VISUAL AND INERTIAL DATA FUSION FOR GLOBALLY CONSISTENT POINT CLOUD REGISTRATION

CLÁUDIO DOS SANTOS FERNANDES

VISUAL AND INERTIAL DATA FUSION FOR GLOBALLY CONSISTENT POINT CLOUD REGISTRATION

Dissertação apresentada ao Programa de Pós-Graduação em Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Computação.

Orientador: Mario Fernando Montenegro Campos

Belo Horizonte

Junho de 2013

CLÁUDIO DOS SANTOS FERNANDES

VISUAL AND INERTIAL DATA FUSION FOR GLOBALLY CONSISTENT POINT CLOUD REGISTRATION

Dissertation presented to the Graduate Program in Computação of the Universidade Federal de Minas Gerais in partial fulfillment of the requirements for the degree of Master in Computação.

Advisor: Mario Fernando Montenegro Campos

Belo Horizonte

June 2013

Fernandes, Cláudio dos Santos

F363v Visual and Inertial Data Fusion for Globally Consistent Point Cloud Registration / Cláudio dos Santos Fernandes. — Belo Horizonte, 2013
xxiv, 68 f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais
Orientador: Mario Fernando Montenegro Campos

1. Computação - Teses. 2. Robótica - Teses.
I. Título.

CDU 519.6*82.9(043)

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Fusão de dados visuais e inerciais para registro globalmente consistente de nuvens de pontos

CLAUDIO DOS SANTOS FERNANDES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

[illegible]

PROF. MARIO FERNANDO MONTENEGRO CAMPOS - Orientador
Departamento de Ciência da Computação - UFMG

[illegible]

PROF. ALEXEI MANSO CORREA MACHADO
Departamento de Ciência da Computação – PUC/MG

[illegible]

PROF. ERICKSON RANGEL DO NASCIMENTO
Departamento de Ciência da Computação - UFMG

Belo Horizonte, 04 de julho de 2013.

Em memória de Neide da Silva, minha avó. Seus gestos de bondade jamais serão esquecidos.

Acknowledgments

It has been a great honor to research under guidance of my advisor, Mario Campos, who has kindly guided me through the research process, helping me understand this continuous, never ending process.

To all my colleagues at the Computer Vision and Robotics laboratory (VeRLab), I'd like to express my gratitude for the amazing experiences we had together during the last three years.

This work has been financially supported by CAPES and CNPq. Our experiments were based on data acquired by a Zebedee sensor, which were kindly provided to us by Alberto Elfes, from CSIRO.

I'd like to thank my parents, Antonio and Maria, for always being on my side, helping me make it through every tough moment of my life. Last, but not least, I would also like to thank the guys at the Computer Graphics study group for the many fruitful discussions about math, the universe and everything.

Resumo

Este trabalho aborda o mapeamento tridimensional de ambientes estáticos utilizando um sensor *RGB-D*, que captura imagem e profundidade, e um sensor *MARG*, composto de sensores inerciais e magnetômetros.

O problema do mapeamento é relevante ao campo da robótica, uma vez que sua solução permitirá a robôs navegarem e mapearem de forma autônoma ambientes desconhecidos. Além disso, traz impactos em diversas aplicações que realizam modelagem 3D a partir de varreduras obtidas de sensores de profundidade. Dentre elas, estão a replicação digital de esculturas e obras de arte, a modelagem de personagens para jogos e filmes, e a obtenção de modelos *CAD* de edificações antigas.

Decidimos abordar o problema realizando o registro rígido de nuvens de pontos adquiridas sequencialmente pelo sensor de profundidade, usando as informações providas pelo sensor inercial como guia tanto no estágio de alinhamento grosseiro quanto na fase de otimização global do mapa gerado. Durante o alinhamento de nuvens de pontos por casamento de features, a rotação estimada pelo sensor MARG é utilizada como uma estimativa inicial da orientação entre nuvens de pontos. Assim, procuramos casar pontos de interesse considerando apenas três graus de liberdade translacionais. A orientação provida pelo MARG também é utilizada para reduzir o espaço de busca por fechamento de *loops*.

A fusão de dados RGB-D com informações inerciais ainda é pouco explorada na literatura. Um trabalho similar já publicado apenas utiliza dados inerciais para melhorar a estimativa da rotação durante o alinhamento par a par de maneira *ad-hoc*, potencialmente descartando-os em condições específicas, e negligenciando o estágio de otimização global. Por utilizar um sensor MARG, assumimos que o drift do sensor é negligível em nossa aplicação, o que nos permite sempre utilizar seus dados, especialmente durante a fase de otimização global.

Em nossos experimentos, realizamos o mapeamento das paretes de um ambiente retangular de dimensões 9,84m $\times$ 7,13m e comparamos os resultados com um mapeamento da mesma cena feito a partir de um sensor Zebedee, estado da arte em mapeamento 3D a laser. Também comparamos o algoritmo proposto com a metodologia RGB-D SLAM, que, ao contrário da nossa metodologia, não foi capaz de detectar a região de fechamento de *loop*.

Palavras-chave: SLAM, Registro de Nuvens de Pontos, Sensoriamento Inercial.

Abstract

This work addresses the problem of mapping 3D static environments by using an *RGB-D* sensor, that captures image and depth, and a *MARG* sensor, composed by inertial sensors and magnetometers.

The approached problem is relevant to the robotics field, since its solution will allow mobile robots to autonomously navigate and map unknown environments. Besides, it has impacts on several applications that perform 3D modeling by using scans obtained from depth sensors. Amongst them, one can mention the digital replication of sculptures and art objects, the modeling of characters for games and movies, and the reconstruction of *CAD* models from old buildings.

We have decided to address the problem by performing a rigid registration of point clouds sequentially captured by the depth sensor, posteriorly using the data provided by the inertial sensor as a guide both during the coarse alignment stage and during the global optimization of the estimated map. During the point cloud alignment based on feature matching, the rotation estimated from the MARG sensor is used as an initial estimation of the attitude between point clouds. Thereby, we seek to match keypoints considering only three translational degrees of freedom. The attitude given by the MARG is also used to reduce the search space for loop closures.

The fusion of RGB-D and inertial data is still very little explored in the related literature. A similar work already published only uses inertial data to improve the attitude estimation during the pairwise alignment in an *ad-hoc* fashion, potentially discarding it under specific conditions, and neglecting the global optimization stage. Since we use a MARG sensor, we assume the sensor drift to be negligible for the purposes of our application, which allows us to always use its data, specially during the global optimization stage.

In our experiments, we mapped the walls of a rectangular room with dimensions 9.84m $\times$ 7.13m and compared the results with a map from the same scene captured by a Zebedee sensor, state of the art in terms of laser-based 3D mapping. We also compared the proposed algorithm against the RGB-D SLAM methodology, which, unlike our methodology, was not capable of detecting the loop closure region.

Keywords: SLAM; Point Cloud Registration; Inertial Sensing.

List of Figures

List of Tables

List of Acronyms

BASE	Binary Appearance and Shape Elements
CAD	Computer-aided Design
EKF	Extended Kalman Filter
ICP	Iterative Closest Points
IMU	Inertial Measurement Unit
KF	Kalman Filter
MARG	Magnetic, Angular Rate, and Gravity sensor
NDT	Normal Distributions Transform
PCA	Principal Component Analysis
RANSAC	Random Sample Consensus
RGB-D	Red, Green, Blue and Depth data
RMS	Root Mean Square
SLAM	Simultaneous Localization and Mapping
SLERP	Spherical Linear Interpolation
SURF	Speeded Up Robust Features

Contents

Chapter 1

Introduction

The general public has recently witnessed the popularization of low cost 3D scanning devices, with prices reaching as low as only a couple hundred of dollars for an Xbox Kinect[1]. Despite having low prices when compared to industrial scanners, these devices produce data with reasonable accuracy, with error dispersions smaller than 3cm under typical usage [Khoshelham and Elberink, 2012]. Therefore a wide range of applications, such as gesture recognition and 3D map acquisition (the former being the focus of this thesis) are now available to a broad public.

Another class of sensors going through a similar process of decreasing cost are the inertial measurement units. These devices can be used to measure orientation in space (which is also referred to as attitude). With the release of the Wii console in late 2006, inertial sensors have also been integrated in smart phones, MP3 players and other console peripherals, which has fostered researches devoted to improving their accuracy and lowering their prices.

One can expect these sensors to serve complimentary purposes in a mapping framework, as the problem of environment mapping can be broken down into grabbing several local depth images and assembling them into a global representation, which requires an estimation of the pose of the depth sensor at the time the depth images were acquired.

The present work aims to develop and evaluate a methodology to produce a globally consistent environment map, by fusing data from a sensor that captures both color and geometry (or *RGB-D* sensor) and a variant of inertial sensors that also contains a digital compass (known as *MARG* sensor).

[1]Market prices as of February, 2013.

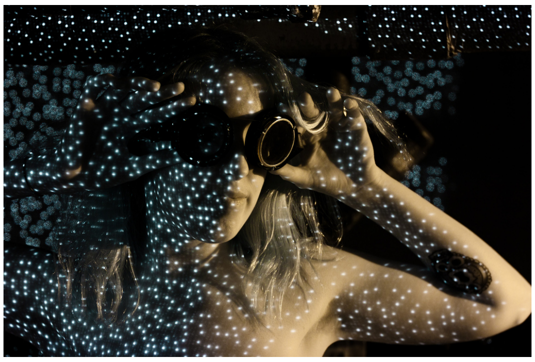


Figure 1.1: Speckle pattern projected by a Kinect sensor and captured by an infra red camera. Courtesy of Audrey Penven[1].

1.1 Motivation

Commercial depth sensors have become increasingly cheaper and accurate, since the development of the first laser range sensors in the eighties [Blais, 2004]. This phenomenon was accompanied by the introduction of this kind of device in many different applications. For instance, range devices are used on sports that require accurate range measurements, such as golf and archery. Yet another important application for this kind of sensors is digital 3D modeling. The entertainment industry has a growing requirement for digitally modeled objects and characters, which, in some circumstances, is achieved by acquiring 3D scans of sculptures and actors. There are companies specialized in digitalizing sculptures and reproducing faithful replicas of the original piece of art.

In the mobile robotics field, several researchers have studied the Simultaneous Localization and Mapping (SLAM) problem. This problem consists of building a map in a scenario where a mobile robot doesn't have *a priori* the map representing the environment being explored. Depending on the environment being mapped, its representation can be either two or three dimensional. Simple structured environments, such as a room or a single floor of a building can be easily represented by a 2D map, while more complex environments, like mines and multistory buildings, may require a 3D representation. Analogously, localization typically has only three degrees of freedom on simple environments, commonly referred to as (x, y, θ), which characterizes a two dimensional translation and an orientation. More complex combinations of environments and robots can allow motions with up to six degrees of freedom – three translational and three rotational.

Another application that benefits from 3D scanning is the monitoring and atten-

[1]www.audreypenven.net

dance of construction sites with the purpose of verifying the compliance with computer-aided design (CAD) models. Point clouds obtained from these environments can be submitted to a cross validation procedure capable of detecting irregularities in project progress [Frédéric and Bosché, 2010]. Additionally, 3D scanning techniques can be used to recover a CAD model from old buildings that had none.

1.2 Problem Definition

For the purpose of this work, the output of an *RGB-D* sensor can be defined as a set $\mathbf{P}$ of tuples $(c, \mathbf{p})$, where c denotes the average of the RGB channels (which represents the gray scale intensity of an image pixel), and $\mathbf{p}$ represents the three-dimensional coordinates of the point captured by this pixel with respect to a frame centered at the sensor. We refer to the set $\mathbf{P}$ by the term Point Cloud.

The MARG sensor outputs the three vectors $\mathbf{a}$, $\mathbf{g}$ and $\mathbf{m}$, all specified in a local frame rigidly attached to the sensor. $\mathbf{a}$ represents the sum of the MARG sensor acceleration and the gravity vector; $\mathbf{g}$, its angular velocity and $\mathbf{m}$ the earth's magnetic field.

Let C be a set of point clouds acquired with the depth sensor being at different poses, such that the union of all point clouds completely covers a static region to be mapped. Also, for any point cloud $\mathbf{P}_i \in C$, there is at least one $\mathbf{P}_j \in C$, $i \neq j$ such that there is an intersection between $\mathbf{P}_i$ and $\mathbf{P}_j$.

Our problem can be divided into the three following subproblems:

1. Find a pairwise transformation between sequential point clouds. For all $({}^{i}\mathbf{p},\ {}^{i+1}\mathbf{p}) \in \mathbf{P}_i \cap \mathbf{P}_{i+1}$, this subproblem consists of finding a ${}^{i}_{i+1}\mathbf{T}$ that best approximates the relation ${}^{i}\mathbf{p} = {}^{i}_{i+1}\mathbf{T}\ {}^{i+1}\mathbf{p}$.

2. Find a global set of transformations such that

$$\forall \mathbf{P}_i,\ \mathbf{P}_j \in C, i \neq j,\ \mathbf{P}_i \cap \mathbf{P}_j \neq \varnothing \Rightarrow \exists\ {}^{i}_{j}\mathbf{T} | {}^{i}\mathbf{p} \approx {}^{i}_{j}\mathbf{T}\ {}^{j}\mathbf{p}.$$

Our goal is to solve this problem by fusing RGB-D data (coming from a single handheld sensor) with information captured from the MARG. In order to do this, we also need to find a rigid transformation from the MARG sensor local frame to the RGB-D sensor local frame. This process, also known as extrinsic calibration, is also a subject of this work.

1.3 Contributions

This work proposes two keypoint matching algorithms that take advantage of attitude data from the *MARG* sensor, one for the purpose of pairwise alignment and one for loop closure detection. Although we present these matchers in our own registration pipeline, they can be embedded in any other registration algorithm if a *MARG* sensor is present, potentially improving its capacity.

We also present a methodology for finding the extrinsic calibration between a *MARG* and an *RGB-D* sensor. The proposed method can also be used with similar sensors that lack color information, such as time of flight cameras.

1.4 Roadmap

This first chapter introduces the reader to the problem approached by this thesis, and also explains its relevance to the scientific community.

Chapter 2 presents an overview of researches that attempted to achieve similar goals as ours. We discuss the evolution of monocular SLAM, in which only a camera is used to map an environment; the progress of point cloud registration techniques, from general purpose algorithms to methodologies that also rely on color information; and different environmental mapping methodologies that concern with environments of many different scales.

In chapter 3, we explain the several stages that compose our work, from extrinsic sensors calibration, going through the pairwise point cloud registration to how loop closures are detected and used in a global optimization stage.

Chapter 4 discusses our experimental procedures and the results obtained from them, both qualitative and quantitative. For this, the walls of a 9.84m$\times$7.13m room were mapped and the results were compared to a map acquired by an industrial scanner. We also used our dataset with an RGB-D SLAM approach, and compare the obtained map to ours.

Finally, chapter 5 presents our conclusions facing experimental results, with remarks to the limitations of our work, and raises possibilities for future works that can address these problems.

Chapter 2

Related Work

We introduce this chapter analyzing the evolution of monocular SLAM. In terms of sensors, this field represents the simplest instantiation of the SLAM problem, as typically only a camera is needed. We then move on to point cloud registration approaches. In the context of this thesis, any point cloud registration can be used in the pairwise alignment, but some approaches have also been proposed having in mind a globally consistent registration. Finally, we discuss some environmental mapping techniques meant to handle regions of different sizes, some of which also use *RGB-D* sensors.

2.1 Monocular SLAM

One of the most intuitive ways to address the localization problem is to use visual clues and associate them to known locations. This is something we humans do routinely, and has inspired researchers from several fields for many years now. In particular, computer vision researchers have taken one step further and have also used vision for mapping purposes [Matthies et al., 1989; Beardsley et al., 1997], instead of just localizing the sensor in space.

Vision-based approaches for the problems of obstacle detection and simultaneous localization and mapping are of practical importance to robotics. As will be discussed further, there are many vision based methodologies that can be used to aid mobile robot navigation in unknown environments, and only require consumer level computational power. Furthermore, cameras have large fields of view, and their cost is significantly lower when compared to range sensors.

Such advantages were taken into account when the first general purpose mobile robot was built. Although it was designed several decades ago, Shakey [Nilsson, 1984] featured a camera for both localizing itself and detecting obstacles. Given the complexity of the localization and obstacle detection problems, the environment to be explored by Shakey was designed to facilitate these tasks.

The basic idea behind Shakey's vision system was that, with a camera capturing the ground, the robot should be capable of telling which pixels of the incoming images belonged to the ground and which were part of obstacles. These ideas have been employed by several other methodologies on mobile robotics ever since [Ulrich and Nourbakhsh, 2000; Kim et al., 2006; Neto et al., 2011].

With the gradual advances both in computational power and computer vision techniques, it was not until recently that real time mapping approaches were proposed. Einhorn et al. [2007] achieved this by estimating the 3D coordinates of image features by using epipolar properties between features contained in consecutive image frames. By using the robot's odometry, it is possible to estimate the epipolar line in which corresponding features should lie between the frames being matched. The estimated feature coordinates are then filtered by an extended Kalman Filter (EKF), which reduces the noise introduced by uncertainties related to the odometry and the feature matching process, while still being able to run in about 40 to 50 frames per second with hardware available at the time of the original publication. Their methodology was later improved by taking into account the changes of features descriptors due to camera motion, and by estimating the camera pose with a particle filter [Einhorn et al., 2009].

Davison et al. [2007] also showed that it is possible to perform real-time, drift free localization and mapping with high quality visual features in small environments. In that work, a full covariance EKF is used to keep track of a set of high quality features. Since their Kalman Filter state contains information from all features, it represents the spatial correlation between distinct features, and contributes to drift-free environment navigation. However, it has an associated $O(n^2)$ computational complexity (where n is the number of features being tracked), which makes it only feasible to small scale environments.

Another important work regarding 3D feature tracking is the *PTAM* (short term for *Parallel Tracking and Mapping*), published by Klein and Murray [2007]. It separates mapping from localization, enabling the use of thousands of image features for mapping purposes. This way, the mapping task runs at a lower frame rate than the localization task, which only relies on a small set of rich features and is expected to perform on real time. Newcombe and Andrew [2010] relied on this approach to generate a dense surface reconstruction of small environments. Their methodology divides the scene into several key frames, and for each one of them, calculates a dense depth map by using neighboring frames. The depth maps are then registered and polygonized, with the final output being a globally consistent mesh model. One problem with that approach is that there can be gaps between neighboring key frames, which would result in holes in the final model. This issue could be mitigated by increasing the overlapping region between key frames, but this would ultimately lead to higher computational requirements.

As computation time became a lesser issue, researchers' attentions were driven towards the representation of the final map. In a more recent publication, the methodology by Einhorn et al. [2009] was used to build a voxel representation of the world explored by the robot [Einhorn et al., 2010]. Since one of its goals was to aid robot navigation, the later work introduced an attention-driven feature selection scheme in which features are only extracted from regions of interest on the input images. Those ROIs are chosen in such a way to favor voxels whose occupancy status are unknown. The feature selection process is also guided by the path planner module, since regions where the robot is headed towards have a higher priority over peripheral regions.

Despite the popularity of voxel-based map representation in the robotics field, other fields such as computer graphics may benefit from finer representations of objects. One such representation is the Point Cloud, which may be extracted from a set of images by estimating the 3D coordinates of all pixels of an arbitrary image. The estimation of depth from camera motion is already a well established field of research, with significant results being reported as far back as a few decades ago [Matthies et al., 1989; Harris and Pike, 1988; Beardsley et al., 1997; Fitzgibbon and Zisserman, 1998]. However, it was not until recently that similar methodologies were designed to run in real time – not only because of the increase in computational power of modern computers, but also due to the recent advances on feature detectors and descriptors, which enabled faster and more robust methodologies [Rosten and Drummond, 2005; Agrawal and Konolige, 2008; Bay et al., 2008].

To the best of our knowledge, the current state of the art in monocular SLAM for producing dense maps is the work published by Newcombe et al. [2011]. Their methodology estimates a dense 3D map of a scene, and immediately uses it for camera pose tracking – which is achieved in real time with a GPGPU implementation. Among the advantages behind dense matching is the fact that it makes camera localization robust to motion blurring and image defocus. Although it has been shown to perform better than previous works, which were based on the *PTAM* feature tracker, their methodology requires several comparison frames for each key frame in order to generate high quality depth maps. This means that, in order for high quality depth maps to be produced, the camera has to be swung around each keyframe, constraining the camera motion.

In general, one can observe that visual SLAM provides good localization estimates with a relatively low computational cost when sparse features are used. Despite not being sophisticated enough to provide detailed environment maps under any circumstance, visual information can already be used for real time accurate localization in environments from rooms to the corridors of a grocery store. The advances made in this field throughout the years also present great value to multi sensor SLAM approaches such as the one proposed in this thesis.

2.2 Point Cloud Registration

The point cloud registration problem can be defined as: Given two point clouds with an overlapping region, find a transformation that will combine them into a unique and more complete point cloud by matching their common points. This formulation can be extended to a set with an arbitrary number of point clouds, provided that the overlap constraint is met. According to Brown [1992], the registration problem can be classified according to the following taxonomies:

- **Multimodal Registration.** When a scene is scanned by different types of sensors. Integrating data from different sensors can be very useful in medical applications, since different scanners are generally meant for specific purposes. It is also used in interactive platforms, such as the one described by Takeuchi et al. [2011].

- **Template Registration.** Some applications require the registration of a prior reference image with a scanned image or point cloud. This is common on face tracking and reconstruction algorithms [Blanz and Vetter, 1999].

- **Viewpoint Registration.** Consists of registering several scans obtained by the same sensor from different poses. This is mostly common on mobile robotics, when an exploring robot has the task to both build a map of the environment and localize itself. Notice that some robots might have several scanning devices (for instance, laser range finders and stereo cameras), and multi modal registration is also performed in these cases.

- **Temporal Registration.** Occurs when one tries to register scans from a particular scene that were obtained at very large time intervals. This is particularly useful for keeping track of structural evolution in the scene.

Our work can be classified in the viewpoint registration category, since a single RGB-D sensor will be used to reconstruct the environment after its exploration. In the context of our work, which divides the pairwise alignment into coarse and fine stages, the methodologies discussed below can be used for fine alignment between point clouds.

One of the most popular registration algorithms is the Iterative Closest Points (ICP), originally proposed by Besl and McKay [1992]. During each iteration, this algorithm finds the correspondences between the point clouds and computes the affine transformation that minimizes the RMS error given by the distances between corresponding points. The algorithm stops either when the RMS error is below an arbitrary threshold or when it reaches a maximum number of iterations. Although the ICP is proven to converge, it is susceptible to convergence on local minima. This means that, in cases where the displacement between the point clouds is not small, the algorithm requires an initial transform estimation in order to converge to the global minimum. Since its original publication, the ICP algorithm has received significant improvements from the community, the most important being summarized by Rusinkiewicz and Levoy [2001]. These improvements concern at least with one of the following stages of the ICP method:

- Sub-sampling of the point clouds (improves computational performance);
- Matching the corresponding points (affects convergence rate);
- Weighting the correspondences between points (for robustness with respect to noise in the point clouds, which helps to avoid convergence on local minima);
- Rejecting spurious matches (for outliers removal, and helps avoid convergence on invalid minima);
- Specification of the error metric (related to both convergence speed and alignment quality);
- Optimization method for minimizing the error metric (usually dependent on the error metric).

For the cases where the point clouds have a large intersecting region, the methodology proposed by Hyun et al. [1998] is able to find a coarse alignment between them with a low computational cost. Their proposal performs the Principal Component Analysis on both point clouds, and finds the rotation that aligns their orthonormal vectors. The translation between the point clouds is then found as the difference of their centers of mass. After this process, a fine alignment process, such as the ICP, can be applied. Their approach has two major drawbacks: It requires the point clouds to have large overlapping regions, and the resulting vectors from the PCA might be flipped on symmetric point clouds, which would lead the methodology to an invalid rotation during the coarse alignment process.

The search for faster and more robust registration algorithms led to the formulation of the Normal Distributions Transform (NDT) representation [Biber and Strasser, 2003]. Their representation consists of a discretized space similar to the Occupancy Grid [Elfes, 1989], with the difference that instead of representing the probability of a region being occupied, each bin represents the probability that the sensor would sample an arbitrary point from it. The original proposal by Biber and Strasser [2003] also shows that this representation can be used to align multiple range scans for SLAM purposes, as long as there are non-redundant 2D features within the reach of the sensor. Because it was initially proposed for two dimensional point clouds, a 3D-NDT extension was proposed by Magnusson et al. [2007]. Huhle et al. [2008] also uses the 3D-NDT approach and an RGB sensor along with a registration algorithm that also considers the color of each point on its score function. Their report shows that the additional data contribute to better alignment results.

Unlike the previously discussed registration methodologies, the approach presented by Chen et al. [1998, 1999] focused on registering point clouds by only using three control points. It starts by randomly selecting three linearly independent control points in one of the point clouds. Then, by taking into account rigidity constraints, the algorithm finds all possible corresponding control points on the second point cloud, and assigns a fitness function to each set of correspondences. The set of matches with the best fitness is then used to calculate the transformation between the point clouds, which may be further refined with an algorithm such as the ICP. One advantage of their method is that it doesn't require an initial transform estimate. However, in order to deal with noisy point clouds, the distance between the control points has to be increased, which increases the search space for correspondences, affecting the computational performance of the algorithm.

A more general approach to the point cloud registration needs to take into account the deformability of certain entities, which is known in the related literature as non-rigid registration [Feldmar and Ayache, 1994]. Some of the oldest works on this field were more concerned with the low frequency deformations created by sensor issues, such as miscalibration [Ikemoto, 2003; Brown and Rusinkiewicz, 2004]. Such methods usually work by decomposing the scanned volume into several patches that are locally registered in a rigid fashion, although different patches are independently registered. Besides pairwise registration methods, globally consistent alignment algorithms have also been proposed based on existing methodologies [Brown and Rusinkiewicz, 2007; Li et al., 2008].

In the present work, we assume that the scene being mapped doesn't contain deformable objects, and that geometry warping due to sensor calibration errors is negligible. Therefore, deformations in the point clouds are assumed to be caused by sensor noise.

2.3 RGB-Based Point Cloud Registration

Since its release in late 2010, the Kinect sensor has been widely used for numerous research purposes worldwide. In particular, several SLAM methodologies have been studied due to the direct impact of this problem on the robotics field.

The first published SLAM methodologies for RGB-D sensors focused on the use of visual features for both registering pairwise point clouds as well as for post-processing the aligned clouds with a loop closure algorithm. Henry et al. [2010] also used a modified version of the ICP algorithm that takes into account the color of each point when calculating the fitness of each iteration. Their algorithm is also referred to as *RGB-D ICP*.

The strategy of using both depth and colour information for registration purposes was explored in a different fashion by Steinbruecker et al. [2011]. Given two point clouds with their corresponding RGB values, and an estimate of the transformation matrix between them, their methodology renders the second point cloud into a frame aligned with the pose of the first frame, and compares the resulting image with the corresponding image of the first point cloud. If the frames are closely aligned, these images should overlap each other with a small pixel-wise difference. Therefore, their work seeks the alignment transformation that minimizes that difference. It is worth noting, however, that their algorithm is highly prone to convergence on local minima when the point clouds were not captured at very close poses.

Stückler and Behnke [2012] developed an RGB-D registration technique similar to the 3D-NDT, in which a subset of the point cloud was inserted in an octree. In their approach, each node at every level contains a mean and covariance matrix of points. These information are then used to estimate the normal to the corresponding surfel at each octree node. The registration consists, therefore, in finding the corresponding surfels between pairs of octrees that maximizes their matching likelihood, from which the transformation matrix between the aligned point clouds is obtained. Their methodology has been shown to perform with a similar error as that proposed by Steinbruecker et al. [2011], but is more stable when the displacement between the point clouds is relatively large.

Regarding feature-based registration, the work published by Huang et al. [2011] presented a valuable insight on outlier rejection. Given two images $\mathbf{I}_a$ and $\mathbf{I}_b$ (containing n and m features, respectively), they modeled the feature matching problem by a graph, in which each vertex represents a match between one feature from $\mathbf{I}_a$ and one feature from $\mathbf{I}_b$ (i.e. there are $n \times m$ vertices). In this graph, edges only connect vertices that do not violate euclidean constraints. For instance, the graph vertex representing a match between points a_x and b_x (from $\mathbf{I}_a$ and $\mathbf{I}_b$, respectively) would only be connected to a graph vertex representing a match between a_y and b_y if $||a_x - a_y|| \approx ||b_x - b_y||$, or $||a_x - a_y|| - ||b_x - b_y|| \leq \epsilon$. With such model, the problem of inliers detection is equivalent to finding the maximum clique in this graph. Since the maximum clique problem is NP-hard [Garey and Johnson, 1990], the authors proposed a greedy approximate algorithm to solve this problem with quadratic complexity. Their algorithm was capable of producing better results than RANSAC [Fischler and Bolles, 1981].

Although the use of visual information together with depth data helps to improve the quality of 3D alignment, mapping systems based on these data may have their kinematics significantly constrained. Typically, the camera has to move along the environment with limited linear and angular velocities, or image blurring will compromise the quality of the detected features; in addition, the number of correspondences between consecutive frames tends to decrease with higher velocities. Furthermore, visual features may be lacking under some lighting conditions, which is a key issue for applications such as prospection of uninhabited environments.

The work presented by des Bouvrie [2011] was the first published attempt to integrate RGB-D and inertial data to solve the SLAM problem with kinect-style depth sensors. Heavily inspired by the work by Henry et al. [2010], it performs the fusion of inertial and visual data after a coarse alignment is computed using visual features only. However, their work depends on the difference between the estimated attitudes from both the IMU and the keypoint matching process, discarding IMU data if the difference is too large. This meas that only closely redundant information are fused, leaving the IMU unused in cases where it could have been helpful, such as during high angular velocities. Also, the loop closure problem was not addressed by their work.

The thesis is inspired by previous methodologies that seek to preserve euclidean constraints when matching point cloud keypoints, but we make use of color, geometric and inertial information to achieve our goal. Unlike the work by des Bouvrie [2011], which uses an IMU, we use a MARG sensor. Assuming that the direction of the magnetic field in the environment being mapped remains unchanged during the point cloud acquisition, the MARG gives us the advantage of a negligible attitude drift. This allows us to use inertial data during the loop closure detection. Also, we use inertial data to reduce the alignment process itself to the problem of finding only a translation between point clouds.

2.4 3D Environmental Mapping

One of the most common purposes for all the previously mentioned techniques is environmental mapping. Commonly, high level methodologies combine several alignment algorithms in a coarse-to-fine fashion in order to obtain high quality and robust alignments between point clouds, to ultimately produce a digital map that is faithful to the scanned environment.

A real-time 3D reconstruction approach has been recently demonstrated by Izadi et al. [2011], with the KinectFusion approach. It consists of a pipeline with three main stages. In the first one, a GPU based implementation of a fast variant of the ICP algorithm detects the changes in sensor pose between two consecutive grabbed frames. Then, the global map represented by a three dimensional signed distance function (SDF) is updated. Finally, the latest frame is globally aligned to the existing map, which has been shown to reduce the effects of pose drift.

Due to the nature of the SDF, mapping a scene typically required large amounts of GPU memory (a cubic volume of size 512^3 voxels would require at least 1GiB). Therefore, their methodology may only be used to map small environments, such as a small room with a few cubic meters.

Considering this limitation, Whelan et al. [2012] further improved the KinectFusion approach, allowing it to map scenes with arbitrary sizes. This was accomplished by allowing the center of the SDF to move according to the sensor position, while data that would fall outside this volume is transformed into a polygonal mesh. Since this mesh requires significantly less storage than the SDF maps, larger environments may be generated in real time without storage being considered a bottleneck.

Bosse and Zlot [2009] presented a strategy for large environment mapping based on an ICP variation to register range measurements obtained by a spinning laser mounted on a vehicle. In their work, the registration of point clouds obtained by a sensor revolution, referred to as *sweep*, are locally aligned in several iterative processes that take into account the sensor trajectory during the data acquisition. Finally, the scans are processed by an optimization framework that generates a globally consistent map [Bosse and Zlot, 2010]. That work represents a significant advance on the field of outdoor mapping; however, its global optimization framework restricted the vehicle kinematics to movements in a 2D plane, with no pitch and roll angles being supported.

These shortcomings were addressed on the development of the *Zebedee* handheld scanner [Bosse et al., 2012]. Their platform consists of a 2D time-of-flight laser scanner rigidly attached to an IMU, with both suspended by a loose string whose motion allowed the sensor to sweep across the environment. The use of a long range scanner (that could reach up to 30m) has allowed their platform to map environments with scales of tens of meters with an RMS error in the order of a few centimeters. However, the used sensors may be too expensive for this platform to be used by the general public. Also, it lacks color information, which limits the range of applications that could benefit from it.

Similarly to the work by [Bosse et al., 2012], our methodology is not concerned with the representation of the final map. Both works produce a globally consistent point cloud and the sensor trajectory. These data can be further processed in order to extract a polygonal mesh, proper for visualization purposes, or an occupancy grid, adequate for mobile robot localization.

Chapter 3

Methodology

We approach the mapping problem by defining the global, fixed frame as the same reference frame of the first available point cloud. If pairwise registration is successfully achieved for all point clouds, it is possible to represent any point with respect to the reference frame. Since pairwise registration is prone to accumulated error, a global optimization step is required to increase global consistency of the final map.

3.1 Extrinsic Calibration of the Sensors

In order to fuse data coming from the RGB-D and MARG sensors, it is necessary to know the rigid rotation that transforms the local frame of the MARG sensor to the local frame of the depth sensor, which we will denote as ${}^{k}_{m}\mathbf{R}$.

Since our sensors are firmly attached to each other, the rotation ${}^{k}_{m}\mathbf{R}$ will remain constant as long as the local frames of the sensors do not change locally.

It is important to note that resultant forces applied to the MARG sensor may induce disturbances in its attitude estimation, effectively causing an effect similar to changing the relative attitude between the sensors. Such disturbances can also be caused by the presence of ferromagnetic materials close to the MARG sensor. This issue is mitigated by a filter that combines all data captured by this sensor, which is explained in Section 3.2.1.

The extrinsic calibration process aims to estimate the transformation ${}^{k}_{m}\mathbf{R}$ in an environment free from magnetic interference, provided the resultant force applied to the sensors is null. The relation between the attitudes of the sensors is such that

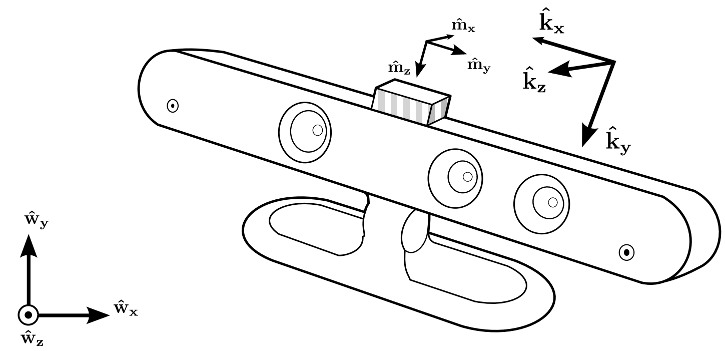

$\hat{\mathbf{m}}_\mathbf{x}$
$\hat{\mathbf{m}}_\mathbf{y}$
$\hat{\mathbf{m}}_\mathbf{z}$
$\hat{\mathbf{k}}_\mathbf{x}$
$\hat{\mathbf{k}}_\mathbf{z}$
$\hat{\mathbf{k}}_\mathbf{y}$
$\hat{\mathbf{w}}_\mathbf{y}$
$\hat{\mathbf{w}}_\mathbf{x}$
$\hat{\mathbf{w}}_\mathbf{z}$

Figure 3.1: We deal with points and vectors specified in three different frame coordinate systems: World fixed $\hat{\mathbf{w}}$, MARG local $\hat{\mathbf{m}}$ and depth sensor local $\hat{\mathbf{k}}$.

$$\begin{aligned} {}^{w}\mathbf{v} &= {}^{w}_{m}\mathbf{M}_t \, {}^{m}\mathbf{v}_t = {}^{w}_{k}\mathbf{K}_t \, {}^{k}\mathbf{v}_t, \\ {}^{k}_{m}\mathbf{R} \, {}^{m}\mathbf{v}_t &= {}^{k}\mathbf{v}_t = {}^{w}_{k}\mathbf{K}_t^{-1} \, {}^{w}_{m}\mathbf{M}_t \, {}^{m}\mathbf{v}_t, \end{aligned}$$

where ${}^{w}\mathbf{v}$, ${}^{m}\mathbf{v}_t$ and ${}^{k}\mathbf{v}_t$ represent a hypothetical point expressed in the three different frame coordinates (world, MARG and depth sensor, respectively) at time t. Also, ${}^{w}_{m}\mathbf{M}$ is the attitude of the MARG sensor with respect to the world, and ${}^{w}_{k}\mathbf{K}$ the attitude of the RGB-D sensor with respect to the world. The three reference frames are illustrated by Figure 3.1. The lack of a subscript t in the world vertex ${}^{w}\mathbf{v}$ reflects our assumption that the world remains static during the calibration process. We can see from this relation that, in order to obtain an estimation of ${}^{k}_{m}\mathbf{R}$, it suffices to know the attitudes of the sensors with respect to a fixed world frame at any given time – that is, ${}^{w}_{m}\mathbf{M}_t$ and ${}^{w}_{k}\mathbf{K}_t$.

We define the world frame such that its $\hat{\mathbf{w}}_\mathbf{z}$ axis is parallel to the gravity field and its $\hat{\mathbf{w}}_\mathbf{x}$ axis is parallel to the orthogonalized magnetic field ${}^{w}\hat{\mathbf{m}}$, whose orthogonalization is made with respect to the vector $\hat{\mathbf{w}}_\mathbf{z}$, as shown by Figure 3.2.

An object attitude is expressed by a rotation matrix whose columns denote this object's local orthonormal axes. Equation 3.1 shows how the frames illustrated by Figure 3.1 can be arranged to express ${}^{w}_{m}\mathbf{M}$ and ${}^{w}_{k}\mathbf{K}$. This property can be explored for estimating ${}^{k}_{m}\mathbf{R}$, provided all sensors can obtain an estimate of, at least, two of these axes.

$$\begin{aligned} {}^{w}_{m}\mathbf{M} &= [{}^{w}\hat{\mathbf{m}}_\mathbf{x} \quad {}^{w}\hat{\mathbf{m}}_\mathbf{y} \quad {}^{w}\hat{\mathbf{m}}_\mathbf{z}] \\ {}^{w}_{k}\mathbf{K} &= [{}^{w}\hat{\mathbf{k}}_\mathbf{x} \quad {}^{w}\hat{\mathbf{k}}_\mathbf{y} \quad {}^{w}\hat{\mathbf{k}}_\mathbf{z}] \end{aligned} \tag{3.1}$$

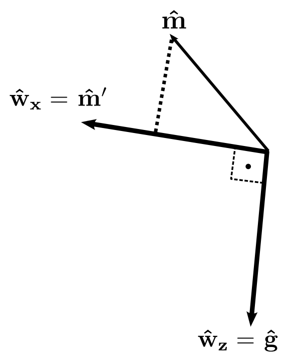

$\hat{\mathbf{m}}$
$\hat{\mathbf{w}}_\mathbf{x} = \hat{\mathbf{m}}'$
$\hat{\mathbf{w}}_\mathbf{z} = \hat{\mathbf{g}}$

Figure 3.2: The axis $\hat{\mathbf{w}}_\mathbf{x}$ is obtained by orthogonalizing the magnetic field vector ${}^w\hat{\mathbf{m}}$ with respect to $\hat{\mathbf{w}}_z$.

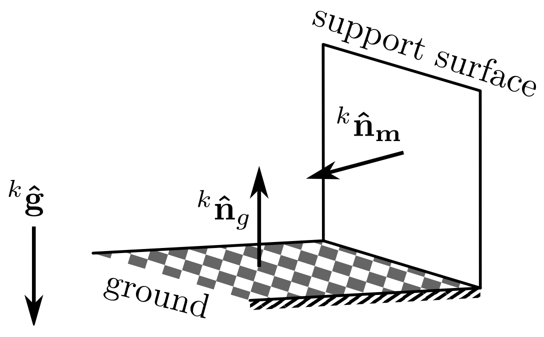


Figure 3.3: Surfaces used during the calibration process.

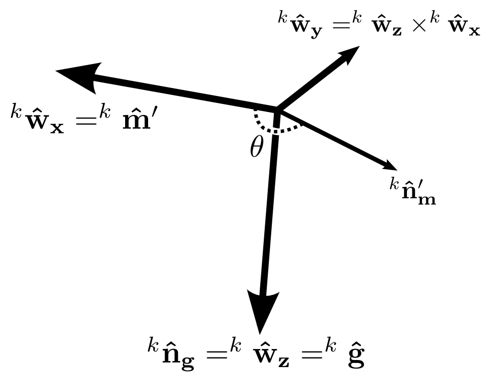
This can be easily achieved with the MARG, since it is able to output both ${}^m\hat{\mathbf{g}}$ and ${}^m\hat{\mathbf{m}}$. Thereon, the attitude of the **world** with respect to the MARG sensor is given by

$${}^w_m\mathbf{M}^{-1} = [{}^m\hat{\mathbf{w}}_\mathbf{x} \; {}^m\hat{\mathbf{w}}_\mathbf{y} \; {}^m\hat{\mathbf{w}}_\mathbf{z}],$$

where ${}^m\hat{\mathbf{m}}_\mathbf{x}$ is the ortho normalized magnetic field in the MARG frame, ${}^m\hat{\mathbf{w}}_\mathbf{z}$ is the normalized gravity (also specified in the MARG frame) and ${}^m\hat{\mathbf{w}}_\mathbf{y} ={}^m \hat{\mathbf{w}}_\mathbf{z} \times^w \hat{\mathbf{w}}_\mathbf{x}$.

The depth sensor, however, has no direct way of measuring either the gravity nor the magnetic field of the environment. It is still possible to estimate these vectors in the local frame of this sensor if they are in some manner associated with the geometric features of the world. That can be accomplished by scanning a planar surface orthogonal to the gravity field (whose normal, ${}^k\hat{\mathbf{n}}_\mathbf{g}$, will be parallel to ${}^k\hat{\mathbf{g}}$), and another planar surface non parallel to the previous one, which we will refer to as the support surface. Both surfaces are illustrated by Figure 3.3.

After orthogonalizing the normal of the support surface, ${}^k\hat{\mathbf{n}}_\mathbf{m}$, with respect to $\hat{\mathbf{n}}_\mathbf{g}$, we obtain the vector ${}^k\hat{\mathbf{n}}'_\mathbf{m}$. As is illustrated by Figure 3.4, this vector can be thought of as ${}^k\hat{\mathbf{w}}_\mathbf{x}$ after a rotation of a constant angle θ around the axis ${}^k\hat{\mathbf{n}}_\mathbf{g}$ (which is the orthonormalized magnetic field, with respect to the RGB-D sensor's frame):



${}^k\hat{\mathbf{w}}_{\mathbf{y}} ={}^k \hat{\mathbf{w}}_{\mathbf{z}} \times{}^k \hat{\mathbf{w}}_{\mathbf{x}}$
${}^k\hat{\mathbf{w}}_{\mathbf{x}} ={}^k \hat{\mathbf{m}}'$
θ
${}^k\hat{\mathbf{n}}'_{\mathbf{m}}$
${}^k\hat{\mathbf{n}}_{\mathbf{g}} ={}^k \hat{\mathbf{w}}_{\mathbf{z}} ={}^k \hat{\mathbf{g}}$

Figure 3.4: World coordinate frame with respect to the depth sensor during the calibration process. The axis ${}^k\hat{\mathbf{n}}_{\mathbf{g}}$ is estimated from a planar surface orthogonal to the gravity, and the axis ${}^k\hat{\mathbf{n}}'_{\mathbf{m}}$ is obtained from orthogonalizing the normal of the support surface with respect to ${}^k\hat{\mathbf{n}}_{\mathbf{g}}$.

$$ {}^k\hat{\mathbf{n}}'_{\mathbf{m}} = R_{\theta,{}^k\hat{\mathbf{n}}_{\mathbf{g}}}\,{}^k\hat{\mathbf{w}}_{\mathbf{x}}, $$

where the notation $R_{\alpha,\mathbf{v}}$ specifies a rotation of an angle α about an arbitrary axis $\mathbf{v}$. Thus, the attitude of the **world** with respect to the depth sensor can be expressed by

$$ \begin{aligned} {}^w_k\mathbf{K}^{-1} &= [{}^k\hat{\mathbf{m}}' \quad {}^k\hat{\mathbf{g}} \times{}^k \hat{\mathbf{m}}' \quad {}^k\hat{\mathbf{g}}], \\ {}^k\hat{\mathbf{m}}' &= R^{-1}_{\theta,{}^k\hat{\mathbf{n}}_{\mathbf{g}}}\,{}^k\hat{\mathbf{n}}'_{\mathbf{m}}. \end{aligned} \tag{3.2} $$

Notice that it is still necessary to know the angle θ in order to calculate the rotation ${}^w_k\mathbf{K}$. In order to find this angle, we must consider the nature of the calibration matrix ${}^k_m\mathbf{R} = {}^w_k\mathbf{K}^{-1}\,{}^w_m\mathbf{M}$. Since it represents a constant rigid transformation, we can say that ${}^k_m\mathbf{R} = {}^w_k\mathbf{K}_t^{-1}\,{}^w_m\mathbf{M}_t$, regardless of the time instant t. Therefore, θ can be found as the solution to the non-linear system given by

$$ {}^w_k\mathbf{K}_{t_i}^{-1}\,{}^w_m\mathbf{M}_{t_i} = {}^w_k\mathbf{K}_{t_j}^{-1}\,{}^w_m\mathbf{M}_{t_j} \;\; \forall \;\; t_i \neq t_j. \tag{3.3} $$

In order to find a solution for this system, the measurements must be made with the sensor fixed at several distinct attitudes, to prevent the system from becoming ill-conditioned (in which case any value of θ would be a solution). Furthermore, since the sensors are subject to measurement noise, it is unlikely that this system will have an exact solution. Therefore, we approach it as a minimization problem of the form given by

$$\theta = \underset{\alpha}{\operatorname{argmin}} \sqrt{\sum_{t_i} \sum_{t_j \neq t_i} \| {}^w_k\mathbf{K}_{t_i}(\alpha)^{-1}\, {}^w_m\mathbf{M}_{t_i} - {}^w_k\mathbf{K}_{t_j}(\alpha)^{-1}\, {}^w_m\mathbf{M}_{t_j} \|^2}, \tag{3.4}$$

where $\mathbf{K}_t(\alpha)$ gives the attitude of the depth sensor with respect to the world at instant t, assuming that the angle θ from Equation 3.2 is α.

After we have an estimate for θ by solving Equation 3.4 (which we computed by using the Nelder-Mead simplex, as it is a derivative-free method), it is possible to calculate the calibration matrix. Since we can compute an estimation of ${}^k_m\mathbf{R}$ from each measurement made by the depth sensor, we have to combine all these estimates into a single, reliable estimate of ${}^k_m\mathbf{R}$.

For each one of the captured attitudes, we have ${}^k_m\mathbf{R}_1 = {}^w_k\mathbf{K}_1^{-1}\, {}^w_m\mathbf{M}_1$, ${}^k_m\mathbf{R}_2 = {}^w_k\mathbf{K}_2^{-1}\, {}^w_m\mathbf{M}_2$, $\ldots$, ${}^k_m\mathbf{R}_{tn} = {}^w_k\mathbf{K}_{tn}^{-1}\, {}^w_m\mathbf{M}_{tn}$. The final rigid rotation between the sensors is estimated as the average of all these rotations. Our approach to the rotation averaging problem consists of firstly converting all rotation matrices to quaternions. Due to the fact that two unit quaternions $\mathbf{q}$ and $-\mathbf{q}$ represent the same rotation, a naïve averaging by adding and normalizing quaternions would be subject to spurious results. Nevertheless, this can be avoided if the average is made between quaternions that have an angular distance shorter than or equal to 90°, which is given by their dot product. Finally, we perform a spherical linear interpolation (SLERP) [Shoemake, 1985] in an incremental fashion similar to the running average algorithm, which will produce the final averaged rotation, which we convert back to the matrix form (`quat2dcm`). This process is illustrated by Algorithm 1.

Algorithm 1 Rotation Averaging

```
Require:
 1: A set of quaternions $\mathbf{q}_1, \mathbf{q}_2, \ldots, \mathbf{q}_n$ representing several noisy estimates of the rigid
    rotation ${}^k_m\mathbf{R}$ between the MARG and the depth sensor.
Ensure:
 1: The average rotation matrix ${}^k_m\mathbf{R}$.

 1: $\mathbf{q}_f \leftarrow \mathbf{q}_1$
 2: for $i \leftarrow 2, n$ do
 3:     if $\mathbf{q}_f \cdot \mathbf{q}_i < \mathbf{q}_f \cdot -\mathbf{q}_i$ then
 4:         $\mathbf{q}_i \leftarrow -\mathbf{q}_i$
 5:     end if
 6:     $\mathbf{q}_f \leftarrow$ SLERP($\mathbf{q}_f$, $\mathbf{q}_i$, $\frac{1}{i}$)
 7: end for
 8: ${}^k_m\mathbf{R} \leftarrow$ QUAT2DCM($\mathbf{q}_f$)
```

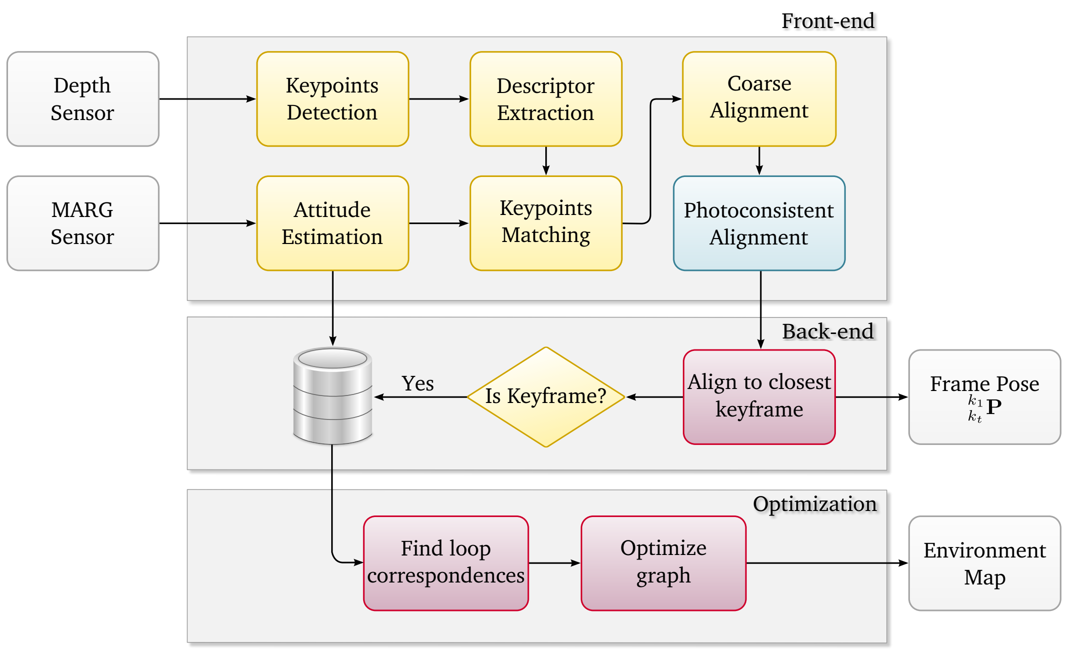


Figure 3.5: A high-level overview of the proposed registration approach. The front-end is divided into three pairwise registration levels. Their alignment result is refined to a key frame level by a backend, which saves all locally consistent poses. These poses are later processed by a SLAM optimizer, which generate the globally consistent final map.

3.2 Point Cloud Registration

We propose a registration technique that consists of a coarse-to-fine alignment pipeline. This decision allows us to compute high quality transformation between adjacent point clouds with a small computational impact, since different pipeline stages can be parallelized with a producer-consumer design. Also, the transformations found by coarser levels reduce the number of iterations required by finer levels to find their optimal solutions. The diagram on Figure 3.5 depicts a high level view of our approach.

The proposed algorithm starts by performing the keypoints detection and descriptor extraction for the first point cloud received. Then, for each new point cloud captured by the depth sensor, the whole pipeline is executed, whilst the alignment processes are performed between the latest two point clouds in a pairwise fashion. Each alignment level will be described in detail in the following subsections.

3.2.1 Front-end

In the first alignment stage, our methodology uses inertial, color and geometric information to find an initial estimation for the rigid transformation between the two latest point clouds. We use the orientation filter proposed by Madgwick et al. [2011] to fuse data coming from sensors in a MARG device. Their methodology uses the gyroscope, accelerometer and magnetometer to estimate the MARG attitude with respect to a fixed world frame specified by the gravity and earth's magnetic field. We have chosen this approach due both to its stability and low computational requirements.

In an independent step, we find a set of SURF keypoints [Bay et al., 2008] from the most recent image. We have chosen the SURF detector based on the fact that it was capable of extracting several hundreds of good quality features from images of our experiment, even during the presence of some blurring due to camera motion. We also label these keypoints with BASE descriptors [Nascimento et al., 2012]. This descriptor is computed by using both intensity and geometry information, which makes them more robust to variations in lighting conditions and textureless scenes.

3.2.2 Keypoint Matching

In order to quickly match keypoints between a pair of frames, we developed a keypoint matching strategy that takes into account a prior knowledge of the displacement between them. The rotation between these frames can be approximated by a quaternion calculated with data from the MARG sensor, and the translation can be roughly estimated by finding any pair of corresponding keypoints between the two frames being aligned.

Let ${}^{k}_{m}\mathbf{R}$ be the extrinsic MARG/Depth sensor calibration matrix computed as described in the previous section, and ${}^{w}_{m}\mathbf{M}_t$ be the MARG attitude at the t-th frame as reported by the orientation filter. If the sensors were perfectly synchronized and the MARG sensor were not susceptible to noise and interference, the attitude of the t-th frame with respect to the $(t-1)$-th frame, ${}^{k_{t-1}}_{k_t}\mathbf{K}$, would be given by

$${}^{k_{t-1}}_{k_t}\mathbf{K} = ({}^{w}_{m}\mathbf{M}_{t-1}\, {}^{k}_{m}\mathbf{R}^{-1})^{-1}\, {}^{w}_{m}\mathbf{M}_t\, {}^{k}_{m}\mathbf{R}^{-1}.$$

In practice, this equation doesn't yield an exact attitude due to problems such as noise from the MARG sensor, small extrinsic calibration errors and the lack of synchrony between MARG and RGB-D sensor. Furthermore, it only accounts for rotation, leaving the translation unknown. We tackle these issues by combining the estimated attitude with a maximum sensor motion speed assumption. Given a keypoint ${}^{k_{t-1}}\mathbf{k}$ on the $(t-1)$-th frame, its corresponding match on frame t is likely to lie within a spherical shell defined by the parametric equation

$$({}^{k_{t-1}}_{k_t}\mathbf{K})^{-1}\ {}^{k_{t-1}}\mathbf{k} + \hat{\mathbf{r}} v_{\max} dt,$$

where the parameter $\hat{\mathbf{r}}$ is a unit vector, $v_{\max}$ is the maximum motion speed assumed, and dt is the elapsed time separating the acquisition of those frames.

Considering the maximum speed assumption, our matching approach starts by selecting a subset of keypoints from the $(t-1)$-th frame in order to keep the computational requirements low. This procedure, which we call SUBSAMPLE on Algorithm 2, is done by selecting the N most prominent keypoints, where keypoints are compared according to their response to the SURF detector. Then, for each keypoint, we search for correspondences on the t-th image. To compute the set of matching candidates from the t-th frame, we perform a radius search for keypoints around a center given by $({}^{k_{t-1}}_{k_t}\mathbf{K})^{-1}\ {}^{k_{t-1}}\mathbf{k}$ and radius length of $v_{\max} dt$. On Algorithm 2, the radius search operation is presented as RADIUSSEARCH(KDTREE, CENTER, RADIUS). Each candidate from the returned set has an associated vector $\mathbf{t}$, which represents a possible translation between the two point clouds. These keypoints are referred to as *support keypoints*, and are connected by a red line in Figure 3.6.

We proceed by comparing all combinations between an arbitrary ${}^{k_{t-1}}\mathbf{k}_i$ and its corresponding results from the radial search. For each pairing, we have a different $\mathbf{t}$ that, together with the attitude guess from the MARG sensor, provide an estimation of the pose of point cloud t with respect to point cloud $t-1$. Then, in order to refine this estimate, for all remaining keypoints ${}^{k_{t-1}}\mathbf{k}_l$ (with $l \neq i$), we perform a nearest neighbor search in the t-th frame around the point given by $({}^{k_{t-1}}_{k_t}\mathbf{K})^{-1}\ {}^{k_{t-1}}\mathbf{k}_l + \mathbf{t}$ – we have found that three neighbors is typically a good compromise between computational performance and alignment quality. The neighbor that best matches the current keypoint is then defined as its pair. Once we have a correspondence for every keypoint subsampled from the $(t-1)$-th frame, we calculate the *Hamming distance* between the binary descriptors of the matched keypoints. The distances are used to calculate a score of the current pairing configuration. We repeat this process from the beginning for all possible ${}^{k_{t-1}}\mathbf{k}_i$, and chose the set of pairing configurations with the smallest total score, which is computed by the summation of all pair scores.

Finally, we estimate the coarse transformation from the configuration with the best score with the Principal Component Analysis (PCA) [Pearson, 1901] of all pairings. The rotation is given by performing an eigenvalue decomposition of the covariance matrix of the keypoints coordinates. The translation is computed from the difference between the centers of mass of each cloud of keypoints. Algorithm 2 illustrates the coarse alignment procedure.

Algorithm 2 Keypoint Matching

```
Require:
 1: Two sets of keypoints ${}^{k_{t-1}}\mathbf{k} = \{{}^{k_{t-1}}\mathbf{k}_1, \ldots, {}^{k_{t-1}}\mathbf{k}_n\}$ and ${}^{k_t}\mathbf{k} = \{{}^{k_t}\mathbf{k}_1, \ldots, {}^{k_t}\mathbf{k}_m\}$ from
    the $(t-1)$-th and $t$-th frames, respectively, with their corresponding descriptors.
 2: The attitude of the depth sensor at the instant $i$ with respect to its orientation at
    the moment $t-1$, ${}^{k_{t-1}}_{k_t}\mathbf{K}$.
Ensure:
 1: The pose of the depth sensor at the instant $t$ with respect to the instant $t-1$,
    ${}^{k_{t-1}}_{k_t}\mathbf{P}$
 2: The score of the best matching configuration.

 1: ${}^{k_{t-1}}\tilde{\mathbf{k}} \leftarrow$ SUBSAMPLE(${}^{k_{t-1}}\mathbf{k}$)
 2: bestScore $\leftarrow \infty$
 3: for all ${}^{k_{t-1}}\tilde{\mathbf{k}}_l \in {}^{k_{t-1}}\tilde{\mathbf{k}}$ do
 4:     ${}^{k_t}\tilde{\mathbf{k}} \leftarrow$ RADIUSSEARCH(${}^{k_t}\mathbf{k}$, $({}^{k_{t-1}}_{k_t}\mathbf{K})^{-1}\ {}^{k_{t-1}}\tilde{\mathbf{k}}_l$, $v_{\max} \cdot dt$)
 5:     for all ${}^{k_t}\tilde{\mathbf{k}}_l \in {}^{k_t}\tilde{\mathbf{k}}$ do
 6:         score $\leftarrow$ HAMMINGDISTANCE(DESCRIPTOR(${}^{k_{t-1}}\tilde{\mathbf{k}}_l$), DESCRIPTOR(${}^{k_t}\tilde{\mathbf{k}}_l$))
 7:         matches $\leftarrow \{({}^{k_{t-1}}\tilde{\mathbf{k}}_l,\ {}^{k_t}\tilde{\mathbf{k}}_l)\}$
 8:         $\mathbf{t} \leftarrow {}^{k_t}\tilde{\mathbf{k}}_l - ({}^{k_{t-1}}_{k_t}\mathbf{K})^{-1}\ {}^{k_{t-1}}\tilde{\mathbf{k}}_l$
 9:         for all ${}^{k_{t-1}}\tilde{\mathbf{k}}_m \in {}^{k_{t-1}}\tilde{\mathbf{k}} \mid {}^{k_{t-1}}\tilde{\mathbf{k}}_m \neq {}^{k_{t-1}}\tilde{\mathbf{k}}_l$ do
10:             $\mathbf{c} \leftarrow ({}^{k_{t-1}}_{k_t}\mathbf{K})^{-1}\ {}^{k_{t-1}}\tilde{\mathbf{k}}_m + \mathbf{t}$
11:             neighbors $\leftarrow$KNNSEARCH(${}^{k_t}\mathbf{k}$, $\mathbf{c}$, AMOUNT_NEIGHBORS)
12:             ${}^{k_t}\tilde{\mathbf{k}}_m \leftarrow$ BESTNEIGHBOR(neighbors)
13:             hamming $\leftarrow$ HAMMINGDISTANCE(DESCRIPTOR(${}^{k_{t-1}}\tilde{\mathbf{k}}_m$),
                                                   DESCRIPTOR(${}^{k_t}\tilde{\mathbf{k}}_m$))
14:             if $\|\mathbf{c} - {}^{k_t}\tilde{\mathbf{k}}_m\| \leq$ DISTANCE_THRESHOLD then
15:                 score $\leftarrow$ score + hamming
16:                 matches $\leftarrow$ matches $\cup \{({}^{k_{t-1}}\tilde{\mathbf{k}}_m, {}^{k_t}\tilde{\mathbf{k}}_m)\}$
17:             end if
18:         end for
19:         score $\leftarrow$ score/SIZE(matches)$^2$
20:         if bestScore > score then
21:             bestScore $\leftarrow$ score
22:             ${}^{k_{t-1}}_{k_t}\mathbf{P} \leftarrow$ POSEFROMPCA(matches)
23:         end if
24:     end for
25: end for
26: return $({}^{k_{t-1}}_{k_t}\mathbf{P}, \text{score})$
```

Algorithm 2 has a time complexity of $O(p(\sqrt{q} + M)\log q + p^2(M\sqrt{q}))$, where:

- p is the number of keypoints in ${}^{k_{t-1}}\tilde{\mathbf{k}}$ (can be set to a maximum limit depending on the subsampling strategy);
- q is the number of keypoints in ${}^{k_t}\mathbf{k}$;
- $\sqrt{q} + M$ is the time complexity of a radius search over a KD-Tree, where M is the expected number of returned nodes (this term is directly proportional to the translational uncertainty, given by $v_{\max} \cdot dt$);
- $M\sqrt{q}$ is the time complexity of the KNN search on ${}^{k_t}\mathbf{k}$.

In order to keep the computational cost of search operations low, we store the set of keypoints ${}^{k_t}\mathbf{k}$ in a KD-tree. The construction cost, $O(q \log q)$, does not represent a significant impact to the algorithm, since we typically deal with a number of keypoints $q < 2000$, when the SURF detector is employed.

Our keypoint subsampling approach takes into account two major factors. Firstly, we determine a maximum number of keypoints that the set ${}^{k_{t-1}}\tilde{\mathbf{k}}$ might contain based on their detection score. Secondly, we only sample keypoints whose distance to the camera is below a threshold. The reason for this second factor is the fact that farther points tend to be too noisy, which would cause a negative impact towards the pose estimation algorithm. Typically, this threshold is 3.5m for RGB-D sensors.

The main difference between the designed matcher and existing brute force techniques is that we apply a transformation to the keypoint around which a neighbor search will be performed. Therefore, after the transformation, the keypoint will lie closer to its correspondent. Also, the distance threshold used after the KNN search can be set to very tight values in order to preserve euclidean constraints, reducing the need for an explicit outlier removal. It is important to notice that the transformation that precedes the radius search doesn't necessarily need to be a rotation. As we will discuss later on, this transformation will also contain a translation when the matching is made between frames with relatively large displacements (which happens every time on the back-end of our methodology, when it tries to align regular frames to key frames).

3.2.3 Photo Consistent Alignment

In many cases, the coarse alignment estimated by the feature matcher is very close to the optimal transformation between point clouds, with displacements that could be

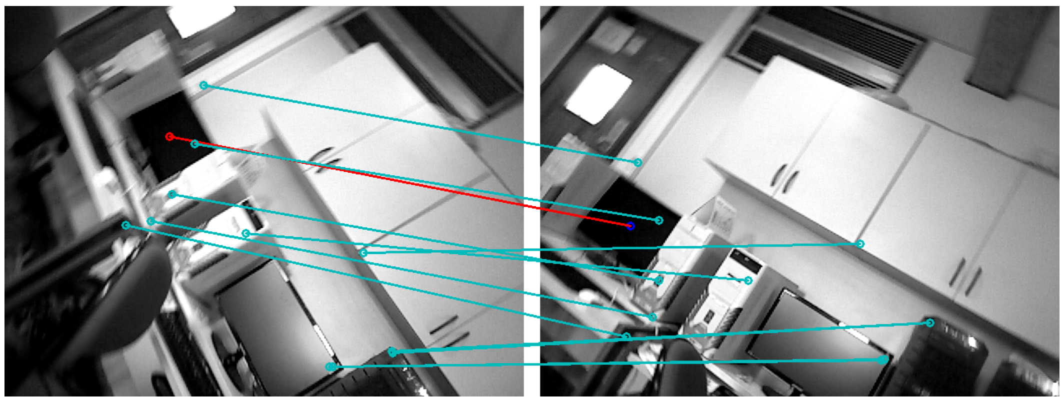


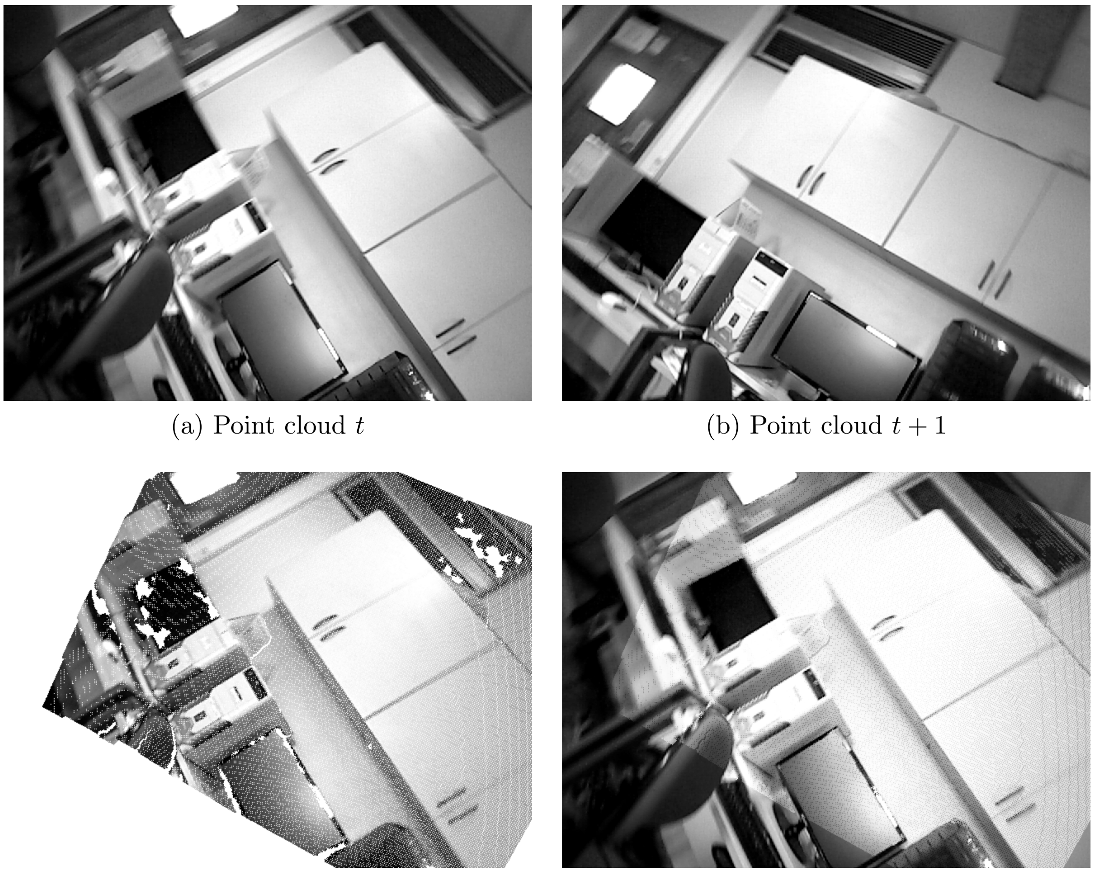
Figure 3.6: Keypoint correspondences found by our keypoint matching approach. The red line connects support keypoints, while the blue lines connects the remaining keypoints. Although there were hundreds of matched keypoints on this pair, we selected a small set of pairs for the sake of clarity. The figures depict a region captured during our experiments.

quickly corrected by a geometry-based registration. However, such approaches have high computational requirements and may be compromised by the lack of geometric features from some environments. On the other hand, the acquired frames often contain a substantial amount of color information, which can be densely used to further improve the alignment, which will help to correct for small displacements on pair-wise registration.

For this alignment stage, we have employed the methodology published by Steinbruecker et al. [2011]. Given two point clouds $\mathbf{C}_a$, $\mathbf{C}_b$ for which each point contains its color estimate, and an initial guess ${}^b_a\tilde{\mathbf{P}}$ for the pose of frame a with respect to frame b, their algorithm searches for a pose ${}^b_a\mathbf{P}$ that leads to the smallest difference between the image from $\mathbf{C}_b$ and the image obtained from applying the perspective projection to the points ${}^b_a\mathbf{P}\mathbf{C}_a$. In other words, at each iteration, it compares the projections from both point clouds as if they were obtained from the sensor at the same spot. If the transformation ${}^b_a\mathbf{P}$ is close to the actual displacement between these clouds, the intersecting pixels of the compared images will be more similar to each other, as it can be seen on Figure 3.7.

Other pairwise alignment methodologies, such as Henry et al. [2010], commonly use the ICP algorithm during their fine alignment stage. We have chosen the photo consistent approach due to its high computational performance, even though it uses all available points, and the fact that it uses intensity as an evaluation metric.

In the pipeline of our methodology, this alignment step starts after the keypoint-based coarse registration, and receives a pose guess from it. It outputs a refined pose



(a) Point cloud t

(b) Point cloud $t+1$

(c) Point cloud $t+1$ after alignment

(d) Aligned point cloud $t+1$ superposed with point cloud t

Figure 3.7: Given a coarse estimate for the pose of frame a w.r.t frame b, ${}^{k_{t-1}}_{k_t}\mathbf{P}'$, the photo consistency step finds a ${}^{k_{t-1}}_{k_t}\mathbf{P}$ that minimizes the difference between figs. (a) and (c).

matrix, ${}^{k_{t-1}}_{k_t}\mathbf{P}'$, which is combined into the global sensor pose ${}^{k_1}_{k_t}\mathbf{P}$ by the matrix product

$$ {}^{k_1}_{k_t}\mathbf{P} = {}^{k_1}_{k_{t-1}}\mathbf{P}\, {}^{k_{t-1}}_{k_t}\mathbf{P}'. $$

This combined sensor pose is the input to the back-end, which will then refine it to a globally consistent pose.

3.2.4 Sampling Key frames and Graph Optimization

The major purpose of our back-end is to select a subset of the captured point clouds, which we refer to as key frames, and to build a globally consistent representation of the environment with them. This is necessary because pairwise registration of point clouds tend to accumulate errors that become very noticeable as the mapped region increases.

There are several entities that can lead to accumulated errors between frames, for instance:

- The sensor discretization of the world makes it impossible for any alignment to be perfect;
- Depth and image warping due to lens distortion, leading to curvy representations of planes;
- Blurring both on depth and color images introduces geometric distortions that do not correspond to the real environment.

Consequently, the back-end must be capable of detecting overlapping regions between key frames, especially if they have a large temporal displacement. Such intersections happen when the sensor returns to a previously mapped region, which may be very difficult to detect, since uncertainty due to accumulated error can make the search space too large when matching temporally farther key frames.

Another reason to use the proposed back end is to prevent normal frames (that is, point clouds that are not distinct enough to be considered key frames) from diverging indiscriminately. This is achieved by aligning them to the closest key frame available. As it will be discussed in more details later on, this realignment operation is typically very fast, since drifting between key frames typically occurs at small scales. Also, the detection of the closest key frame doesn't require the world representation to be globally consistent, because our key frame detection policy ensures that a key frame t is always a good candidate for matching subsequent point clouds that are not considered key frames.

3.2.4.1 Key frame Detection

The criteria for determining whether a point cloud can be considered a key frame or not is of crucial importance to the registration system, as it is related to the quality of the alignments as well as the overall algorithm run time. A large amount of key frames may induce higher run times by cluttering the graph to be optimized, as well as increase the accumulated error after mapping the whole environment. On the other hand, a small number of key frames implies in a smaller intersection region between them (assuming they all cover the same volume), which makes alignment a more difficult task, increasing the chances for registration divergence.

With that in mind, our key frame detection policy takes into account the area of the intersecting region between the depth images from the candidate point cloud and all other key frames. This allows us to address the divergence issue, since a good registration depends on the size of the overlapping region, but also gives control over the amount of key frames detected, which can be reduced by decreasing the intersection threshold.

To determine whether or not a given point cloud $\mathbf{C}_t$ is a key frame, let $\tilde{\mathbf{C}}_\mathbf{k}$ be the set of all point clouds categorized as key frames. At the beginning of the alignment procedure, this set will be initialized with the first point cloud available – i.e. the first point cloud acquired by the depth sensor is automatically categorized as a key frame. If $\tilde{\mathbf{C}}_k$ is not empty, we perform a comparison between $\mathbf{C}_t$ and each $\mathbf{C}_l \in \hat{\mathbf{C}}_\mathbf{k}$. This is done by transforming all the points from $\mathbf{C}_t$ to the reference frame of $\mathbf{C}_l$, and projecting these points to the projection plane that contains all points from $\mathbf{C}_l$ (by multiplying them by the projection matrix of the depth sensor, and normalizing their resulting homogeneous coordinates). Since the depth sensor might have moved after $\mathbf{C}_l$ was captured, some of the points from $\mathbf{C}_t$ will be likely to be projected outside of the projection plane of point cloud $\mathbf{C}_l$. If the number of points that fall inside this projection plane is below a threshold, we define $\mathbf{C}_t$ as a key frame. This procedure is detailed in Algorithm 3.

3.2.4.2 Alignment to Closest Key Frame

During the registration process, it is desirable that regular point clouds be consistent with their closest key frames. This reduces the drift effect, restricting the error accumulation to the process of incorporating new key frames – an error that will be further reduced by graph optimization, should a loop be closed. Since our goal is to perform this alignment quickly as well as to obtain a high quality registration, we solve it with a specialized version of the feature matcher previously described.

The alignment error between a regular point cloud and the most recently detected key frame is typically small enough that their estimated poses can be used as an initial guess for the alignment between them. This alignment is computed by a slightly modified version of the keypoint matcher algorithm previously described. Also, since we expect to have small displacement errors, it is possible to prune the search space for feature correspondences, thus speeding up the feature matcher algorithm. Hence, given a pose estimate ${}^{k_i}_{k_t}\mathbf{P}'$ of the t-th point cloud with respect to the key frame (which was captured at instant i), the radius search of the keypoint matcher receives all support keypoints ${}^{k_i}\tilde{\mathbf{k}}_l$ transformed by ${}^{k_i}_{k_t}\tilde{\mathbf{P}}'^{-1}$ which can be quickly and precisely determined by

Algorithm 3 Key frame Detection

```
Require:
  1: A candidate point cloud $\mathbf{C}_t$ and its pose ${}^{k_1}_{k_t}\mathbf{P}$.
  2: A set of key frames $\tilde{\mathbf{C}}_{\mathbf{k}}$, and their corresponding poses $\tilde{\mathbf{P}}_{\mathbf{k}}$.
  3: A camera projection matrix $\mathbf{K}$.
Ensure:
  1: A boolean variable indicating if $\mathbf{C}_t$ is a key frame; in which case, $\tilde{\mathbf{C}}_{\mathbf{k}}$ is updated to
     contain it as well.

 1: isKeyFrame ← true
 2: for all $\mathbf{C}_l \in \tilde{\mathbf{C}}_{\mathbf{k}}$ do
 3:     intersection ← $\varnothing$
 4:
 5:     for all $\mathbf{p}_l \in \mathbf{C}_l$ do
 6:         $\mathbf{p}'_l \leftarrow \mathbf{K}\, {}^{k_1}_{k_t}\mathbf{P}^{-1}\, {}^{k_1}_{k_l}\mathbf{P}\, \mathbf{p}_l$
 7:         $\mathbf{p}''_l \leftarrow$ homogeneousNormalize($\mathbf{p}'_l$)
 8:         if $(0,\ 0) \le \mathbf{p}''_l \le$ (FRAME_WIDTH, FRAME_HEIGHT) then
 9:             intersection ← intersection $\cup\ \mathbf{p}''_l$
10:         end if
11:     end for
12:     if size(intersection) $\ge$ INTERSECTION_SIZE_THRESHOLD then
13:         isKeyFrame ← false
14:         break for
15:     end if
16: end for
17:
18: if isKeyFrame then
19:     $\tilde{\mathbf{C}}_{\mathbf{k}} \leftarrow \tilde{\mathbf{C}}_{\mathbf{k}} \cup \mathbf{C}_t$
20: end if
21: return (isKeyFrame, $\tilde{\mathbf{C}}_{\mathbf{k}}$)
```

$$
{}^{k_i}_{k_t}\tilde{\mathbf{P}}'^{-1} = \begin{pmatrix} {}^{k_i}_{k_t}\mathbf{R}^T & -{}^{k_i}_{k_t}\mathbf{R}^T\mathbf{t} \\ \mathbf{0}_{1\times 3} & 1 \end{pmatrix},
$$

where ${}^{k_i}_{k_t}\mathbf{R}$ and $\mathbf{t}$ are, respectively, the rotation matrix and translation vector of $\hat{\mathbf{P}}^k_t$.

The pose ${}^{k_i}_{k_t}\tilde{\mathbf{P}}'$ can be computed by using the global pose of point cloud t, ${}^{k_1}_{k_t}\mathbf{P}$ and the global pose of the key frame i, ${}^{k_1}_{k_i}\mathbf{P}$, both calculated as the combined pairwise transformations between all frames that preceded them:

$$
{}^{k_1}_{k_t}\mathbf{P} = \prod_{j=2}^{t} {}^{k_{j-1}}_{k_j}\mathbf{P},
$$

$$
{}^{k_1}_{k_i}\mathbf{P} = \prod_{j=2}^{i} {}^{k_{j-1}}_{k_j}\mathbf{P}.
$$

In this sense, ${}^{k_i}_{k_t}\mathbf{P}' = {}^{k_1}_{k_i}\mathbf{P}^{-1}\,{}^{k_1}_{k_t}\mathbf{P}$.

Given the transformation ${}^{k_i}_{k_t}\mathbf{P}$ resulting from the alignment between current point cloud and the last detected key frame, the current stage outputs ${}^{k_1}_{k_i}\mathbf{P}\,{}^{k_i}_{k_t}\mathbf{P}$, which stands for the corrected, locally consistent, global pose of the t-th point cloud.

3.2.4.3 Optimizing Environment Graph and Closing Loops

In order to estimate a reliable representation of the environment, mapping trajectories that return to a previously visited region must be subject to loop closure techniques, which reduce significantly the accumulated error, an inevitable outcome of pairwise frame alignment.

This is done by associating a hypergraph to the key frame structure, in which each key frame is represented by a vertex, and each keypoint correspondence between them becomes an edge, as shown in Figure 3.8. Since our goal is to perform environment mapping with 6 degrees of freedom in terms of sensor motion, this framework can be processed by the methodology proposed by Borrmann et al. [2008], which was an extension of a graph optimization framework for 2D scans with 3 degrees of freedom [Lu and Milios, 1997]. The latter takes as input a graph in which each vertex represents the sensor's pose $\mathbf{X}_i = \begin{pmatrix} x & y & \theta \end{pmatrix}$ (which may be estimated by pairwise registration of all previous point clouds), while edges between them contains the measured displacement between these poses $\mathbf{D}_{i,j} = \begin{pmatrix} \Delta x & \Delta y & \Delta\theta \end{pmatrix}$ (typically obtained from the alignment between those vertices). If all scans were perfectly matched and there were no cumulative error, there should be no difference between $\mathbf{X}_i - \mathbf{X}_j$ and $\mathbf{D}_{i,j}$ for any two connected nodes i and j. When accumulated errors are present, however, the graph optimizer must seek to minimize this difference. Therefore, its fitness function is given by the following Mahalanobis distance:

$$
\sum (\mathbf{X}_i - \mathbf{X}_j - \mathbf{D}_{i,j})^T \mathbf{C}_{i,j}^{-1} (\mathbf{X}_i - \mathbf{X}_j - \mathbf{D}_{i,j}),
$$

where $\mathbf{C}_{i,j}$ is the measurement covariance matrix.

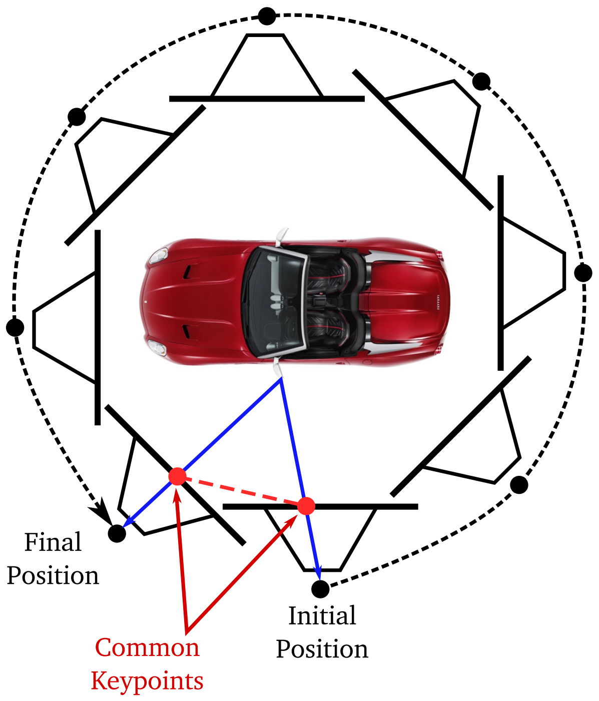


Figure 3.8: Illustration of a key frame hypergraph to be processed by the SLAM step. Each key frame is represented by a vertex, while the keypoint connections are expressed by edges connecting their corresponding vertices. In this illustration, the keypoint correspondence between initial and final positions are found by the loop closure detector.

An important aspect of a global alignment procedure is how loop closures are detected between two candidate key frames. Our approach makes use of the attitude reported by the MARG sensor in this process, since it does not suffer from significant drift under the assumed conditions. Therefore, we reduce the loop detection problem to finding a translation that connects two candidate key frames, while being able to reject candidates with a large angular displacement. This is done with an algorithm that follows the underlying idea behind Algorithm 2. That is, the transform between key frames is estimated from a set of corresponding keypoints. Since the magnitude of the accumulated error cannot be generically estimated with a reasonable precision, the support pairing from which the translation between the key frames is estimated is obtained from testing all possible combinations between the subset of keypoints $\hat{\mathbf{k}}^i$ and a subset $\hat{\mathbf{k}}^j$ of $\mathbf{k}^j$. Although this increases the computational complexity of the matching procedure, it also increases the odds of correctly matching two key frames captured at time instants too far apart from each other, regardless of the accumulated error between them. The complete procedure is summarized in Algorithm 5.

The keypoint matcher used in pairwise alignment (described in Algorithm 2) has a search space delimited by an uncertainty region proportional to the maximum dis-

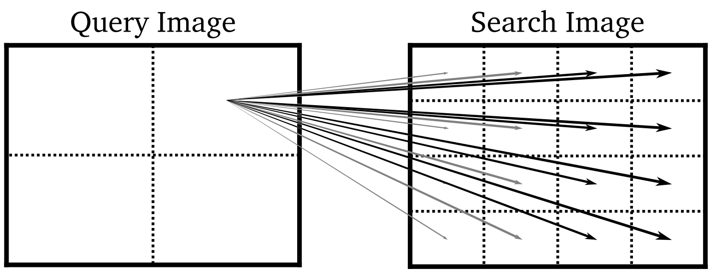


Figure 3.9: In order to perform a thorough search for correspondents of each subsampled keypoint from the query image, we extract the best keypoints from each bin of the search image.

placement the sensor could have had in the elapsed time between the acquisition of two point clouds. When dealing with any two arbitrary point clouds from our set, however, we acknowledge that this uncertainty region could be as large as to occupy the whole volume delimited by these point clouds. This means that the previously described keypoint matcher would essentially test all combinations of keypoints. Facing this issue, and concerned with the computational complexity of the keypoint matching procedure among arbitrary key frames (which is the essence of our loop closure detection), we decided to adopt a more sophisticated keypoint subsampling at the loop closure detection stage. Essentially, it divides both query and search images into sub regions and samples the most prominent keypoints (according to their response to the SURF detector) from both, up to a maximum threshold. As Figure 3.9 illustrates, each keypoint subsampled from the query image is later matched with all keypoints subsampled from the search image. The keypoint subsample procedure is described by Algorithm 4.

Considering the time complexity of the global optimization procedure, it is advisable to refrain from performing comparisons between all pairs of key frames. Since the graph is always locally aligned, edges between neighboring vertices can be detected by Algorithm 2 by making use of their pose estimates. Furthermore, it is possible to use the orientation data from the MARG sensor to discard far off candidates based on a threshold regarding the angle between their view direction vector. For instance, a loop cannot be closed between two key frames if the first was captured with the sensor pointing towards the floor and, the second, pointed to the ceiling. Given two rotation matrices representing the depth sensor attitudes at distinct instants, their angular distance is calculated as the angle α between their direction vectors, $\hat{\mathbf{Z}}_i$ and $\hat{\mathbf{Z}}_j$, which are given by the third column of their corresponding rotation matrices. That is, the angle α can be calculated as $\alpha = \arccos(\mathbf{Z}_i \cdot \mathbf{Z}_j)$.

Algorithm 4 Relaxed Keypoint Subsampling.

```
Require:
 1: Set of keypoints $\mathbf{k}$.
 2: Dimension $d$ of sampling bins.
 3: Maximum amount of points per bin $n$.
Ensure:
 1: Subsample $\tilde{\mathbf{k}}$ of keypoints from $\mathbf{k}$.

 1: $\tilde{\mathbf{k}} \leftarrow \varnothing$
 2: binCount $\leftarrow$ ZEROS(FRAME_HEIGHT/$d$, FRAME_WIDTH/$d$)
 3: for all $\mathbf{k}_m \in \mathbf{k}$ do
 4:     binIndex $\leftarrow$ FLOOR($\mathbf{k}_m/d$)
 5:     if binCount[binIndex.y, binIndex.x] $< n$ then
 6:         $\tilde{\mathbf{k}} \leftarrow \tilde{\mathbf{k}} \cup \mathbf{k}_m$
 7:         binCount[binIndex.y, binIndex.x] $\leftarrow$ binCount[binIndex.y, binIndex.x] $+ 1$
 8:     end if
 9: end for
10: return $\tilde{\mathbf{k}}$
```

Algorithm 5 Relaxed Keypoint Matching

```
Require:
 1: Sets of keypoints ${}^{k_i}\mathbf{k}$ and ${}^{k_j}\mathbf{k}$ from the $t_i$-th and $t_j$-th key frames, respectively, and
    their corresponding descriptors.
 2: The depth sensor's attitude at the instant $t_j$ with respect to its orientation at
    instant $t_i$, as estimated by the MARG sensor, ${}^{k_i}_{k_j}\mathbf{K}$.
Ensure:
 1: The depth sensor's pose at instant $t_j$ with respect to its local frame at instant $t_i$,
    ${}^{k_i}_{k_j}\mathbf{P}$.
 2: The best match score.
 3: The number of matched keypoints $n$.

 1: ${}^{k_i}\tilde{\mathbf{k}} \leftarrow$ RELAXEDSUBSAMPLE(${}^{k_i}\mathbf{k}$, QUERY_DIM, QUERY_NUM_PTS)
 2: ${}^{k_j}\tilde{\mathbf{k}} \leftarrow$ SUBSAMPLE(${}^{k_j}\mathbf{k}$, MATCH_DIM, MATCH_NUM_PTS)
 3: bestScore $\leftarrow \infty$
 4: bestCandidates $\leftarrow \varnothing$
 5: for all ${}^{k_i}\tilde{\mathbf{k}}_l \in {}^{k_i}\tilde{\mathbf{k}}$ do
 6:     for all ${}^{k_j}\tilde{\mathbf{k}}_l \in {}^{k_j}\tilde{\mathbf{k}}$ do
 7:         score ← HAMMINGDISTANCE(DESCRIPTOR(${}^{k_i}\tilde{\mathbf{k}}_l$), DESCRIPTOR(${}^{k_j}\tilde{\mathbf{k}}_l$))
 8:         matches ← $\varnothing$
 9:         $\mathbf{t} \leftarrow {}^{k_j}\tilde{\mathbf{k}}_l - ({}^{k_i}_{k_j}\mathbf{K})^{-1}\ {}^{k_i}\tilde{\mathbf{k}}_l$
10:         for all ${}^{k_i}\tilde{\mathbf{k}}_m \in$ SUBSET(${}^{k_i}\tilde{\mathbf{k}}$) | ${}^{k_i}\tilde{\mathbf{k}}_m \neq {}^{k_i}\tilde{\mathbf{k}}_l$ do
11:             $\mathbf{c} \leftarrow ({}^{k_i}_{k_j}\mathbf{K})^{-1}\ {}^{k_i}\tilde{\mathbf{k}}_m + \mathbf{t}$
12:             neighbors←KNNSEARCH(${}^{k_j}\mathbf{k}$, $\mathbf{c}$, AMOUNT_NEIGHBORS)
13:             ${}^{k_j}\tilde{\mathbf{k}}_m$ ← BESTNEIGHBOR(neighbors)
14:             hamming ← HAMMINGDISTANCE(DESCRIPTOR(${}^{k_i}\tilde{\mathbf{k}}_m$),
                        DESCRIPTOR(${}^{k_j}\tilde{\mathbf{k}}_m$))
15:             if $\|\mathbf{c} - {}^{k_j}\tilde{\mathbf{k}}_m\| \leq$ DISTANCE_THRESHOLD then
16:                 score ← score + hamming
17:                 matches ← matches $\cup \{({}^{k_i}\tilde{\mathbf{k}}_m,\ {}^{k_j}\tilde{\mathbf{k}}_m)\}$
18:             end if
19:         end for
20:        INSERTCANDIDATE(bestCandidates, score/SIZE(matches)$^2$,
                              $\{{}^{\mathbf{k_i}}\tilde{\mathbf{k}}_l, \tilde{\mathbf{k}}_l^{t_j}\}$, BEST_CANDIDATES_LIMIT)
21:     end for
22: end for
23: for all $\{{}^{k_i}\tilde{\mathbf{k}}_l,\ {}^{k_j}\tilde{\mathbf{k}}_l\} \in$ bestCandidates do
24:     score ← HAMMINGDISTANCE(DESCRIPTOR(${}^{k_i}\tilde{\mathbf{k}}_l$), DESCRIPTOR(${}^{k_j}\tilde{\mathbf{k}}_l$))
25:     matches ← $\varnothing$
26:     $\mathbf{t} \leftarrow {}^{k_j}\tilde{\mathbf{k}}_l - ({}^{k_i}_{k_j}\mathbf{K})^{-1}\ {}^{k_i}\tilde{\mathbf{k}}_l$
27:     for all ${}^{k_i}\tilde{\mathbf{k}}_m \in {}^{k_i}\tilde{\mathbf{k}} \lfloor {}^{k_i}\tilde{\mathbf{k}}_m \neq {}^{k_i}\tilde{\mathbf{k}}_l$ do
28:         $\mathbf{c} \leftarrow ({}^{k_i}_{k_j}\mathbf{K})^{-1}\ {}^{k_i}\tilde{\mathbf{k}}_m + \mathbf{t}$
29:         neighbors←KNNSEARCH(${}^{k_j}\mathbf{k}$, $\mathbf{c}$)
30:         ${}^{\mathbf{k_j}}\tilde{\mathbf{k}}_m$ ← BESTNEIGHBOR(neighbors)
31:         hamming ← HAMMINGDISTANCE(DESCRIPTOR(${}^{k_i}\tilde{\mathbf{k}}_m$), DESCRIPTOR(${}^{k_j}\tilde{\mathbf{k}}_m$))
32:         if $\|\mathbf{c} - {}^{k_j}\tilde{\mathbf{k}}_m\| \leq$ DISTANCE_THRESHOLD then
33:             score ← score + hamming
34:             matches ← matches $\cup \{({}^{k_i}\tilde{\mathbf{k}}_m,\ {}^{k_j}\tilde{\mathbf{k}}_m)\}$
35:         end if
36:     end for
37:     score ← score/SIZE(matches)$^2$
38:     if bestScore > score then
39:         ${}^{k_i}_{k_j}\mathbf{P}$ ← POSEFROMPCA(matches)
40:         bestScore ← score
41:         $n$ ← SIZE(matches)
42:     end if
43: end for
44: return $({}^{k_i}_{k_j}\mathbf{P}, \mathrm{score}, \mathrm{n})$
```

3.3 Parameters

For the sake of practicality, we compile in Table 3.1 a list of all constant parameters of our methodology.

`DISTANCE_THRESHOLD`	When matching keypoints, it's the maximum acceptable distance between a query point and its candidate match neighbors. Increasing this parameter can help to mitigate the effect of noise in the MARG sensor estimate, but increases the computational cost of the matcher.
`AMOUNT_NEIGHBORS`	Similar to `DISTANCE_THRESHOLD`, with the difference it specifies the number of neighbors to be considered in a KNN key point search. Increasing this parameter helps mitigate the noise from the MARG sensor, but also decreases computational performance.
`MAX_SENSOR_SPEED` ($v_{\max}$)	We assume the sensor will not move faster than this speed. This parameter is to be set in meters per second. Increasing this parameter is the equivalent of increasing uncertainty in translation between two pairwise key frames, and will translate to higher computational requirements as well.
`FRAME_WIDTH`	Width of the frames acquired by the depth sensor, in pixels.
`FRAME_HEIGHT`	Height of the frames acquired by the depth sensor, in pixels.
`INTERSECTION_SIZE_THRESHOLD`	After aligning two candidate point clouds in the loop closure detection step, we project the aligned point clouds in a 2D frame and discard the loop closure if the pixel intersection in this alignment is smaller than this parameter. Increasing it reduces the probability of matching loop closures, as decreasing it makes it more likely to find false positives. This value must lie within the range $[0, \texttt{FRAME\_WIDTH} \cdot \texttt{FRAME\_HEIGHT}]$.

`QUERY_DIM`	In the keypoint subsampling context, specifies the dimension of the square patches from where keypoints will be selected on the query image. Decreasing it makes the subsampled points to be well distributed across the whole frame, but increases computational requirements.
`QUERY_NUM_PTS`	In the keypoint subsampling context, specifies the maximum number of keypoints that will be selected from each image patch on the query image. Increasing it makes the matcher more robust to outliers in each set, although it leads to a higher computational cost.
`MATCH_DIM`	Analogous to `QUERY_DIM`, but applies to the match set.
`MATCH_NUM_PTS`	Analogous to `QUERY_NUM_PTS`, but applies to the match set.
`BEST_CANDIDATES_LIMIT`	In the relaxed keypoint matching algorithm, defines the maximum number of support pairs that will be submitted to a thorough score evaluation. Greater limits diminishes the likelihood of finding a local minimum, at the same time it increases computational cost.

Table 3.1: List of constant parameters used by the proposed methodology.

Chapter 4

Experiments

Our experimental analysis sought to compute the fidelity of a map generated by the proposed methodology to a ground truth, which is generated from a Zebedee device [Bosse et al., 2012]. We were also concerned with how each block of our pipeline contributed to the final reconstruction.

During our experiments, we captured the walls of a cluttered rectangular room with dimensions 9.84m x 7.13m with an XTion PRO LIVE sensor attached to a 3DM-GX1 MARG sensor, as shown by figure 4.1. The mapping process took about 2 minutes to complete, giving an average speed of roughly 0.3m/s. During this interval, we captured 3404 point clouds, which consumed a total of 5GiB in disk space. We ignored the first 150 point clouds (the equivalent of 5 seconds) because they had color artifacts, and we wanted to wait for the MARG filter to converge to its initial attitude.

The computer used to capture the point clouds and the MARG data was a MSI GX660r-us notebook with a 1.73GHz Core i7 CPU running the operating system Ubuntu 12.04. The registration algorithms were executed in a desktop equipped with a 3.5 GHz Core i7 CPU with 32GiB of memory running the same operating system.

This room has been chosen due to the fact that it contained regions with rich geometric and texture features as well specific regions that lack both, as illustrated by Figure 4.2. This has enabled us to test the robustness of the proposed approach on such different conditions.

Table 4.1 lists the values associated with our methodology's parameters used in our experiments.

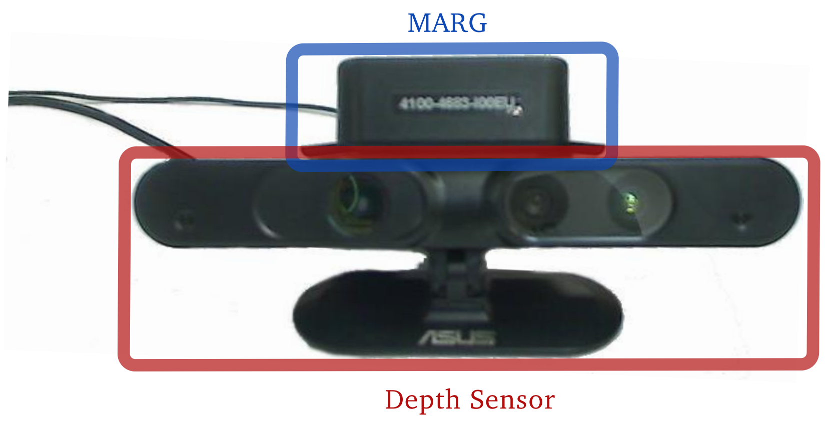


Figure 4.1: Devices used during our experiments.

`DISTANCE_THRESHOLD`	0.02m
`AMOUNT_NEIGHBORS`	6
`MAX_SENSOR_SPEED`	0.6m/s
`FRAME_WIDTH`	640px
`FRAME_HEIGHT`	480px
`INTERSECTION_SIZE_THRESHOLD`	150,000
`QUERY_DIM`	10
`QUERY_NUM_PTS`	1
`MATCH_DIM`	10
`MATCH_NUM_PTS`	5
`BEST_CANDIDATES_LIMIT`	100

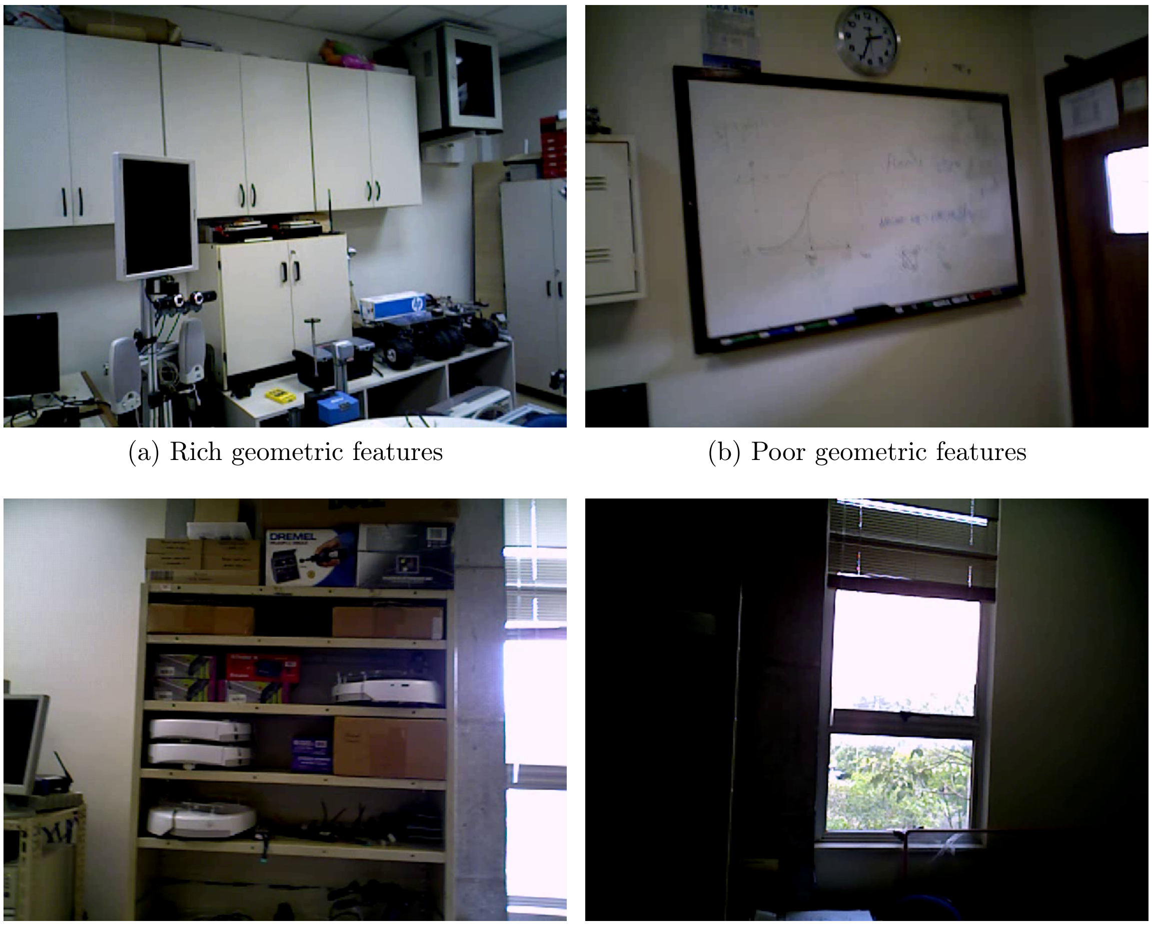
Table 4.1: List of constant parameters used during our experiments.

4.1 Synchronization Issues

Since we were dealing with multiple asynchronous sensors, we had to create a mechanism to match data from the MARG sensor to frames coming from the depth sensor. Due to the nature of our sensors, there were several factors that made this task difficult: The delay between data acquisition and depth availability, regarding the depth sensor, the different operation frequencies of both sensors, possible slowdowns during depth data transferring and the lack of a global reference clock, to name a few. In order to solve this problem, we accounted for the following information:

- **MARG time stamp.** The time instant at which the inertial data was available, provided by a clock within the MARG device.

- **Depth Sensor time stamp.** The time instant at which the depth data was acquired, computed by a clock within the depth sensor.


(a) Rich geometric features (b) Poor geometric features

(c) Rich texture features (d) Poor texture features

Figure 4.2: Different texturing and geometric conditions of the experimental setup.

- **Host USB time stamp.** The time instant at which data was received at the USB port, according to the computer's clock.

Despite the number of available clocks, there were no synchronization mechanisms among them. Not only there were initial offsets between these clocks, but these offsets were not static and there was nothing to prevent them from drifting over time. Therefore, the most suitable decision was to use only one of these clocks to estimate the data acquisition instants for these sensors.

Since we configured the MARG sensor to operate at 80Hz, transferring 144 bits (two bytes per axis, three bytes per sensor type, three sensor types – magnetometer, accelerometer and gyroscope) for each measurement (or 11.25Kibs per second), we assumed the delay between data acquisition and data availability on the host machine was negligible. That means that the data transfer delay could be neglected for the purposes of our experiments. Also, since the MARG device employed by this work was designed for real time applications [MicroStrain, 2006], we assumed its time stamp could be approximated by the host USB time stamp.

Such an approximation, however, was not possible for the depth sensor. This is because it had a considerable delay between the instant at which the cameras captured the world and the instant when the range data was available. According to Shpunt et al. [2008], the depth sensor works by projecting on the scene a pattern whose orientation changes as a function to distance. Besides calculating the orientation of the captured patterns, the sensor also performs a stereo matching refinement of the estimated range. The whole procedure can take about 40ms to finish [Kvalbein, 2012]. Nevertheless, the standard deviation around this value is negligible for our purposes, which means that was possible to assume such a delay between the depth sensor and the host USB time stamp without a major negative impact. Also, by limiting the maximum speed the sensors could move during our experiments, we helped to mitigate undesirable effects from the lack of a synchronization hardware.

4.2 Performance of the globally optimized registration

In order to evaluate the accuracy of our methodology, we handpicked sixteen salient points from both the reconstructed map and the ground truth, and calculated the distances between all combinations of pairs. The points were chosen so as to create a thorough distribution of points in the whole map, allowing us to evaluate the errors for several distinct distances. Figure 4.3 illustrates the distribution of points and the evaluated pairs across the experimental setup.

Of the 3253 point clouds processed by the pairwise alignment stage, 75 were classified as key frames by our backend. As can be seen on Figure 4.4-b, the pairwise alignment stage resulted in significant accumulated pose error after all point clouds were aligned. However, the global optimization stage managed to detect the correct loop closure for this scene, resulting in a visually consistent final map, as shows Figure 4.4-c.

Since we had the distances between these points in both the map generated by the Zebedee sensor and the one provided by our methodology, we computed the difference between equivalent distances on these maps. That is, given a pair of points $\mathbf{p}_a$ and $\mathbf{p}_b$, we first calculated the distances between these points in both maps, ${}^{o}d = {}^{o}\mathbf{p}_a - {}^{o}\mathbf{p}_b$ for the map provided by our methodology and ${}^{z}d = {}^{z}\mathbf{p}_a - {}^{z}\mathbf{p}_b$ for the map provided by the Zebedee, and then calculated the error of our map as the difference between

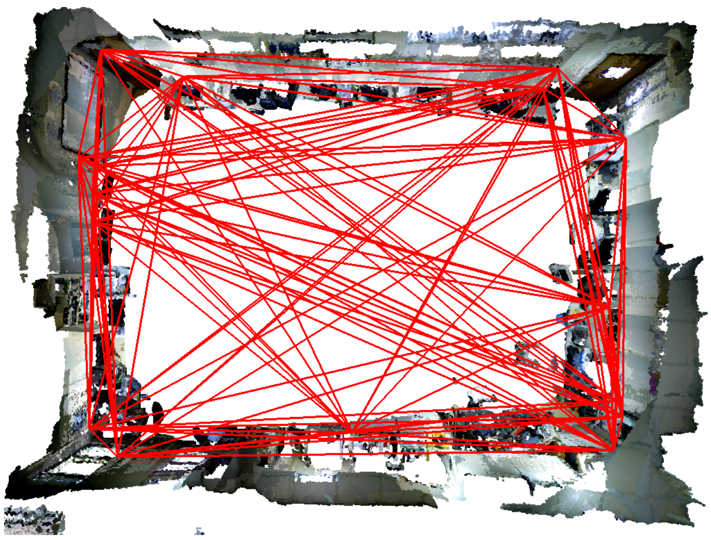


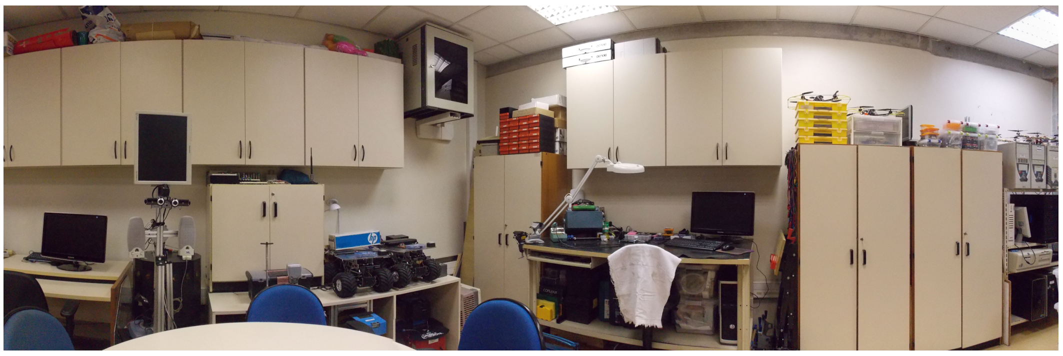
Figure 4.3: Geometric features selected for comparison purposes. We calculate the distances between all pairs of geometric features, and compare the results to the distances estimated from the ground truth reconstruction.

these distances, $e = {}^{o}d - {}^{z}d$. To study the behavior of our error as a function of distance between the points, we generated Figure 4.5-a, in which each point is given by $(x, y) = ({}^{z}d, e)$. Figure 4.5-b was plotted with the errors shown as a percentage of the distance between the points from which they were computed, i.e., $(x, y) = ({}^{z}d, 100\frac{e}{{}^{z}d})$.

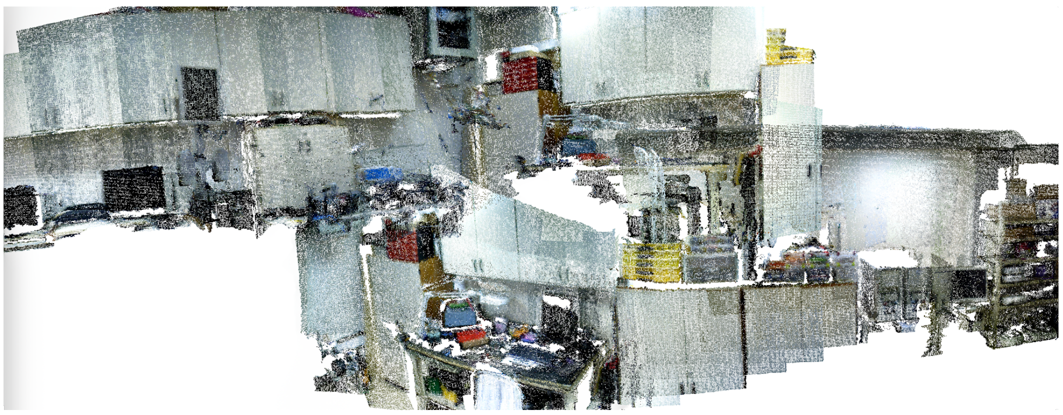
We can see, from Figure 4.5-a, that the error behaves as a random variable whose mean tends to increase for larger distances. This insight can be useful for us to determine the accuracy of our map as a random function of the distance between points. As can be seen on Figure 4.5-b, we have a random variable that seems to be bound by a maximum error throughout all distances. To show that this is is a statistical property of the error of our map, we should be able to fit a unimodal probability distribution function to this data with a quantile-quantile plot.

In a quantile-quantile plot, it is possible to infer the probability distribution of a sample if the plot has a linear behavior [Jain, 1991] – that is, if the plot is well approximated by a linear function, then the sample can be approximated by the known distribution used in the plot. Due to the shape of the histogram given by the normalized errors (depicted by Figure 4.10-a), we decided to perform a quantile-quantile plot with the standard normal distribution followed by a linear regression, which gave us a coefficient of determination of 0.809. This plot is shown by Figure 4.6. Due to the light tails on both of its ends, the normal distribution is likely to be just an approximation for the actual distribution that models this result.

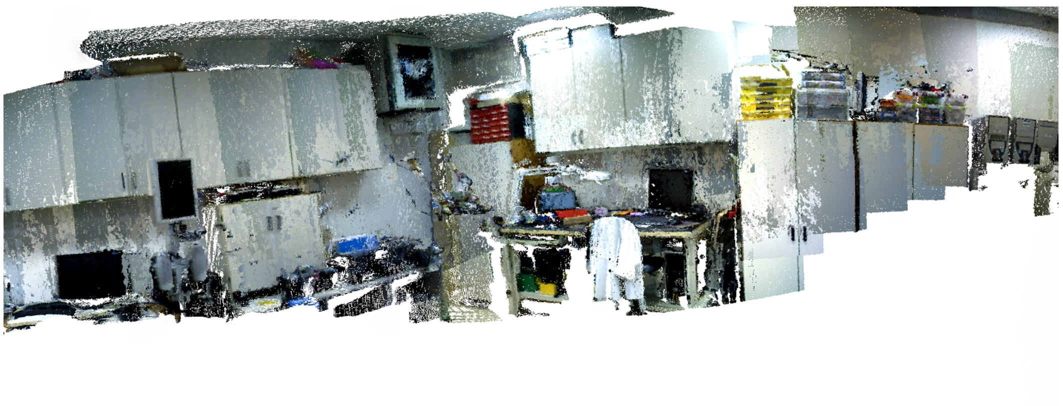
Due to the convenient properties of the normal distribution, it is desirable to



(a)



(b)



(c)

Figure 4.4: (a) Panoramic view of a corner of the experimental setup; (b) reconstruction of this region before global optimization; (c) reconstruction of this region after global optimization. The desk at the center marks the region where reconstruction started, and also corresponds to where the loop should be closed.

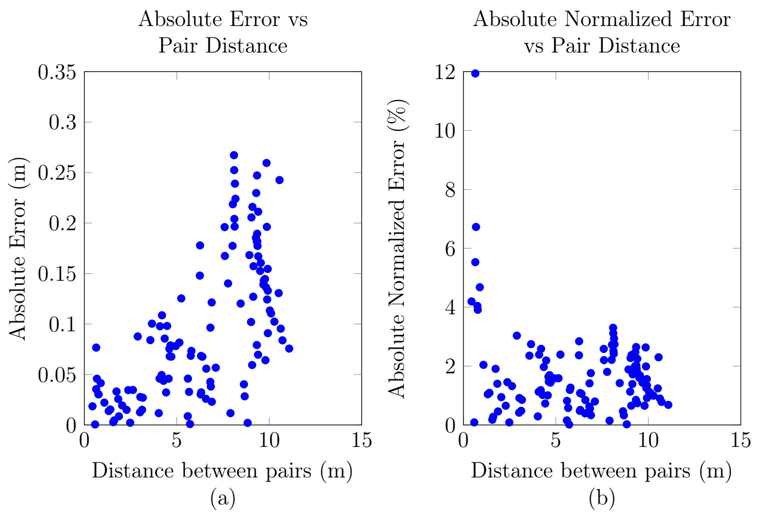


Figure 4.5: Accuracy of the global reconstruction method with respect to the ground truth provided by a Zebedee sensor. The graphic on the right represents the error as a percentage to the distance between the compared points.

model the error of the map generated by our methodology with this distribution, as the coefficient of determination of the quantile-quantile is high enough. In order to validate the normal distribution as an acceptable approximation to such error, we performed a Shapiro-Wilk test with significance level $\alpha = 0.01$. This test didn't discard our approximation of the error as a normally distributed random variable. Therefore, we can say that the normalized error of the reconstruction provided by our methodology, in percents, can be modeled by $\hat{e} = \mathcal{N}(\mu : 1.4310,\ \sigma : 1.1513)$.

4.3 Back-end Robustness

We tested the robustness of our loop closure stage by running it with two distinct pairwise alignment methodologies. We expected that the global optimizer would receive different inputs from these alignments (a requirement that will be discussed posteriorly), but would produce similar results for each one of them. The pairwise alignment methodologies employed in this stage were small modifications of our front-end stage, which were as follows:

(A) **No photo consistent alignment.** The output pose from our coarse alignment was directly forwarded to the back-end block. In practical terms, this means that this strategy didn't perform a dense, fine point cloud registration.

(B) **No MARG-based keypoint matcher.** The coarse alignment was not per-

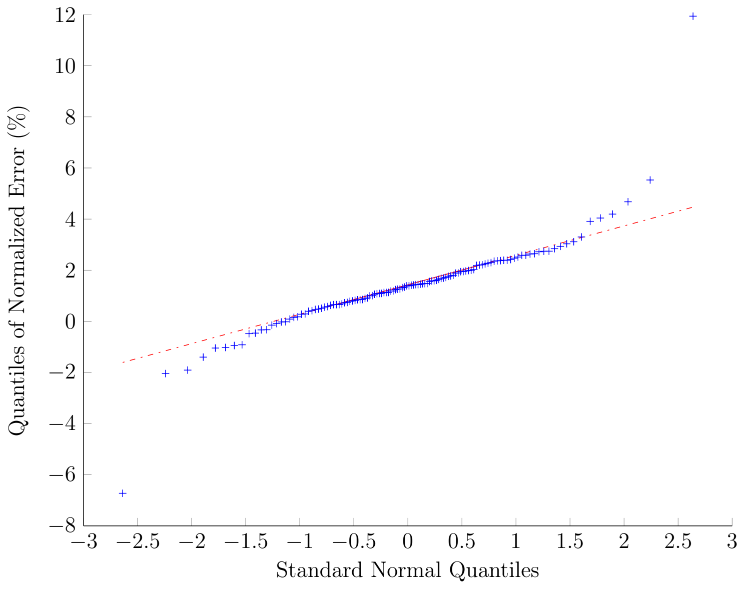


Figure 4.6: Quantile-quantile plot of normalized error versus standard normal distribution.

formed with input from the MARG sensor; instead, keypoints were matched with a regular brute force matcher, and its output were forwarded to the photo consistent stage.

While the backend detected 76 key frames from Strategy (A), Strategy (B) resulted in 73 key frames. Our first intention was to compute the translational error among the key frames generated by our methodology and strategies (A) and (B) before all of them were globally optimized. Since different strategies could classify different point clouds as key frames, we could not directly compare their key frames according to their acquisition time, since a point cloud ${}^{t}\mathbf{C}$ could have been classified as key frame by our methodology but not by strategies (A) and (B). To solve this, we picked the translational component of the poses of all key frames detected by our methodology, and the translational component of the poses of these same point clouds as computed by strategies (A) and (B), regardless of whether or not they were classified as key frames. Finally, we computed the translational errors as the norm of the difference between translational component of equivalent point clouds:

$$\begin{aligned} {}^{o,a}e_k &= \|{}^{o}\mathbf{t}_k - {}^{a}\mathbf{t}_k\|, \\ {}^{o,b}e_k &= \|{}^{o}\mathbf{t}_k - {}^{b}\mathbf{t}_k\|, \\ {}^{a,b}e_k &= \|{}^{a}\mathbf{t}_k - {}^{b}\mathbf{t}_k\|, \end{aligned}$$

where:

- ${}^{o,a}e_k$ is the translational error between our complete approach and Strategy (A) at key frame k,
- ${}^{o,b}e_k$ is the translational error between our complete approach and Strategy (B) at key frame k,
- ${}^{o,b}e_k$ is the translational error between Strategy (A) and Strategy (B) at key frame k,
- ${}^{o}\mathbf{t}_k$ is the translational component of the pose of key frame k as estimated by our complete approach,
- ${}^{a}\mathbf{t}_k$ is the translational component of the pose of key frame k as estimated by Strategy (A),
- ${}^{b}\mathbf{t}_k$ is the translational component of the pose of key frame k as estimated by Strategy (B).

Figures 4.7-a and 4.7-b illustrates the differences between our methodology and strategies (A) and (B); the results from Figure 4.7-c were acquired in a similar fashion, except we used the key frame acquisition times from strategy (A).

As it can be seen in Figure 4.7, the pairwise alignments (A) and (B) yielded different results from those obtained from the proposed methodology. Figures 4.8 and 4.9 illustrate the estimated trajectory from these strategies, in comparison to our registration in the same conditions. An important observation that can be made from these results is that, when comparing the map generated by our methodology and strategies (A) and (B), there may be different alignments between adjacent key frames after global optimization. This may happen because our global optimization stage doesn't seek to refine the alignment between adjacent point clouds; it focuses on detecting poorly closed loops instead. Therefore, the alignment error between adjacent point clouds will remain the same as it was when provided by the pairwise alignment stage (which depends on the adopted strategy, as those graphics suggest).

After performing global optimization, strategies (A) and (B) yielded maps with loop closure errors that could have been readily noticed on a visual inspection, as shown in Figures 4.11. Quantitatively, the normalized error from strategy (A) has a mean 2.62% and standard deviation 1.66 – both values are larger than the corresponding values obtained from our methodology. As for strategy (B), the data dispersion is

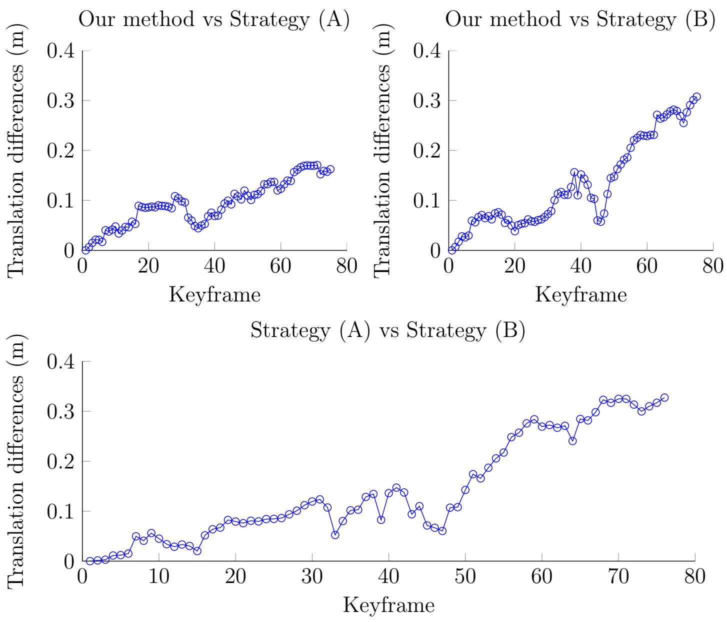


Figure 4.7: Differences between key frame translations estimated by pairwise trajectories.

significantly larger, with a mean of 0.14% and standard deviation of 27.32%. These values can be observed on the histograms of Figure 4.10

Since the match between loop closing frames doesn't depend on the pre- alignment stage, we would expect this region of the global map to be consistent with the result obtained from our complete method presented in the previous section, unless the data provided to the global alignment stage was different for both strategies (A) and (B). There is one way this could have happened: by providing the global alignment stage with a different set of key frames.

As it can be recalled from our methodology, the key frame detection is a byproduct of the pairwise alignment stage. Therefore, different pairwise alignment strategies might yield different sets of key frames. This was the case for both strategies (A) and (B), as can be seen from the instants at which each assessed strategy detected a key frame, depicted by Figure 4.12.

We suspected that the poorly closed loops obtained by Strategies (A) and (B) was a result of a different set of key frames given to the global optimization stage. To verify this hypothesis, we changed the set of point clouds categorized as key frames by those strategies. Instead of using the key frames detected by them, we used the key frames detected by our methodology – but still used their poses as estimated by strategies (A) and (B). Should our original hypothesis be correct, the expected result would be a map whose loop is consistent with the one generated by our methodology,

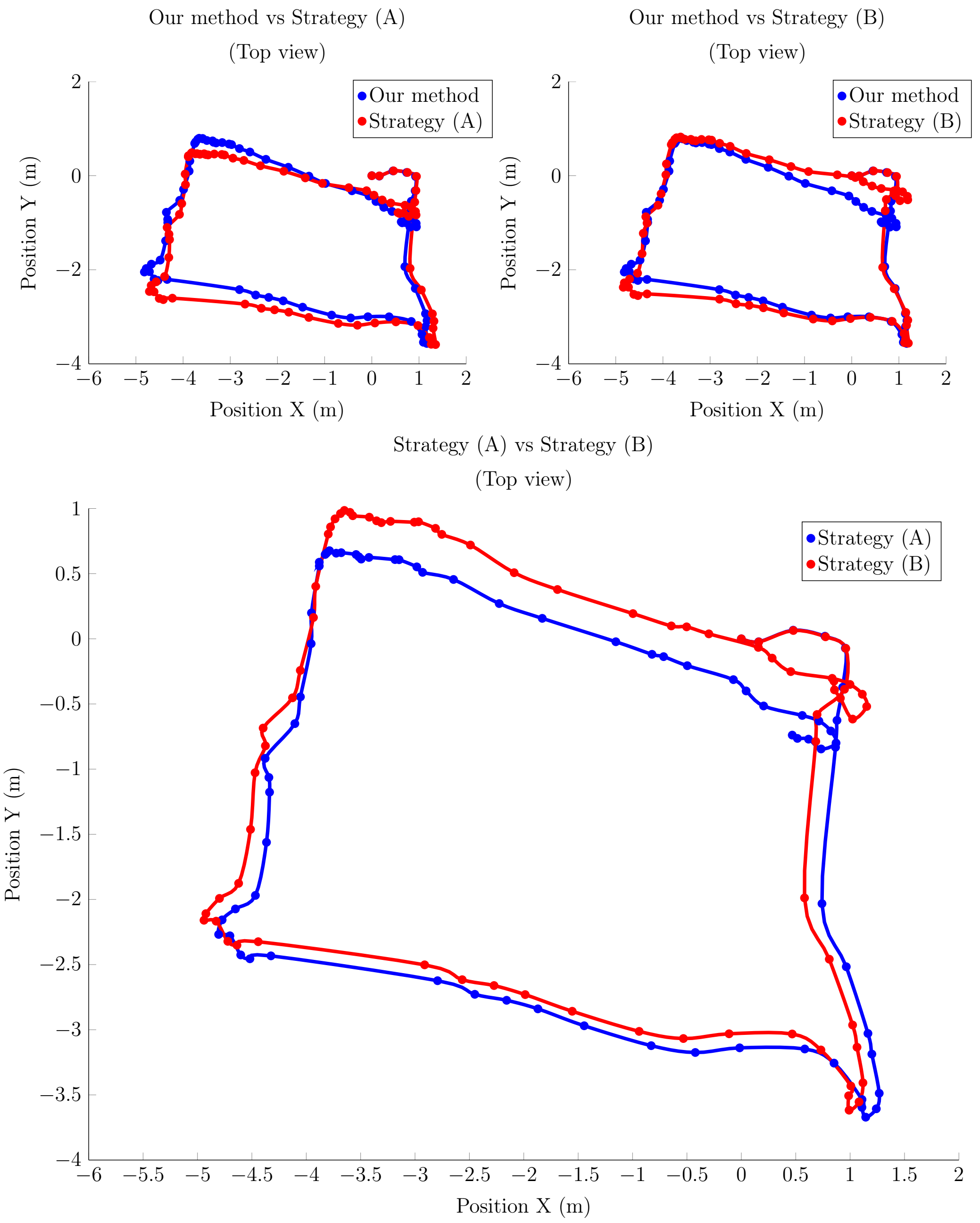


Figure 4.8: Comparison of trajectories as computed by all employed pairwise strategies.

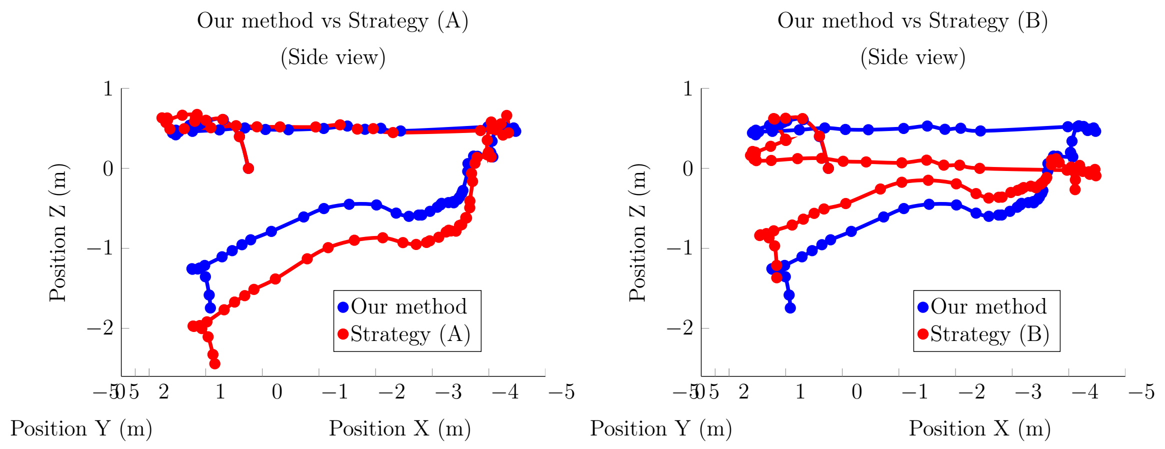


Figure 4.9: Side view of trajectories estimated by our methodology, Strategy (A) and Strategy (B). As can be seen, the trajectory estimated by Strategy (B) has larger displacements in the Z axis.

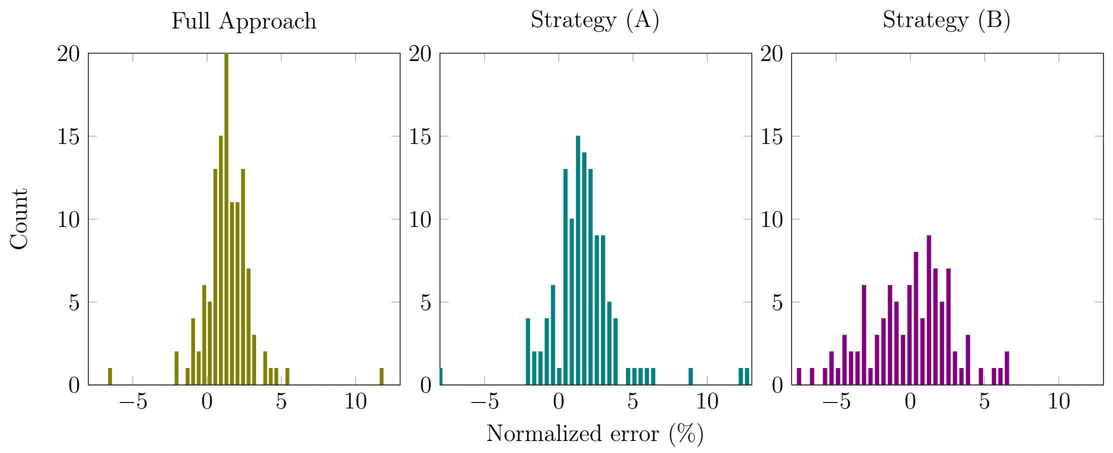


Figure 4.10: Dispersion of normalized error according to the several tested methodologies. Strategy (B) dispersion is zoomed into the same region as the other histograms for comparison purposes.

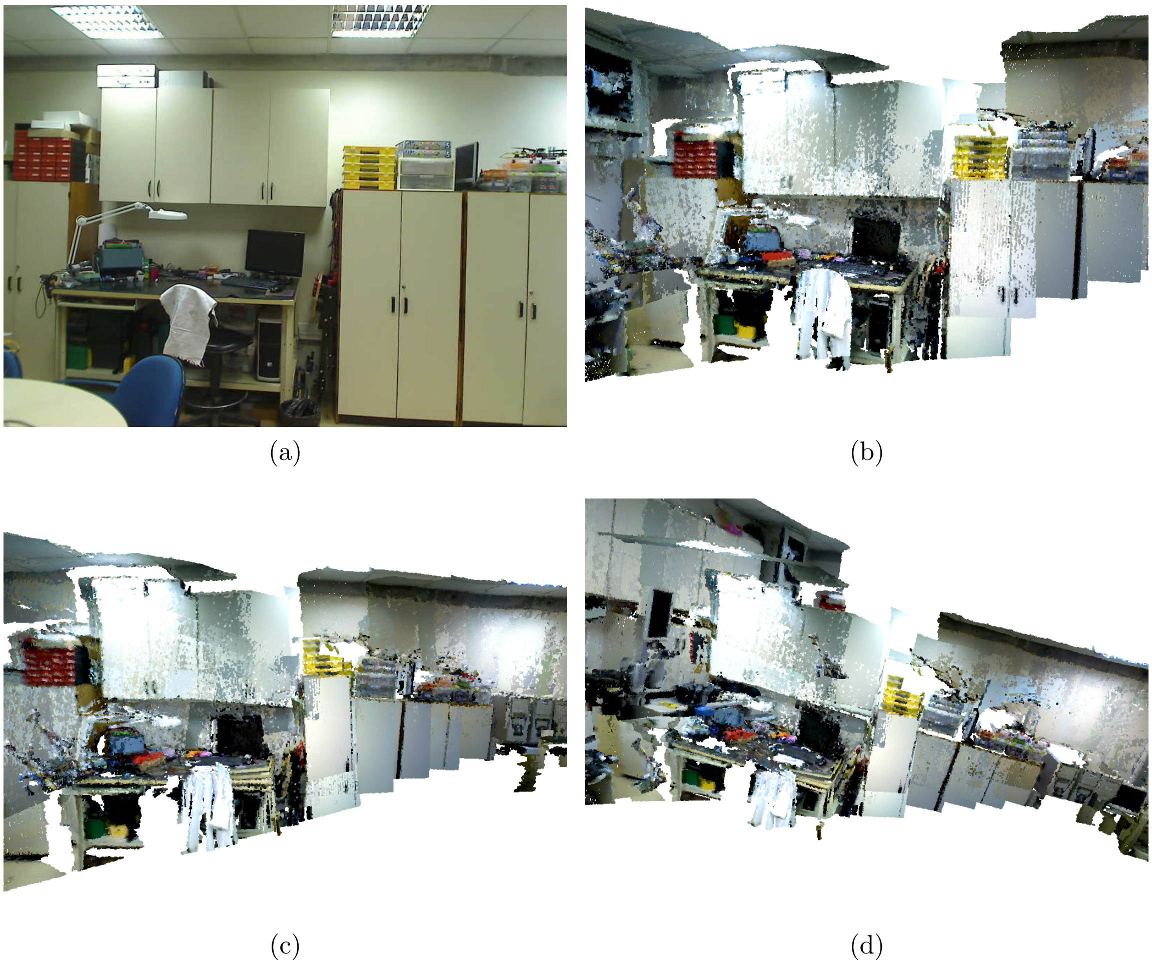


Figure 4.11: (a) real scene; (b) global registration by our full methodology; (c) global registration after strategy (A); (d) global registration after strategy (B). Comparison of loop closing region as obtained from all strategies. Figure (b) doesn't contain any visible errors. A discontinuity can be spotted in the cabinet on the right, in Figure (c); the same occurs in Figure (d), with several other discontinuities that makes it the worst map estimated.

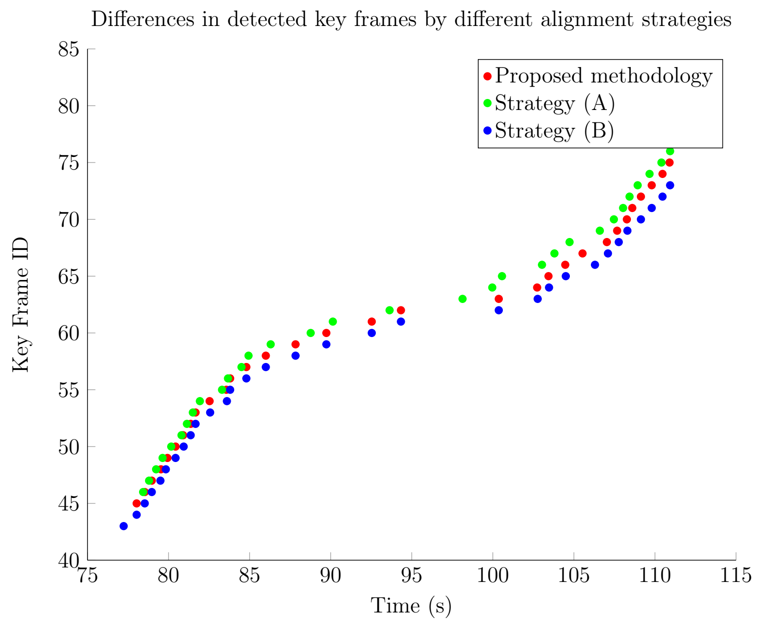


Figure 4.12: Instants at which each strategy detected their last 30 key frames.

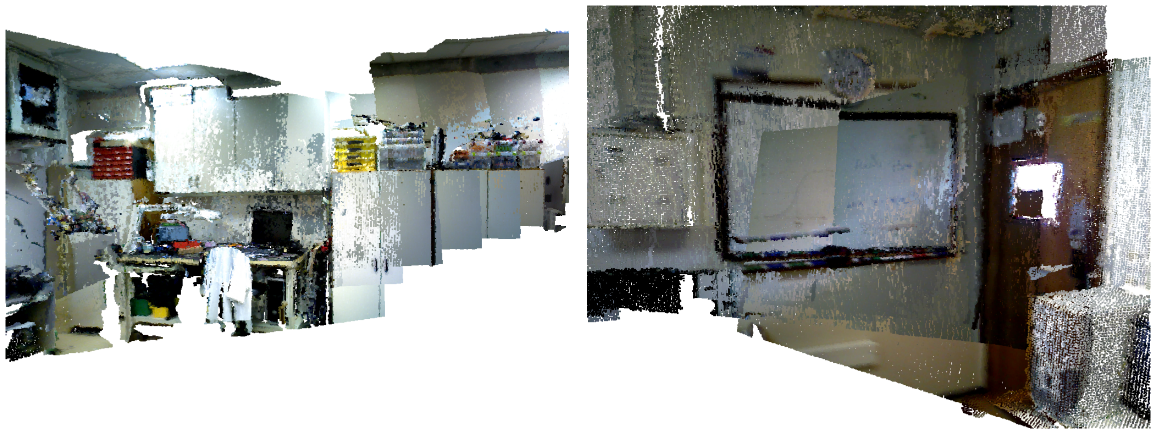


Figure 4.13: (a) Loop closure region with no visually detectable discontinuities; (b) errors due to pairwise misalignment (original scene is shown by Figure 4.2-b). Those are alignment results after performing the pairwise alignment by strategy (A) followed by our global optimization step on a key frame set computed by our methodology.

the only differences being for adjacent key frames aligned during pairwise registration.

As it can be seen in Figure 4.13, strategy (A) yields a global map visually consistent with our methodology after we used the set of key frames detected by our approach, despite local errors introduced by its pairwise registration (also shown in the same figure). After this change, the normalized error was changed to a mean of 1.38% and standard deviation 1.64.

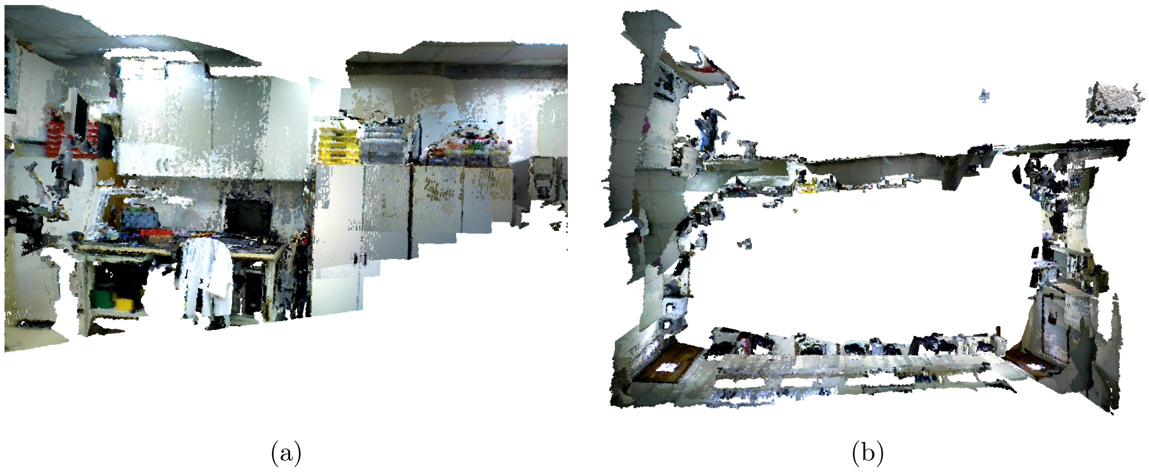


Figure 4.14: (a) Loop closure region with no visually detectable discontinuities; (b) errors due to pairwise misalignment (top view of the experimental set). Those are results after performing the pairwise alignment by strategy (B) followed by our global optimization step on a key frame set computed by our methodology.

In spite of producing a good loop closure, strategy (B) presented several discontinuities as illustrated by Figure 4.14. These discontinuities appear between key frames that were registered by a pairwise method, which suggests that their bonds were too weak and, therefore, were disregarded by the global optimization process. A close analysis revealed that no reliable matches were found for the key frame with acquisition id $t = 7$, shown in Figure 4.15-a. That is, adjacent key frames were matched by a very small number of keypoints – all of them fell below our threshold of 20 keypoints for an acceptable match. In fact, those matches had visually protruding discontinuities, as seen in Figure 4.15-b. This happened because the global optimization takes into account the pairwise registration transform in order to align adjacent key frames. Since the alignment provided by strategy (B) had a significant accumulated error at this point, the key point matching algorithm had to deal with a larger uncertainty than it was supposed to, resulting in spurious alignments with its adjacent frames.

Although this explains how different results could be obtained after global optimization, it still leaves questions as to why such results emerge. Since such differences are being observed in a region of the map where loop closure is expected to take place, our search for an explanation should concentrate on what is happening on the relaxed coarse transform algorithm, responsible for detecting loop closures and finding their respective transformations.

In a frame-by-frame analysis, we found that, in several occasions, the ambiguity between candidates of loop closing key frames was the cause of their incorrect align-

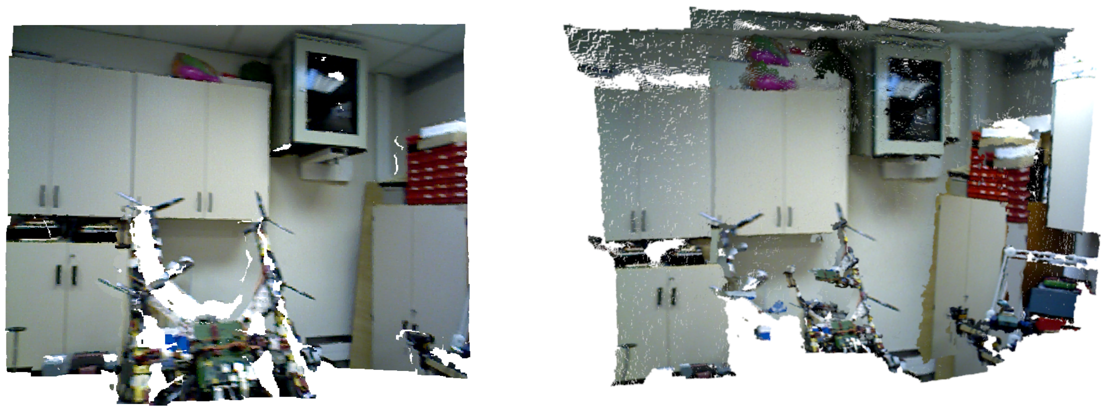

(a) (b)

Figure 4.15: (a) Key frame of acquisition ID $t = 7$, for which no reliable correspondents were found when globally optimizing the results of strategy (B) with key frames detected by our methodology; (b) key frame of acquisition ID $t = 7$, aligned with its predecessor. This alignment was discarded due to the small number of keypoint correspondences. This frame was the source of the discontinuity in experiment with strategy (B) when the key frame list provided by our method.

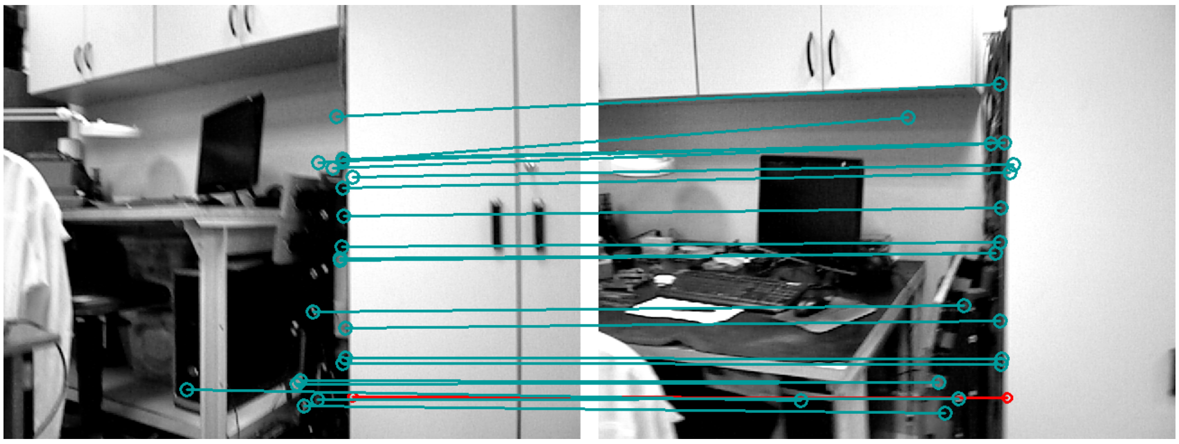
ment. As Figure 4.16 shows, the edge of the leftmost cabinet was vertically constant, allowing for keypoint matches that had a significant vertical error. In some other cases, similarities between the two adjacent cabinet led to a condition where several local minima could be found in which keypoints on the top cabinet were matched to the one in the bottom, but in these circumstances, the intersection between the frames after alignment fell below our acceptable threshold, which eliminated these wrong transforms.

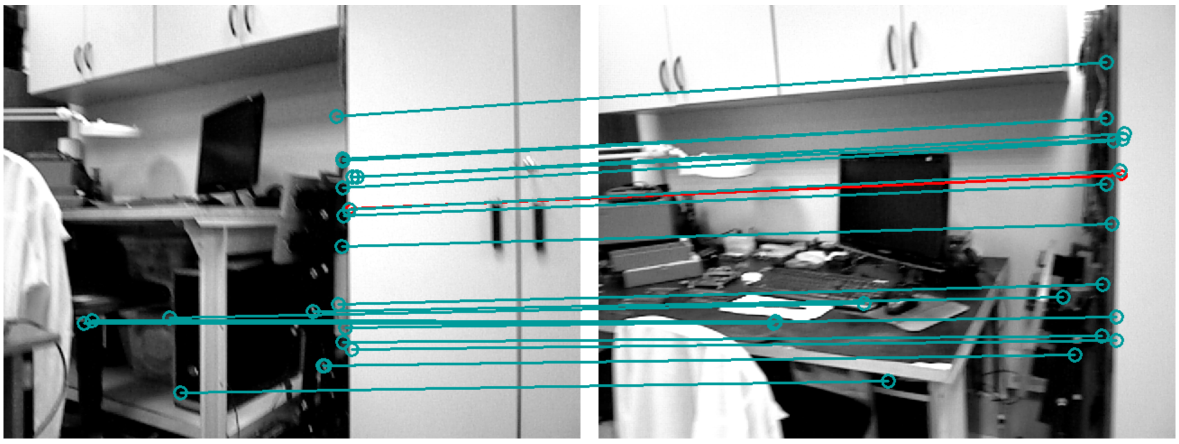
4.4 Comparison to RGB-D SLAM

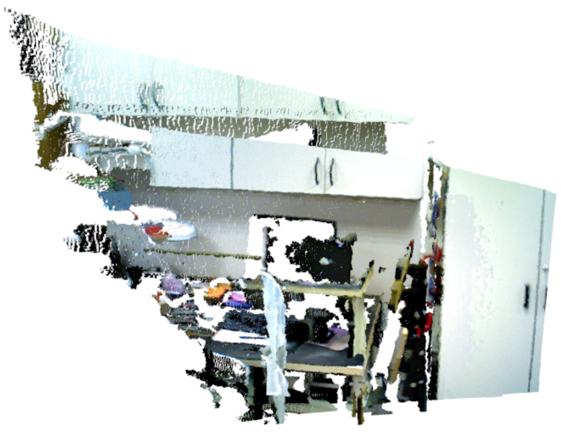
We decided to compare our approach to the RGB-D SLAM methodology [Endres et al., 2012], as it is similar to our methodology in the respect that it comprises a front-end where it performs pairwise feature matching followed by a fine alignment, and a back-end where global optimization is executed. Also, its implementation is available as open source, which makes our results easily reproducible.

This experiment was performed by using the same point cloud set from our laboratory, which was processed by our approach and the results were studied in the previous sections.

During the RGB-D SLAM alignment, the loop closure region was not detected, leaving the final map globally inconsistent, as Figure 4.17-a illustrates. The trajectory


(a)


(b)


(c)

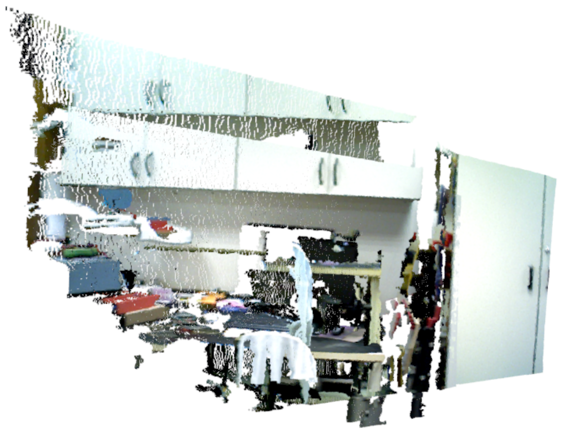

(d)

Figure 4.16: (a) Spurious keypoint match computed by strategy (A); (b) spurious keypoint match computed by strategy (B); (c) key frames aligned by transform computed from matches in fig. (a); (d) key frames aligned by transform computed from matches in fig. (b). Figs. (a) and (b) illustrate the keypoint match relative to the local minima found by the relaxed keypoint matcher algorithm, while (c) and (d) show the key frames aligned by these transforms, respectively.

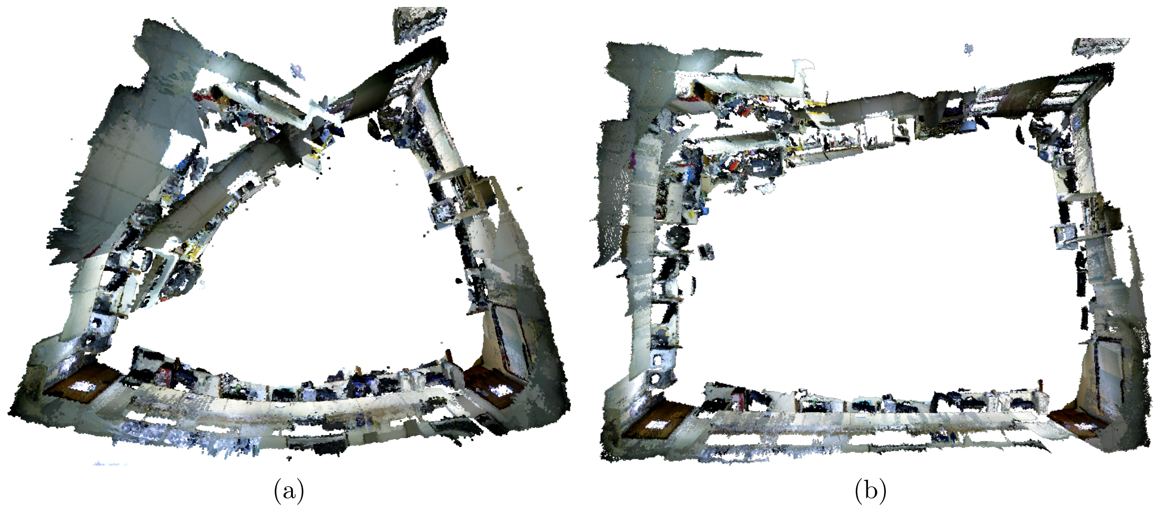


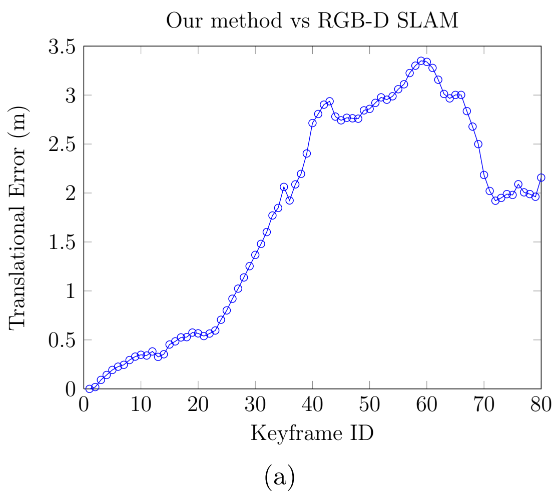
Figure 4.17: (a) RGB-D SLAM map; (b) map by our pairwise methodology. Not only the translational error of our methodology is smaller throughout the map, but its attitude estimates suffer from a smaller rotation drift.

corresponding to this map is shown by Figure 4.19, where the trajectory estimated by our methodology after loop closure is also displayed. Figure 4.18-a depicts the translational difference between the trajectory estimated by our methodology and the one computed by RGB-D SLAM.

Since the loop in our map was not closed by RGB-D SLAM, we did not perform a quantitative analysis by computing the distance between known points in the final map. However, for comparison purposes, we show in Figure 4.18-b the translational error of the pairwise alignment of our methodology, before global optimization took place. As can be seen, the accumulated error of our pairwise methodology is still smaller than the one by RGB-D SLAM. As Figure 4.17 shows, the map by RGB-D SLAM was prone to a larger drift in its attitude, while the attitude error was smaller in the map provided by our methodology. At the end of the trajectory, the RGB-D SLAM had a translational error of 2.16m, while our pairwise approach had an error of 1.76m. Although it cannot be seen on Figure 4.17 (which depicts a top view of our map), the trajectory computed by our pairwise methodology has a displacement of 1.55m in the z axis, with respect to the loop closed trajectory.

4.5 Qualitative Analysis

Our qualitative analysis sought to validate our mapping methodology in an environment that differs from the control environment. We selected a coffee room with several windows, which provided natural illumination to the room.



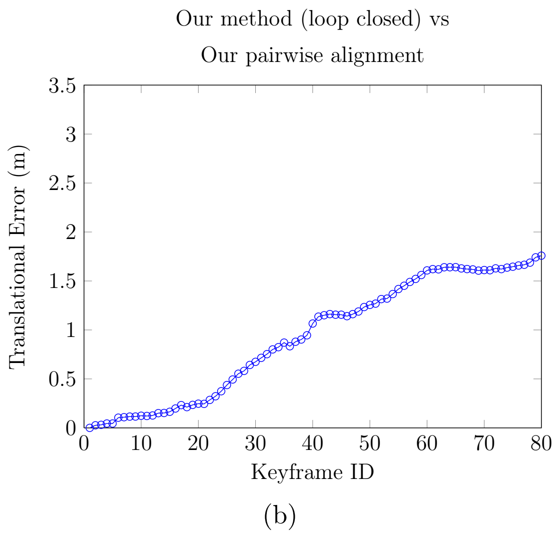


Figure 4.18: (a) Translational error from RGB-D SLAM; (b) translational error from our pairwise methodology. The error is computed using our loop closed map as a reference.

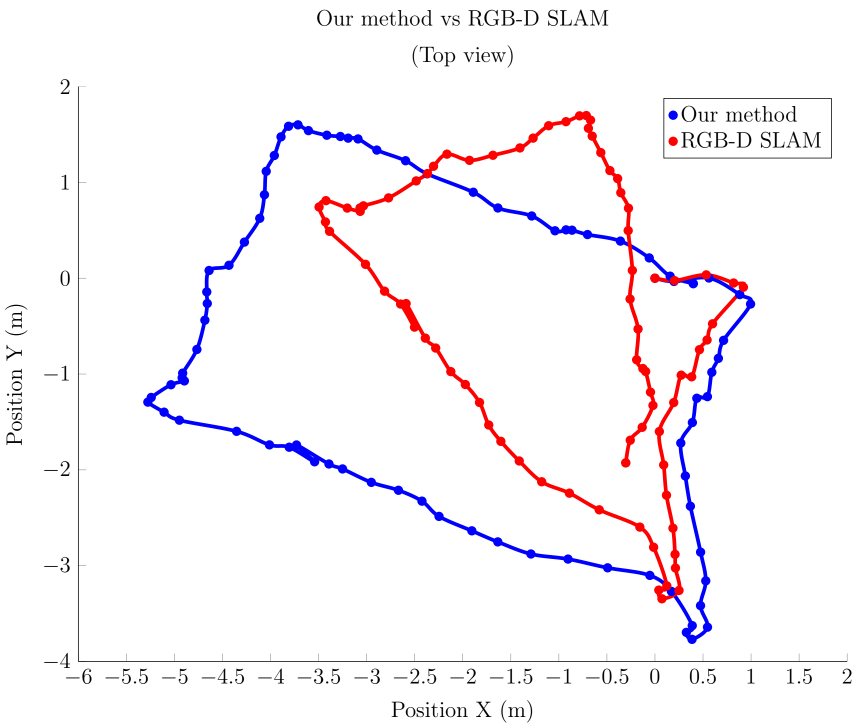
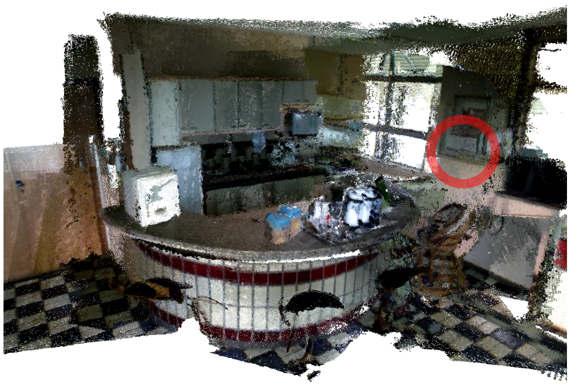


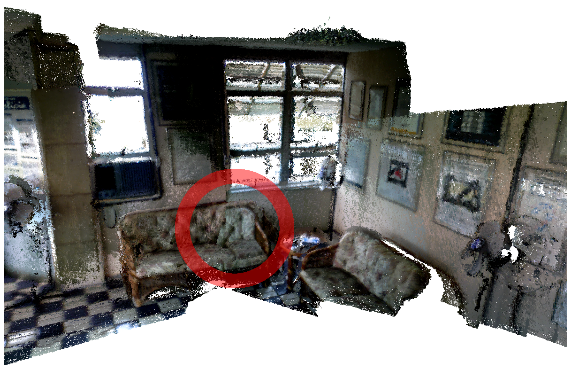
Figure 4.19: Top view trajectories from RGB-D SLAM and our methodology (after loop closure).

In this experiment, we did not map all the walls of this room, since some of the walls consisted mostly of windows, which would not provide us with enough geometric and color information. Furthermore, it is likely that the depth sensor would not be capable of capturing significant data due to sunlight interference, which could saturate the infrared patterns emitted by the sensor. Therefore, we did not perform global optimization, since there were no loops to be closed.

As Figure 4.20 illustrates, our methodology was capable of completing the mapping process without major divergences. Small misalignments still exist, mostly close to the windows. They can be explained by the fact that, near windows, the exposure time of the camera got lower, which could hide visual features and making it difficult to perform the photoconsistent alignment.

We also executed the RGB-D SLAM methodology with the same point cloud set in order to have another map for comparison purposes. Shown by Figure 4.21, the map provided by RGB-D SLAM had a smooth representation of a couch, in contrast to a small discontinuity in the map generated by our methodology. However, it contains several visible discontinuities throughout the whole scene.


(a)


(b)

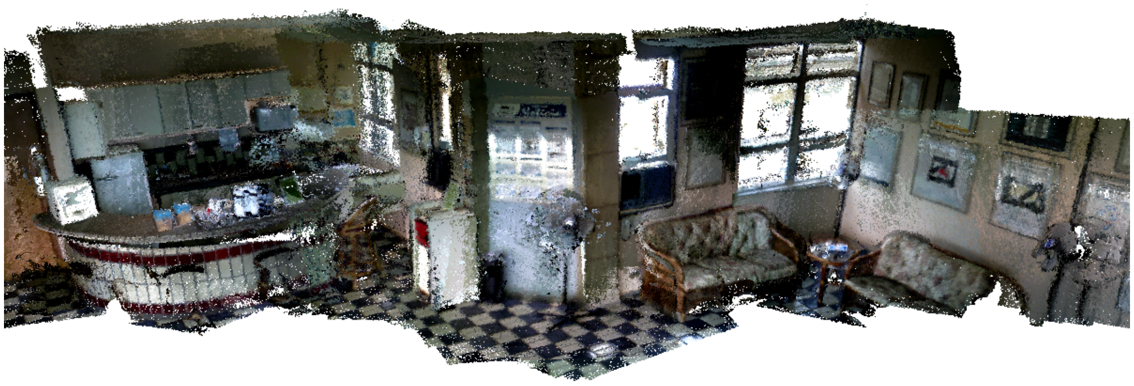

(c)

Figure 4.20: Mapping results for a naturally lit cafe room by using our methodology. (a) A misalignment on a frame hanging on the wall; (b) a misalignment on the couch; (c) the whole scene.

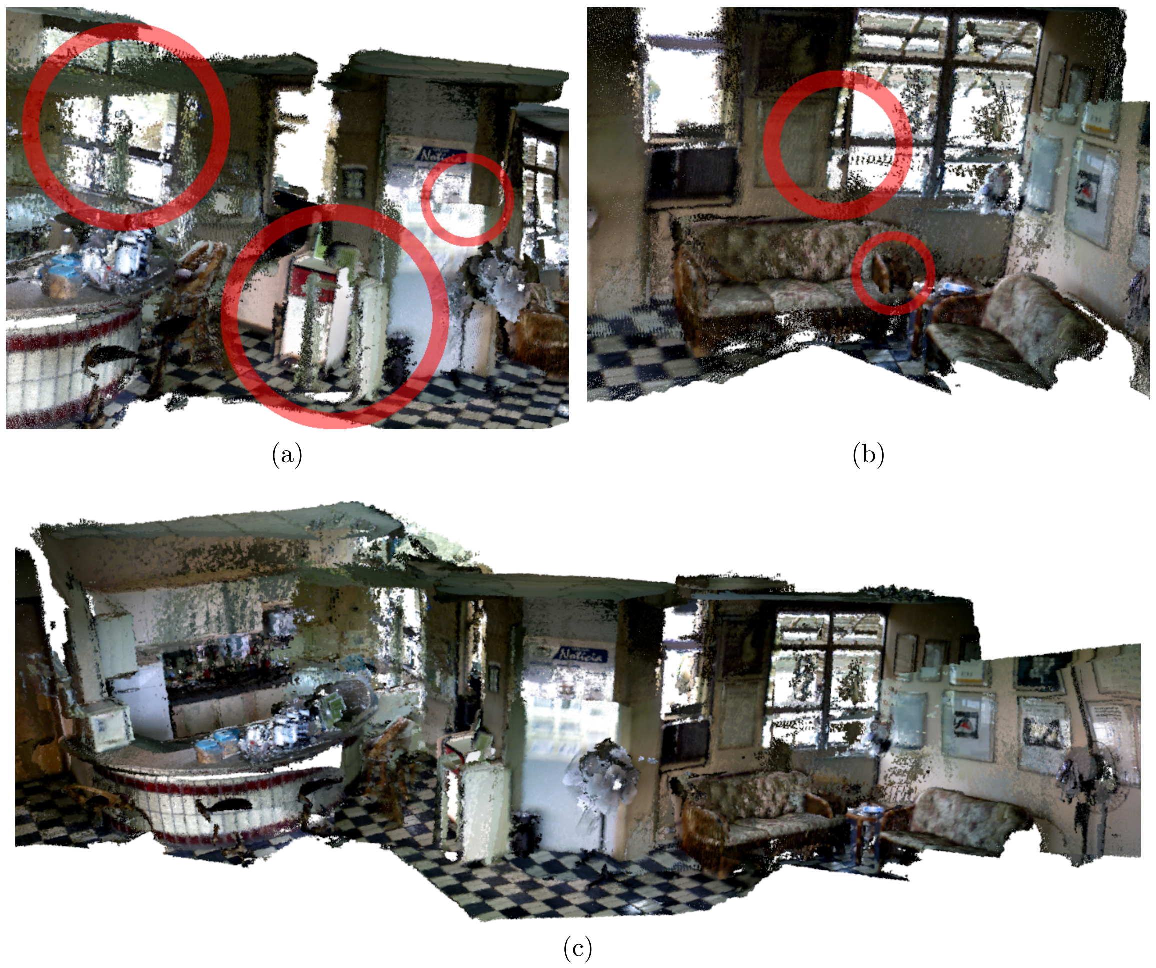


Figure 4.21: Mapping results for a naturally lit cafe room by using RGB-D SLAM. (a) Major misalignments can be spotted at the wall board (right), trash bin (middle) and the leftmost window; (b) misalignments at the window and at the corner of the couch; (c) the whole scene.

Chapter 5

Conclusions

This work has presented a methodology for creating globally consistent maps of static environments by using depth, color and inertial information. The global consistency property specifies that any two adjacent frames captured by the depth sensor must be represented without discontinuities in the final map – a property that becomes difficult to maintain when a particular region of the environment is revisited after a long time is spent mapping other areas.

Although several methodologies have been presented in the literature for the problem of environment mapping, few publications have devoted some attention to the possibility of fusing inertial information to RGB-D data for mapping purposes. Considering that many robots include MARG sensors, this methodology could be used with no additional cost if an RGB-D sensor were present. Furthermore, with the growth of devices that incorporate inertial sensors, we expect the prices of these sensors to further drop in the near future, making it feasible for a larger number of applications to benefit from them.

In this dissertation, we have used the inertial information in a keypoint matching algorithm that seeks the best correspondences at the same time it preserves Euclidean constraints. It is also used to discard false positives in the loop closure detection procedure, as loop closing key frames must have a small angular distance from each other.

In our experiments, we not only have validated the proposed method in an experiment, but we also studied in detail the robustness of the global optimization module, particularly analyzing its response to a poor pairwise alignment and to ambiguities in the environment. We have also seen that disabling the input from the MARG sensors, and our keypoint matcher by using the brute force matcher from OpenCV instead, the pairwise alignment ultimately led to a map with several inconsistencies after global optimization, especially in regions with few color features.

5.1 Known issues of our work

As we have previously discussed, regions with a significantly varying magnetic field would render our methodology useless, since the keypoints matching stage would be too much likely to find wrong correspondences, which would ultimately lead to either misalignments or even divergence in pairwise registration.

Yet another problematic condition we have observed is that points that lie outside the depth sensor practical range (typically up to 3.5m in RGB-D sensors) are subject to a great deal of noise and uncertainty, and such points can not only introduce a large uncertainty in the coarse alignment stage, but also make it unfeasible if we employ a keypoint descriptor that makes use of normal information at each point. This happens because, by introducing depth error, the normals estimated at each point may vary largely.

5.2 Future Work

A few future work directions may be considered:

- **Image Deblurring.** Image blurring is of major concern especially when the scene being mapped lacks geometric features. The RGB-D mapping field would greatly benefit from advances in RGB deblurring techniques, especially if they become computationally inexpensive.

- **Sensor synchronization.** Our experimental setup relied on slow sensor movements to mitigate the synchronization issue. Real world applications, however, may not impose such a restriction, which means that further research must be made seeking to synchronize MARG and RGB-D sensors in order to make this technology commercially available.

- **Optimize the feature matcher algorithm.** Not only is it possible to increase computational speed by parallelizing the feature correspondence method, but we also can decrease its complexity by avoiding similar configurations from being evaluated multiple times. Another principle for reducing computational expenses would be to terminate the algorithm as soon as it finds a score that lies below a certain threshold, in contrast to trying all combinations and returning the best possible configuration. This would prove to be very useful for platforms with a relatively small computational power, such as notebooks.

- **Avoid unconnected key frames on global optimizer.** As we saw during our experiments, it is possible for the global optimization stage to generate unconnected graphs if the bond between adjacent key frames is too weak. The usage of the relaxed keypoint matching algorithm in such circumstances might, at least, help to mitigate this problem.

- **Disregard points outside safe range.** We know that points outside the depth sensor's practical range display a great amount of noise, which makes them unreliable for mapping purposes. Future works should be concerned with this problem, possibly by eliminating these points. It is important to notice, however, that such strategy could make the point cloud too small, leaving it useless for the purpose of registration.

- **Reconstruct dynamic environments.** Our methodology has been developed to reconstruct static environments. This means that the subject being reconstructed must remain still when the point clouds are being grabbed. This requirement may not be available under some circumstances, in which objects from the scene can be removed or deformed during the mapping procedure. This is a field that may benefit from inertial data.

Bibliography

Agrawal, M. and Konolige, K. (2008). Censure: Center surround extremas for real-time feature detection and matching. In *European Conference on Computer Vision (ECCV)*.

Bay, H., Ess, A., Tuytelaars, T., and Van Gool, L. (2008). Speeded-Up Robust Features (SURF). *Comput. Vis. Image Underst.*, 110(3):346--359. ISSN 1077-3142.

Beardsley, P. A., Zisserman, A. P., and Murray, D. W. (1997). Sequential updating of projective and affine structure from motion. *International Journal of Computer Vision*, 23(3):235--259.

Besl, P. J. and McKay, N. D. (1992). A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 14(2):239--256. ISSN 0162-8828.

Biber, P. and Strasser, W. (2003). The normal distributions transform: a new approach to laser scan matching. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 2743 – 2748 vol.3. ISSN .

Blais, F. (2004). Review of 20 years of range sensor development. *Journal of Electronic Imaging*, 13(1):231–243.

Blanz, V. and Vetter, T. (1999). A morphable model for the synthesis of 3d faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, SIGGRAPH ’99, pages 187--194, New York, NY, USA. ACM Press/Addison-Wesley Publishing Co.

Borrmann, D., Elseberg, J., Lingemann, K., Nüchter, A., and Hertzberg, J. (2008). Globally consistent 3d mapping with scan matching. *Robot. Auton. Syst.*, 56(2):130--142.

Bosse, M. and Zlot, R. (2009). Continuous 3d scan-matching with a spinning 2d laser. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4244--4251, Piscataway, NJ, USA. IEEE Press.

Bosse, M. and Zlot, R. (2010). Place recognition using regional point descriptors for 3d mapping. In Howard, A., Iagnemma, K., and Kelly, A., editors, *Field and Service Robotics*, volume 62 of *Springer Tracts in Advanced Robotics*, pages 195–204. Springer Berlin Heidelberg.

Bosse, M., Zlot, R., and Flick, P. (2012). Zebedee: Design of a spring-mounted 3-d range sensor with application to mobile mapping. *Robotics, IEEE Transactions on*, 28(5):1104--1119. ISSN 1552-3098.

Brown, B. and Rusinkiewicz, S. (2004). Non-rigid range-scan alignment using thin-plate splines. In *Symposium on 3D Data Processing, Visualization, and Transmission*.

Brown, B. and Rusinkiewicz, S. (2007). Global non-rigid alignment of 3-D scans. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 26(3).

Brown, L. G. (1992). A survey of image registration techniques. *ACM Comput. Surv.*, 24(4):325--376. ISSN 0360-0300.

Chen, C.-S., Hung, Y.-P., and Cheng, J.-B. (1998). A fast automatic method for registration of partially-overlapping range images. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 242--, Washington, DC, USA. IEEE Computer Society.

Chen, C.-S., ping Hung, Y., and bo Cheng, J. (1999). Ransac-based darces: A new approach to fast automatic registration of partially overlapping range images. 21:1229--1234.

Davison, A. J., Reid, I. D., Molton, N. D., and Stasse, O. (2007). MonoSLAM: Real-Time Single Camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29(6):1052--1067. ISSN 0162-8828.

des Bouvrie, B. (2011). Improving RGBD Indoor Mapping with IMU data. Master's thesis, Delft University of Technology.

Einhorn, E., Schröter andter, C., and Gross, H. (2010). Can't take my eye off you: Attention-driven monocular obstacle detection and 3d mapping. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, pages 816--821. ISSN 2153-0858.

Einhorn, E., Schröter, C., Böhme, H.-J., and Gross, H.-M. (2007). A hybrid kalman filter based algorithm for real-time visual obstacle detection. In *EMCR*.

Einhorn, E., Schröter, C., and Groß, H.-M. (2009). Monocular scene reconstruction for reliable obstacle detection and robot navigation. In Petrovic, I. and Lilienthal, A. J., editors, *ECMR*, pages 7–12. KoREMA.

Elfes, A. (1989). Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46--57. ISSN 0018-9162.

Endres, F., Hess, J., Engelhard, N., Sturm, J., Cremers, D., and Burgard, W. (2012). An evaluation of the RGB-D SLAM system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, St. Paul, MA, USA.

Feldmar, J. and Ayache, N. (1994). Rigid, affine and locally affine registration of free-form surfaces. *IJCV*, 18(18):99--119.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381--395. ISSN 0001-0782.

Fitzgibbon, A. W. and Zisserman, A. (1998). Automatic camera recovery for closed or open image sequences. In *European Conference on Computer Vision (ECCV)*, pages 311--326. Springer-Verlag.

Frédéric and Bosché (2010). Automated recognition of 3d cad model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction. *Advanced Engineering Informatics*, 24(1):107–118. ISSN 1474-0346. Informatics for cognitive robots.

Garey, M. R. and Johnson, D. S. (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA. ISBN 0716710455.

Harris, C. and Pike, J. (1988). 3d positional integration from image sequences. *Image and Vision Computing*, 6(2):87 -- 90. ISSN 0262-8856. 3rd Alvey Vision Meeting.

Henry, P., Krainin, M., Herbst, E., Ren, X., and Fox, D. (2010). Rgbd mapping: Using depth cameras for dense 3d modeling of indoor environments. In *In RGB-D: Advanced Reasoning with Depth Cameras Workshop in conjunction with RSS*.

Huang, A. S., Bachrach, A., Henry, P., Krainin, M., Maturana, D., Fox, D., and Roy, N. (2011). Visual Odometry and Mapping for Autonomous Flight Using an RGB-D Camera. In *Int. Symposium on Robotics Research (ISRR)*, Flagstaff, Arizona, USA.

Huhle, B., Magnusson, M., Strasser, W., and Lilienthal, A. (2008). Registration of colored 3d point clouds with a kernel-based extension to the normal distributions transform. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4025 –4030. ISSN 1050-4729.

Hyun, D., Yun, I. D., and Lee, S. U. (1998). Registration of multiple range views using the reverse calibration technique.

Ikemoto, L. (2003). A hierarchical method for aligning warped meshes. In *Proc. Intl. Conf. on 3D Digital Imaging and Modeling*, pages 434--441.

Izadi, S., Kim, D., Hilliges, O., Molyneaux, D., Newcombe, R., Kohli, P., Shotton, J., Hodges, S., Freeman, D., Davison, A., and Fitzgibbon, A. (2011). Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pages 559--568, New York, NY, USA. ACM.

Jain, R. K. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. Wiley, 1 edition. ISBN 0471503363.

Khoshelham, K. and Elberink, S. O. (2012). Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437--1454. ISSN 1424-8220.

Kim, D., Sun, J., Oh, S. M., Rehg, J., and Bobick, A. (2006). Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 518--525. ISSN 1050-4729.

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

Kvalbein, M. (2012). The use of a 3d sensor (kinect$^{\text{TM}}$) for robot motion compensation: The applicability in relation to medical applications. Master's thesis, University of Oslo, Department of Informatics.

Li, H., Sumner, R. W., and Pauly, M. (2008). Global correspondence optimization for non-rigid registration of depth scans. In *Proceedings of the Symposium on Geometry Processing*, SGP '08, pages 1421--1430, Aire-la-Ville, Switzerland, Switzerland. Eurographics Association.

Lu, F. and Milios, E. (1997). Globally consistent range scan alignment for environment mapping. *AUTONOMOUS ROBOTS*, 4:333--349.

Madgwick, S., Harrison, A., and Vaidyanathan, R. (2011). Estimation of imu and marg orientation using a gradient descent algorithm. In *Rehabilitation Robotics (ICORR), 2011 IEEE International Conference on*, pages 1 –7. ISSN 1945-7898.

Magnusson, M., Lilienthal, A., and Duckett, T. (2007). Scan registration for autonomous mining vehicles using 3d-ndt: Research articles. *J. Field Robot.*, 24(10):803--827. ISSN 1556-4959.

Matthies, L., Kanade, T., and Szeliski, R. (1989). Kalman Filter-based Algorithms for Estimating Depth from Image Sequences. *International Journal of Computer Vision*, 3(3):209--238.

MicroStrain (2006). 3dm-gx1 gyro enhanced orientation sensor. Technical product overview, MicroStrain Inc.

Nascimento, E. R., Schwartz, W. R., Oliveira, G. L., Vieira, A. W., Campos, M. F. M., and Mesquita, D. B. (2012). Appearance and geometry fusion for enhanced dense 3d alignment. *Conference on Graphics, Patterns and Images, 25. (SIBGRAPI).*

Neto, A., Victorino, A., Fantoni, I., and Zampieri, D. (2011). Robust horizon finding algorithm for real-time autonomous navigation based on monocular vision. In *Intelligent Transportation Systems (ITSC), 14th International IEEE Conference on*, pages 532--537. ISSN 2153-0009.

Newcombe, R. A. and Andrew, J. (2010). Live Dense Reconstruction with a Single Moving Camera Davison, CVPR 2010. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

Newcombe, R. A., Lovegrove, S. J., and Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2320--2327. IEEE. ISSN 1550-5499.

Nilsson, N. J. (1984). Shakey the robot. Technical report Technical Note 323, SRI International.

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(6):559--572.

Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In *IEEE International Conference on Computer Vision*, volume 2, pages 1508--1511.

Rusinkiewicz, S. and Levoy, M. (2001). Efficient variants of the ICP algorithm. In *Third International Conference on 3D Digital Imaging and Modeling (3DIM).*

Shoemake, K. (1985). Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '85, pages 245--254, New York, NY, USA. ACM.

Shpunt, A., Tikva, P., Zalevsky, Z., and Ha'ayin, R. (2008). Depth-varying light fields for three dimensional sensing. Patent US 2008/0106746 A1, PrimeSense.

Steinbruecker, F., Sturm, J., and Cremers, D. (2011). Real-Time Visual Odometry from Dense RGB-D Images. In *Workshop on Live Dense Reconstruction with Moving Cameras at the Intl. Conf. on Computer Vision (ICCV).*

Stückler, J. and Behnke, S. (2012). Integrating depth and color cues for dense multi-resolution scene mapping using rgb-d cameras. *IEEE International Conference on Multisensor Fusion and Information Integration (MFI).*

Takeuchi, T., Nakashima, T., Nishimura, K., and Hirose, M. (2011). PRIMA: Parallel reality-based interactive motion area. In *SIGGRAPH Posters'11*, pages 80–80.

Ulrich, I. and Nourbakhsh, I. (2000). Appearance-based obstacle detection with monocular color vision. In *Proceedings of AAAI 2000.*

Whelan, T., McDonald, J., Kaess, M., Fallon, M., Johannsson, H., and Leonard, J. (2012). Kintinuous: Spatially extended KinectFusion. In *RSS Workshop on RGB-D: Advanced Reasoning with Depth Cameras*, Sydney, Australia.