JUAN CARLOS PERAFÁN VILLOTA

ADAPTIVE REGISTRATION USING 2D AND 3D FEATURES FOR INDOOR SCENE RECONSTRUCTION

São Paulo
2016

JUAN CARLOS PERAFÁN VILLOTA

ADAPTIVE REGISTRATION USING 2D AND 3D FEATURES FOR INDOOR SCENE RECONSTRUCTION

Thesis submitted to Escola Politécnica da Universidade de São Paulo in partial fullfillment of the requirements for the degree of Doctor of Science.

São Paulo
2016

JUAN CARLOS PERAFÁN VILLOTA

ADAPTIVE REGISTRATION USING 2D AND 3D FEATURES FOR INDOOR SCENE RECONSTRUCTION

Thesis submitted to Escola Politécnica da Universidade de São Paulo in partial fullfillment of the requirements for the degree of Doctor of Science.

Area of research:

Systems Engineering

Advisor:

Prof. Dr. Anna Helena Reali Costa

São Paulo
2016

Catalogação-na-publicação

Perafán Villota, Juan Carlos
Adaptive registration using 2D and 3D features for indoor scene reconstruction / J. C. Perafán Villota -- São Paulo, 2016.
75 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Imagem 3D 2.Imagem 3D (Reconstrução) 3.Descritores locais 4.Detectores de saliências I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

I dedicate this thesis to my beloved family. Juan and Doris, my parents. Maritza, Janeth and Erika, my sisters. Juan Sebastian, my son. Carla, Pedro, Lucas and Ramon, my nephews. Thank you for your unbounded loving gestures, let me assure you that these signs were recorded by my 576-megapixel camera .

Acknowledgments

First and foremost, I thank God.

I would like to thank my advisor, professor Anna Helena Reali, for accepting me into her group. Anna has been a constant source of encouragement and enthusiasm during these years, not only by supporting me in this journey into the image processing world, but also by demanding a high quality of work in my research.

I would additionally like to thank professor Denis Wolf for his warm reception in LRM. He fully supported my stay in ICMC/USP with his interest, and his extensive knowledge in mobile robots.

I also thank all the people of the LTI (Laboratório de Técnicas Inteligentes - USP) for the valuable discussions and comments regarding my research, specially Walter, Felipe and Ricardo, who provided a friendly and cooperative atmosphere at work and also useful feedback and insightful comment on my work.

I also appreciate the financial support from CAPES during my Ph.D study.

Finally, my deepest gratitude goes to my family for their unconditional love and unflagging support all these years.

Familia Perafán Villota, esta tesis es un sentido homenaje hacia cada uno de ustedes.

Resumo

O alinhamento entre pares de nuvens de pontos é uma tarefa importante na construção de mapas de ambientes em 3D. A combinação de características locais 2D com informação de profundidade fornecida por câmeras RGB-D são frequentemente utilizadas para melhorar tais alinhamentos. No entanto, em ambientes interno com baixa iluminação ou pouca textura visual o método usando somente características locais 2D não é particularmente robusto. Nessas condições, as características 2D são difíceis de serem detectadas, conduzindo a um desalinhamento entre pares de quadros consecutivos. A utilização de características 3D locais pode ser uma solução para se extrair tais características diretamente de pontos 3D. Como as condições de variações em cenas reais em ambientes internos são inevitáveis, essa tese apresenta um novo sistema desenvolvido com o objetivo de melhorar o alinhamento entre pares de quadros usando uma combinação adaptativa de características 2D e 3D. Tal combinação esta baseada nos níveis de estrutura geométrica e de textura visual contidos em cada cena. Esse sistema foi testado com conjuntos de dados RGB-D, incluindo vídeos com movimentos irrestritos da câmera e mudanças naturais na iluminação. Os resultados experimentais mostram que a nossa proposta supera aqueles métodos usando características 2D e 3D separadamente. Como resultado, foi possível obter um sistema que melhora a precisão nos alinhamento das cenas.

Palavras-Chave – Reconstrução 3D, Registro de imagens, Descritores locais, Detectores de saliências, sensor RGB-D.

Abstract

Pairwise alignment between point clouds is an important task in building 3D maps of indoor environments with partial information. The combination of 2D local features with depth information provided by RGB-D cameras are often used to improve such alignment. However, under varying lighting or low visual texture, indoor pairwise frame registration with sparse 2D local features is not a particularly robust method. In these conditions, features are hard to detect, thus leading to misalignment between consecutive pairs of frames. The use of 3D local features can be a solution as such features come from the 3D points themselves and are resistant to variations in visual texture and illumination. Because varying conditions in real indoor scenes are unavoidable, we propose a new framework to improve the pairwise frame alignment using an adaptive combination of sparse 2D and 3D features based on both the levels of geometric structure and visual texture contained in each scene. Experiments with datasets including unrestricted RGB-D camera motion and natural changes in illumination show that the proposed framework convincingly outperforms methods using 2D or 3D features separately, as reflected in better level of alignment accuracy.

Keywords – 3D reconstruction, Image registration, Local descriptors, Keypoint detectors, RGB-D sensor.

List of Figures

List of Tables

List of Abbreviations

SLAM	Simultaneous Localization and Mapping
ICP	Iterative Closest Point
RANSAC	Random Sample Consensus
ANN	Approximate Nearest Neighbor
SIFT	Scale-Invariant Feature Transform
DoG	Difference-of-Gaussian
SURF	Speeded-Up Robust Feature
LoG	Laplacian-of-Gaussian
ORB	Oriented FAST and Rotated BRIEF
FAST	Features from Accelerated Segment Test
BRIEF	Binary Robust Independent Elementary Features
NARF	Normal Aligned Radial Feature
ISS3D	Intrinsic Shape signatures
FPFH	Fast Point Feature Histogram
PFH	Point Feature Histograms
KNN	K-nearest neighbor
K-D tree	K-Dimensional tree
FLANN	Fast Library Approximate Nearest Neighbor
PCL	Point Cloud Library
SUSAN	Smallest Univalue Segment Assimilating Nucleus
SURE	Surface Entropy
SHOT	Unique Signature of Histograms
ReACC	Registration by Adaptive Combination of Cues
RPE	Relative Pose Error
RMSE	Root Mean Squared Error

Contents

Chapter 1

Introduction

This thesis is made in the context of methodologies applied to the mobile robotic problem known as Simultaneous Localization and Mapping (SLAM). More specifically, in problems presented in the method known as pairwise registration which is applied to build 3D maps linking partial information extracted by sensors that capture depth and visual information from the surrounding environment.

In order to understand the crucial role played by the pairwise registration in the SLAM problem, section 1.1 presents an overview of this problem and the techniques to resolve it when using visual information (Visual SLAM); the section continues with the commonly used steps to build 3D maps, and ends with a brief explanation of pairwise registration and its associated issues. Section 1.2 provides a brief overview of the framework developed in this thesis and the contributions achieved. Finally, Section 1.4 presents an outline of the organization for the rest of this document.

1.1 Simultaneous Localization and Mapping (SLAM)

SLAM is the problem facing a mobile robot when neither its motion nor the structure of the surrounding environment is known in advance and the goal is to estimate both simultaneously. SLAM builds a representation of the state (pose) of the robot and an environment map, which evolves in response to motion and new robot sensor measurements. This problem appears to be the chicken-and-egg problem, since the localization needs a map to infer the robot location and the mapping concurrently needs the robot location to infer a map.

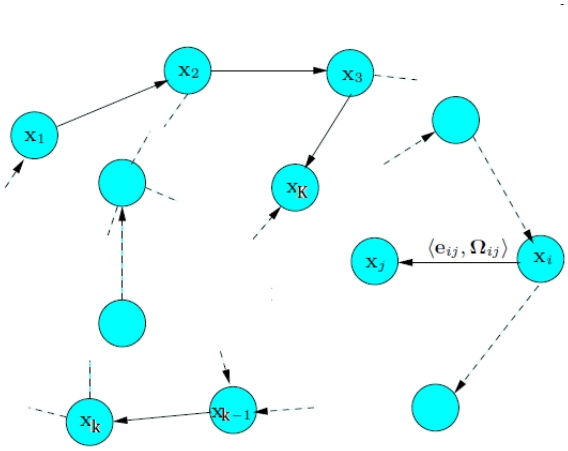
There are many algorithms used to solve this problem in two dimensions (ELIAZAR;

PARR, 2003),(DURRANT-WHYTE; BAILEY, 2006),(BAILEY; DURRANT-WHYTE, 2006). However, 3D maps provide more and better information to different applications such as robot navigation, manipulation, semantic mapping, and telepresence. Depth and color information provided by Lidar sensors and digital cameras, respectively, were integrated in order to increase 3D maps accuracy; however, the costs rise too. A recent development of low-cost sensors RGB-D providing less accurate color for a small field of view (in a range of approximately 60° of visual angle) and depth information (in a range varying from about 3cm to 3m depth), has renewed the interest of researchers in finding new methods to solve the SLAM problem (ENDRES et al., 2012),(HENRY et al., 2012),(RUSU et al., 2008), (SCHERER et al., 2012),(PRAKHYA; QAYYUM, 2015).

In the past the problem of estimating the robot's pose was resolved using techniques such as the Extended Kalman Filter (*EKF*) and the Particle Filter (GRISETTI et al., 2007),(THRUN et al., 2005),(ELIAZAR; PARR, 2003); recently, the graph models are considered a good alternative for pose estimation (KUMMERLE et al., 2011),(BORRMANN et al., 2008).

A graph model combining both graph and probabilistic theories describing the SLAM problem is known as Graph-based SLAM (see Figure 1), and was firstly proposed by Lu and Milios in 1997 (LU; MILIOS, 1997). In this graph every node represents a pose of the robot during mapping which is labelled with their position x_k; meanwhile, the edges represents spatial constraints Ω_{ij} that result from odometer measurements u_k or from relative pose observations z_k relating the poses between two nodes i and j.

Figure 1. Pose-Graph describing the SLAM problem, where each node has the pose of the sensor and each edge has the constraints between connected nodes.

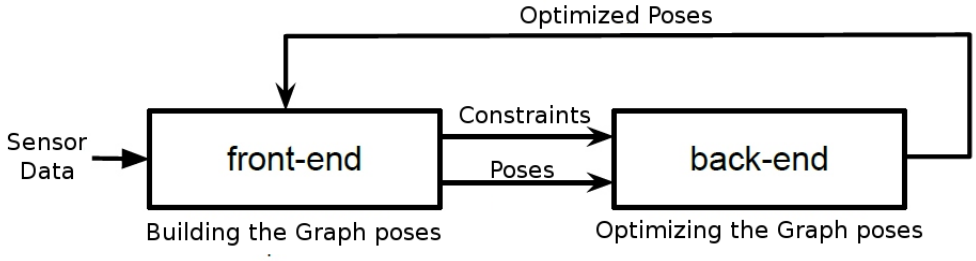


Source: Reproduced from (GRISETTI et al., 2010).

A typical Graph-based SLAM system is illustrated in Figure 2 and consists of two

blocks, i.e.,the front-end that builds the graph, and the back-end that optimizes the graph poses given the edges constraints.

Figure 2. General representation of the Graph Based SLAM system containing two blocks: the front-end block used to build the graph poses and the back-end block used to optimize the graph poses.



Source: Author.

The goal of Graph-based SLAM is to find the configuration of a set of poses $x_1, ..., x_n$ that minimizes error e introduced by a set of given relative pose observations z. The optimization of the graph consists in finding the optimal configuration of nodes to respect all the given constraints.

The transformation T maximizing the overlap between two consecutive frames is known as the expected relative pose $\hat{z}_{ij}$. In particular, in the case of 3D points, this transformation is a perspective projection composed of a rotation matrix R and a translation vector t in 3 dimensions. The idea is to transformthe points of a frame and to calculate the average error regarding the point of the ohter frame.

This method presents a problem related with the initial belief that one image is a perfect rotation and translation of another image. This assumption is not possible since the uncertainty due to inherent noisy measurements of the sensor and the correspondences by itself. Therefore, since the correct correspondences are not known, it is generally imposible to determine the optimal transformation in one step. Then, an iterative method to estimate the parameters R and t of the rigid transformation must be used. An algorithm called Iterative Closest Point (ICP) (ZHANG, 1994) is commonly used to solve this problem as follows: Firstly a possible correspondence between the points in the two frames is calculated, then a transformation T necessary to align them is found. Consequently, this transformation applies to align the two frames and to calculate de function error $E(R, t)$. If $E(R, t)$ is less than a threshold value given, the alignment is considered achieved. Otherwise, the steps are repeated until a correct alignment is reached.

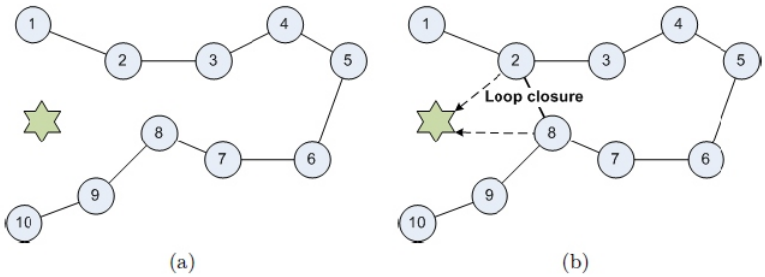
The convergence of this algorithm is only guaranteed if the starting misalignment

between the two sets of points is quite small, since it might reach a local minimum if the two sets of points analyzed are not closer from each other. Furthermore, the basic ICP has a high computational cost, since in the first step it uses all points in both sets to calculate correspondences. Moreover, ICP is sensitive to outliers (bad correspondences).

One technique vastly used to decrease the number of correspondences for ICP is known as Feature-based Sampling (THRUN et al., 2005), which consists in detecting a subset of sparse interest points from within the whole set of points enabling an accurate alignment between frames, thereby increasing the efficiency of the ICP . On the other hand, to guarantee a good starting position for ICP, an algorithm called Random Sample Consensus (RANSAC) (FISCHLER; BOLLES, 1981) is used. RANSAC is quite resistant to outliers and can provide a good initial transformation for ICP.

In order to minimize the error, graph optimization relies on constraints between the nodes. With more constraints, the cumulated error of the graph can be reduced. These new constraints appear when the robot visits places previously visited, which is known as loop closure detection. By keeping the history of the past frames, it is possible to check if the current frame matches some of the previous ones. If two frames are similar enough, a transformation can be computed between these frames and a new constraint can be inserted from it, as shown in Figure 3. Once the graph has been optimized, the nodes are corrected and the new estimations of the camera poses can be finally extracted.

Figure 3. Loop closure detection (a) shows the nodes linked by edges that represents transformations between them, (b) shows a observation that was seen in the past and can be useful to add a new link between nodes visited previously.



Source: (HÖGMAN, 2012).

Global optimization could be more beneficial in cases of large loop closures; revisiting known places on the map generates loop closing edges that reduce the accumulated error. Therefore, a good global optimization needs to obtain some loop closures to guarantee a globally consistent trajectory. However, the idea of revisiting places can be translated in the pose graph as the task of comparing the actual node against all the others. This task

is computationally costly and is currently considered a bottleneck.

1.1.1 Steps to build a full RGB-D Map

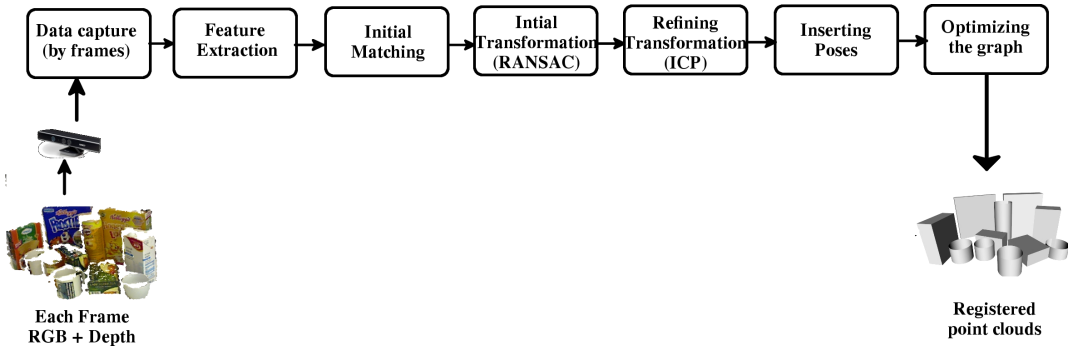
The previous section presented a brief introduction to the SLAM problem and an intuitive technique to address this problem via the so-called graph-based formulation. In brief, the frames captured with the help of sensors, such as RGB-D cameras, provide information on the environment. Each frame has a set of point clouds associated, which is useful to build partial models of the environment that are brought together to obtain a full model, known as global map. Ideally, a full model is reduced to achieve the matching points between two consecutive frames, which could be much easier if the match was perfect . In fact, a perfect match is almost impossible, considering that the quality of the information acquired is drastically reduced by the inherent noise associated with the sensors. For this reason, it is necessary to apply some steps to obtain a full map of the environment surrounding the robot. This idea is depicted in Figure 4. In the first step, an algorithm to obtain sparse interest points and their respective features, is applied on each frame and then these feature points are matched on two consecutive frames in the second step. The information acquired is the input in the third step to the RANSAC algorithm that calculates an initial transformation T_0 between two frames; then, an algorithm for refinement, e.g., the ICP algorithm is applied to the fourth step to refine T_0; in the last step, all the transformations and pose information are placed on a pose graph which can be optimized by using a framework to compute a globally consistent map.

It is not difficult to guess that a good initial transformation, T_0, plays a crucial role in avoiding wrong alignment in step three (ICP) and in decreasing the computational cost in step four (Global Optimization). This initial transformation can be achieved by computing the alignment between pairs of consecutive frames, which is known as pairwise registration .

1.1.2 Pairwise registration

Pairwise registration of multiple scenes in indoor environments is a fundamental problem not only in SLAM, but also in other areas such as semantic segmentation (RUSSELL et al., 2009; GUPTA et al., 2015; SHAO et al., 2012), object reconstruction (GEIGER; WANG, 2015; FIRMAN et al., 2013), and robust tracking (BYLOW et al., 2013; TYKKÄLÄ et al., 2014). In general, this type of registration requires attaching a set of consecutive frames taken from different views to a single coordinate system, which

Figure 4. Block diagram showing all the necessary steps to build a full RGB-D Map.



Source: Author.

is a difficult, yet extensively studied problem.

The advent of affordable consumer RGB-D cameras, which provide both RGB color and depth information, has renewed the interest in improving or creating new registration algorithms for indoor environments. Most of these algorithms have focused on the RGB color information (ENDRES et al., 2012; HENRY et al., 2012), whereas those focused on 3D point cloud information alone (HOLZ et al., 2015; PRAKHYA; QAYYUM, 2015) emerged more recently. Regardless of the type of information available for the registration task, algorithms that use sparse data, i.e., that select a reduced set of interest points are more efficient and require less memory than those using dense methods that use all the RGB-D information available (LOWE, 2004; HENRY et al., 2012).

The feature-based pairwise registration process is usually performed in the following way. Each feature in a frame is compared with features of the other frame to detect correspondences between their keypoints, and the best match is selected. Usually the Approximate Nearest Neighbor (ANN) algorithm (MUJA; LOWE, 2014) is used to reduce the search space for matches. However, despite the fast results, the resulting matches may contain many outliers. Then, to remove such outliers and to find an adequate initial spatial transformation that matches the two frames, the RANSAC algorithm is used.

Algorithms based on RGB color image detect 2D local features from a set of interest points for each pair of consecutive frames. These features are subsequently matched and then projected to 3D using the depth information. The most significant weakness associated with this type of algorithms is that sparse 2D features can become undetectable in areas of low visual texture or insufficient illumination, a situation in which it is not possible to find a transformation between frames, consequently leading to accuracy loss in the registration task. This weakness can be adequately addressed using 3D local features obtained directly from 3D point clouds that give us depth information. These features

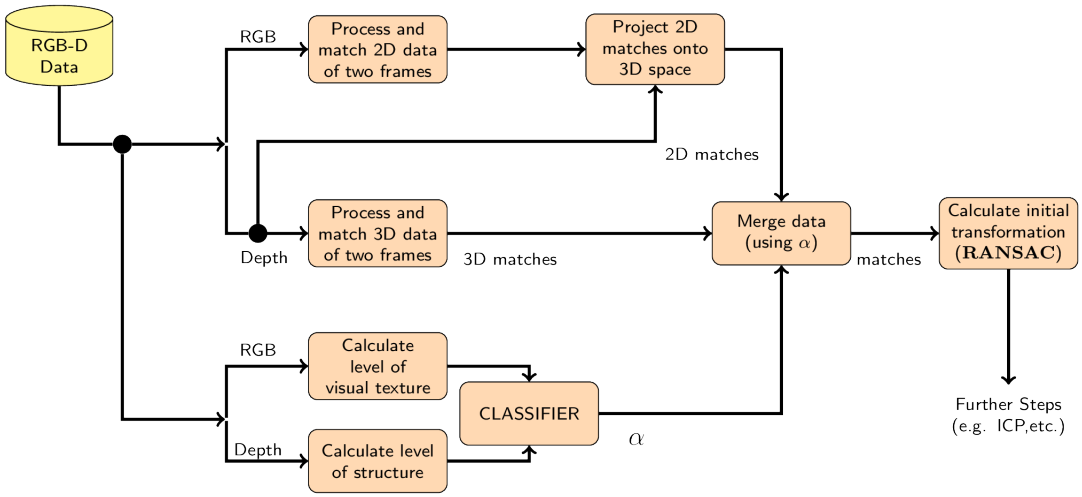
describe geometrical information around each interest point and are immune to variations in illumination. Hence, they are expected to have a better performance in structured environments containing little or no visual texture. However, in scenes with low structure level, it is difficult to find good 3D features.

2D and 3D features have been separately used in different algorithms so far, but, as noted above, each of these types of feature may become undetectable, according to the visual texture and scene structure present in the environment surrounding the robot. Thus, techniques that combine RGB color with depth information, according to the visual texture and scene structure of the data, appear very promising.

1.2 Objective

This thesis presents a proposal and further development of a framework combining 2D and 3D features to improve the estimation of the initial transformation between pairs of frames. Furthermore, since definitions and criteria on how to combine 2D and 3D features continued to be an open issue, the framework proposed also provides an innovative system to calculate a parameter (α) based on the structure and visual texture information. This parameter allows combining 2D and 3D features in an automatic way. Figure 5 briefly describes the operation of the framework proposed.

Figure 5. Overview of the framework proposed.



Source: Author.

We start by acquiring the RGB and depth images for each frame. Then, RGB image pairs are used to obtain 2D features. Then, these features are matched. Since each match has an associated distance-based error metric, all 2D matches are sorted in an

ascending order of distance. All sorted 2D matches are then projected into 3D using the corresponding depth information

A similar procedure is used to obtain 3D features directly from the depth images and then to match them; here, the projection step is no longer needed.

Then, matches from RGB and depth are selected and merged using the α parameter, aiming to create an unified vector of 3D matches. The α parameter indicates the proportion of the use of each type of information. If the scene has little structure and a lot of texture, the ratio of RGB matches used in the unified vector of 3D matches is greater than depth matches; if the scene has plenty of structure and little texture, depth matches are used in greater proportion.

Finally, the RANSAC algorithm is used to find the initial transformation between frames.

In summary, the thesis main goal is to present a new framework for pairwise registration automatically combining 2D and 3D features based on the current texture and structure level of the scene. From this goal, a secondary objective emerges, which is to carry out a comprehensive study and a further selection of both 2D and 3D detectors and descriptors used in the proposed framework.

1.3 Contributions

Previous researches have used 2D or 3D features in the pairwise alignment technique to align consecutive pairs of frames. We here propose a new framework to improve the initial transformation in the pairwise alignment for indoor scenes using the combination of 2D and 3D features and, including scenes in which low or no level of structure or visual texture exists. Throughout the thesis we carefully explain the selection of each algorithm or method used in our proposal, presenting a detailed discussion about selecting the best 2D keypoint detector and descriptor combination, as well as the selection of the best 3D detector. We also propose a full methodology for defining the parameter value that automatically balances the combination of the 2D and 3D features.

Part of the contributions of the thesis are 4 publications in international conference and symposiums, and once article submitted to a journal. These publications are:

- **Simulation platform for cooperative vehicle systems**, published in 17th International IEEE Conference on Intelligent Transportation Systems (ITSC 2014).

- **Adaptive Selection of Color Images or Depth to Align RGB-D Point Clouds**, published in the Joint Conference on Robotics and Intelligent Systems (JCRIS 2014).
- **A Simulation Framework for Multi-Vehicle Communication**, and **Aligning RGB-D Point Clouds through Adaptive Integration of Color and Depth Cues**, published in XII LARS Latin American Robotics Symposium (LARS 2015).
- **Pairwise Registration in Indoor Environments using Adaptive Combination of 2D and 3D Cues**, submitted to the journal Image and Vision Compting.

Finally, we make available the software code used herein.

1.4 Outline

The remainder of this thesis is organized as follows. Chapter 2 presents the fundamental concepts and theories of the elements involved in the registration technique, which are applied to the later chapters. Chapter 3 presents previous work in the related literature and studies by other researchers in registration techniques. Chapter 4 describes our proposal in detail. Chapter 5 presents the evaluation criteria used to characterize the performance level of the different components of our framework, as well as the experiments carried out. Chapter 6 concludes this thesis with a critical analysis of our framework and a discussion of possible future work.

Chapter 2

Theoretical Background

This chapter provides a technical introduction to the topics more relevant to this thesis.

2.1 RGB-D Sensor

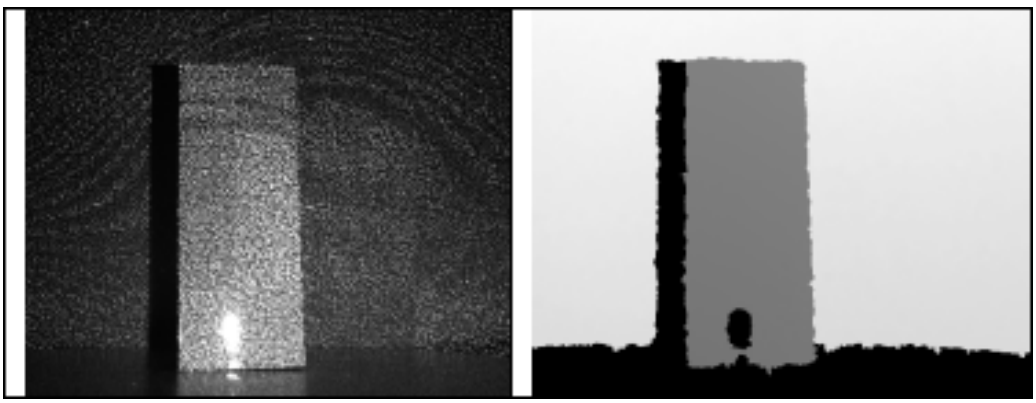
A RGB-D camera is a sensor that provides color image information as well as an estimated depth for each pixel. Despite the fact that this type of sensors already existed a long time ago, it was until the existence of the consumer RGB-D sensors, e.g. Microsoft Kinect and Axus Xtion Pro, that these have been considered by several researchers addressing well-known problems in computer vision such as SLAM, 3D mapping, object recognition, interactive 3D modeling, autonomous flight, etc.

RGB-D sensors captures depth and color images simultaneously at a frame rate about 30 Hz. For the measurement depth the sensor uses an infrared camera and an infrared laser emitter (See fig. 7).

//

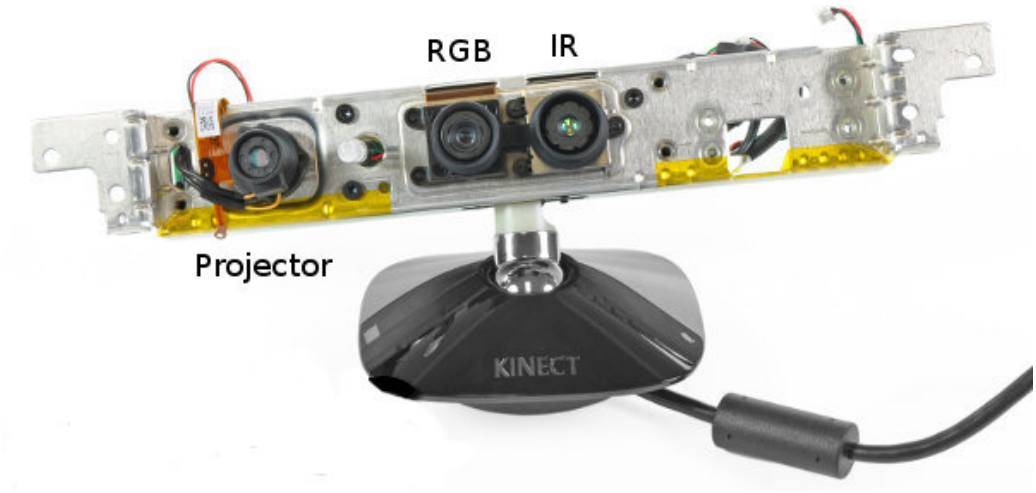
The laser source emits a single beam which is split in multiple beams creating a pattern of speckles projected onto the scene. The infrared camera captures the reflected pattern which is correlated with a reference pattern. Each speckle projected on an object whose distance to the sensor is larger or smaller than the distance between the sensor and the reference plane generates a shift between the laser projector and the infrared camera. Each shift is measured by a simple correlation of triangles producing the depth-disparity image as shown in fig. 6.

Figure 6. Infrared image with the pattern of speakles projected on the object (Left), and the resulting depth image (Right)



Source: Reproduced from (KHOSHELHAM, 2012).

Figure 7. Microsoft Kinect sensor with the cover taken off, showing the infrared laser emitter, the RGB camera, and, the infrared camera.



Source: Reproduced from ROS [1]

An example for computing the disparity is shown in fig 8. This example assumes that the object is located in the reference plane at a distance Z_o to the sensor and a speckle k on the object is captured on the image plane. Since the speckle is shifted closer to the sensor, the speckle location will be displaced on the image plane. The distance of the speckle Z_k in object space is computed as,

$$Z_k = \frac{Z_o}{1 + \frac{Z_o}{fb}d}, \tag{2.1}$$

where f is the focal length of the infrared camera, b is the base length, and, d is the

[1] ⟨http://wiki.ros.org/kinect_calibration/technical⟩, [Accessed: Aug. 7, 2016].

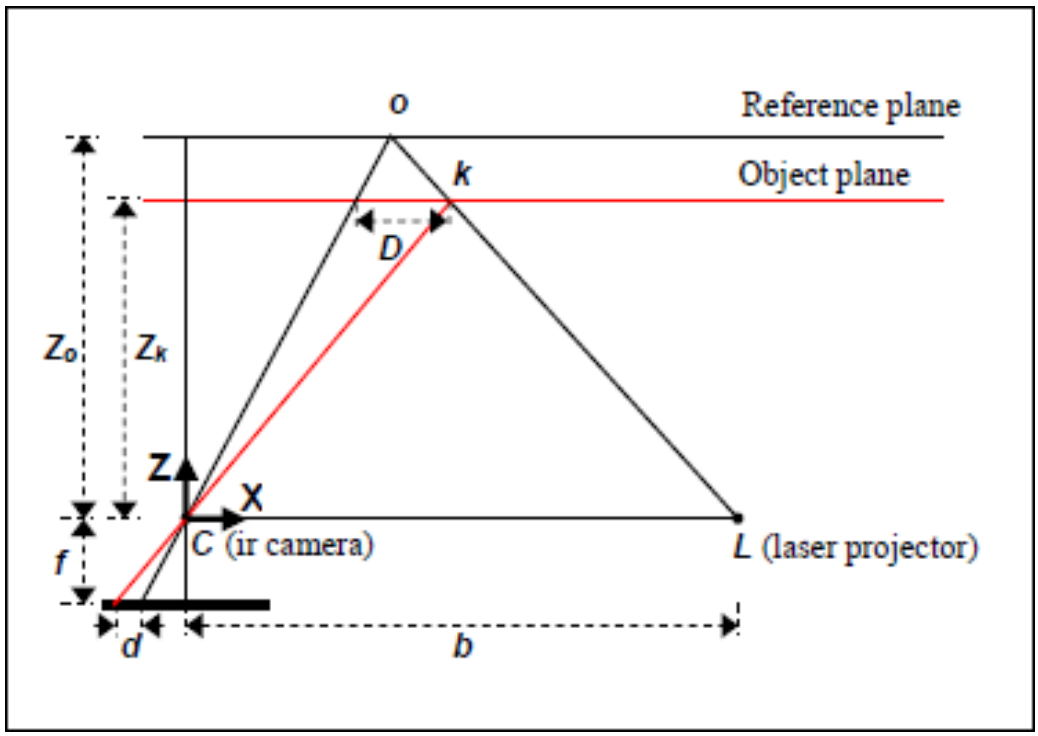
observed disparity in image space, which can be obtained from the similarity of triangles,

$$\frac{D}{b} = \frac{Z_o - Z_k}{Z_o}, \tag{2.2}$$

$$\frac{d}{f} = \frac{D}{Z_k}, \tag{2.3}$$

where D is the displacement of the speckle k in object space.

Figure 8. Example for computing the depth of a speckle by using similarity between triangles



Source: Reproduced from (KHOSHELHAM, 2012).

2.2 Detectors and descriptors

Visual perception is the dominant sense in humans. This sense enables humans to understand and to learn about the world around them. Computer vision tries to imitate this behavior in the best possible way by the use of sensors that capture images of the real world. However, the analysis of images is a complex task due to the different appearances that can represent a same object in different images depending on illumination, shadows and occlusions, among others. Since redundant information presented by the image pixels and the high computational cost achieved if the whole image is used, computer vision presents a technique using a minimal set of local features of the image that allows to efficiently represent it and encode it. These features can refer to specific locations in the

image which are known as interest points or keypoints.

Then, the feature detectors are methods used to locate interest points from images, whereas, the feature descriptors are a set of local measurements capturing essential and no redundant information about the pixels surrounding interest points.

A brief explanation about the functioning of the detectors and descriptors used in this research is provided as follows.

2.2.1 2D local detectors and descriptors

Five different approaches to extract features from 2D images are described below.

2.2.1.1 SIFT (LOWE, 1999)

Scale-Invariant Feature Transform (SIFT) is one of the best known 2D local detector and descriptor algorithm which is not only invariant to scale but also invariant to rotation, illumination and small point of view shifts.

Algorithm 1 SIFT Algorithm

Require:
u: 2D image
Ensure: $feat\_vect$: Feature descriptor vector
Initialize:
$keypoint\_list \leftarrow 0$
$feat\_vect \leftarrow 0$
$v \leftarrow$ Compute the Gaussian scale-space with n scales and m octaves
for all Octaves **do**
  for all Scales **do**
    $DoG \leftarrow$ Compute the Difference of Gaussians
    $kp \leftarrow$ Detect interest points (position and scale)
    $keypoint\_list \leftarrow$ Add kp to list
  end for
end for
$keypoint\_list \leftarrow$ Filter unstable keypoints
for every kp in $keypoint\_list$ **do**
  $kp \leftarrow$ Assign a reference orientation
  $kp \leftarrow$ Build the keypoints descriptor
  $feat\_vect \leftarrow$ Add feature keypoint to feature descriptor vector
end for
return $feat\_vect$

The algorithm is described in 1 and starts constructing a Gaussian scale space representation of the original image to ensure the invariance to scale. This scale space is

a set of images obtained by progressively blurring (scale) and resizing (octave) the original image. Then, the original image is considered the first octave; the second octave is achieved by resizing the original image to half size and so on. In the original article of SIFT, Lowe (1999) suggests using 4 octave levels. On the other hand, each octave is blurred through the convolution of a Gaussian operator and an image (See Eq. 2.4). The blur levels suggested by Lowe are 5.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{2.4}$$

where L is the blurred image, I is the image being blurred and $G(.)$ is a variable-scale Gaussian,

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.5}$$

To locate stable edges and corners, the second-order derivatives are calculated on blur images. However, this operation is computationally costly; therefore, an efficient approximation by convolving the Difference-of-Gaussian (DoG) function, $D(x, y, \sigma)$ with the image can be used (See Eq. 2.6).

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y), \tag{2.6}$$

where k is a multiplicative factor separating two nearby scales.

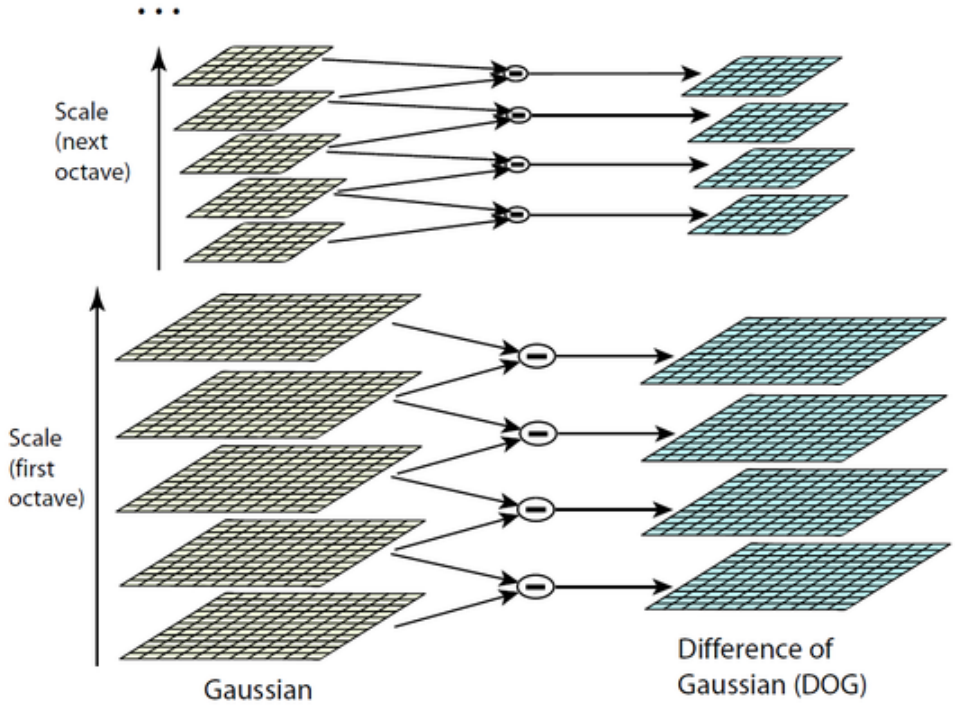
Figure 9 shows the process to convert Gaussians in DoG images by subtracting adjacent image scales.

The second step in the SIFT algorithm lies in finding the maxima and the minima of $D(x, y, \sigma)$ for each sampled point comparing to its eight neighbors in the current image and nine neighbors in the scale above and below as shown in Fig. 10, where the sampled point "X" is considered a keypoint if this is the greatest or smallest of all its 26 neighbors. Because maximal and minima rarely lie exactly on a pixel, a mathematical subpixel location can be found by using the Taylor expansion of the DoG image closer to a different sample point,

$$D(x) = D + \frac{\partial D^T}{\partial \mathbf{x}}\mathbf{x} + \frac{1}{2}\mathbf{x}^T \frac{\partial^2 D}{\partial \mathbf{x}^2}\mathbf{x}, \tag{2.7}$$

where D and its derivatives are evaluated at the sample point and $\mathbf{x} = (x, y, \sigma)^T$ is the offset from this point.

Figure 9. Adjacent Gaussian images on the left are substracted to produce the DoG images on the right.



Source: Reproduced from (LOWE, 2004).

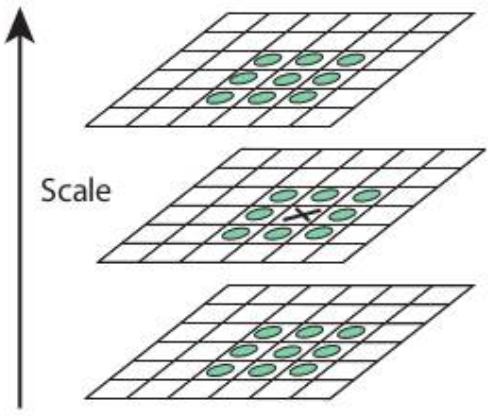
Keypoints that lie along edges or flat regions, or have low contrast are removed in the third step. To define whether a subpixel (maxima or minima) must be removed, Eq. 2.7 is again used. If the intensity magnitude is less than a threshold value previously defined, the keypoint is removed. On the other hand, to remove keypoints along edges or flat regions, the Hessian matrix, **H**, is computed at the location and scale of the keypoint,

$$\mathbf{H} = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix} \tag{2.8}$$

Eigenvalues λ_1 and λ_2 of **H** are proportional to the principal curvatures of D. As shown in Fig. 11, there are three different relations between the eigenvalues which determine the type of region for each keypoint: if λ_1 and λ_2 are small, then this is a flat region. When $\lambda_1 \gg \lambda_2$ or $\lambda_2 \gg \lambda_1$, this is an edege region, and if $\lambda_1 \approx \lambda_2 \gg 0$, this is a corner region. Then, keypoints that lie in the flat or edge regions are removed, leaving only keypoints that lie in corner regions as good keypoints.

In the fourth step, the keypoints are fully described by a geometric frame of four parameters: the keypoint center coordinates x and y, its scale and its orientation (see Fig. 12). To calculate the orientation, the magnitude and orientation for a small region

Figure 10. Each point in DoG is compared with its 8 neighbors in the current scale and 18 more in the superior and inferior scale. If this point is the greatest or least of all its neighborhood, then, it is considered a keypoint.



Source: Reproduced from (LOWE, 2004).

of radio R around the keypoint are calculated using Eq. 2.9 and Eq. 2.10. The magnitudes associated with the orientations for each 10 degrees are accumulated to form the bins for a 36-bin histogram ranging between 0 to 360 degrees. Then the peak in this histogram is used to normalize the other bins. Any peaks above 80% are converted into a new keypoint with the same location but with the orientation given by the bin.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \tag{2.9}$$

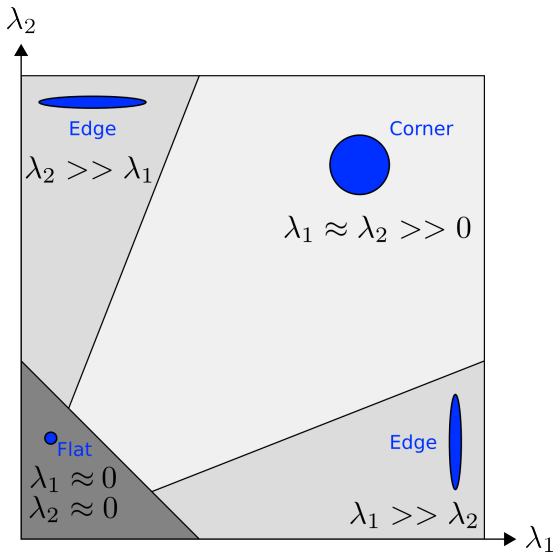
$$\theta(x,y) = \tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \tag{2.10}$$

Finally, in the fifth step, a 128-dimensional feature vector describing any keypoint is found. To obtain this feature vector for each keypoint, first, a 16×16 window subdivided into sixteen 4×4 windows is selected around the keypoint as shown in Fig. 13. Within each 4×4 window, gradient magnitudes and orientations are calculated and put into an 8-bin histogram. To lend more weight to windows closest to the keypoint a Gaussian function is multiplied to each 8 bin histograms. Once all orientation histogram entries have been completed, $4 \times 4 \times 8 = 128$ numbers are obtained and normalized to form the feature vector.

2.2.1.2 SURF

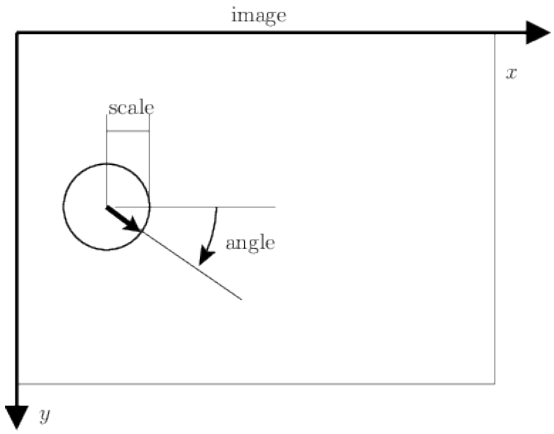
Speeded-Up Robust Feature (SURF) is a detector/descriptor proposed by Bay et al. (2006) as an efficient alternative to SIFT. The first change in this algorithm is the method

Figure 11. Different types of regions for a keypoint subject to eignevalues λ_1 and λ_2 of the Hessian matrix.



Source: Adapted from (HARRIS; STEPHENS, 1988).

Figure 12. Each keypoint is fully described by its scale, orientation and location in a 2D plane.



Source: Author.

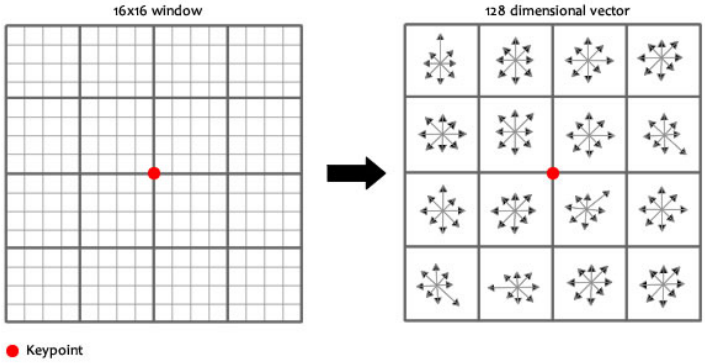
used to approximate the Laplacian-of-Gaussian (LoG) with Dog.

They use 2D box filters (Haar wavelet) as an approximation of the second-order Gaussian derivatives as shown in Fig. 14.

In the convolution process with these box filters, an integral image is used, which is more efficient that creating octaves in the original image; in other words, whereas with SIFT the scale space is analyzed by iteratively reducing the image size, SURF up-scales the box filter size (See Fig. 15).

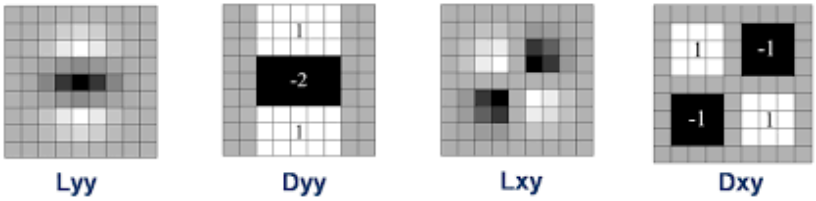
To describe each keypoint, SURF uses a squared region centered around the keypoint and split it into smaller 4×4 regions. Instead of using gradient orientation histograms as SIFT, SURF computes wavelet responses in the horizontal d_x and the vertical d_y direction

Figure 13. Left figure shows the divisions of the region surrounding a keypoint and the right figure shows the gradient magnitudes and orientations within each 4×4 window, which are used to form the 128-feature descriptor.



Source: Reproduced from (LOWE, 2004).

Figure 14. Examples of second order Gaussian derivatives (L_{yy} and Lxy), and their aproximations using 2D box filters (D_{yy} and D_{xy}).



Source: Reproduced from (BAY et al., 2006).

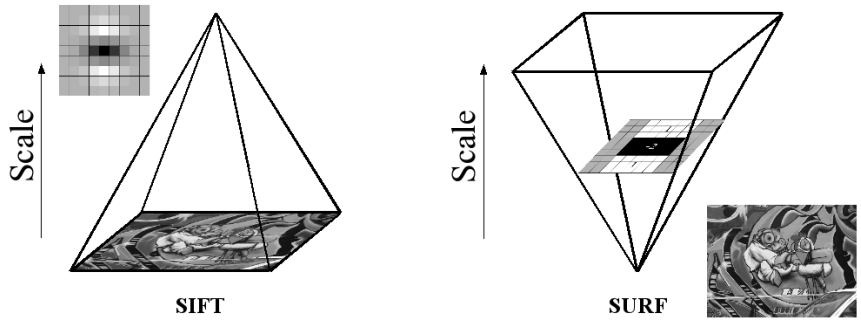
for each region, summed up over each region and extract a four-dimensional descriptor vector $\mathbf{v} = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|)$. Once all these vectors have been completed, $4 \times 4 \times 4 = 64$ numbers are obtained and normalized to form the feature vector.

2.2.1.3 ORB (RUBLEE et al., 2011)

Oriented FAST and Rotated BRIEF (ORB) uses an improved FAST as detector and BRIEF as descriptor. Features from Accelerated Segment Test (FAST) detector algorithm (ROSTEN et al., 2010) was developed as an computational efficient alternative for those using LoG, e.g. SIFT, and SURF. FAST creates a circle of sixteen pixels around each corner candidate and calculates the intensity difference between them. The candidate is considered a corner if a set of n contiguous pixels in the circle are all brighter or all darker (See Fig. **??**).

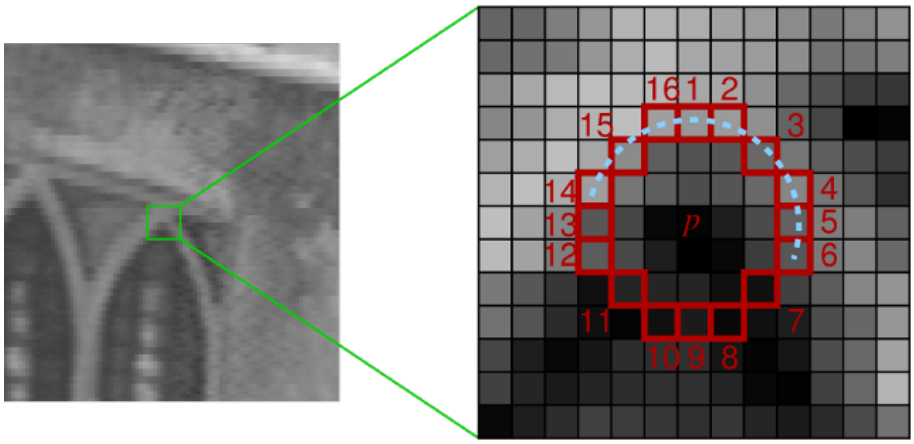
Since FAST produces neither a measure of corneness nor multi-scale features, ORB improves it by using a Harris corner measure and a scale pyramid of the image, respectively. Furthermore, FAST does not include an orientation operator; ORB uses the vector

Figure 15. This figure shows how instead of iteratively reducing the image size (SIFT), the use of integral images allows the up-scaling of the filter at constant cost (SURF).



Source: Reproduced from (BAY et al., 2006).

Figure 16. The left figure shows an image with a possible candidate corner. The right figure shows a magnified image of the region surrounding the pixel p tha is the centre of the candidate corner. In this example, sixteen pixels around p are highlighted and the dashed line passing through $n = 9$ contiguous pixels which are brighter than p indicate that the candidate indeed is a corner.



Source: Reproduced from (ROSTEN et al., 2010).

from the corner center O to the centroid C, $\vec{OC}$, and calculates the orientation Θ of the patch (See Fig. **??**). Let the moments of a patch can be calculated as,

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y), \tag{2.11}$$

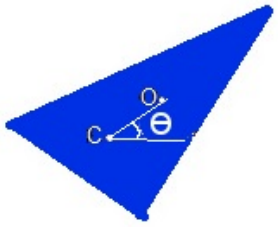
and centroid C,

$$C = (\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}), \tag{2.12}$$

The orientation patch is simply:

$$\Theta = atan2(m_{01}, m_{10}). \tag{2.13}$$

Figure 17. The corner orientation is calculated by using the orientation of the line segment formed by the center corner O and its centroid C.



Source: Author.

In order to describe these keypoints, ORB uses the Binary Robust Independent Elementary Features (BRIEF) descriptor (CALONDER et al., 2012), by building a vector of 256 binary intensity comparisons between sample points selected randomly from a patch image previously smoothed and centered at the keypoint location. Finally, BRIEF is steered according to orientation Θ of the keypoint.

2.2.1.4 LBP (OJALA et al., 1994)

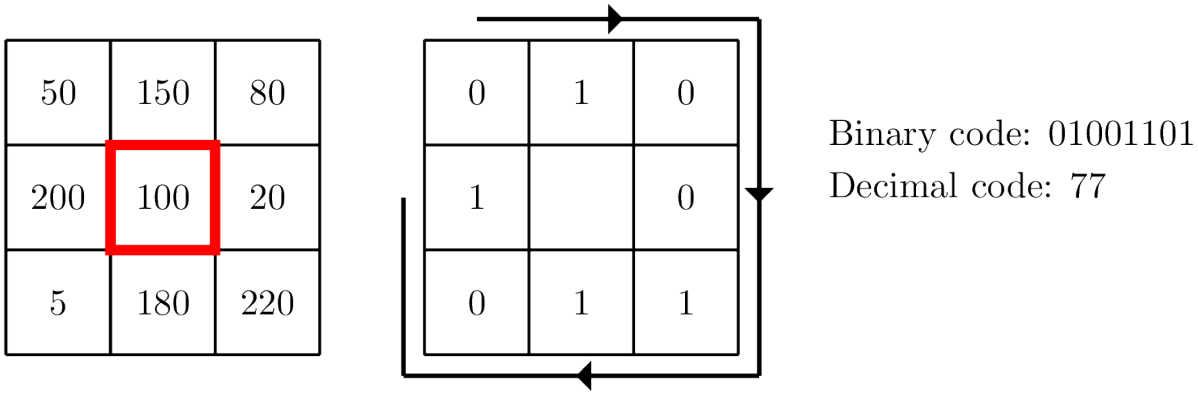
Local binary Pattern (LBP) is a visual descriptor that captures and encodes the local texture of an image by subtracting the value of each pixel from the value of its neighbors. In the example in Fig. 18-left, the highlighted pixel is encoded using LBP and, therefore, its value is subtracted from each neighbor value. When this difference is negative, the neighbor is encoded with 0, otherwise it is encoded with 1. This binary representation of the neighborhood is used to obtain an 8-digit binary code by concatenating each neighbor, starting from the upper left neighbor. For convenience, this binary number is converted into decimal number from 0 to 255 (see Fig. 18-right). This procedure is repeated for each pixel in the image.

Finally, a histogram is defined with all the codes of an image, generating a 256-dimensional vector representing the level of texture of the image.

2.2.1.5 CLBP

Curvature Local Binary Pattern (CLBP) is an extension of LBP proposed by Chun et al. (2013). The CLBP algorithm is the same as that for creating the texture vector, the only change is that it uses the curvature of the depth map instead of the pixel intensity. Curvature (k) is the magnitude of the second derivative of the curve at a given point:

Figure 18. Example of LBP. Left: LBP for a pixel (red) and its neighborhood. Right: The generated LBP code.



Source: Author.

$$k = \left| \frac{d^2 y}{dx^2} \right|, \tag{2.14}$$

where (x, y) is the location in the map pixel space.

Finally, a histogram is defined with all the codes of the depth map, generating a 256-dimensional vector representing the level of structure of the image.

2.2.2 3D local detectors and descriptors

In this, Normal Aligned Radial Feature(NARF), Intrinsic Shape signatures 3D (ISS3D) and HARRIS3D are presented as interesting approaches to detect 3D keypoints in point cloud; a Fast Point Feature Histogram (FPFH) is presented as an excellent 3D feature descriptor.

2.2.2.1 NARF (STEDER; KONOLIGE, 2010)

A Normal Aligned Radial Feature (NARF) detector, begins with the transformation of the point cloud into a range image used to find borders; then, for each point p, all the neighbor points that lie in the square patch of size s around p are sorted in increasing order by their 3D distances to p. Point p is marked as a probable border if any score computed in its four directions (right, bottom, left, top) with the Eq. 2.15 is greater than a specified threshold.

$$s_i = \max\left(0, 1 - \frac{d_M}{d_i}\right), \tag{2.15}$$

where d_i is the average distance from p and its next neighbors in direction $i \in \{right, bottom, left, top\}$ and d_M is the mean distance for the set of points into the square of size s and computed as $d_M = (0.5.(s+1))^2$.

Any probable border is considered an interest point if it can be robustly detected in the same place even if observed from different perspectives. Therefore, the probable border p and all its neighboring points $n_0, ..., n_k$ within a radius σ, which do not have a border in between, are used to compute an interest score $I(p)$ using the following equations,

$$I(p) = I_1(p).I_2(p) \tag{2.16}$$

$$I_1(p) = \min_i \left(1 - w_{ni} \max\left(1 - \frac{10.\|p - n_i\|}{\sigma}, 0\right)\right) \tag{2.17}$$

$$I_2(p) = \max_{i,j} \left(f(n_i).f(n_j)\left(1 - |\cos\left(\alpha_{ni} - \alpha_{nj}\right)|\right)\right) \tag{2.18}$$

$$f(n) = \sqrt{w_n \left(1 - \left|\frac{2.\|p - n_i\|}{\sigma} - 0.5\right|\right)}, \tag{2.19}$$

where α is the orientation of both p and all its neighboring points that do not have a border; and w_n is a weight that is 1 for every border point and $1-(1-\lambda)^3$ for every other point. λ represents the magnitude of the curvature associated with every non-border point.

Therefore, a probable border p with an interest score $I(p)$ value higher than a user specified threshold is marked as a keypoint of the point cloud.

2.2.2.2 ISS3D (YU, 2009)

Intrinsic Shape Signature for 3D (ISS3D) is a method that obtains a set of salient basis points through the use of the analysis of the eigenvalues of a 3×3 covariance matrix, which are computed for each point and its neighbors within a spherical region with radius $r_{density}$. Here, a point p_i has associated three eigenvalues $\lambda_1, \lambda_2, \lambda_3$, that are used to evaluate on whether or not p_i is considered a salient point. Then, p_i must fulfil the next two ratios between the eigenvalues:

$$\lambda_2/\lambda_1 < \tau_{21}, \tag{2.20}$$

$$\lambda_3/\lambda_2 < \tau_{32}, \tag{2.21}$$

where τ_{21} and τ_{32} are threshold values. The final result is a reduced point cloud whose points are used as keypoints.

2.2.2.3 HARRIS3D (RUSU et al., 2008)

HARRIS3D is based on the HARRIS 2D detector, a method that uses the gradient of the image in the vertical and horizontal directions to detect corners. Since corners represent a change of intensity for a shift $[u, v]$, this change is computed as:

$$E(u, v) = \sum_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2, \tag{2.22}$$

where $w(x, y)$ is usually a window Gaussian function at position $x, y)$ and I is the intensity.

Since this computation at every pixel is slow, $E(u, v)$ can be approximate using the Taylor series:

$$E(u, v) \approx [u, v]\mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix}, \tag{2.23}$$

where $\mathbf{M}$ is a 2×2 matrix computed from the image derivatives,

$$\mathbf{M} = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \sum \begin{bmatrix} I_x \\ I_y \end{bmatrix} \begin{bmatrix} I_x & I_y \end{bmatrix} = \sum \nabla I (\nabla I)^T, \tag{2.24}$$

The measure of corner response R is:

$$R = det\mathbf{M} - k(trace\mathbf{M})^2, \tag{2.25}$$

where $det\mathbf{M} = \lambda_1\lambda_2$ and $trace\mathbf{M} = \lambda_1 + \lambda_2$, and k is an empirical constant.

Since a corner should have a large intensity change in all directions, R should be greater than certain threshold.

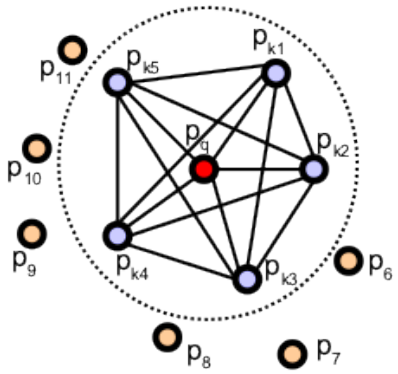
For the 3D case, Gedikli replaces the gradients in Harris matrix by surface normals N_I and uses the same responses R extending it for 3D.

$$\mathbf{M} = \sum N_I (N_I)^T, \tag{2.26}$$

2.2.2.4 FPFH (RUSU et al., 2009)

Fast Point Feature Histograms (FPFH) is a 3D descriptor, and is a variant that reduces the computational time used by Point Feature Histograms (PFH). For each keypoint p_q, PFH draws a sphere with radius r containing k neighbors, which are fully interconnected as shown in Fig. 19, where the keypoint and its $k = 5$ neighbors are highlighted in the area enclosed by the circumference.

Figure 19. PFH Diagram



Source: Reproduced from (RUSU et al., 2008).

To find the descriptor for each pair of points p_s and p_t in this sphere, first a fixed coordinate frame **uvw** (see Fig. 20),is defined as:

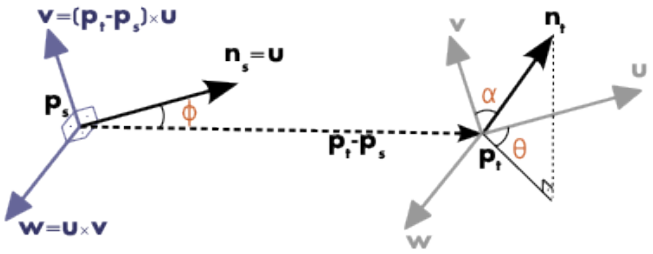
$$\mathbf{u} = \mathbf{n}_s, \tag{2.27}$$

$$\mathbf{v} = \mathbf{u} \times \frac{p_t - p_s}{\|p_t - p_s\|_2}, \tag{2.28}$$

$$\mathbf{w} = \mathbf{u} \times \mathbf{v}, \tag{2.29}$$

where $\mathbf{n}_s$ is the normal associated with point P_s and $\|p_t - p_s\|_2$ is the Euclidean distance d, between p_s and p_t.

Figure 20. PFH Coordinates



Source: Reproduced from (RUSU et al., 2008).

Then, a quadruplet formed by three angular features and the Euclidean distance, $\langle \alpha, \phi, \theta, d \rangle$, between the two points is computed as:

$$\alpha = \mathbf{v}.\mathbf{n}_t, \tag{2.30}$$

where $\mathbf{n}_t$ is the normal associated with the point P_t

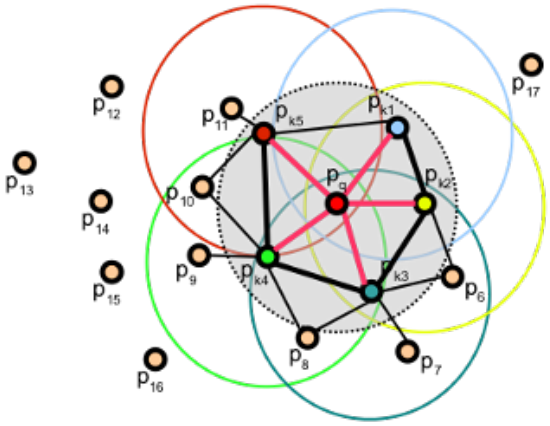
$$\phi = \mathbf{u}.\frac{p_t - p_s}{d}, \tag{2.31}$$

$$\theta = arctan(\mathbf{w}.\mathbf{n}_t, \mathbf{u}.\mathbf{n}_t). \tag{2.32}$$

These quadruplets are normalized and binned into a histogram to form the descriptor for a given keypoint. However the computational complexity of PFH is $O(nk^2)$ for n keypoints with k neighbors, is reduced to $O(nk)$ using FPFH.

FPFH does not include the Euclidean distance d, and the tuple $\langle \alpha, \phi, \theta \rangle$ is computed only for the keypoint, p_q, and its neighbors in a process called Simplified Point Feature Histogram (SPFH) as shown in fig. 21.

Figure 21. FPFH Diagram



Source: Reproduced from (RUSU et al., 2009).

To counteract the loss of accuracy, for each k neighbor of p_q, the SPFH is also computed and its values are used to weight the final histogram:

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{w_k}.SPFH(p_k), \tag{2.33}$$

where weight w_k represents a distance between p_q and its neighbor p_k.

Finally, one histogram using 11 binning divisions is used for each angular feature and

then concatenated to form a 33-dimensional feature vector.

2.3 Matching

Once the features have been computed on the whole image, the goal is to track them while the camera moves. This tacking can be done by associating the features between a couple of consecutive frames (pairwise).

Due to the movement of the camera and to the sensor noise, the descriptors are not exactly the same between two different frames; the best thus consists in finding the correct associations with a good belief.

Most of the algorithms work with different steps, first from a sparse level where the hypothesis is wide, to eliminate the most obvious mismatches at lower computational cost. The matches that fit the model are called the inliers, and the matches being discarded are called the outliers.

Matching can be found using a nearest neighbor search based on one distance metric, e.g. Euclidean distance computed on the descriptors of the features. There are several algorithms to implement this search; the most intuitive algorithm is Naive Nearest Neighbor, which scans all the features in the second frame (searching region) trying to find the lower distance of each of the features in the first frame. One computationally efficient alternative for such searching is the K-nearest neighbor (KNN) algorithm, which uses a binary tree that splits the searching region with hyperplanes that allow a faster search.

2.3.1 KNN

KNN is an algorithm used for classification and regression, which is non-parametric, i.e., that it does not make any assumptions about the data distribution, which is an important characteristic since most of the data derived from the real world does not obey assumptions, e.g., Gaussian distributions, linearly separable,etc.

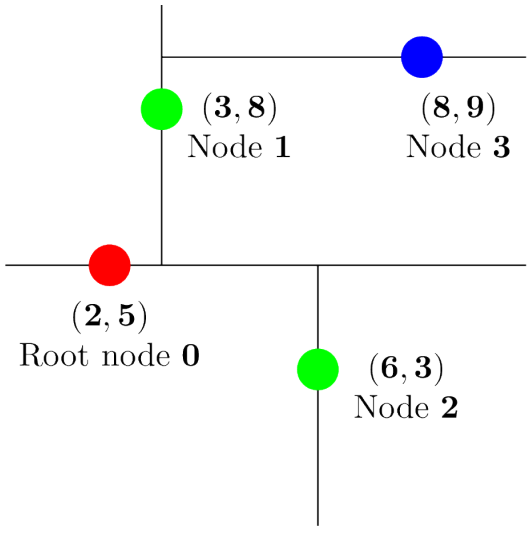
The input for KNN consists of the object to be classified and a predefined number k of training samples closest in distance to the object. The k number of neighbors is a value defined by the user, as well as the distance that can be any metric measure defined in the feature space of the object. The KNN output is the object classification assigned to the most common class among its k nearest neighbors.

2.3.2 K-Dimensional tree

The nearest neighbor search with the use of the naive neighbor search algorithm involves the brute-force computation of distances between all the pairs of points in a dataset. This computation can be very efficient in small datasets. However, this exhaustive search computing distances is inefficient when the number of samples grows, since this algorithm has time complexity $O(N)$, where N is the size of the dataset. This issue can be addressed by using tree-based data structures.

A K-Dimensional tree (K-D tree) is a geometrical data structure constructed to store points in D- dimensional space. A K-D tree is a binary tree where every non-leaf node divides the space into two parts called half-spaces. Points in the right half are represented with the right subtree and points in the left half are represented with the left subtree. Each subtree can be divided into two news subtrees and so on. A simple example illustrating how a two dimensional dataset $S = \{(2,5),(3,8),(6,3),(8,9)\}$ is represented using a K-D tree (See fig. 22), where, for instance, Root node in $(2,5)$ splits the plane in the y-axis forming two subspaces, whereas Node 1 in $(3,8)$ splits the top plane in the x-axis forming two new subspaces.

Figure 22. A simple 2-dimensional K-D tree

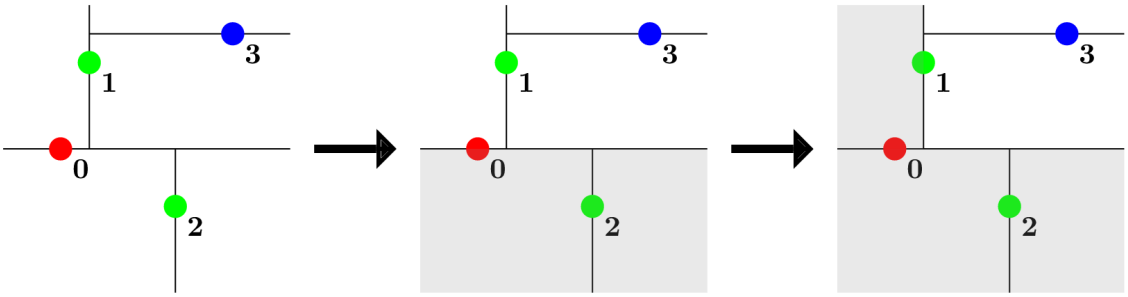


Source: Author.

This arrangement enables the improvement of the KNN search by discarding a subplane whenever a node is visited as shown in fig. 23. Let us suppose that the query point is Node 3. We begin in the root of this K-D tree and we consider whether the y-coordinate of Node 3 is less than or greater than the y-coordinate of Root node 0. Thus, we can ignore the bottom half-space, since Node 3 happens to be in the top half-space. Now we

repeat this procedure by comparing the x-coordinates of Node 1 and Node 3, and we can ignore the left half-space and look in the rigth.

Figure 23. Example about searching a query point in K-D tree. The grayed-out regions correspond to parts of the plane being discarded by the K-D tree searching process.



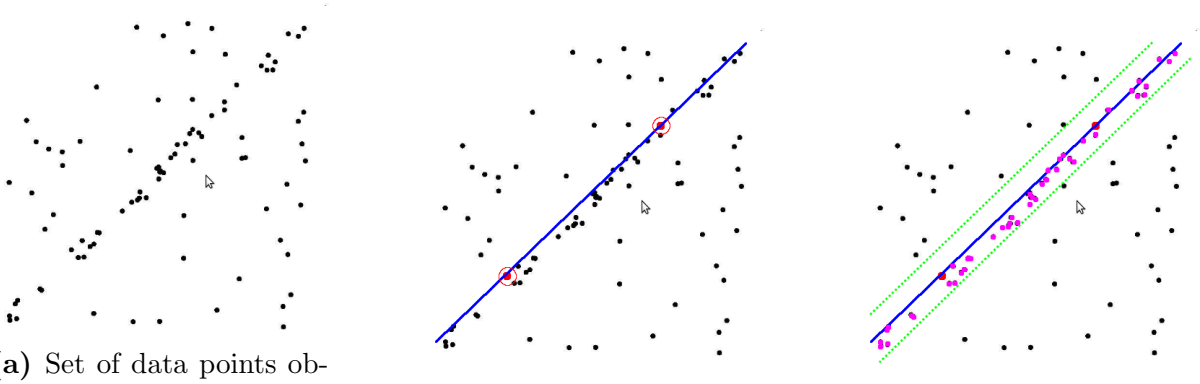
Source: Author.

2.4 RANSAC

Randon Sample Consensus (RANSAC) is a method proposed by Fischler and Bolles (1981), which is used to estimate parameters of a model M from a set of observed data. The basic approach of this method is composed of two steps that are iteratively repeated. In the first step, a subset of data points $S1$ is randomly selected from a set of data points P. This subset is composed for the minimum number of points required to instantiate the model M^* and the corresponding model parameters. The instantiated model M^* is used in the second step to determine the subset $S1^*$ of points in P that are within some error tolerance of M^*, where each point in $S1^*$ is known as an inlier and the whole set of these inliers is called consensus set. If $S1^*$ is less than some threshold t_h, a new subset $S2$ is randomly selected from P and the process is repeated again a fixed number of times.

A simple example is fitting of a line in two dimensions, as shown in Fig. 24. In the fig(a) is depicted a set of points observed. In fig (b) two points are randomly selected and a model is estimated. In fig (c) all other points are tested against this model, and, those points fitting the model according some error threshold represented here as dotted lines, are considered as part of the consensus set. , The major strength of RANSAC over other estimators lies in the fact that RANSAC works satisfactorily even with data contamined by large number of outliers, which are usually present in the matching process by using local feature detectors, because these detectors often make mistakes i.e., to identify a bad portion of an image as a feature.

Figure 24. A simple example is fitting of a line in two dimensions. Figure (a) shows the whole data set of points observed. Figure (B) shows the subset $S1$ composed by two points in this case and the estimated model (solid line). Figure (C) shows the model and the consensus set of points fitting the model with a threshold represented for the dotted lines.



(a) Set of data points observed P

(b) Estimated model M^*

(c) consensus set (inliers)

Source: Reproduced from (KHOSHELHAM, 2012).

Chapter 3

Related work

Related works focus not only on the proposed frameworks for pairwise registration, but also on the detectors and descriptors that determine the initial correspondence between the pair of frames.

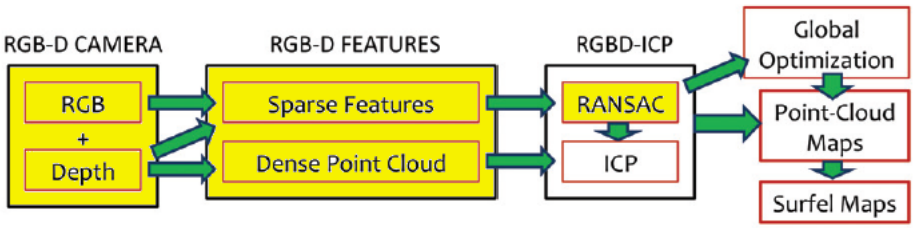
3.1 Frameworks using RGB-D data

Some architectures are built on the 2D features provided by the RGB images, then project the keypoints found for the three-dimensional space using the depth of information given by the RGB-D cameras (ENDRES et al., 2012; HENRY et al., 2012) . Other architectures such as Holz et al. (2015) propose to build on the 3D points and attach them at the end of the process, the RGB information of interest.

The approaches presented in Endres et al. (2012) and Henry et al. (2012) are the first ones to show the advantages of using consumer RGB-D cameras. They focused on visual Simultaneous Localization And Mapping (SLAM) for mobile robots and showed the advantages of exploiting depth information to improve the accuracy of the registration task.

Henry et al. (2012) match pairs of 2D local features extracted from two consecutive RGB frames, and then project them into 3D using depth information. After that, the RANSAC algorithm is used to determine a transformation that aligns the frames. With this transformation applied to the whole set of matched features, outliers are detected and discarded, and only the best matches are kept. The set of matches, the initial transformation given by RANSAC, and the two original point clouds are then used by ICP to refine the initial alignment by minimizing the transformation error.

Figure 25. SLAM framework proposed by Henry et al. (2012)



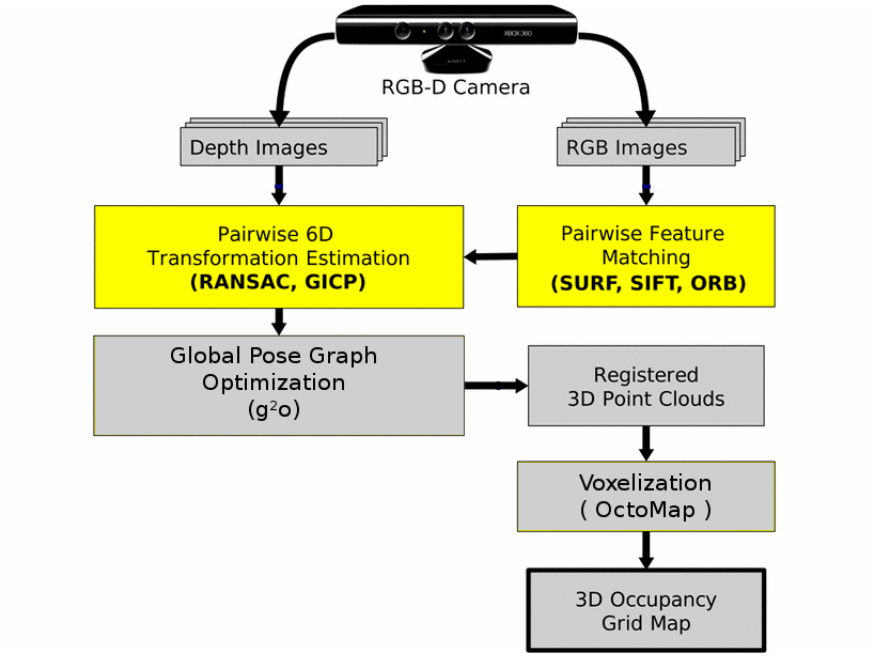
Source: Adapted from (HENRY et al., 2012).

Endres et al. (2012) use a similar procedure in their registration module. However, they do not use ICP and prefer to use only RANSAC exhaustively. They start by finding and matching 2D features between each new frame and the previous frame. They propose to find a first approximate matching using Fast Library Approximate Nearest Neighbor (FLANN), since it reduces the time required for frame-to-frame registration by a factor of two. From this set of matches they randomly select three matches and find a transformation that is applied to the full set, using the Euclidean distance error to determine the number of inliers and outliers. Then, the inliers are used to compute a refined transformation. These steps are iterated until the transformation with most inliers is found. Since they are not doing a local refinement of the initial transformation, the RANSAC process is repeated for matches between the new frame and the nineteen previous frames, trying to find new nodes (frames) and edges (transformation between frames) which are added to a pose graph used in a later step of the global refining.

In 2011, Rusu and Cousins (2011) present the Point Cloud Library (PCL), an open source library of algorithms for 2D and 3D image processing.

Holz et al. (2015) present a framework for 3D registration suggesting different methods incorporated into the PCL library. They propose using registration algorithms based on 3D features (instead of 2D features as in Endres et al. (2012) and Henry et al. (2012)) to calculate the initial alignment in cases in which the point clouds are not roughly aligned previously. They claim that the advantage in this case lies primarily in improving the 3D correspondences computed, and that this improvement will be reflected in the robustness of the 3D registration scheme. In the matching process, they propose the use of K-D trees as data structure for rapid searches using the FLANN library. The authors propose four methods to reject invalid matches: rejection based on the distance, rejection based on the median distance, rejection of pairs with duplicate target matches, and rejection based on RANSAC. The latter method is the most interesting, since it allows obtaining an initial

Figure 26. SLAM framework proposed by Endres et al. (2012)



Source: Adapted from (ENDRES et al., 2012).

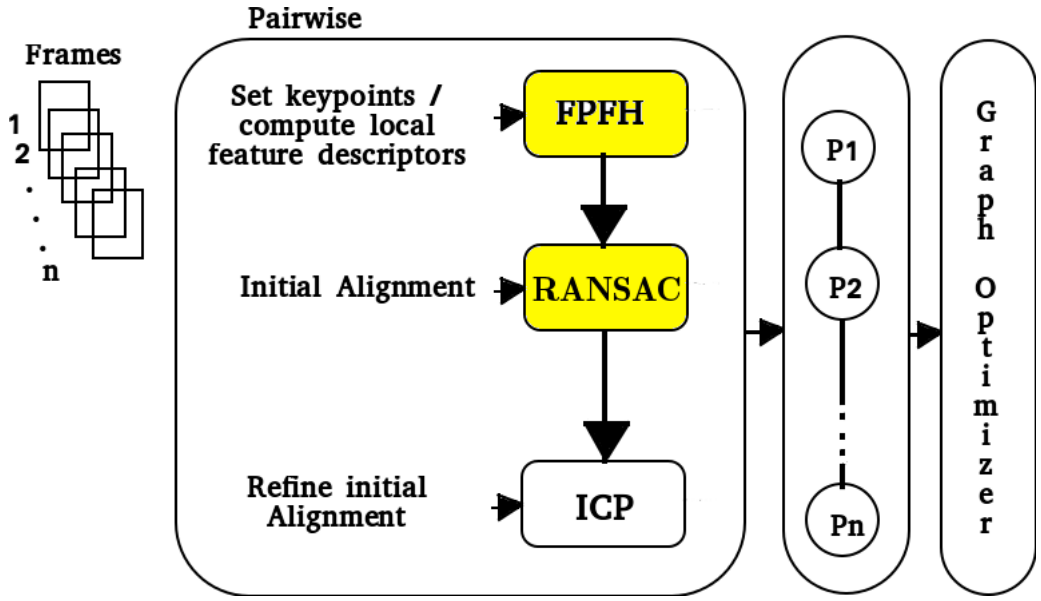
transformation while eliminating outlier matches. In the refining process, they use ICP with two metrics as alternatives: point-to-point error and point-to-plan error.

3.2 2D detectors and 2D descriptors

Different detectors and descriptors were used in these works. Henry et al. (2012) used SIFT as a 2D descriptor and 2D detector in a previous work, and then they used FAST (ROSTEN et al., 2010) detector along with the Calonder descriptor (CALONDER et al., 2008). They claim that this combination of detector and descriptor, FAST and Calonder, is not only faster than SIFT, but also provides more reliable visual keypoints. However, this claim is not supported by any explicit assessment in this work.

Endres et al. (2012) showed the influence of the chosen detector on the SLAM accuracy when using RGB-D cameras. They evaluated three different detectors, ORB, SURF, and SIFT, in nine different sequences. They found that the SURF and the SIFT detectors obtained similar results in accuracy, whereas ORB showed a significantly lower accuracy. However, experiments were only performed in sequences with a high level of visual texture and this could have affected the evaluation of the detectors. Thus, there is still a lack of a better assessment of 2D descriptors and 2D detectors in realistic scenes with low visual texture and variation in brightness.

Figure 27. SLAM framework proposed by Holz et al. (2015)



Source: Adapted from (HOLZ et al., 2015).

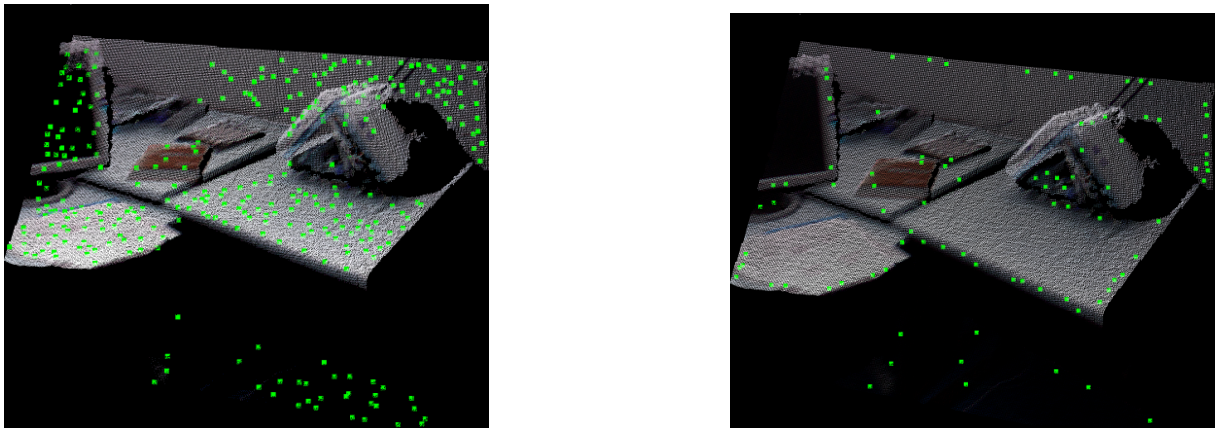
3.3 3D detectors and 3D descriptors

Similarly, Holz et al. (2015) suggest using some 3D descriptors and 3D detectors without conducting any assessment in this regard. However, there are two comprehensive studies (FILIPE; ALEXANDRE, 2014; PRAKHYA; QAYYUM, 2015) on the selection of the best 3D local detector.

Filipe and Alexandre (2014) compare eight different 3D local detectors: HARRIS3D (HARRIS; STEPHENS, 1988), Kanade-Lucas-Tomasi (KLT) (TOMASI, 1991), 3D Scale Invariant Feature Transform (SIFT3D) (FLINT et al., 2007), Smallest Univalue Segment Assimilating Nucleus (SUSAN) (SMITH; BRADY, 1997), ISS3D, Surface Curvature (DESBRUN et al., 1999), and two variants of HARRIS3D called Lowe and Noble (RUSU; COUSINS, 2011). They conclude that SIFT3D and ISS3D achieve the best results in terms of repeatability, and ISS3D is more robust with respect to small transformations in the model, such as scaling, translation, and rotation. However, the parameter settings used in each experiment are not entirely clear. In the specific case of ISS3D, two parameters (r_{21} and r_{32}) define the saliency threshold (YU, 2009); however, they produce a number of keypoints in planar regions that do not seem to be good salient keypoints, and can lead to poor data associations, as can be seen in Fig. 28. In addition, this number of points can unnecessarily increase the time to find descriptors for each keypoint.

Prakhya and Qayyum (2015) propose a completely different method to evaluate the performance of 3D keypoint detectors. They show that the repeatability-based meth-

Figure 28. ISS3D local detector: comparison of the quality and quantity of key points obtained with different saliency thresholds.



(a) ISS3D with $r_{21} = r_{32} = 0.9$ **(b)** ISS3D with $r_{21} = r_{32} = 0.6$

Source: Author.

ods use individual object models (i.e., objects completely separate from other objects and scene background), unlike the problem of estimating the pose between two subsequent frames for indoor scenes where the detectors must be assessed, taking into account the background and the other objects in the scene. They claim that a good keypoint must have neighborhood with high variance (to ensure a discriminant feature) and must represent the scene comprehensively. They propose two tests to quantify these properties: the neighborhood variance test, and the neighborhood structure test that classifies the keypoints according to the structure of their neighborhoods, which can have planar or spherical shape. Comparing five different 3D detectors: HARRIS3D, ISS3D, NARF, SIFT3D, and Surface Entropy (SURE) (FIOLKA et al., 2012), they find that a combination of SURE and NARF detectors represents the scene more comprehensively than either of them alone. However, in individual evaluations of each detector, without the possibility of combinations, NARF and ISS3D offered a better performance than SURE in the neighborhood structure test.

In both studies, Filipe and Alexandre (2014) and Prakhya and Qayyum (2015), HARRIS3D achieved similar results under certain specific conditions. In Filipe and Alexandre (2014), HARRIS3D achieves good repeatability if the rotations between scenes compared are less than 15 degrees. In Prakhya and Qayyum (2015), HARRIS3D obtains good keypoints in places that have higher variance when applied to sequences that have scenes with smooth changes in the rotation.

Regarding the selection of the best 3D descriptor, there are two recent and comprehensive studies (HÄNSCH et al., 2014; GUO et al., 2016). Hänsch et al. (HÄNSCH et al.,

2014) present a discussion about the pros and cons of using 3D keypoints for registering point clouds. For this purpose, they use different combinations of two 3D local detectors, NARF and SIFT3D, and two 3D local descriptors, FPFH and Unique Signature of Histograms (SHOT). They show that the best results are obtained by applying the descriptors directly on point clouds, but this is not feasible due to its excessive computational cost. In the evaluation of the best descriptor, they conclude that FPFH is faster to compute and more robust against different viewpoints than the SHOT descriptor. In their assessment of the best detector, they claim that NARF is faster than SIFT3D, although NARF presents poor results in scenes with few objects when used along with the SHOT descriptor; Thus, they concluded that using FPFH in combination with NARF can yield better results.

There are other interesting studies. Guo et al. (2016) present a fairly detailed evaluation of 3D local descriptors used directly on point clouds. They compare ten different descriptors and show that, in applications in which time is crucial and that have a small number of points, FPFH outperforms SHOT. Holz et al. (2015) and Rusu et al. (2009) note that FPFH has a poor runtime performance when used in dense clouds of 3D points, which could be substantially improved by using reduced resolution in point clouds, and by limiting the keypoint search area.

3.4 Final remarks

Our proposal is based on the three articles mentioned above, (HENRY et al., 2012), (ENDRES et al., 2012) and (HOLZ et al., 2015), which show that the matches set used to generate the initial transformation is of paramount importance to ensure a significant improvement in the refining process. However, none of them took into account scenes with little or no structure, or low visual texture level.

In a preliminary study (VILLOTA; COSTA, 2015), we took this into consideration by combining 2D and 3D features. We used SURF as 2D detector and descriptor, HARRIS3D as 3D detector and FPFH as 3D descriptor. However, we used a small dataset that lacked groundtruth. The sum of squared distances was used in pairs of aligned point clouds as a distance error metric to validate the transformation. This metric is not sufficient to determine significant misalignment; for instance, in cases in which all features are placed on a straight line, the use of visual inspection was necessary, resulting in a less reliable evaluation.

With respect to selecting the 2D detector-descriptor combination, SIFT and SURF detectors and their respective descriptors appear to be the best options. In turn, while Endres et al. (2012) states that ORB shows the worst performance compared to SIFT and SURF, Henry et al. (2012) states that the FAST detector produces more reliable keypoints than SIFT. However, the ORB detector is an improved version of the FAST detector, and Rublee et al. (2011) demonstrated that Oriented FAST (oFAST) was better than the FAST detector. Rublee et al. (2011) also showed that oFAST combined with an improved version of the BRIEF descriptor (CALONDER et al., 2012) offers similar performance to that of SIFT and SURF, and was more robust to orientation changes. Because of this lack of consensus, we extend the assessment made by Endres et al., including assessments in scenes in which there is little or no level of structure or visual texture (see Section 5.1). With respect to 3D descriptor selection, the studies presented by Hänsch et al. (2014) and Guo et al. (2016) reached similar conclusions that FPFH is the best choice for 3D descriptors because it is robust, computationally efficient and lightweight. Therefore, we adopt FPFH as the 3D descriptor to be used. Regarding the choice of 3D detector, there is no consensus in the literature. Based on the studies presented by Filipe and Alexandre (2014) and Prakhya and Qayyum (2015), we decided to make our own assessment using the best 3D detectors found in each study: NARF and ISS3D. The HARRIS3D detector was also included in our assessment because of its robustness and efficiency established in studies by Hänsch et al. (2014) and Guo et al. (2016) (see Section 5.2).

Chapter 4

Pairwise registration using adaptive combination of 2D and 3D cues

"If you always do what you always did, you will always get what you always got."

-- H. Ford

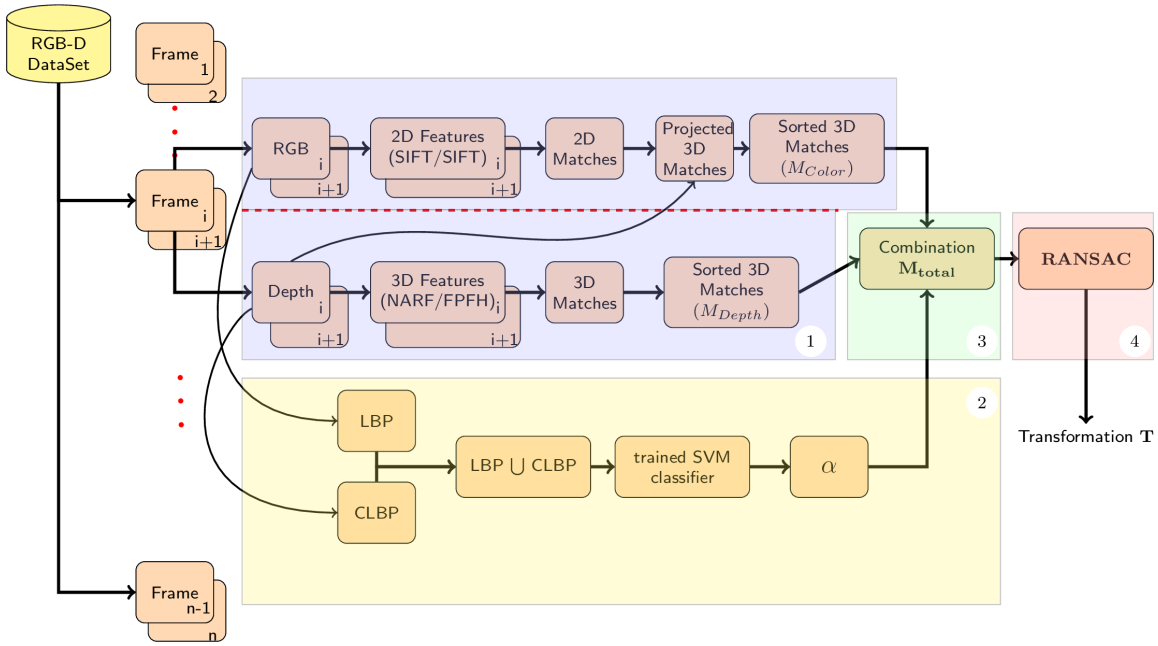
In this chapter we present a new framework for pairwise registration of 3D indoor scene. We call this framework: Registration by Adaptive Combination of Cues (ReACC). Figure 29 is a detailed block diagram of ReACC. The input to this framework is a 3D point cloud data set that is simultaneously used by stages 1 and 2 to find and to combine features respectively from a pair of frames. The output of ReACC is a rigid transformation matrix representing the rotation and translation, which must be applied on one frame so it is properly aligned with the other frame. The ReACC framework consists of four stages which are explained in greater detail below.

4.1 Stage 1

The first stage of our framework consists in obtaining 2D and 3D local features that allow efficiently matching two RGB images and two depth point clouds, respectively. In general, features need to meet two main conditions: On the one hand, it should be possible to track them under varying conditions such as illumination and view point changes. On the other hand, these features should be invariant to scale changes. Keypoints local detectors fulfill the first condition, while local descriptors meet the second one.

The selection of the best combination of 2D local detector and descriptor in our proposal is based on the experiments carried out by Endres et al. (2012) in which SIFT, SURF, and ORB were compared. However, Endres et al. (2012) only used video sequences

Figure 29. ReACC, a framework that uses adaptive combination of 2D and 3D cues for pairwise registration



Source: Author.

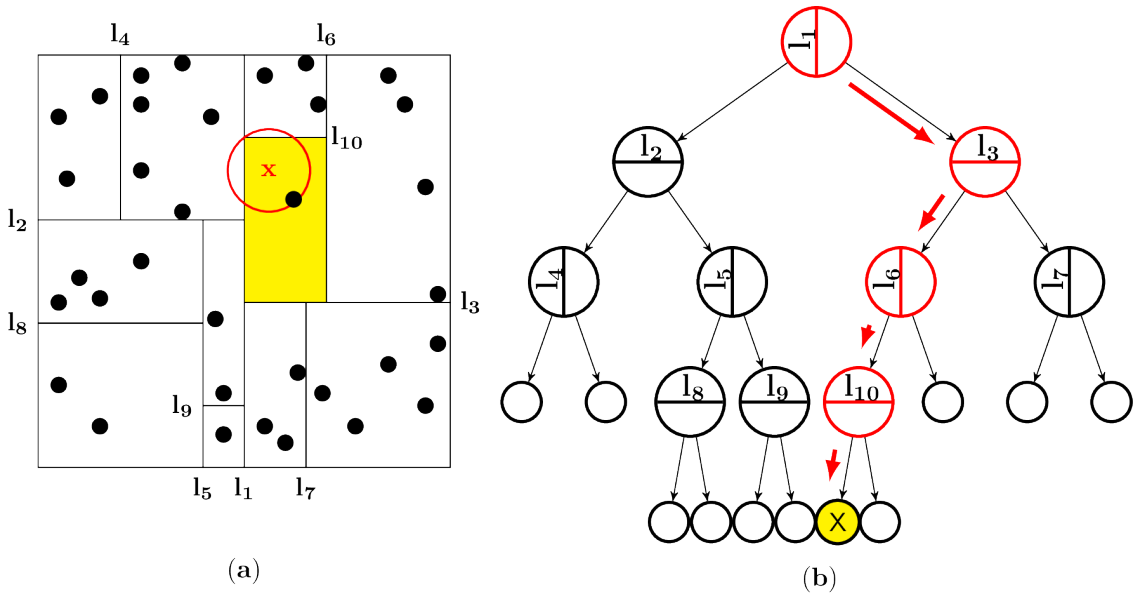
that had high visual texture level. Since we want to select the best 2D detector for all types of scenes, with or without visual texture, we compare the accuracy of these detectors using the same platform and parameters used by Endres et al. (2012) in a dataset with low visual texture level. Our experiments are described in Section 5.1. The resulting selection is SIFT as the 2D detector and also as the 2D descriptor.

The process of selecting the 3D local detector and descriptor was different. The 3D descriptor was selected without carrying out new experiments because most of the reviewed literature (HÄNSCH et al., 2014; GUO et al., 2016; VILLOTA; COSTA, 2015) agree that FPFH is the best option. However, there is no consensus about the best 3D detector. Therefore, we evaluated three different keypoint detectors: HARRIS (RUSU; COUSINS, 2011), ISS3D (YU, 2009), and NARF (STEDER; KONOLIGE, 2010), all them using FPFH (RUSU et al., 2009) as the fixed descriptor. These experiments are described in Section 5.2. The resulting selection is NARF as the 3D detector, and FPFH as the 3D descriptor.

The 2D features in each consecutive pair of images can be matched using a K-D tree with k-nearest neighbor search (KNN) algorithm. Let $Fm = \{fm_1, fm_2, ..., fm_n\}$ be a set of n features in metric space X with dimensionality δ in the first frame (let us call this frame: model), and $Fs = \{fs_1, fs_2, ..., fs_m\}$ be a set of m features in the same metric space X in the second frame (let us call this frame: scene). We can build a K-D tree with

the model features in a canonical way by splitting them either horizontally or vertically across the median element, forming new subsets (leaf nodes). Then, subsets can be partitioned over and over again until obtaining either **log** m levels with each feature in its own leaf or a predetermined number of features in each branch of the tree. The k-searching process for each feature $fs \in Fs$ is finding the k neighbors $KNN(fs, x), \forall x \in Fm$ that are the closest to fs with respect to a distance metric $d : X \times X \rightarrow \mathbb{R}$. We use Euclidean distance. Figure 30 is an example depicting a tree for two-dimensional features $\delta = 2$, and $k = 1$ representing a search for the closest neighbor. The block decomposition is shown in Fig. 30a and the tree representation is shown in Fig. 30b. In this example, the space for Fm is decomposed in ten levels from l_1 to l_{10}, and the feature being searched fs is marked with an "X" in the block model. In this case, with dimensionality equal to two, the use of KNN proves to be efficient, because it generates a circle search with fewer nodes to be visited.

Figure 30. K-D tree to features with dimensionality $\delta = 2$ and k-search equal to one, where (a) is the block decomposition and (b) the tree representation of the space formed by Fm. The space is vertically split first and this process continues alternating horizontal and vertical divisions. The node with the feature fs, is marked with "X" in the block model and highlighted in the tree representation.



Source: Author.

However, in cases with more dimensions for instance, the SIFT descriptor, whose dimensionality is $\delta = 128$, or the SURF descriptor, whose dimensionality is $\delta = 64$, the circle search is converted into a hypersphere involving more leaves and, therefore, the search for the closest neighbors can be inefficient. The library FLANN uses an ANN search to improve the search speed of the classical K-D tree algorithm. ANN introduces a parameter $\varepsilon \in (0, 1]$, such that any fm within distance at most ε times is an ε-approximate

nearest neighbor of fs, which, in practice, is faster than finding the exact nearest neighbor.

In Muja and Lowe (2014), they found that the best performance to find a good ANN is achieved using either the hierarchical k-means tree or multiple randomized K-D trees. We here use the randomized K-D trees which are built by choosing the split dimension randomly from the first D dimensions on the data with the greatest variance. For instance, consider a K-D tree for SIFT features where $\delta = 128$. If we select $D = 20$, during a search for a fs in the tree, no more than 20 of the 128 entries of the vector are considered. We also use $k = 1$ to select only the closest neighbor and d_{Color} as distance metric:

$$d_{Color} = \sqrt{\sum_{i=1}^{128}(fs_i - fm_i)^2}. \tag{4.1}$$

Finally, the 2D feature matches are sorted by the Euclidean distance and the 2D points associated with them are projected to 3D using the depth information, forming the M_{Color}. The procedure to match 3D features is the same. However, since the FPFH descriptor is a 33-dimensional vector, the distance metric is calculated as:

$$d_{Depth} = \sqrt{\sum_{i=1}^{33}(fs_i - fm_i)^2}, \tag{4.2}$$

and the 3D matched keypoints are ordered by d_{Depth}, forming the M_{Depth} set.

In Fig. 29, this stage has a subdivision represented by a horizontal dotted line, which means that the 2D process may run in parallel with the 3D process.

4.2 Stage 2

The α parameter value is an essential factor in our framework. It automatically allows a better balance of the matches from the RGB and depth images based on the existing levels of visual texture and structure in the scene frame.

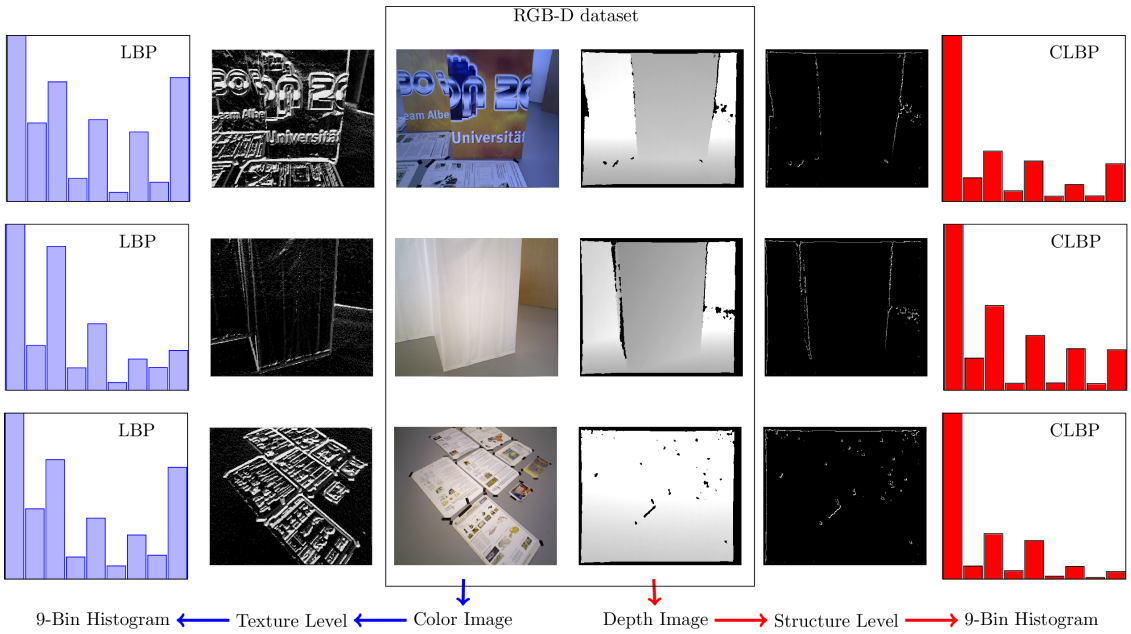
We use the intensity image and its Local Binary Patterns (see Chapter 2 for details), to extract a feature vector representing the level of visual texture in each RGB frame. On the other hand, we use the depth image and the CLBP to extract a 256-dimensional vector representing the level of structure in each frame.

Finally, we concatenate both vectors to form a 512-dimensional feature vector. In Fig. 31 we show the LBP and CLBP representations for images with different levels of

2D visual texture and 3D structure.

The resulting feature vector is then placed as input to a classifier that provides as output the value of the combination parameter α of the two types of information: the resulting RGB matches M_{Color}, and the resulting depth matches M_{Depth}.

Figure 31. Histograms of LBP and CLBP of images with different levels of 3D structure and 2D visual texture.



Source: Author.

Training and assessing the SVM Classifier. Stage 2 of our proposal requires a previously trained classifier, which is able to predict the optimal α value by observing the LBP and CLBP values of the *model* frame. We adopted the SVM classifier to execute that task.

Before being able to predict α values, a data set of feature values (extracted from images) with the correct α values must be provided to SVM. Then, the classifier builds a model of the classification task and is able to fulfill its role in our proposal without requiring further training. Thus, this Section describes how we built the data set, trained the classifier, and evaluated its performance, in order to assess if an automated classifier would be precise enough to be used in our framework.

The training set was built following this process: First, a set of point cloud pairs $< scene - model >$ are obtained from a dataset with groundtruth. From the scene' point cloud, we extract the gray and depth images and proceed to obtain a feature vector representing the different levels of visual texture and structure. The feature vector is then labeled with the α value that produces a better balance of 2D and 3D features for the

alignment task. Evidently, the correct α is not known in advance by only observing the image set; we thus use the point cloud pair $< scene-model >$ at stage 1 of our framework to find the 3D matches (M_{Color} and M_{Depth}), and combine them using five fixed α values: $0.1, 0.25, 0.5, 0.75$, and 0.9. For each α value, we obtain a pose $P_{i+\Delta}$ of the model (stage 4), and with Eq. 5.4 (see Section 5.1.1), we calculate the respective translational error in the alignment using the groundtruth poses Q_i, and $Q_{i+\Delta}$, assuming that $P_i = Q_i$. Finally, we select the α value associated with the smallest translational error as the correct label of the feature vector. In possession of this set of labeled feature vectors, a standard classifier training is performed and the SVM is ready to perform its task in our framework.

However, we firstly assess the SVM performance before further proceeding in our experimental evaluations. This assessment is carried out by the evaluation of the confusion matrix obtained in a preliminary experiment in which SVM tries to predict unlabeled image pairs. In this matrix, each row represents a label of a class and each column represents a prediction. Each cell $< i, j >$ indicates the number of times label i was predicted as label j. Therefore, the diagonal cells $< i, i >$ indicates the classes correctly predicted.

From the confusion matrix, we calculate four measurements: accuracy (ACC), precision (PREC), recall (REC) and F1 score (F1) as evaluation metrics (RUSSELL; NORVIG, 2003). Assuming that MC is the $c \times c$ confusion matrix for c classes with columns as predictions and rows as the labels,

$$ACC = \frac{\sum MC_{ii}}{n}, \tag{4.3}$$

where MC_{ii} represents the elements of the main diagonal in MC, and n is the number of instances. In the same way, we define:

$$PREC_i = \frac{MC_{ii}}{\sum_{j=1}^{c} MC_{ji}}, \tag{4.4}$$

$$REC_i = \frac{MC_{ii}}{\sum_{j=1}^{c} MC_{ij}}, \tag{4.5}$$

$$F1_i = \frac{2 * PREC_i * REC_i}{PREC_i + REC_i}. \tag{4.6}$$

The data sets used for training were: $fr3\_struct\_notexture$, $fr1\_desk2$, $fr3\_nostruc\_texture$, and $fr3\_struct\_texture$. The SVM was trained with a linear kernel, the complexity constant C equal to 1, and 10-fold cross-validation experiments were run for evaluation. We obtain the confusion matrix depicted in Table 1.

Table 1. Confusion Matrix

	Predictions				
Labels	$\alpha_{0.1}$	$\alpha_{0.25}$	$\alpha_{0.5}$	$\alpha_{0.75}$	$\alpha_{0.9}$
$\alpha_{\mathbf{0.1}}$	106	57	0	0	0
$\alpha_{\mathbf{0.25}}$	70	117	41	24	0
$\alpha_{\mathbf{0.5}}$	0	50	33	30	2
$\alpha_{\mathbf{0.75}}$	0	28	33	99	77
$\alpha_{\mathbf{0.9}}$	0	0	0	75	58

Source: Author.

Note that, in our case, small mistakes in the classification do not make a great difference (e.g., if an image labeled as $\alpha_{0.1}$ is predicted as $\alpha_{0.25}$, the error is minute when compared with the case in which this image was predicted as $\alpha_{0.9}$). Thus, the standard ACC metric is not adequate to evaluate the classifier, and we evaluate the classifier by changing equations (4.4), (4.5), and (4.6), so as to calculate the diagonal values taking into account small classification mistakes in the confusion matrix MC_{ii} as:

$$MC_{ii} = 0.5 * MC_{i,j-1} + MC_{ii} + 0.5 * MC_{i,j+1}. \tag{4.7}$$

For instance, in the specific case of $\alpha_{0.25}$, the original $MC_{ii} = 117$ is modified as $MC_{ii} = 0.5 * 70 + 117 + 0.5 * 41$ and this value change the scores of $PREC$, REC and $F1$, as shown in Table 2. In the experiments of the next Section, the SVM showed to be an adequate classifier.

Table 2. SVM Scores

Labels	**PREC**	**REC**	**F1**
$\alpha_{\mathbf{0.1}}$	0.80	0.83	0.81
$\alpha_{\mathbf{0.25}}$	0.74	0.68	0.71
$\alpha_{\mathbf{0.5}}$	0.65	0.63	0.64
$\alpha_{\mathbf{0.75}}$	0.55	0.65	0.60
$\alpha_{\mathbf{0.9}}$	0.70	0.72	0.71

Source: Author.

4.3 Stage 3

The α parameter can take values between 0 and 1. $\alpha = 0$ indicates a structured environment without any visual texture, while $\alpha = 1$ indicates a visual textured environment without any structure. Hence, when the environment presents a high degree of visual

texture and a low degree of structure, better results are achieved using M_{Color} matches in a greater proportion. Otherwise, a greater proportion of M_{Depth} matches is better suited. Such contribution can be calculated as,

$$|\mathbf{M_{total}}| = \lceil \alpha |M_{Color}| \rceil + \lfloor (1-\alpha)|M_{Depth}| \rfloor, \tag{4.8}$$

where $\lceil . \rceil$ and $\lfloor . \rfloor$ indicate the ceiling and the floor functions, respectively, and $|.|$ represents the set cardinality. The matched points of each data set, M_{Color} and M_{Depth}, are selected in order, according to the value calculated by Eq. 4.8.

4.4 Stage 4

The matches found by FLANN (Section 4.1) are in fact noisy, and if we use them directly to find the transformation matrix aligning pairs of point clouds, we could obtain extremely imprecise estimations. To overcome this problem we use RANSAC, which is considered one of the most powerful algorithms to refine the match set. In an iterative process described in Algorithm 2, RANSAC starts by randomly selecting a minimal subset of points composed of p_{sc_min} points extracted from the frame$_{i+1}$ (scene). This data subset (d_{rand}) is used in the next step to compute the homography transformation (T) relating the model (frame$_i$) with the scene (frame$_{i+1}$). Let us call this transformation T_h, since this transformation is not necessarily the optimal transformation. In step 5, the model is transformed using T_h and overlapped with the scene. A distance error is computed between each point in the scene and the respective nearest point in the transformed model. If this error is less than or equal to a threshold distance (d_{th}), the point is considered an inlier. Then, if the number of inliers is equal or greater than a threshold ($P_{inliers}$), T_h is considered a reasonably good homography transformation and an error between the model and the inliers (E_{model}) is calculated. Finally, T_h and E_{model} are saved as T and $error_{max}$, respectively. These steps are repeated until either the maximum number of iterations ($Iter_{max}$) is executed or $error_{max}$ is below a threshold (e_{th}). At the end of the execution of the RANSAC algorithm, we have transformation T that makes the best registration of the current frame with the previous frame.

Algorithm 2 Finding the transformation matrix using RANSAC

Require:
$model$: points matched in frame i
$scene$: points matched in frame $i+1$
p_{sc_min}: minimum number of points from scene required to fit the model
It_{max}: maximum number of iterations allowed
d_{th}: distance for determining if a point from scene fits a model
$p_{inliers}$: Number of points from scene to assert that the transformation matrix is reasonably good
e_{th}: Maximum error to assert an optimal transformation T
Ensure: T: Transformation matrix between frames i and $i+1$

Initialize:
  $h \leftarrow 0$
  $T \leftarrow null$
  $error_{max} \leftarrow \infty$
while $h \leq It_{max}$ **Or** $error_{max} \leq e_{th}$ **do**
  $d_{rand} \leftarrow$ select randomly p_{sc_min} points from $scene$
  $T_h \leftarrow$ Compute the hypothesis of T using d_{rand} points
  $P_{model} \leftarrow$ Use T_h to find the model transformed
  $inliers \leftarrow \emptyset$
  for every point in $scene$ **do**
    if point fits P_{model} with error $\leq d_{th}$ **then**
      Add point to $inliers$
    end if
  end for
  if $inliers \geq p_{inliers}$ **then**
    E_{model} ▷ Compute a distance error between $model$ and $inliers$
    if $E_{model} \leq error_{max}$ **then**
      $T \leftarrow T_h$
      $error_{max} \leftarrow E_{model}$
    end if
  end if
  $h \leftarrow h + 1$
end while
return T

4.5 Final remarks

In this chapter we describe ReACC, a new framework that we propose to improve the alignment of two RGB-D frames, taking into account both 2D and 3D data, whose combination is a function of the characteristics of the current scene. If the scene has little geometric structure and good visual texture, more RGB data are used. On the other hand, if the scene has little visual texture and a lot o geometric structure, depth data is used with greater emphasis.

The automatic adjustment of the use of the data depending on the scene observed is a new and effective contribution. Our experiments described in the next chapter demonstrate the efectiveness of our proposal.

Chapter 5

Experiments

"Science, my lad, has been built upon many errors; but they are errors which it was good to fall into, for they led to the truth."

-- Jules Verne, Journey to the Center of the Earth

This chapter shows the experiments carried out to select the most appropriate methods to be used in each stage of ReACC, and, additionally, to show the improvements achieved in the initial alignment, when our framework combining 2D and 3D features is used.

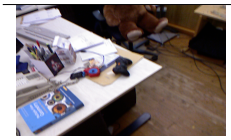
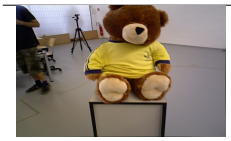
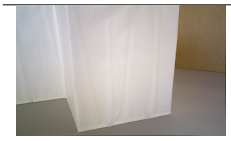
We evaluate each stage of our framework by comparing it either with other works presenting similar results or using different combinations of methods. While in section 5.1, we compare our results with Endres et al. (2012) for evaluating 2D local detectors and descriptors, in section 5.2 we compare three different 3D detectors combining them with a fixed 3D descriptor for evaluating 3D local detectors. Finally, in section 5.3 the whole framework using the best 2D and 3D combinations is evaluated by comparing it with two cases: one case using only 2D local features (ENDRES et al., 2012; HENRY et al., 2012), and the other case using only 3D local features (HOLZ et al., 2015).

In these experiments, we use the datasets provided by the Computer Vision Group of Technische Universiät München[1] to evaluate our proposal. We selected datasets with large variations among them, aiming to perform a comprehensive experimental evaluation. These datasets come with an associated groundtruth camera pose and include cases with variation in illumination, visual texture, scene structure, and with an unrestricted camera motion speed (resulting in frames with blurred images). We did not use synthetic datasets so as to gain a better insight into real indoor environment problems.

Table 3 shows the datasets used with their main specifications and a brief description.

[1] ⟨http://vision.in.tum.de/data/datasets⟩

Table 3. Main specifications of datasets used.

Name	Video Duration (s)	Average Transl. Velocity (m/s)	Average Angular Velocity (deg/s)	Description
 *fr1_360*	28.69	0.210	41.600	This sequence contains a 360 degree turn in a typical office environment.
 *fr1_desk2*	24.86	0.426	29.308	This sequence contains several sweeps over four desks in a typical environment office
 *fr3_large_cabinet*	33.98	0.362	8.747	The RGBD camera was moved in a circle around a large office cabinet. The cabinet has little texture and structure, but has mostly planar surfaces and right angles.
 *fr3_teddy*	80.79	0.248	20.410	The RGBD camera was moved around a teddy bear in two rounds at different heights. The teddy bear has a soft fur and wears a yellow, smooth shirt.
 *fr3_struct_notexture*	27.28	0.166	4.000	The RGBD camera was moved in one meter height along a zig-zag structure built from wooden panels. The object is fully wrapped in a white plastic foil with little to no texture.
 *fr3_nostruc_texture*	15.53	0.299	2.890	The RGBD camera has been moved in two meters height along a textured, planar surface. The texture is highly discriminative as it consists of several conference posters.
 *fr3_struct_texture*	31.55	0.193	4.323	The RGBD camera was moved in one meter height along a zig-zag structure built from wooden panels. The object is fully wrapped in a colorful plastic foil with strong texture. The floor in front of the object has been covered with several posters which have also a strong texture.

* The data was recorded at full frame rate ($30Hz$) and sensor resolution (640×480)

Source: Author.

We also use available detectors and descriptors, namely, SIFT, SURF, and ORB, included in OpenCV[2]. While, we use NARF, ISS3D, HARRIS3D and FPFH, included in PCL[3]. Furthermore, we use RGBDSLAM[4] to evaluate 2D detectors and descriptors, and LibSVM[5] to train the SVM classifier.

5.1 2D detector/descriptor selection

To select the most suitable 2D detector/descriptor pair in our main experiment, we complement the Endres' evaluation (ENDRES et al., 2012) including a dataset with low texture level, which enables us to have a better indicator. We select the dataset called *fr3_struct_notexture* and evaluate it using the RGBDSLAM system and 5 possible combinations of 2D local descriptors and detectors.

5.1.1 Metric used to assess 2D local detector and descriptor

We use the evaluation metric proposed by Sturm et al. (STURM et al., 2012), called Relative Pose Error (RPE) to evaluate the 2D detector/descriptor selection and also to evaluate ReACC.

RPE measures the local accuracy of the trajectory over a fixed time interval Δ. Special Euclidean group, $SE(3)$, is the group of affine rigid motions formed by the rotation T_R and translation T_T of the transformation matrix T,

$$T = \left[\begin{array}{ccc|c} & T_R & & T_T \\ \hline 0 & 0 & 0 & 1 \end{array}\right]. \tag{5.1}$$

Assuming that $Q_1, Q_2, ...Q_i \in SE(3)$ is the sequence of groundtruth poses on a full trajectory, and $P = P_1, P_2, ...P_i \in SE(3)$ is the estimated poses in the same sequence, a relative pose error (E_i) between each pair $< i, i + \Delta >$ is calculated as,

$$E_i = (Q_i^{-1} Q_{i+\Delta})^{-1} (P_i^{-1} P_{i+\Delta}), \tag{5.2}$$

[2] ⟨http://opencv.org/.⟩
[3] ⟨http://pointclouds.org/.⟩
[4] ⟨http://felixendres.github.io/rgbdslam_v2/⟩
[5] ⟨https://www.csie.ntu.edu.tw/~cjlin/libsvm/.⟩

where $\Delta = 1$.

We assume that $P_1 = Q_1$ and each next pose P_j, $j = i + 1$, is calculated using this equation,

$$T_{ij} = Q_i^{-1} P_j, \tag{5.3}$$

where T_{ij} is the relative transformation between the frames f_i and f_j, previously calculated using our framework.

Then, the Root Mean Squared Error (RMSE) is computed over all time n in the translational component:

$$RMSE(\mathbf{E}_{1:n}, \Delta) = \left(\frac{1}{m} \sum_{i=1}^{m} \|E_{Ti}\|^2 \right)^{\frac{1}{2}}, \tag{5.4}$$

where $m = n - \Delta$ and E_{Ti} is the translational component of the i^{th} relative pose error. In this equation only the translational component is used, because in (STURM et al., 2012), they found that the comparison by translational error is sufficient since the rotational error appears as a translational error when the camera is moved.

5.1.2 Results to 2D local detector and descriptor selection

Table 4 presents the translational and rotational errors obtained for each combination, showing similar results to those in Endres et al. (2012). Furthermore, the ORB detector is observed to greatly improve if used in combination with the SIFT descriptor. However, the best possible combination arises when a SIFT detector is used together with its descriptor.

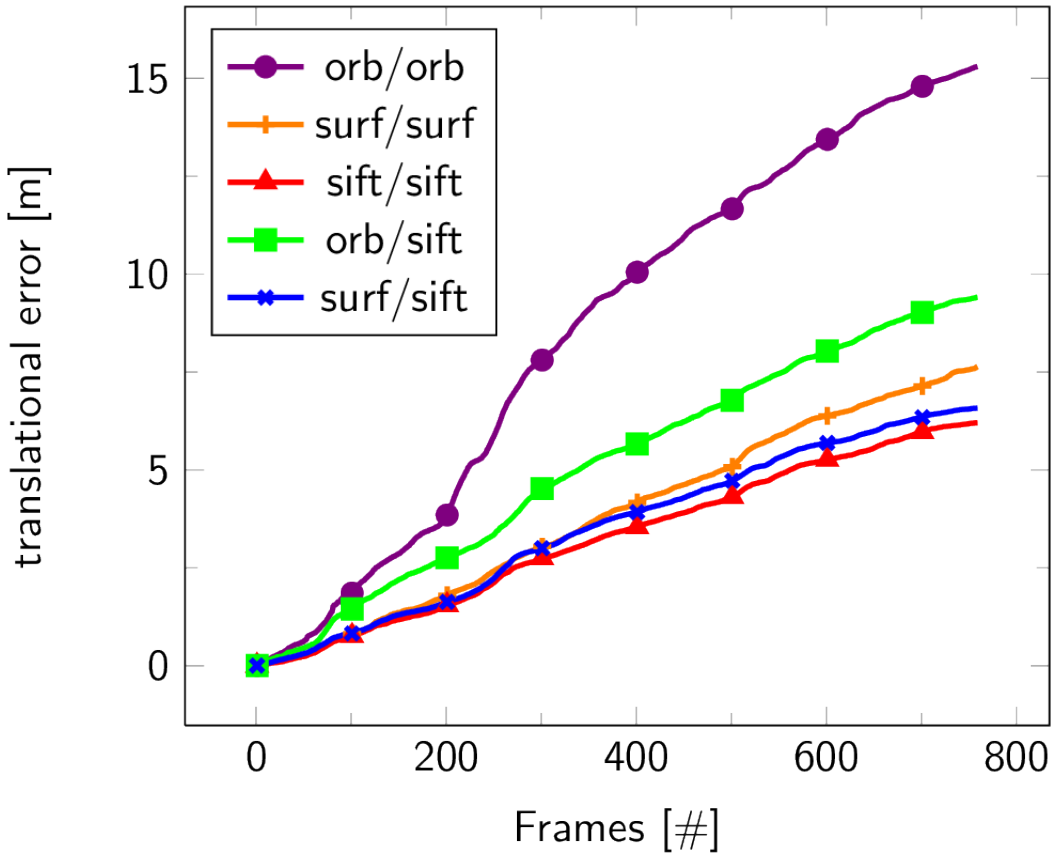
A visualization of the cumulative relative pose error throughout the entire sequence for each of these combinations is given in Fig. 32.

Table 4. Evaluation of the accuracy with respect to 2D detector/descriptor combination

Detector/Descriptor	**Transl. RMSE** Avg	**Rot. RMSE** Avg
ORB/ORB	0.0201 m	0.0313°
SURF/SURF	0.0100 m	0.0081°
SIFT/SIFT	**0.0081 m**	**0.0064°**
ORB/SIFT	0.0123 m	0.0086°
SURF/SIFT	0.0086 m	0.0071°

Source: Author.

Figure 32. Cumulative relative pose error for each 2D detector/descriptor combinations. SIFT/SIFT combination is slightly better than SURF/SURF and SURF/SIFT.



Source: Author.

5.2 3D detector/descriptor selection

We selected 3 different keypoint detectors: HARRIS, ISS3D and NARF with FPFH as fixed descriptor. Table 5 depicts the parameters used in our experiments that guarantee a similar average of keypoints for each detector.

5.2.1 Metric used for 3D local detectors

In order to select 3D detectors, we chose four different datasets with an associated groundtruth camera pose, and proceed as follows: first, for each two consecutive frames f_i and f_j in a sequence, $j = i + 1$, we extract the keypoints and their respective descriptors, match them and lastly we estimate the rigid-body transformation T_{ij}between the two frames.

We use T_{ij} to align the two frames and calculate the Euclidean distance between aligned matches as the error metric. Let M be a set of n matches, $M = \{m_1, m_2, ..., m_n\}$, where each m_k corresponds to each keypoint in frame i and its respective keypoint matched

Table 5. Parameter values for the 3D detectors.

Detector	Parameters
NARF	angularResolution 0.3° maxAngleWidth 360° maxAngleHeight 180° support_size 0.1
ISS3D	setSalientRadius 5*resolution setNonMaxRadius 3*resolution setNormalRadius 5*resolution setBorderRaidus 2*resolution setMinNeighbors 20 setThreshold21 0.6 setThreshold32 0.6
HARRIS	setRadius 0.07 setRadiusSearch 0.07 setNonMaxSupression true

Source: Author.

in frame j. We calculate the Euclidean distance d_i for each matched pair and calculate the averaged distance error $AVG(\mathbf{E_m})$:

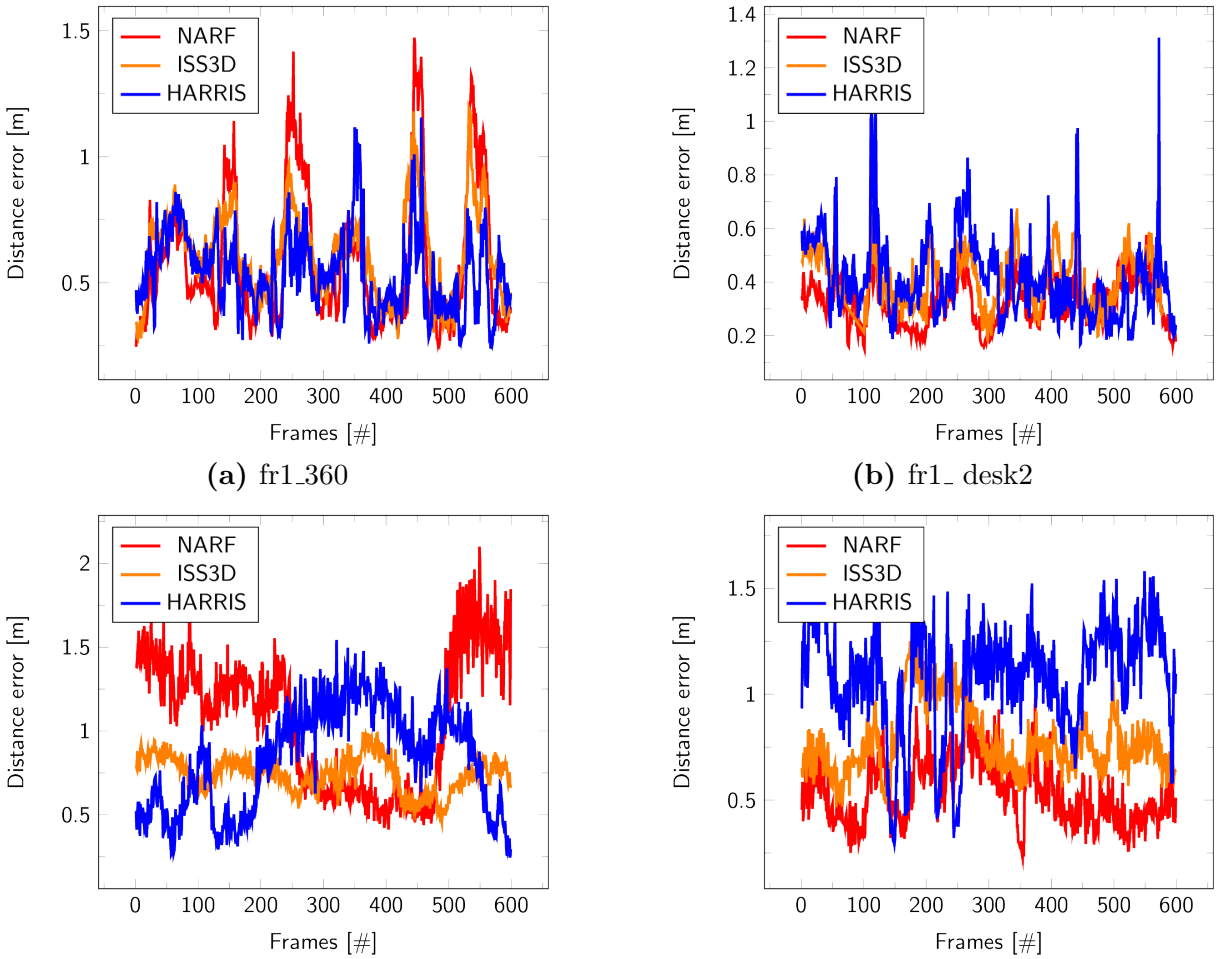
$$AVG(\mathbf{E_m}) = \frac{1}{n}\sum_{i=1}^{n} d_i, \tag{5.5}$$

where d_i is the Euclidean distance between pairs of matched points, and n is the cardinality of the set M.

5.2.2 Results of the 3D local detector selection

A visualization of the performance of each detector is shown in Fig. 33. NARF presents a better and more stable performance to the sequences $fr1\_desk2$ and $fr3\_teddy$. In the $fr1\_360$ sequence, all the descriptors present similar performance, while for the $fr3\_large\_cabinet$ sequence, NARF presents an unstable behavior. It is clear that with sequences in which the camera presents greater movement causing blurred objects boundaries, NARF has the worst performance. This assessment is in line with the works presented by Hänsch et al. (2014) and Guo et al. (2016). However, we selected NARF taking into account the best global performance as shown in Table 6.

Figure 33. Distance error frame by frame for each sequence comparing three different local descriptors: NARF, ISS3D and HARRIS3D



(a) fr1_360 **(b)** fr1_ desk2

(c) fr3_large_cabinet **(d)** fr3_teddy

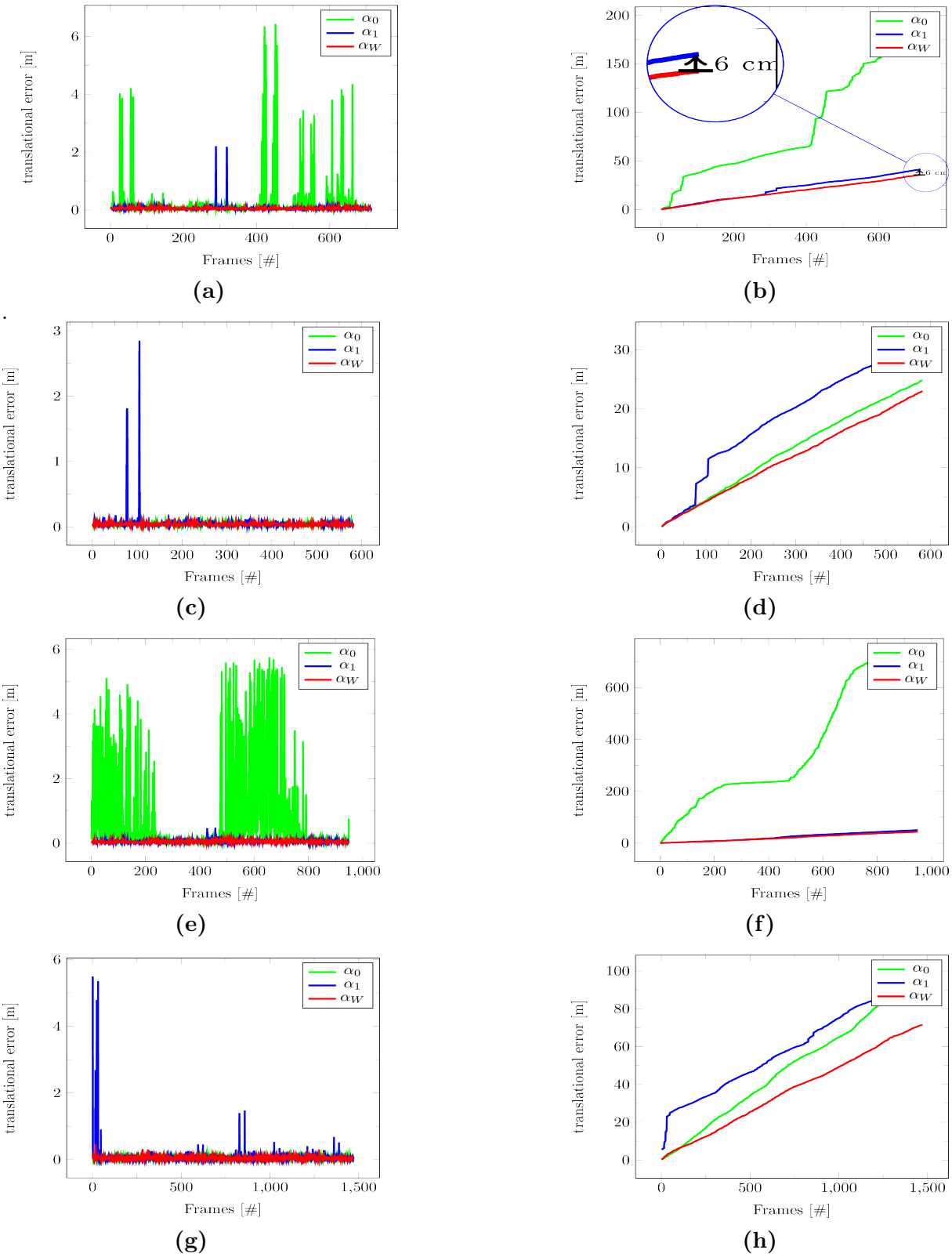
Source: Author.

5.3 Evaluation of our proposal

Based on the previous experiments, we selected the best combinations of detectors and descriptors and included them in our framework. We compare our proposal with the cases in which only either 2D or 3D detectors/descriptors are used, by evaluating the initial alignment achieved. We use the RMSE of the relative pose error (See section 5.2.1) in four datasets: $fr1\_360$, $fr1\_desk2$, $fr3\_large\_cabinet$ and $fr3\_teddy$, of which only $fr1\_desk2$ was previously used in the SVM training. Table 7 and Fig. 34 summarize the results.

In table 7, we can see that our proposal combining 2D and 3D matches always outperforms those proposals using only either 2D matches or 3D matches. Since this table shows only the average error in the whole sequence, we include a more detailed visual comparison

Figure 34. Relative pose error. Each row contains the figures foreach dataset which are in order : $fr1\_360$, $fr1\_desk2$, $fr3\_large\_cabinet$ and $fr3\_teddy$ respectively. Figures (a), (c),(e) and (g) show the relative pose error between pair of frames. Figures (b),(d),(f) and (h) show the acummulate error through all sequence



Source: Author.

Table 6. Evaluation of the 3D detectors accuracy, using FPFH as descriptor.

Dataset	**NARF** Avg(E_M)±Std. Dev	**ISS3D** Avg(E_M)±Std. Dev	**HARRIS3D** Avg(E_M)±Std. Dev
fr1_360	0.58 m±0.27 m	0.62 m±0.18 m	**0.54 m±0.16 m**
fr1_ desk2	**0.33 m±0.08 m**	0.41 m±0.09 m	0.41 m±0.15 m
fr3_cabinet	1.07 m±0.40 m	**0.74 m±0.10 m**	0.84 m±0.31 m
fr3_teddy	**0.53 m±0.15 m**	0.78 m±0.16 m	1.08 m±0.26 m
$\sum_{Avg}$	**2.5098 m**	2.5562 m	2.8787 m

Source: Author.

Table 7. RMSE of the relative pose error for each sequence

Dataset	**Average Error**		
	$\alpha = 0$	$\alpha = 1$	$\alpha_{adaptative}$
fr1_360	0.265 m	0.057 m	**0.049 m**
fr1_desk2	0.042 m	0.054 m	**0.039 m**
fr3_large_cabinet	0.755 m	0.052 m	**0.045 m**
fr3_teddy	0.066 m	0.066 m	**0.048 m**

Source: Author.

in Fig. 34. In this figure items (a),(c),(e) and (g), show the translational error for each pair of frames in the whole sequence for each dataset. We can see that the great translational errors using 2D or 3D matches separately, can be significantly decreased using the combination proposed. Items (b),(d),(f) and (h), show the cumulative error reached throughout the trajectory for each dataset. Similar behaviors for all sequences can be achieved if error values above $0.1m$ are removed. However, the number of times this value exceeds $0.1m$ is important, as verified in Table 8, where these are placed in brackets for each dataset. This table clearly shows that our proposal remains for most of the time within a more reasonable range ($00.1m$) which should allow refinement algorithms, such as ICP, a faster convergence also avoiding convergence to a local minimum.

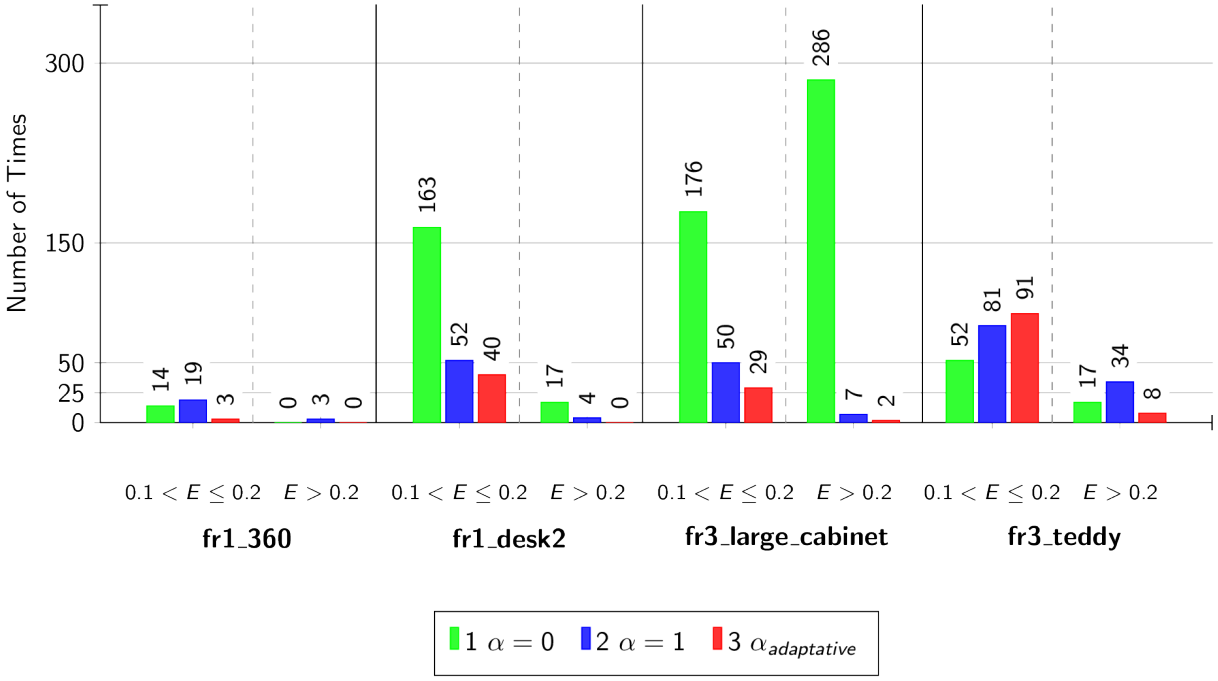
Table 8. RMSE of the relative pose error for each sequence removing errors above $0.1m$

Dataset	**Average Error [m] / number of exceeding 0.1m error**		
	$\alpha = 0$	$\alpha = 1$	$\alpha_{adaptative}$
fr1_360	0.034/(280)	0.042/(56)	**0.042/(40)**
fr1_desk2	0.040/(14)	0.039/(22)	**0.038/(3)**
fr3_large_cabinet	0.022/(561)	0.043/(57)	**0.041/(31)**
fr3_teddy	0.041/(269)	0.036/(115)	**0.037/(99)**

Source: Author.

Finally, Fig. 35 shows in detail the number of times each case has translational errors between $0.1m$ and $0.2m$ and translational errors above $0.2m$, since the latter would cause poor performance for refinement algorithms such as ICP. We can see that, in datasets

Figure 35. Number of times at which the relative pose error is greater than $0.1m$. In addition, this number of errors is subdivided into two groups: $0.1m < E \leq 0.2m$ and $E > 0.2m$.



Source: Author.

$fr3\_large\_cabinet$ and $fr3\_teddy$, our proposal rarely exceeds $0.2m$, showing a more stable behavior in all cases.

Chapter 6

Conclusions

We presented a new framework for pairwise registration, which improves the initial alignment between pairs of frames in a sequence for indoor environments. Our contribution is threefold in the area of pairwise alignment using sparse features.

Firstly, we presented a method for combining 2D and 3D matches that always produces a better alignment regardless of whether the environment has little or no texture and/or structure degree. This is a substantial improvement in the earlier works that used only one type of feature, in which the lack of features between a pair of frames was poorly settled using the same transformation aligning the previous pair of frames.

Secondly, we compared several possible combinations of detectors and descriptors using datasets in real indoor environments, presenting significant light and motion variation, which allowed us to select a robust detector/descriptor pair.

Finally, we also presented a novel method using a SVM classifier to adaptively calculate the parameter that controls the appropriate amount of 2D and 3D features to be combined in the matching process. Our experiments demonstrated that our adaptive parameter produces a better combination of 2D and 3D features, as reflected in the excellent results for alignment between pairs of frames. Experiments also showed a significant reduction in the translational error range throughout the whole sequence for all the datasets used.

Future works include extending our adaptive parameter not only to select the best combination between 2D and 3D features, but also to select the most appropriate detector as a function of the scene; for instance, HARRIS3D and ISS3D showed better performance than NARF in some frames in our evaluations; hence, switching detectors according to

the current frame pair could be an improvement to our framework. Conversely, a parallel implementation of ReACC shall be made, since in this thesis we showed a high degree of reliability and robustness in the alignment, but the cost of computation was no taken into account. The parallel ReACC may be used in a full visual SLAM scheme for testing performance improvements over other works presented in the current literature.

Finally, we would like to point out that the modularity of ReACC enables users to easily combine different types of 2D and 3D detectors and descriptors. ReACC can also be easily adapted to 3D object reconstruction tasks, in which, there are similar problems in matching image intensities of the same objects in different views. These problems can be better solved without some of the currently used (incorrect) assumptions, such as the brightness constancy assumption, which does not hold when the object changes its orientation.

References

BAILEY, T.; DURRANT-WHYTE, H. Simultaneous localization and mapping (SLAM): Part II. **Robotics & Automation Magazine, IEEE**, v. 13, n. 3, p. 108–117, 2006.

BAY, H.; TUYTELAARS, T.; GOOL, L. V. Surf: Speeded up robust features. **Computer Vision–ECCV**, p. 404–417, 2006.

BORRMANN, D. et al. The Efficient Extension of Lu and Milios SLAM to 6 DoF. In: **Proceedings of the 4th International Symposium on 3D Data Processing, Visualization and Transmission (3DPVT'08).** [S.l.: s.n.], 2008. p. 29–36.

BYLOW, E. et al. Real-Time Camera Tracking and 3D Reconstruction Using Signed Distance Functions. **Robotics: Science and Systems (RSS) Conference**, v. 9, 2013. ISSN 2330-765X (online).

CALONDER, M.; LEPETIT, V.; FUA, P. Keypoint signatures for fast learning and recognition. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2008. v. 5302 LNCS, n. Part 1, p. 58–71. ISBN 3540886818. ISSN 03029743.

CALONDER, M. et al. BRIEF: Computing a local binary descriptor very fast. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 34, n. 7, p. 1281–1298, 2012. ISSN 01628828.

CHUN, S.; LEE, C.; LEE, S. Facial expression recognition using extended local binary patterns of 3D curvature. **Multimedia and Ubiquitous Engineering**, p. 1005–1012, 2013. Available at: ⟨http://link.springer.com/chapter/10.1007/978-94-007-6738-6_124⟩.

DESBRUN, M. et al. Implicit fairing of irregular meshes using diffusion and curvature flow. **Proceedings of the 26th annual conference on Computer graphics and interactive techniques - SIGGRAPH '99**, v. 33, p. 317–324, 1999. ISSN 00978930. Available at: ⟨http://portal.acm.org/citation.cfm?doid=311535.311576⟩.

DURRANT-WHYTE, H.; BAILEY, T. Simultaneous localization and mapping: part I. **Robotics & Automation Magazine, IEEE**, v. 13, n. 2, p. 99–110, 2006.

ELIAZAR, A.; PARR, R. DP-SLAM: Fast, robust simultaneous localization and mapping without predetermined landmarks. In: **IJCAI**. [S.l.: s.n.], 2003. v. 3, p. 1135–1142.

ENDRES, F. et al. An evaluation of the RGB-D SLAM system. **2012 IEEE International Conference on Robotics and Automation**, Ieee, v. 3, n. c, p. 1691–1696, may 2012. Available at: ⟨http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6225199⟩.

FILIPE, S.; ALEXANDRE, L. A comparative evaluation of 3D keypoint detectors in a RGB-D object dataset. **Computer Vision Theory and**, 2014. Available at: ⟨http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=7294⟩.

FIOLKA, T. et al. SURE: Surface entropy for distinctive 3D features. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2012. v. 7463 LNAI, p. 74–93. ISBN 9783642327315. ISSN 03029743.

FIRMAN, M. et al. Learning to discover objects in RGB-D images using correlation clustering. In: **IEEE International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2013. p. 1107–1112. ISBN 9781467363587. ISSN 21530858.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. **Communications of the ACM**, v. 24, n. 6, p. 381–395, 1981.

FLINT, A.; DICK, A.; HENGEL, A. V. D. Thrift: Local 3D structure recognition. In: **Proceedings - Digital Image Computing Techniques and Applications: 9th Biennial Conference of the Australian Pattern Recognition Society, DICTA 2007**. [S.l.: s.n.], 2007. p. 182–188. ISBN 0769530672.

GEIGER, A.; WANG, C. Joint 3D object and layout inference from a single RGB-D image. In: **Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)**. [S.l.: s.n.], 2015. v. 9358, p. 183–195. ISBN 9783319249469. ISSN 16113349.

GRISETTI, G. et al. A tutorial on graph-based SLAM. **Intelligent Transportation Systems Magazine, IEEE**, v. 2, n. 4, p. 31–43, 2010.

GRISETTI, G.; STACHNISS, C.; BURGARD, W. Improved techniques for grid mapping with rao-blackwellized particle filters. **Robotics, IEEE Transactions on**, IEEE, v. 23, n. 1, p. 34–46, 2007.

GUO, Y. et al. A Comprehensive Performance Evaluation of 3D Local Feature Descriptors. **International Journal of Computer Vision**, v. 116, n. 1, p. 66–89, 2016. ISSN 15731405.

GUPTA, S. et al. Indoor Scene Understanding with RGB-D Images: Bottom-up Segmentation, Object Detection and Semantic Segmentation. **International Journal of Computer Vision**, v. 112, n. 2, p. 133–149, 2015. ISSN 15731405.

HÄNSCH, R.; WEBER, T.; HELLWICH, O. Comparison of 3D interest point detectors and descriptors for point cloud fusion. **ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences**, II-3, n. September, p. 57–64, 2014. ISSN 2194-9050. Available at: ⟨http://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/II-3/57/2014/⟩.

HARRIS, C.; STEPHENS, M. A Combined Corner and Edge Detector. **Procedings of the Alvey Vision Conference 1988**, p. 147–151, 1988. ISSN 09639292. Available at: ⟨http://www.bmva.org/bmvc/1988/avc-88-023.html⟩.

HENRY, P. et al. RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. **The International Journal of Robotics Research**, SAGE Publications, v. 31, n. 5, p. 647–663, 2012.

HÖGMAN, V. **Building a 3D Map from RGB-D Sensors.** Thesis (Master) — KTH, School of Computer Science and Communication (CSC)., 2012.

HOLZ, D. et al. Registration with the point cloud library: A modular framework for aligning in 3-D. **IEEE Robotics and Automation Magazine**, v. 22, n. 4, p. 110–124, 2015. ISSN 10709932.

KHOSHELHAM, K. Accuracy Analysis of Kinect Depth Data. **International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, XXXVIII-5, n. August, p. 133–138, 2012. ISSN 1682-1777.

KUMMERLE, R. et al. g 2 o: A general framework for graph optimization. **Robotics and Automation (ICRA), 2011 IEEE International Conference**, p. 3607–3613, 2011.

LOWE, D. G. Object recognition from local scale-invariant features. In: **Proceedings of the Seventh IEEE International Conference on Computer Vision**. [s.n.], 1999. v. 2, n. 8, p. 1150–1157. ISBN 0-7695-0164-8. ISSN 0-7695-0164-8. Available at: ⟨http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=790410⟩.

LOWE, D. G. Distinctive image features from scale-invariant keypoints. **International Journal of Computer Vision**, v. 60, n. 2, p. 91–110, 2004. ISSN 09205691.

LU, F.; MILIOS, E. Globally consistent range scan alignment for environment mapping. **Autonomous robots**, v. 4, n. 4, p. 333–349, 1997.

MUJA, M.; LOWE, D. G. Scalable nearest neighbor algorithms for high dimensional data. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 36, n. 11, p. 2227–2240, 2014. ISSN 01628828.

OJALA, T.; PIETIKAINEN, M.; HARWOOD, D. Performance evaluation of texture measures with classification based on Kullback discrimination of distributions. In: **Pattern Recognition, 1994. Vol. 1 - Conference A: Computer Vision amp; Image Processing., Proceedings of the 12th IAPR International Conference on**. [S.l.: s.n.], 1994. v. 1, p. 582–585 vol.1. ISBN 0-8186-6265-4. ISSN 00313203.

PRAKHYA, S. M.; QAYYUM, U. Sparse Depth Odometry: 3D keypoint based pose estimation from dense depth data. **2015 IEEE International Conference on Robotics and Automation (ICRA)**, p. 4216–4223, 2015. Available at: ⟨http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7139780⟩.

ROSTEN, E.; PORTER, R.; DRUMMOND, T. Faster and better: A machine learning approach to corner detection. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 32, n. 1, p. 105–119, 2010. ISSN 01628828.

RUBLEE, E. et al. ORB: An efficient alternative to SIFT or SURF. In: **Proceedings of the IEEE International Conference on Computer Vision**. [S.l.: s.n.], 2011. p. 2564–2571. ISBN 9781457711015. ISSN 1550-5499.

RUSSELL, B. C. et al. Segmenting Scenes by Matching Image Composites. **Advances in Neural Information Processing Systems**, p. 1–9, 2009.

RUSSELL, J.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. [s.n.], 2003. 1132 p. ISSN 00206539. ISBN 0137903952. Available at: ⟨http://amazon.de/o/ASIN/0130803022/⟩.

RUSU, R. B.; BLODOW, N.; BEETZ, M. Fast Point Feature Histograms (FPFH) for 3D registration. **2009 IEEE International Conference on Robotics and Automation**, p. 3212–3217, 2009. ISSN 1050-4729.

RUSU, R. B. et al. Aligning point cloud views using persistent feature histograms. In: **2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS**. [S.l.: s.n.], 2008. p. 3384–3391. ISBN 9781424420582. ISSN <null>.

RUSU, R. B.; COUSINS, S. 3D is here: point cloud library. **IEEE International Conference on Robotics and Automation**, p. 1 – 4, 2011. ISSN 1050-4729. Available at: ⟨http://pointclouds.org/⟩.

RUSU, R. B. et al. Towards 3D point cloud based object maps for household environments. **Robotics and Autonomous Systems**, v. 56, n. 11, p. 927–941, 2008.

SCHERER, S. A.; DUBE, D.; ZELL, A. Using depth in visual simultaneous localisation and mapping. **Robotics and Automation (ICRA), 2012 IEEE International Conference**, p. 5216–5221, 2012.

SHAO, T. et al. An interactive approach to semantic modeling of indoor scenes with an RGBD camera. **ACM Trans. Graph.**, v. 31, n. 6, p. 136:1—-136:11, 2012. ISSN 07300301. Available at: ⟨http://doi.acm.org/10.1145/2366145.2366155⟩.

SMITH, S.; BRADY, J. SUSAN—a new approach to low level image processing. **International journal of computer vision**, v. 23, n. 1, p. 45–78, 1997. ISSN 1573-1405. Available at: ⟨http://link.springer.com/article/10.1023/A:1007963824710⟩.

STEDER, B.; KONOLIGE, K. NARF : 3D Range Image Features for Object Recognition. **October**, v. 215, p. 11, 2010. Available at: ⟨http://ais.informatik.uni-freiburg.de/publications/papers/steder10irosws.pdf⟩.

STURM, J. et al. A benchmark for the evaluation of RGB-D SLAM systems. In: **IEEE International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2012. p. 573–580. ISBN 9781467317375. ISSN 21530858.

THRUN, S.; BURGARD, W.; FOX, D. **Probabilistic robotics**. [S.l.]: MIT press, 2005.

TOMASI, C. Detection and Tracking of Point Features. **School of Computer Science, Carnegie Mellon Univ.**, v. 91, n. April, p. 1–22, 1991. ISSN 00313203. Available at: ⟨http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.5899{&}rep=rep1{&}ty⟩.

TYKKÄLÄ, T. et al. Live RGB-D camera tracking for television production studios. **Journal of Visual Communication and Image Representation**, v. 25, n. 1, p. 207–217, 2014. ISSN 10473203.

VILLOTA, J.; COSTA, A. Aligning RGB-D Point Clouds through Adaptive Integration of Color and Depth Cues. **2015 12th Latin American Robotics**, p. 309–314, 2015. Available at: ⟨http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=7402⟩.

YU, Z. Intrinsic shape signatures: A shape descriptor for 3D object recognition. In: **2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009**. [S.l.: s.n.], 2009. p. 689–696. ISBN 9781424444427.

ZHANG, Z. Iterative point matching for registration of free-form curves and surfaces. **International Journal of Computer Vision**, v. 13, n. 2, p. 119–152, 1994. ISSN 09205691.