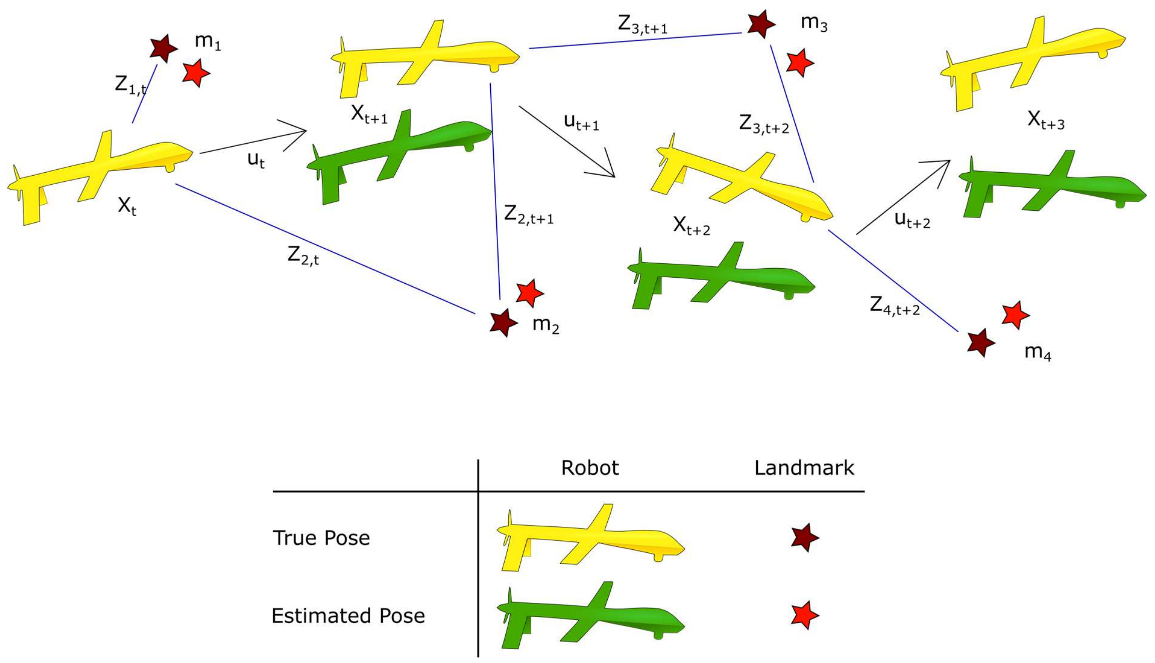


Figure 2.1 A common SLAM scheme. The initial known pose X_t and the sequential estimated poses X_{t+n} after each respective robot's input controls u_{t+n-1}. The landmarks m_k and their respective observations $Z_{k,t+n}$. Image generated by the author.

As introduced earlier, sometimes a robot movement could not be executed as planned. This motion problem usually leads the robot to have a wrong belief about its localization. A SLAM approach could try to improve the robot position exploiting two observations of the same landmark in different positions. In our scenario, the robot could estimate the position of the landmark m_2 at time t using X_t and $Z_{2,t}$. Then, after applying the input control u_t and estimate its new position X_{t+1}, the robot could increase the precision of its estimated pose comparing the computed position of the landmark m_2 at time t to its new observation $Z_{2,t+1}$ at time $t+1$. With a more precise position X_{t+1} as base, the observation $Z_{3,t+1}$ allows the robot to calculate a better position to the first time observed m_3, consequently creating a better representation of the environment.

We must consider that the observations of presented scheme are not free of errors. The sensors used to perceive the environment are susceptible to misreading. For instance, a sonar used to measure the range from an obstacle, commonly surfer interference from another source of acoustic wave or spectral reflection of a previously sent signal. Therefore, all measurements made by the robot must have an associated uncertainty. There are many forms to model this uncertainty. While the Extended

Kalman Filter SLAM (EKF SLAM) (Smith; Self; Cheeseman, 1988) represents that uncertainty using an approximated Gaussian distribution, the GMapping (Grisetti; Stachniss; Burgard, 2005) technique uses a Rao-Blackwellized particle filter scheme.

Beside their differences, all SLAM techniques are confronting the same problem, which can be described by a Bayesian Network. Figure 2.2 shows the relations between the observed and the hidden variables of this network. The observed variables, the set of actions $u_{1:t-1} = \{u_1, u_2, u_3, u_4, \dots, u_{t-1}\}$ performed by the robot up to time $t-1$ and the set observations $Z_{1:t} = \{Z_1, Z_2, Z_3, Z_4, \dots, Z_t\}$ measured up to time t, are those ones that can be directly measured. Hidden variables, the set of robots poses $X_{1:t} = \{X_1, X_2, X_3, X_4, \dots, X_t\}$ and the map m, are the variables not directly measured by the robot and what a SLAM technique have to estimate. According to this model, the observed states are dependent of the hidden states. Therefore, the hidden variables can be inferred from the observed ones.

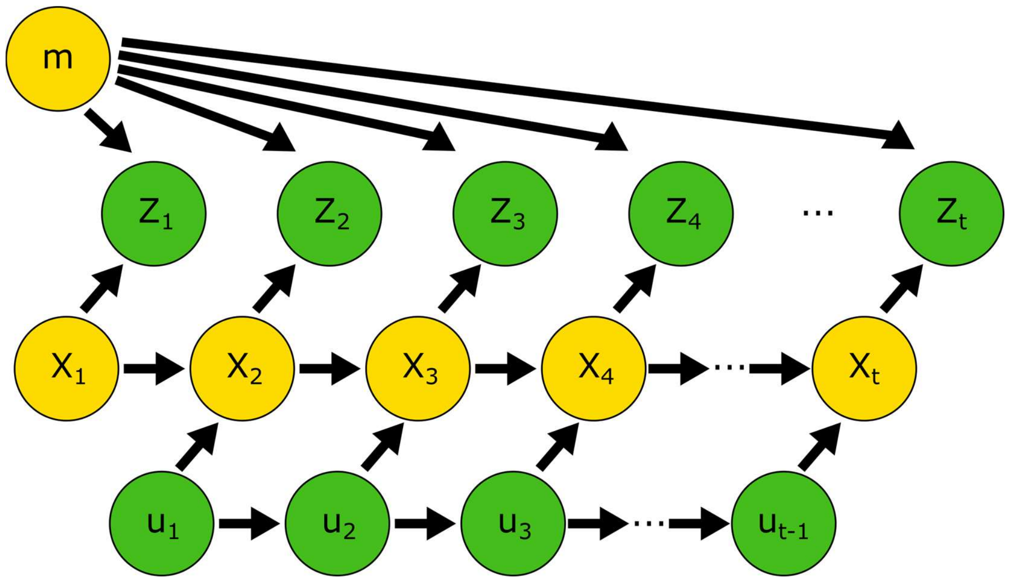


Figure 2.2 The SLAM problem modeled as a Bayesian Network. The observed variables are the actions u_n and observations Z_n. While the hidden variables are the robot's poses X_n and map m. Image generated by the author.

The SLAM techniques can be divided in two groups: Online SLAM and Full SLAM (Thrun et al., 2005). In Full SLAM the entire path $X_{1:t}$ and the map m are the posterior probability computed from all control inputs $u_{1:t-1}$ and observations $Z_{1:t}$, as presented in Figure 2.1. At each new information acquired by the robot, the entire path and map estimation are re-computed to increase the precision. This improvement could be done concerning the relation of each estimated element to the new observations acquired.

$$p(X_{1:t}, m|Z_{1:t}, u_{1:t-1})$$

On the other hand, in Online SLAM formulation, all variables are estimated to a specific moment t. As shown at Equation 2.1, the robot's pose X_t and the map m estimation are the posterior probability computed from all observed states. Usually, many online SLAM algorithms solve this problem by incrementally integrating the past robot's pose estimation. Instead of using all control inputs and observations, they use the last estimation of map and robot's pose to compute the next robot's state. Discarding the inputs already used to calculate an estimation.

$$p(X_t, m|Z_{1:t}, u_{1:t-1}) \tag{2.1}$$

That way, an Online SLAM strategy can be described as the integration of the past estimations from Full SLAM over time (Thrun et al., 2005):

$$p(X_t, m|Z_{1:t}, u_{1:t-1}) = \int\int\int \dots \int p(X_{1:t}, m|Z_{1:t}, u_{1:t-1})\, dx_1 dx_2 dx_3 dx_4 \dots\, dx_{t-1}$$

By operating incrementally, the Online SLAM has the advantage of be lightweight and less computation need to be performed. On the other hand, an imprecise estimation is propagated to all following poses and probably will never be corrected, because the relation between new observations to the older ones is not full exploited. For instance, the recognition that the robot's pose X_t is the same of previously visited pose X_{t-n} can help to correct only the current robot's pose and observations, but not to reduce the uncertainty of the hidden variables from the last time that the robot was on that position.

2.2 Graph-based SLAM and Visual SLAM

Maybe the most studied algorithms in Full SLAM are the graph-based ones. The formulation of this problem was first introduced at 1997 by Lu and Millos (Lu; Milios, 1997), but only became popular years later due to advances on error minimization techniques (Thrun; Montemerlo, 2006). In graph-based SLAM, the robot's poses $X_{1:t}$ and the landmarks $m_{1:n}$ are represented as nodes in a graph and labeled according to their positions on the world. The control inputs $u_{1:t-1}$ and the observations $Z_{1:t}$ acquired from the environment are the constraints represented by the edges of this graph.

The graph-based methods are divided into two parts. The front-end part is responsible of constructing the graph representation and linking the learnt poses by edges. These edges are created by the relation between two poses, given the inputs $u_{1:t-1}$, and by the observations $Z_{1:t}$, between a robot's pose and landmark $m_{1:n}$, or when the front-end method recognizes a revisited area. The second part, called back-end, aims to determine the most reasonable global configuration of the robot's poses, respecting the constraints introduced by the edges of the graph. This configuration is usually computed by an error minimization of a matrix that expresses the relation between the estimated poses of the robot and the measured landmarks.

The graph construction does not care about global optimization, only about local constraints. Therefore, the procedure of this task relies heavily on the raw sensor data. In contrast, the back-end depends mainly on the graph representation of the environment which is sensor agnostic. This knowledge is important to understand that these two tasks can be resolved separately. A front-end strategy does not need to be aware about the back-end methods to fulfil its duty properly. In the same way, the back-end strategy does not need to be aware about the front-end strategy or the kind of sensor used to construct the graph.

Since the first publications of graph-based methods, strategies to the back-end segment have been proposed. Lu and Milios (Lu; Milios, 1997) work have demonstrated how to globally optimize a graph-based map by reducing the error introduced by constraints in a system of equations that express that graph. Although this process has produced good results, it was costly. Exploiting the fact that this technique usually creates a sparse matrix to represent the graph, Dallaert and Kaess (Dellaert; Kaess, 2006) have proposed a sparse matrix factorization to improve the computation time of these approaches. One year later, they add a column ordering heurist to take advantage of the local dependency inherent to the SLAM problem (Kaess; Ranganathan; Dellaert, 2007).

From a different point of view, Olson et. al. (Olson; Leonard; Teller, 2006) have proposed an optimization method based on stochastic gradient descent to iteratively converge to optimal solution. Their results have overcome the main iterative methods to solve a system of equations, such as Gauss-Seidel or LU Decomposition. Following this line of research, Grisetti et. al. (Grisetti; Stachniss; Burgard, 2009) extended Olson's algorithm to the 3D world introducing a parametrization to the nodes of the graph that decreases the speed of convergence.

Finally, probably one of the major contributions to the back-end of the graph-based SLAM theory was introduced by the method named GraphSLAM (Thrun; Montemerlo, 2006). Thrun et. al. introduced a method that method reduces the graph optimization complexity using a variable elimination technique that allows the robot to build a representation with more than 10^8 features. This advance had a great impact at graph-based methods area and remains as the state-of-art.

All these approaches have been substantially reducing the computation complexity across the years. In this work, we explain the common idea to compute the maximum likelihood of a graph configuration that is behind all introduced methods. For a graph where the poses are described by the vector $x = (x_1, \dots, x_T)^T$ and the Gaussian distribution that represents an edge between two poses x_i and x_j is represented by the mean z_{ij} and the information matrix Ω_{ij}. The error $e_{ij}(x_i, x_j, z_{i,j})$ between the observation z_{ij} and the predicted measurement $\hat{z}_{ij}(x_i, x_j)$ is given by:

$$e_{ij}(x_i, x_j, z_{i,j}) = z_{ij} - \hat{z}_{ij}(x_i, x_j)$$

The goal of the back-end of a graph-based SLAM is to minimize the global error produced by all disagreements between the observations and predictions. This is done by moving the nodes inside its zone of uncertainty in a manner to reach an optimized global consistence, without losing the local constraints identified by the robot. Let $\mathcal{C}$ be the set of pairs of indices for all measured constraints z. The sum of forces $F(x)$ given by a graph configuration x and weighted by the uncertainties Ω_{ij} is given by:

$$F(x) = \sum_{(i,j)\in\mathcal{C}} e_{i,j}^{T}\Omega_{i,j}e_{i,j}$$

In order to compute the maximum likelihood configuration x^* of the graph, we seek for an argument that minimize the sum of forces $F(x)$. Which can be expressed by:

$$x^* = \operatorname*{argmin}_{x} F(x)$$

Methods to solve a non-linear least square problem, such as Gauss-Newton and Levenberg-Marquardt, are usually employed to compute the maximum likelihood x^*. However, the approach based on a stochastic gradient descent method introduced by Olso has shown better results with less computation time (Olson et al., 2006), as commented previously.

Until this point we assume that we have a graph representation of the environment which was abstracted from the raw measurements collected by the robot. Such construction is done by converting the raw measurements to edges that connect the poses of the robot to the landmarks identified in the environment. The construction of a graph free of wrong connections is decisive to the right convergence of x^*. However, it must consider that the observation model $p(x_t|m, z_t)$, that expresses those connections, have a multi-modal distribution. Therefore, a single observation z_t has multiple hypotheses to come from a specific landmark. Many front-end SLAM approaches make a greedy decision and link the current robot's pose to the most likely landmark observation. Such strategy is weak and susceptible to false positive matches. Especially when the sensor used by the robot is poor of information, such as rangefinders or beacons system.

In this sense, cameras are a rich source of information to distinct two similar landmarks. These sensors are low cost, have small size and low power consumption. Besides, cameras provide a lot of information that can be used not only for performing SLAM, but also for applications that involve semantic information processing, risk situation analysis, etc. All these qualities have been driving researchers to seek for a vision-based front-end SLAM strategy in the last years, which is commonly called VisualSLAM (Milford et al., 2013). With a more descriptive data from the environment, VisualSLAM have been enabling robots to explore areas substantial larger than before.

While many of those methods combines the depth information obtained from a rangefinder or the heading information informed by a compass with the disambiguation strength of visual information, other methods prefer to pursuit an approach based exclusively on cameras, exploiting all information available from images to perform the robot's mapping and localization.

In the remainder of this work we will discuss about VisualSLAM techniques based exclusively on cameras. First, we present the current state-of-art approaches, FAB-MAP (Cummins; Newman, 2007) and SeqSLAM (Milford; Wyeth, 2012), followed by a detailed explanation of S4FE, our contribution in this area of research, and a comparison between these methods.

2.3 Related Work

At this section we will detail two loop closure methods that are the state-of-art of VisualSLAM literature. These methods will serve as comparison to our approach at Section 4. First, an explanation and mathematical derivation of FAB-MAP (Cummins; Newman, 2007) is introduced. Later, a detailed revision of SeqSLAM (Milford; Wyeth, 2012) is presented. FAB-MAP is known by using a Bayesian Network tree to probabilistic infers the robot's pose while deal with missing features detections. While SeqSLAM strength relays on the use of sequences of images to determine a loop closure in drastic light changing scenarios.

2.3.1 FAB-MAP

FAB-MAP (Cummins; Newman, 2007, 2010) is an online probabilistic front-end method to identify possible loop closures on the robot's path. It uses a visual BoW approach, introduced by Sivic et al (Sivic; Zisserman, 2003), to express the raw information acquired from the environment into words to increase the robustness of the features detector. As a probabilistic method, FAB-MAP expresses the SLAM problem using a discrete Bayesian Network where the robot's poses are conditioned on the features observed. In real experiments (Cummins; Newman, 2009), FAB-MAP has shown a great potential to recognize places in routes of 1000km in length using only images.

For Cummins and Newman, the occurrence of the features is not independent from each other. The presence of an object in the scene usually makes the features appear in groups. Based on this assumption, FAB-MAP aims to model the co-occurrence of those appearances in a Bayesian Network in order to increase the robustness of the loop closure detection, even If there are some features not detected on the scene, given the presence of other features. In other words, using such probabilistic model, the method could better estimate the probability of an image to represent a local in the space, even if not all features used to describe that location are detected. Moreover, the co-occurrence probability measurement also decreases the robot's belief on a wrong feature detection, when performed. Also, FAB-MAP is based on the Markov Chain assumption, which means that the robot's pose belief at time k is dependent only on the belief at time $k-1$. This allows the use of a Recursive Bayesian Filter be performed online.

Computing the Bayesian Network model

In order to represent scenes in a probabilistic manner given the features detected, firstly a generative model of the observation need to be learnt. For that, FAB-MAP uses a training dataset to learn this probabilistic model at a preliminary offline phase. This model represents the co-occurrence dependency among the words which is used, in the online phase, to probabilistically identify the possible robot's pose. Let $Z = \{z_1, z_2, \dots ,\ z_{|v|-1}, z_{|v|}\}$ be the BoW's vocabulary with size $|v|$. Using the co-occurrence of the features in training dataset, FAB-MAP constructs a graph representation where an edge (z_i, z_j) express how much the occurrence of one variable predicts the occurrence of the other, as shown in Figure 2.3 (a). In order to build this graph, FAB-MAP uses the mutual information measurement $I(z_i, z_j)$ to express the probability of two words z_i and z_j appear together, which can be computed by:

$$I(z_i, z_j) = \sum_{z_i \in \Omega,\ z_j \in \Omega} p(z_i, z_j) \log \frac{p(z_i, z_j)}{p(z_i)p(z_j)}$$

where $\Omega = \{0,1\}$ is the binary variable that represents the presence or absence of a word. Thus, the mutual information $I(z_i, z_j)$ is zero if the variables are independent. Otherwise, the associated value to edge tends to increase.

Relying on a complete graph to compute a probabilistic model, could be computational expensive. Work with a complete graph implies to compute the associate probability to $\frac{|v|(|v|-1)}{2}$ edges. In addition, working with a visual information usually means have to deal with a high dimensional data. So, the use of a complete graph to represent a probabilistic model of visual information could lead the method to a problem with intractable size in the online phase. To deal with this, FAB-MAP approximates the graph distribution $Q(Z)$ to a tree-structured Bayesian network of distribution $P(Z)$,as shown in Figure 2.3 (b). The tree-structured Bayesian network needs to keep the main characteristics of the graph, like the strong relations between the features and the joint distribution, while reduces the computational cost associated to the problem. Then, the $P(Z)$ can be seen as the maximum spanning tree of $Q(Z)$ that minimizes the Kullback-Leibler divergence, computed by:

$$D_{KL}(Q,P) = \sum_{Z} Q(Z) \log \frac{Q(Z)}{P(Z)}$$

Chow and Liu algorithm (Chow; Liu, 1968) was chosen by FAB-MAP to compute the Bayesian Network tree. The Chow-Liu tree was selected by the authors due to its capability to deal with large vocabulary and its requirement of only the first order conditional probabilities, which can be estimated from the training dataset. Furthermore, for sparse cases, as visual-based scenarios, the computation of this tree can be accelerated by the algorithm described by Meilă (Meilă, 1999), which takes advantage of the graph sparsity to speed up the pairwise of the marginal probabilities.

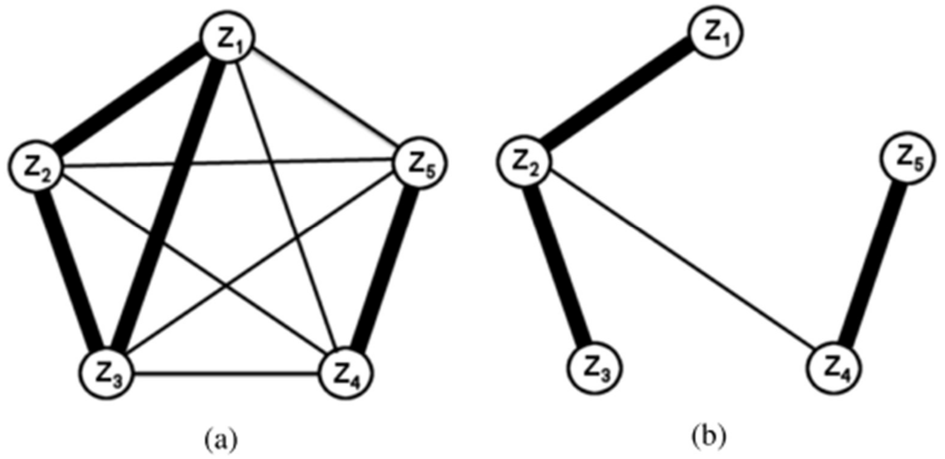


Figure 2.3 Bayesian Networks of the features. (a) The complete mutual information graph that encodes the distribution $Q(Z)$. (b) The spanning tree that maximizes the Kullback-Leibler divergence. Image extracted from (Cummins; Newman, 2007).

The use of a tree-structured approximation allows us to compute the joint probability of $P(Z)$ as a product of first-order conditionals, given by:

$$p(Z) = p(z_1, \dots, z_n) = p(z_r) \prod_{i=1}^{n} p\left(z_i \middle| z_{p_i}\right) \qquad 2.2$$

where z_{p_i} is parent node of z_i and z_r is the root variable of the three. Thus, the full distribution of $Q(Z)$ in Figure 2.3 (a):

$$p(Z) = p(z_1|z_2, z_3, z_4, z_5)p(z_2|z_3, z_4, z_5)p(z_3|z_4, z_5)p(z_4|z_5)p(z_5)$$

Can be approximated to $P(Z)$, in Figure 2.3 (b), with distribution computed by:

$$p(Z) \approx p(z_2)p(z_1|z_2)p(z_3|z_2)p(z_4|z_2)p(z_5|z_4)$$

The conditional probability $p\left(z_i \middle| z_j\right)$ required by Equation 2.2 can be obtained by computing the co-occurrence frequency of the involved words at the training phase. However, usually the training dataset does not totally represent the real world. To prevent $p(z_i|z_j)$ to be assigned to an unrealistic probability of 0 or 1, a pseudo-Bayesian p^* estimator should be used.

Estimating a location via Recursive Bayes

As a front-end to VisualSLAM, FAB-MAP is based on images to determine the robot's pose on the environment. Then, a scene gathered by the robot at time k is encoded as an observation set $\mathcal{Z}^k = \{z_1, z_2, \dots, z_{|v|-1}, z_{|v|}\}$, where z_i is a binary term indicating if the i^{th} word of the vocabulary is present on the respective image or not. Let $\mathcal{L}^k = \{L_1, \dots, L_{n_k}\}$ be a set of n_k disjoint locations at time k. A location L_i is described by the set of probabilities $\{p(e_1 = 1|L_i), \dots, p(e_{|v|} = 1|L_i)\}$, where $p(e_i = 1|L_i)$ encodes the probability of the i^{th} word of the vocabulary be present at location L_i. How $p(e_i = 1|L_i)$ belief is set and updated will be explained at section 0. Thus, estimating the conditional probability of a location L_i given the observation set $\mathcal{Z}^k$, can done by the recursive Bayesian filter:

$$p(L_i|\mathcal{Z}^k) = \frac{p(Z_k|L_i)p(L_i|\mathcal{Z}^{k-1})}{p(Z_k|\mathcal{Z}^{k-1})} \tag{2.3}$$

where Z_k is the set of features presented on the environment, $p(L_i|\mathcal{Z}^{k-1})$ is the prior belief about the robot's location, $p(Z_k|\mathcal{Z}^{k-1})$ is a normalization term of the observation model and $p(Z_k|L_i)$ is the observation likelihood that can be expanded by the Chow-Liu tree:

$$p(L_i|\mathcal{Z}^k) \approx \frac{p(L_i|\mathcal{Z}^{k-1})p(z_r|L_i)\prod_{q=1}^{|v|} p\left(z_q|z_{p_q}, L_i\right)}{p(Z_k|\mathcal{Z}^{k-1})}$$

In order to go forward on this expansion, a probabilistic model of the feature detector algorithm needs to be done. Influenced by light changing or perspective transformations, sometimes a feature detector could fail in detecting a word. Therefore, the probability of a word not be observed by FAB-MAP given it exist, $p(z_i = 0|e_i = 1)$, and the probability of a word be declared as present on the scene given it not exist,

$p(z_i = 1|e_i = 0)$, need to considered here. To build a probabilistic model of the feature detection algorithm, a ground truth of which words are presented in each image from the training dataset needs to be provided, what is difficult to have in a real scenario. Therefore, these two terms, $p(z_i = 0|e_i = 1)$ and $p(z_i = 1|e_i = 0)$, are pre-defined parameters for FAB-MAP. Then, the probability $p\left(z_q \middle| z_{p_q}, L_i\right)$ can be obtained from:

$$\sum_{s_{e_q} \in \Omega} p\left(e_q = s_{e_q} \middle| z_q, z_{p_q}, L_i\right) = 1$$

$$\sum_{s_{e_q} \in \Omega} \frac{p\left(z_q \middle| e_q = s_{e_q}, z_{p_q}, L_i\right) p\left(e_q = s_{e_q} \middle| z_{p_q}, L_i\right)}{p\left(z_q \middle| z_{p_q}, L_i\right)} = 1$$

$$p\left(z_q \middle| z_{p_q}, L_i\right) = \sum_{s_{e_q} \in \Omega} p\left(z_q \middle| e_q = s_{e_q}, z_{p_q}, L_i\right) p\left(e_q = s_{e_q} \middle| z_{p_q}, L_i\right)$$

Assuming that the detection errors are independent of the location L_i and the $p(e_j)$ is independent of the observation z_i, for all $i \neq j$, the probability $p\left(z_q \middle| z_{p_q}, L_i\right)$ can be simplified to:

$$p\left(z_q \middle| z_{p_q}, L_i\right) = \sum_{s_{e_q} \in \Omega} p\left(z_q \middle| e_q = s_{e_q}, z_{p_q}\right) p\left(e_q = s_{e_q} \middle| L_i\right) \qquad 2.4$$

The probability of $p\left(z_q \middle| e_q, z_{p_q}\right)$ can be computed by applying the Bayes theorem:

$$p\left(z_q \middle| e_q, z_{p_q}\right) = \frac{p\left(e_q \middle| z_q, z_{p_q}\right) p\left(z_q \middle| z_{p_q}\right)}{p\left(e_q \middle| z_{p_q}\right)} \qquad 2.5$$

Where $p(e_q|z_p)$ can be expanded by the law of total probability to:

$$\sum_{s_{z_q} \in \Omega} p\left(z_q = s_{z_q} \middle| e_q, z_p\right) = 1$$

$$\sum_{s_{z_q} \in \Omega} \frac{p\left(e_q \middle| z_q = s_{z_q}, z_p\right) p\left(z_q = s_{z_q} \middle| z_p\right)}{p(e_q|z_p)} = 1$$

$$p(e_q|z_p) = \sum_{s_{z_q} \in \Omega} p\left(e_q \middle| z_q = s_{z_q}, z_p\right) p\left(z_q = s_{z_q} \middle| z_p\right)$$

Finally, assuming again that $p(e_q|z_q, z_p) \cong p(e_q|z_q)$, Equation 2.5 become:

$$p\left(z_q \middle| e_q, z_{p_q}\right) = \frac{p\left(e_q \middle| z_q, z_{p_q}\right) p\left(z_q \middle| z_{p_q}\right)}{\sum_{s_{z_q} \in \Omega} p\left(e_q \middle| z_q = s_{z_q}, z_{p_q}\right) p\left(z_q = s_{z_q} \middle| z_{p_q}\right)}$$

$$p\left(z_q \middle| e_q, z_{p_q}\right) = \left(\frac{\sum_{s_{z_q} \in \Omega} p\left(e_q \middle| z_q = s_{z_q}, z_{p_q}\right) p\left(z_q = s_{z_q} \middle| z_{p_q}\right)}{p\left(e_q \middle| z_q, z_{p_q}\right) p\left(z_q \middle| z_{p_q}\right)}\right)^{-1}$$

$$p\left(z_q \middle| e_q, z_{p_q}\right) = \left(1 + \frac{p\left(e_q \middle| \overline{z_q}, z_{p_q}\right) p\left(\overline{z_q} \middle| z_{p_q}\right)}{p\left(e_q \middle| z_q, z_{p_q}\right) p\left(z_q \middle| z_{p_q}\right)}\right)^{-1}$$

$$p\left(z_q \middle| e_q, z_{p_q}\right) = \left(1 + \frac{p(z_q) p(\overline{z_q}|e_q) p\left(\overline{z_q} \middle| z_{p_q}\right)}{p(\overline{z_q}) p(z_q|e_q) p\left(z_q \middle| z_{p_q}\right)}\right)^{-1} \qquad 2.6$$

Note that all probabilities at Equation 2.6 can be obtained from the offline phase or from the pre-defined parameters. So, they are computed once at the preliminary phase and then reused as requested. Looking back at the Recursive Bayes Filter, Equation 2.3, the only term that is needed at online phase to compute the observation likelihood $p(Z_k|L_i)$ is the $p\left(e_q = s_{e_q} \middle| L_i\right)$, seen at Equation 2.4. How this probability is estimated according to the robot's readings will be explained at the next section.

Representing and updating a location

FAB-MAP represents a location in the environment by a set of probabilities $\{p(e_1 = 1|L_j), \dots, p(e_{|v|} = 1|L_j)\}$, as described previously in this section. Initially, as any information cannot be obtained from the location, all probabilities $p(e_i = 1|L_j)$ are initialized to the marginal probability of $p(e_i = 1)$, which is obtained by the training dataset. When an observation related to a specific location is obtained, the method updates its belief by the Recursive Bayes Filter given by:

$$p(e_i = 1|L_j, \mathcal{Z}^k) = \frac{p(Z_k|e_i, L_j)p(e_i|L_j, \mathcal{Z}^{k-1})}{p(Z_k|L_j)}$$

where $p(e_i|L_j, \mathcal{Z}^{k-1})$ is the priori belief of the i^{th} word of vocabulary exists at location L_j, $p(Z_k|e_i, L_j)$ is the likelihood of a word generating an observation on that location and $p(Z_k|L_j)$ is the normalization factor. The last term can be compute by the law of total probability:

$$\sum_{s_e \in \Omega} p(e_i = s_e|L_j, \mathcal{Z}^k) = 1$$

$$\sum_{s_e \in \Omega} \frac{p(Z_k|e_i = s_e, L_j, \mathcal{Z}^{k-1})p(e_i = s_e|L_j, \mathcal{Z}^{k-1})}{p(Z_k|L_j)} = 1$$

$$p(Z_k|L_j) = \sum_{s_e \in \Omega} p(Z_k|e_i = s_e, L_j, \mathcal{Z}^{k-1})p(e_i = s_e|L_j, \mathcal{Z}^{k-1}) \tag{2.7}$$

Applying Equation 2.7 in Equation **Error! Reference source not found.**:

$$p(e_i = 1|L_j, \mathcal{Z}^k) = \frac{p(Z_k|e_i, L_j)p(e_i|L_j, \mathcal{Z}^{k-1})}{\sum_{s_e \in \Omega} p(Z_k|e_i = s_e, L_j, \mathcal{Z}^{k-1})p(e_i = s_e|L_j, \mathcal{Z}^{k-1})} \tag{2.8}$$

Finally, assuming that the detector behavior is independent of location and $p(e_i)$ is independent of z_j, for all $i \neq j$, Equation 2.8 can be simplified to:

$$p(e_i = 1 | L_j, \mathcal{Z}^k) = \frac{p(z_i|e_i)p(e_i|L_j, \mathcal{Z}^{k-1})}{\sum_{s_e \in \Omega} p(z_i|e_i = s_e)p(e_i = s_e|L_j, \mathcal{Z}^{k-1})}$$

It is important to notice that any inference about the existence of a word i is made based on the observations of other words at same location. Beside such inference could be performed using the Chow-Liu tree, FAB-MAP as presented by Cummins (Cummins; Newman, 2007, 2008) does not make use of that information. Also, this updated is made up on a given location which the method believes that the observations are related. To solve this data association problem, FAB-MAP chooses the location with maximum likelihood (ML). However, as a SLAM technique, FAB-MAP should not be limited to the mapped places. If the ML decision indicates an unmapped place to be the most likely origin of the observations, then a new location is initialized with all probabilities $p(e_i = 1|L_j) = p(e_i)$ and then the update procedure is performed.

New place or old place

If FAB-MAP were limited to deal only with the localization problem, the normalization term of the Equation 2.3 could be easily computed by the total probability of all locations learnt by the method. However, as a SLAM technique, this approach should consider that the sensor information gathered by the robot could be observed from an unmapped location. Thus, FAB-MAP considers that the world is divide in two sets: the mapped region M and the unmapped region $\overline{M}$. Then, we can obtain the normalization term by the law of total probability:

$$\sum_{m \in M} p(L_m|\mathcal{Z}^k) + \sum_{n \in \overline{M}} p(L_n|\mathcal{Z}^k) = 1$$

$$\sum_{m \in M} \frac{p(Z_k|L_m)p(L_m|\mathcal{Z}^{k-1})}{p(Z_k|\mathcal{Z}^{k-1})} + \sum_{n \in \overline{M}} \frac{p(Z_k|L_n)p(L_n|\mathcal{Z}^{k-1})}{p(Z_k|\mathcal{Z}^{k-1})} = 1$$

$$p\left(Z_k \middle| \mathcal{Z}^{k-1}\right) = \sum_{m \in M} p(Z_k|L_m)p\left(L_m \middle| \mathcal{Z}^{k-1}\right) + \sum_{n \in \overline{M}} p(Z_k|L_n)p\left(L_n \middle| \mathcal{Z}^{k-1}\right)$$

While the first term can be computed from the mapped locations, the second part cannot be directly calculated from the available data. So, an approximation of the all unknown locations is performed by a sampling procedure. This sampling procedure creates random place models by sampling possible features observations, according to the relations encoded on the Chow-Liu tree. The last parameter to be user-defined is the overall probability of an image be originated from a new location. Then, the priori beliefs $p\left(L_n \middle| \mathcal{Z}^{k-1}\right)$ are uniformly distributed among all these new locations sampled by the method.

Although FAB-MAP has shown good results in a set of experiments at daylight by using all available information in a complete probabilistic framework, the fact that it still based on the Markovian assumption could represent a drawback. For the Markovian Chain assumption, the robot's pose x_t is strongly dependent of the last pose estimation x_{t-1}. Although this assumption makes sense when we are thinking about the robot's navigation point of view, a false-positive match in the localization estimation procedure could reflect directly on the future estimations and be propagated indefinably, until a strong loop closure detection overcomes the false influence. In next section, we present a method that aims to be robust to this perceptual light changing while performs a global localization inside a VisualSLAM approach is presented.

2.3.2 SeqSLAM

The FAB-MAP main weakness is to ignore spatial or temporal correlations between the collected information about the world. Observing this fact, Milford's SeqSLAM (Milford et al., 2013; Milford; Wyeth, 2012; Pepperell et al., 2014) introduced a front-end method for VisualSLAM that exploits these relations in search for loop closures across different times of the day, disturbances of illumination, seasons or weather conditions. This method declares that one of its key innovation is to consider the loop closure problem not as a search problem for a single best match correspondence. Instead of that, the algorithm searches for a sequence of high similarity correspondence among two sets of images, which is more robust to false positives than searching for a single pair of images (Milford; Wyeth, 2012).

This robustness allows SeqSLAM to reduce the amount of information needed to describe each scene without sacrifice its precision. The insight of low information data required by the method, carry another relevant goal. With less information to be stored, this approach is able to represent very large environments through a slim map. To illustrate how much data can be compressed, Figure 2.4 (a) shows seven datasets that totalize 131 km of road travel and 1160 meters of indoor travel. They were encoded in a map representation of size comparable to a single camera image of 2 megapixels, Figure 2.4 (b), extracted from one of those datasets (Milford, 2013).

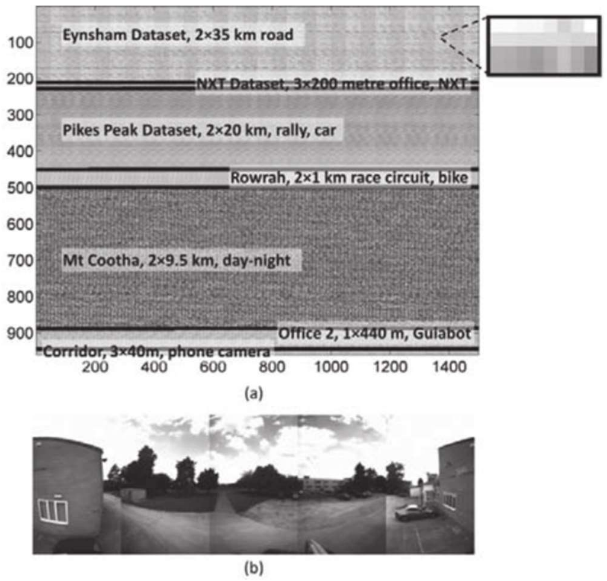


Figure 2.4 Image extracted from (Milford, 2013). (a) SeqSLAM's place recognition map representing 7 datasets that sum 131 km of road travel and 1160 meters of indoor travel, into an image of about 2 megapixels. (b) A single image of 2 megapixels extracted from one of those datasets.

Before computing a similarity estimation among two set of images, SeqSLAM firstly processes all collected images converting them to grayscale and reducing them to a thumbnail of dimensions $R_x \times R_y$. For each acquired frame by the robot, the method associates a vector $D = [d_1, d_2, \dots, d_{n-1}, d_n]$ that encodes the difference between the current image I_c to all n images already learnt and recorded into a database $\mathcal{I} = \{I_1, I_2, \dots, I_{n-1}, I_n\}$. To compute a similarity measurement d_i among two images, a Sum of Absolute Differences (SAD) approach is used on the reduced grayscale representations of the scenes. To highlight the image details to the SAD algorithm, SeqSLAM algorithm divides the camera images into a grid-like structure with enhanced patches of size R_p. Let $P_{x,y}$ be a patch at column x and line y of the grid structure, where $0 < x \leq \left\lfloor \frac{R_x}{R_p} \right\rfloor$ and $0 < y \leq \left\lfloor \frac{R_y}{R_p} \right\rfloor$. In order to increase the contrast and highlight the features of the images, each patch $P_{x,y}$ is normalized. Then, the difference d_i is computed by:

$$d_i = \frac{1}{R_x R_y} \sum_{x=1}^{R_x} \sum_{y=1}^{R_y} \left| p_{x,y}^{i} - p_{x,y} \right|$$

where $p_{x,y}$ is the pixel of the current image and $p_{x,y}^{a}$ is the pixel value of an image I_a at column x and row y, in the coordinate system of the image.

To improve the sequence match process, a local contrast enhancement is performed over the vector D to produce an enhanced image difference vector $\widehat{D} = \{\hat{d}_1, \hat{d}_2, \dots, \hat{d}_{n-1}, \hat{d}_n\}$, as can seen in Figure 2.5, by the equation:

$$\hat{d}_i = \frac{d_i - \overline{D}}{\sigma_D}$$

where $\overline{D}$ is the mean and σ_D is the standard deviation of the vector D.

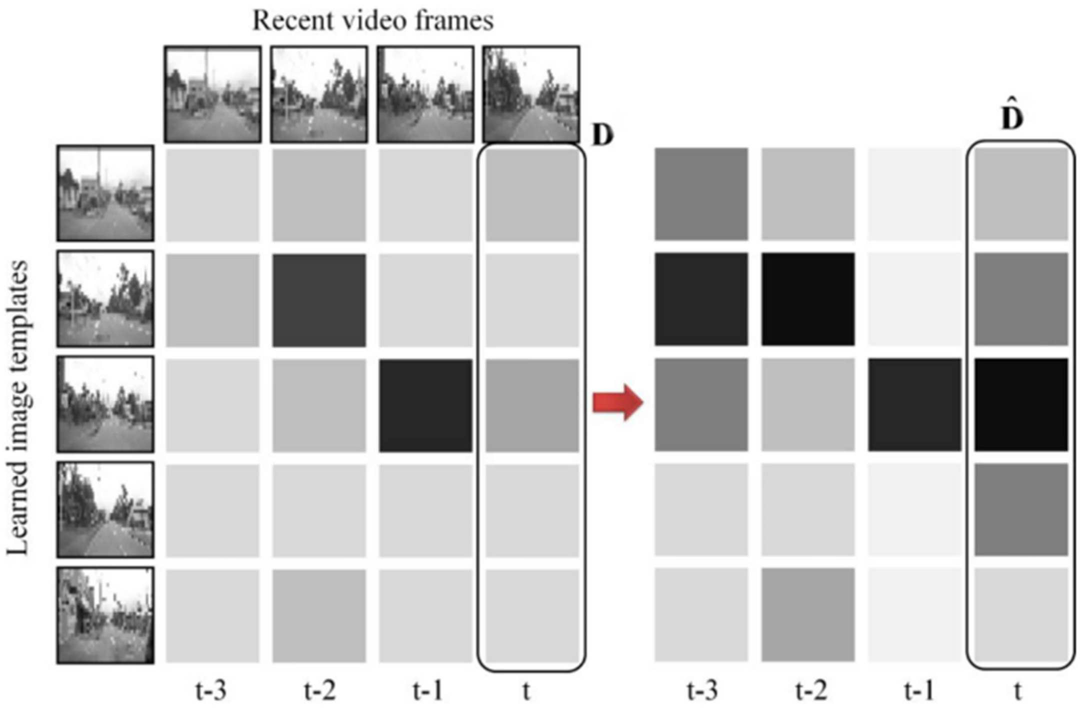


Figure 2.5 Contrast enhancement of the difference vector D to vector $\widehat{D}$ to highlight the strongly matchings. Dark cells represent stronger similarity and white cells represent weak relation between images. Image extracted from (Milford; Wyeth, 2012).

The goal of SeqSLAM is to localize the robot using a sequence of images, which is more robust to false positive. Hence, to recognize familiar places, this method keeps an updated matrix M of the normalized difference vectors associated to the last images captured by the robot. Let T be the current time and Δ_s determine how back in time a search needs to be performed to efficiently localize the robot. Therefore, the difference matrix M can be described as:

$$M = \left[\widehat{D}^{T-\Delta_s+1}, \widehat{D}^{T-\Delta_s+2}, \dots, \widehat{D}^{T}\right]$$

where $\widehat{D}^t$ is the enhancement difference vector of the image I_t computed in relate to all images from the database.

Determining a value to the parameter Δ_s is not a trivial task and usually depends on the environment structure. Searching for long sequences improves the matching process in environments with repetitive patterns by offering more images to be compared and, consequently, disambiguating the robot's location. However, it reduces the probability of detecting small loops. When the difference between Δ_s and the number of images that form a loop is high, more images unrelated to the loop closure are evaluated to fill the sequence length requirement. On other hand, seeking for small sequences enables the robot to identify short loop closures on its path. But it could also increase the false positive response of the method. With less images to compare, this approach become less robust to outliers. Therefore, just few wrong associating high similarities are necessary to lead SeqSLAM to a wrong result.

To determine which sequence from the matrix M is close to the last Δ_s images captured by the robot, SeqSLAM projects several possible matching sequences over M. Milford's method assumes that the images are learnt in a fixed FPS rate and the robot is crossing the environment with an approximated constant velocity. Thus, all possible sequences projected are linear, simulating different traversing velocities from $V_{\min}$ to $V_{\max}$ in steps of V_{step}, starting from each image template of $\widehat{D}^{T-\Delta_s+1}$. An example is shown in Figure 2.6. A difference score S, associated to each possible sequence, is computed by:

$$S = \sum_{t=T-\Delta_s+1}^{T} \hat{d}_k^t$$

where k is the element of vector $\widehat{D}^t$ that the associated trajectory is passing through, at time t, given by:

$$k = s + \lfloor V(t - T + \Delta_s - 1) \rfloor$$

where s is the template number where the trajectory started and V is the velocity of the current trajectory.

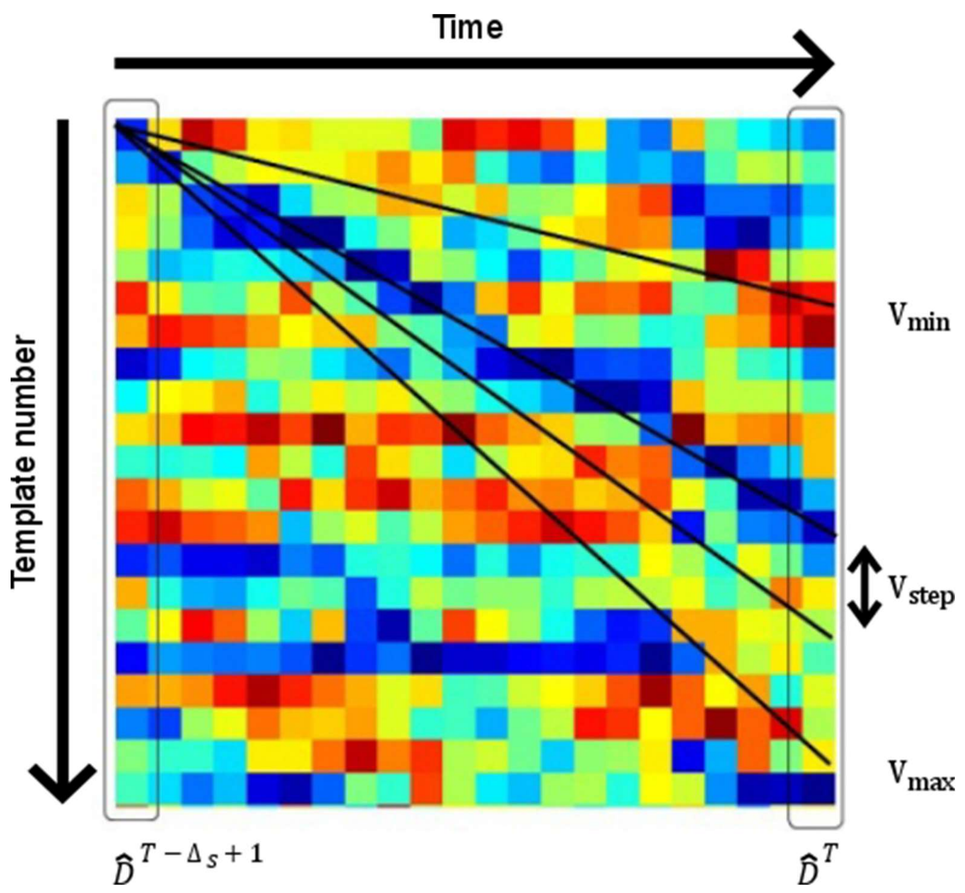


Figure 2.6 Searching for a sequence of images with smallest difference score. For clarity, only two projected trajectories over the matrix M is presented. Image extracted from (Milford; Wyeth, 2012).

Finally, SeqSLAM indicates if the current area that the robot is traveling is a revisited area and, if it is the case, which evaluated sequence seems to be a true matching for the loop closure. Directly choosing the sequence with the smallest score S could make the method to return many false positives. If none sequence of images is similar to the last images captured by the robot, all scores S will indicate a high degree of difference. So, choosing the smallest score S will make the method to select a sequence that does not in fact represent a loop closure. Following the same idea, if more than one sequence has a small difference score, SeqSLAM cannot be totally sure about which sequence better represents the match.

Let $\mathcal{S}$ be a vector of all smallest difference scores S associated to each start element in $\hat{D}^{T-\Delta_S+1}$. To give a certainty index, SeqSLAM computes the factor μ between the smallest difference score of $\mathcal{S}$ and the second smaller difference score outside of a window R_{window}, as shown in Figure 2.7. If this μ is smaller than a threshold μ_{min}, than the sequence is deemed to be a match.

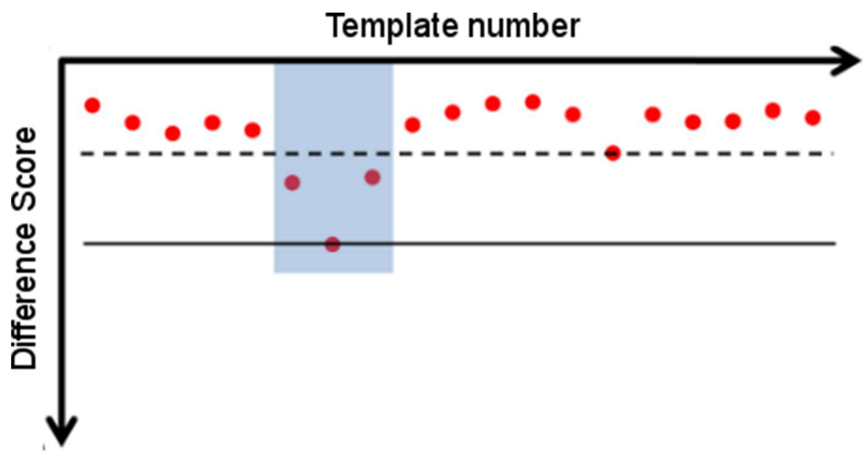


Figure 2.7 Red points represent the vector S indexed by the relative template number in $\widehat{D}^{T-\Delta_s+1}$. After the smallest element of S is selected, the second smaller element outside the window R_{window}, blue region, is searched. Image extracted from (Milford; Wyeth, 2012).

SeqSLAM have been demonstrating good results in a range of scenarios not tractable by others state-of-art algorithms. Milford et. al. (Milford et al., 2013) demonstrated that their method could localize the robot at night from a map gathered at daylight by using a camera in maximum exposure duration mode. Although the long exposure mode generates extremely blurred images, SeqSLAM was robust enough to identify similar sequences of images of the same location. In another experiment, Neubert (Neubert; Sunderhauf; Protzel, 2013) demonstrated the long-term SLAM capabilities of SeqSLAM evaluating this method in a dataset containing images in all four seasons of the year. SeqSLAM was able to obtain good results localizing sequences of images from one season using a map representation built in any other season of the year.

Although some assumptions of this method are not so easily found in many real situations, some improvements to SeqSLAM have been developed in the last years. Hansen (Hansen; Browning, 2014) showed how to relax the constant velocity assumption by using an approach based on Dynamic Time Warping techniques, commonly found on speech recognition researches, to find the optimal alignment between two sequences. In the same year, Naseer et. al. (Naseer; Spinello; Burgard;

Stachniss, 2014) formulated the sequence matching process as a minimum cost flow problem in a graph built over the matrix M, which allowed to deal with loop closures partially occluded. On the following year, the same authors decided to use deep convolutional neural networks to extract an image descriptor, instead of use a patch-normalized thumbnail as in SeqSLAM (Naseer; Ruhnke; Stachniss; Spinello; Burgard, 2015).

Looking back to Milford's researches, a successor of SeqSLAM called SMART was introduced in 2014 (Pepperell et al., 2014). This proposed method uses a source of odometry to learn images equally spaced on the environment to mitigate the constant velocity assumption. In addition to that, it uses a sky detector component to eliminate the weather condition variable from the SAD comparison. One year later, the same authors updated their technique to use depth information estimated from the scenes to auto scale images collected near each other, with the purpose of increase their similarity (Pepperell; Corke; Milford, 2015).

Although SeqSLAM have been demonstrated its strength at scenarios that suffer with high degree of illumination changing, the SAD comparison represents a weak point of this method. A comparison by patches, as SAD or the Sum of Squared Differences (SSD), are dependent of camera's point of view. Depend on collect data from the same point of view to have a loop closure detection is not ideal to many scenarios in robotics. For instance, a robot that is exploring an unknown area, without a fixed road to follow, should be able to self-localize even if it is not following the exactly same path as before.

3 S4FE

Besides the goods results obtained by the global localization process of SeqSLAM, Milford's method defends the use of SAD technique to compare images. The use of such approach have demonstrated to be robust to light changing scenarios, but makes the method dependent to the robot's point of view. In this work, we present our proposal method named Sequential Feature Frequency Filter – Front-End for SLAM (S4FE). S4FE uses a combination of BoW (Sivic; Zisserman, 2003), TF-IDF (Nakashima; Nakamura, n.d.) and Mean Shift Algorithm (MSA) (Fukunaga; Hostetler, 1975) to detect loop closure during the robot navigation in an outdoor environment. This combination aims to improve the image sequence matching through the use of features descriptors which are robust to illumination differences and affine transformations while performs an efficient global localization tracking method that focuses precisely in the correct matchings that appears in the similarity matrix.

This subsection is divided as follows. First, we will formalize the loop closure detection using sequences of images. Then, S4FE will be presented divided into two parts: the preprocessing and processing phases. Preprocessing phase constructs the BoW vocabulary from the training video and estimates the occurrence frequency of the words of the vocabulary (IDF method) from this video. Processing phase identifies words frequency (TF method) for each video image, constructs the similarity matrix and tries to find loops using MSA.

3.1 Loop Closure using Image Sequences

The robot captures images sequentially at a constant time interval during its motion. This image sequence is described by $\mathcal{I}_{1:T} = \{I_1, \ldots, I_{T-1}, I_T,\}$, where I_k represents the image captured at time instant $I \ \leq k \leq T$; and T corresponds to the sequence size. In general, a loop closure strategy tries to find correspondence among images spaced temporally to identify if the robot is revisiting a specific environment area.

Without loss of generality, consider two image sub-sequences $\mathcal{I}^a = \mathcal{I}_{p:q}$ and $\mathcal{I}^b = \mathcal{I}_{r:u}$ from $\mathcal{I}$ that may contain images captured in a same environment area, with $1 \leq p \leq q < r \leq u \leq T$. After detecting the first correspondence between one image from each sequence, (I_i^a, I_j^b), it is plausible to assume that exists a set of matches

$$\mathcal{M} = \{I^a_{i+\Delta_m}, I^b_{j+\Delta_n} | \Delta_m > 0 \text{ and } \Delta_n > 0\}$$

If $|\mathcal{M}| = 0$ then probably the correspondence (I^a_i, I^b_j) is a false positive because there are no other evidence that a loop has been found. The larger the set $\mathcal{M}$, the higher the probability that a loop has been detected. As a loop should be detected as soon as possible, then $\mathcal{I}^b = \mathcal{I}_{T-\Delta_s:T}$ should contain the most recent images, with a sequence length $\Delta_s > 0$, while $\mathcal{I}^a = \mathcal{I}_{p:q}$ should contain old images sequence, with $q < \Delta_s$.

3.2 Preprocessing Phase

As FAB-MAP, our approach also relies on a training dataset to learn an appearance model of the world. Given a set of image samples of an area with similar structures to the environment that the robot will pass through, our proposal uses an algorithm to extract the most relevant environment features and to cluster them into *words*, using a visual BoW scheme. These words comprise a reduced discrete space, which gives us the advantage to be more robust to illumination variations. Each word has an assigned weight that is used to reduce the influence of not relevant words in the loop detection process in order to filter false positives.

All features are extracted using a combination of STAR detector (Agrawal; Konolige; Blas, 2008) and SURF descriptors (Bay et al., 2008), which have obtained good results during the preliminary testing stage of the method. Initially, STAR is used to find relevant points in images that may represent distinguishable features. After, a SURF descriptor is computed for each point detected by STAR to guarantee that the features associated with that point are recognized under different lighting conditions and affine transformations. The descriptors extracted are clustered using the clustering scheme proposed by Teynor et al (Teynor; Burkhardt, 2007), which is able to deal with a high number of features. Each cluster represents a visual word of BoW.

Some words may appear repeatedly along the environment without increasing the perceptual information that distinguishes the current scene from the others. For instance, consider a robot navigating in an indoor environment where all walls have the same specific pattern. The words that describe this common pattern will not be relevant to distinguish a particular scene from the others. Moreover, two distinct scenes that contain a high occurrence of this pattern might be considered highly similar and, consequently, may badly influence the loop detection process. Thereby, words that are associated to not frequent patterns are better to distinguish different scenes and identify correctly similar ones.

To deal with this situation, the words are weighted using a TF-IDF statistical model (Sparck Jones, 1972) commonly used in text mining. The weight corresponds to the Inverse Document Frequency (IDF) of that word in a training image sequence. That is, for a word v, its weight w_v in a training image sequence $\mathcal{T}$ is:

$$w_v = \log\left(\frac{|\mathcal{T}|}{|\{I \in \mathcal{T} | v \in_* I\}|}\right)$$

where $v \in_* I$ means that word v appears in image I.

3.3 Processing

After classifying the visual features into words and determining the most relevant words that could describe each scene, a matching strategy can be performed. Each new frame captured by the robot is represented by a weighted histogram of words $\mathcal{H} = \{H_v | v \in \mathcal{V}\}$, where $\mathcal{V}$ is the BoW vocabulary and $0 \leq H_v \leq 1$ is the frequency of word v in that particular frame weighted by its IDF measure. This histogram is calculated as follows. The algorithm extracts the features from a particular image I_j, in a similar way, as presented before in the preprocessing phase, and classifies each one into one of the words contained in $\mathcal{V}$. For each word v that appears in that frame I_j, its corresponding entry in $\mathcal{H}_j$ is

$$H_v^j = \rho w_v f(v, I_j)$$

where $\rho = \left(max_{p \in_* I_j} f(p, I_j)\right)^{-1}$ and $f(v, I_j)$ counts the number of occurrences of word v in the image I_j.

Using this information is possible to compute the similarity matrix $\mathcal{S} = \{S_{m,n}\}$ where $S_{m,n}$ is the similarity among two images I_m and I_n defined by

$$S_{m,n} = \sum_{v \in \mathcal{V}} H_v^m H_v^n$$

Once the similarity matrix is built and normalized, for every possible sequence match we calculate a disorder measurement defined by

$$\mathrm{D}_{r_1:r_2;c_1:c_2} = \phi \sum_{i=r_1}^{r_2} \sum_{j=c_1}^{c_2} S_{i,j}$$

where $\phi = (\Delta_s^2)^{-1}$, $|r_1 - r_2 + 1| = |c_1 - c_2 + 1| = \Delta_s$, to discard regions in this matrix with indistinguishable patterns and focus only on relevant ones. The probability to have a loop in a given sequence increases insofar as $\mathcal{D}_{:,:;:,:} \to 1$. Therefore, we search for loop closures only in sequences with disorder measurements bigger than a given threshold $\mu_{\mathcal{D}}$ defined *a priori*. If this $\mathcal{D} \geq \mu_{\mathcal{D}}$, then we search for loops in the similarity matrix using a tracking algorithm.

In real situations, a loop will not correspond exactly to a line pattern in the similarity matrix. However, this pattern will be the first guess to our tracking algorithm in a similar way as SeqSLAM does. Thus, a preliminary estimation of the sequence match inside the squared window is an array of similarity matrix points $\mathcal{P} = [P_1, P_2, \dots, P_{\Delta_s}]$ aligned in at an orientation of $\frac{\pi}{4}$. Each point $P_i = (m, n)$ corresponds to a match pair (I_m^a, I_n^b). For all points $P_i \in \mathcal{P}$ a refined position is reached displacing the respective match pair to a local mode present in the matrix S. The use of a MSA technique aids S4FE to converge the estimated path followed by the robot to a local consensus observed at matrix S. Given a window centered at a point P_i, a displacement δ_i can be computed by:

$$\delta_i = \sum_{x \in \mathcal{W}} \sum_{y \in \mathcal{W}} x S_{m+x,n+y} \left(\sum_{x \in \mathcal{W}} \sum_{y \in \mathcal{W}} S_{m+x,n+y} \right)^{-1} \quad \forall \ P_i \in \mathcal{P}$$

where $\mathcal{W} = \{-\Delta_{ms}, \dots, 0, \dots, \Delta_{ms}\}$. The respective match pair is updated, until $\delta_i = 0$, by

$$P_i \leftarrow (m, |n + \delta_i|)$$

When $\mathcal{P}$ converges, it will represent a most likely alignment of the two sequences. Then the pair indicated by the middle point of $\mathcal{P}$ is chosen as a loop closure match. Different alignments of $\mathcal{P}$ over the matrix S will represent different loop closure points. In some of them we are more confident that the middle point in fact represents a loop closure than in others, due the respective similarities encoded in S. Therefore, we express a matching confidence level η as

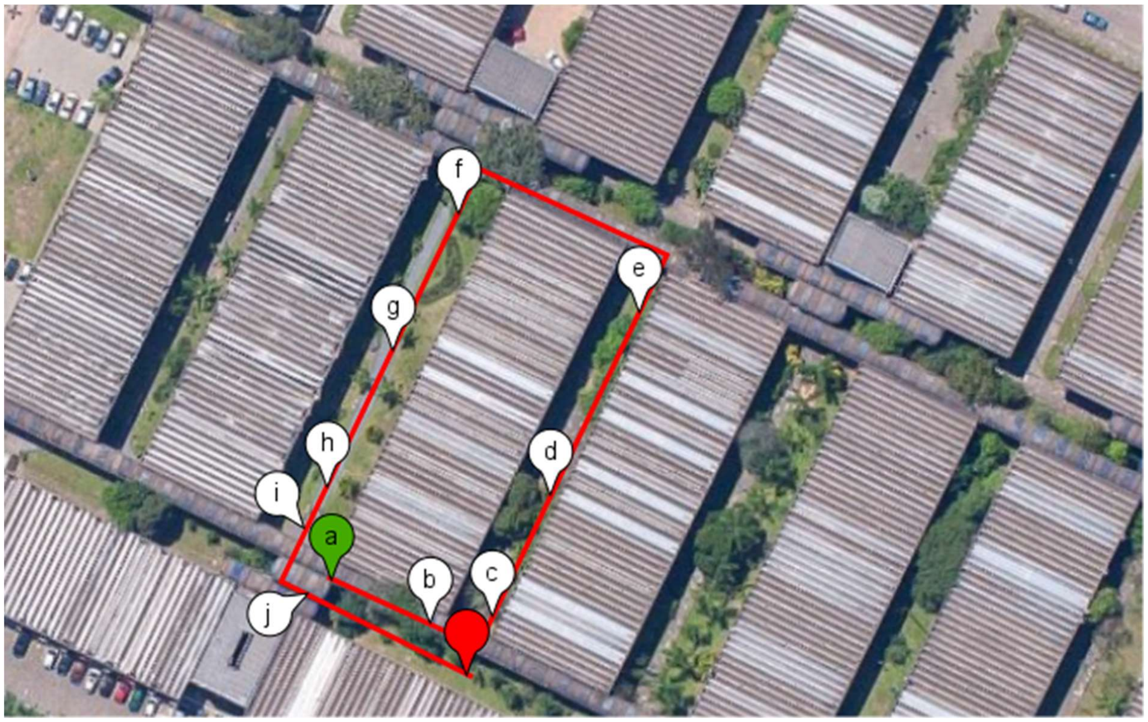
$$\eta = \Delta_s^{-1} \sum_{P_i \in \mathcal{P}} S_{P_i}$$

This value represents a loop closure insofar as $\eta_{.:.;.:.} \to 1$. Thus, only pairs with a η above a threshold μ_η are then returned as a valid matching by S4FE.

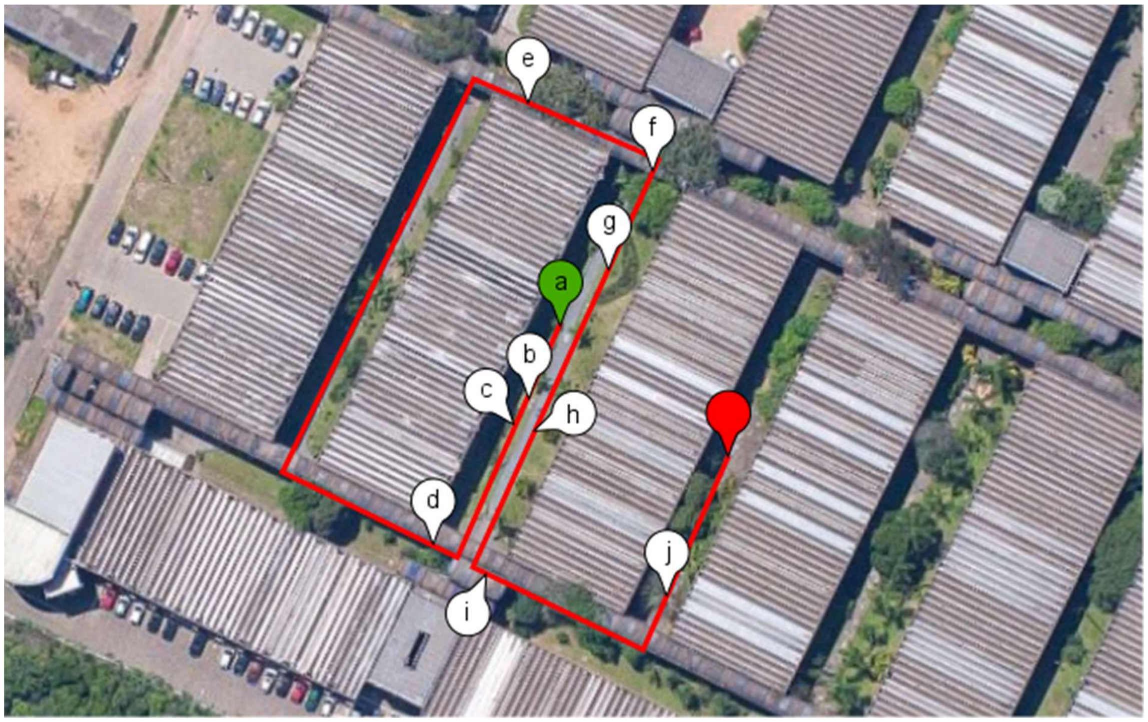
4 TEST ENVIRONMENTS

This section presents the challenges introduced by each one of our test environments with all video properties of the gathered data. After, we discuss the results highlighting the main differences between FAB-MAP, SeqSLAM and S4FE. At the end, we compare the methods showing the benefits introduced by our proposal. Two datasets will be presented. The first one was gathered using a handheld camera and the second one, using a Parrot ArDrone 2.0 robot. Both were obtained in the same outdoor environment at our university in Brazil. This environment is GPS-denied, therefore, we can count only on images to detect loop closure. The ground truth of the experiments was produced by frame-to-frame visual correspondence. Figure 4.1 shows an aerial image of the testing area along with the paths followed by the handheld camera (a) and the ArDrone (b).

Using a handheld camera, we recorded a video at resolution of 1920×1080 with 24 FPS. To improve the algorithm performance and get more distinguishable points of view, we first resampled the frames of this video producing a new one with 10 FPS. Each frame was converted to grayscale and reduced to 960×540 using a 3×3 gaussian kernel to prevent alias. Finally, the histogram of all frames were equalized to minimize the influence of lighting effects along the path. Some samples of the resulting images can be seen in Figure 4.2. As this dataset was recorded without any abrupt camera rotation or displacement, the resulted trajectory is smooth along the path, which does not offer additional challenges to the image matching process.



(a)



(b)

Figure 4.1 Trajectories of handheld camera (a) and Parrot ArDrone 2.0 (b) datasets. The green mark represents the start of the trajectory, while the red mark represents the end. The samples from Figure 4.2 and Figure 4.3 are labelled in (a) and (b), respectively. Image generated by the author.

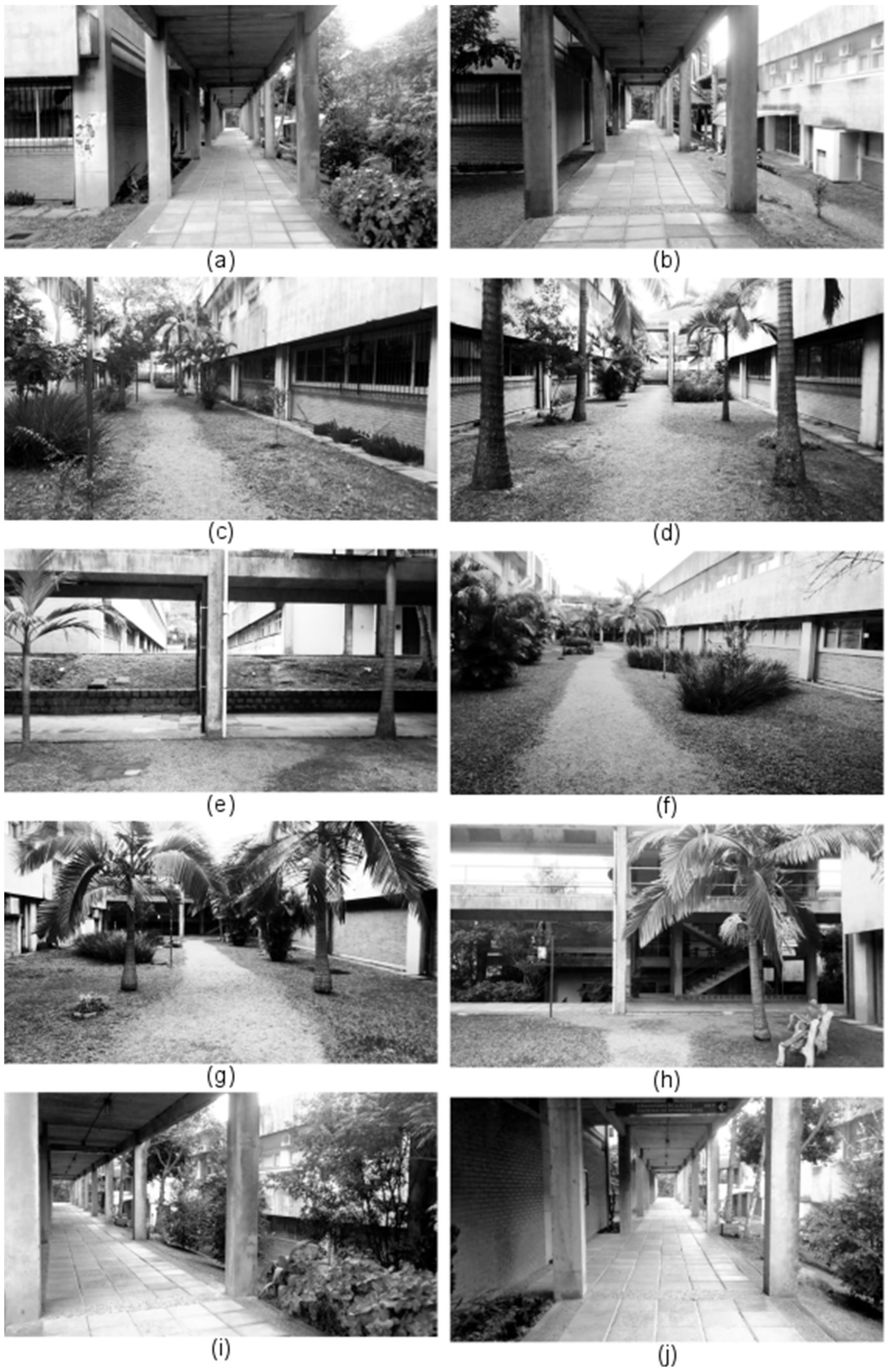


Figure 4.2 Samples of the images registered by the handheld camera. Image generated by the author.

Using the Parrot ArDrone 2.0, we recorded a video at resolution of 1280×720 with 30 FPS. Again, we first resampled the video to produce a new one with 10 FPS. Then, each frame was converted to grayscale and reduced to 640×320 using a 3×3 gaussian kernel to prevent alias. Lastly, the histogram of all frames were equalized. Some samples of the processed images and the challenges offered by the aerial robot can be seen in Figure 4.3. As a flying robot is more susceptible to wind bursts than a ground robot, more drifts and brusque 3D rotations happened in this experiment, as demonstrated in Figure 4.3 (b), Figure 4.3.(c) and Figure 4.3.(d). These disturbances interfered on the visual information capture, resulting in a more challenging visual recognition scenario. For instance, as can be seen in Figure 4.3 (i) and Figure 4.3 (j), these abrupt rotations resulted in blurred images and, also due to the low quality of the Parrot ArDrone 2.0 camera, in an illumination problem.

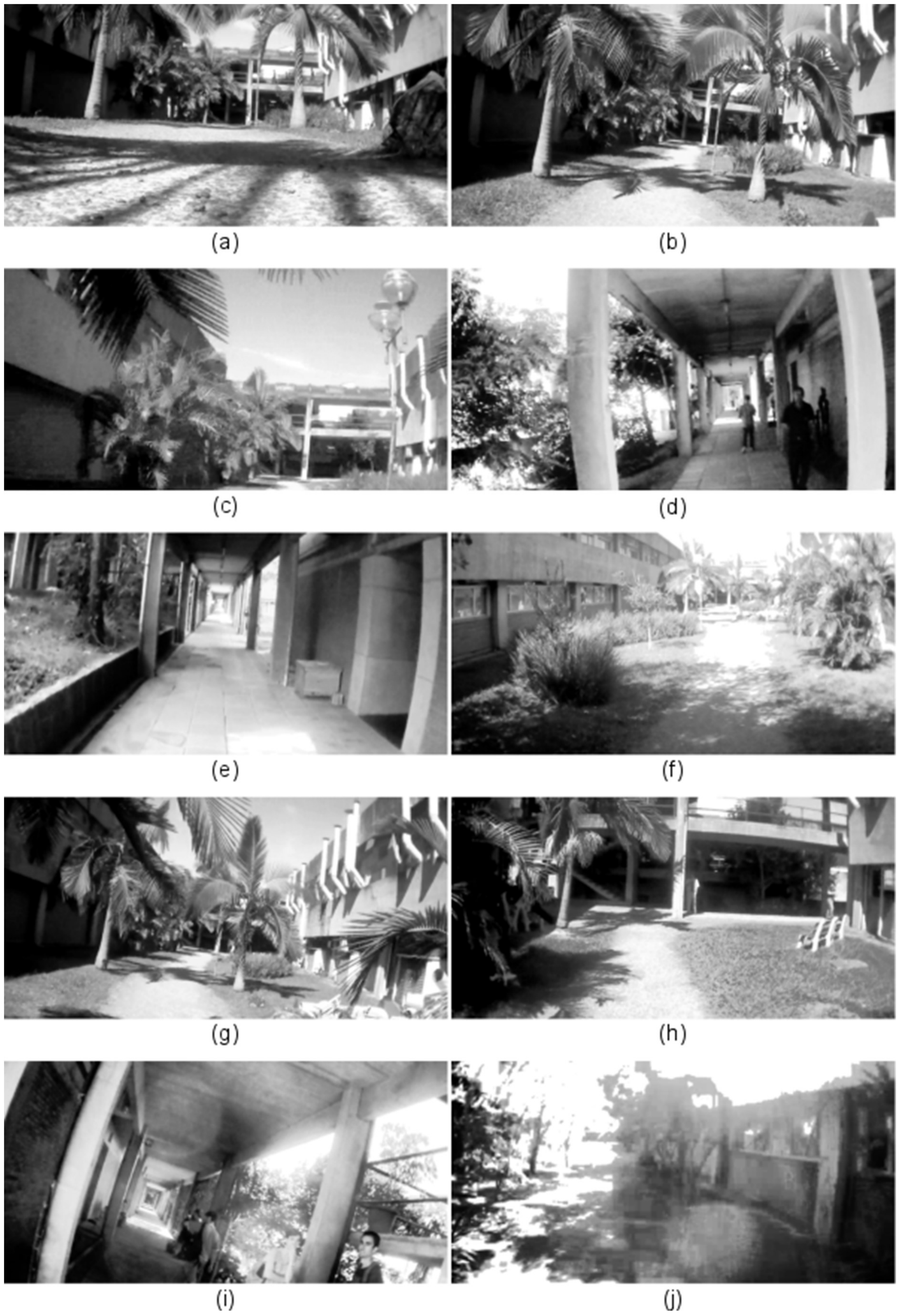


Figure 4.3 Samples of images registered by the Parrot ArDrone 2.0. Image generated by the author.

4.1 Similarity Matrix vs Difference Matrix

As explained at Section 2.3.2 and Section 3.3, the similarity matrix and the difference matrix are representations built to encode the distance between all images from pair-to-pair visits. In S4FE's similarity matrix, all values are normalized in the range from zero to one. The closer to one, the higher similarity identified to that pair. On the other hand, the closer to zero, the lesser similar is the pair. In the contrary, the difference matrix of SeqSLAM have a distinct behavior, once it uses SAD comparison. A high similar pair of images results in an entry close to zero and strictly large in other way. On this section, we will compare the results obtained from both matrices in order to exhibit the advantages and the precision of methods.

The difference matrix of SeqSLAM and the similarity matrix of S4FE were obtained from two different sets of images called *test* and *query* sets. In the first experiment, the *test* set corresponds to the images collected by the handheld camera from the subpath illustrated in Figure 4.4 (a), while the *query* set was collected from the subpath in Figure 4.4 (b). The matrices generated by SeqSLAM and S4FE are shown in Figure 4.6 (b) and (c).

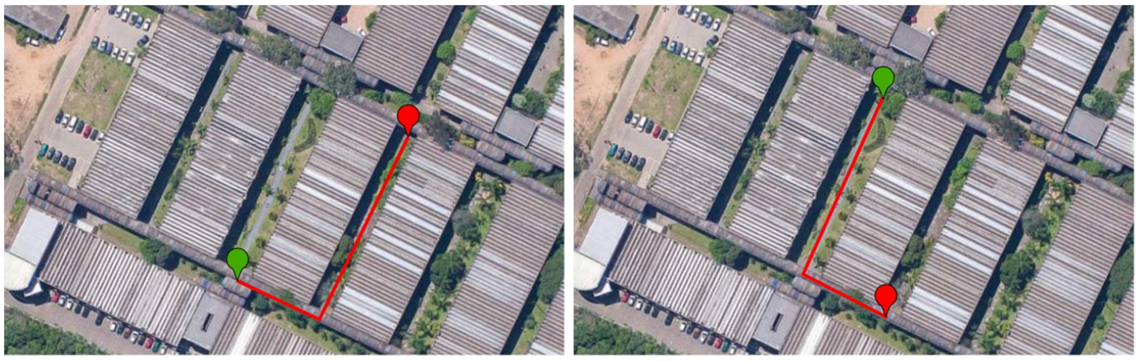
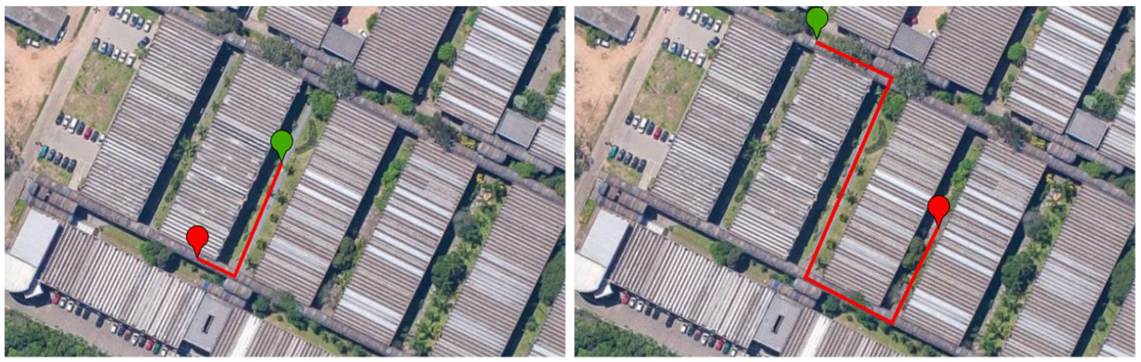

(a) (b)

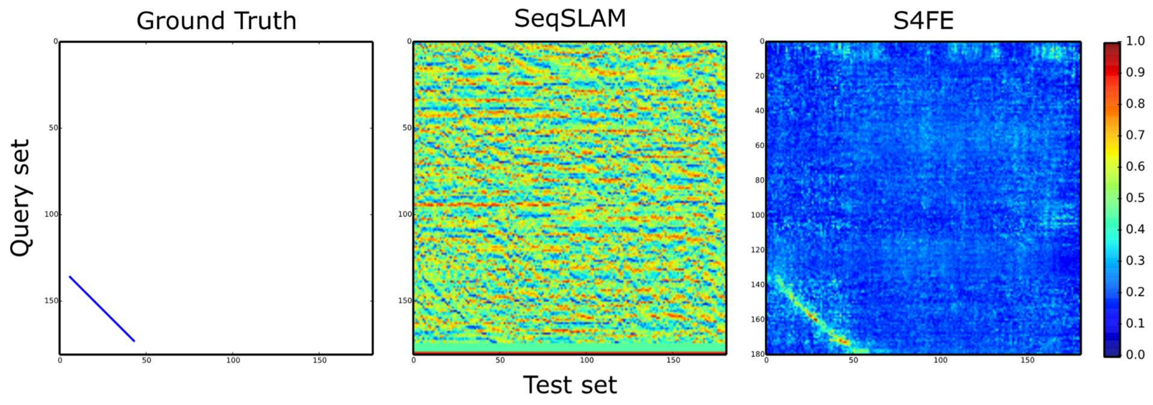
Figure 4.4 Subpaths followed by handheld camera, according to Figure 4.1. Image generated by the author.



(a) (b)

Figure 4.5 Subpaths followed by Parrot ArDrone 2.0, according to Figure 4.1. Image generated by the author.

SeqSLAM exhibits a difference matrix with several values near 0 (i.e., $S_{:,:} \rightarrow 0$) indicating there is a high similarity among the images of test and query sets. However, according to Figure 4.4 (a) and (b), we expect only inputs, associated to the overlap of both paths, to have high similarity values. This happens because many parts of environments can seem symmetric according to SAD. To circumvent this situation, SeqSLAM searches for long sequences preventing small ones which can be false positives produced by SAD. This avoids small sequences that may lead to false positives, but also disregards true positives. On the other hand, S4FE produces a similarity matrix that highlights loop closures, as we can see in Figure 4.6 (c). Besides,



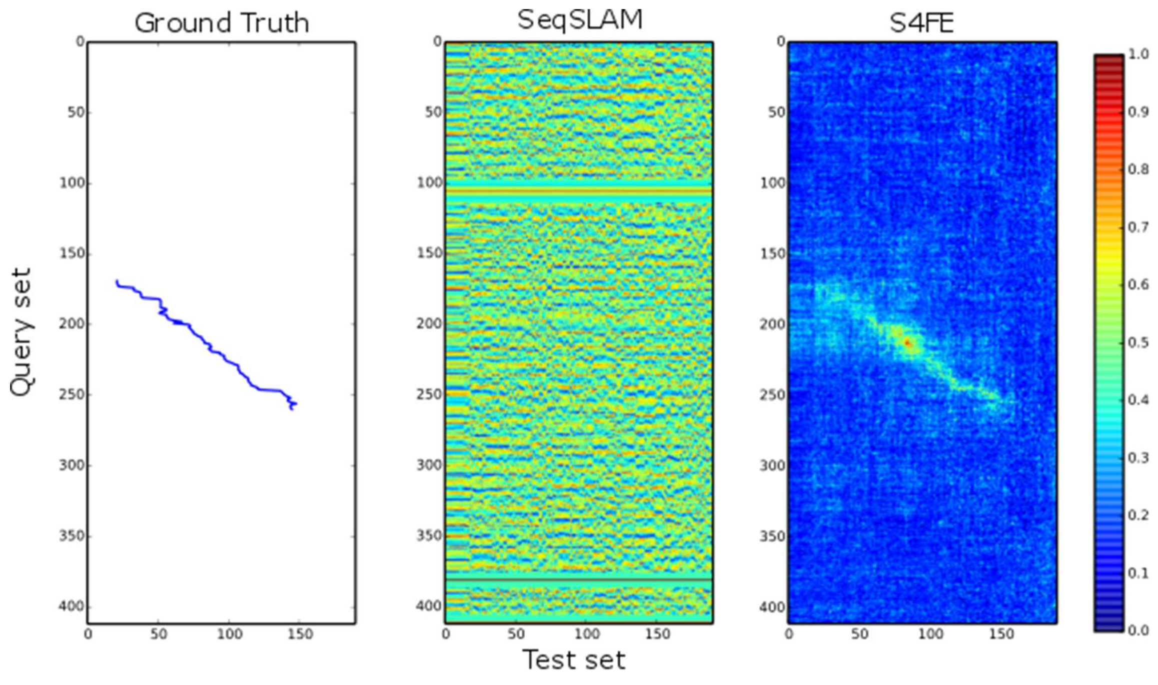
(a) (b) (c)

Figure 4.6 Matrices generated by SeqSLAM (b) and S4FE (c) for the handheld dataset. The ground-truth (a) shows where the loop closure should have been matched. Image generated by the author.

S4FE is able to create a cleaner matrix than SeqSLAM, where short sequences with high similarity are more likely to be a true positive, this allows the identification of small loops.

In the second experiment, the *test* set corresponds to the images collected by the Parrot ArDrone 2.0 from the subpath illustrated in Figure 4.5 (a), while the *query* set was collected from the subpath in Figure 4.5 (b). The matrices generated by SeqSLAM and S4FE are shown in Figure 4.7 (b) and (c).

Again, the matrix obtained by SeqSLAM shows several high similarity values. However, this time, the identification of loop closure is harder than the previous experiment. At the first experiment, the trajectory had a constant speed, a stable movement along the path and almost the same point of view on both visits. All these factors make the image matching process easier to SeqSLAM. However, in the second experiment the robot's flight was very susceptible to wind bursts and many drifts and brusque 3D rotations happened. Once that SeqSLAM expects images gathered by the same point of view, these disturbances interfere on its final result. In spite of the



(a) (b) (c)

Figure 4.7 Matrices generated by SeqSLAM (b) and S4FE (c) for the Parrot ArDrone 2.0 dataset. The ground-truth (a) shows where the loop closure should have been matched. Image generated by the author.

turbulent flight, S4FE similarity matrix still shows a line, though a bit diffused, that indicates the presence of loop closure.

4.2 Precision-Recall

We evaluated the performance of the loop closure methods using a precision-recall analysis. The precision is defined as the percentage of true positives of all returns generated by a front-end method, and the recall is the percentage of ground truth matches returned by the method. A precision-recall curve is computed varying a threshold that indicates the acceptable matching confidence level. These parameters are represented by $\mu_{\min}$, to SeqSLAM, and μ_{η}, to S4FE. FAB-MAP does not foresee a threshold for its returns. Instead, it always chooses the most probable match. As a method is not limited to the known world, if a scene was not seen previously, the most probable match will be in the unmapped set $\overline{M}$. However, in the sense of building a comparison between all methods, we filtered the accepted returns of FAB-MAP by the associated probability of the returned images.

Table 4.1 Parameters

Method	Parameter	Value
FAB-MAP	$p(z_i = 0 \mid e_i = 1)$	0.39
	$p(z_i = 1 \mid e_i = 0)$	0.00
	Number of random places	3000
SeqSLAM	Sequence Length (Δ_s)	5 – 20
	Thumbnail Image Size $(R_x \times R_y)$	64×32
	Patch of size (R_p)	8
	$V_{\min}$	0.8
	$V_{\max}$	1.2
	V_{step}	0.1
	R_{window}	20
S4FE	Sequence Length (Δ_s)	5 – 20
	Mean shift window size (Δ_{ms})	5 - 20
	Disorder threshold $(\mu_{\mathcal{D}})$	0.70

In Table 4.1, all parameters used in each method are presented. To evaluate FAB-MAP algorithm, we used the same BoW approach applied to S4FE. The feature detector and the feature descriptor algorithms used were STAR and SIFT, respectively.

We model the false negative $p(z_i = 0|e_i = 1)$ of the feature detector algorithm to 0.39 and the false negative $p(z_i = 1|e_i = 0)$ to 0.00, as suggest by FAB-MAP (Cummins; Newman, 2007). Thereby, our feature detector model has a high probability for not detecting an existing feature in the scene, but does not indicate any feature that does not exist. We also created 3000 random places samples, based on the Chow-Liu tree, to model the unmapped set. For SeqSLAM, the only varying parameter used in the experiments was the sequence length $\mathbf{\Delta}_s$ to be searched. We evaluate the sequences with length in the range from 5 to 20, in steps of 5. We decided to use the same parameters of thumbnail size $R_x \times R_y$ of 64×32 and patch size R_p of 8 as suggested by SeqSLAM (Milford; Wyeth, 2012). The tested velocities are in range from 0.8 to 1.2, in steps of 0.1, and the R_{window} parameter was also fixed in 20 to guarantee that the second smallest $\mathcal{S}$ was not an overlap of the smallest one. Finally, S4FE has the same variety of sequence lengths $\mathbf{\Delta}_s$ of SeqSLAM, the Mean Shift window size $\mathbf{\Delta}_{ms}$ was set to this same range and the disorder threshold $\boldsymbol{\mu}_{\mathcal{D}}$ was fixed in 0.70, after a range of experiments.

As any false positive in the loop closure could lead a graph-based SLAM to an irreversible wrong environment representation, the best method is the one with the highest recall at 100% of precision. However, the sequence length is also important. The smaller the matching sequence length is, the less resources of memory are required and the less likely the method is to miss small loops closures.

Following this metric, we can see at Table 4.2 that in the first experiment the highest recall with 100% of precision and the smallest sequence length was obtained by S4FE with a recall of 55.26%, sequence length of ten, and MSA window size of five. The smallest recall of 15.79% was obtained by FAB-MAP. As we can see in Figure 4.9, S4FE has shown a better precision-recall curve than SeqSLAM and FAB-MAP. Although FAB-MAP has not returned any false positive on this experiment, its recall was easily overcome by any configuration of the other methods. Moreover, with sequence length of ten, our method has surpassed the recall obtained by SeqSLAM with sequence length of twenty. Also, we can observe that the recall of S4FE tends to reduce insofar the sequence length Δ_s and the MSA window size increase. This can be explained by the presence of the loop closure next to the border of the similarity matrix, see Figure 4.6. As the parameters values of S4FE increase, the entries of the matrix necessary to evaluate a matching, also increase. Thereby, it is impossible for

S4FE evaluating sequences centered in entries close to the border, which in this experiment are the one that represent a loop closure.

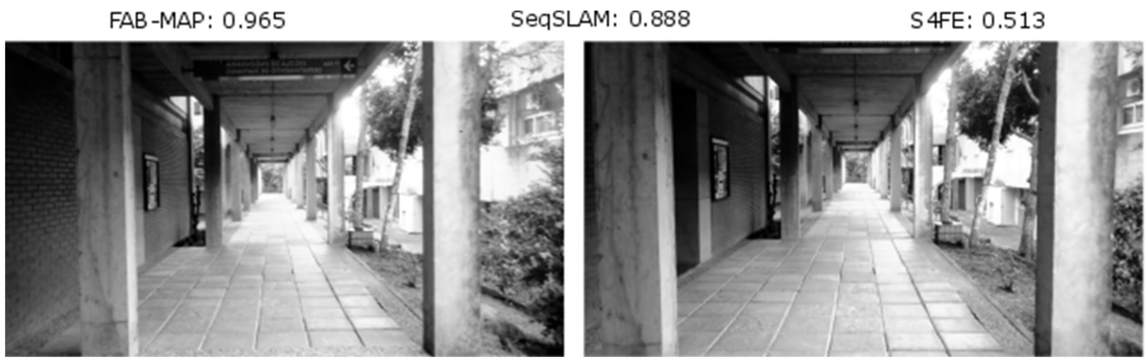


Figure 4.8 Sample of true positive returned by FAB-MAP, SeqSLAM and S4FE for the handheld camera dataset with their coefficients $p(L_i|\mathcal{Z}^k)$, μ and η, respectively. Image generated by the author.

However, the decay of precision of SeqSLAM has another explanation. As can be seen in Figure 4.10, the false positives obtained by the SeqSLAM has a coefficient μ close to its true positives, seen in Figure 4.8, demonstrating how difficult is to determine a value for this threshold. Also can be seeing that a difference of 0.03 in the threshold μ_{min} makes the robot belief that it is in a localization distant from its actually pose. This behavior can be explained by the noisy matrix seen in Figure 4.6 (b). On the other hand, S4FE demonstrate that even a false positive close in the space, which was expected to have a closer confidence level η according to Figure 4.6 (c), has a bigger difference in terms of confidence level than SeqSLAM.

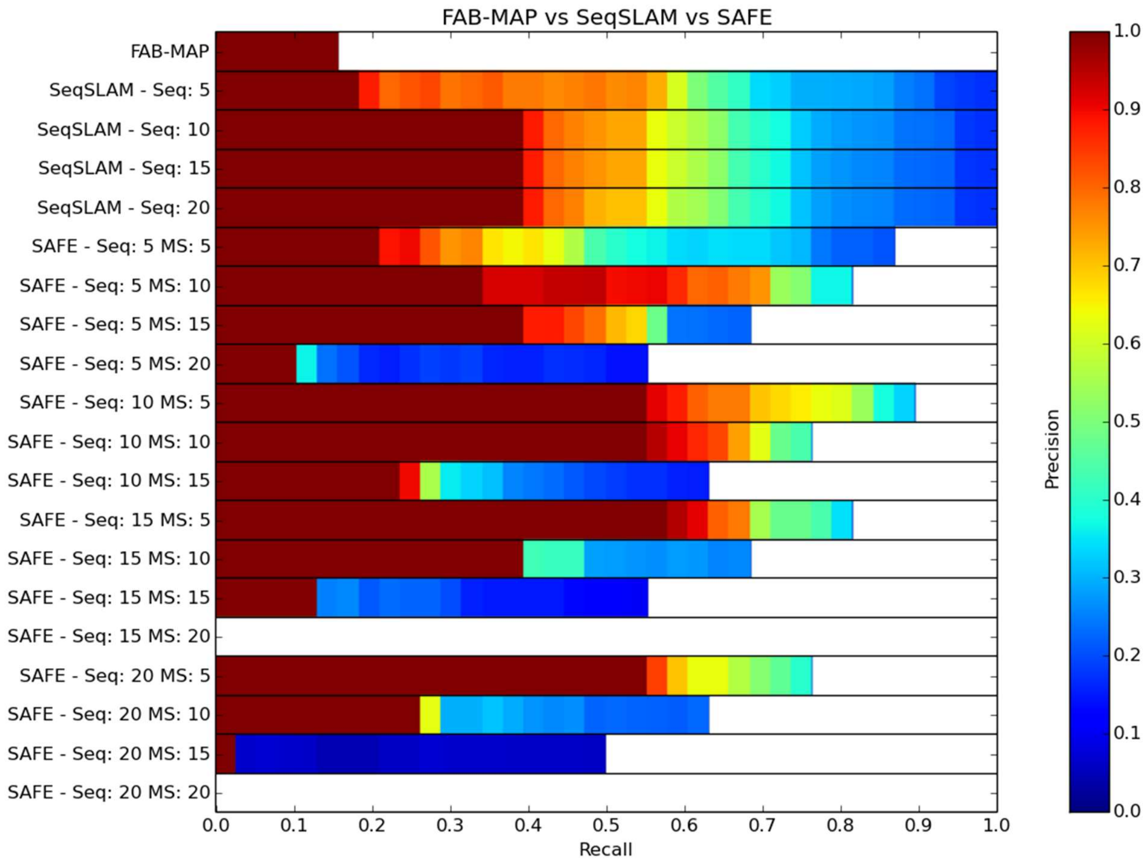


Figure 4.9 Precision-Recall obtained by FAB-MAP, SeqSLAM and S4FE in the challenging handheld outdoor dataset. Image generated by the author.

Table 4.2 Maximum Recall with 100% of precision from Handheld Camera dataset.

	Sequence Length	Mean Shift Window Size	Max. Recall with 100% of precision
FAB-MAP	-	-	15.79%
SeqSLAM	5	-	18.42%
	10	-	39.47%
	15	-	39.47%
	20	-	39.47%
S4FE	5	10	34.21%
		15	39.47%
	10	5	55.26%
		10	55.26%
	15	5	57.89%
		10	39.47%
	20	5	55.26%
		10	26.32%

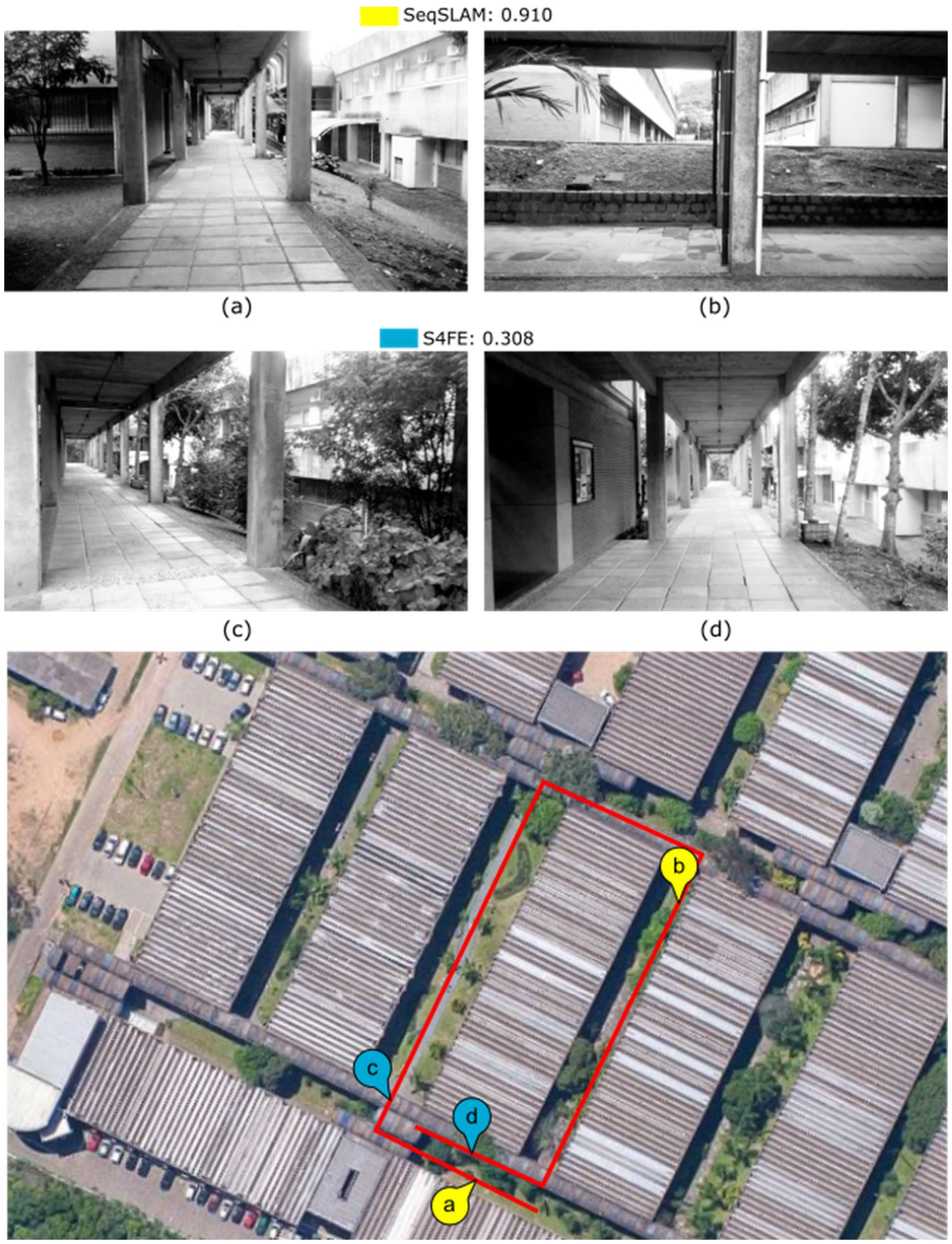


Figure 4.10 Samples of false positive obtained by SeqSLAM and S4FE in the handheld camera dataset with their coefficients μ and η, respectively. FAB-MAP did not return any false positive. Image generated by the author.

As discussed before, in the first experiment, the images were obtained in good conditions making easier to perform the correct sequence matching. However, in the second experiment, the turbulent flight of the robot adds a new challenge. The results obtained from the ArDrone 2.0 dataset can be seen below. As Table 4.3 demonstrates, the highest recall of 56.52%, with 100% of precision, was also obtained by S4FE with sequence length of fifteen and MSA window size of fifteen. FAB-MAP and SeqSLAM did not obtained any result with 100% of precision. Figure 4.12 shows the robustness of S4FE to deal with this difficult traversing scenario. With all tested sequence length, our approach obtained a higher precision-recall curve than SeqSLAM and FAB-MAP. As the images were not aligned, the low performance of SeqSLAM was expected. FAB-MAP, however, has returned several false positives with a probability $p(L_i|\mathcal{Z}^k)$ close to its true positive returns, as can be seen in Figure 4.11 and Figure 4.13. This small difference between true e false positives can also be noticed by the low precision-recall observed in Figure 4.12. Once FAB-MAP is based on features and BoW as S4FE, the behavior of these two methods was expected to be closer. Nevertheless, the environment that the robot was traversing was contaminated by repetitive appearances related to the grass and trees that do not well distinguish a location. This high probability can be related by the fact that FAB-MAP only consider the presence or the absence of a word to determine the probability of a pose, but not the number of occurrences of this repetitive words. Moreover, the weighting procedure of S4FE performed by the TF-IDF algorithm helps our method to filter which words we should rely on to determine a location.

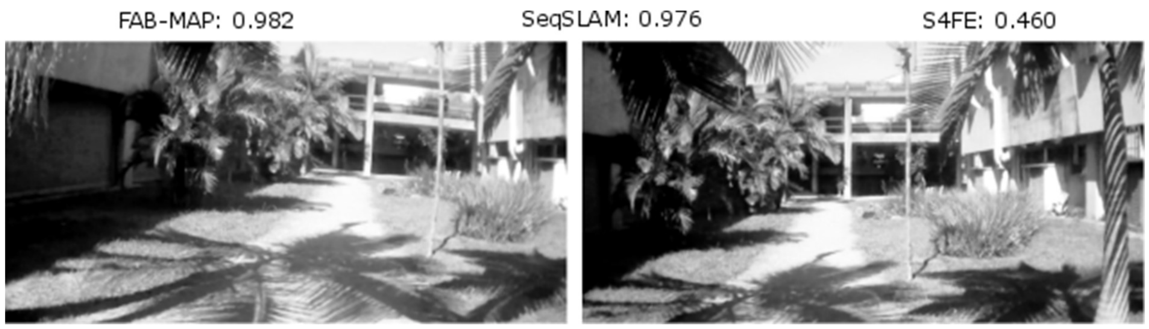


Figure 4.11 Sample of true positive returned by FAB-MAP, SeqSLAM and S4FE for the Parrot ArDrone 2.0 dataset with their coefficients $p(L_i|\mathcal{Z}^k)$, μ and η, respectively. Image generated by the author.

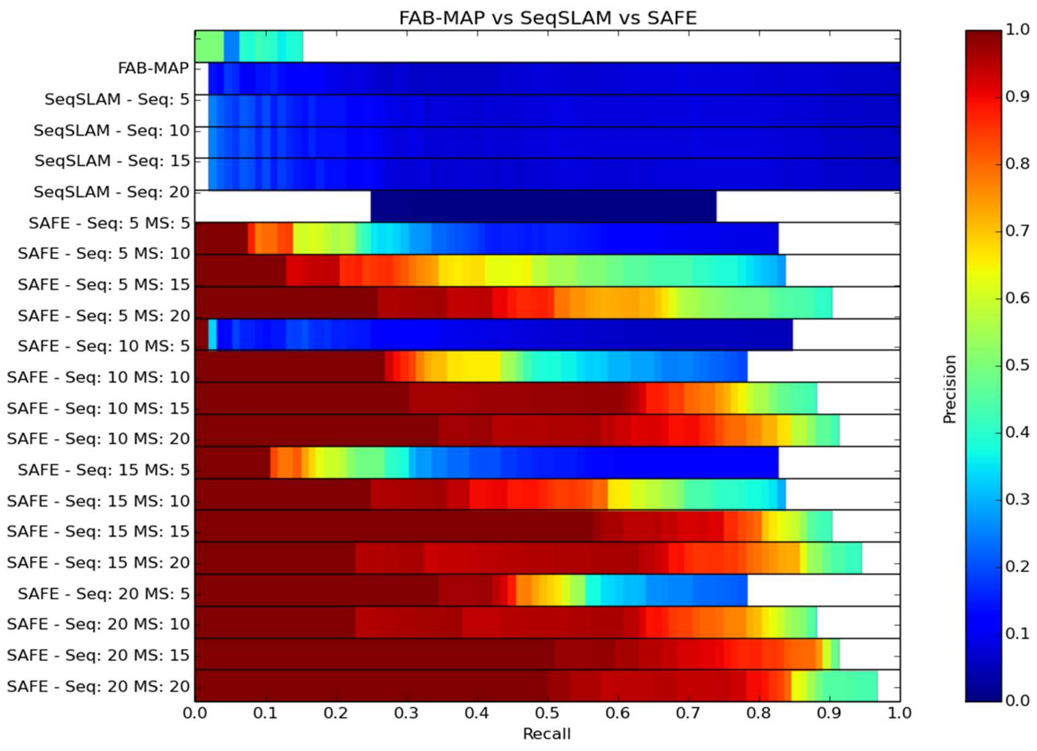


Figure 4.12 Precision-Recall obtained by FAB-MAP, SeqSLAM and S4FE in the challenging Parrot ArDrone 2.0 outdoor dataset. Image generated by the author.

Table 4.3 Maximum Recall with 100% of precision from ArDrone 2.0 dataset.

	Sequence Length	Mean Shift Window Size	Max. Recall with 100% of precision
FAB-MAP	**-**	**-**	**0.00%**
SeqSLAM	5	-	0.00%
	10	-	0.00%
	15	-	0.00%
	20	-	0.00%
S4FE	5	10	7.60%
		15	13.04%
		20	26.00%
	10	10	27.17%
		15	30.43%
		20	34.78%
	15	10	25.00%
		15	56.52%
		20	22.83%
	20	5	34.78%
		10	22.82%
		15	51.08%
		20	50.00%

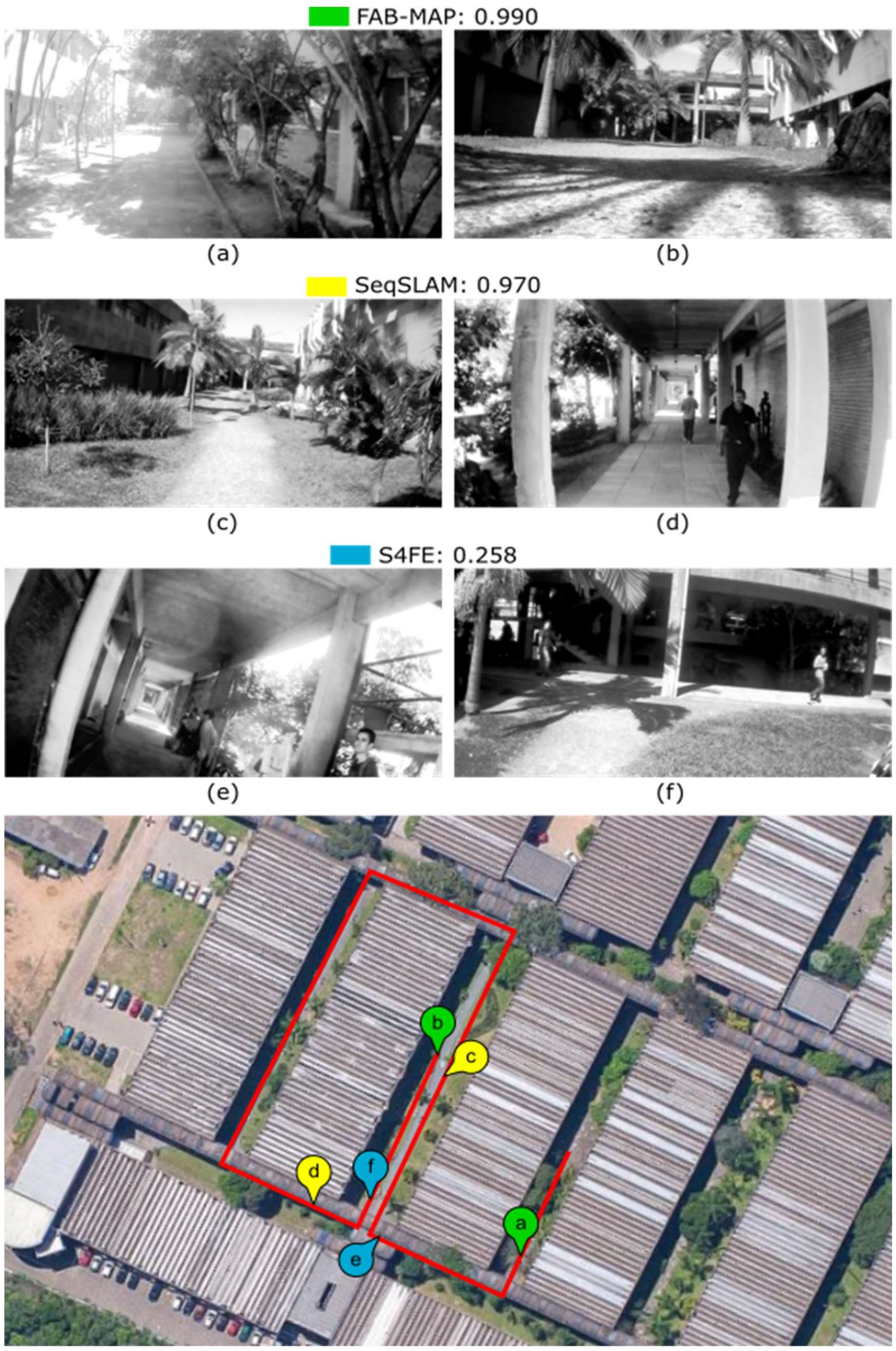


Figure 4.13 Samples of false positive obtained by FAB-MAP, SeqSLAM and S4FE in the Parrot ArDrone 2.0 dataset with their coefficients $p\left(L_i \middle| \mathcal{Z}^k\right)$, μ and η, respectively. Image generated by the author.

5 FINAL REMARKS AND FUTURE WORK.

In this work we demonstrated how the combination of BoW approach and a feature frequency filter, such as TF-IDF, can highlight the revisited loops in the similarity matrix. This combination together with MSA allowed to transform the process of path tracking into a search for a match sequence under a local consensus over the similarity matrix. We also showed that S4FE improves the loop closure recall up to 16% , with 100% of precision, using a sequence with half of the entries needed by the SeqSLAM. Comparing to FAB-MAP, our method can overcome the state-of-the-art recall up to 40%. Moreover, our method reached 56.52% of recall, with 100% of precision, in a scenario where FAB-MAP and SeqSLAM failed to guarantee true positives.

In spite of good results, there are some points that could be further explored. For instance, the convergence of MSA could be improved by using multiresolution image pyramids. Since the loop closure based on sequences searches for several high similar entries, tracking a concentration these entries in multiresolution approach would help our method to avoid the convergence to a noisy data at the similarity matrix. Also, as the experiments demonstrate, the MSA window size Δ_{ms} and the sequence length Δ_s are determinant to the precision of our method. The choice of these parameters is crucial to a good performance and this choice is highly dependent of the environment. In this sense, our method could be further improved by an adaptive bandwidth selection as presented by Gallant et. al. (Gallant et al., 2011). This would lead not only to a precision improvement, but also to a reduction in terms of computational cost of our method. We also noticed that the use of the same fixed threshold μ_D is not ideal for every scenario. This gives rise to further investigations towards a dynamic threshold technique, such as the one proposed by Hui-Fung Ng (Hui-Fuang Ng, 2004).

REFERENCES

Agrawal, M., Konolige, K., & Blas, M. R. CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching. Springer. In D. Forsyth, P. Torr, & A. Zisserman (Eds.), **ECCV**. (Vol. 5305, pp. 102–115). 2008. Available at http://doi.org/10.1007/978-3-540-88693-8_8.

Bay, H., Ess, A., Tuytelaars, T., & Gool, L. Van. Speeded-Up Robust Features (SURF). **Computer Vision and Image Understanding**. , *110*(3), 346–359. 2008. Available at http://doi.org/http://dx.doi.org/10.1016/j.cviu.2007.09.014.

Bay, H., Tuytelaars, T., & Van Gool, L. SURF: Speeded Up Robust Features. Berlin, Heidelberg: Springer Berlin / Heidelberg. In A. Leonardis, H. Bischof, & A. Pinz (Eds.), **ECCV**. (Vol. 3951, pp. 404–417). 2006.

Bosse, M., Newman, P., Leonard, J., Soika, M., Feiten, W., & Teller, S. An Atlas framework for scalable mapping. Piscataway, NJ, USA: IEEE Press. In **Proceedings of the 2003 IEEE International Conference on Robotics and Automation (ICRA)**. (Vol. 2, pp. 1899–1906). 2003. Available at http://doi.org/10.1109/ROBOT.2003.1241872.

Chow, C., & Liu, C. Approximating Discrete Probability Distributions with Dependence Trees. **IEEE Transations of Information Theory**. , *14*(3), 462–467. 1968. Available at http://doi.org/10.1109/TIT.1968.1054142.

Cummins, M., & Newman, P. Probabilistic Appearance Based Navigation and Loop Closing. Rome. In **International Conference on Robotics and Automation (ICRA)**. 2007.

Cummins, M., & Newman, P. FAB-MAP: Probabilistic Localization and Mapping in the Space of Appearance. **The International Journal of Robotics Research**. , *27*(6), 647–665. 2008. Available at http://doi.org/10.1177/0278364908090961.

Cummins, M., & Newman, P. Highly scalable appearance-only SLAM - FAB-MAP 2.0. Seattle, USA. In **Proceedings of Robotics: Science and Systems**. 2009.

Cummins, M., & Newman, P. Appearance-only SLAM at large scale with FAB-MAP 2.0. **The International Journal of Robotics Research**. 2010. Available at http://doi.org/10.1177/0278364910385483.

Davison, A. J., & Kita, N. 3D simultaneous localisation and map-building using active vision for a robot moving on undulating terrain. In **Computer Vision and Pattern Recognition**. (Vol. 1, pp. I–384–I–391 vol.1). 2001. Available at http://doi.org/10.1109/CVPR.2001.990501.

Dellaert, F., & Kaess, M. Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing. **Journal of Robotics Research**. , *25*(12), 1181–1204. 2006. Available at http://doi.org/10.1177/0278364906072768.

Fukunaga, K., & Hostetler, L. The estimation of the gradient of a density function, with applications in pattern recognition. **Information Theory, IEEE Transactions on**. , *21*(1), 32–40. 1975. Available at http://doi.org/10.1109/TIT.1975.1055330.

Gallant, a. J., Kaliteevski, M. a., Brand, S., Wood, D., Petty, M., Abram, R. a., & Chamberlain, J. M. Bandwidth Selection for Mean-shift based Unsupervised Learning

Techniques: a Unified Approach via Self-coverage. **Journal of Applied Physics**. , *102*(October), 0–103. 2011. Available at http://doi.org/10.1063/1.2756072.

Grisetti, G., Kummerle, R., Stachniss, C., & Burgard, W. A Tutorial on Graph-Based SLAM. **IEEE Intelligent Transportation Systems Magazine**. , *2*(4), 31–43. 2010. Available at http://doi.org/10.1109/MITS.2010.939925.

Grisetti, G., Stachniss, C., & Burgard, W. Improving Grid-based SLAM with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling. Piscataway, NJ, USA: IEEE Press. In **Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)**. (pp. 2443–2448). 2005.

Grisetti, G., Stachniss, C., & Burgard, W. Nonlinear constraint network optimization for efficient map learning. **IEEE Transactions on Intelligent Transportation Systems**. , *10*(3), 428–439. 2009. Available at http://doi.org/10.1109/TITS.2009.2026444.

Hansen, P., & Browning, B. Visual place recognition using HMM sequence matching. In **Intelligent Robots and Systems (IROS)**. (pp. 4549–4555). 2014. Available at http://doi.org/10.1109/IROS.2014.6943207.

Ho, K. L., & Newman, P. Detecting loop closure with scene sequences. **International Journal of Computer Vision**. , *74*(3), 261–286. 2007.

Hui-Fuang Ng. Automatic Thresholding for Defect Detection. Hong Kong, China: IEEE. In **Third International Conference on Image and Graphics (ICIG)**. (pp. 532–535). 2004. Available at http://doi.org/10.1109/ICIG.2004.43.

Kaess, M., Ranganathan, A., & Dellaert, F. iSAM: Fast Incremental Smoothing and Mapping with Efficient Data Association. In **IEEE International Conference on Robotics and Automation**. (pp. 1670–1677). 2007. Available at http://doi.org/10.1109/ROBOT.2007.363563.

Lee, S.-J., & Song, J.-B. A new sonar salient feature structure for EKF-based SLAM. Piscataway, NJ, USA: IEEE Press. In **Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. (pp. 5966–5971). 2010. Available at http://doi.org/10.1109/IROS.2010.5650169.

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

Lu, F., & Milios, E. Globally Consistent Range Scan Alignment for Environment Mapping. **Autonomous Robots**. , *4*(4), 333–349. 1997. Available at http://doi.org/10.1023/A:1008854305733.

Maffei, R., Jorge, V. A. M., Prestes, E., & Kolberg, M. Integrated Exploration using Time-based Potential Rails. In **Proc. of IEEE ICRA**. 2014.

Makarenko, A. A., Williams, S. B., Bourgault, F., & Durrant-Whyte, H. F. An experiment in integrated exploration. In **Proc. of IEEE/RSJ IROS**. (pp. 534–539). 2002. Available at http://doi.org/10.1109/IRDS.2002.1041445.

Meilă, M. An Accelerated Chow and Liu Algorithm: Fitting Tree Distributions to High-Dimensional Sparse Data. **Proceedings of the 16th International Conference on Machine Learning**. , (C), 249–257. 1999. Available at http://dl.acm.org/citation.cfm?id=645528.657640.

Milford, M., Schill, F., Corke, P., Mahony, R., & Wyeth, G. Aerial SLAM with a single camera using visual expectation. In **IEEE International Conference on Robotics and Automation (ICRA)**. (pp. 2506–2512). 2011. Available at http://doi.org/10.1109/ICRA.2011.5980329.

Milford, M., Turner, I., & Corke, P. Long exposure localization in darkness using consumer cameras. In **IEEE International Conference on Robotics and Automation (ICRA)**. (pp. 3755–3761). 2013. Available at http://doi.org/10.1109/ICRA.2013.6631105.

Milford, M., & Wyeth, G. SeqSLAM: Visual route-based navigation for sunny summer days and stormy winter nights. In **IEEE International Conference on Robotics and Automation (ICRA)**. (pp. 1643–1649). 2012. Available at http://doi.org/10.1109/ICRA.2012.6224623.

Milford, M. Vision-based place recognition: how low can you go? **The International Journal of Robotics Research**. , *32*(7), 766–789. 2013. Available at http://doi.org/10.1177/0278364913490323.

Nakashima, T., & Nakamura, R. Daily Clustering for the Electronic Newspaper based on the Analysis of Trend. , (3).n.d.

Naseer, T., Ruhnke, M., Stachniss, C., Spinello, L., & Burgard, W. Robust Visual SLAM Across Seasons. In **IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)**. (pp. 2529–2535). 2015.

Naseer, T., Spinello, L., Burgard, W., & Stachniss, C. Robust Visual Robot Localization Across Seasons using Network Flows. In **AAAI**. 2014. Available at http://www.informatik.uni-freiburg.de/~naseer/publications/naseer14aaai.pdf.

Neubert, P., Sunderhauf, N., & Protzel, P. Appearance change prediction for long-term navigation across seasons. **European Conference on Mobile Robots (ECMR)**. , 198–203. 2013. Available at http://doi.org/10.1109/ECMR.2013.6698842.

Neuland, R., Nicola, J., Maffei, R., Jaulin, L., Prestes, E., & Kolberg, M. Hybridization of Monte Carlo and set-membership methods for the global localization of underwater robots. In **IEEE/RSJ International Conference on Intelligent Robots and Systems**. (pp. 199–204). 2014. Available at http://doi.org/10.1109/IROS.2014.6942561.

Olson, E., Leonard, J., & Teller, S. Fast iterative alignment of pose graphs with poor initial estimates. **Proceedings - IEEE International Conference on Robotics and Automation**. , *2006*(May), 2262–2269. 2006. Available at http://doi.org/10.1109/ROBOT.2006.1642040.

Pepperell, E., Corke, P., & Milford, M. All-environment visual place recognition with SMART. In **Robotics and Automation (ICRA), 2014 IEEE International Conference on**. (pp. 1612–1618). 2014. Available at http://doi.org/10.1109/ICRA.2014.6907067.

Pepperell, E., Corke, P., & Milford, M. Automatic Image Scaling for Place Recognition in Changing Environments. In **International Conference on Robotics and Automation**. 2015.

Prestes, E., & Idiart, M. Sculpting potential fields in the BVP Path Planner. In **Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on**. (pp. 183–188). 2009. Available at http://doi.org/10.1109/ROBIO.2009.5420620.

Prestes, E., Trevisan, M., Idiart, M. A. P., & Engel, P. M. BVP-exploration: further improvements. In **Proc. of IEEE/RSJ IROS**. (pp. 3239–3244). 2003. Available at http://doi.org/10.1109/IROS.2003.1249655.

Siegwart, R., & Nourbakhsh, I. R. **Introduction to Autonomous Mobile Robots**. Scituate, MA, USA: Bradford Company. 2004.

Sivic, J., & Zisserman, A. {Video Google}: {A} Text Retrieval Approach to Object Matching in Videos. In **Proceedings of the International Conference on Computer Vision**. (Vol. 2, pp. 1470–1477). 2003. Available at http://www.robots.ox.ac.uk/~vgg.

Smith, R., Self, M., & Cheeseman, P. A stochastic map for uncertain spatial relationships. **Proceedings of the 4th International Symposium on Robotics Research**. , (0262022729), 467–474. 1988. Available at http://portal.acm.org/citation.cfm?id=57472.

Sparck Jones, K. A statistical interpretation of term specificity and its application in retrieval. **Journal of Documentation**. , *28*(1), 11–21. 1972.

Teynor, A., & Burkhardt, H. Fast Codebook Generation by Sequential Data Analysis for Object Classification. Springer Berlin Heidelberg. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, N. Paragios, S.-M. Tanveer, … T. Malzbender (Eds.), **Advances in Visual Computing**. (Vol. 4841, pp. 610–620). 2007. Available at http://doi.org/10.1007/978-3-540-76858-6_59.

Thrun, S., Burgard, W., & Fox, D. **Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)**. Cambridge, MA. 2005.

Thrun, S., & Montemerlo, M. The Graph SLAM Algorithm with Applications to Large-Scale Mapping of Urban Structures. **The International Journal of Robotics Research**. , *25*(5-6), 403–429. 2006. Available at http://doi.org/10.1177/0278364906065387.

Viola, P., & Wells, W.M., I. Alignment by maximization of mutual information. **Proceedings of IEEE International Conference on Computer Vision**. , *24*(2), 16–23. 1995. Available at http://doi.org/10.1109/ICCV.1995.466930.

Yol, A., Delabarre, B., Dame, A., Dartois, J.-É., & Marchand, E. Vision-based Absolute Localization for Unmanned Aerial Vehicles. **IEEE/RSJ Int. Conf. on ….** , *14*(IROS), 3429–3434. 2014. Available at http://doi.org/10.1109/IROS.2014.6943040.