Global pose estimation and tracking for RGB-D localization and 3D mapping

Fernando Israel Ireta Muñoz

UNIVERSITÉ CÔTE D'AZUR

ÉCOLE DOCTORALE
SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION

CONACYT
Consejo Nacional de Ciencia y Tecnología

THÈSE DE DOCTORAT

Estimation de pose globale et suivi pour la localisation RGB-D et cartographie 3D

Fernando Israel IRETA MUNOZ

Laboratoire CNRS / I3S

Présentée en vue de l'obtention du grade de docteur en SIGNAL,
SYSTÉMES ET TRAITEMENT DES IMAGES
d'Université Côte d'Azur
Dirigée par : Andrew I. Comport
Soutenue le : 04/04/2018

Devant le jury, composé de :

Pascal Vasseur, Mr, Université de Rouen
Mezouar Youcef, Mr, Institute Pascal
Jonghyuk Kim, Mr, Australian National University
Christian Laugier, Mr, INRIA
Jean-Bernard Hayet, Mr, CIMAT

DOCTORAL THESIS

Presented to

L'UNIVERSITÉ DE NICE SOPHIA ANTIPOLIS

To obtain

The title of : PhD

Mention : SIGNAL, SYSTEMS AND IMAGE PROCESSING

BY

FERNANDO ISRAEL IRETA MUÑOZ

Laboratory: CNRS-I3S, SIS Team

Doctoral School: EDSTIC Ecole doctorale des Sciences & Technologies
de l´Information et de la Communication

Thesis title:

Global Pose Estimation and Tracking for RGB-D Localization and 3D Mapping

4th April 2018

Jon	KIM	(Australian National University, Australia)	Rapporteur
Pascal	VASSEUR	(University of Rouen, France)	Rapporteur
Youcef	MEZOUAR	(Pascal Institute, CNRS, France)	Rapporteur
Christian	LAUGIER	(INRIA, France)	Examinateur
Jean-Bernard	HAYET	(CIMAT, Mexico)	Examinateur
Andrew Ian	COMPORT	(Laboratory I3S, CNRS, France)	Directeur de thèse

"The minute you think of giving up, think of the reason why you held on so long"

To my family...

Acknowledgments

First of all, I would like to thank to the support of my beloved family: my parents Fernando and Hilaria, my sisters Adriana, Alejandra and Lorena, my aunts Rosa and Margarita, my cousin Juan Simon, my brother-in-law Roberto and my nephews Fernanda, Carol y Mateo. All of them have encouraged me since the first day I came to France and they were following my academic and personal progress.

I would like to thank specially the support and guidance of Andrew I. Comport who supervised my progress and proposed the method that will be presented in this manuscript. Thanks for the expert advice and encouragement throughout this project.

I am specially grateful with all my colleagues and friends that I have meet during this PhD. Thanks for all the funny and sportive moments that have made my stage in France much more pleasant. You supported me greatly and you are always willing to help me.

This thesis would have been impossible without the funding of CONACYT scholarship, the institutions CONCYTEG and the Laboratory CNRS-I3S. In the same way, I am grateful also with various members of my family who supported me during my arrival in France and during the missions in this PhD. I know I owe you some money, so expect a payout soon.

Finally, but not less important, I would like to thank the members of the jury for having accepted to analyze and discuss the research done in this manuscript.

Abstract
(English version)

This thesis presents a detailed account of novel techniques for pose estimation by using both, color and depth information from RGB-D sensors. Since pose estimation simultaneously requires an environment map, 3D scene reconstruction will also be considered in this thesis. Localization and mapping has been extensively studied by the robotics and computer vision communities and it is widely employed in mobile robotics and autonomous systems for performing tasks such as tracking, dense 3D mapping and robust localization.

The central challenge of pose estimation lies in how to relate sensor measurements to the state of position and orientation. When a variety of sensors, which provide different information about the same data points, are available, the challenge then becomes part of how to best fuse acquired information at different times. In order to develop an effective algorithm to deal with these problems, a novel registration method named Point-to-hyperplane Iterative Closest Point will be introduced, analysed, compared and applied to pose estimation and key-frame mapping. The proposed method allows to jointly minimize different metric errors as a single measurement vector with n-dimensions without requiring a scaling factor to tune their importance during the minimization process.

Within the Point-to-hyperplane framework two main axes have been investigated. Firstly, the proposed method will be employed for performing visual odometry and 3D mapping. Based on actual experiments, it has been shown that the proposed method allows to accurately estimate the pose locally by increasing the domain of convergence and by speeding up the alignment. The invariance is mathematically proven and results in both, simulated and real environments, are provided. Secondly, a method is proposed for global localization for enabling place recognition and detection. This method involves using the point-to-hyperplane methods within a Branch-and-bound architecture to estimate the pose globally. Therefore, the proposed method has been combined with the Branch-and-bound algorithm to estimate the pose globally. Since Branch-and-bound strategies obtain rough alignments regardless of the initial position between frames, the Point-to-hyperplane can be used for refinement. It will be demonstrated that the bounds are better constrained when more dimensions are considered. This last approach is shown to be useful for solving mistracking problems and for obtaining globally consistent 3D maps. In a last part of the thesis and in order to demonstrate the proposed approaches and their performance, both visual SLAM and 3D mapping results are provided.

Resumé
(Version en Française)

Ce rapport de thèse présente une analyse détaillée de nouvelles techniques d´estimation de pose à partir des images de couleur et de profondeur provenant de capteurs RGB-D. Etant donné que l´estimation de la pose nécessite d´établir une cartographie en simultanée, la reconstruction 3D de la scène sera aussi étudié dans cette thèse. La localisation et la cartographie ont été largement étudiés par la communauté de robotique et de vision par ordinateur, et ces techniques ont aussi été largement employés pour la robotique mobile et les systèmes autonomes afin d´exécuter des tâches telles que le suivi de caméra, la reconstruction 3D dense ou encore la localisation robuste.

Le défi de l´estimation de pose réside dans la façon de relier les mesures des capteurs pour estimer l´état système en position et en orientation. Lorsqu´une multitude de capteurs fournisse différentes observations des mêmes variables, il devient alors complexe de fusionner au mieux ces informations acquises à des instants différents. De manière à développer un algorithme efficace pour traiter ces problèmes, une nouvelle méthode de recalage nommée Point-to-hyperplane sera introduite, analysée, comparée et appliquée à l´estimation de pose et à la cartographie basée sur des frames-clés. La méthode proposée permet de minimiser différentes métriques sous la forme d´un seul vecteur de mesure en n-dimensions, sans avoir besoin de définir un facteur d´échelle qui pondère l´influence de chaque terme durant la minimisation d´énergie.

Au sein du concept Point-to-hyperplane, deux lignes principales ont été examinées. Premièrement, la méthode proposée sera employée dans des applications d´odométrie visuelle et de cartographie 3D. Compte-tenu des résultats expérimentaux, il a été montré que la méthode proposée permet d´estimer la pose localement avec précision en augmentant le domaine et la vitesse de convergence. L´invariance est mathématiquement prouvée et des résultats sont fournis à la fois pour environnements réels et synthétiques. Deuxièmement, une méthode pour la localisation globale a été proposée qui adresse les problèmes de reconnaissance et de détection de lieux. Cette méthode s´appuie sur l´utilisation du Point-to-hyperplane combinée à une optimisation Branch-and-bound pour estimer la pose globalement. Etant donné que les stratégies de Branch-and-Bound permettent d´obtenir des alignements grossiers sans la nécessité d´avoir la pose initial entre les images, le Point-to-hyperplane peut être utiliser pour raffiner l´estimation. Il sera démontré que cette stratégie est mieux contrainte quand davantage de dimensions sont utilisées. Cette stratégie s´avère être utile pour résoudre les problèmes de désalignement et pour obtenir des cartes 3D globalement consistantes. Pour finir cette thèse et pour démontrer la performance des méthodes proposées, des résultats sur des applications de SLAM visuel et de cartographie 3D sont présentés.

Contents

CHAPTER 1

INTRODUCTION

Robotics is an important technology that has been employed for performing tasks that facilitate the daily life. It is a research domain that has been explored for many years and it has provided products ranging from simple toys to self-driving cars. Since the apparition of the term robot (from the Czech robota, which colloquially means hardwork), the automation of the robot's mobility has been extensively studied. The first generation of robots used were fixed and their working area was limited in a controlled environment. Eventually, the robots were upgraded with mechanical parts as wheels, legs, wings, propels, etc. in order to provide mobility through the ground, air or water. Finally, the technology advancement in electronics and computer science provided the possibility to perform robotic tasks automatically. By performing this, the concept of *mobile robotics* appeared as a research area.

Mobile robotics can mainly be decomposed into three main stages: Perception, localization and navigation, which provide the mobility to any robotic platform [79]. The perception stage mainly involves how the environment is perceived by the robot. This requires fusing the information together from various sensors that digitalize physical measurements from the environment. Once the measurements are acquired, the processing of those data can be used to generate a model that can be used as a database for localizing the robot. Therefore, the robot can be capable of replying the question "Where am I?" by giving its position w.r.t. an established reference. Hence, the robot can automatically or manually execute the mission commands by activating its actuators and navigate in its working area.

Computer vision is a field that emerged also as a research domain with strong connections between mathematics and computer science. Particularly, robot vision have appeared as a research area of computer vision which involves a combination of camera sensors and computer science to allow robotic platforms to process visual information from the world. This facilitates the processing and analysis of acquired digital images.

Nowadays, the availability of low-cost sensors based on RGB and Infrared cameras have provided the possibility to acquire color and depth information simultaneously at a considerable high framerate. These sensors have been employed to perform visual SLAM (Simultaneous Localization and Mapping) and they provide the capability to recognize the environment and to navigate in it. Furthermore, the localization stage based on RGB-D sensors is crucial and it leads to perform other tasks such as path planning or obstacle avoidance, which are very useful for mobile robotic platforms.

In this thesis, an alternative approach for performing visual SLAM and multi-view registration based on RGB-D sensors will be proposed. A novel strategy called Point-to-hyperplane ICP will be introduced, analyzed and compared w.r.t. classic approaches. Results obtained during this research will demonstrate that the method can obtain robust and accurate results for both, local and global RGB-D registration.

Motivation

During my research career in mobile robotics, I have been studying the stages shown in Figure 1.1 and for this particular PhD research I have been focused on the localization stage by building 3D maps and by globally estimating the position of a camera. Particularly, during this PhD project I have been working on improving the estimation of trajectories and generation of accurate 3D maps using RGB-D images. The application of RGB-D registration for the robotics field are meaningful since the computed 3D maps can be used as a database, which is updated in order to keep a valid representation of a dynamic environment, and it can guide the navigation of a robotic platform. The use of cameras for performing those tasks put a particular interest on me for exploring the computer vision research area, which has an extensive application in mobile robotics by performing visual odometry, 3D reconstruction and autonomous navigation. This has motivated me to explore in the future the cognition stage, where the prediction of the mission is given mainly by deep learning methods.

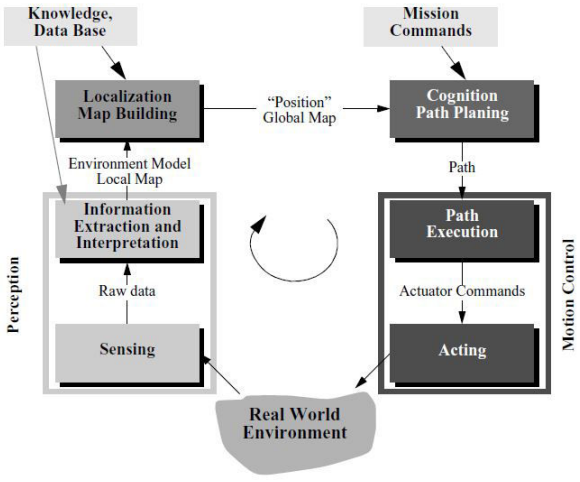


Figure 1.1: Control scheme for mobile robotic systems [79].

Funding

This thesis has been funded by the Mexican National Council for Science and Technology (Consejo Nacional de Ciencia y Tecnologia) **CONACYT** under grant **265807** between March 2014 and February 2018. Likewise, this work has been supported by the European H2020 project: COMANOID, Université Côte d´Azur and Centre National de la Recherche Scientifique (CNRS) in the Laboratory I3S (Laboratorie d´informatique, Signaux et Systémes de Sophia Antipolis).

Objectives

The objective of this PhD thesis is to investigate the general problem of visual mapping of complex 3D environments that evolve dynamically over time. This requires explicitly taking into account the variability of viewing conditions and content within a 3D geometric and photometric model of the environment. A central objective will be to investigate how to efficiently and accurately represent this model. The aim is to build models that remain usable over long periods of time that can be used for navigation tasks (localisation and path following) for autonomous systems or for people that require guidance. This research problem, which is already active in the vision community, has only just begun to be investigated by the robotics community via the new perspective of incremental SLAM.

This topic is in continuity with the research carried out by the ANM project from CNRS-I3S UNSA. The existing approach relies on a sensor-based representation that combines color and a depth maps within a topological graph of key-frames. This representation synthesizes information collected in a local area of space by an embedded acquisition system. The optimal representation of the global environment consists of a collection of visual key-frames that provide the necessary coverage of an operational area and each cloud is precisely referenced globally. A "pose" graph that links these frames together, in six degrees of freedom, also defines the domain which is reachable by an autonomous agent and that is potentially exploitable for navigation tasks in real time.

As part of this research, it is proposed to develop an approach to Life-long learning map-based representation by considering the following issues:

- How to optimally model a local dynamic environment containing non-rigid moving objects within the scene.
- How to compactly represent the information contained in this representation so as to maintain real-time interaction, learning, robustness, accuracy and stability over time (saliency map).
- How to exploit this representation in a navigation task in real time.

Contributions

The contributions given during the development of this thesis can be enclosed in various outstanding publications that are directly related with the proposed method here named Point-to-hyperplane, which perform RGB-D registration by fusing color and depth.

1. **Direct matching for improving image-based registration**, *IROS 2015* [39]. In this paper a method that perform a matching step on direct approaches is proposed. The method allows to increase the convergence domain and speed up the alignment whilst maintaining the robust and accurate properties of direct approaches. The proposed method is inspired from closest point matching in ICP, but instead of matching geometric points, the closest point is found in image space. A matching strategy, which is based on kd-tree (k-dimensional) approaches, is proposed based only on the intensities of the pixels without feature extraction, where the best match is decided by the closest points in intensity and image formation.

2. **Point-to-hyperplane RGB-D Pose Estimation: Fusing Photometric and Geometric Measurements**, *IROS 2016* [40]. The Point-to-hyperplane ICP method was proposed in this paper for incremental pose estimation by fusing color and depth measurements. A framework that allows to formulate the problem with a unique measurement vector and not to combine them in an ad-hoc manner has been proposed. In this paper, an invariance to a scale factor between different measurement types was experimentally observed. The proposed method was employed for performing visual odometry and 3D reconstruction for both, real and synthetic environments. This paper has been awarded as the best student paper finalist.

3. **A Proof that Fusing Measurements Using Point-to-Hyperplane Registration is Invariant to Relative Scale**, *MFI 2016* [41]. A mathematical proof of the invariance to any scale factor between measurements in the Point-to-hyperplane ICP is explained here. The method Point-to-hyperplane ICP is then generalized for considering higher dimensions and extended results are shown. This article has been awarded as the best student paper 2nd runner up.

4. **Point-to-hyperplane: Fusing Metric Measurements for Pose Estimation**, *Advanced Robotics Journal 2017* [43] In this Journal, previous articles have been presented together with an extended experimental results. Particularly, the method was carried on real-time visual SLAM and comparisons between classic hybrid approaches are shown. For the experimental part, well known benchmark sequences were employed for evaluating the robustness and accuracy of the Point-to-hyperplane ICP and higher dimensions were evaluated (3D Euclidean points + 3 RGB channels).

5. **Global Point-to-hyperplane ICP: Local and Global Pose Estimation by Fusing Color and Depth**, *MFI 2017* [42]. In this article, the Point-to-hyperplane ICP has been combined with the Branch and Bound strategy in order to estimate the 6DOF (degrees of freedom) pose parameters globally. Registration is performed by considering color and geometry at both, the matching and the error minimization stages. Results in real and synthetic environments demonstrate that the proposed method can improve global registration under challenging conditions such as partial overlapping and noisy datasets.

Structure of the thesis

This thesis will be presented in four main chapters and divided in two main contributions based on the proposed method:

1. **Introduction.** This chapter will provide the structure of the thesis. Along with it, the personal motivation and main objectives for this thesis will be presented as well as the acknowledgements.

2. **Theoretical foundations of pose estimation**. This chapter will introduce a general overview and the mathematical basis of camera pose estimation based on the Iterative Closest Points (ICP) method. The estimation of the pose will be categorized and presented in two main strategies:

   - Local pose estimation
   - Global pose estimation

   For the first, a common framework based on the Iterative Re-weighted Least Squares (IRLS) will be explained and particular strategies for improving the stages of the method will be introduced. For the second category, the working operation of the Branch and Bound (BnB) method will be particularly introduced here since it is employed in this thesis for estimate the pose globally. In order to provide a better survey, variants of global ICP based on feature extraction will be equally cited since they can estimate the pose globally by registering correspondences.

3. **RGB-D registration**. The second chapter of the thesis will introduce the mathematical basis about how to estimate the poses that relate a set of measurements obtained by moving image sensors. The underlying 3D representation will be introduced including a 3D keyframe graph and its relation with local 3D pointclouds. Commonly, 3D Euclidean pointclouds obtained at different views are not in correspondence. Therefore, pose estimation processes are employed to estimate the 6 degrees of freedom (DOF) pose vector that explains the alignment between the pointclouds. The proposed work will focus on one of the most common pose estimation frameworks for obtaining the pose vector locally and globally. Three main error functions for local registration will be explained in this chapter: depth-based, image-based and hybrid. Particularly, hybrid methods will be introduced and categorized depending on how the uncertainty between measurements is handled for simultaneous error minimization. It will be shown how they can perform better pose registration than just considering color-based or depth-based methods alone. Global registration methods will be introduced in this chapter. Particularly, Branch and Bound (BnB) methods that are refined with local ICP approaches will extensively used and detailed. In this chapter, it will be demonstrated how global approaches can estimate the pose regardless of the initial pose between sets of measurements.

4. **Local Point-to-hyperplane ICP**. The main contribution of the thesis will be explained in this chapter. Here a novel Point-to-hyperplane ICP method will be proposed. This is a method which allows to fuse different sensors measurements together (color and depth in the proposed approach) without using a tuning parameter to balance the contribution of the measurements during the minimization process. This is achieved by staking the measurements into a high-dimensional vector and performing Point-to-hyperplane minimization in this high dimensional space. Furthermore, the mathematical demonstration of the invariance to the tuning parameter will be introduced and demonstrated for n-dimensions. Results of the proposed method will be shown in this chapter for both, real and and simulated environments.

5. **Global Point-to-hyperplane ICP**. This chapter is concentrated in an extension of the previous chapter. With the plethora of devices and multitude of 3D maps available from a wide range of devices (telephones, tablets, robotic platforms, security cameras, ...), there is a need to exploit these 3D models acquired previously for the localization and current sensor device. The first part of the thesis has considered how to align two locally close images together. This part of the thesis will investigate the problem of global alignment between the current frame and a pre-existing map of the environment (i.e. no information is available about the poat the previous time instant). Finding the global correspondences between a new set of measurements and already acquired 3D maps can be a challenge. Methods that can achieve alignment for this case, are based on global convergence methods such as the well known BnB method. This method attempts to solve a the non-convex error function by exploring the rotational and translational space of solutions to align the image to the map. In this chapter the BnB method will extensively used and detailed. It will be shown how this method can be improved by incorporating the Point-to-hyperplane method proposed in the previous chapter of this thesis.

NOTATIONS

General notation:

a	scalar	$\overline{A}$	Homogeneous representation of A
$\mathbf{a}$	Vector	$f(\cdot)$	Transformation function
$\mathbf{A}$	Matrix	$\omega(\cdot)$	Warping function
$\mathcal{I}$	Image	$\mathbf{A}^\top$	Transpose of matrix $\mathbf{A}$

Mathematical notation:

$\mathbf{M}$	Measurements vector
$\mathbf{P}$	3D Euclidean Points
$\mathbf{I}$	Intensity
$\mathbf{p}$	Pixel coordinates
$\mathbf{N}$	Normals vector
$\mathbf{K}$	Intrinsic calibration matrix
$\widehat{\mathbf{N}}$	Normalized normal vector
$\mathbf{x}$	Pose parameter
$\boldsymbol{\omega}$	Angular velocity
$\boldsymbol{\upsilon}$	linear velocity
$\mathbf{T}(\mathbf{x})$	Homogeneous transformation matrix
$\mathbf{R}(\mathbf{x})$	Rotation matrix
$\mathbf{t}(\mathbf{x})$	Translation matrix
$\mathbf{J}$	Jacobian matrix
$\mathbf{J}_G$	Geometric Jacobian matrix
$\mathbf{J}_I$	Photometric Jacobian matrix
$\mathbf{W}$	Weight matrix
ρ	Weight scalar
$\mathbf{E}(\mathbf{x})$	Error function
$\overline{\mathbf{E}(\mathbf{T}(\mathbf{x}))}$	Bound error function
$\mathbf{e}$	Error vector
$\mathbf{e}_G$	Geometric error vector
$\mathbf{e}_I$	Intensity error vector
$\mathbf{e}_H$	Hybrid error vector
$\overline{\mathbf{e}(\mathbf{x})}$	Upper error bound
$\underline{\mathbf{e}(\mathbf{x})}$	Lower error bound
$\boldsymbol{\lambda}$	Tuning parameter
λ_G	Geometric scale parameter
λ_I	Photometric scale parameter
ψ_R	Uncertainty radius for rotation
ψ_t	Uncertainty radius for translation

CHAPTER 2

THEORETICAL FOUNDATIONS OF POSE ESTIMATION

2.1 Introduction

Pose estimation can be defined as the problem of determining the position and orientation of a sensor (or sensors) relative to the environment or vice-versa. Commonly, sets of measurements obtained at different times or at different viewpoints are not in correspondence and a pose estimation process is employed to align correspondences between them. A common example is shown in Figure 2.1, where it can be seen that a color image was taken in a different viewpoint of the same building and at a different time of the day.

Figure 2.1: View registration problem example. These images were taken at different viewpoints, times and with a different camera. The transformation that relates the acquired measurements in both scenes can be locally or globally estimated by iteratively minimizing over the error generated between corresponding measurements.

The correct and robust estimation of the pose is a very important task for robotics and computer vision, and is widely used to perform applications such as 3D tracking, odometry, localization and mapping. This chapter is aimed at introducing the mathematical basis about how the pose that relates a set of measurements obtained by a moving sensor can be estimated. The first section will define the homogeneous transform representation of the pose parameter, while the second will introduce the common framework to estimate the pose locally and globally.

2.2 Euclidean rigid transformation

Denoted by $SE(3)$, the special Euclidean group is the set of 3D rigid motions of an object. The condition that defines a rigid object is that the distance between any points is constant. If two sets $\mathbf{M}^1$ and $\mathbf{M}^2$ are considered, the distance between them must satisfy:

$$\left|\left|\mathbf{M}_i^1(t) - \mathbf{M}_i^2(t)\right|\right| = constant, \forall t \in \mathbb{R} \; and \; \forall i \tag{2.1}$$

Therefore, a rigid body motion is a family of transformations that describe how the coordinates of every point on the object change as a function of time t. Algebraically, an Euclidean group is a set of transformations $\mathbf{T}$, that involves a rotational motion $\mathbf{R}$ and a translational motion $\mathbf{t}$. The $SE(3)$ group relates the 3D coordinates $\mathbf{M_i} \in \mathbb{R}^3$ of a point in a world frame (reference) and its coordinates relative to a current frame along time t represented here as $\overline{\mathbf{M}_i^*}$ and $\mathbf{M}_i(t)$, respectively. This relation is shown in (2.2).

$$\mathbf{M}_i(t) = \mathbf{R}(t)\mathbf{M}_i + \mathbf{t}(t) \in \mathbb{R}^3 \tag{2.2}$$

This transformation can also be written as a homogeneous representation such that $\overline{\mathbf{M}_i^*}(t) = \overline{\mathbf{T}}(t)\overline{\mathbf{M}_i} \in \mathbb{R}^4$, where $\overline{\mathbf{T}}(t)$ is called the homogeneous matrix, that is written as:

$$\overline{\mathbf{T}}(t) = \begin{bmatrix} \mathbf{R}(t) & \mathbf{t}(t) \\ \mathbf{0}_3 & 1 \end{bmatrix} \in \mathbb{R}^{4\times4} \tag{2.3}$$

Therefore, the natural matrix representation of the special Euclidean group $SE(3)$ in homogeneous coordinates is:

$$SE(3) = \{\overline{\mathbf{T}}(t) = \begin{bmatrix} \mathbf{R}(t) & \mathbf{t}(t) \\ \mathbf{0}_3 & 1 \end{bmatrix} | \mathbf{R}(t) \in SO(3), \mathbf{t}(t) \in \mathbb{R}^{3\times1}\} \subset \mathbb{R}^{4\times4} \tag{2.4}$$

Another useful representation of the $SE(3)$ group, is the canonical exponential representation. By deriving (2.3) w.r.t. time t, the structure of the matrix $\dot{\mathbf{T}}(t)\mathbf{T}^{-1}(t)$ can be demonstrated in the following equation:

$$\dot{\mathbf{T}}(t)\mathbf{T}^{-1}(t) = \begin{bmatrix} \dot{\mathbf{R}}(t)\mathbf{R}^\top(t) & \dot{\mathbf{t}}(t) - \dot{\mathbf{R}}(t)\mathbf{R}^\top(t)\mathbf{t}(t) \\ 0 & 0 \end{bmatrix} \tag{2.5}$$

where the skew symmetric matrix $\dot{\mathbf{R}}(t)\mathbf{R}^{-1}(t)$ can be written as an operator $[\boldsymbol{\omega}]_\times(t)$ and a vector can be created as $\boldsymbol{v}(t) = \dot{\mathbf{t}}(t) - \dot{\mathbf{R}}(t)\mathbf{R}^T(t)\mathbf{t}(t) \in \mathbb{R}^3$.

Therefore, (2.5) can be rewritten as:

$$[\boldsymbol{\xi}]_{\wedge}(t) = \dot{\mathbf{T}}(t)\mathbf{T}^{-1}(t) = \begin{bmatrix} [\boldsymbol{\omega}]_{\times}(t) & \boldsymbol{v}(t) \\ 0 & 0 \end{bmatrix}, \hat{\xi}(t) \in \mathbb{R}^{4\times4} \tag{2.6}$$

where $[\boldsymbol{\xi}]_{\wedge}$ is called twist matrix operator. It is the tangent vector along the curve $\mathbf{T}(t)$ and it is used to approximate the curve $\boldsymbol{\xi}$ locally. The set of all twist matrices are denoted as:

$$se(3) = \{[\boldsymbol{\xi}]_{\wedge}(t) = \begin{bmatrix} [\boldsymbol{\omega}]_{\times}(t) & \boldsymbol{v}(t) \\ 0 & 0 \end{bmatrix} \mid [\boldsymbol{\omega}]_{\times}(t) \in \mathbb{R}^{3\times3}, \boldsymbol{v}(t) \in \mathbb{R}^{3}\} \subset \mathbb{R}^{4\times4} \tag{2.7}$$

where $se(3)$ is called the tangent space of the matrix group $SE(3)$.

For simplicity of notation, the linear and angular velocities will not be shown as a function of time, but as a matrix as: $[\boldsymbol{\omega}]_{\times} = [\boldsymbol{\omega}]_{\times}(t)$, $\boldsymbol{v} = \boldsymbol{v}(t)$ and $[\boldsymbol{\xi}]_{\wedge} = [\boldsymbol{\xi}]_{\wedge}(t)$. Furthermore, the pose matrices will be displayed as a function of the pose parameter $\mathbf{x} = [\boldsymbol{\omega}\ \boldsymbol{v}]^{\top} \in \mathbb{R}^6$ such as: $\mathbf{R}(\mathbf{x}) = \mathbf{R}(t)$, $\mathbf{t}(\mathbf{x}) = \mathbf{t}(t)$ and $\mathbf{T}(\mathbf{x}) = \mathbf{T}(t)$.

2.3 General overview of pose estimation

The view registration problem has been widely studied in the field of computer vision and it is especially applied in robotics for computing visual odometry, performing autonomous navigation and 3D reconstruction. One of the most common fundamental problems is estimating the pose that relates two sets of measurements obtained from the same scene at different viewpoints, different times or from different sensors (Figure 2.2). The general approaches presented here will attempt to generalize across different measurements types and then be instantiated by considering specific sensor modalities and estimation models.

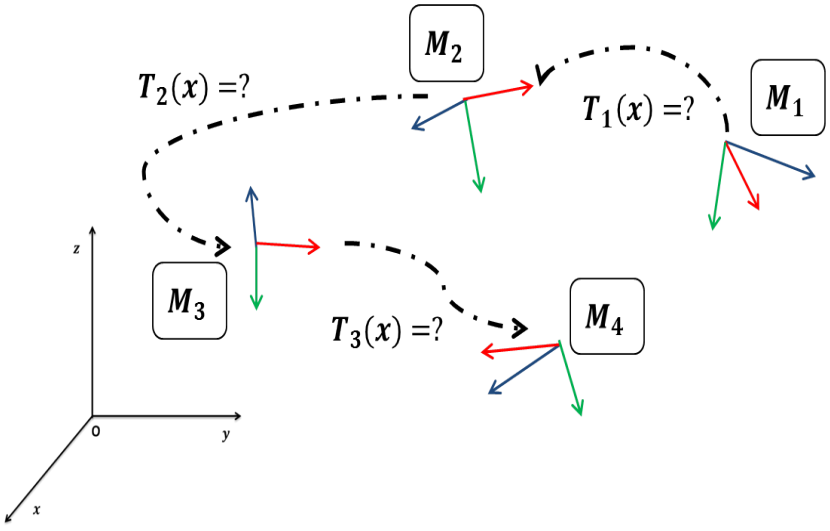


Figure 2.2: Unknown poses in the 3D world coordinates. The most common problem in view registration is estimating the pose $\mathbf{T}(\mathbf{x})$ that relates the set of measurements $\mathbf{M}_i$ obtained at different times.

Two main situations that arise when estimating the unknown 6 DOF (Degrees of freedom) pose parameter $\mathbf{x}$ will be analyzed. Getting in a particular situation will depend on the initial pose of the acquired datasets:

- If a large overlap is present between sets of measurements (small motion between acquisitions) or if the transformation $\mathbf{T}(\mathbf{x})$ is close to the solution, then the unknown pose can be estimated locally by iteratively transforming local correspondences until convergence.
- Sets of measurements that have large displacements in between and where is common to have a small overlap, can be aligned globally. Normally, these global methods perform an exhaustive search in the transformation space to find an optimal solution which minimizes the error between correspondences.

The aforementioned situations have been analyzed in how they minimize the error function. Depending in how the unknown pose parameter is obtained, the approaches can be categorized here as **local** and **global** approaches, respectively.

Generally, local and global approaches attempt to solve the following error function between corresponding measurements $\mathbf{M}_i^*$ and $\mathbf{M}_i$ by estimating the transformation $\mathbf{T}(\mathbf{x})$:

$$\mathbf{T}(\mathbf{x}) = \underset{\mathbf{T}(\mathbf{x})}{argmin} \sum_{i=1}^{N} ||\mathbf{M}_i^* - f\left(\mathbf{T}(\mathbf{x}), \mathbf{M}_i\right)||_{\mathbf{W}}^2 \in SE(3) \tag{2.8}$$

where N is the number of correspondences and $f\left(\mathbf{T}(\mathbf{x}), \mathbf{M}_i\right)$ is the function that transforms a set of measurements $\mathbf{M}_i$ by the transformation $\mathbf{T}(\mathbf{x})$. $\mathbf{W}$ is the weighting matrix obtained by robust estimation. The superscript $*$ will be used throughout this thesis to identify the set of measurements that were obtained first (reference dataset).

When the datasets are close enough, a local minimization as the well known Iterative Closest Point (ICP) [9] can be used for estimating the pose whilst for global minimization the Branch and Bound (BnB) algorithm and feature matching methods can guarantee convergence without considering an initial alignment between datasets. The main concept of both approaches can be summarized in two main steps:

- Estimate correspondences between datasets.
- Compute the transformation that minimizes the distance between corresponding points.

Often, a key difference between local and global approaches is that the steps shown above are performed in different order. If the rotation and translation are known, then the correspondences can be estimated. Otherwise, if correspondences are known then the rotational and translational motion can be estimated. In any case, the non-convex function (2.8) becomes solvable in its closed-form. It must be mentioned here that a local minima can be reached in both cases when wrong correspondences are found (vulnerability to outliers). Despite this, local approaches are faster than global approaches and they can obtain robust alignments if the initial transformation between datasets is close to the solution. On the other hand, when a global method is used only a rough

alignment can be obtained between frames due to the fact of an extensive exploration of the $SE(3)$ transformation space, which is prohibitively inefficient and memory consuming.

For the purposes of this thesis only a non-linear iterative pose estimation strategy will be considered along with robust pose estimation. However, other approaches could equally be considered. Therefore, the non-linear error (2.8) can be minimized by iteratively using a Gauss-Newton approach with increments given by:

$$\mathbf{x} = -(\mathbf{J}^\top \mathbf{W}\mathbf{J})^{-1}\mathbf{J}^\top \mathbf{W}\mathbf{E}(\mathbf{T}(\mathbf{x})) \in \mathbb{R}^6 \tag{2.9}$$

where $\mathbf{J}$ represents the stacked Jacobian matrices obtained by derivating the stacked error functions $\mathbf{E}(\mathbf{T}(\mathbf{x})) = [\mathbf{e}_1\ \mathbf{e}_2\ \cdots \mathbf{e}_n]^\top$. The Jacobian is obtained by partial derivation of the error function as:

$$\mathbf{J} = \frac{\partial \mathbf{E}(\mathbf{T}(\mathbf{x}))}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathbf{e}_1}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{e}_1}{\partial \mathbf{x}_6} \\ \vdots & \ddots & \vdots \\ \frac{\partial \mathbf{e}_n}{\partial \mathbf{x}_1} & \cdots & \frac{\partial \mathbf{e}_n}{\partial \mathbf{x}_6} \end{bmatrix} \in \mathbb{R}^{n\times 6} \tag{2.10}$$

The weight matrix $\mathbf{W}$ contains the stacked weights ρ_i associated with each set of coordinates obtained by M-estimation [36] as:

$$\mathbf{W} = \begin{bmatrix} \rho_1 & 0 & \cdots & 0 \\ 0 & \rho_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \rho_n \end{bmatrix} \in \mathbb{R}^{n\times n} \tag{2.11}$$

Often, M-estimation is performed separately for each different metric measurement.

The 6DOF pose $\mathbf{x}$ can be decomposed into rotational and translational components and it will be defined here as the homogeneous transformation matrix such as:

$$\mathbf{T}(\mathbf{x}) = (\mathbf{R}(\mathbf{x}), \mathbf{t}(\mathbf{x})) \in \mathbb{SE}(3) \tag{2.12}$$

which is the parametrization of the linear and angular velocity $\boldsymbol{\upsilon} = [\upsilon_x\ \upsilon_y\ \upsilon_z]^\top, \boldsymbol{\omega} = [\omega_x\ \omega_y\ \omega_z]^\top$, respectively. The relationship between both is given by the exponential map as:

$$\mathbf{T}(\mathbf{x}) = e^{[\mathbf{x}]\wedge} \in \mathbb{R}^{4\times 4} \tag{2.13}$$

where $[\cdot]_\wedge$ is the twist matrix operator presented in (2.6).

Furthermore, when n dimensions are considered a tuning parameter $\boldsymbol{\lambda}$ can be integrated in the error function as a matrix to deal with the uncertainty between different types of measurements, written in here as: $\mathbf{E}(\boldsymbol{\lambda}, \mathbf{T}(\mathbf{x}))$. This will be explained later on Section 3.5.1.

Since local methods share much similarity while estimating the pose, a pipeline common to all local approaches will be introduced in 2.4. On the other hand, methods that can estimate the pose globally without extracting features will be introduced in 2.5 and they can obtain the pose regardless of the initial pose.

2.4 Local pose estimation framework

A pipeline common to all local approaches will be introduced here. The pose estimation strategy common to many classic techniques uses a non-linear iteratively re-weighted least squares method (IRLS) which is an effective algorithm for solving L_p norm problems. The IRLS pipeline to solve (2.8) involves the stages shown in Figure 2.3. This method minimizes the error generated between corresponding measurements.

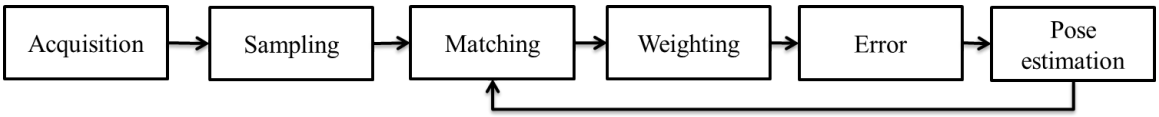


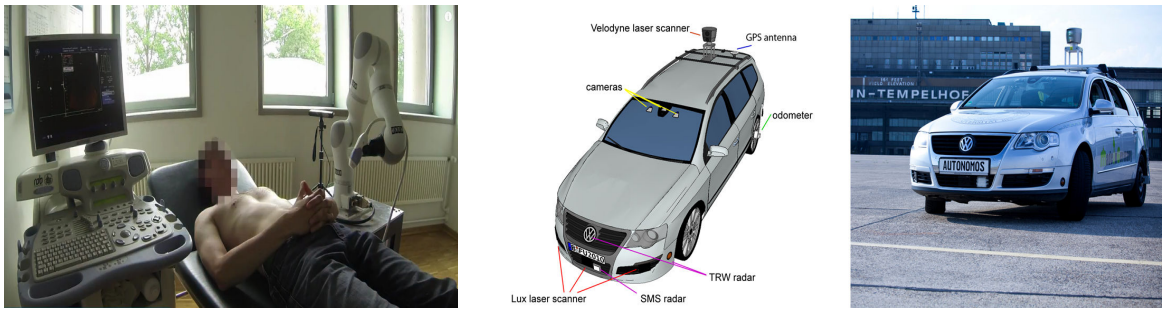
Figure 2.3: Stages of the IRLS framework.

The IRLS method can be improved by optimizing one or more stages mentioned above that can be even pre-computed (sampling and matching e.g.). For the acquisition for example, the synchronization of the sensors has been used for a best association of different types of measurements (e.g. an inertial sensor and a RGB camera working at different framerates). For subsampling, extraction of corresponding features in both datasets have been used to minimize the number of operations and memory consumption. The matching can be improved if more metric information is available for finding corresponding points while for the weighting the different influence functions on M-estimators have demonstrated robustness and accuracy while estimating the pose. Therefore, each stage will be briefly described in Sections 2.4.1 - 2.4.6. Essentially, local methods in the literature differ in how each stage is performed but here the interest is in how the matching and the error minimization are performed.

2.4.1 Acquisition

Variety of sensors provide a general set of measurements $\mathbf{M}$ that represent changes in the environment. The working principle of a sensor is to convert a physical measurement into a digital signal that can be processed by a computer. Sensors are often employed in on-board robotic platforms (e.g. a end-effector mounted camera on a robotic arm or sensors mounted in an autonomous vehicle as in Figure 2.4(b)), but they can be employed also externally, by surveying and controlling a robot like in visual servoing applications on ground robots as in Figure 2.4(a), where an external RGB-D sensor is employed to control the end effector of the robotic arm, which has an on-board ultrasound sensor.

Acquisition of measurements from different sensors can improve the representation of the environment, due to that acquired measurements from one sensor can be employed to complement information obtained by other sensors. The acquired measurements can contain depth, color, extracted features, temperature, etc. and they are often obtained simultaneously in order to have a consistency among data by synchronizing the acquisitions. Multiple measurements from the scene can be stacked in a n-vector $\mathbf{M}_i$ (n-dimensional), which is defined here as a vector that contains all the measurements obtained by the sensor (or sensors) at the i-th point.

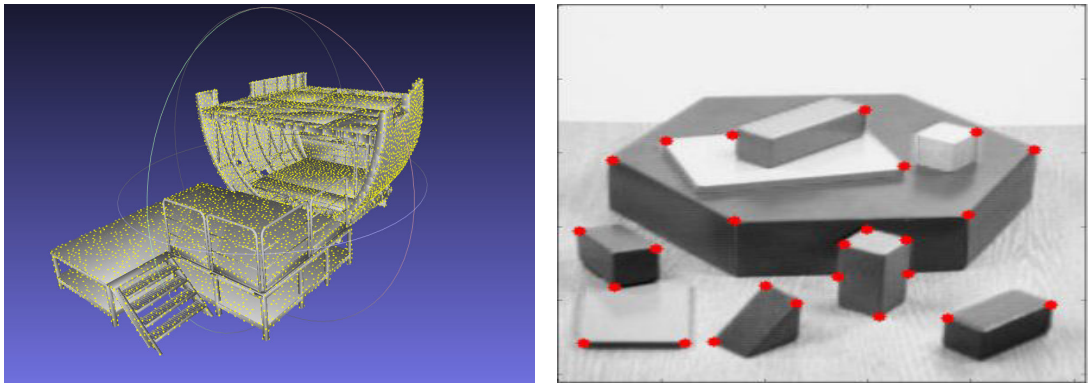


(a) (b)

Figure 2.4: Variety of sensors (such as encoders, GPS, radar and cameras) are employed to obtain a representation of the environment. (a) Automated system for 3D ultrasound [Institute of Robotics, University of Lubeck]. (b) MadeInGermany autonomous car [Intelligent Systems and Robotics, Freie Universitat Berlin].

2.4.2 Sampling

Sampling the set of measurements is often performed to accelerate the alignment between two datasets. When the datasets are large, a sampling stage may help in reducing the total amount of characteristic points to handle, and consequently the number of operations is less memory consuming. This is particularly the case for feature based methods which require feature extraction algorithms in order to have a more manageable dataset and they get a rid of measurements. Several strategies can be performed and they can categorized depending on how points are selected [73]. The selection can be done by randomly or uniformly by selecting keypoints based on their metric properties at each iteration or by pre-selecting before the iterative loop. Furthermore, the selection of keypoints can be performed manually or automatically and they can be represented here as $\mathrm{M}^s \subset \mathrm{M}$. It can be mentioned here that dense registration is a concept that is employed when all available measurements are used in the framework.



(a) 40000/99346 Points (b) 25/62 Corners

Figure 2.5: Subsampling example of a dataset. (a) By using strategies as Poisson disk 3D subsampling [3D model data, Airbus] or by (b) Harris corner detector, the number of photometric data to be processed is reduced as well as the computational cost. The ratio is presented as $number\ of\ samples/total\ number\ of\ features$.

Two examples are given in Figure 2.5, where keypoints are extracted from the entire 3D model and the entire image (Figures 2.5(a) and 2.5(b)). In the first case, a uniform sampling over all the model is used in order to reduce the total number of points (from 99346 to 40000 3D points). In the second case, only 25 corners are randomly selected from the image by using a Harris corner detector algorithm.

2.4.3 Matching

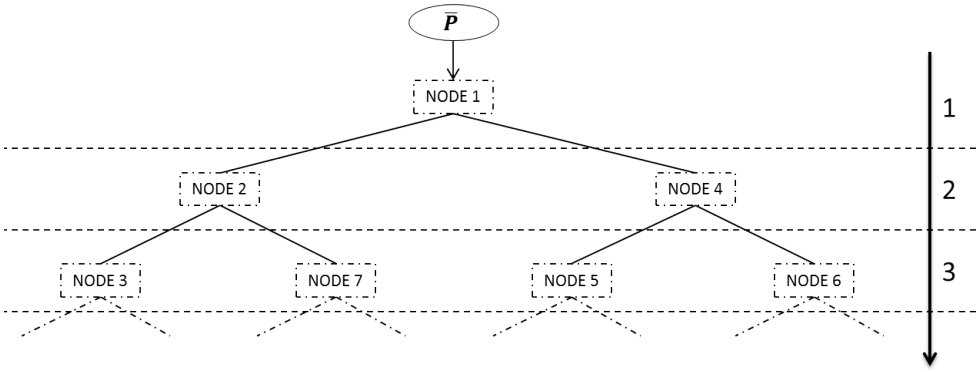
A matching step is performed to obtain correspondences so that an objective error function may be defined if it is necessary. Without this step in the pose estimation algorithms, the distance is computed for every point and its performance depends on the dimension of the measurement vectors and on good correspondences. The matches $\mathbf{M}^m$ can be estimated using many different approaches that search for the closest points such as interpolating values [26], or performing feature matching using various correlation strategies [51]. In non-linear estimation approaches, matching may even be performed at each iteration of the estimation process. It should be noted that the matching step, depending on the strategy employed, is often the most computationally expensive stage in the estimation model.

Alternatively, a classic strategy to obtain initial point matching is proposed in [8], where the search for the closest points are performed using *k*d-trees (*k*-dimensional). They are a useful data structure for nearest neighbor searches because they allow to accelerate the point matching if more information about the point is available. This approach is used in the survey [66], where a classic *k*d-tree search is used to find and compute the closest points in the first iteration, and an optimized method called cached *k*d-tree search is used for the following iterations, which reduces significantly the number of matching operations. It will be shown later on Chapter 4 how the kd-tree can improve the closest points finding when more dimensions are considered and how they can speed up the alignment if the matching is performed in the first iteration only. For the purposes of this thesis, a *k*d-tree strategy will be employed and introduced below.

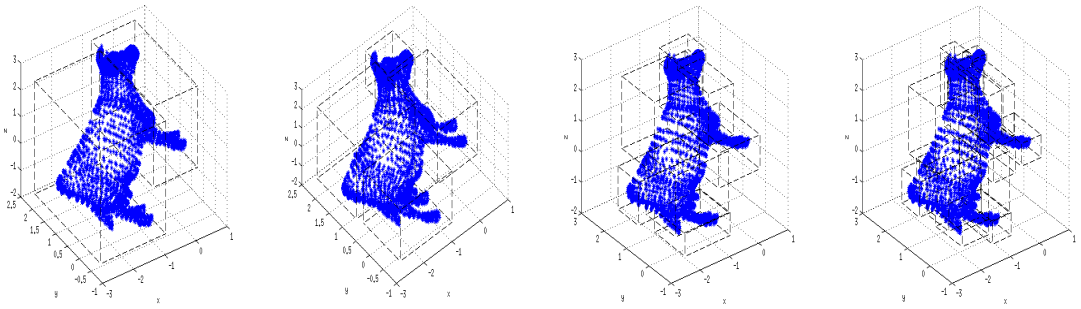
*k*d - trees are a generalization of binary search trees [8] and they provide an acceleration for closest points finding in high dimension vectors. Every node represents a partition of a point set in two successor nodes where the root represents all the measurements in a dataset $\mathbf{P}$ and the leaves (so-called buckets) contain a defined number of $\mathbf{P}_i \subset \mathbf{P}$. The median value of the data set is the factor that determines the bucket where each point belongs and it depends on the chosen coordinate axis parameter to split the dataset. In case of 3D Euclidean points e.g., the coordinates X, Y and Z can be chosen in a specific order for each level of the kd-tree to split the dataset. In Figures 2.6(b), 2.6(c), 2.6(d) and 2.6(e), the dataset was recursively divided by considering the median in the X, Y, Z and X axes, respectively.

In the case of a cloud of points with 2D or 3D points representing the geometry of the object, the median value is computed respectively for each axis x, y for 2D points and x, y & z for 3D points. The coordinate axis should be different in each level of the *k*d - tree in order to create a balanced *k*d - tree. The strategy ensures that every point in the cloud has the same probability to be placed in any leaf of the *k*d - tree in which each leaf node is about the same distance from the root.

Exact searching in *k*d-trees is too costly for many applications. This has generated interest in approximate nearest neighbors search algorithms such as the kNN algorithm [64] (See figure 2.7).



(a) *k*d-tree construction



(b) Level 1 (c) Level 2 (d) Level 3 (e) Level 4

Figure 2.6: kd - tree construction and example on 3D points. The method divides a dataset into nodes with a determined number of points per node, based on the median of each coordinate.

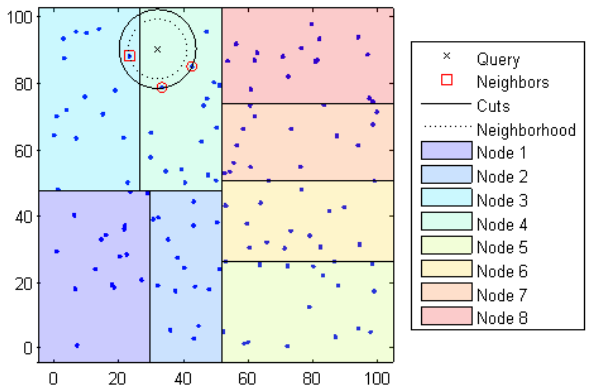


Figure 2.7: Example of a kd - tree searching for 2D points [kNN algorithm, Matlab]. The K-Nearest Neighbours (kNN) of a query point are those who are found in an established search radius.

Generally, the kNN searching algorithm performs the following steps:

1. Determine the node to which the query point belongs
2. Find the closest k points within that node and their distance to the query point.
3. Choose all other nodes having any area that is within the same distance, in any direction, from the query point to the k-th closest point.
4. Search nodes within that range for any points closer to the query points.

Various parameters can be modified to improve the efficiency of the kNN searching, such as the maximum number of points contained in a node, the length of the search radius, the distance measure (Euclidean, Mahalanobis, Hamming,etc.) and the selection of the order to compute the median per level of the *k*d-tree. In this particular case, FLANN (Fast Library for Approximate Nearest Neighbors) [64, 63] is a library for performing fast approximate nearest neighbour searches in high dimensional spaces (An example is shown in Figure 2.8). It contains a collection of algorithms that gives the best closest point matches using a system that automatically chooses the best algorithm and optimum parameters depending on the strategy employed. This library will be used in the experimental part of this thesis.

(a) True matches (b) 3 Dimensions (c) 4 Dimensions (d) 6 Dimensions

Figure 2.8: Example of matched points by using a multidimensional kd-tree (FLANN library) by considering different features dimensions, where the correspondences are shown in green (a) Exact correspondences when two datasets are aligned. A random transform is applied to the current dataset and the nearest neighbours are found by considering (b) 3D geometric points, (c) Geometry + Intensity and (d) Geometry + Color. It can be seen that more true correspondences can be found if higher dimensions are considered, however it requires more computational time.

2.4.4 Weighting

While acquiring sets of measurements, incorrect data can be registered and they might not have a correspondence in other datasets (partial overlapping between sets). In statistics, the term outliers is used to identify these points that cannot be paired or are wrongly paired. Generally, outliers correspond to occlusions, illumination changes, noise in images, etc. and they are an important source of error in the matching step, reducing notably the performance of the method.

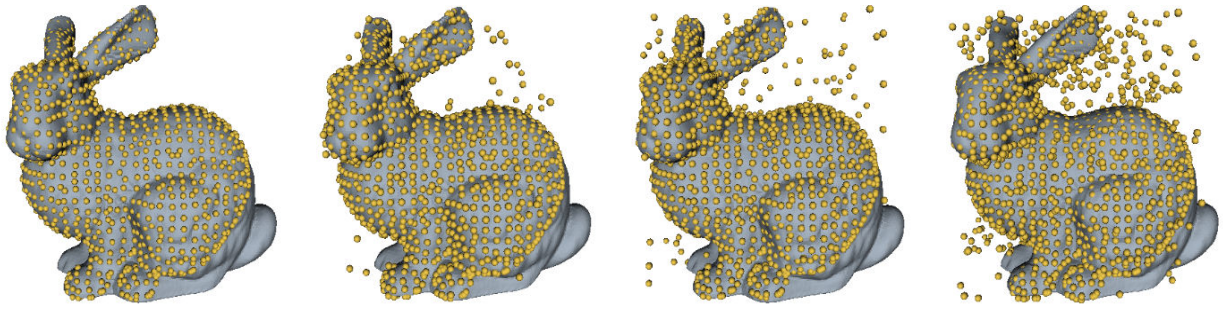


Figure 2.9: Sampled model set with 0,10, 20 and 50% of outliers [69]. The reference model is shown in grey and the noisy sampled dataset in yellow. Robust methods can obtain an accurate alignment by rejecting outliers.

In order to make the error function more robust, outliers are indentified and removed from the list of matches. For this purpose, the weights ρ_i are scalars that are stacked in the error function to increase or decrease their influence in the error function as:

$$\mathbf{E}(\mathbf{x}) = \sum_{i=1}^{N} \rho_i \left(\mathbf{e}_i\right) \tag{2.14}$$

Robust techniques such as the least median squares (LMS) [56], least trimmed squares (LTS) [17], random sample and consensus (RANSAC) [22], the Hough transform or M-estimation [36] are extensively used to reject outliers (See Figure 2.10). Therefore, a brief introduction of these strategies will be shown in the following.

Least median squares

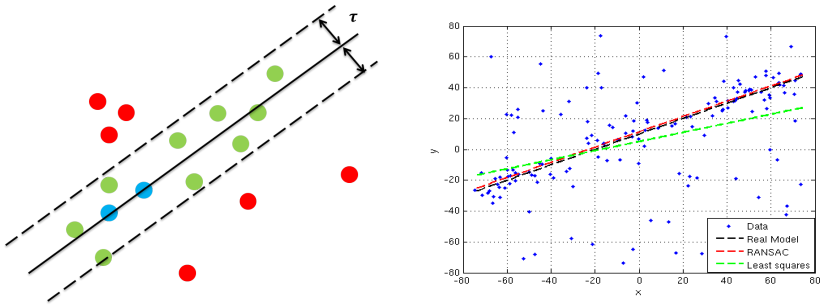
A robust technique used to reject outliers is LMS estimation, which selects a sample from the dataset and estimates a solution based on the point pairs by solving the non-linear error function (2.8) which is assumed to have a Gaussian distribution with zero mean. The median is efficient at discarding strongly deviating outliers, therefore, the LMS and LTS replaces the mean by the median of the residuals as:

$$min \left(\underset{i}{med} \, ||\mathbf{e}_i||_2\right) \tag{2.15}$$

The main difference between LTS and LMS is that LMS finds the median without ordering squared residuals. LTS identifies the data points with the largest residuals to be omitted and it has demonstrated a better convergence and a smoother objective function, which makes it more suitable for iterative methods. A comparative survey between both strategies has been done in [17].

RANSAC

The random sample consensus (RANSAC) is an iterative method for robust fitting of models in the presence of a large number of outliers (more than 50%). Its implementation is very simple since it maximizes the number of data within a defined distance threshold τ as $\mathbf{e}_i^2 \leq \tau^2$. If the error is bigger, then the error bound is considered as an outlier.



(a) (b)

Figure 2.10: Robust methods fitting a line with a large number of outliers. (a) 2 points are randomly selected (blue) in the dataset to fit a line model. Points that have a larger distance than an established threshold are considered as outliers (red). (b) Comparisons between Least squares and RANSAC model fitting. It is clearly seen that that the best fit of least squares is away from the true model line.

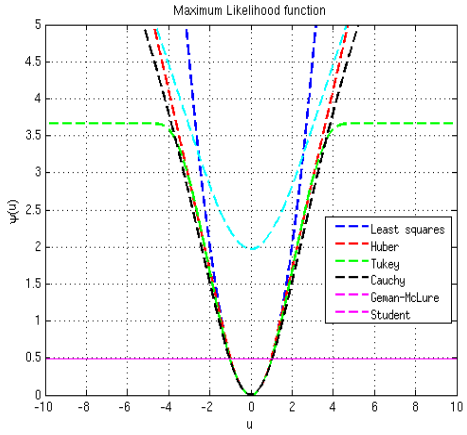
As a randomized algorithm, RANSAC is tested as follows:

1. A model is fitted to the randomly sampled subset (At least 2 points in the subset).
2. Test the model against all the other candidate pairs and determine the set of inliers within a distance τ of the model.
3. Repeat from 1.

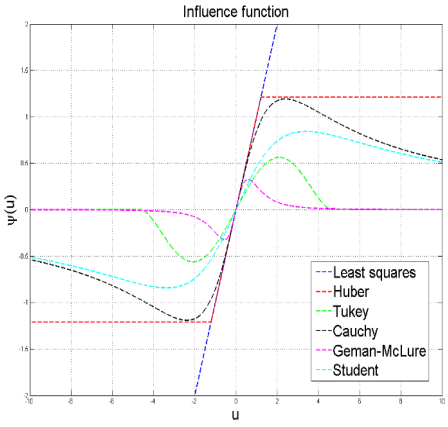
The procedure terminates when the percentage of inliers is below a certain threshold or after N trials.

M-estimators

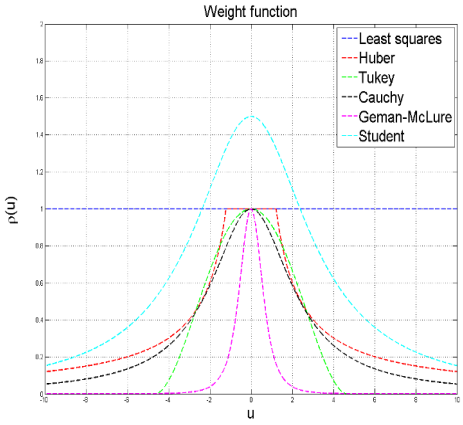
For more complex datasets, a much better alternative is to use robust M-estimators, which are discussed in terms of their influence function as $\psi_i = \frac{d\rho(\mathbf{u})}{du} \rightarrow \rho_i(\mathbf{u}, \sigma)$, where $u_i = e_i - median(e_i)$ is the residual error and $\sigma = 1.4826\ median|\mathbf{u}|$ is the standard deviation that smoothly refines the convergence. M-estimators reduce the effect of dynamic outliers as illumination changes or moving objects, however, they do not completely eliminate the effect of large outliers. The residuals can be considered as normally distributed where only least squares and Huber functions guarantee unique solutions (See Table 2.1 and Figure 2.11). A robust alternative distribution to the normal distribution is the t-distribution (student's distribution), which belongs to the family of elliptical distributions [57]. The Student distribution [48] has useful applications in robust statistical analysis and it is suited to model data distributions that covers outliers with a low probability, which is the main difference with a normal distribution. However, the estimation of its degrees of freedom v along with the other parameters, can become no longer locally robust. Therefore, the parameter v is usually assumed to be known and considered as a tuning constant. Decreasing the tuning constant increases the weight assigned to large residuals and by increasing v the distribution can converge to a normal distribution. In fact, a t-distribution coincides with the Cauchy distribution with one degree of freedom.



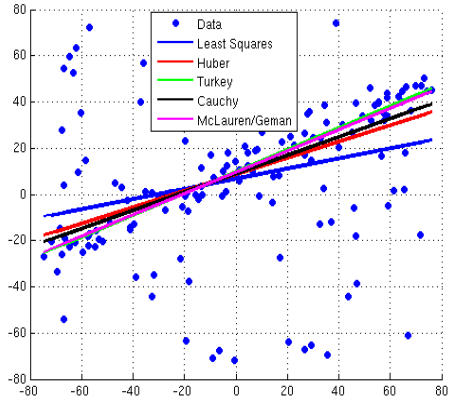
(a) Maximum Likelihood functions



(b) Influence functions



(c) Weight functions



(d)

Figure 2.11: M-estimators robust functions. (d) Outliers rejection example using M-estimators.

Table 2.1: M-estimator functions. The estimators require a sensible estimate of the scale σ, which is supplied at each iteration of the method as $\sigma = 1.4826 median|\mathbf{u}|$, where $u_i = e_i - median(e_i)$. The proportionality factors $b = 4.6851$, $c = 2.3849$ and $a = 1.2107$ ensures 95% of efficiency in the case of Gaussian noise [3].

Function	**Influence function** $\psi(u) = \int \rho(u)du$	**Weight function** $\rho(u)$
Least squares	u	1
Huber	$\begin{matrix} u & \|u\| \leq a \\ a\frac{u}{\|u\|} & \|u\| > a \end{matrix}$	$\begin{matrix} u & 1 \leq a \\ \frac{a}{\|u\|} & \|u\| > a \end{matrix}$
Tukey	$\begin{matrix} -\frac{u}{\sigma}\left(1-\left(\frac{u}{b}\right)^2\right)^2 & \|u\| \leq b \\ 0 & \|u\| \leq b \end{matrix}$	$\begin{matrix} \left(1-\left(\frac{u}{b}\right)^2\right)^2 & \|u\| \leq b \\ 0 & \|u\| \leq b \end{matrix}$
Cauchy	$\frac{u}{\left(1+\left(\frac{u}{c}\right)^2\right)}$	$\frac{1}{\left(1+\left(\frac{u}{c}\right)^2\right)}$
Geman-McLure	$\frac{u}{((1+u^2)^2)}$	$\frac{1}{((1+u^2)^2)}$
Student $v \rightarrow$DOF	$\frac{u}{\left(v+\left(\frac{u}{c}\right)^2\right)}$	$\frac{v+1}{v+\left(\frac{u}{c}\right)^2}$

2.4.5 Minimization

This step relies on the definition of an error metric calculated from the association of features by using an error model. One approach is to design an error function based on the sensor measurements that depends on the unknown parameters so that it can be used to estimate the pose. The error function presented in (2.8) is generated by considering the difference between correspondences of two datasets and it can be minimized by considering different distance metrics. In particular, the L_2 norm is used to estimate the Euclidean distance between two n-dimensional points.

$$\begin{aligned} ||\mathbf{e}_i||_2^2 &= \sqrt{\sum_{i=1}^{n} ||\mathbf{M}_i^* - \mathbf{M}_i||_2^2} \\ &= \sqrt{\left([\mathbf{M}_i^*]_1 - [\mathbf{M}_i]_1\right)^2 + \left([\mathbf{M}_i^*]_2 - [\mathbf{M}_i]_2\right)^2 + \cdots + \left([\mathbf{M}_i^*]_n - [\mathbf{M}_i]_n\right)^2} \end{aligned} \tag{2.16}$$

For purposes of this thesis, the sum of least squared distance (SSD) will be employed as the objective function:

$$\mathbf{E}(\mathbf{x}) = \sum_{i=1}^{N} ||\mathbf{e}_i||_2^2 \tag{2.17}$$

where $\mathbf{x}$ is the unknown pose parameter, and $\mathbf{e}_i$ is the residual error at the i-th measurement of the N correspondences.

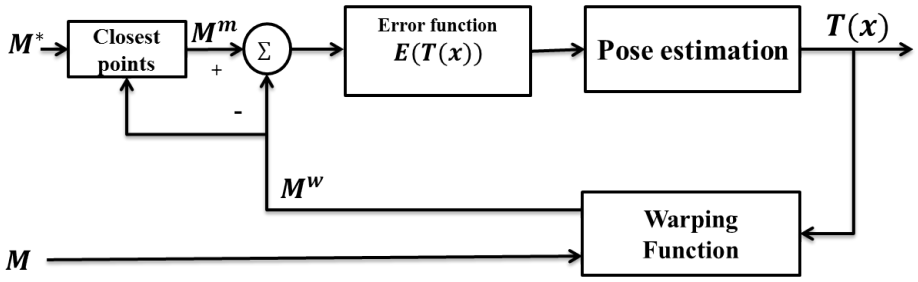


Figure 2.12: Pose estimation pipeline. The pose $\mathbf{T}(\mathbf{x})$ is updated at each iteration until convergence.

2.4.6 Tracking

The tracking problem will essentially be considered as a pose estimation problem which is related directly with the minimization of the error function. The pose estimation $\mathbf{T}(\mathbf{x})$ is computed at each iteration (See Figure 2.12) and is updated incrementally as $\widehat{\mathbf{T}} \leftarrow \widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})$ that solves the following equation:

$$\mathbf{T}(\mathbf{x}) = \underset{\mathbf{T}(\mathbf{x})}{argmin} \sum_{i=1}^{N} ||\rho_i\left(\mathbf{e}_i\right)||_2^2 \tag{2.18}$$

Local methods based on the IRLS framework can obtain faster alignments between datasets when the transformation between them is close to the solution. Otherwise, local approaches can converge to an incorrect pose. There are two criteria that can be employed to stop the tracking, the first one is by comparing the error function w.r.t. an established minimum threshold and the second one is by assigning a maximum number of iterations. In any case, the error function is solved in its closed-form and robustly minimized with an outliers rejection strategy.

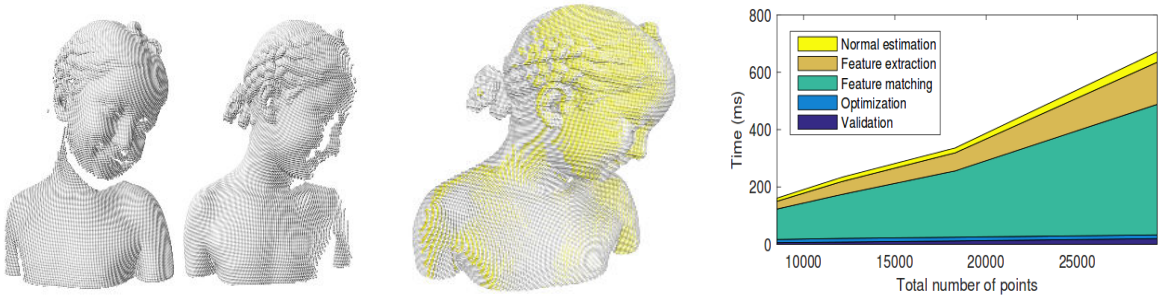
Local methods, however, do not guarantee convergence when datasets are not close enough to the solution and they are susceptible to get trapped into local minima. In the following section, methods that can guarantee convergence while avoiding the local minima issue will be introduced. These methods can maintain the main advantages of the aforementioned stages and they do not depend on the initial pose. In the literature, the methods are often referred as global approaches.

2.5 Global pose estimation framework

Global pose estimation was introduced to obtain a coarse alignment between scan pairs in an arbitrary initial positions. Most global alignment methods in the literature rely on point-pair correspondences, which can require manual or automatic selection of good correspondences and a robust estimator to reject outliers. This is not too different from local approaches, and feature-based global approaches can be seen as a local approach with an extended convergence rate that focuses in the matching stage [92, 35, 24, 54, 27, 52, 6, 12, 74, 2]. If good matches are found, then the error function (2.8) can be solved in few steps by minimizing over the point-pair correspondences and it can be validated in the entire datasets. However, feature extraction can demand a high computational cost, local minima still can be reached if inconsistent correspondences are found and its performance drops with noisy data.

One of the simplest global approaches involves shifting the centroids of the pointclouds to the origin of the coordinate system and statistical methods such as the Principal Component Analysis (PCA) algorithm can estimate an initial orientation by bringing out strong patterns in a dataset. The PCA algorithm has limitations for the 3D registration, such as 180 degrees ambiguity or eigenvalues similar in value (axis may switch). Additionally, PCA requires a high overlap between datasets and noisy values can change the direction of the estimated principal axes.

Global registration methods that pre-select correspondences differ in how the matches are estimated and they are often refined by a local method such as Point-to-point ICP. In [92], the matching stage is performed by using the Fast Point Feature Histogram (FPFH) which is a method that obtain a faster alignment by aligning only good pre-computed correspondences in the iterative loop without using a local approach for refinement.

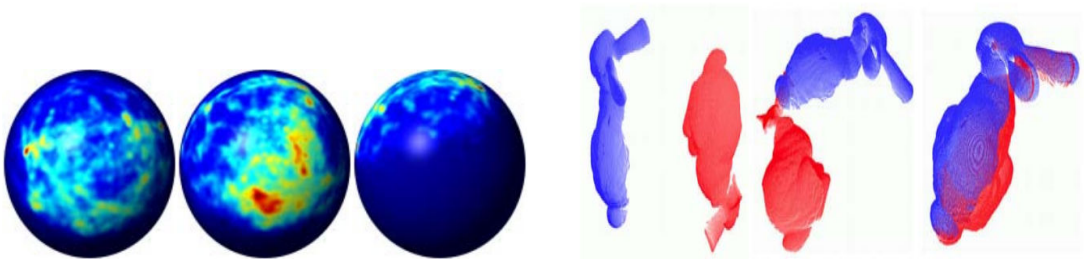


(a) Input datasets (b) Result (c) Elapsed time

Figure 2.13: Example of global registration by extracting features [92]. It can be noted in the graph that feature extraction, estimation of the normals and feature matching are the most computationally demanding stages which are lineal with the number of points.

Approaches that use both, global and local registration with extracted features, can be found in [35, 24, 54] and an application for RGB-D registration can be found in [27]. In [35] the method performs registration by finding a so-called model graph, which contains a connection between correct matches. A global approach named Super 4PCS [59] uses a sub-sampling type strategy to extract coplanar 4-point sets, which are assumed to remain invariant under rigid motion. The 4-point sets are created in all scan pairs and the algorithm finds all equivalent 4-points in between.

An approach based on the Branch-and-bound (BnB) method can be found in [24]. The method is improved by finding potential corresponding points based on performing integral operations on the underlying shape. In [54], the transformation is globally estimated by maximizing the correlation between extended Gaussian images (EGIs) in the Fourier domain. The method uses the convolution of the spherical harmonics of the EGIs and the rotational Fourier transform as features to estimate rotation as is shown in Figure 2.14. The translation is found similarly by employing the fast Fourier transform.



(a) Estimated EGIs (b) Pair of scans

Figure 2.14: Example of aligning two scans by considering the Extended Gaussian Images (EGIs) [54] (a) left: EGI of blue pointcloud, middle: EGI of red pointcloud and right: EGI of the aligned dataset). The pointcloud presented in (b) left is the initial pose between datasets, in the middle is presented an erroneous alignment by correlating two orientation histograms and in the right the alignment between datasets is presented.

A RGB-D application is found in [27], where the global estimation is found by minimizing the correspondences of extracted linear edges along creases and contours in depth images. As a side note, well known feature-based strategies in image processing can be equally used to extract geometric features from the scene and match them in other similar frame. Methods such as SIFT (Scale-Invariant Feature Transform) [52], SURF (Speeded Up Robust Features) [6], BRIEF (Binary Robust Independent Elementary Features) [12], ORB (Oriented FAST and Rotated BRIEF) [74] or A-KAZE [2] have been widely used for this purpose and their performance have been recently compared in [18].

On the other hand, the non-convex function (2.8) can be also solved by exploring the transformation space $\mathbf{T}(\mathbf{x})$ and estimating an optimal solution that maximizes the quality of match correspondences. Methods that do not require any feature detection, such as the Branch and Bound algorithm [47, 49] (See Section 2.5.1) or stochastic optimizations [69, 70], can be employed to estimate the pose by exploring and sub-dividing all the rotational and translational space.

Various methods like [30, 10, 67] consider pure rotation for the BnB method since the translation is known apriori or the clouds have sufficient overlap to be estimated by a local approach. In [67], the rotation space is sub-divided based on its quaternion representation and the translation is estimated by variants of local approaches [34]. An optimized version of [30] has been presented in [10]. Both strategies use the axis-angle representation to parametrize a solid radius-π sphere for rotations. The BnB method sub-divides the sphere by using an octree data structure, where the optimal global solution is found by bounding the generated subsets.

Strategies such as [90, 13, 83] have extended the exploration of the transformation space by subdividing the rotational and translational space together. Basically, these methods differ in: 1) how the $SE(3)$ space is sub-divided (branching) and 2) how the upper and lower bounds are computed. As in [30], the Go-ICP [90], GOGMA [13] and Go-PAC [14] algorithms propose an angle-axis representation to sub-divide the $SO(3)$ space, where an inner BnB search structure is included to explore the translational space. Whilst Go-ICP minimizes the L_2 norm of the Point-to-point residual error, GOGMA uses a Gaussian mixture alignment (GMA) and Go-PAC employs cardinality maximization to the simultaneous pose and correspondence problem, which is less sensitive to partial overlap. All methods estimate the uncertainty radius for rotation and translation, which is the key for estimating the lower bounds of the error function. The computational cost is linear with the size of datasets and it is improved by a subsampling stage. Recently, a more efficient method has been proposed in [83]. The method follows the same pipeline of minimizing the L_2 norm of the residual errors, but the BnB exploration decouples the rotational and translational space via the use of surface normals. The distribution of the surface normals is modeled as a von-Mises-Fisher mixture model (vMF-MM) for rotation and as a Gaussian Mixture Model for translation.

2.5.1 Branch and Bound algorithm

The Branch and Bound algorithm (BnB) is used to solve non-convex problems by searching the complete space of solutions for the best solution. Variants of the BnB method have been employed for 3D registration. The BnB method has three main components: Partition of the solution space, bound calculation and selection of the subset to process (See Figure 2.15). The sequence of these components can vary according to the strategy chosen for selecting the subset to process. If a subspace cannot contain the optimal solution, then it is discarded, otherwise it is kept to be processed.

In order to provide a generalized framework for pose estimation using the BnB algorithm, the following stages can be established:

1. Subdivide the transformation space into subspaces.
2. Transform the current measurements vector and estimate the correspondences w.r.t. the reference measurements vector for each subspace.
3. Evaluate the upper-bounds and lower-bounds for all subspaces of the queue.
4. Discard subspaces from the queue by a searching strategy, which selects the next subspace to process.
5. Repeat 2) if a branching is performed before bounding the error, or 1) if a branching is performed after bounding the error, until convergence.

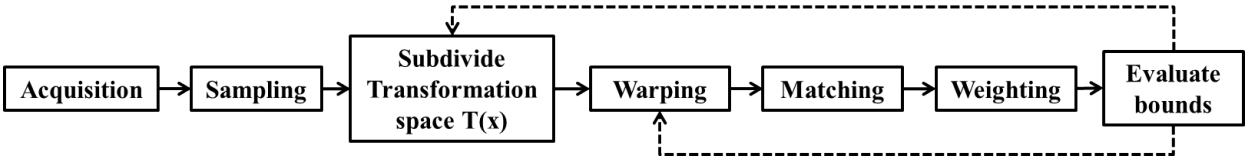


Figure 2.15: Global framework for pose estimation using the Branch and Bound strategy.

The performance of the BnB method depends on the strategy that is employed at each stage. For the branching, various methods that subdivide the transformation space into two or more subspaces have been proposed. Strategies such as well known octrees, bisecting along each axis or by modeling surface normals are the most used for this purpose. Calculating the bound may be one of the most demanding part of the BnB algorithm in order to keep the size of the search as small as possible. Finally, the selection of a good search strategy can reduce the memory consumption and accelerate the optimal solution finding. Each stage will be explained in the following.

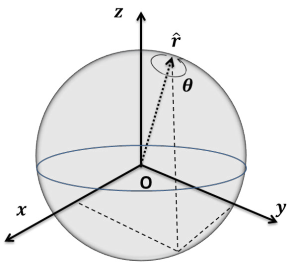
Branching methods

All branching rules in the context of BnB can be seen as a subdivision of the search space through the addition of constraints. Depending on the order of the branching and the bounding, the strategies are so-called eager or lazy BnB. The difference is that eager BnB subdivides the transformation space as soon as new subspaces are available (Branching is performed first), whilst a lazy BnB subdivides the $SE(3)$ subspace only if necessary (Bounds are evaluated first).

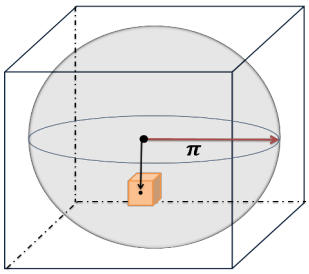
Similarly to [90] and [13], an octree structure can be employed to subdivide the rotation and translation space, which is parametrized using the *axis-angle* representation as a vector $\mathbf{r} = \theta\hat{\mathbf{r}} \in \mathbb{R}^3$ where $\hat{\mathbf{r}} \in \mathbb{R}^3$ is the direction of the axis and θ is the angle (See Figure 2.16). The strategy guarantees a rough alignment between datasets, where the alignment can be performed better if small sub-sets are created.

Another strategy to perform branching is by subdividing the rotational space in its spherical coordinates along each axis into spherical boxes of equal volume. This method has been used in [70] (Figure 2.17), where the translational space has been separated into smaller boxes of the same size by dividing along the x, y and z axes. Recently, an optimized branching method has been performed in [83] by considering the quaternion representation of rotations. A tessellation of the rotational space is made by considering a 4D tetrahedra, which follows the same principles of subdividing an Icosahedron via triangles in 3 dimensions, but it considers a geodesic grid on the surface of a 3D sphere as the base for the 4D representation as is shown in Figure 2.18. The translation is equally subdivided into 8 regular cells that shrink the range of translations.

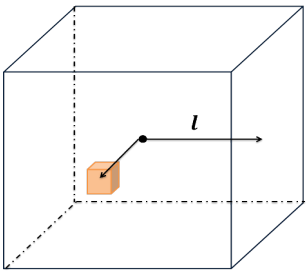
A better performance of the BnB method can be expected if the created subspaces are small enough, but it requires more computational cost. Therefore, this BnB method is often employed in collaboration with local approaches to refine and accelerate the alignment.



(a) Angle-axis representation

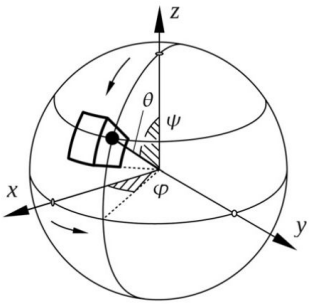


(b) Rotational Parametrization

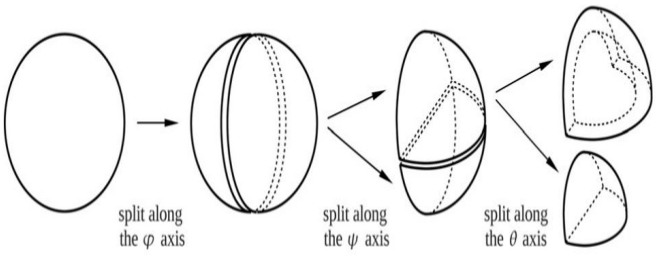


(c) Translational parametrization

Figure 2.16: (a) Axis-angle representation. (b) A 3D cube that encloses the π-sphere is proposed for an easy manipulation of the rotational space in [90, 13] and (c) a 3D cube of half-side length l to parametrize the translational space. The colored sub-cubes represent a sub-space of the transformation space.

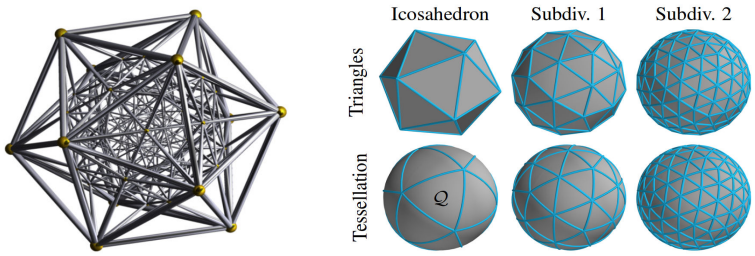


(a) Spherical coordinates representation



(b) Spherical boxes

Figure 2.17: Parametrization of the $SE(3)$ space by dividing along each axis of the axis-angle representation [70].



(a) 4D tetrahedra (b) Triangles division and Geodesic tessellation

Figure 2.18: (a) Projection of a 4D tetrahedra on a 3D space, where 600 sub-division have been performed. (b) Sub-division of the rotational space by considering triangles and a 4D tetrahedra [83]. At each subdivision 4 triangles are created.

Bounding methods

The bounding function is the key component of the BnB algorithm. The objective of bounding a non-convex function is to be as close as possible to the optimal solution using the least number of explorations in the search strategy. The bounding allows to determine whether the non-convex function being minimized may attain a value inferior to the current minimum error on a restricted domain in the parameter space [30]. Depending on how the solutions produced get close to the optimal solution, the bounding function can be categorized as strong or weak for the minimization process which will determine the living subspaces in the queue (The queue is initialized with a state representing all possible solutions). Each subspace is associated with a subregion of the transformation space $\mathbf{T}(\mathbf{x})_i \subset \mathbf{T}(\mathbf{x})$ so that $\mathbf{T}(\mathbf{x})_i \cup \mathbf{T}(\mathbf{x})_{i+1} = \mathbf{T}(\mathbf{x})$. Usually, the bounding is required to satisfy the following three conditions:

1. The value of the bound function is equal or less than the value of the objective function (for all subspaces in the tree) $\underline{\overline{\mathbf{E}(\mathbf{T}(\mathbf{x})_i)}} \leq \mathbf{E}(\mathbf{T}(\mathbf{x}))$.

2. The value of the bound function is at least equal to a feasible solution (for all live feasible solutions in the queue) $\underline{\overline{\mathbf{E}(\mathbf{T}(\mathbf{x})_i)}} = \mathbf{E}(\mathbf{T}(\mathbf{x}))$.

3. The value of a new generated bound is less or equal than its predecessor: $\underline{\overline{\mathbf{E}(\mathbf{T}(\mathbf{x})_i)}} \leq \underline{\overline{\mathbf{E}(\mathbf{T}(\mathbf{x})_{i+1})}}$, that can be written as $\underline{\overline{\mathbf{E}(\mathbf{T}(\mathbf{x})_i)}} \subset \underline{\overline{\mathbf{E}(\mathbf{T}(\mathbf{x})_{i+1})}}$.

where the operator $\underline{\overline{\cdot}}$ identifies both, upper and lower bound functions. All created subspaces are bounded and if the value of a bound contains a feasible solution or an optimal solution, then its value is used as the current best solution $\mathbf{E}(\mathbf{T}(\mathbf{x})^*)$ and other subspaces are compared to this value. A bounded subspace that is not better than the current best solution is discarded. Otherwise, the possibility to find a better solution is joined to the queue of subspaces to be analyzed. The lower and upper bound compare if the current solution is greater or equal to an optimal solution. Upper and lower bounding error functions can be identified as: $\overline{\mathbf{E}(\mathbf{T}(\mathbf{x}))}$ and $\underline{\mathbf{E}(\mathbf{T}(\mathbf{x}))}$, respectively. A bounding function provides for a given subspace a lower bound for the best solution.

The selection of live subspaces can be performed by removing states by removing the highest upper bounds or the lowest lower bounds from the priority queue (depending on the subspace searching strategy). A numeric example can be seen in Figure 2.19 where the subspaces with the lowest bounds are $\{1, 2, 3, 5, 7, 10, 11, 12, 14\}$.

Subspace searching methods

One of the main issues of BnB algorithms is the particular search strategy employed. Different strategies have different properties regarding time efficiency and memory consumption for sequential and parallel configuration. There are three main strategies in the BnB algorithm for exploring and for selecting generated subspaces which may contain an optimal solution: *Best-first*, *breadth-first* and *depth-first* strategies. These strategies mainly differ in the priority provided to a subspace to be evaluated.

The best first search strategy selects, among live subspaces, the one with the lowest bound. This has the advantage that no superfluous bound calculations take place after the optimal transformation has been found but memory problems can arise if the number of subspaces becomes too large. In order to optimize this strategy with respect to processed subspaces, the search strategy is often used with an eager BnB method, which calculates the bound when a subspace is created. However, the strategy can be equally used with a lazy evaluation which postpone bound calculation as long as possible. From the example given in Figure 2.19 the order of the explored subspaces by using this strategy is $1, 2, 3, 7, 14, 5, 11$.

In the breadth first search strategy all subspaces at one level of the search tree are processed before any other at a higher level, but the number of subspaces at each level of the search grows exponentially and it is not recommended for large problems. Despite this, the method can guarantee the finding of the optimal solution since all created subspaces are evaluated by the bounding. By following the example shown above and by using Figure 2.19, the order of explored subspaces will be $1, 2, 3, 7, 5, 14, 10, 11$.

An alternative used strategy is depth first search, where the live subspace with the largest level in the search tree is chosen for exploration. As well as the best first search strategy, the depth first strategy can be used for both eager or lazy subspace evaluation and the memory use depends on the chosen accuracy of the feasible solutions in subspaces, which is usually a manageable when not too many subspaces are created. One inconvenient is that if the best current solution is far from the optimal solution, unnecessary bounding computations may take place. Referring one more time to the given example in Figure 2.19, the order of explorations is $1, 2, 5, 11$, by following the depth first search strategy.

The speed of the BnB algorithm depends on the number of subspaces that need to be evaluated. A subspace is discarded from the live subspaces if the calculated lower bound for it exceeds the current minimum error, otherwise, the subspace is subdivided and the evaluation is repeated for the new generated subspaces. In conclusion, the better the lower bound is, fewer subdivisions will be necessary.

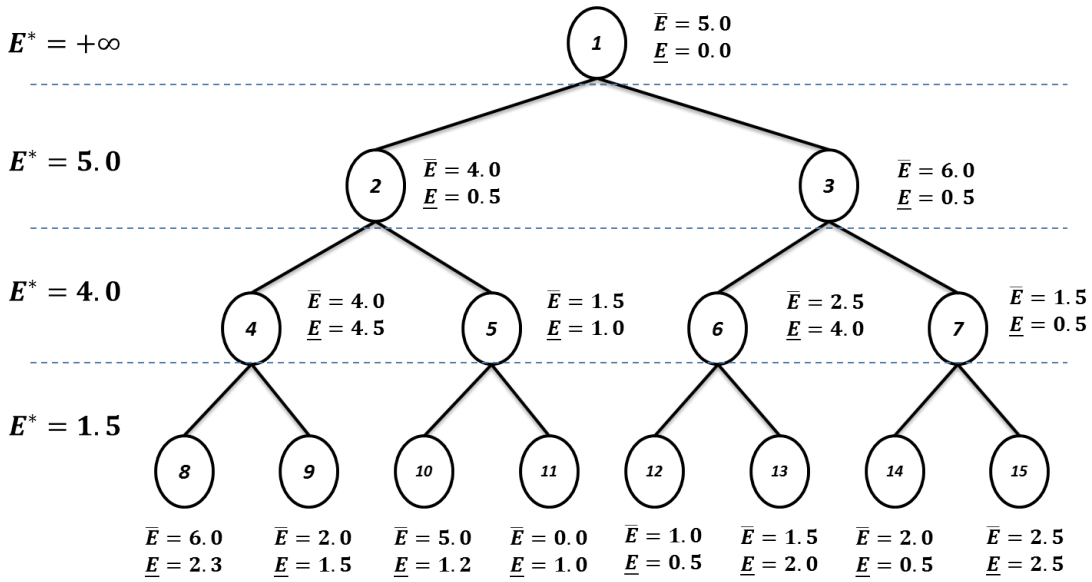


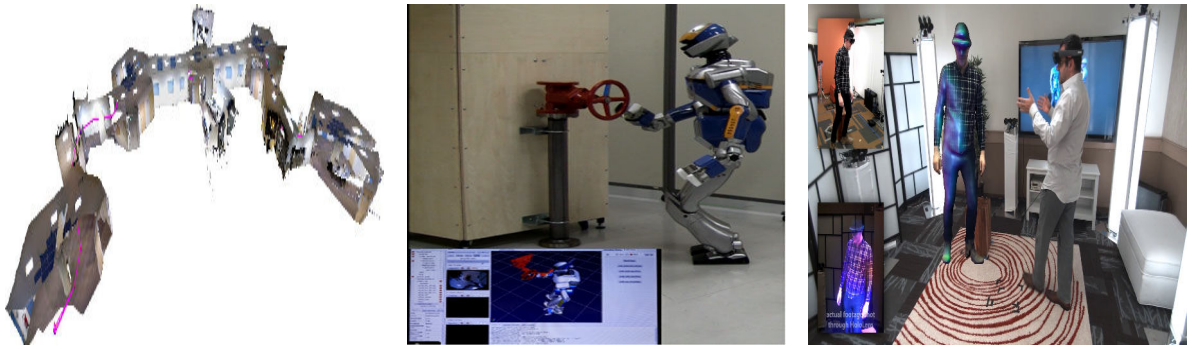
Figure 2.19: Search strategies in BnB example. Each numerated node contain a subset of the transformation $\mathbf{T}(\mathbf{x})$ where the subspace 11 contain the optimal solution. $\mathbf{E}^*$ is the computed current best error which is initialized as ∞ and updated if the upper bound is less than it $\overline{\mathbf{E}} < \mathbf{E}^*$. Upper and lower error bounds are identified as $\overline{\mathbf{E}}$ and $\underline{\mathbf{E}}$ respectively. During the BnB process, live subspaces are discarded from the queue if the lower bound is greater or equal than the best current error at each level $\underline{\mathbf{E}} \geq \mathbf{E}^*$.

CHAPTER 3

RGB-D REGISTRATION

3.1 Introduction

RGB-D registration is fundamental in many research areas of computer vision and extensively employed in robotics, ranging from 3D reconstruction, object tracking, mobile robotics, surveillance and augmented reality applications (Figure 3.1). The RGB-D sensor pose and the environment structure can be estimated simultaneously and in real-time. Therefore, it is an important task to robustly and accurately estimate the pose of the sensors. In this chapter, RGB-D registration will be introduced by explaining how color and depth can be used to define an error function based on this metric information.



(a) 3D Mapping (b) Mobile robotics (c) Augmented reality

Figure 3.1: Applications of RGB-D registration. (a) 3D visual real-time SLAM [58], (b) A RGB-D sensor is employed in a HRP 2 humanoid robot for modeling, tracking and operating a valve [Joint Robotics Laboratory (CNRS/AIST)] (c) A system of various RGB-D sensors are employed to perform virtual reality telepresence, called holoportation [68].

The first part of this chapter is aimed at introducing the working principle of RGB-D sensors and how the distance between correspondences can be estimated. Particularly, depth-based error functions such as point-to-point, point-to-plane, and plane-to-plane ICP will be explained in 3.3 and a color-based error function based on a direct image-based approach in 3.4. The second part of this chapter (Section 3.5), will introduce how depth-based and image-based approaches can work together to estimate the pose in a so-called hybrid manner. The derivation of the joint error function will be introduced, which will provide a strategy to find a solution to the view registration problem.

3.2 RGB-D sensors

The availability of RGB-D sensors have provided the possibility to acquire color and depth information simultaneously at a considerable high framerate, which has been useful for real-time pose estimation. Ranging sensors such as stereo systems, infrared cameras or 3D laser scanners provide a direct estimation of the distance between the camera and an object. They are commonly fused with high resolution cameras for texturing pointclouds. Sensors such as the Photonic Mixer Device (PMD) SR4000 or PMD Camboard pico series, use infrared (IR) technology to estimate depth measurements by modulating the IR frequency. Depth sensors such as the ZED or ENSENSO 3D can estimate distance by computing the disparity map of a camera stereo system, while LIDAR (Light Imaging Detection And Ranging) devices can provide 3D coordinates of a point by using a finer laser beam combined with a scanning form as rotating mirrors or pant-tilt systems.

(a) IR cameras (b) Stereo system (c) LIDAR

Figure 3.2: Depth sensors. (a) PMD Camboard pico Max and Monster (b) ZED (c) Vedolyne LIDAR

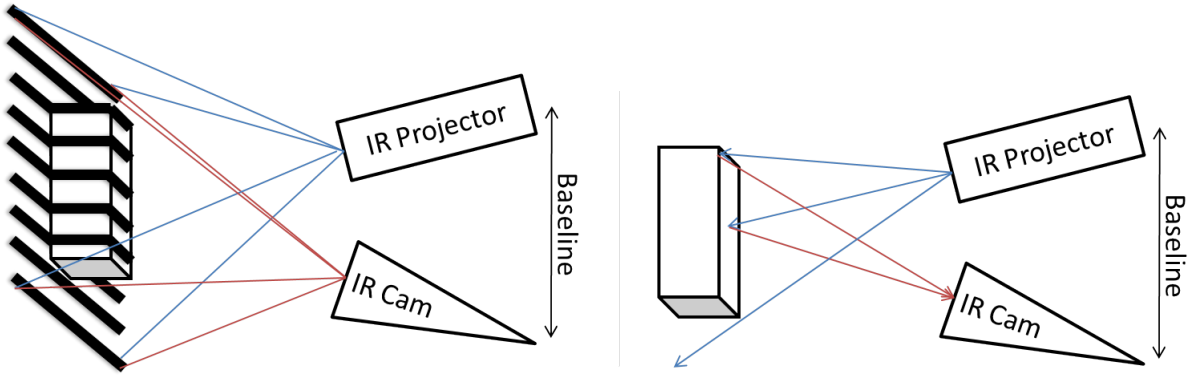
Generally, embedded RGB-D sensors such as the Microsoft Kinect, Asus Xtion or Intel RealSense technologies have integrated color cameras along with infrared sensors to estimate depth with two technologies: *time-of-flight* (ToF) and *structured-light* (SL) (Figure 3.4). A detailed comparison between both technologies can be found in [76] and a list of RGB-D sensors widely used in robotics can be found in Appendix A.1. These two technologies have been recently integrated in smartphones (Figure 3.3) and used for augmented reality experiences and 3D mapping. Examples as Asus Zenfone AR and Lenovo Phab2 Pro employ the Google Tango technology which incorporates a RGB-D sensor system (along with other sensors such as IMU and GPS).

Figure 3.3: RGB-D Smartphones. In the left the Asus Zenfone AR and in the right the Lenovo Phab2 Pro

The working principle of the time-of-flight technology to estimate distance is by measuring the time that the light needs to travel from the light source to the object and back to the camera where the entire scene is captured with each light pulse. The scene is illuminated with a modulated light source (near-infrared range 850nm) and an imaging sensor that responds to the same spectrum receives the light and it digitalize the signal. In ToF sensors, distance is measured for every pixel in a 2D array. The depth information is obtained by measuring the reflected component of the light (the light entering the sensor has an ambient component and a reflected component) which is acquired by modulating a continuous-wave in the light emitter (light source is pulsed) typically as a sinusoid or square wave (Figure 3.5). The depth Z_k is estimated as follows:

$$Z_k = \frac{c}{4\pi\omega}\phi \tag{3.1}$$

where ϕ is the phase shift, ω is the modulation frequency (in the range 10 to 100 MHz) and $c = 3 \times 10^8 m/s$ is the speed of light. The emitted sinusoidal signal $g(t) = cos(\omega t)$ is received after reflection as $s(t) = b + acos(\omega t + \phi)$, where b is a constant bias and a is the amplitude.



(a) Structured light (b) Time of flight

Figure 3.4: Depth IR technologies. (a) A known pattern is projected on the scene. The deformation of the pattern is registered by the IR camera and estimate distances (b) Depth is measured by counting the time that it takes for a light signal to travel to an object and back to the camera.

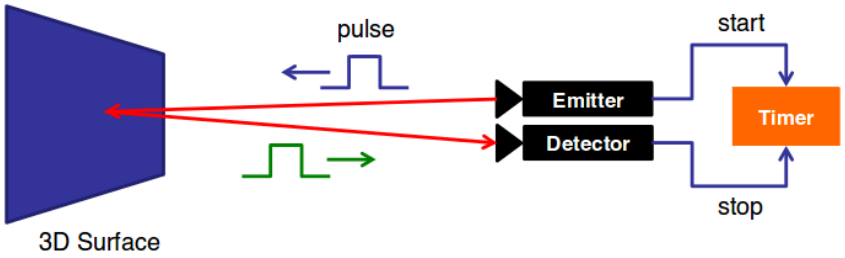
The cross correlation of both signals is represented as:

$$c(\tau) = s * g = \int_{-\infty}^{\infty} s(t) \cdot g(t+\tau)dt = \frac{a}{2}cos(\omega\tau + \phi) + b \tag{3.2}$$

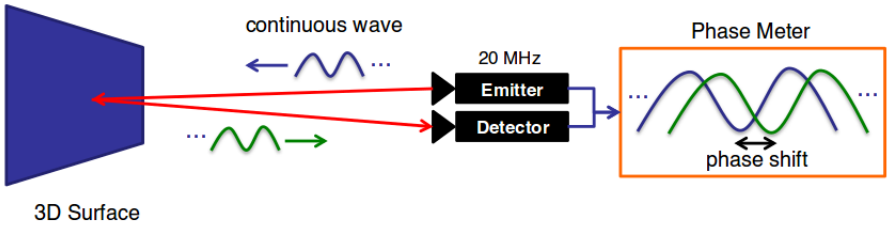
where τ is the offset. If the cross-correlation function is sampled at four sequential instants with different phase offset τ, then the sought parameters can be directly obtained as:

$$\begin{aligned} \phi &= arctan2(A_3 - A_1, A_0 - A_2) \\ a &= 1/2\sqrt{(A_3 - A_1)^2 + (A_0 - A_2)^2} \end{aligned} \tag{3.3}$$

where $A_i = c(i \cdot \pi/2)$. Time-of-flight sensors can deliver higher accuracy, which is reflected in the maximum measurable distance as $Z_{max} = \frac{c}{2\omega}$, where it can be seen that the measurable distance can be extended by reducing the modulation frequency.



(a) Pulsed modulation



(b) Continuous-wave modulation

Figure 3.5: Time of flight wave modulation [Computed Aided Medical Procedures, TUM]. (a) The distance is obtained by measuring the absolute time that a light pulse needs to travel from a source into the 3D scene and back, after reflection. (b) Depth is obtained by measuring the phase shift from a reflecting surface. A continuous wave (sinusoidal) is emitted instead of short light pulses.

On the other hand, structured-light depth sensors project a known pattern onto the object and it obtain depth by analyzing the deformation of the pattern. A basic structured-light system consists of one camera (or multiple) that projects optical patterns. Different design options of the pattern projection are cited in [76]. The source emits multiple beams to create a constant pattern of speckles projected onto the scene. This pattern is captured by the infrared camera and is correlated against a reference pattern, which is obtained by capturing a plane at a known distance

from the sensor and stored in the firmware of the sensor. A structured-light device generates dense reconstructed points by locating image points on each light pattern in the image. Generally, the measurement of depth Z_k is obtained as a triangulation process w.r.t. the reference Z_0 as:

$$Z_k = \frac{Z_0}{1 + \frac{Z_0}{bf} d} \tag{3.4}$$

where b is the baseline between the IR camera and the emitter, f is the focal length of the IR camera, D is the displacement of the k-th 3D point and d is the disparity. The reflected pattern is sensitive to optical interference from the environment, where it can be inferred that structured-light RGB-D devices are designed to work best indoors at moderate distances (which can vary depending on the conditions of the environment).

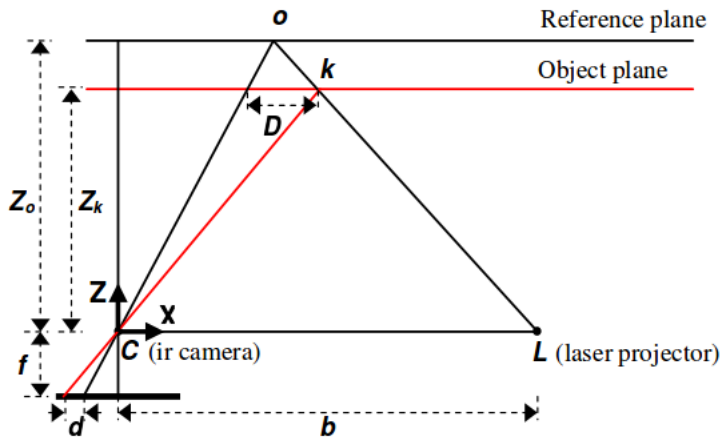
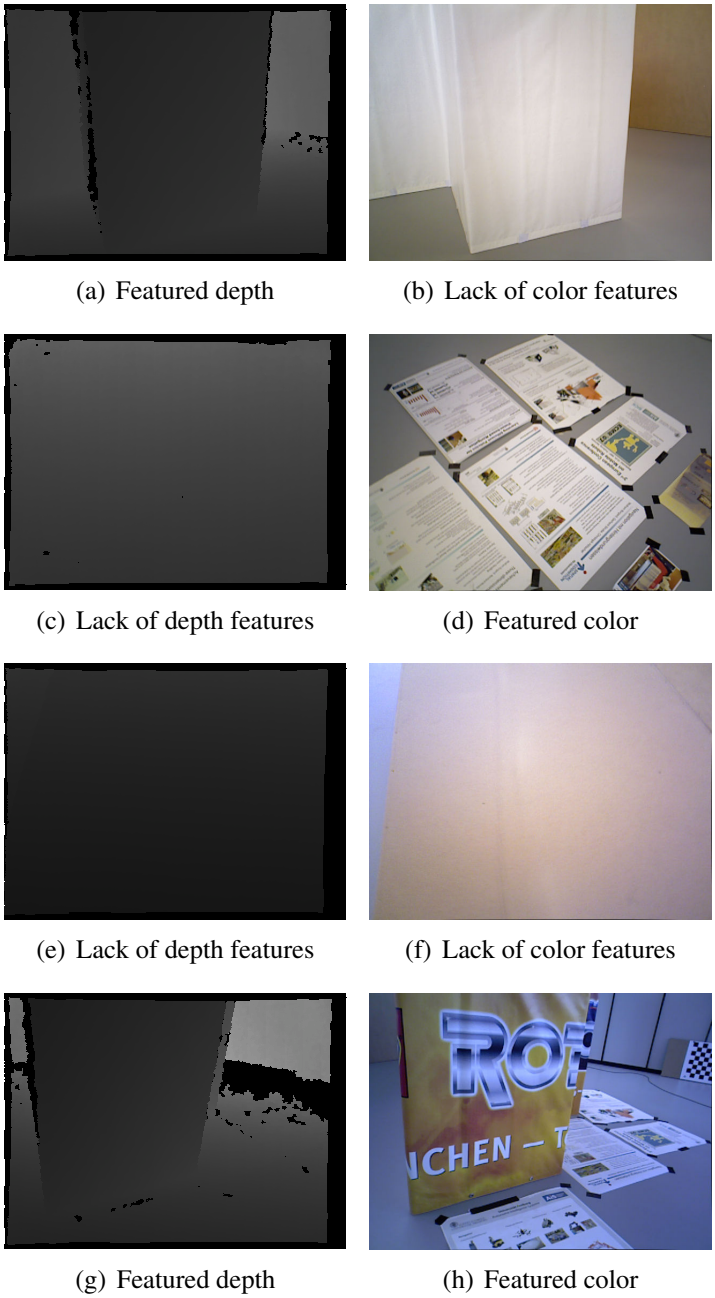


Figure 3.6: Schematic representation of how depth is obtained by structured light technology [46]. The variables are described in the text.

Color and depth information obtained by RGB-D sensors have been separately studied in the literature to estimate the 6DOF pose. One case is by using depth images, where *geometric*-based methods, such as the well known Iterative Closest Point (ICP) [9] and its variants, have demonstrated the ability to obtain robust alignments when enough geometric information is available and they can obtain fast alignment if the datasets are close and contain a large overlap. Particular variants such as the Point-to-plane strategy [16] and the Generalized-ICP [78] have demonstrated to be the most effective. These methods can be made robust when combined with robust estimation approaches such as M-estimators [36]. On the other hand, color images have been used for pose estimation processes by performing *photometric*-based minimization. Strategies as direct approaches based on view synthesis [38] or feature-based strategies have been widely employed for minimizing the photometric term.



(a) Featured depth (b) Lack of color features

(c) Lack of depth features (d) Featured color

(e) Lack of depth features (f) Lack of color features

(g) Featured depth (h) Featured color

Figure 3.7: Examples of RGB-D images: Texture vs Color [84]. (a), and (b) are images taken from an environment where the geometric measurements are more significant than photometric measurements while (c), and (d) were taken from a scenario where the texture is more significant than geometry. (e) and (g) shown the lack of color and depth while (g) and (f) shown rich texture and depth. The depth image is visualized as an intensity image. For depth images in the first column, black indicates a non valid depth measurement.

Furthermore, hybrid strategies have been proposed for combining the main benefits of both, geometric and photometric approaches. Hybrid strategies can obtain faster and more accurate alignments than geometric or photometric approaches alone but they require the definition of a tuning parameter that weighs the importance of the measurements. The contribution of each should be weighted to compensate for the relative uncertainty between the different error functions that are simultaneously minimized.

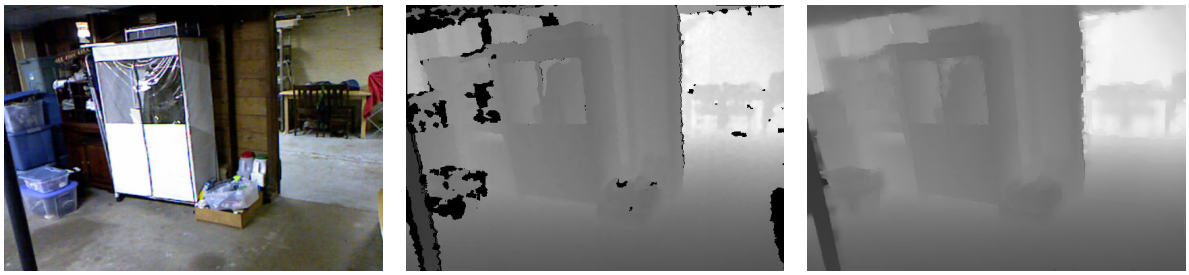
Consider as an example the case when the intensity is almost uniform in the scene but the geometric features are not (Figure 3.7(b) and 3.7(a)). The error function should give more importance to the geometric features since the error generated in the photometric term does not have enough gradient to constrain the alignment. Furthermore, visual features may change significantly between viewpoints due to the fact of light conditions, where geometry is often less affected. On the other hand, the opposite case can be found when rich texture can be registered from flat surfaces (Figure 3.7(d) and 3.7(c)), as the geometric information has no gradient and does not constrain the alignment but texture does. In order to better introduce how hybrid approaches handle the uncertainty factor between different measurements types, the estimation of the tuning parameter will be explained in detail in 3.5.1.

One of the main applications of using color and/or depth images has been the reconstruction of 3D structures of an unknown environment. Simultaneous localization and mapping (SLAM) techniques allow to estimate the sensor motion and reconstruct an unknown environment which has been widely employed in 3D modeling, augmented reality and autonomous vehicles. A survey of recent SLAM contributions have been done in [85]. For purposes of this thesis, however, only visual SLAM approaches that uses color and depth together will be analyzed.

3.3 Depth-based error functions

Depth perception provides the ability to understand the distance to objects in the scene at different view points and it is often used to create a 3D representation of the environment. Aforementioned, depth maps can be generated by 3D cameras with various technologies (Structured light or time-of-flight), where the depth is often encoded as a gray scale image in a linear or logarithmic scale of eight or more bits of resolution. Each pixel of the depth image is associated with the RGB image, which can be one-to-one relation if the images are the same size, otherwise the depth map (or the RGB image) can be resized in order to obtain the associations.

One of the main issues with estimating depth from the IR sensor, is the presence of non valid data or "holes", where the depth data are missing. These holes mainly appear in depth images due to occlusions or matching ambiguity. The latter occurs when correspondences cannot be found due to light sources high in IR like sunlight or incandescent bulbs, or objects that do not reflect IR light. Another scenario is when multiple reflections return to the camera as translucent objects that return reflections from the object and the background behind. This can be a problem for some 3D cameras that use active illumination such as SL technologies, due to the fact that the pattern can become corrupted by ambient light and shadows. In the case of the ToF technology, despite the fact that it is more robust to ambient lighting than SL and it works outdoors, the IR depth sensor can get saturated when the scene is too bright and in consequence non valid values are obtained.



(a) Color image (b) Raw Depth image (c) Filtered Depth image

Figure 3.8: Pre-processing example of a depth image. The holes in the depth maps can be filled with information of the surrounded pixels.

Another important problem is depth discontinuities (depth edges). This problem mainly depends on the camera viewpoint, where surface points may not correspond to edges. It is important to preserve sharp depth edges in order to correctly map the scene and the ToF technology can often obtain sharper edges since only one viewpoint is used to obtain depth.

In order to deal with non valid data, image pre-processing over the depth map can be performed in order to "fill" the holes and sharp edges. If the pre-processing is performed correctly, then it can improve the quality of the depth image. Basically, strategies to estimate depth for non valid data involve employing information of the surrounding valid pixels (Figure 3.8). One of the most classic approaches is to interpolate the neighbouring pixels [65, 53, 11], by a modified bilateral filter method [71, 15, 90] or more complex segmentation methods [80, 37, 77]. For purposes of this thesis, bilinear interpolation and a bilateral filter will be employed to render the depth image.

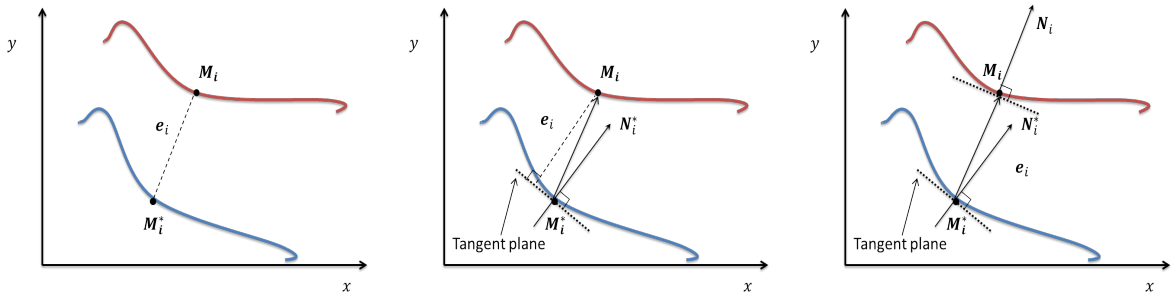
An error function based on the depth image will be introduced. Consider here two RGB-D frames $\mathcal{I}$ of dimensions $m \times n$ associated with an intensity function $\mathbf{I}(\mathbf{p})$ and a depth function $\mathbf{Z}(\mathbf{p})$, where the parameter $\mathbf{p} = [u\ v]^\top \in \mathbb{R}^{2\times mn}$, correspond to the pixel coordinates of $\mathcal{I}$. Assuming that the information of the metric distance $Z_k \in \mathbb{R}^+$ for each pixel is already known, a 3D Euclidean point is computed from this value according to the following equation:

$$\mathbf{P}_i = \mathbf{K}^{-1}\overline{\mathbf{p}_i}\, Z_k = \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \in \mathbb{R}^{3\times 1} \tag{3.5}$$

where $\overline{\mathbf{p}_i}$ is the homogeneous representation of $\mathbf{p}_i$ with $i = 1, 2, \cdots, mn$ and $\mathbf{K} \in \mathbb{R}^{3\times 3}$ is the calibration matrix which contains the intrinsic parameters of the camera.

Let there be established two measurements vectors that contains a set of 3D coordinates obtained at different viewpoints as $\mathbf{M}^* = [\mathbf{P}_1^*, \mathbf{P}_2^*, \ldots, \mathbf{P}_{mn}^*]$ and $\mathbf{M} = [\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_{mn}]$. The classic ICP algorithm fixes one of the clouds and it tries to find the closest points in the other set for each point. Two cases can be established by performing this. If the reference measurements is fixed (case 1), the following error function can be used:

$$\mathbf{e}_{G_i} = \mathbf{M}_i^* - f\left(\mathbf{T}(\mathbf{x}), \overline{\mathbf{M}_i}\right) \in \mathbb{R} \tag{3.6}$$



(a) Point-to-point distance (b) Point-to-plane distance (c) Plane-to-plane distance

Figure 3.9: Variants of the ICP algorithm. The distance between 3D correspondences can improve the performance of the ICP algorithm, since it is considered as the error function that is iteratively minimized.

Alternatively, it is also possible to fix the current dataset (case 2) and solve the optimization problem as:

$$\mathbf{e}_{G_i} = \mathbf{M}_i - f\left(\mathbf{T}\left(\mathbf{x}\right), \overline{\mathbf{M}_i^*}\right) \in \mathbb{R} \tag{3.7}$$

where the motion parameter $\mathbf{x}$ is an unknown parameter.

The difference between selecting case 1 or case 2 as the error function is the computational cost when real-time performance is required. For case 1 several parameters such as the normals and the matching points can be pre-computed only once and not at each iteration, in case 2 the estimation of these parameters can be computationally expensive. Therefore, for the purpose of this thesis, the reference measurements are considered as the fixed dataset and the current measurements are warped.

The objective of the ICP algorithm is to reduce the distance between correspondences of two pointclouds at each iteration. The computation of the geometric distance between exact correspondences can speed up the alignment. Three main ICP approaches have been considered in the literature as Point-to-point (P2P) [9], Point-to-plane (P2Pl) [16] and Plane-to-plane (Pl2Pl) [78]. The difference between these strategies is how the distance between the closest points is found. In the Point-to-point distance, the closest squared distance between a current point and a reference point is minimized. For the Point-to-plane distance, the closest point is obtained by finding the closest distance between a transformed query point and the tangent plane of a point in the reference dataset. Finally, the Plane-to-plane finds correspondences by analyzing the surface orientations in both datasets, where points with normal orientation inconsistencies are discarded (See Figure 3.9).

The P2P and P2Pl strategies can be seen as a special case of the Pl2Pl strategy, the difference lies in how the covariance matrices (which are associated with the measured points) are established and in an extra step to compute the normals. The performance of these geometric strategies while aligning 3D pointclouds is compared in [32]. In order to introduce the error functions for each strategy (which are used throughout this manuscript and employed in the IRLS pipeline), special cases of the Pl2Pl method will be shown along with the derivation of its respective Jacobian.

3.3.1 Plane-to-plane ICP

The Pl2Pl method assumes the existence of an underlying set of points in a probabilistic model, which generates $\mathbf{M}^*$ and $\mathbf{M}$ according to $\mathbf{M}^* \sim \mathcal{N}(\mathbf{M}^*, \mathcal{E}_i^{\mathbf{M}^*})$ and $\mathbf{M} \sim \mathcal{N}(\mathbf{M}, \mathcal{E}_i^{\mathbf{M}})$, where $\mathcal{E}_i^{\mathbf{M}^*} \in \mathbb{R}^{3\times3}$ and $\mathcal{E}_i^{\mathbf{M}} \in \mathbb{R}^{3\times3}$ are covariance matrices associated with the measured points. Since $\mathbf{M}^*$ and $\mathbf{M}$ are assumed to be drawn from independent Gaussians the following geometric error function can be written:

$$\begin{aligned} \mathbf{e}_{G_i} &\sim \mathcal{N}\left(\left(\mathbf{M}_i^* - \Pi_3\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\overline{\mathbf{M}_i}\right), \mathcal{E}_i^{\mathbf{M}^*} + \mathbf{T}^*(\mathbf{x})\,\mathcal{E}_i^{\mathbf{M}}\mathbf{T}^*(\mathbf{x})^\top\right) \\ \mathbf{e}_{G_i} &\sim \mathcal{N}\left(0, \mathcal{E}_i^{\mathbf{M}^*} + \mathbf{T}^*(\mathbf{x})\,\mathcal{E}_i^{\mathbf{M}}\mathbf{T}^*(\mathbf{x})^\top\right) \end{aligned} \tag{3.8}$$

where $\mathbf{T}^*(\mathbf{x}) = \Pi_3\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})$ is the correct transformation. $\Pi_3 = [\mathbb{I}_{3\times3}, \mathbf{0}_3] \in \mathbb{R}^{3\times4}$ is the projection matrix. When the transformation parameter is unknown, its value can be found by iteratively minimizing (2.8), where it is possible to integrate (3.8) as:

$$\mathbf{T}(\mathbf{x}) = \underset{\mathbf{T}(\mathbf{x})}{argmin} \sum_{i=1}^{N} \mathbf{e}_{G_i}^\top \left(\mathcal{E}_i^{\mathbf{M}^*} + \mathbf{T}^*(\mathbf{x})\,\mathcal{E}_i^{\mathbf{M}}\mathbf{T}^*(\mathbf{x})^\top\right)^{-1} \mathbf{e}_{G_i} \tag{3.9}$$

that defines the called Generalized ICP error function [78]. The Pl2Pl error function can be obtained by verifying the orientation of corresponding normals in both sets. This is achieved by setting:

$$\mathcal{E}_i^{\mathbf{M}^*} = \mathbf{R}(\mathbf{x})_{\mathbf{N}_i^*} \begin{pmatrix} \psi & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{R}(\mathbf{x})_{\mathbf{N}_i^*}^\top \tag{3.10}$$

$$\mathcal{E}_i^{\mathbf{M}} = \mathbf{R}(\mathbf{x})_{\mathbf{N}_i} \begin{pmatrix} \psi & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \mathbf{R}(\mathbf{x})_{\mathbf{N}_i}^\top \tag{3.11}$$

where ψ is the term that represents the uncertainty along the surface normal and $\mathbf{R}(\mathbf{x})_{\mathbf{N}_i^*}$, $\mathbf{R}(\mathbf{x})_{\mathbf{N}_i}$ are the rotation matrices which transform the reference $\mathbf{N}_i^*$ and current $\mathbf{N}_i$ normal vector, respectively. The use of information from both sets of measurements decreases the influence of incorrect correspondences. Consequently, the Pl2Pl distance becomes less critical for performing ICP and it increases its accuracy [61].

3.3.2 Point-to-point ICP

The Euclidean distance can be estimated between the warped pointcloud $\mathbf{M}^w$ by $\mathbf{T}(\mathbf{x})$ and the matched cloud $\mathbf{M}^m$ in a one-to-one relation. By associating each point of the transformed cloud with its closest point in the reference cloud at each iteration, the norm of the geometric error $\mathbf{e}_G$ can be compared with an established threshold ϵ, which is one criteria to stop the iterative method if $||\mathbf{e}_G|| \leq \epsilon$. It can be seen that the classic Point-to-point ICP error function can be obtained by setting $\mathcal{E}_i^{\mathbf{M}^*} = \mathbb{I}_{3\times3}$ and $\mathcal{E}_i^{\mathbf{M}} = \mathbf{0}_{3\times3}$ in (3.9). In this case, (3.9) becomes:

$$\mathbf{T}(\mathbf{x}) = \underset{\mathbf{T}(\mathbf{x})}{argmin} \sum_{i=1}^{N} \mathbf{e}_{P2P_i}^\top \mathbf{e}_{P2P_i} = \underset{\mathbf{T}(\mathbf{x})}{argmin} \sum_{i=1}^{N} ||\mathbf{e}_{P2P_i}||_2^2 \tag{3.12}$$

where the error function is determined by the Point-to-point distance between correspondences, which can be given as:

$$\mathbf{e}_{P2P_i} = \mathbf{M}_i^m - \mathbf{M}_i^w \in \mathbb{R}^3 \tag{3.13}$$

where $\mathbf{M}_i^m$ is the matched point and $\mathbf{M}_i^w = \Pi_3 \widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\overline{\mathbf{M}_i}$ is the warped point.

The geometric Jacobian can be derived from (3.13), where each element of the matrix is as follows:

$$\mathbf{J}_{P2P_i} = \frac{\partial \mathbf{e}_{P2P_i}}{\partial \mathbf{x}} = \begin{bmatrix} 1 & 0 & 0 & 0 & Z_i^w & -Y_i^w \\ 0 & 1 & 0 & -Z_i^w & 0 & X_i^w \\ 0 & 0 & 1 & Y_i^w & -X_i^w & 0 \end{bmatrix} = [\mathbb{I}_{3\times 3} - [\mathbf{P}_i^w]_x] \in \mathbb{R}^{3\times 6} \tag{3.14}$$

The P2P ICP algorithm needs to find the nearest point for each point in the warped pointcloud at the point of the reference pointcloud, therefore the computational cost of this strategy depends on the size of the datasets and the finding of corresponding points. The average complexity of the P2P ICP algorithm is $O(Nlog(N))$, where N is the number of corresponding points. An optimal solution of the P2P ICP is demonstrated under the assumption that the number of correspondent points remains constant and all points have a one-to-one match in the other dataset, however, the estimation of a proper initial transformation is needed to avoid the local minima problem.

3.3.3 Point-to-plane ICP

The Point-to-plane error function proposed in [16] computes a distance between two points projected onto the normal direction of the tangent plane associated with one or both of the points. In other words, the error function (3.13) between the transformed point and the surface plane tangent is projected along the normal direction of either the warped point or the matched point. Similarly, the Point-to-plane error function can be analyzed in probabilistic terms by considering the normals as information (3.16) into (3.9), which is performed as:

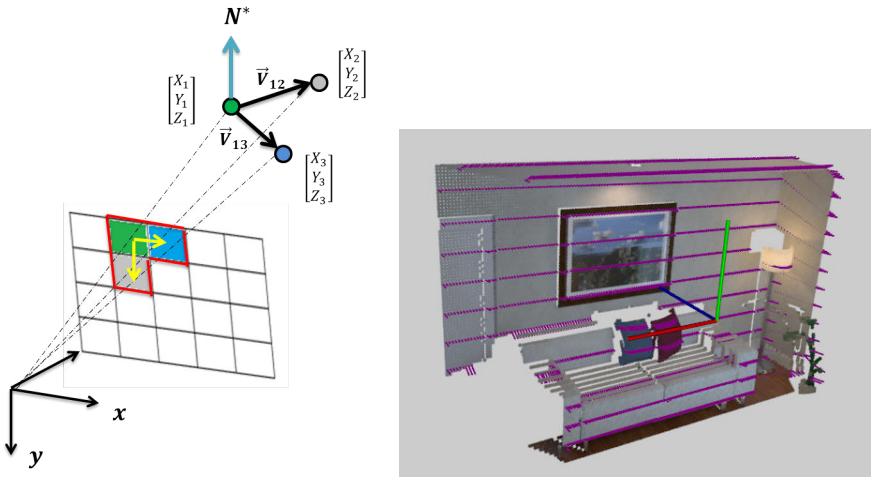
$$\mathbf{T}(\mathbf{x}) = \underset{\mathbf{T}(\mathbf{x})}{argmin} \sum_{i=1}^{N} \left|\left|\mathbf{N}_i^{m\top} \left(\mathbf{M}_i^m - \mathbf{M}_i^w\right)\right|\right|_2^2 = \underset{\mathbf{T}(\mathbf{x})}{argmin} \sum_{i=1}^{N} \mathbf{e}_{P2P_i}^\top \mathbf{N}_i^m \mathbf{e}_{P2P_i} \tag{3.15}$$

Since $\mathbf{N}_i^m$ is an orthogonal projection matrix, then $\mathbf{N}_i^m = \mathbf{N}_i^{m2} = \mathbf{N}_i^{m\top}$. $\left|\left|\mathbf{N}_i^{m\top}\mathbf{e}_{P2P_i}\right|\right|_2^2$ can be reformulated as a quadratic form: $\left|\left|\mathbf{N}_i^{m\top}\mathbf{e}_{P2P_i}\right|\right|_2^2 = \left(\mathbf{N}_i^m\mathbf{e}_{P2P_i}\right)^\top \left(\mathbf{N}_i^m\mathbf{e}_{P2P_i}\right) = \mathbf{e}_{P2P_i}^\top \mathbf{N}_i^m \mathbf{e}_{P2P_i}$. It can be shown that the P2Pl ICP is a case of the Pl2Pl ICP by setting $\mathcal{E}_i^{\mathbf{M}^m} = \mathbf{N}_i^{m-1}$ and $\mathcal{E}_i^{\mathbf{M}} = \mathbf{0}_{3\times 3}$ in (3.9). Therefore, the Point-to-plane error function is defined then as:

$$\mathbf{e}_{P2Pl_i} = \mathbf{N}_i^{m\top} \left(\mathbf{M}_i^m - \mathbf{M}_i^w\right) \in \mathbb{R} \tag{3.16}$$

where $\mathbf{N}_i^m = [N_{X_i}\ N_{Y_i}\ N_{Z_i}]^\top \in \mathbb{R}^3$ is the normal. Notice that the normal is added as a scalar in the error function, which modifies the Jacobian as follows:

$$\mathbf{J}_{P2Pl_i} = \begin{bmatrix} N_{X_i} & N_{Y_i} & N_{Z_i} & N_{Z_i}Y_i^w - N_{Y_i}Z_i^w & N_{X_i}Z_i^w - N_{Z_i}X_i^w & N_{Y_i}X_i^w - N_{X_i}Y_i^w \end{bmatrix}^\top \in \mathbb{R}^6 \tag{3.17}$$



(a) Estimation of a 3D normal (b) 3D normals

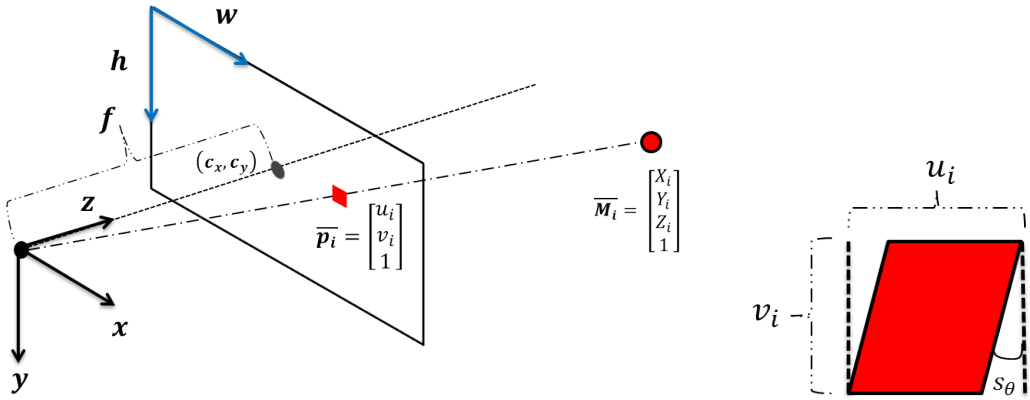
Figure 3.10: (a) Estimation of the normal by using the neighbouring pixels to a central pixel (in green) in depth frames. The cross product between the vectors $\mathbf{V}_{12}$ and $\mathbf{V}_{13}$ is performed for each pixel of the depth frame where the corners are avoided. (b) 3D normals samples of the estimated cloud from the RGB-D frame is shown.

In 3D geometry, at least three 3D points are needed to compute the surface normal. These points will generate a plane $ax + by + cz + d = 0$, where $\langle a, b, c \rangle$ are the normal coordinates. Various strategies ranging from classic cross product between points to probabilistic models can be considered to estimate the normal. When depth frames are employed and a faster computation is required, the closest neighbours to a central pixel in a 2×2 window of the depth image can be used to perform the cross product between the vectors formed by these 3D neighbours and estimate the normal (An example is shown in Figure 3.10(a), where $\overrightarrow{\mathbf{V}}_{12}$ and $\overrightarrow{\mathbf{V}}_{13}$ are the vectors that generate the plane formed by $\mathbf{M}_1$, $\mathbf{M}_2$ and $\mathbf{M}_3$). An extended window can be equally used, which increases the accuracy of the estimation of the normal but implies more computational cost. A similar performance is obtained by employing probabilistic models [78]. The window is applied over all the pixels in the image (avoiding the corners) and the normals are estimated for each central pixel by performing the cross product of the associated vectors.

3.4 Image-based error functions

RGB-D camera motion can also be directly estimated by minimizing over the intensity error between the current and reference frames. There exist various image registration approaches that can estimate the pose from corresponding points. These matched points are often easily and robustly detectable from the image by considering salient geometric features that are spread over the image (corners, lines, etc.). However, the features need to have an accurate localization and should be invariant to expected image changes.

Alternatively to feature-based methods, direct approaches do not perform any feature extraction or matching processes and they operate directly on the intensity values in a close neighborhood of each intensity value. Direct approaches are widely used if the image does not contain distinctive features and the overlap of imaged scenes is large. When frame-to-frame RGB-D registration is performed, the framerate of acquisitions allows to maintain sufficient overlap and similar features in acquired RGB images. Therefore, a direct error function based on the intensity image warping function will be defined in this section.



(a) Pinhole camera model (b) Skew angle

Figure 3.11: The coordinates of a 3D point $\mathbf{M}_i$ are projected onto a 2D frame with pixel coordinates $\mathbf{p}_i$. (a) The model of a pinhole camera is employed, where (c_x, c_y) are coordinates (in pixels) usually placed at the center of the $w \times h$ frame and f is the focal distance (b) the skew factor s_θ of a pixel is the angle between the image axis u_i and v_i.

An image is an array of intensities captured by a camera sensor which represent the scene. This makes sense of perspective, depth and color of the 3-D world. The most common way to register an image is through the pinhole camera model (Figure A.2), which projects a 3D point $\mathbf{P}_i = [X_i\ Y_i\ Z_i]^\top$ onto a 2D frame at the pixel coordinates $\mathbf{p}_i = [u_i\ v_i]^\top$ as follows:

$$Z_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \begin{bmatrix} fw & fs_\theta & c_x \\ 0 & fh & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix} \tag{3.18}$$

where f is the focal distance, s_θ is the skew angle of a pixel (which is usually set to 0), w and h are the width and height of the image, respectively and c_x, c_y are the coordinates of the center of the image. These are the intrinsic parameters of the camera. For simplicity, (3.18) can be written as: $Z_i\overline{\mathbf{p}_i} = \mathbf{K}\Pi_3\overline{\mathbf{P}_i}$.

In order to introduce an error function based on image view synthesis, the color brightness function at each pixel coordinates will be considered here as a measurement vector defined by $\mathbf{I}_i^* = \mathbf{I}^*(\mathbf{p}_i^*)$ and $\mathbf{I}_i = \mathbf{I}(\mathbf{p}_i)$ for the reference and current frame, respectively. As is shown in the Figure 3.12, a reference image represented by $\mathcal{I}^* = [\mathbf{I}_1^*, \mathbf{I}_2^*, \ldots, \mathbf{I}_{mn}^*]$ and a current image by $\mathcal{I} = [\mathbf{I}_1, \mathbf{I}_2, \ldots, \mathbf{I}_{mn}]$ are established. A new image $\mathcal{I}^w$ can be synthesized between two positions

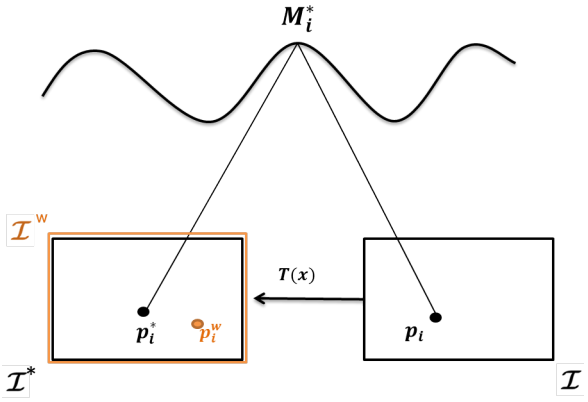


Figure 3.12: For the intensity-based minimization of the error function e_I, a new image $\mathcal{I}^w$ is synthesized from $\mathcal{I}$ based on the pose $\mathbf{T}(\mathbf{x})$. $\mathcal{I}^*$ is the reference image.

by projecting the reference poincloud $\mathbf{P}^*$ into the current image $\mathcal{I}$ and then by linearly interpolating the intensity values on a regular array such as:

$$\mathcal{I}^w = \mathcal{I}\left(\omega(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \mathbf{P}^*)\right) \tag{3.19}$$

where the geometric warping function $\omega(\cdot)$ obtain the warped pixels such as:

$$\overline{\mathbf{p}_i}^w = \frac{\mathbf{K}\Pi_3\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\overline{\mathbf{M}_i}^*}{\mathbf{e}_3^\top\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\mathbf{M}_i^*} = \begin{bmatrix} u_i^w \\ v_i^w \\ 1 \end{bmatrix} = \begin{bmatrix} c_x + f_x X_i^w/Z_i^w \\ c_y + f_y X_i^w/Z_i^w \\ 1 \end{bmatrix} \in \mathbb{R}^3 \tag{3.20}$$

where $\mathbf{K}$ is the intrinsic calibration matrix and $\Pi_3 = [\mathbb{I}_{3\times3}, \mathbf{0}_3]$ projects the 4×4 pose matrix onto the 3×4 space and $\mathbf{e}_3^\top = [0\ 0\ 1]$ extracts the depth component of a 3D point.

Generally, the warped points $\mathbf{p}^w$ do not correspond to an exact pixel coordinate and the intensity function $\mathbf{I}(\mathbf{p}_i)$ of the image $\mathcal{I}$ must be interpolated at the coordinates $\mathbf{p}^w$ to obtain the corresponding intensity [1] as:

$$\mathcal{I}^w(\mathbf{p}^*) = \mathcal{I}(\mathbf{p}^w) \tag{3.21}$$

With the aim to use the intensity functions in the error minimization stage and by assuming a Lambertian surface (A point keeps the same brightness at different viewpoints), the photometric error between two corresponding pixels could be written as:

$$\mathbf{e}_{I_i} = \mathbf{I}^*(\mathbf{p}_i^*) - \mathbf{I}(\mathbf{p}_i^w) \tag{3.22}$$

[1] This is detailed in Section 4.3 where an improved direct method was proposed as part of the contribution of this thesis.

A photometric Jacobian can be obtained by partially deriving 3.22 at each pixel as:

$$\mathbf{J}_I = \nabla \mathbf{I}_i^w \frac{\partial \mathbf{p}_i^w}{\partial \mathbf{P}_i^w} \frac{\partial \mathbf{P}_i^w}{\partial \mathbf{T}(\mathbf{x})} = \begin{bmatrix} \frac{\nabla I_{x_i}^w f_x}{Z_i^*} \\ \frac{\nabla I_{y_i}^w f_y}{Z_i^*} \\ \frac{-\nabla I_{y_i}^w Y_i^* f_y - \nabla I_{x_i}^w X_i^* f_x}{Z_i^{*2}} \\ -\nabla I_{y_i}^w f_y - \frac{Y_i^* \left(\nabla I_{x_i}^w X_i^* f_x + \nabla I_{y_i}^w Y_i^* f_y\right)}{Z_i^{*2}} \\ \nabla I_{x_i}^w f_x - \frac{X_i^* \left(\nabla I_{x_i}^w X_i^* f_x + \nabla I_{y_i}^w Y_i^* f_y\right)}{Z_i^{*2}} \\ \frac{\nabla I_{y_i}^w X_i^* f_y - \nabla I_{x_i}^w Y_i^* f_y}{Z_i^*} \end{bmatrix}^\top \in \mathbb{R}^{1\times 6} \tag{3.23}$$

where $\nabla \mathbf{I}_i^w = [\nabla I_{x_i}^w, \nabla I_{y_i}^w]$ is the gradient of the image. Since direct approaches operate directly on the intensity values of the image, the computational time for feature detection can be saved and the convergence can be improved by employing the Efficient Second order Minimization (ESM) method (See appendix A.3 for more details).

3.5 Hybrid error functions

Hybrid approaches have been useful when color or depth alone are not sufficient for obtaining a correct alignment between RGB-D frames. The aforementioned IRLS pose estimation process can be employed to minimize the geometric or the photometric error function alone, but hybrid methods can obtain the unknown pose by iteratively minimizing over the non-linear error functions simultaneously or by switching the minimization between error functions based on weighting functions. The advantages of hybrid approaches that combine different measurements include increased efficiency, accuracy and robustness for pose estimation processes. However, the contribution of each measurement during the minimization process is usually weighted by a tuning parameter $\boldsymbol{\lambda} = diag(\lambda_1, \lambda_2, \cdots, \lambda_n)$, which scales the relative importance of each measurement (See Figure 3.13).

Various hybrid methods proposed in the literature are those that simultaneously minimize the geometric and photometric error functions in real-time such as [45, 58, 89] for visual SLAM, [62] for 3D tracking and [86, 28] for dense visual odometry. The aforementioned methods differ in how the tuning parameters $\boldsymbol{\lambda} = diag(\lambda_G, \lambda_I)$ are estimated and how the closest points are found. Therefore, a general framework about how the joint minimization is performed and how the tuning parameter has been estimated will be shown in the sections 3.5.2 and 3.5.1, respectively.

In [45], a dense map density is obtained by performing a global optimization and loop closure strategy over RGB-D keyframes. The alignment is obtained between the current image and the keyframe, where as long as the camera stays close enough to the keyframe no drift is accumulated. Robust large-scale 3D modeling has been proposed in [58], where voxel-based and keyframe-based representations have been unified. Both strategies work together in order to update the 3D model by using the RGB-D sensor pose estimation and the current image data. In [89], a dense globally consistent surfel-based in combination with frame-to-model tracking and non-rigid deformation is proposed. It uses model-to-model loop closure detection without post-processing

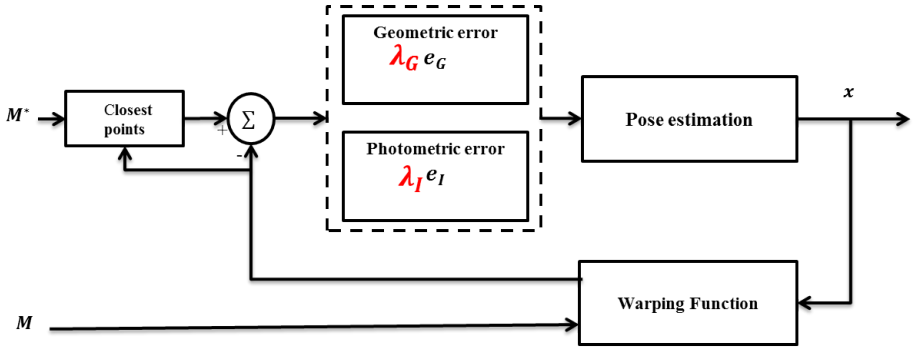


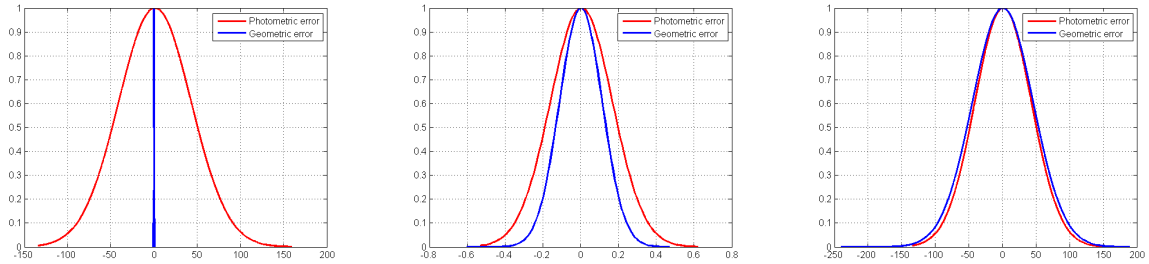
Figure 3.13: Classic hybrid approach diagram. The geometric and photometric errors ($\mathbf{e}_G$ and $\mathbf{e}_I$, respectively) are jointly minimized. A tuning parameter $\boldsymbol{\lambda} = (\lambda_G, \lambda_I)$ weights the contribution of each measurement during the minimization process. The metric measurements are represented in a $i-th$ single vector $\mathbf{M}_i = \left[\mathbf{P}_i^\top \ \mathbf{I}_i\right]^\top \in \mathbb{R}^4$, which contains the 3D Euclidean points $\mathbf{P}$ and their associated intensities $\mathbf{I}$.

refinements. Tracking is performed in [62] for detecting the pose of a human head in 3D. The method optimizes the tracking by optimizing over color and depth through an activation function. Finally, the performance of visual odometry has been demonstrated for 3D reconstruction by using stereo image sequences captured using a Mars Rover robotic platform in [86] (The minimization process has been improved in [28]).

3.5.1 Uncertainty between depth and color

The choice of the scale parameters of $\boldsymbol{\lambda} = diag(\lambda_1, \lambda_2, \cdots, \lambda_n)$ has an influence on estimating the pose. If the parameters are well determined, then it can speed-up the alignment and increase the convergence rate. Various methods have been proposed to choose these parameters, ranging from manual tuning to more complex estimation approaches. Manually fixing $\boldsymbol{\lambda}$ is not optimal nor efficient for real-time applications, and estimating its value can require extra computational cost. Various strategies for RGB-D pose estimation have been cited in [40] and they have been categorized into *adaptive* or *non-adaptive* methods in [41] (depending on how $\boldsymbol{\lambda}$ is estimated).

Adaptive methods are those that determine the tuning parameter at each iteration of the minimization process. These methods weighs the contribution of the geometric error or photometric error during the minimization. Adaptive methods can improve the convergence rate and they can increase the accuracy of the pose estimations. However, it can be computationally expensive to estimate a $\boldsymbol{\lambda} = diag(\lambda_G, \lambda_I)$ for each new RGB-D frame. Adaptive methods increase the importance of the geometric error when the photometric error decreases (or vice versa). There are however more complex methods that compute an adequate scalar factor for each RGB-D image. These methods are usually employed to perform real-time tasks such as 3D visual tracking [62, 5], visual odometry [86, 82, 28], mapping [55] and SLAM [45, 58]. In [62] and [5] the closest points are estimated by searching in a *k*d-tree in 4D-space. The parameter $\boldsymbol{\lambda}$ is a sigmoid function that depends on the 4D-metric distance of matched points, its average and the variance of the Gaussian that balances the contribution of each error. In [55], a similar adaptive method is used as an activation function that performs geometric-based minimization for first coarse iterations and it activates the



(a) $\lambda = 1$ (b) Non-adaptive λ (c) Adaptive λ

Figure 3.14: Influence of the uncertainty factor $\boldsymbol{\lambda}$ on the error function residuals (Equation (4.15)) of the scale coefficient when (a) it is not estimated, (b) when the intensities and 3D points are normalized between $[0, 1]$ [60] and (c) when a λ is estimated at each iteration [86]. The residuals have been fitted into a normalized Gaussian function.

photometric minimization for refinement depending on the geometric and photometric cost functions.The coarse alignment with geometric ICP accelerates the convergence of direct approaches. A method that computes the scale factor as a ratio between the Median Absolute Deviation (MAD) of each error function is proposed in [86], and two applications of this adaptive ratio are found in [28] and [58]. A similar technique is employed in [45, 82], but $\boldsymbol{\lambda}$ is estimated by computing the covariance of the residuals for each point individually, assuming a t-distribution of the error. This improves the convergence rate. However, it is computationally expensive to iteratively compute a $\boldsymbol{\lambda}$ for each pixel.

On the other hand, *non-adaptive* methods estimate $\boldsymbol{\lambda}$ only once and its value is used for all the following iterations. These methods are commonly faster than adaptive methods but they assume that the acquired measurements do not vary much during the iterations. An example of the influence of $\boldsymbol{\lambda}$ in the minimization process is shown in Figure 3.14 where the residuals have been fitted into a normalized Gaussian distribution. It can be observed that the measurements are not in the same order of magnitude, but adaptive and non-adaptive methods scales the contribution of the error functions. Non-adaptive $\boldsymbol{\lambda}$ approaches such as [31, 60, 61, 21, 44, 91, 25] have been proposed. Various methods that include the color as an available criterion for the closest points searching and for minimizing the unified 4D error are found in [31], where the experimentally chosen scale parameter (values between 0 and 1) weighs the contribution of the SIFT alignment or the ICP algorithm, and in [60], where the hue value of intensities is normalized as well as the 3D coordinates that are rescaled to a 0-1 range. Methods that minimize the distance metric form of a 6D vector (3D points + 3 channels of color) have been proposed in [61, 21] and [44]. In the first approach (an extended version of the Generalized-ICP algorithm [78]), a ratio between minimum and maximum values of both, photometric and geometric distances, is computed and $\boldsymbol{\lambda}$ is chosen experimentally between the estimated range. A method called ICP-GCT is introduced in [21], where $\boldsymbol{\lambda}$ is equal to the median of distances for each color component in L*a*b space and it is adjusted according to data quality, which depends on acquisition noise. In [44], $\boldsymbol{\lambda}$ is chosen by the color model, where the intensity of light, hue and the saturation is conveyed by the color channels. The method proposed in [91] can perform global convergence when real-time pose esti-

mation is not required, the basic idea is to alternate between optimizing color and optimizing the extrinsic matrix, where $\boldsymbol{\lambda}$ balances the strength of the deformation function over the image plane. A method that uses image-based feature extraction is proposed in [25], where the closest points in the image are adapted when the pose and the 3D point locations are refined. The scale $\boldsymbol{\lambda}$ is computed at initialization for each error by inverting the variance of the geometric and photometric error. Furthermore, real-time RGB-D Simultaneous Localization and Mapping (SLAM) using a non-adaptive scale factor is found in [87, 89, 88] where the $\boldsymbol{\lambda}$ was also set empirically to reflect the difference in metrics used for color and depth costs.

Table 3.1: Examples of different strategies that estimates a tuning parameter $\boldsymbol{\lambda}$ used for real-time SLAM

Method	Type of $\boldsymbol{\lambda}$	Function
Normalize measurements [60]	non-adaptive	$\lambda_I = \mathbf{I}(\mathbf{p_i})/255$
Experimentally chosen [89]	non-adaptive	$\lambda_I = [0-1]$
MAD ratio [86]	adaptive	$\lambda_I = MAD(\mathbf{e_G})/MAD(\mathbf{e_I})$
Covariance matrix for each pixel [45]	adaptive	$\lambda_I = cov(\mathbf{e_G}, \mathbf{e_I})$
Sigmoid function [62]	adaptive	$\lambda_G = \lambda(d)$ $\lambda_I = 1 - \lambda(d)$ where $\lambda(d) = 1/1 + e^{-c(\bar{d}-d_G)}$
Activation function [55]	adaptive	$\lambda_G = \mu(\mathbf{x})$ $\lambda_I = 1 - \mu(\mathbf{x})$ where $\mu(\mathbf{x}) = k_1\mathbf{1}(C_I/C_D)$

3.5.2 Joint error for RGB-D pose estimation

The generated errors between two sets of extended measurements (color + depth) can be jointly minimized to estimate the pose since the color and depth pose estimation pipeline shares too much similarity. So-called hybrid methods have been introduced to minimize both error functions simultaneously, where a 3D Euclidean point $\mathbf{P}_i \in \mathbb{R}^3$ is associated with an unique intensity $\mathbf{I}_i$ by weighting each contribution with an uncertainty factor $\boldsymbol{\lambda}$. Consider here two augmented point clouds obtained at different times. Let $\mathbf{M}^*$ be the reference point cloud and $\mathbf{M}$ be the current point cloud measurements. The hybrid error function for the $i-th$ joint measurement vector, $\mathbf{e}_{G_i}$ and $\mathbf{e}_{I_i}$ (geometric and photometric error, respectively) can be represented as:

$$\mathbf{e}_{H_i} = \boldsymbol{\lambda} \begin{bmatrix} \mathbf{e}_{G_i} \\ \mathbf{e}_{I_i} \end{bmatrix} = \boldsymbol{\lambda} \begin{bmatrix} \mathbf{P}_i^* - f(\mathbf{P}_i, \mathbf{x}) \\ \boldsymbol{I}_i^* - f(\boldsymbol{I}_i, \mathbf{x}) \end{bmatrix} \in \mathbb{R}^6 \tag{3.24}$$

where $\boldsymbol{\lambda} = diag(\lambda_X, \lambda_Y, \lambda_Z, \lambda_I) = \begin{bmatrix} \boldsymbol{\lambda}_G & 0 \\ 0 & \lambda_I \end{bmatrix}$ are the tuning parameters which weigh the contribution of each error function. For 3D Euclidean points, $\boldsymbol{\lambda}_G = diag(\lambda_X, \lambda_Y, \lambda_Z)$.

The alignment between the measurement vector $\mathbf{M}^* = [\mathbf{P}^* \ \mathbf{I}^*]^\top$ and $\mathbf{M} = [\mathbf{P} \ \mathbf{I}]^\top$ is found by iteratively minimizing the error function (3.24). This involves transforming the current dataset $\mathbf{M}$ with the estimated pose $\mathbf{x}$ with the transformation function represented here as: $f(\cdot)$.

The non-linear error shown in (3.24) is minimized iteratively using a Gauss-Newton approach to compute the unknown 6DOF pose parameter with increments given by:

$$\mathbf{x} = -(\mathbf{J}^\top \mathbf{W} \mathbf{J})^{-1} \mathbf{J}^\top \mathbf{W} \begin{bmatrix} \boldsymbol{\lambda_G} \mathbf{e}_G \\ \lambda_I \mathbf{e}_I \end{bmatrix} \in \mathbb{R}^4 \tag{3.25}$$

where $\mathbf{J} = [\ \mathbf{J}_G^\top \ \mathbf{J}_I^\top \ ]^\top$ is the stacked Jacobian matrix obtained by deriving the error function (3.24), and the weight matrix $\mathbf{W} = diag(w_1, w_2, \cdots, w_n)$ contains the stacked weights associated with each set of coordinates separately obtained by M-estimation [36]. The geometric and photometric tuning parameters are defined as $\boldsymbol{\lambda}_G$ and λ_I, respectively.

The previously cited hybrid strategies [45, 58, 89, 62, 86] are based on the ICP Point-to-Plane algorithm [16] and a direct image-based method [38] whilst minimizing the error simultaneously. Generally, these strategies minimize the following error function:

$$\mathbf{e}_{H_i} = \begin{pmatrix} \lambda_G \left(\mathbf{N}_i^{*\top} \left(\mathbf{P}_i^m - \mathbf{P}_i^w \right) \right) \\ \lambda_I \left(\boldsymbol{I}_i^m - \boldsymbol{I}_i^w \right) \end{pmatrix} \in \mathbb{R}^4 \tag{3.26}$$

where $\mathbf{P}_i^w = \Pi_3 \widehat{\mathbf{T}} \mathbf{T} \left(\mathbf{x} \right) \overline{\mathbf{P}_i} \in \mathbb{R}^3$ is the warped 3D point and $\boldsymbol{I}_i^w = \boldsymbol{I} \left(\omega(\widehat{\mathbf{T}} \mathbf{T}(\mathbf{x}), \overline{\mathbf{P}}_i^*) \right)$ is the warped image. $\Pi_3 = [\mathbf{1}, \mathbf{0}] \in \mathbb{R}^{3 \times 4}$ is the projection matrix and $\mathbf{N}_i^* \in \mathbb{R}^3$ is the surface normal for each homogeneous 3D point $\overline{\mathbf{P}}_i \in \mathbb{R}^4$. The closest image intensity is found by interpolating the current intensity function at the warped pixel coordinates. Therefore, the corresponding intensities can be estimated as: $\mathbf{I}_i^w(\mathbf{p}_i^*) = \mathbf{I}_i(\mathbf{p}_i^w) \in \mathbb{Z}^+$. The 3D correspondences and the matched intensities are defined as $\mathbf{P}_i^m$ and $\mathbf{I}_i^m$, respectively. Note that in (3.26) the geometric tuning parameter is a scalar as: $\lambda_G = det(\boldsymbol{\lambda_G})$.

Hybrid strategies do not necessarily consider the color and depth simultaneously when computing the closest points. All the methods [45, 58, 89, 62, 86] perform the closest point searching separately for both color and depth except for [62, 61], which estimate the closest points using a *k*d-tree (*k*-dimensional) in a 4-dimensional space (3D Euclidean points + intensity) and a 6-dimensional space (3D Euclidean points + 3 channels of color), respectively. Finding the closest points by considering the fused information vector increases the accuracy of finding the true nearest neighbours, however, this approach requires an efficient search in a higher dimensional space.

CHAPTER 4

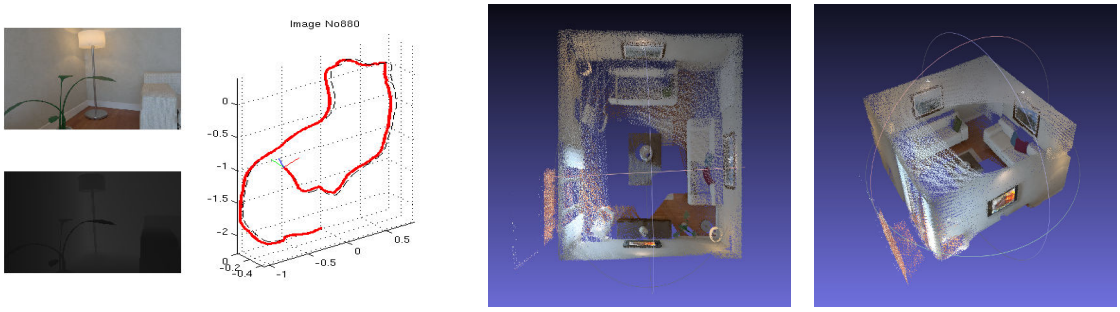
LOCAL POINT-TO-HYPERPLANE ICP

4.1 Introduction

The objective of this chapter is to introduce a method that can perform both, matching and minimization, by using a fused measurement vector that contains depth and color. The fusion of measurements leads to generate a surface in higher dimensions formed by n-dimensional points. In order to introduce the new Point-to-hyperplane ICP method, the error function will first be introduced, the proof of the invariance will be shown and finally results in real and simulated environments will demonstrate that the proposed method can further improve the convergence domain and speed up the alignment between RGB-D frames, whilst maintaining the robust and accurate properties of hybrid approaches. Particularly, a joint error function will be generated by extending the Point-to-plane distance to higher dimensions and a matching stage will be performed by considering geometry and color. This generates a n-dimensional space that has additional degrees of freedom and where a computed normal spans both, geometry and color, and where it was observed an invariance to a scale parameter.

4.2 Point-to-hyperplane

As demonstrated in [16], the ICP algorithm can be improved if the closest points are estimated via the Point-to-plane distance (P2Pl). However, the P2Pl ICP strategy consider only geometric 3D points to compute the closest points and color has been limited to find more true correspondences in the matching stage. As mentioned already, the geometric P2Pl ICP error function has been minimized by hybrid methods simultaneously with the photometric term as in (5.19). In order to extend the P2Pl ICP to higher dimensions for hybrid RGB-D pose estimation, the computation of the shortest distance between two surfaces in higher dimensions is estimated by considering the hyperplane formed by n-dimensional points.



(a) Visual odometry (b) Top view (c) Perspective view

Figure 4.1: Examples of performance of the Point-to-hyperplane method in the sequence living room 2 [29]. (a) Visual odometry was performed on a frame-to-frame RGB-D sequence, the red line show the trajectory estimated by the proposed method and the black line is the groundtruth, (b) and (c) 3D reconstruction obtained by the proposed method.

An hyperplane will be defined here as a surface formed between n-dimensional points (e.g. $\mathbf{M}_1 \in \mathbb{R}^n$ and $\mathbf{M}_2 \in \mathbb{R}^n$) that extends to the infinity. The general equation of an affine hyperplane (all dimensions are in the same scale) can be written as follows where at least one of the coefficients a_n is non-zero:

$$a_1[\mathbf{M}_1]_1 + a_2[\mathbf{M}_1]_2 + \cdots + a_n[\mathbf{M}_i]_n + d = 0 \tag{4.1}$$

where $d = -a_1[\mathbf{M}_2]_1 - a_2[\mathbf{M}_2]_2 - \cdots - a_n[\mathbf{M}_2]_n$ and $[\cdot]_n$ is the operator that extracts the n-th coordinate of the measurement vector.

Based on the Point-to-plane method for 3D points, the shortest distance from a point to a plane is along a line perpendicular to the plane. Therefore, the distance from a n-point $\mathbf{M}_i$ to the hyperplane is along a line perpendicular to the n-plane. To calculate an expression for this distance in terms of the n-dimensional space, the definition of the normal vector in higher dimensions will be established.

The Point-to-hyperplane error function can be defined as follows:

$$\mathbf{e}_{H_i} = \boldsymbol{\lambda}\mathbf{N}_i^{*\top}\left(\mathbf{M}_i^* - f(\mathbf{T}(\mathbf{x}), \mathbf{M}_i)\right) \in \mathbb{R} \tag{4.2}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^{n\times n}$ balances the uncertainty between the different measurements [1].

For the n-dimensional space, at least $n-1$ points are needed to compute the normal. The normals can be computed by performing the n-dimensional cross product between points, but other strategies can be equally used. In the Generalized-ICP strategy [78], the PCA algorithm is employed for estimating the normals. The covariance matrices of the measurements are estimated, where the eigenvector associated with their lowest eigenvalue is considered as the normal. Recently, an alternative solution is provided in [4]. The normals are fast and accurately computed by performing the Prewitt operator on the projected spherical coordinates $\mathbf{m}_i = [r\ \theta\ \phi]^\top$ onto a spherical range image (SRI). A SRI stores an image observed from a single viewpoint, providing a 2.5D

[1] In 4.2, $\mathbf{N}_i^{*\top} \in \mathbb{R}^{n\times 1}$ and $(\mathbf{M}_i^*, \mathbf{M}_i) \in \mathbb{R}^n$

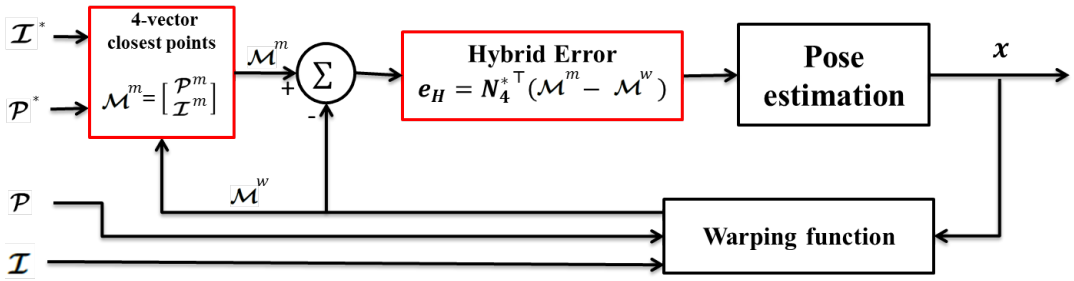


Figure 4.2: Point-to-hyperplane approach diagram. The closest points and the minimization error can be estimated by optimizing over the 4D vector. The method have demonstrated to be invariant to a tuning parameter $\boldsymbol{\lambda}$ by projecting the normal onto the formed hyperplane (Read text for demonstration).

representation of the scene. The Prewitt operator ∇ is widely used for edge detection algorithms by computing the gradient of the SRI, which is a function $s(\theta, \phi)$ that represents a distance.

The 3D spherical coordinates (azimuth θ, elevation ϕ and range r) are related with the Cartesian coordinates as:

$$\mathbf{m}_i = \begin{bmatrix} r \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arctan(x/z) \\ \arcsin(y/\sqrt{x^2 + y^2 + z^2}) \end{bmatrix} \tag{4.3}$$

and the normals estimated as: $\mathbf{N}_i = \nabla s(\theta, \phi)$.

For purposes of this thesis, the n-vector is defined as $\mathbf{M}_i = [\mathbf{P}_i^\top \ \mathbf{I}_i]^\top \in \mathbb{R}^4$. The normals $\mathbf{N}^*$ are computed on the reference 4D measurements vector $\mathbf{M}^*$. The 4-normal is estimated similarly to the 3D case, where the closest points are selected from their associated neighbouring pixels in a RGB-D frame (Figure 4.4). It was observed that the projection of the error onto the normal direction has the effect of canceling out the effect of $\boldsymbol{\lambda}$ between the geometric and photometric error since the direction of the normal is invariant to the value of the tuning parameter. This is demonstrated in the following section.

4.2.1 Invariance to a relative scale

To simplify the problem, consider an example scenario here with 2 sets of 2D measurements taken at different times, given as $\mathbf{M}^* = [X^* \ Y^*]^\top$ and $\mathbf{M} = [X \ Y]^\top$. Each measurement in the vector is taken to have a different scale (e.g. centimeters and millimeters). The measurements are scaled to the same order of magnitude by the tuning parameter $\boldsymbol{\lambda}$. The equation of the tangent line $a_1 x + a_2 y + c = 0$ at a i-th reference point (which plays the role of a plane or hyperplane in higher dimensions) can be obtained from the definition of the slope of a line:

$$\frac{y - \lambda_Y Y_i^*}{x - \lambda_X X_i^*} = \frac{\lambda_Y (Y_{i+1}^* - Y_i^*)}{\lambda_X (X_{i+1}^* - X_i^*)} \tag{4.4}$$

that can be written as:

$$\lambda_Y (Y_i^* - Y_{i+1}^*) x + \lambda_X (X_{i+1}^* - X_i^*) y + c = 0 \tag{4.5}$$

where $c = \lambda_X\lambda_Y(X^*_{i+1}Y^*_i - Y^*_{i+1}X^*_i)$ and $i+1$ represents any other point in the same dataset (usually the closest to i).

The generalized normal $\langle a_1, a_2\rangle = \left\langle \lambda_Y(Y^*_i - Y^*_{i+1}), \lambda_X(X^*_{i+1} - X^*_i)\right\rangle$ can be given by the following equation:

$$\overrightarrow{\mathbf{N}^*_i} = \det(\boldsymbol{\lambda}) \cdot \boldsymbol{\lambda}^{-1}\mathbf{V}^*_i \tag{4.6}$$

where $\mathbf{V}^*_i = \begin{bmatrix} Y^*_i - Y^*_{i+1} & X^*_{i+1} - X^*_i \end{bmatrix}^\top = [[N^*_i]_1 \ [N^*_i]_2]^\top$.

The projection of the point-to-point error $\overrightarrow{\mathbf{M}^*_i\mathbf{M}_i} = \boldsymbol{\lambda}(\mathbf{M}_i - \mathbf{M}^*_i)$ onto the normal direction $\overrightarrow{\mathbf{N}^*_i}$ defines the distance of a point to a line (Figure 4.3(a)). It is clear that the error function can be computed just as: $\mathbf{e}_{H_i} = \overrightarrow{\mathbf{N}^*_i}^\top \boldsymbol{\lambda}\,(\mathbf{M}^*_i - \mathbf{M}_i)$ (proof in Annexes). Substituting $\overrightarrow{\mathbf{N}^*_i}$ into this error function allows to rewrite it as:

$$\mathbf{e}_{H_i} = \lambda_X\lambda_Y\left([N^*_i]_1\,(X_i - X^*_i) + [N^*_i]_2\,(Y_i - Y^*_i)\right) \in \mathbb{R}$$

where is demonstrated that the tuning parameter $\lambda = det(\boldsymbol{\lambda})$ has no effect on the minimization process since it has no influence on the direction of the normal and it scales its magnitude only.

Consider now a more complex set of 3D hybrid points, where each point is formed as $\mathbf{M}^*_i = [X^*_i\ Y^*_i\ I^*_i]$. A 3D normal is obtained by performing the cross product between two vectors (which are scaled by $\boldsymbol{\lambda}$) as: $\mathbf{N}^*_i = \det(\boldsymbol{\lambda}) \cdot \boldsymbol{\lambda}^{-1}\left(\mathbf{V}^{ik} \times \mathbf{V}^{il}\right)$, where $\mathbf{V}^{ik}$ and $\mathbf{V}^{il}$ are defined as the vectors pointing to the $k-th$ and the $l-th$ closest point to $\mathbf{M}^*_i$ that lie on the reference dataset as $\mathbf{V}^{ik} = \boldsymbol{\lambda}\,(\mathbf{M}^*_k - \mathbf{M}^*_i)$ and $\mathbf{V}^{il} = \boldsymbol{\lambda}\,(\mathbf{M}^*_l - \mathbf{M}^*_i)$, respectively (See Figure 4.3(b)). Therefore, the 3D normal at a i-th reference point is:

$$\mathbf{N}^*_i = \det(\boldsymbol{\lambda}) \cdot \boldsymbol{\lambda}^{-1}\left(\mathbf{V}^{ik} \times \mathbf{V}^{il}\right) = \begin{bmatrix} \lambda_Y\lambda_I\left(V^{ik}_Y V^{il}_I - V^{ik}_I V^{il}_Y\right) \\ \lambda_X\lambda_I\left(V^{ik}_I V^{il}_X - V^{ik}_X V^{il}_I\right) \\ \lambda_X\lambda_Y\left(V^{ik}_X V^{il}_Y - V^{ik}_Y V^{il}_X\right) \end{bmatrix} = \begin{bmatrix} \lambda_Y\lambda_I N_X \\ \lambda_X\lambda_I N_Y \\ \lambda_X\lambda_Y N_I \end{bmatrix} \tag{4.7}$$

Therefore, a 3D normal at the i-th reference point is obtained as follows:

$$\mathbf{N}^*_i = \begin{bmatrix} \lambda_Y\lambda_I\left(V^{ik}_Y V^{il}_I - V^{ik}_I V^{il}_Y\right) \\ \lambda_X\lambda_I\left(V^{ik}_I V^{il}_X - V^{ik}_X V^{il}_I\right) \\ \lambda_X\lambda_Y\left(V^{ik}_X V^{il}_Y - V^{ik}_Y V^{il}_X\right) \end{bmatrix} = \begin{bmatrix} \lambda_Y\lambda_I N_X \\ \lambda_X\lambda_I N_Y \\ \lambda_X\lambda_Y N_I \end{bmatrix} \tag{4.8}$$

The error function generated between two sets of 3D hybrid measurements can be defined then as:

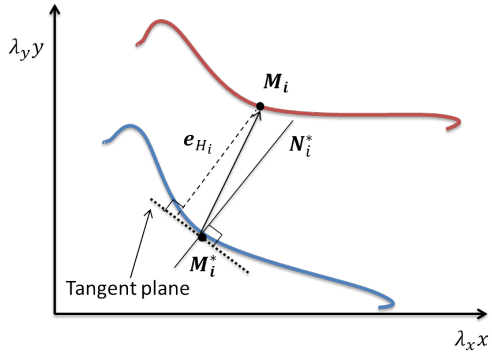
$$\mathbf{e}_{H_i} = \det(\boldsymbol{\lambda})\boldsymbol{\lambda}^{-1}\left(\mathbf{V}^{ik} \times \mathbf{V}^{il}\right)^\top \boldsymbol{\lambda}(\mathbf{M}^*_i - \mathbf{M}_i) = \det(\boldsymbol{\lambda})\left(\mathbf{V}^{ik} \times \mathbf{V}^{il}\right)^\top (\mathbf{M}^*_i - \mathbf{M}_i) \tag{4.9}$$

The estimation of the normal presented in (4.8) can be easily extended to higher dimensions. The n-dimensional normal is estimated by performing the n-dimensional cross product between the $n-1$ vectors such as:

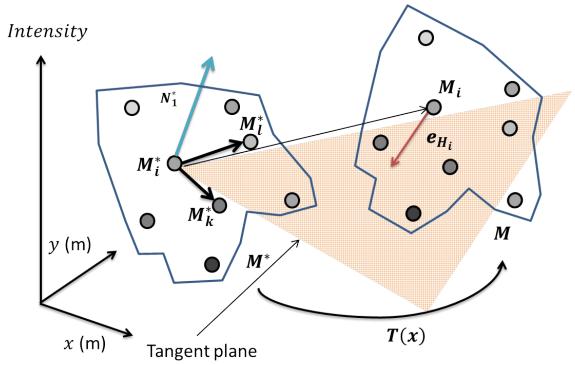
$$\mathbf{N}^*_i = \det(\boldsymbol{\lambda}) \cdot \boldsymbol{\lambda}^{-1}\left(\mathbf{V}^1 \times \mathbf{V}^2 \times \cdots \times \mathbf{V}^{n-1}\right) \in \mathbb{R}^n \tag{4.10}$$

The n-dimensional error function [41] between two sets of measurements can be defined as: $\mathbf{e}_{H_i} = \det(\boldsymbol{\lambda})\boldsymbol{\lambda}^{-1}\left(\mathbf{V}^1 \times \mathbf{V}^2 \times \cdots \times \mathbf{V}^{n-1}\right)^\top \boldsymbol{\lambda}(\mathbf{M}^*_i - \mathbf{M}_i)$, which can be re-written as:

$$\mathbf{e}_{H_i} = \det(\boldsymbol{\lambda})\left(\mathbf{V}^1 \times \mathbf{V}^2 \times \cdots \times \mathbf{V}^{n-1}\right)^\top (\mathbf{M}^*_i - \mathbf{M}_i) \tag{4.11}$$

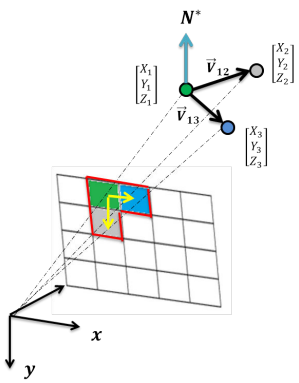


(a) 2D hybrid points

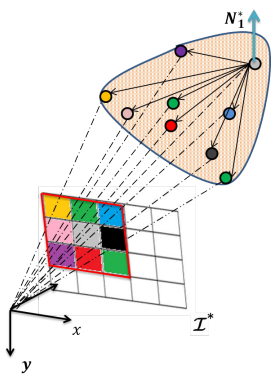


(b) 3D hybrid points

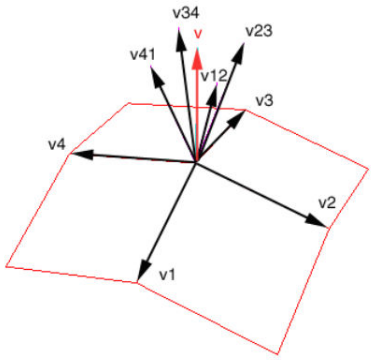
Figure 4.3: Principle of the point-to-hyperplane applied in (a) 2 dimensions Point-to-line and (b) 3 dimensions Point-to-plane. It should be noted that the axis x and y are not in the same scale for the 2D case. A tuning parameter λ is added in order to demonstrate the invariance of the Point-to-hyperplane method in the 2D case. The distance $\mathbf{e}_{H_i}$ is the result of projecting the vector $\mathbf{M}_i^*\mathbf{M}_i$ onto the normal direction $\mathbf{N}_i^*$.



(a) 3 dimensions



(b) 4 dimensions



(c) Resultant normal

Figure 4.4: The neighbours of a central pixel in a 3×3 window on a image are employed to associate its closest 3D points and estimate the resultant normal to the formed plane (or hyperplane).

where $\det(\boldsymbol{\lambda}) = \lambda_1\lambda_2\cdots\lambda_n$.

For the 4-dimensional case presented here, the normal is obtained as:

$$\mathbf{N}_i^* = \begin{bmatrix}\lambda_Y\lambda_Z\lambda_I N_{X_i} & \lambda_X\lambda_Z\lambda_I N_{Y_i} & \lambda_X\lambda_Y\lambda_I N_{Z_i} & \lambda_X\lambda_Y\lambda_Z N_{I_i}\end{bmatrix}^\top$$

where $N_{X_i} = [N_i^*]_1, N_{Y_i} = [N_i^*]_2, N_{Z_i} = [N_i^*]_3$ and $N_{I_i} = [N_i^*]_4$ are the components of the normal which can be computed by performing the 4D cross product of the nearest 4D points to a i-th 4D vector $\boldsymbol{\lambda}\mathbf{M}_i^*$. Therefore, the equation (4.2) can be rewritten for the 4-dimensional space as follows:

$$\mathbf{e}_{H_i} = \det(\boldsymbol{\lambda})\left(N_{X_i}(X_i^* - X_i^w) + N_{Y_i}(Y_i^* - Y_i^w) + N_{Z_i}(Z_i^* - Z_i^w) + N_{I_i}(I_i^* - I_i^w)\right)$$

where $\det(\boldsymbol{\lambda}) = \lambda_G^3\lambda_I$. It can be seen that the tuning parameter is just a scalar that has no effect on the direction of the normal for any dimension.

Therefore, the following lemma is proposed, which is the cornerstone of the proposed method and the main contribution of the research during this thesis.

Lemma. *The integrated error $\mathbf{e}_H$ in n-dimensions is invariant to the relative scale λ if it is minimized by a Point-to-hyperplane method.*

$$\mathbf{e}_{H_i} = \mathbf{N}_i^{*\top}\boldsymbol{\lambda}\left(\mathbf{M}_i^* - f(\mathbf{M}_i, \mathbf{x})\right) = det(\boldsymbol{\lambda})\mathbf{N}_i^{*\top}\left(\mathbf{M}_i^* - f(\mathbf{M}_i, \mathbf{x})\right) \tag{4.12}$$

Furthermore, the calculation of the unit normal vector demonstrates clearly the invariance to any factor $\boldsymbol{\lambda}$. In order to demonstrate this, the normalization of the normal in (4.12) is calculated such as:

$$\widehat{\mathbf{N}_i^*} = \begin{bmatrix}\frac{a_1}{\sqrt{a_1^2+a_2^2+\cdots+a_n^2}} & \frac{a_2}{\sqrt{a_1^2+a_2^2+\cdots+a_n^2}} & \cdots & \frac{a_n}{\sqrt{a_1^2+a_2^2+\cdots+a_n^2}}\end{bmatrix}^\top \tag{4.13}$$

where $a_1 = det(\boldsymbol{\lambda})N_1^*, a_2 = det(\boldsymbol{\lambda})N_2^*, \cdots, a_n = det(\boldsymbol{\lambda})N_n^*$.

The computation of the normals can be performed by considering the neigbouring n-points, where at least n-1 points are needed to estimate the normal. In the case of RGB-D images, the nearest neighbours to a central pixel in the image are considered to find its associated 3D point. In case of 4 dimensions, at least three 4D points should be considered to estimates the 4D normal. For purposes of this thesis, a 3×3 window was considered for the experiments, which estimate the resultant normal of the 8 nearest neighbours to a central pixel (Figure 4.4). This guarantees that at least three 4D points will be found to estimate the normal. As mentioned in section 3.2, the presence of non valid data can appear in the color or depth image. This can generate an erroneous estimation of the direction of the normal. A 3×3 window can guarantee that at least three 4D points will be found. Otherwise, the point is discarded. Two strategies for computing the normal were compared, n-dimensional cross product and PCA, where PCA has shown better accuracy in performing visual odometry and 3D reconstruction. For the real-time SLAM application, however, the cross product in a 2×2 window is employed to speed up the performance since the computation of the normals using the PCA strategy is computationally expensive. It should be mentioned here that the computational cost is linear with the size of this window, but the accuracy of the normal direction is increased.

Finally, the pose vector $\mathbf{x}$ can be estimated by iteratively minimizing the error function that projects the Point-to-point distance onto the normal direction as:

$$\mathbf{x} = -(\mathbf{J}^\top \mathbf{W}\mathbf{J})^{-1}\mathbf{J}^\top \mathbf{W}\mathbf{e}_H \tag{4.14}$$

where $\mathbf{J}$ is the stacked Jacobian represented as $[\mathbf{J}_G\ \mathbf{J}_I]$.

4.2.2 Results

In order to evaluate the Point-to-hyperplane ICP method, some parameters are established. All the experiments were performed on both, real and synthetic 640×480 RGB-D grayscale images in MATLAB. Furthermore, visual SLAM in real-time was performed in C++ based on CPU only.

- A multi-resolution pyramid with 3 levels was used to improve computational efficiency (resolution: 160×120 at the top), where a pose is estimated at the top of the pyramid and the estimated transformation is employed to initialize the transformation in the next level (See Appendix A.2 for details).
- The minimization process can be stopped by two criteria: an established maximum number of iterations (200) or if the norm of the pose parameter is less than 1×10^{-6} in rotation (deg) and 1×10^{-5} in translation (m).
- To reject outliers, the Huber influence function was employed in only one M-estimator in the Point-to-hyperplane ICP method (as opposed to [58, 45] where the M-estimation is performed separately). This allows the use of different minimization functions not necessarily corresponding to normally distributed data.
- All the experiments were validated on a PC with Ubuntu 14.04, Inter core i7-4770K and 16GB ram.

The Point-to-hyperplane method is compared in this paper with variants (different strategies to estimate the uncertainty factor $\boldsymbol{\lambda}$) of the error function proposed in [58] and shown in (4.15). For these experiments, different strategies in the literature were compared by replacing in the error function the estimation of the tuning parameter $\boldsymbol{\lambda}$. It should be mentioned here that during the experiments, no other refinement post-processing was performed (e.g. loop closure detection, bundle adjustment, etc.). For the compared strategies, the classic Point-to-plane [16] approach is employed to minimize the geometric term and a direct method for the photometric term as:

$$\mathbf{e}_{H_i} = \boldsymbol{\lambda} \begin{pmatrix} \left(\widehat{\mathbf{R}}\mathbf{R}(\mathbf{x})\mathbf{N}_i^*\right)^\top \left(\mathbf{P}_i^m - \Pi_3 \widehat{\mathbf{T}}\mathbf{T}\left(\mathbf{x}\right)\overline{\mathbf{P}_i^*}\right) \\ \mathbf{I}_i\left(w(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \mathbf{P}_i^*)\right) - \mathbf{I}_i^*\left(\mathbf{p}_i^*\right) \end{pmatrix} \in \mathbb{R}^4 \tag{4.15}$$

where the first and second row correspond to the geometric and photometric error function, respectively. $\mathbf{P}_i^m \in \mathbb{R}^3$ is the closest 3D Euclidean point in the current cloud, $\hat{\mathbf{R}} \leftarrow \hat{\mathbf{R}}\mathbf{R}(\mathbf{x})$ is the incremental update of rotations, $\mathbf{N}_i^* = [N_{x_i}\ N_{y_i}\ N_{z_i}]^\top$ are the normals of the reference points

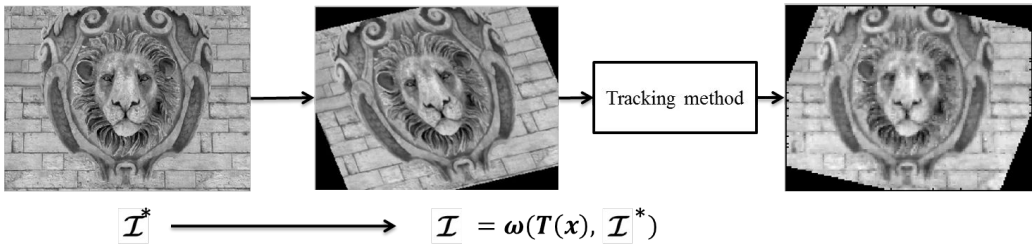


Figure 4.5: Synthetic RGB-D frame. The warping function is used to generate a new image, which is considered as the current frame. At the solution, the estimated transformation $\mathbf{T}(\mathbf{x}^*)$ is the same as the initial transformation $\mathbf{T}(\mathbf{x})$, where $\mathbf{T}(\mathbf{x}^*)\mathbf{T}(\mathbf{x})^{-1} = \mathbb{I}$, and gets values close to the identity matrix $\mathbb{I}$.

(which can be pre-computed in order to improve the computational time) and $\Pi_3 = [\mathbf{1}, \mathbf{0}] \in \mathbb{R}^{3\times4}$ is the projection matrix. For the purposes of this thesis, the photometric term is minimized by using the Second Order Minimization (ESM) method [7] (See apendix A.3). The reference normals $\mathbf{N}^*$ are rotated since the reference 3D points are transformed by $\mathbf{T}(\mathbf{x})$.

For the comparisons, the performance of the Point-to-hyperplane method is compared with three different strategies that compute a non-adaptive or adaptive λ:

1. The intensities are normalized $\lambda_I = I_i/255$ (non adaptive) and $\boldsymbol{\lambda_G} = ones(3 \times 3)$.

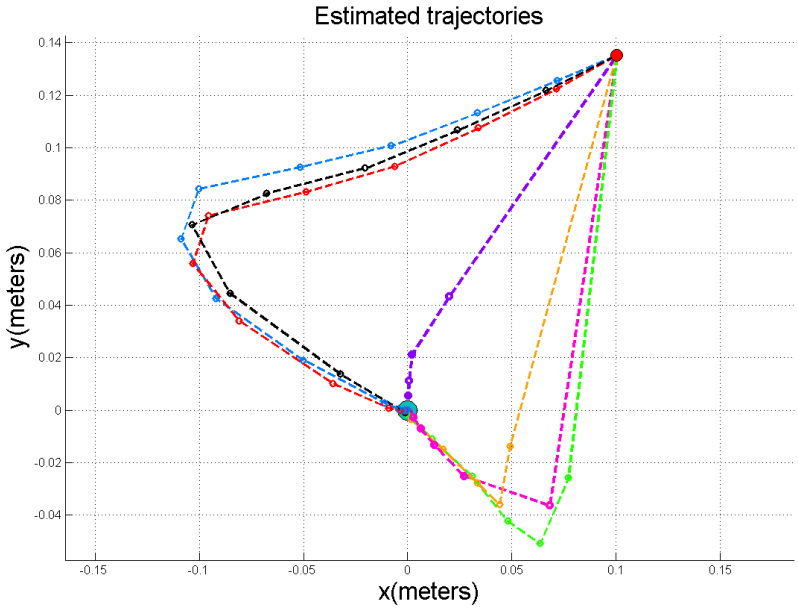
2. An adaptive λ as in [86], where the scale parameter is the ratio between the Median Absolute Deviation (MAD) of the errors $\lambda_G = MAD(\mathbf{e}_I)/MAD(\mathbf{e}_G)$ and $\lambda_I = ones(3)$.

3. The covariance matrix of the metric errors as $\boldsymbol{\lambda} = cov(e_G, e_I)$. For this last strategy, the T-distribution was employed to reject outliers as in [45].

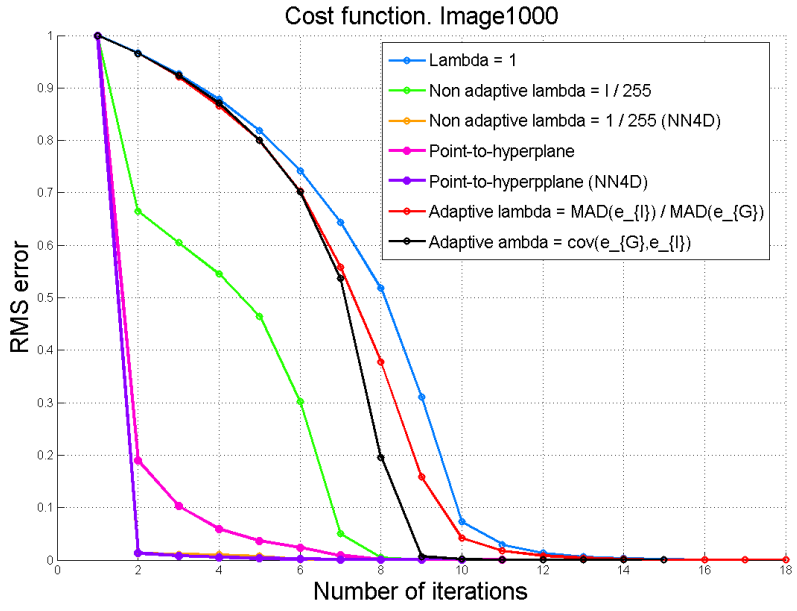
The minimization of the error presented in (4.15) will also be compared with a $\lambda = 1$ (λ is not estimated) in order to demonstrate that the parameter $\boldsymbol{\lambda}$ can improve the hybrid methods if it is well estimated. Additionally, the estimation of the closest points were also done by searching in a *k*d-tree (Labeled as NN4D and NN6D). This strategy demonstrated a better performance while aligning the frames when they are not close enough, but increasing the computational cost.

Simulated environment

The performance of the aforementioned strategies were compared in a synthetic environment. The motivation for using synthetic data is that the generated images provide a groundtruth for the evaluation since the transformation between the frames is known. For this experiment, a reference RGB-D image is transformed via the warping function (an example can be seen in Figure 4.5, where a rotation along the z axis was applied). After performing the RGB-D tracking, the correct alignment provides a transformation matrix close to the initial pose. For the comparisons, 1000 synthesized RGB-D frames were generated with random poses and, in order to evaluate the performance when outliers are present, the transformed RGB-D image was perturbed with Gaussian noise ($\sigma = 0.01$). The random poses were generated in MATLAB by using normally distributed pseudorandom numbers through the function $randn$ as $\mathbf{x} = [randn(1,3)/360 * randn(1/30)/pi/180]$.



(a) Estimated trajectories



(b) Cost functions

Figure 4.6: Example of the estimated trajectories between a current and a reference frame in the transformation space. (a) The red and green dot indicates the initial and final pose respectively. The Point-to-hyperplane method improves the other hybrid methods by obtaining more direct trajectories and less number of iterations (b). A similar performance was observed in the 1000 synthetic frames that were equally tested. The label NN4D indicate that the nearest neighbours were obtained by using a kd-tree in the first iteration only by considering 4 dimensions.

Table 4.1: Averages in time and in number of iterations until convergence for 1000 synthesized images at random poses. The legend NN4D or NN6D indicates that the closest points were estimated in the first iteration only by searching the nearest neighbours in the 4D or 6D kd-tree. The time shown was measured in the iterative loop.

Method	# Iterations	Time (sec)
hybrid ($\boldsymbol{\lambda} = ones$)	157.668	2.046
hybrid + non-adaptive $\lambda_I = I_i/255$	124.419	1.598
hybrid + non-adaptive $\lambda_I = I_i/255$ (NN4D)	116.609	1.563
hybrid + adaptive $\lambda_G = MAD(e_I)/MAD(e_G)$ [86]	154.966	2.010
hybrid + adaptive $\boldsymbol{\lambda} = cov(\mathbf{e}_G, \mathbf{e}_I)$ [45]	155.455	6.079
Point-to-hyperplane (3D points + Intensity)	**48.038**	**0.531**
Point-to-hyperplane (NN4D)	**13.224**	**0.191**
Point-to-hyperplane (3D points + RGB)	96.79	2.1572
Point-to-hyperplane (NN6D)	79.439	1.7978

Averages in time and number of iterations obtained during the experimentation are shown in Table 4.1. The time shown, however, does not consider the computation of the normals or construction of the *k*d-tree in the reference image. Therefore, for the employed RGB-D reference image in this experiment (lyon), the estimation of the normals were obtained in 9.33 seconds by considering a 3×3 window and the construction of the *k*d-tree obtained 0.0056 seconds for its construction in MATLAB at the top of the pyramid. For this experiment, the matching points obtained by the *k*d-tree were estimated by considering 4D (3D points + intensity) and 6D points (3D points + color) in the first iteration only (labeled as NN4D and NN6D, respectively). It was observed that this closest point matching strategies reduces the number of iterations and convergence time. The chances of finding the true nearest neighbours increases when more dimensions are considered. This is due to the fact that more characteristic of a point can better constrain the closest point searching. This is useful when the overlapping area between RGB-D frames is not large enough. However, the searching of the closest points by using *k*d-trees requires extra computational time as it can be seen in the last two rows of Table 4.1. Therefore, for purposes of this thesis, only 4 dimensions were considered (this balances the computational cost and accuracy while estimating the pose offline).

In Figure 4.6(a) an example of the estimated trajectories during convergence by the different approaches in Table 4.1 is shown. One random pose is applied to the RGB-D image and the transformation matrix at the solution is the identity. The solution is marked as a green dot with coordinates $(0, 0, 0)$. It can be seen that a more direct trajectory is obtained by the proposed Point-to-hyperplane ICP. Two strategies were improved with the matching stage (Adaptive lambda and the Point-to-hyperplane ICP) by using a 4 dimensional *k*d-tree. It was observed that more direct trajectories and a lower value of the cost function is obtained. However, for purposes of frame-to-frame visual odometry they have demonstrated a similar performance.

Real environment

Well known *living room* ICL-NUIM RGB-D [29], *freiburg1* and *freiburg3* TUM [84] benchmark datasets were employed to perform visual odometry by using hybrid strategies. These datasets provide a sequence of RGB-D images that can be synchronized by associating timestamps in the color and depth frames. The sequences also provide a groundtruth trajectory obtained by an Optitrack system.

(a) Optitrack system (b) Tracked RGB-D sensor

Figure 4.7: Benchmark sequences in [84] were obtained by tracking a RGB-D sensor with an Optitrack system.

For this experiment, a frame-to-frame alignment was employed. The estimated poses were used to evaluate the ATE (Absolute Trajectory Error) and RPE (Relative Pose Error).

The relative pose error measures the drift of the trajectory over a fixed time interval δ as:

$$RPE_i = \left(\mathbf{Q}(\mathbf{x})_i^{-1}\mathbf{Q}(\mathbf{x})_{i+\delta}\right)^{-1}\left(\mathbf{T}(\mathbf{x})_i^{-1}\mathbf{T}(\mathbf{x})_{i+\delta}\right) \tag{4.16}$$

where $\mathbf{Q}(\mathbf{x}) \in SE(3)$ is the groundtruth pose and $\mathbf{T}(\mathbf{x}) \in SE(3)$ is the estimated pose.

On the other hand, the absolute trajectory error measures the global consistency of the estimated trajectory. It compare the absolute distances between the estimated and groundtruth poses as:

$$ATE_i = \mathbf{Q}(\mathbf{x})_i^{-1}\mathbf{S}(\mathbf{x})\mathbf{T}(\mathbf{x})_i \tag{4.17}$$

where $\mathbf{S}(\mathbf{x})$ is the rigid-body transformation that maps the estimated trajectory onto the groundtruth trajectory.

Various examples of the ATE evaluation for the Point-to-hyperplane strategy are shown in Figure 5.13, where it can be seen that the Point-to-hyperplane method can obtain close solutions w.r.t. the groundtruth without employing extra strategies for pose refinement as loop closure methods.

The numerical results of the ATE and RPE are shown in Table 5.4, where the methods are listed as follows:

1. hybrid + non-adaptive $\lambda_I = I_i/255$
2. hybrid + non-adaptive $\lambda_I = I_i/255$ (NN4D*[2])

[2]The legend NN4D means that the closest points were estimated by a *k*d-tree in the first iteration only

3. Point-to-hyperplane
4. Point-to-hyperplane (NN4D)
5. hybrid + adaptive $\lambda_G = MAD(\mathbf{e}_I)/MAD(\mathbf{e}_G)$ [86]
6. hybrid + adaptive $\boldsymbol{\lambda} = cov(\mathbf{e}_G, \mathbf{e}_I)$ [45]

From Table 4.2 and 4.3, it can be seen that the Point-to-hyperplane ICP methods improve on the other methods while obtaining less computational cost and less number of iterations. It was observed that when the frame-to-frame alignment is employed, the strategies 2 and 4 obtain about the same results as strategies 1 and 3, respectively. Therefore, the results for these strategies are shown together in Table 4.2 and 4.3, where it can be noted that the adaptive $\boldsymbol{\lambda}$ methods obtained less ATE and RPE for the *freiburg3* sequences (benchmarck structure vs texture), however, the computational cost is high w.r.t. Point-to-hyperplane strategies. This is due to that the method becomes purely geometric or photometric approach for these datasets. On the other hand, it can be noted that the Point-to-hyperplane method obtained more accurate results closed indoor scenarios in sequences as: *360, room, teddy, plant* w.r.t. groundtruth. An example of the performance of the Point-to-hyperplane method can be shown in Figure 4.9, where the 3D reconstruction of challenging 360 degree sequences is obtained by transforming the cloud of points by the estimated 6DOF pose parameter. The reconstructions can be refined by any post-processing algorithms. The post-processing refinement strategies will not be investigated in this chapter, but strategies that perform global convergence will be considered [90, 13, 83] in Chapter 5. The refinement of the Point-to-hyperplane method has been recently performed by estimating the global pose of a RGB-D frame w.r.t.the generated 3D model [42].

Visual SLAM

The Point-to-hyperplane method has been implemented successfully for real time applications by employing the ASUS Xtion sensor. The point-to-hyperplane method has demonstrated its robustness in a long corridor which contains few geometric and photometric features 4.10. The depth map obtained by the sensor contains holes in the depth map which are not valid values. This can generate a problem in the estimation of the normal for the Point-to-hyperplane. In previous works [40, 41], the corresponding intensities with non-valid depth values were considered as outliers. One solution can involve interpolating the surrounding valid depth values or assigning the value of the closest point as it is introduced in 3.3. By doing this, the method can achieve the main advantages of the Point-to-plane or the direct method if the color or the 3D point information is not available together at a i-th pixel coordinate.

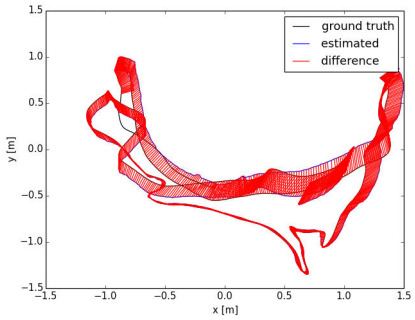
The implementation of the point-to-hyperplane ICP in C++ uses the PCL library for handling the 3D poinclouds, visualization and the estimation of the normals. In the experiments, cross product and PCA algorithms were tested in a 2×2 and 3×3 window. The fastest strategy was the cross product in a 2×2 window, but it is less accurate than performing PCA. In a 3×3 window, the PCA performed a faster estimation and more accurate results w.r.t. cross product. However, the timing obtained in both strategies was not efficient for performing real-time SLAM.

Table 4.2: Averages in Time (miliseconds), number of iterations, Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) for the synthetic dataset [29]. It can be seen that the Point-to-hyperplane method [40] improves hybrid methods that combine the direct approach and the geometric Point-to-plane approach.

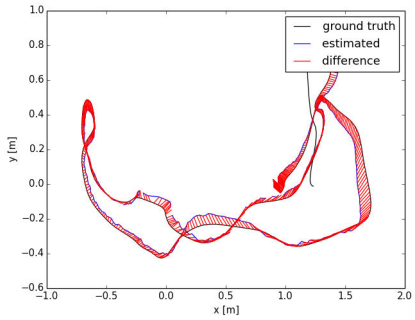
Sequence	Method	ATE (m)		RPE translational (m)		RPE rotational (deg)		AVERAGE	
		RMSE	MEAN	RMSE	MEAN	RMSE	MEAN	Time(sec)	#Iterations
lvr/traj0	1	**0.107**	**0.096**	0.003	0.002	0.083	0.067	0.203	15.68
	2	**0.107**	**0.096**	0.003	0.002	0.083	0.067	0.230	16.51
	3	0.128	0.114	**0.002**	**0.001**	**0.042**	**0.026**	**0.179**	**16.07**
	4	0.128	0.114	**0.002**	**0.001**	**0.042**	**0.026**	0.214	17.23
	5	0.230	0.210	0.006	0.004	0.132	0.105	0.541	41.47
	6	0.320	0.300	0.007	0.005	0.179	0.141	1.308	33.25
lvr/traj1	1	0.211	0.190	0.003	0.003	0.082	0.072	0.227	17.73
	2	0.211	0.190	0.003	0.003	0.082	0.072	0.240	17.95
	3	**0.041**	**0.032**	**0.001**	**0.001**	**0.021**	**0.017**	**0.148**	**14.87**
	4	**0.041**	**0.032**	**0.001**	**0.001**	**0.021**	**0.017**	0.181	15.99
	5	0.397	0.330	0.006	0.005	0.128	0.112	0.451	38.89
	6	0.341	0.291	0.007	0.006	0.155	0.136	1.228	34.30
lvr/traj2	1	0.152	0.146	0.003	0.003	0.085	0.074	0.189	16.13
	2	0.152	0.146	0.003	0.003	0.085	0.074	0.220	16.61
	3	**0.039**	**0.036**	**0.001**	**0.001**	**0.024**	**0.019**	**0.172**	**16.06**
	4	**0.039**	**0.036**	**0.001**	**0.001**	**0.024**	**0.019**	0.203	17.07
	5	0.323	0.297	0.007	0.005	0.139	0.118	0.519	41.93
	6	0.398	0.363	0.008	0.007	0.176	0.149	1.205	34.78
lvr/traj3	1	0.445	0.403	0.003	0.003	0.120	0.097	0.300	22.65
	2	0.445	0.403	0.003	0.003	0.119	0.097	0.317	22.95
	3	0.080	0.066	0.001	0.001	0.044	0.027	**0.185**	**15.97**
	4	**0.072**	**0.056**	**0.001**	**0.001**	**0.044**	**0.027**	0.212	16.62
	5	0.526	0.459	0.005	0.004	0.145	0.119	0.584	46.51
	6	0.484	0.436	0.007	0.006	0.179	0.150	1.689	42.07

Table 4.3: Averages in Time (miliseconds), number of iterations, Relative Pose Error (RPE) and Absolute Trajectory Error (ATE) for the dataset freiburg1 and freiburg3 [84].

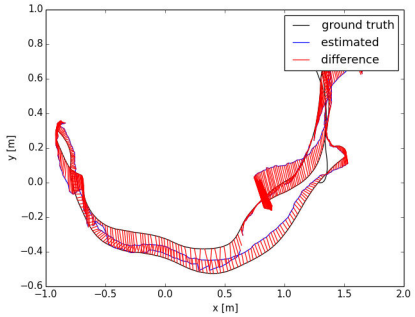
Sequence	Method	ATE (m)		RPE translational (m)		RPE rotational (deg)		AVERAGE	
		RMSE	MEAN	RMSE	MEAN	RMSE	MEAN	Time(sec)	#Iterations
fr1/xyz	1 & 2	0.068	0.064	0.041	0.035	2.485	2.068	0.227	18.41
	3 & 4	**0.045**	**0.038**	**0.021**	**0.019**	**1.106**	**0.998**	**0.301**	26.94
	5	0.092	0.087	0.040	0.035	2.407	2.034	0.364	28.97
	6	0.086	0.080	0.041	0.036	2.421	2.080	0.946	**24.51**
fr1/rpy	1 & 2	0.102	0.087	0.040	0.034	2.918	**2.566**	**0.273**	**21.16**
	3 & 4	**0.035**	**0.032**	**0.038**	**0.032**	**2.820**	2.652	0.445	36.79
	5	0.129	0.111	0.045	0.037	3.034	2.643	0.288	23.15
	6	0.131	0.114	0.046	0.038	2.947	2.594	0.997	25.09
fr1/360	1 & 2	0.353	0.332	0.111	0.090	3.917	3.550	0.295	22.95
	3 & 4	0.322	0.296	0.152	0.114	**3.159**	**2.859**	0.460	38.68
	5	**0.190**	**0.179**	**0.094**	**0.085**	4.113	3.583	**0.285**	**22.18**
	6	0.268	0.245	0.191	0.149	5.210	4.539	0.962	24.55
fr1/room	1 & 2	0.375	0.353	0.075	0.055	3.373	2.836	**0.255**	**19.75**
	3 & 4	**0.152**	**0.131**	**0.056**	**0.047**	**2.673**	**2.329**	0.375	33.36
	5	0.323	0.286	0.063	0.051	3.250	2.743	0.308	23.34
	6	0.363	0.305	0.068	0.055	3.434	2.902	0.896	23.41
fr1/desk	1 & 2	**0.064**	**0.060**	**0.043**	**0.036**	2.738	2.403	**0.236**	**19.78**
	3 & 4	0.071	0.067	0.044	**0.036**	**2.310**	**2.028**	0.408	34.98
	5	0.069	0.065	0.044	**0.036**	2.580	2.233	0.300	23.79
	6	0.065	0.062	0.045	0.037	2.667	2.285	0.877	24.19
fr1/desk2	1 & 2	**0.083**	**0.079**	**0.058**	**0.049**	3.816	3.133	**0.243**	**20.00**
	3 & 4	0.133	0.116	0.060	0.051	**3.026**	**2.641**	0.496	38.33
	5	0.560	0.233	0.624	0.154	24.448	7.055	0.328	25.02
	6	0.409	0.188	0.463	0.129	16.883	5.887	0.940	25.20
fr1/floor	1 & 2	**0.124**	**0.105**	**0.035**	**0.023**	**1.946**	**1.364**	**0.198**	**15.61**
	3 & 4	0.473	0.405	0.080	0.051	3.909	1.915	0.355	31.67
	5	0.273	0.224	0.085	0.037	2.846	1.754	0.281	21.64
	6	2.050	1.765	0.384	0.089	21.096	4.932	0.873	22.75
fr1/plant	1 & 2	0.066	0.056	0.035	0.030	1.740	1.568	**0.262**	**20.63**
	3 & 4	0.101	0.093	0.055	0.043	2.130	1.947	0.395	34.88
	5	0.067	0.055	**0.031**	**0.027**	**1.608**	**1.405**	0.329	25.48
	6	**0.065**	**0.054**	0.033	0.028	1.623	1.427	0.914	23.52
fr1/teddy	1 & 2	0.260	0.219	**0.061**	**0.046**	**1.996**	**1.656**	**0.282**	**20.97**
	3 & 4	**0.169**	**0.158**	0.070	0.056	2.287	1.954	0.424	36.86
	5	0.303	0.267	0.086	0.049	2.432	1.734	0.322	23.74
	6	0.339	0.299	0.100	0.055	2.681	1.821	0.926	24.12
fr3/s_t_far	1 & 2	0.136	0.133	0.028	0.025	0.936	0.856	0.201	16.71
	3 & 4	0.401	0.361	0.067	0.061	1.561	1.428	**0.172**	**15.52**
	5	**0.044**	**0.040**	0.024	0.021	0.707	0.639	0.274	22.32
	6	**0.044**	0.042	**0.021**	**0.018**	**0.649**	**0.590**	0.617	16.32
fr3/s_t_near	1 & 2	0.152	0.143	0.027	0.021	1.568	1.176	**0.161**	**13.59**
	3 & 4	0.185	0.157	0.045	0.036	1.862	1.525	0.240	21.32
	5	**0.056**	**0.052**	**0.018**	**0.016**	**0.965**	**0.848**	0.345	27.43
	6	**0.066**	0.063	0.019	0.016	0.997	0.856	0.813	21.25



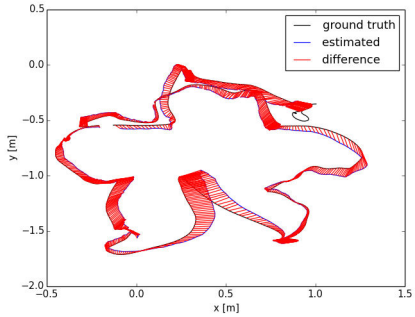
(a) fr1_room



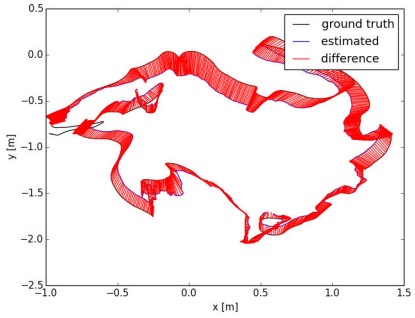
(b) fr1_desk



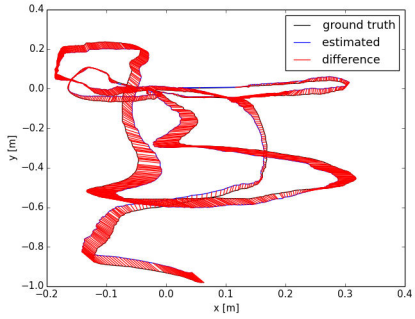
(c) fr1_desk2



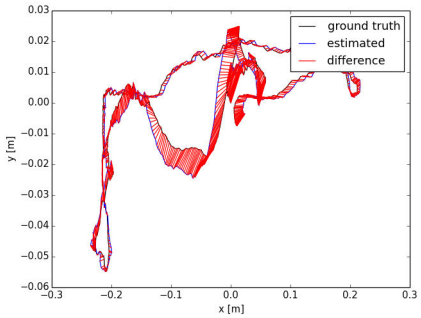
(d) fr1_plant



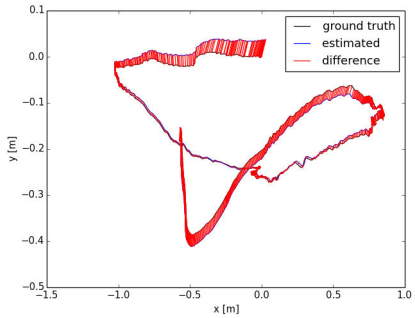
(e) fr1_teddy



(f) lvr_traj0

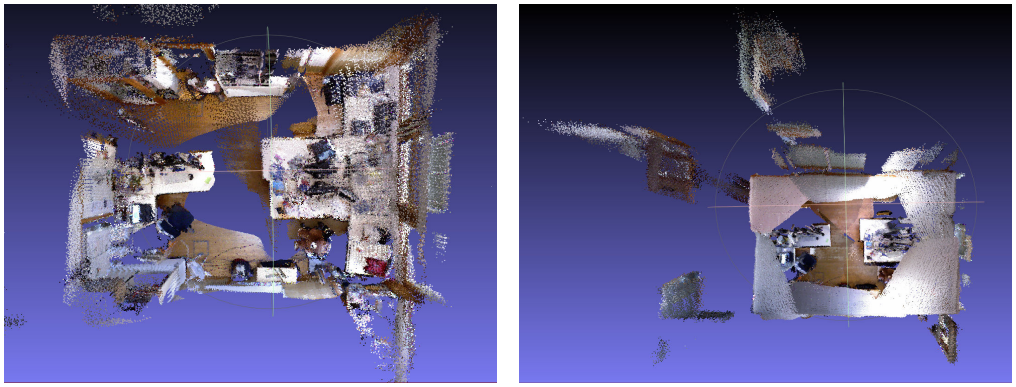


(g) lvr_traj1

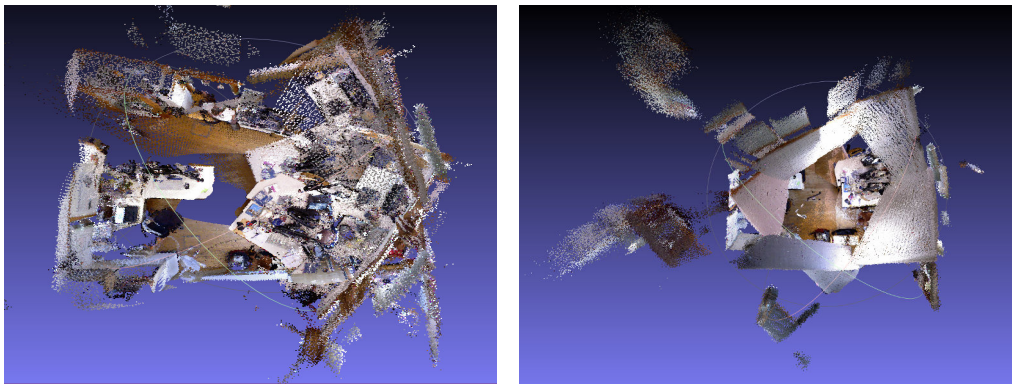


(h) lvr_traj2

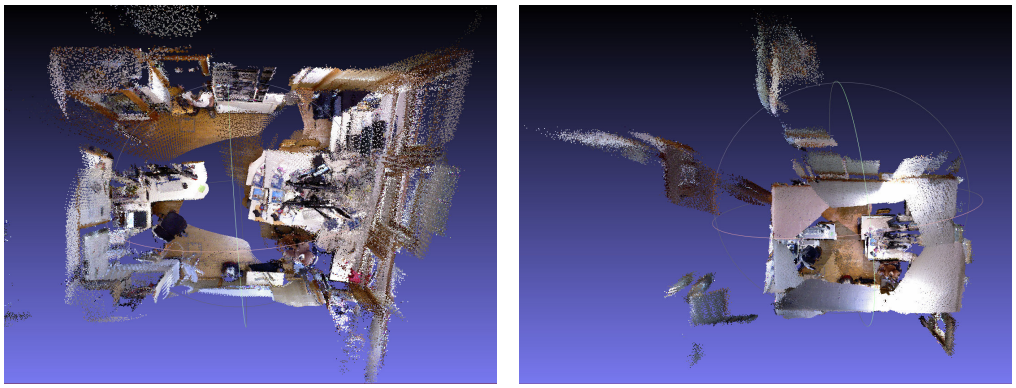
Figure 4.8: Examples of the Absolute Trajectory Error evaluation obtained by the Point-to-Hyperplane method. The benchmarck datasets [29] and [84] were used.



(a) fr1/room groundtruth (b) fr1/360 groundtruth



(c) Direct method + P2Pl (d) Direct method + P2Pl

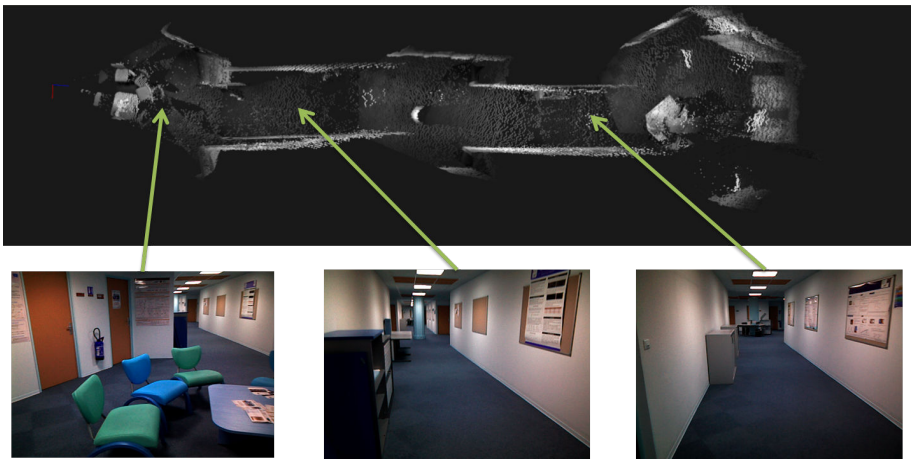


(e) **Point-to-hyperplane** (f) **Point-to-hyperplane**

Figure 4.9: 3D reconstruction of sequences freiburg1: room, 360, plant and teddy (shown at each row, respectively). In the first row the groundtruth obtained by an external motion capture system is shown, the second row illustrates the results obtained by a classic hybrid approach, where the uncertainty radius normalizes the data. Finally, the third row shows the result of the Point-to-hyperplane method. These difficult 360 degree sequences with motion blur clearly show that the proposed method can achieve more robust estimations.

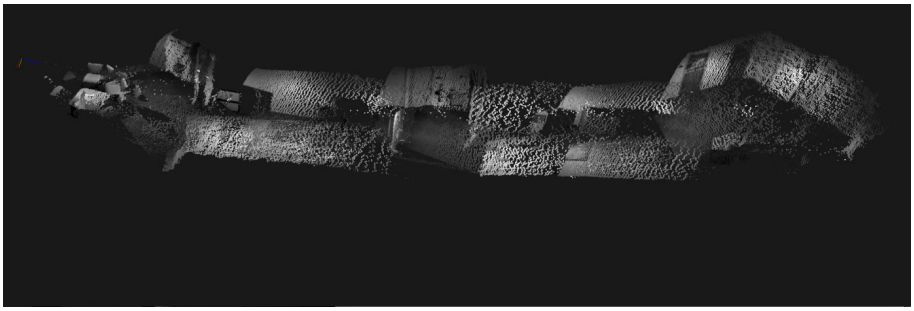
During the experiments, the estimation of the normals obtained an average about 1.2 seconds for a 3×3 window using a PCA strategy, but it obtains more accurate normals. For a real-time SLAM application, the cross product was performed in a 2×2 window by considering all four pixels in the window of the reference frame. If no depth or intensity data is available at the central pixel, it is considered as an outlier.

Contrary to the Point-to-hyperplane ICP, classic hybrid approaches are often somewhat ad-hoc because they do not necessarily consider color and depth together for computing closest points and they simply combine classic geometric ICP approaches with image-based approaches by minimizing both errors simultaneously. Based on this hybrid minimization, a contribution made during this research will be presented in the following section 4.3. The contribution proposes a matching stage for direct methods, which considers the pixel coordinates as an available criterion for both, closest points computation and minimization of the error, where no features are extracted. This provides an increased convergence for direct methods and more robust alignments.



(a) Top view

(b) Perspective view



(c) Side view

Figure 4.10: 3D reconstruction of a long corridor by the Point-to-hyperplane method

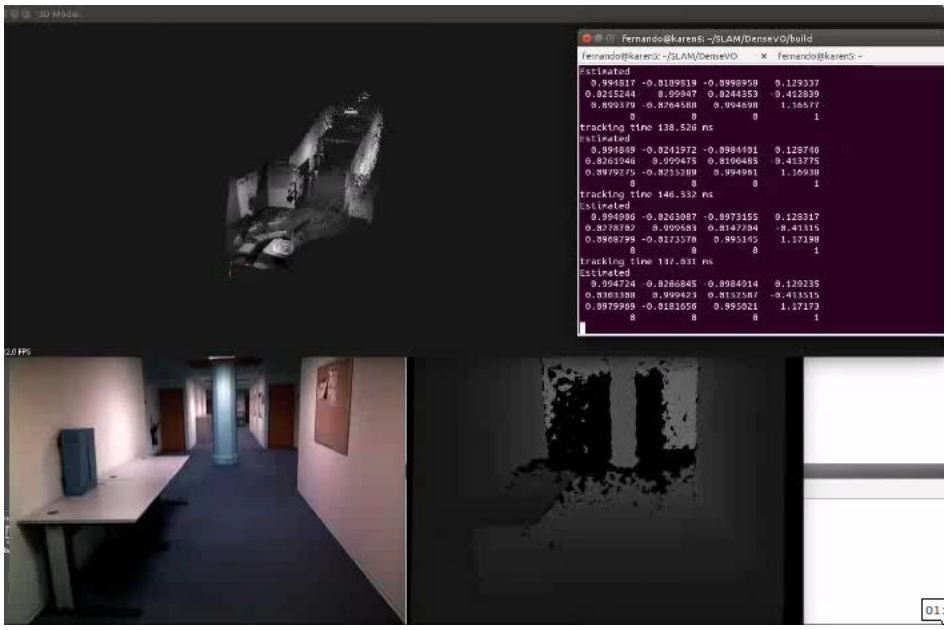
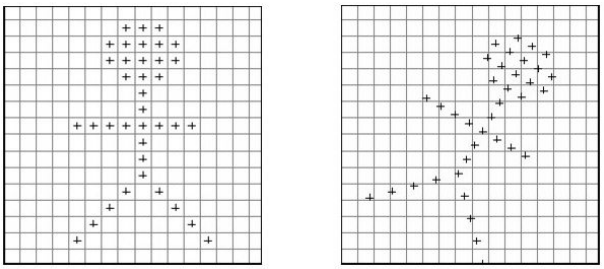


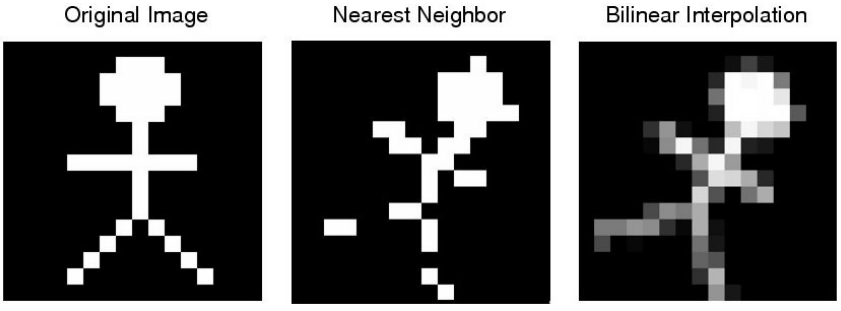
Figure 4.11: Dense VO interface. This C++ algorithm performs visual SLAM in both, real time and offline. In the bottom left and right, the color and depth images are displayed, respectively. The 3D poincloud is shown in grayscale and the transformation $\mathbf{T}(\mathbf{x})$ along with the tracking time is displayed in a terminal.

4.3 Direct matching for improving image-based approaches

The aforementioned feature-based approaches first extract relevant geometric information from the color images (lines, corners, etc.) before matching these features. This results into an increased speed while obtaining the alignment between color images, but the drawback can be the robustness of feature extraction and the accuracy of feature matching selected for registration either in manual or automatic way. The matching stage can increase the possibility to find the transformation by minimizing just over matched features. The algorithm should converge faster, but it can require extra computational cost. Furthermore, extracted features are assumed to remain invariant to the image transformations. On the other hand, direct approaches operate directly on intensity values and they use all available image information without prior extraction of features. The main drawback of direct methods is the assumption that the intensity values remain constant across different acquisitions. Therefore, direct methods are often employed when the images are closely acquired and a Lambertian surface [3] is assumed, which results in a limited convergence rate. Despite this limitation, direct methods have an easy implementation based on direct image warping, making them useful for real-time applications. In fact, feature-based approaches can be considered as a sub-part of direct approaches with an improved matching stage.



(a) Warped central pixels



(b) Examples of image interpolation

Figure 4.12: (a) Original Pixel Center Locations (Left) and Rotated Pixel Center Locations (Right). It can be noted that warped central pixels do not correspond to exact pixel coordinates in the current frame or they correspond to the same pixel coordinates. Therefore, interpolation strategies can be employed to assign a valid coordinate. (b) Examples of interpolation methods applied to assign a pixel coordinate in the warped image [20].

[3] A Lambertian surface assumes that apparent brightness does not change with a different angle of view.

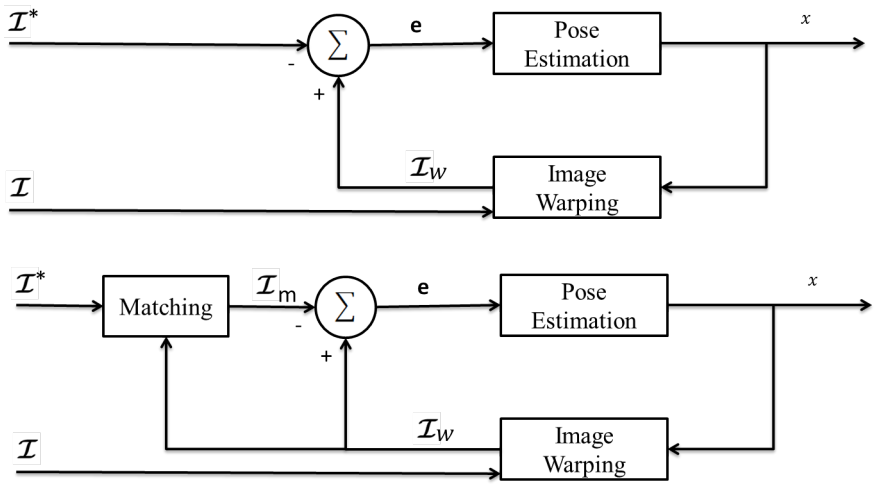


Figure 4.13: Robust direct minimization approaches. Interpolation (Top), proposed image-based matching (bottom).

When no feature extraction is performed (direct approach), the correspondences between two pixels is performed by interpolation methods, that can be achieved by mathematical functions (cited in [26]) such as nearest neighbors (as in the ICP), bilinear interpolation, bi-cubic, etc. or scattered data interpolation approaches [75]. Nearest-neighbor interpolation assigns the value of the nearest pixel to the pixel in the output visualization. This is the fastest interpolation method but the resulting image may contain jagged edges. On the other hand, linear interpolation computes the average of the nearest 2^n pixels. (4 for bilinear, 8 for bi-cubic, etc.). An example of both strategies is shown in Figure 4.12.

Similarly to several ICP strategies that accelerate the alignment, a matching stage based on the creation of a balanced *k*d-tree for the reference image, which is created before the iterative loop, is proposed (Figure 4.13) for direct approaches. The main difference with nearest neighbors interpolation $\mathcal{I}(\cdot)$ is that the *k*d-tree allows finding a nearest neighbor at an arbitrary distance from the warped point location.

Consider an image $\mathcal{I}$ of dimensions $m \times n$ associated with an intensity function $\mathbf{I}(\mathbf{p}_i)$ and a second image $\mathcal{I}^*$ with the same size with intensities $\mathbf{I}^*(\mathbf{p}_i^*)$. Assuming that the 3D pointcloud $\mathbf{P}^* \in \mathbb{R}^3$ and $\mathbf{P} \in \mathbb{R}^3$ for each image are measured and that the transformation $\mathbf{x}$ is unknown, a new image $\mathcal{I}^w$ is computed by the reverse warping function shown previously in (3.19).

The *k*d-tree is defined as: $\mathbf{k}^* = \mathbf{k}(\mathbf{M}^*)$, where each measurements vector of $\mathbf{M}^* = [\mathbf{M}_1^*, \mathbf{M}_2^*, \ldots, \mathbf{M}_{mn}^*]$ can be defined now as:

$$\mathbf{M}_i^* = \begin{bmatrix} \mathbf{p}_i^* \\ \mathbf{I}^*(\mathbf{p}_i^*) \end{bmatrix} \in \mathbb{R}^{3\times 1} \tag{4.18}$$

where $\mathbf{I}(\cdot)$ is the intensity value of the pixel at the $\mathbf{p}^*$ coordinates $[u^*\ v^*]^\top$.

A new set $\mathbf{M}^w = [\mathbf{M}_1^w, \mathbf{M}_2^w, \ldots, \mathbf{M}_{mn}^w]$ is defined here as the warped measurements vector, where:

$$\mathbf{M}_i^w = \begin{bmatrix} \mathbf{p}_i^w \\ \mathbf{I}\left(\omega(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \mathbf{P}_i^*)\right) \end{bmatrix} \in \mathbb{R}^3 \tag{4.19}$$

where $\mathbf{p}_i^w$ are the warped pixels as:

$$\overline{\mathbf{p}_i^w} = \frac{\mathbf{K}\Pi_3\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\overline{\mathbf{P}}_i^*}{\mathbf{e}_3^\top\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\mathbf{P}_i^*} \in \mathbb{R}^3 \tag{4.20}$$

Therefore, a new photometric error can be computed as:

$$\mathbf{e}_{I_i} = \mathbf{M}_i^m - \mathbf{M}_i^w = \begin{bmatrix} u_i^m \\ v_i^m \\ I_i^m \end{bmatrix} - \begin{bmatrix} u_i^w \\ v_i^w \\ I_i^w \end{bmatrix} \tag{4.21}$$

where $\mathbf{M}_i^m$ is a vector with the matching correspondences.

During the experimentation, a better performance for this strategy was found when the intensity matching is done in the first iteration only. The use of the FLANN library [63] in each iteration may obtain less number of iterations, but each iteration takes more time to estimate the closest points, increasing then the computational cost. This is demonstrated in the results section. An interesting result obtained during the experiments, was a metric error generated in pixel coordinates if the matching is performed at each iteration. Following this motivation, the matching and the minimization of the error function (4.21) were improved by considering pre-computed parameters in the reference frame. The result of this alternative approach is presented in the following section.

Similarly to depth-based approaches presented in (3.6) and (3.7), classic image warping can be performed in two ways:

1. By projecting the reference 3D point into the current image and transforming the current image into the reference image as: $\mathbf{e}_I = \mathcal{I}^* - \mathcal{I}(\omega(\mathbf{T}(\mathbf{x}), \mathbf{P}_i^*))$.

2. By projecting the current points into the reference frame and transforming the reference image into the current image as: $\mathbf{e}_I = \mathcal{I} - \mathcal{I}^*(\omega(\mathbf{T}(\mathbf{x}), \mathbf{P}_i))$.

In any case, the pose between the reference and current frame can be obtained. In case 1) the error minimization depend on transforming the reference pointcloud $\mathbf{P}_i^*$ for obtaining the warped pixels. However, this is computationally expensive when considering pixel coordinates and intensities together for matching, since closest points are computed in the reference frame at each iteration of the minimization process. In case 2) the current cloud is transformed and the reference pointcloud is fixed. However, it is not considered as an optimal measurement for real sequences when a RGB-D sensor is used for the acquisition step, due to the fact that it depends of the current depth map $\mathbf{Z}(\mathbf{p})$ acquired at each new frame. Furthermore, the current depth image is commonly not properly synchronized with the current color image. In other words, the difference between both cases is that during the estimation of the pose $\mathbf{T}(\mathbf{x})$, the photometric term and the geometric error in pixel coordinates are minimized in opposite directions.

Pre-computing parameters to accelerate the matching as the *k*d-tree by considering fixed reference measurements can speed up the alignment between frames. In order to find the correspondences using the information from the reference measurements instead of the current measurements, the projected warped points in (4.20) can be projected-back into the reference image through the inverse of the transformation matrix $(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}))^{-1}$, generating pixel coordinates that depend on the pose $\mathbf{x} \in \mathbb{R}^6$ (Figure 4.14).

The metric error generated between the warped and the matched pixel coordinates can be minimized as in (4.21). As a result, a new strategy can be extended for the rows that were not considered for the minimization but for searching the closest points only, such as:

$$\mathbf{e}_{UVI_i} = \boldsymbol{\lambda}\left(\mathbf{M}_i^m - \mathbf{M}_i^w\right) \in \mathbb{R}^{3\times 1} \tag{4.22}$$

where $\mathbf{e}_{H_i}$ can be seen as an error in pixels and intensities just as: $\mathbf{e}_{H_i} = [\mathbf{e}_{UV_i}\ \mathbf{e}_{I_i}]^\top$ and $\boldsymbol{\lambda} = diag(\lambda_U, \lambda_V, \lambda_I)$ is the uncertainty factor. The *k*d-tree $\mathbf{k}^*$ is also created only once with reference measurements $\mathbf{M}^*$, and $\mathbf{M}_i^w$ is defined now as follows:

$$\mathbf{M}_i^w = \begin{bmatrix} \left(\frac{\mathbf{K}\Pi_3\left(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\right)^{-1}\overline{\mathbf{P}}^*_{w_i}}{\mathbf{e}_3^\top\left(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x})\right)^{-1}\mathbf{P}^*_{w_i}}\right) \\ \mathbf{I}\left(\omega(\widehat{\mathbf{T}}\mathbf{T}(\mathbf{x}), \mathbf{P}_i^*)\right) \end{bmatrix} \tag{4.23}$$

where $\overline{\mathbf{P}}^*_{w_i} = \mathbf{T}(\mathbf{x})\overline{\mathbf{P}}^*_i$.

Therefore, considering the full vector of $\mathbf{M}^w$ and its corresponding match points, the expression (3.25) can be rewritten as follows:

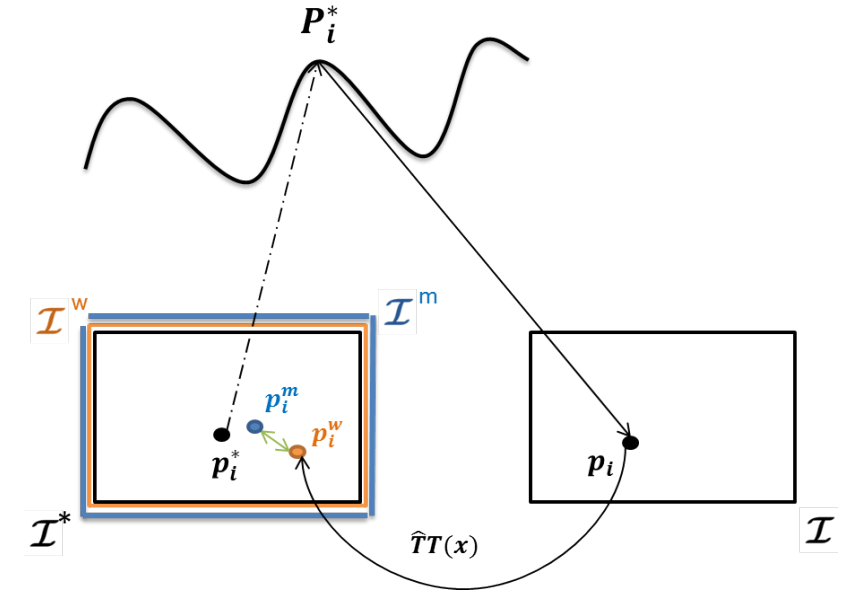
$$\mathbf{x} = -(\mathbf{J}^\top\mathbf{W}\mathbf{J}^\top)\mathbf{J}^\top\mathbf{W}\begin{bmatrix}\boldsymbol{\lambda}_{UV}\mathbf{e}_{UV} \\ \lambda_I\mathbf{e}_I\end{bmatrix} \tag{4.24}$$

where $\mathbf{J} = [\mathbf{J}_{UV}\ \mathbf{J}_{ESM}]^\top$ represents the stacked Jacobian of the geometric error (in pixel coordinates) and the photometric parameters and $\boldsymbol{\lambda}_{UV} = diag(\lambda_U, \lambda_V)$. The Jacobian $\mathbf{J}_{UV}$ is computed through the derivation of the geometric error function in pixel coordinates:

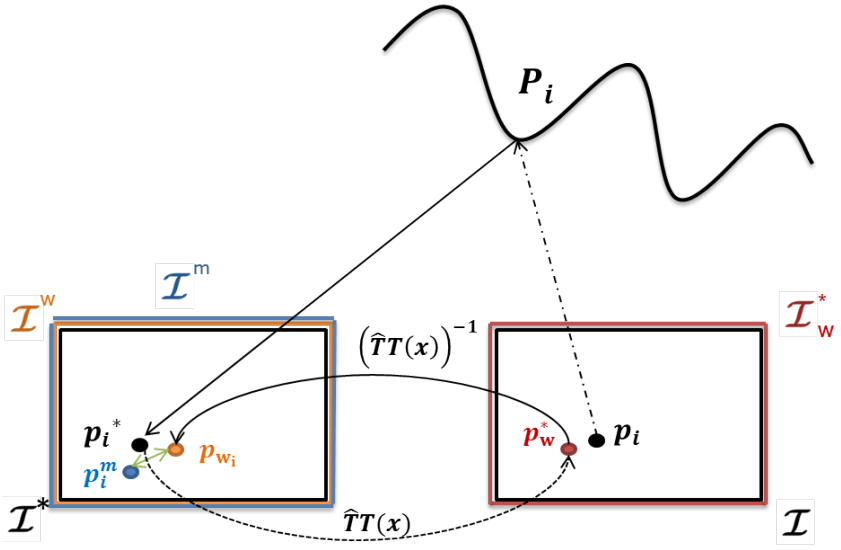
$$\mathbf{e}_{UV_i} = \Pi_2\boldsymbol{\lambda}_{UV}(\overline{\mathbf{p}_i^m} - \overline{\mathbf{p}_i^w}) \tag{4.25}$$

where Π_2 projects the 3×3 matrix onto 3×2 space.

During the experimentation, it was observed that during the first iteration only (transformation is equal to the identity) the warped pixels and the reference pixels are the same. Therefore, the pixels coordinates have no influence in the first iteration of the minimization process, but they are useful for estimating the closest points. Furthermore, after the first iteration, an error function in pixels coordinates is obtained as in 4.25 with an increased convergence domain and less error is obtained. However, the look-up of the closest points by using a *k*d-tree is computationally expensive.



(a) Direct method with matching



(b) Proposed hybrid direct matching

Figure 4.14: Proposed direct matching approach. (a) When pixel coordinates are considered along with intensities for estimating the closest points in classic direct approaches, more true nearest neighbours can be found. (b) It is possible to synthesize a new image $\mathcal{I}^w$ from $\mathcal{I}^*$, at the position of $\mathcal{I}$ if $[\mathbf{R}(\mathbf{x}), \mathbf{t}(\mathbf{x})]$ is known. The strategy proposed here to compute the geometric error in pixel coordinates is by projecting back the warped points $\mathbf{p}_w^*$ into the reference frame, generating an image with the matched intensities $\mathcal{I}^m$ and pixel coordinates $\mathbf{p}^m$ that depend on the pose parameter $\mathbf{x}$.

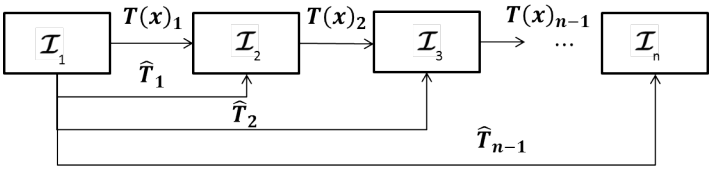


Figure 4.15: Frame-to-keyframe tracking diagram.

4.3.1 Results

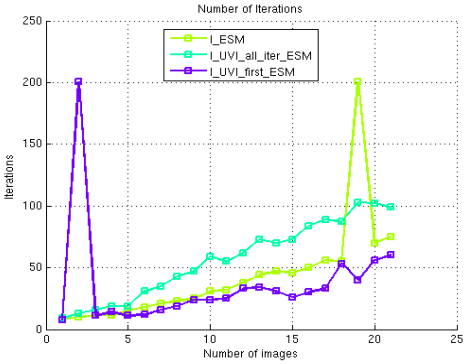
The performance of the proposed direct matching approach will now be shown. The experiments were performed on real environments, where each RGB-D frame was acquired at different poses. The pose is estimated by performing the direct method with ESM minimization (See apendix A.3) and compared for both, with and without the matching stage. In order to evaluate the performance of the direct matching method, the pose is obtained by warping different images w.r.t a common reference image in a sequence. The sequences contain motion for translation and rotation along and around all three axes of the camera frame, where each new image in the sequence generates a bigger displacement along an axis or a bigger spin about an axis (sequences $fr1/xyz$ and $fr1/rpy$ in [84]. The results obtained are presented in Table 4.4 and an example is given in Figure 4.16 where 22 frames with translational motion along one axis is registered [4].

From Figure 4.16, it can be noted that the convergence rate can be extended if the pixel coordinates are considered to compute the closest points and less number of iterations can be obtained if the matching is performed in the first iteration only. In this thesis, better results were obtained by using pixel coordinates to find the closest points in all iterations, and better accuracy was observed when the error generated between pixels coordinates is minimized in an hybrid manner. However, it requires more memory consumption.

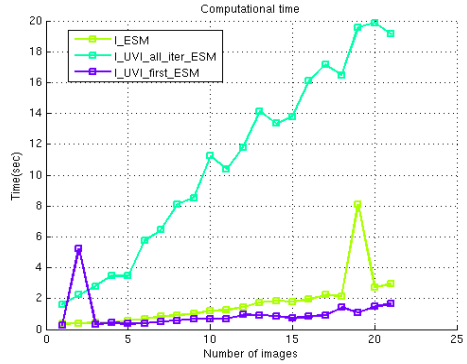
For the experimental part of this chapter, the scaling between pixel coordinates and intensities was estimated by normalizing the measurements as: $\boldsymbol{\lambda} = diag(\lambda_U = [\mathbf{p}_i]_1/width, \lambda_V = [\mathbf{p}_i]_2/height, \lambda_I = \mathbf{I}_i/255)$. The images were sampled by using a multi-resolution pyramid (160×120 at the top) in order to accelerate the alignments. M-estimators were employed to reject outliers and closest points are estimated by employing a balanced kd-tree generated by the FLANN library [63].

When frame-to-frame visual odometry was performed initially, a similar performance was observed for both strategies (with or without matching in the first iteration). Therefore, the improvement of a classic direct method was observed when larger motions between frames are presented. The selection of the RGB-D frames was done by visually observing the motion of the camera. It can be mentioned here that all the frames employed in Table 4.4 obtained convergence for all direct methods introduced above, but a more extended convergence rate was observed when the matching stage is added for direct methods.

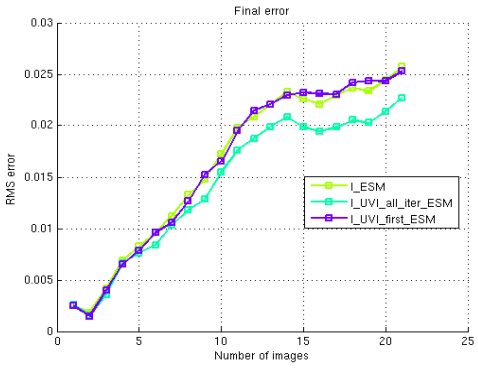
[4]Note that for iterations 2 and 19 in Figure 4.16, the pikes are obtained due to the fact that the convergence could not obtain a value below the established error threshold. Therefore, the algorithm stops when the maximum number of iterations is reached. However, it can be seen in (c) that the error is close to other methods



(a) Number of iterations



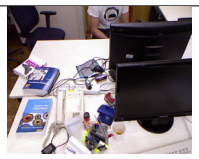
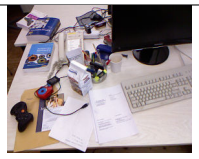
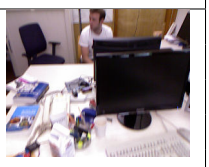
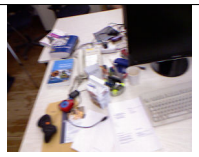
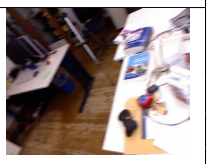
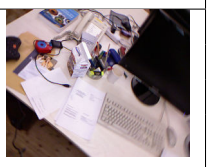
(b) Computational time



(c) RMS error

Figure 4.16: Performance of aligning synthetic RGB-D frames, where each new frame generates a bigger error w.r.t. the reference frame. It can be noted that when the closest points are computed in all the iterations, less iterations are performed, however the computational cost is high. Therefore, for purposes of this thesis, the closest points were computed in the first iteration only by considering a kd-tree (FLANN library was employed for this purpose [63]). The methods are identified as follows: 1) I_ESM = Direct method, 2) I_UVI_all_iter_ESM = Direct method + matching in all iterations (by considering also pixel coordinates for minimizing the error function) and 3) I_UVI_first_ESM = Direct method + matching in the first iteration only.

Table 4.4: Averages obtained by performing RGB-D registration with a direct method (DM) and the direct matching proposed. The norm of the final pose $||\mathbf{x}||$, angular velocity $||\boldsymbol{\omega}||$ and linear velocity $||\boldsymbol{v}||$ are compared along with number of iterations and computational time. It can be seen that the matching can improve the registration and it can obtain faster convergence if it is performed in the first iteration only. Each new frame in the interval generates a bigger error w.r.t. the initial frame.

Method	**Direct method** (DM)	**DM + matching (all iterations)**	**DM + matching (first iteration)**	Initial frame	Last frame
Translation along x axis (sequence *fr1/xyz*)				285	305
$\|\|\mathbf{x}\|\|$	0.723	**0.250**	0.494		
$\|\|\boldsymbol{\omega}\|\|$	0.5430	**0.170**	0.388		
$\|\|\boldsymbol{v}\|\|$	0.471	**0.181**	0.299		
# Iterations	206.14	**201.52**	220.52		
Time (sec)	2.47	10.05	**2.19**		
Translation along y axis (sequence *fr1/xyz*)				490	510
$\|\|\mathbf{x}\|\|$	0.139	0.138	**0.134**		
$\|\|\boldsymbol{\omega}\|\|$	**0.036**	0.076	0.053		
$\|\|\boldsymbol{v}\|\|$	0.133	**0.106**	0.117		
# Iterations	**111.91**	187.29	159.72		
Time (sec)	**1.356**	8.979	1.826		
Translation along z axis (sequence *fr1/xyz*)				1	20
$\|\|\mathbf{x}\|\|$	0.198	**0.155**	0.171		
$\|\|\boldsymbol{\omega}\|\|$	0.102	**0.080**	0.084		
$\|\|\boldsymbol{v}\|\|$	0.169	**0.131**	0.147		
# Iterations	134.8	177.3	**124.5**		
Time (sec)	1.662	8.966	**1.529**		
Rotation about x axis (sequence *fr1/rpy*)				130	150
$\|\|\mathbf{x}\|\|$	0.502	**0.223**	0.250		
$\|\|\boldsymbol{\omega}\|\|$	0.366	0.169	**0.162**		
$\|\|\boldsymbol{v}\|\|$	0.334	**0.141**	0.171		
# Iterations	186.76	**191.95**	194.57		
Time (sec)	2.266	9.079	**2.188**		
Rotation about y axis (sequence *fr1/rpy*)				25	45
$\|\|\mathbf{x}\|\|$	1.590	**0.463**	0.507		
$\|\|\boldsymbol{\omega}\|\|$	1.128	**0.344**	0.362		
$\|\|\boldsymbol{v}\|\|$	1.089	**0.294**	0.317		
# Iterations	**206.71**	259.43	218.76		
Time (sec)	2.356	11.915	**2.317**		
Rotation about z axis (sequence *fr1/rpy*)				370	390
$\|\|\mathbf{x}\|\|$	0.714	**0.240**	0.491		
$\|\|\boldsymbol{\omega}\|\|$	0.549	**0.188**	0.476		
$\|\|\boldsymbol{v}\|\|$	0.407	**0.128**	0.093		
# Iterations	**173.42**	210.43	181.29		
Time (sec)	2.145	10.204	**2.076**		

CHAPTER 5

GLOBAL POINT-TO-HYPERPLANE ICP

5.1 Introduction

Global registration has been proposed for guaranteeing convergence between datasets regardless of its initial pose. Existing global approaches have used two main strategies: The first category are those approaches based on the IRLS framework [24, 27, 35, 54, 59, 92] with an extended convergence domain that find the optimal correspondences between datasets when geometric or color information is known. These strategies however, demand a pre-selection of good correspondences and they can require extra computational time. The second category are those strategies that explore the transformation space for finding a solution to the non-convex error function. In this category, the Branch-and-Bound (BnB) algorithm is widely used to solve non-convex problems by searching the complete space of solutions for the best solution. In the literature, variants of the BnB method have been employed for 3D registration and applied to globally localizing pointclouds which can contain a small overlapping area. These methods mainly differ in:

- How the transformation space is parametrized [1] and divided (branching).
- How the upper and lower bounds are evaluated.
- How the subspaces are selected for exploration.

Three prominent strategies that employ BnB methods for estimating the 6 DOF pose parameters without extracting features can be found in [90, 13, 83] and they are compared in Table 5.1. All methods explore a 6 dimensional space (3 for rotation and 3 for translation) and they employ a collaboration between nested BnB (The translational space is explored after exploring the rotational space) and a local approach. While the BnB can obtain rough alignments, the local approach

[1] See Appendix A.4 for details

can refine the alignment. A strategy followed by all three methods is to initialize the BnB method with the result obtained by the local approach. If the estimated pose obtained by the local approach gets trapped in a local minima or if it cannot converge, the BnB is initialized with that estimated pose. Once the BnB method obtains a rough alignment between the 3D pointclouds, the pose is refined by the local method until convergence. Particularly, this is the strategy followed by the Go-ICP pipeline. Therefore, it will be explained in detail here since a similar pipeline was followed in the contribution of this thesis by considering higher dimensions. However, other strategies can be equally considered.

Table 5.1: 6D Global approaches. These methods explore the rotational and translational space together to estimate the unknown pose parameter with the BnB strategy and they refine the pose with a local method.

Method	Go-ICP [90]	GOGMA [13]	BB+ICP [83]
Rotation representation	Angle-axis	Angle-axis	Quaternions
Translation representation	π-cube	π-cube	Gaussian Mixture Models (GMM)
Branching strategy	Octrees	Octrees	4D Geodesic grid
Bounding strategy	Uncertainty radius	Uncertainty radius	Dirichet Process von-Mises-Fisher Mixture (DP-vMF-M)
Searching strategy	Best first search	Depth first search	Best first search
Local approach	Point-to-Point ICP	Gaussian Mixture Alignment (GMA)	Point-to-Point ICP

5.1.1 Branching of transformation space

In [13, 30, 90] an ease manipulation of the rotational space for the BnB method is proposed. The rotational $SO(3)$ space is represented in a solid sphere of radius π [30]. The π-sphere formed by the angle-axis representation, is circumscribed in a 3D cube $C_R = [-\pi, \pi]^3$ as the rotation domain. Similarly, the translation space is represented as a constrained cube $C_t = [-l, l]^3$, where it is assumed that the optimal translation will be found. The rotational and translational space is sub-divided by employing an octree data structure as is shown in Figure 5.1, where each generated sub-cube $C_{R_j} \subset C_R$ and $C_{t_j} \subset C_t$ (Figure 5.1(d) and 5.1(b), respectively) represents a search state with possible solutions to (2.8). The generated sub-cubes that can contain a feasible solution are selected to be branched and bounded.

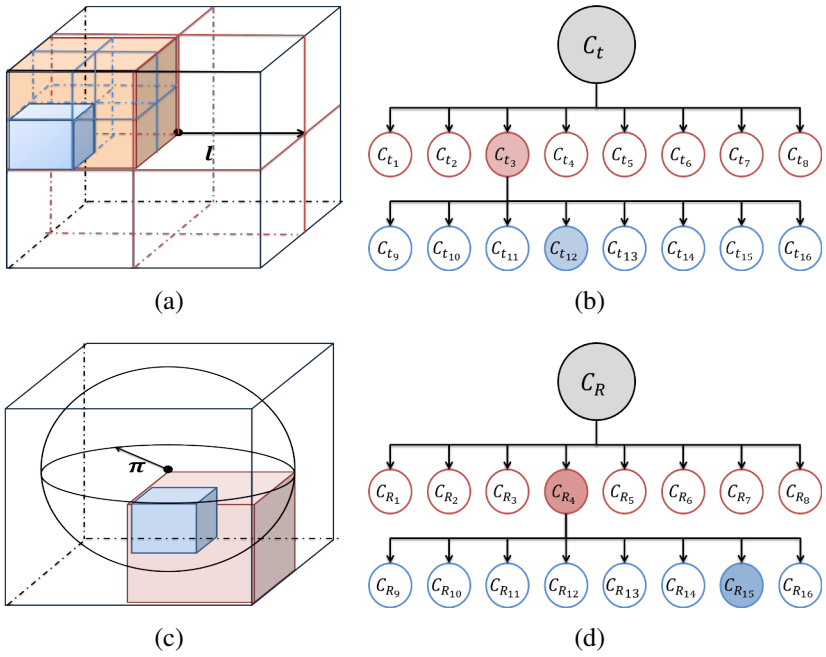
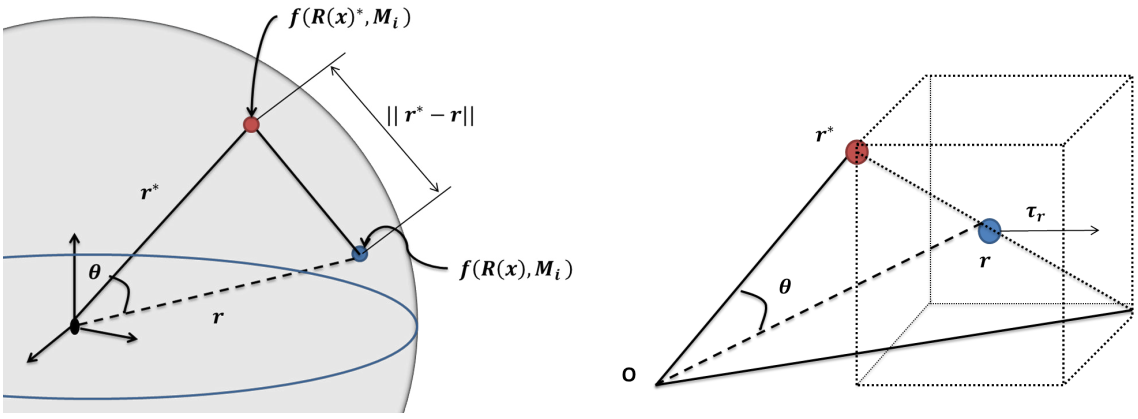


Figure 5.1: Parametrization of the $SE(3)$ space and subcubes. An example of an octree branching is presented in (b) and (d). (c) The rotation space can be represented in a π-sphere within a cube of 2π length as $C_R = [-\pi, \pi]^3$ and (a) a 3D cube half side-length l for translation as $C_t = [-l, l]^3$. The subspace candidates that may contain the global solution are colored.

The angle-axis representation, however, does not lead to a uniform subdivision of the rotation space and it covers part of the rotation space twice (the method has been better constrained in [13]). Despite this, this representation will be employed for purposes of this thesis since the aim is to extend the method to higher dimensions based on this Branch-and-Bound method. An optimized branching method can be found in [83], where a 4D tetrahedra is employed instead in a quaternion representation. This representation optimizes the branching stage to better approximate the distance on the rotation manifold and it can be equally considered for the Branch-and-Bound method.

The branching stage of the formed hypercubes can be done directly by considering the 6D space (3 rotational and 3 translational degrees). However, this strategy is inefficient because a large number of 6D sub-cubes ($2^6 = 64$ in total) can be generated, which can lead to a high memory consumption when the operations (bounding) are performed for all the generated sub-cubes. Two main strategies to optimize the Branch-and-Bound method can be found in the references given above, where the rotational and translational spaces have been branched in an ad-hoc manner and computed in parallel [69, 83] or by calling an inner Branch-and-Bound [13, 90] after performing an outer Branch-and-Bound. The rotational space is considered first for the outer Branch-and-Bound, since translation operations are computationally cheaper and it avoids redundant operations.



(a) (b)

Figure 5.2: Distance computation between two rotations $\mathbf{R}(\mathbf{x})^*$ *and* $\mathbf{R}(\mathbf{x})$

5.1.2 Upper and lower bounds

The performance of the Branch-and-Bound method mainly depends on the quality of the bounds. In order to solve the registration problem, bounds for the L_2-norm error function (2.8) have been proposed within a domain of the generated subspaces C_R and C_t. An important inequality used to determine how a 3D point is perturbed by an arbitrary rotation is the distance between two rotations defined by $\mathbf{R}(\mathbf{x})$ and $\mathbf{R}(\mathbf{x}^*)$ and the angle θ between them. The distance is the angle θ lying in the range $0 \leq \theta \leq \pi$ and lemmas established in [30, 90] can be summarized in the lemma shown below:

Lemma. *For any vector* $\mathbf{M}_i$ *and two rotations* $\mathbf{R}(\mathbf{x})$ *and* $\mathbf{R}(\mathbf{x}^*)$:

$$\angle\left(f(\mathbf{R}(\mathbf{x}), \mathbf{M}_i), f(\mathbf{R}(\mathbf{x}^*), \mathbf{M}_i)\right) \leq \angle\left(\mathbf{R}(\mathbf{x}), \mathbf{R}(\mathbf{x}^*)\right) \leq ||\mathbf{r} - \mathbf{r}^*|| \tag{5.1}$$

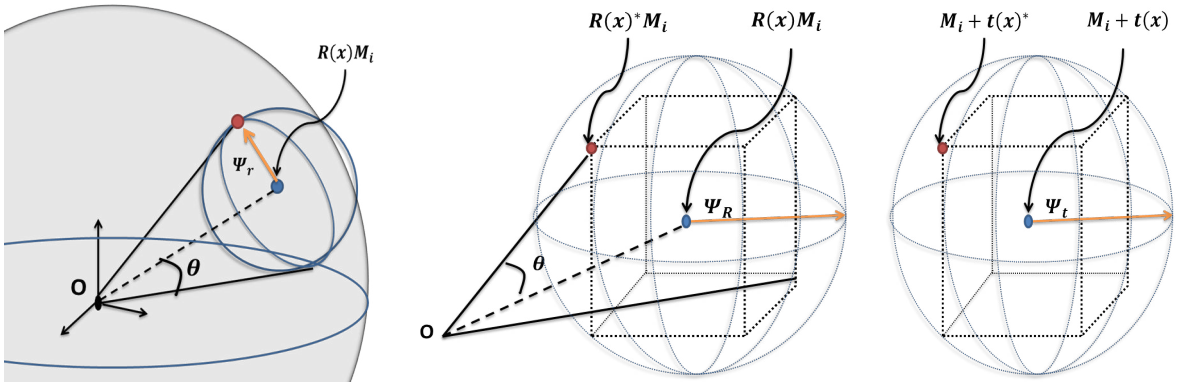
where $\angle\left(\mathbf{R}(\mathbf{x}), \mathbf{R}(\mathbf{x}^*)\right) = \theta$ and $\mathbf{r}$ and $\mathbf{r}^*$ as their angle-axis representation (A.7). The lemma above states that the angle distance is less than the Euclidean distance in the angle-axis representation. From this, the maximum aperture angle θ between two rotations can be found by considering the rotation cube C_R of half-side length τ_R centered at $\mathbf{r}$ as: $max(\theta) \leq \sqrt{3}\tau_r$, which is the half-diagonal of the rotation cube (Figure 5.2(b)).

In [90, 13] a strategy based on the estimation of the maximum aperture of θ lead to introduce a new derivation of bounds by using a local spherical space around a centered point at $\mathbf{r}$. The upper and lower bounds for the 3D case are provided in the following bounds of per-point residuals theorems:

Theorem. *For a 3D motion domain* $C_R \times C_t$ *centered at* $\mathbf{T}(\mathbf{x})$*, the upper bound* $\overline{\mathbf{e}_i}$ *and lower bound* $\underline{\mathbf{e}_i}$ *of the optimal registration error* $\mathbf{e}_i(\mathbf{T}(\mathbf{x}))$ *at* $\mathbf{M}_i$ *can be chosen as:*

$$\overline{e_i} \doteq \mathbf{e}_i\left(\mathbf{T}(\mathbf{x})\right) \tag{5.2}$$

$$\underline{e_i} = max\left(\mathbf{e}_i\left(\mathbf{T}(\mathbf{x})\right) - (\psi_{R_i} + \psi_t), 0\right) \tag{5.3}$$



(a) Rotational searching space (b) Rotation uncertainty radius (c) Translation uncertainty radius

Figure 5.3: Uncertainty region for rotation and translation. The uncertainty radius is computed by enclosing the generated subspaces in a sphere.

where ψ_{R_i} and ψ_t are the uncertainty radius of the local spheres that enclose the generated sub-cubes. Their values can be found by considering the maximum distance from r to r^*, provided in the following theorem:

Theorem. *Consider a 3D point* $\mathbf{M}_i$*, a rotation cube* C_R *of half-side length* τ_r *centered at* $\mathbf{r}$*. The maximum distance from* $\mathbf{R}(\mathbf{x})\mathbf{M}_i$ *to* $\mathbf{R}(\mathbf{x}^*)\mathbf{M}_i$ *is* $\forall \mathbf{R}(\mathbf{x}) \in C_R$*:*

$$||\mathbf{R}(\mathbf{x})\mathbf{M}_i - \mathbf{R}(\mathbf{x})^*\mathbf{M}_i|| \leq 2\sin\left(\min\left(\sqrt{3}\tau_r/2, \pi/2\right)\right)||\mathbf{M}_i|| \doteq \psi_R \tag{5.4}$$

Similarly, consider a translation cube C_t *with half-side length* τ_t *centered at* $\mathbf{t}(\mathbf{x})$*. Therefore* $\forall \mathbf{t}(\mathbf{x}) \in C_t$*:*

$$||(\mathbf{M}_i + \mathbf{t}(\mathbf{x})) - (\mathbf{M}_i + \mathbf{t}(\mathbf{x})^*)|| \leq \sqrt{3}\tau_t \doteq \psi_t \tag{5.5}$$

The upper and lower bounds for 3D registration introduced in (5.4) and (5.5) for all $\mathbf{M}_i$ 3D points, the L_2-error can be summarized in the following corollary.

Corollary. *For a 3D motion domain* C_R *and* C_t *centered at* $\mathbf{T}(\mathbf{x})$ *with uncertainty radius* ψ_R *and* ψ_t*, the upper and lower bounds to solve the sum of squares error can be chosen as:*

$$\overline{\mathbf{E}(\mathbf{T}(\mathbf{x}))} = \sum_{i=1}^{N} (\mathbf{e}_i)^2 \tag{5.6}$$

$$\underline{\mathbf{E}(\mathbf{T}(\mathbf{x}))} = \sum_{i=1}^{N} \max\left(\mathbf{e}_i - (\psi_{R_i} + \psi_t), 0\right)^2 \tag{5.7}$$

The geometric explanation of the lower bound can be seen as a residual error after perturbing a 3D data point $\mathbf{M}_i$ by a 3D rigid transformation $\mathbf{T}(\mathbf{x}) \in C_R \times C_t$. Any transformed 3D point $f(\mathbf{T}(\mathbf{x}^*), \mathbf{M}_i)$ lies in the uncertainty sphere centered at $f(\mathbf{T}(\mathbf{x}), \mathbf{M}_i)$ with radius $\psi = \psi_R + \psi_t$. A matching point in the other dataset $\mathbf{M}_i^m$ will be always closer to the surface of the sphere due to the fact that the closest distance is always found between a matching point and the center of the

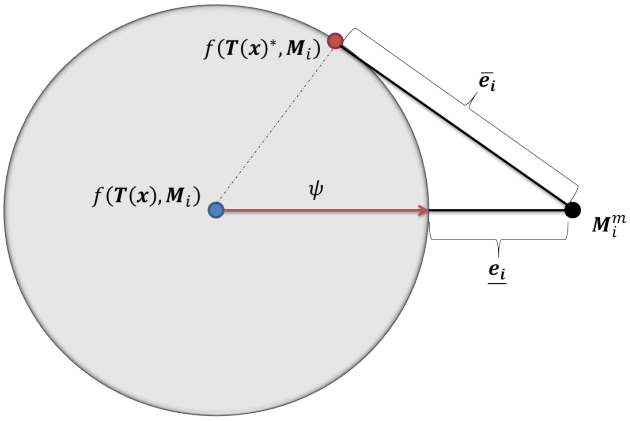


Figure 5.4: Geometric lower bound by considering the uncertainty radius strategy. The closest distance is always found between the matched point $\mathbf{M}^m$ and the surface with no matter if the transformed point lies inside the sphere.

uncertainty sphere. Therefore, any transformed data point $f(\mathbf{T}(\mathbf{x}^*), \mathbf{M}_i)$ can lie inside the sphere and its closest distance to a matching point will not be less than the lower bound [90] (Figure 5.4).

The proposed global approach in this thesis follows the pipeline shown in Section 5.1.2, but the point-to-hyperplane ICP method is considered to refine the pose estimates obtained by the branch-and-bound strategy. In the following, the branching, bounding and searching strategies employed here will be introduced by considering a higher dimensional space.

Branching the SE(3) space: Aforementioned in Section 2.5.1, the subdivision method can be performed before or after bounding the error function (the strategies are so-called eager and lazy Branch-and-Bound, respectively). In this thesis, a lazy-Branch-and-Bound pipeline with a nested 6D branching strategy is employed. A nested Branch-and-Bound strategy performs an inner translation Branch-and-Bound after evaluating live subspaces for the rotational space. This is less memory consuming than estimating a direct 6D space Branch-and-Bound.

Similarly to [90] and [13], an octree structure is employed in this thesis to subdivide the rotation and translation spaces. Rotation space is parametrized using the axis-angle representation as a vector $\mathbf{r} = \alpha\hat{r} \in \mathbb{R}^3$ where $\hat{r}$ is the axis and α is the angle. Therefore, the rotational $SO(3)$ space can be represented in a solid n-sphere of radius π [30]. The translation space can be represented as a constrained n-cube $C_t = [-l, l]^3$, within which it is assumed that the optimal translation will be found. Similarly, the π-sphere in n dimensions can be circumscribed in a n-cube $C_r = [-\pi, \pi]^3$. Each generated sub-cube $C_{R_j} \subset C_R$ and $C_{t_j} \subset C_t$ represents a search state with possible solutions to (2.8) (See Fig. 5.1).

Bounding the error function: Similar to Go-ICP [90], the lower bound is derived by considering a mathematical concept called *uncertainty radius* $\psi = \psi_{R_i} + \psi_t$ in terms of the angle metric [30, 90] as is presented in Section 5.1.2 but in higher dimensions. It examines the uncertainty region of a point $\mathbf{M}_i$ perturbed by an arbitrary rotation $\mathbf{R}_j(\mathbf{x}) \in C_R$ or a translation $\mathbf{t}_j(\mathbf{x}) \in C_t$ corresponding to the center of the cubes with half-side length τ_r and τ_t, respectively. These uncertainty regions C_R and C_t are enclosed in a n-dimensional sphere centered at $f(\mathbf{T}_j(\mathbf{x}), \mathbf{M}_i)$ and estimated separately for rotation and translation as is shown in Fig. 5.3(b) and 5.3(c), where

$\mathbf{T}(\mathbf{x}^*) = (\mathbf{R}(\mathbf{x}^*), \mathbf{t}(\mathbf{x}^*))$ runs over all rotations and translations, and represented by points in the cubes. Similarly to the 3D case, the maximum length of the uncertainty radius is obtained by considering the maximum distance between two arbitrary transformations (subspaces) $(\mathbf{T}(\mathbf{x}^*)_i, \mathbf{T}(\mathbf{x})_i)$ and it can be extended to higher dimensions with the following lemma, which summarizes the uncertainty radius in n-dimensions:

$$||f\left(\mathbf{t}_j(\mathbf{x}^*), \mathbf{M}_i\right) - f\left(\mathbf{t}_j(\mathbf{x}), \mathbf{M}_i\right)|| \leqslant \sqrt{n}\tau_t \doteq \psi_t \tag{5.8}$$

$$||f(\mathbf{R}_j(\mathbf{x}^*), \mathbf{M}_i) - f(\mathbf{R}_j(\mathbf{x}), \mathbf{M}_i)|| \leqslant min(\sqrt{n}\tau_r, \pi) \doteq \psi_{\mathbf{R}_i} \tag{5.9}$$

An important point from (5.9), is that the angle distance is less than the Euclidean distance between two arbitrary transformations in the angle-axis representation. Therefore, the maximum distance between two rotations cannot be bigger than the maximum angle $\theta \leq \sqrt{n}\tau$ [30], no matter if the rotations lie inside the sphere. By considering the 4-vector, the uncertainty radius of a 4-dimensional space can be defined here as:

$$\psi_t = 2\tau_t \tag{5.10}$$

$$\psi_{\mathbf{R}_i} = 2\sin\left(\tau_r\right)||\mathbf{M}_i|| \tag{5.11}$$

which are employed in (5.3) to compute the 4D lower bounds. The minimization of the upper and lower bounds in 4 dimensions can be summarized as:

$$\overline{\mathbf{E}(\mathbf{T}(\mathbf{x}))} = \sum_{i=1}^{N} max\left(\mathbf{e}_{H_i}, 0\right)^2 \tag{5.12}$$

$$\underline{\mathbf{E}(\mathbf{T}(\mathbf{x}))} = \sum_{i=1}^{N} max\left(\mathbf{e}_{H_i} - (\psi_{R_i} + \psi_t), 0\right)^2 \tag{5.13}$$

The geometric analysis of extending the bounding in higher dimensions will be shown in Section 5.1.3.

Therefore, the error function shown in (2.8) can be replaced with the following error function:

$$\mathbf{e}_{H_i} = \mathbf{N}_i^{*\top}\left(\mathbf{M}_i^m - \mathbf{M}_i^w\right) \in \mathbb{R}^n \tag{5.14}$$

where $\mathbf{M}_i^m$ denotes corresponding matches found between the reference and the transformed current measurements. $\mathbf{M}_i^w$ is the measurement vector transformed by the warping function $w(\cdot)$, which projects a reference measurement vector onto the current reference frame. For the experiments in this paper, each 3D point will be associated with a unique intensity value I_i as: $\mathbf{M_i} = [x_i \quad y_i \quad z_i \quad I_i]^\top$. $\mathbf{N}_i^* \in \mathbb{R}^4$ are the 4D normals of the reference measurements.

Searching a global solution: In this thesis, a best-first search is employed. The best-first strategy always selects the subspaces with the lowest bound, where no extra bound calculations take place after getting the optimal solution. The Branch-and-Bound searching stops when the difference between the globally minimal error e_i and the lower bound $\underline{e_i}$ of the current subspace is less than an established threshold ($e_i - \underline{e_i} < \epsilon$) or when the remaining subspaces are small enough (upper bound is smaller than the globally minimal error: $\overline{e_i} < e_i$).

5.1.3 Uncertainty n-radius in the n-space

The analysis of upper and lower bounds in higher dimensions cannot be easily done. When the Point-to-hyperplane error function is employed, the main issue lies in how the uncertainty radius of a sphere in higher dimensions is represented, which is the key for evaluating the lower bounds. Shown in (5.8) and (5.9), the uncertainty regions in n-dimensions centered at $f(\mathbf{T}_j(\mathbf{x}), \mathbf{M}_i)$ can be computed. However, when color and depth information are obtained, the different scale between 3D Euclidean points and intensities changes the n-sphere into a n-ellipsoid, creating a new searching region for evaluating the bounds. In order to analyze the uncertainty radius for color and depth, a 3D scenario will be considered first. It is well known that the equation of a nD-sphere can be represented in its general form by $\mathbf{M}_1^2 + \mathbf{M}_2^2 + \cdots + \mathbf{M}_n^2 = d^2$, where d is the radius and $\mathbf{M} \in \mathbb{R}^n$ the coordinates vector. In fact, the n-sphere can be represented as a particular case of an ellipsoid as:

$$\frac{\mathbf{M}_1^2}{A_1^2} + \frac{\mathbf{M}_2^2}{A_2^2} + \cdots + \frac{\mathbf{M}_n^2}{A_n^2} = 1 \tag{5.15}$$

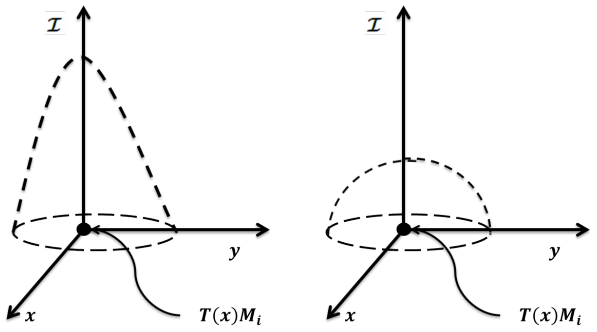
where the length of the semi-axes $(A_1, A_1, \cdots, A_n)$ is equal.

The analysis made below, considers all measurements are in the same units. However, a new analysis can be given by analyzing the n-dimensional space where the measurements are not in the same order of magnitude. For simplicity, consider first a 3D hybrid measurement vector as: $\mathbf{M}_i = \boldsymbol{\lambda}[x_i \ y_i \ I_i]^\top$, where $\boldsymbol{\lambda} = diag(\lambda_x, \lambda_y, \lambda_I)$ is the uncertainty factor between the different metric information. For the hybrid 3-vector, the search space can be computed as $\frac{\mathbf{M}_x^2}{A^2} + \frac{\mathbf{M}_y^2}{B^2} + \frac{\mathbf{M}_I^2}{C^2} = 1$, $A^2 = \frac{A_1^2}{\lambda_x^2}, B^2 = \frac{A_2^2}{\lambda_y^2}$ and $C^2 = \frac{A_3^2}{\lambda_I^2}$, where it can be seen that the semi-axes of the ellipsoid (A^2, B^2, C^2) are affected by the factor $\boldsymbol{\lambda}$. Furthermore, in the case of acquiring color information the intensity axis is constrained by the minimum and maximum intensities (0-255), which constrain the ellipsoid to a paraboloid centered at the intensity value I_i as is shown in Fig. 5.5. Based on how the bounds are estimated for 3D Euclidean points and in order to evaluate better bounds by considering the uncertainty radius in higher dimensions, the measurements can be normalized. The normalization of the measurements allows to keep a half n-sphere volume (n-paraboloid with same length in all axes), instead of a n-ellipsoid.

In the 4D case (3D Euclidean points + Intensity), the hyper-ellipsoid can be written as: $\frac{\mathbf{M}_x^2}{A^2} + \frac{\mathbf{M}_y^2}{B^2} + \frac{\mathbf{M}_z^2}{C^2} + \frac{\mathbf{M}_I^2}{D^2} = 1$, where A, B, C and D are affected by $\boldsymbol{\lambda_G}$ and λ_I. For the experiments of this paper, a normalized 4-vector $\mathbf{M}$ was considered.

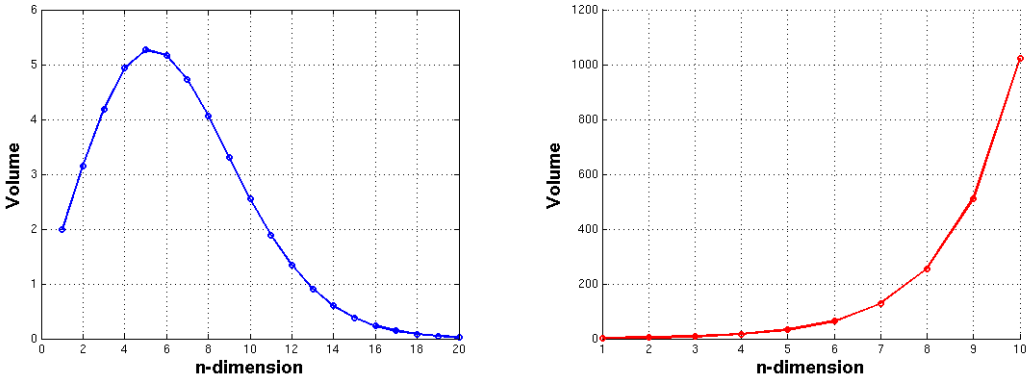
On the other hand, it is well known [72] that a n-sphere decreases its volume after the 5-th dimension while for the cube the volume increases exponentially (See Figure 5.6). Therefore, it is considered that an inscribed n-cube in the n-sphere may not properly represent the uncertainty radius (as is done for the 3D Euclidean points in 5.1.2) for higher dimensions. In order to explain the statement below, the analysis of an n-cube circumscribing a n-sphere will be presented.

The n-volume of a n-sphere of radius d falls into two series, these are referred in mathematics as the n-ball (interior of the sphere) and n-sphere (surface of the sphere) and they can be represented



(a) 3D half-paraboloid (b) 3D half-sphere

Figure 5.5: Uncertainty region of 3D hybrid points $\mathbf{M}_i = [x_i \; y_i \; I_i]^\top$. (a) The different metric between color and geometric points, which are perturbed by a transformation $\mathbf{T}(\mathbf{x})$, creates an uncertainty n-paraboloid. It is assumed that an uncertainty radius cannot be properly estimated from this. Therefore, the geometric and photometric measurements can be normalized to maintain a unitary half-sphere in n-dimensions as is shown for the 3D hybrid case in (b).



(a) n-sphere volume (b) n-cube volume

Figure 5.6: Volume of the first 20 dimension of the hypersphere and hypercube with radius $d = 1$ and half-side length $\tau = 0.5$, respectively. It can be seen that the volume of the n-cube increases exponentially while for the n-sphere the volume decreases after reaching a maximum pike in the 5th dimension.

depending on the even or odd value of n as:

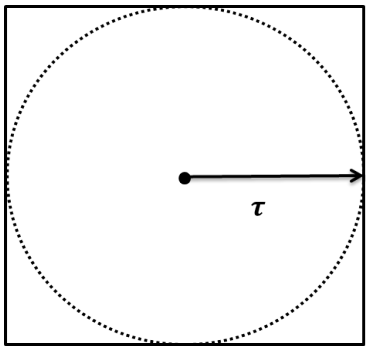
$$\begin{aligned} V(d)_{ball} &= \left(\frac{\pi^{n/2}}{(n/2)!}\right) d^n \qquad if\ n = even \\ V(d)_{sphere} &= \left(\frac{\pi^{(n-1)/2} 2^n \left(\frac{n-1}{2}\right)!}{n!}\right) d^n \quad if\ n = odd \end{aligned} \tag{5.16}$$

These two formulas can be summarized in a single expression to express the total volume of a n-sphere by the Gamma function $\Gamma(\cdot)$ as:

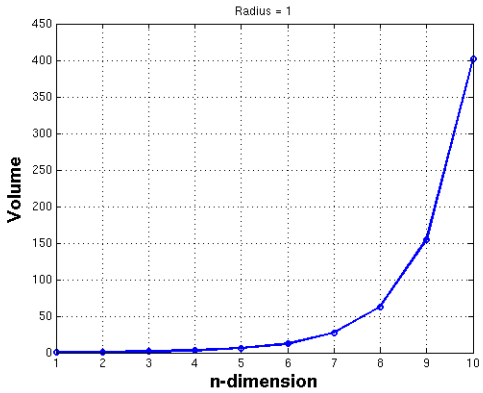
$$V(d) = \left(\frac{\pi^{n/2}}{\Gamma(n/2+1)}\right) d^n \tag{5.17}$$

In contrast, the volume of an n-cube with the half-side length τ is $V(d)_{cube} = (2\tau)^n$ which is the shortest distance between any vertices in the n-cube. The maximum distance, however, is found between opposite vertices in its diagonal, which can be expressed as the half-length $d = \sqrt{n}\tau$. This expression can lead to a counter-intuitive analysis when the n-cube encloses a n-sphere or a n-sphere encloses a n-cube.

Firstly, assume that the n-sphere is inscribed in a n-cube, where the centroids of each correspond to the same point and the side length of the n-cube is two times the radius of the n-sphere. The boundaries of the n-sphere touch all the centers of the n-faces of the n-cube. The ratio between both volumes can be evaluated as: $V(d)_n = V(d)_{cube}/V(d)_{sphere}$, which represents the total n-cube's volume occupied by the n-sphere's volume. From Figure 5.7(b) it can be noted that the ratio increases while increasing the number of dimensions.



(a) n-sphere in n-cube



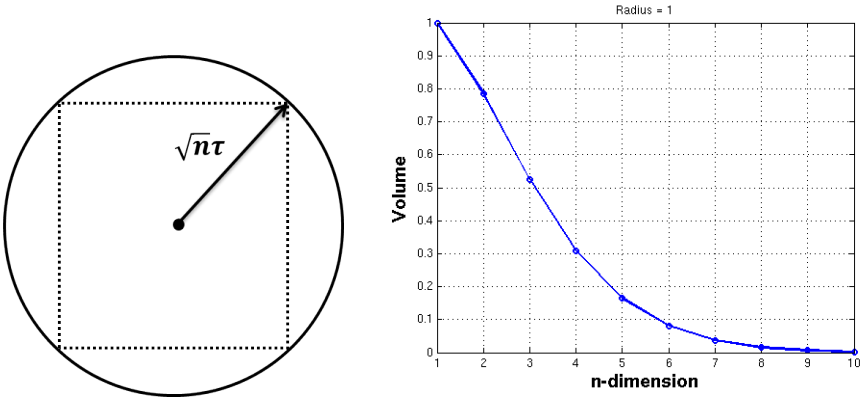
(b) n-cube/n-sphere volume

Figure 5.7: Volume of the first 10 dimension of the n-cube containing a n-sphere with radius $d = 1$.

Consider now that the n-sphere contains the n-cube, the boundaries of the n-sphere touch all vertices of the n-cube. In this case, the volume of the n-cube is strictly less than the n-sphere volume. If n is increased, the n-sphere volume increases as well as the volume of the n-cube. The volume of the inscribed n-cube can be expressed as follows:

$$V(d)_{cube} = \left(\frac{2d}{\sqrt{n}}\right)^n \tag{5.18}$$

where it can be noted that the maximum volume occupied by the n-sphere in the n-cube decreases while increasing n. An example can be seen in Figure 5.8(b), where a unit n-sphere (radius = 1) inscribed in a n-cube is projected onto the 2D plane. If the ratio volumes between the n-sphere and the n-cube are compared as $V(d)_n = V(d)_{sphere}/V(d)_{cube}$, it will be seen that almost all the mass of the n-cube is outside the unit ball and it is concentrated in the corners.



(a) n-cube in n-sphere (b) n-sphere/n-cube volume

Figure 5.8: Volume of the first 10 dimension of the n-sphere circumscribing a n-cube with radius $d = 1$.

For purposes of this thesis, the 4D Branch-and-Bound method was used to initialize the Point-to-hyperplane ICP approach when a local minima is found. The fusion of both methods can improve the convergence rate for global registration while obtaining robust alignments. The results of the proposed Global Point-to-hyperplane ICP (Go-HICP) method are shown in the following Section.

5.1.4 Combining local and global Point-to-hyperplane ICP

In the classic Go-ICP algorithm, an ICP method is often used to refine the alignment obtained by the Branch-and-Bound algorithm if the upper bound is smaller than the current best error ($\overline{\mathbf{E}(\mathbf{T}(\mathbf{x}))} < \mathbf{E}(\mathbf{T}(\mathbf{x})^*)$). The Branch-and-Bound updates the upper bounds with the current best error $\mathbf{E}(\mathbf{T}(\mathbf{x})^*)$ obtained by ICP. By performing this, more subspaces can be discarded for the Branch-and-Bound search. For purposes of this thesis, the 4D Branch-and-Bound method was used to initialize the Point-to-hyperplane ICP approach when a local minima or mistracking is present in the upper bound. An example is shown in Fig 5.9. The fusion of both methods can improve the convergence rate for global registration while obtaining robust alignments. The algorithm is presented in the following, where it can be seen how the local approach is integrated into the Branch and Bound algorithm 1. This helps to avoid local minima and discard more subspaces from the searching livesubspaces. This accelerates the convergence and improve the efficiency of the Branch-and-Bound algorithm by refining the upper bound.

Data: Reference 4D-measurement $\mathbf{M}^*$, Current 4D-measurement $\mathbf{M}$, Threshold ϵ, initial cubes C_R and C_t.
Result: Minimal error $\mathbf{E}(\mathbf{T}(\mathbf{x}))$ and 6DOF pose $\mathbf{x}$.

1 Set current best error $\mathbf{E}(\mathbf{T}(\mathbf{x}))^* = +\infty$;
2 **while** $(\mathbf{E}(\mathbf{T}(\mathbf{x})^*) - \underline{\mathbf{E}(\mathbf{T}(\mathbf{x})} > \epsilon)$ **do**
3 Divide C_R and C_t with the lowest bound $\underline{\mathbf{E}(\mathbf{T}(\mathbf{x}))}$ into 8 subspaces;
4 **for** *each sub-space* **do**
5 Compute $\overline{\mathbf{E}(\mathbf{T}(\mathbf{x}))}$ for $(C_R.C_t)$;
6 **if** $(\overline{\mathbf{E}(\mathbf{T}(\mathbf{x}))} < \mathbf{E}(\mathbf{T}(\mathbf{x})^*))$ **then**
7 Run Point-to-hyperplane with initialization $\mathbf{T}(\mathbf{x})$;
8 Update $\mathbf{E}(\mathbf{T}(\mathbf{x})^*)$ and $\mathbf{T}(\mathbf{x})^*$;
9 **else**
10 Compute $\underline{\mathbf{E}(\mathbf{T}(\mathbf{x}))}$ for (C_R, C_t) with $\mathbf{T}(\mathbf{x})$,$\boldsymbol{\psi}$;
11 **if** $(\underline{\mathbf{E}(\mathbf{T}(\mathbf{x}))} \geq \mathbf{E}(\mathbf{T}(\mathbf{x})^*)$ **then**
12 Discard C_R, C_t
13 **else**
14 Put C_R and C_t into live subspaces
15 **end**
16 **end**
17 **end**
18 **end**

Algorithm 1: Global Point-to-hyperplane ICP algorithm

The global point-to-hyperplane ICP algorithm 1 is based on the Go-ICP algorithm. Similarly, the algorithm proposed here use a nested Branch-and-Bound algorithm to improve the computational cost. A Branch-and-Bound performed directly in the 6D space (3 dimensions for rotation and 3 dimensions for translation) is inefficient since 2^6 sub-cubes are generated by using a octree. A nested Branch-and-Bound consist on an outer Branch-and-Bound algorithm for the rotational space $SO(3)$ and an inner Branch-and-Bound for translations. It can be mentioned here that the computational cost for translation is cheaper than rotations. Furthermore, nested Branch-and-Bound avoids redundant point-set operations for each rotation region. Lines 7 and 8 of Algorithm 1 show that when the upper bound is less than the current best error function value, the local approach will be called with initialization in the center of (C_R, C_t).

5.1.5 Results

In order to evaluate the Global Point-to-hyperplane ICP algorithm (Go-HICP), three experiments were performed:

1. Registration of pairs of RGB-D frames with a large initial transformation between them.
2. Keyframe visual odometry.
3. RGB-D global keyframe-to-model registration of keyframes with small overlap.

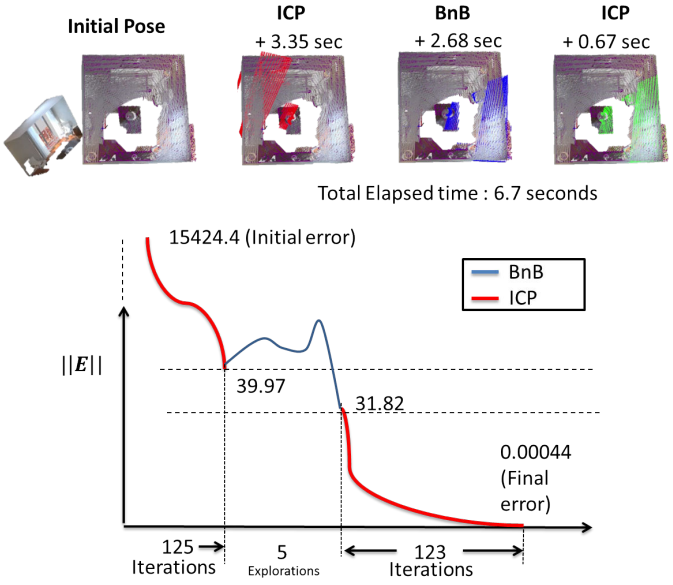


Figure 5.9: Iterations and elapsed time of Global Point-to-hyperplane ICP registration (Implementation details given in text). A subset of the cloud was extracted and transformed with a random pose. It can be seen on the top that a local minima is reached in the first trial of ICP. The Branch-and-Bound bypasses the local minima and obtains a closer solution, which is again refined by the ICP algorithm.

For all experiments, a multi-resolution pyramid was employed to improve computational efficiency (resolution 160×120 at the top). Rejection of outliers was handled with M-estimation [36]. For the matching stage, two strategies were tested: a 6-dimensional kd-tree [63] for local approaches and the Distance Transform [23] strategy for Branch-and-Bound. The method was compared among geometric-based, photometric-based and hybrid error functions which differ in how the tuning parameter $\boldsymbol{\lambda} = (\boldsymbol{\lambda_G}, \lambda_I)$ is estimated. For all the strategies, the following error function was considered:

$$\mathbf{e}_i = \begin{pmatrix} \boldsymbol{\lambda_G} \left(\mathbf{N}_i^{*\top} \left(\mathbf{P}_i^m - \mathbf{P}_i^w \right) \right) \\ \lambda_I \left(I_i^m - I_i^w \right) \end{pmatrix} \in \mathbb{R}^4 \tag{5.19}$$

where $\mathbf{P}_i^w \in \mathbb{R}^3$ is the warped 3D point and I_i^w is the warped intensity. $\mathbf{P}_i^m \in \mathbb{R}^3$ and I_i^m are the correspondences. The variants of the hybrid methods will be identified as follows:

1. *G-P2Pl*. Geometric Point-to-plane ($\lambda_I = 0$) [16].
2. *I-ESM*. Direct-method [2] ($\lambda_G = 0$) [19].
3. *H-NA-λ*. 1) + 2) + non adaptive lambda [58].
4. *H-A-λ-1*. 1) + 2) + adaptive lambda [86].
5. *H-A-λ-2*. 1) + 2) + adaptive lambda [45].

[2] In this thesis, the photometric term is minimized by using the Efficient Second Order minimization (ESM)

Experiment 1: A synthetic RGB-D frame was employed as a reference and transformed. 100 Images were synthesized by warping the reference image with a large random pose. Initially, local methods could not obtain convergence or a local minima was reached, but they could align the frames when they were initialized with the solution obtained by the Branch-and-Bound algorithm. The performance of local methods after being initialized by the Branch-and-Bound is presented in Table 5.2.

Table 5.2: Averages in Time and in Number of Iterations until Convergence for 100 Synthesized Images at Random Poses. The RGB-D image was generated from a synthetic environment, where Gaussian noise was added. The local approaches were initialized with the Branch-and-Bound method.

Method	# Iterations	Time (sec)
1) *G-P2Pl* [16]	154.34	1.8144
2) *I-ESM* [19]	144.12	1.6731
3) *H-NA-λ* [58]	115.77	1.3472
4) *H-A-λ-1* [86]	140.32	1.6583
5) *H-A-λ-2* [45]	138.93	5.2731
6) *Go-HICP* [40]	**100.56**	**1.0979**

The Global Branch-and-Bound was computationally expensive for each new position, obtaining an average of 223.9 seconds (the maximum value registered was 546.2 sec and the minimum was 3.25 sec) for 100 random poses. However, all 100 images converged to the true pose. The most computationally demanding part was the exploration of sub-cubes in the Branch-and-Bound algorithm and the time shown in Table 5.2 is the elapsed time of the local method to get convergence after the Branch-and-Bound initialization (non cumulative). It can be noted that many of the compared strategies are improved since the Branch-and-Bound gives a rough pre-alignment for these local approaches.

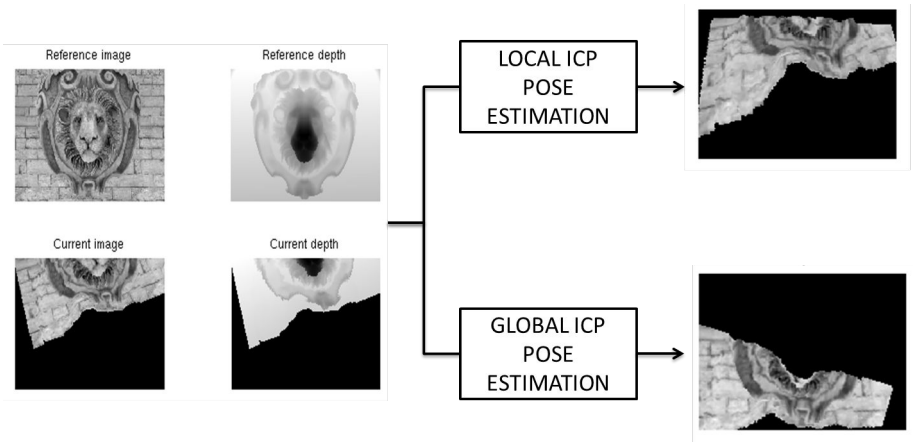


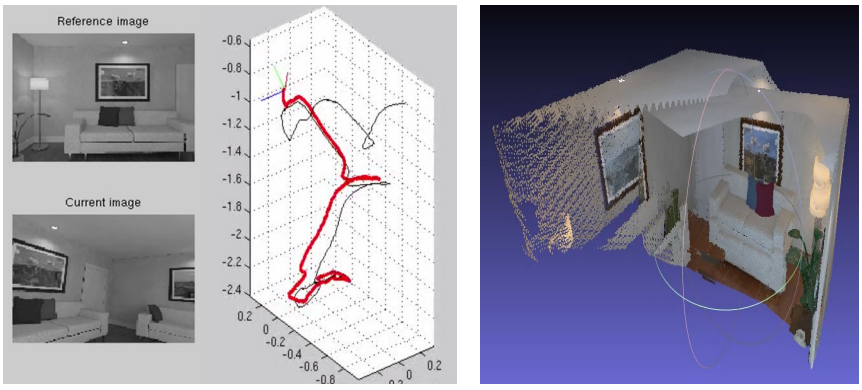
Figure 5.10: Example of a synthesized RGB-D frame with small overlapping and not close to the solution. Local approaches can get trapped into local minima but global methods can obtain the alignment by guaranteeing convergence. For testing the Global Point-to-hyperplane ICP approach, 100 images were synthesized with a random pose and the method obtained the good alignment in all of them.

Finally, the proposed Point-to-hyperplane method has been compared w.r.t. Go-ICP [90]. The results demonstrated faster convergence and a more robust estimation of the pose when color and depth are considered together for the minimization and matching. This is due to the fact that less bounds are explored and evaluated when the Point-to-hyperplane function is considered (Table 5.3).

Table 5.3: Averages in Time and in Number of Iterations for local approaches after initializing with BnB. 100 Synthesized RGB-D Images at Random Poses were synthesized at large transformations and small overlapping.

Method	**# Iterations (Local)**	**# Subspaces explored (Global)**	**Time (sec) (Local)**
Go-ICP [90]	154.34	890.3	1.8144
Go-HICP [40]	**100.56**	**735.7**	**1.0979**

Experiment 2: Well known RGB-D Benchmarks [29] were used to perform keyframe visual odometry. Keyframes are manually selected from the sequences at every $n-th$ frame. The local methods initialize the registration. If the local method cannnot obtain an optimal convergence, then it is initialized with the Branch-and-Bound algorithm. This avoid the mistracking problems in RGB-D registration as it is presented in Figure 5.11, where the local Point-to-hyperplane ICP could not properly estimate the pose for the keyframes shown. However, the branch and bound can fix the mistracking by globally estimating the pose.



(a) Mistracking in visual odometry (b) Mistracking in 3D reconstruction

Figure 5.11: Mistracking problem example presented during keyframe-to-keyframe RGB-D registration. In these situations, the algorithm run call the Branch and Bound algorithm to globally find the correct pose. This guarantees a globally consistent 3D map. The black and red trajectories are the groundtruth and the estimated trajectories, respectively.

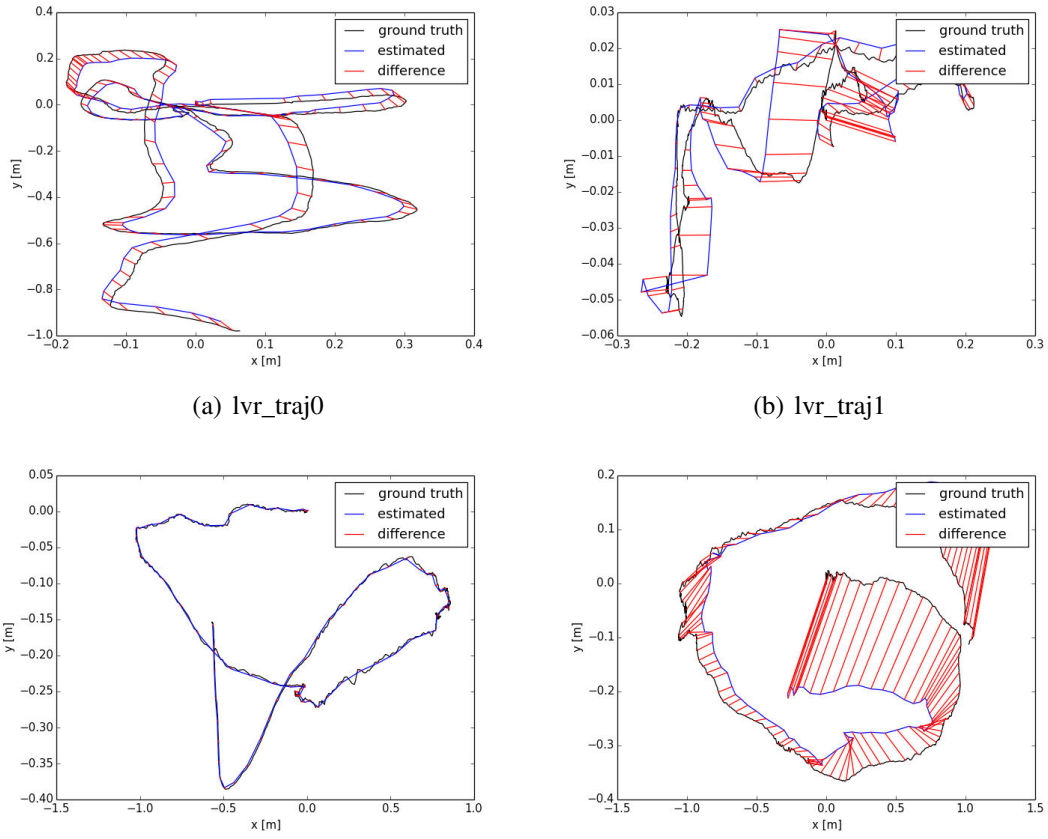
In Table 5.4, the Absolute Trajectory Error and Relative Pose Error evaluation are shown. They demonstrate that the initialization with Branch-and-Bound improve the estimations. Less error is obtained w.r.t. other local classic approaches by combining the local Point-to-hyperplane ICP, which reduces the computational cost. The time shown does not consider the computation of the normals or the kd-tree, but time for convergence. It can be observed that the local method can

get trapped in local minima during the registration process (less number of times for the Point-to-hyperplane), which is avoided by calling the Branch-and-Bound method to initialize the pose. Since less frames are employed for the 3D visual odometry, the representation of the environment can be done by considering keyframes only.

Table 5.4: Averages in Time (miliseconds), number of iterations for convergence, Absolute Trajectory Error (ATE). The number of times that the method got trapped in local minima is shown as a ratio $\#localminima(Lmin)/\#keyframes(KF)$. A RGB-D keyframe was selected at every $n-th$ frame in the sequence which is shown below the sequence name along with the number of total of RGB-D frames in the sequence as $n-frames/total-frames$.

Sequence	Method	ATE (m)		AVERAGE $\frac{Time(sec)}{\#Iterations}$	$\frac{\#Lmin}{\#KF}$
		RMSE	MEAN		
lvr/traj0	Go-ICP	0.125	0.111	**0.376/28.20**	4/75
(20/1508)	Go-HICP	**0.028**	**0.026**	0.486/42.32	**1/75**
lvr/traj1	Go-ICP	0.110	0.090	**0.265/19.40**	**2/30**
(30/965)	Go-HICP	**0.035**	**0.031**	0.368/30.59	**2/30**
lvr/traj2	Go-ICP	0.040	0.038	0.409/32.03	6/88
(10/880)	Go-HICP	**0.018**	**0.017**	**0.337/29.38**	**0/88**
lvr/traj3	Go-ICP	0.771	0.729	**0.376/28.20**	4/124
(10/1240)	Go-HICP	**0.231**	**0.218**	0.642/46.27	**2/124**

Experiment 3: Pre-selected RGB-D frames were employed to generate 3D maps of closed loop sequences. The keyframes have a small overlap between them and they are globally aligned by using the Point-to-hyperplane strategy. The method performs RGB-D registration by following the sequence as in Experiment 2, but the alignment w.r.t. the generated model is employed if the optimal solution is not found. This can be referred as a frame-to-model registration with the advantage that the model is generated simultaneously. Normally, a frame cannot be aligned when similar geometric and photometric properties are present in the overlapping area. When the entire generated cloud is considered, more correspondences can be found by using the extended measurements (color + depth).

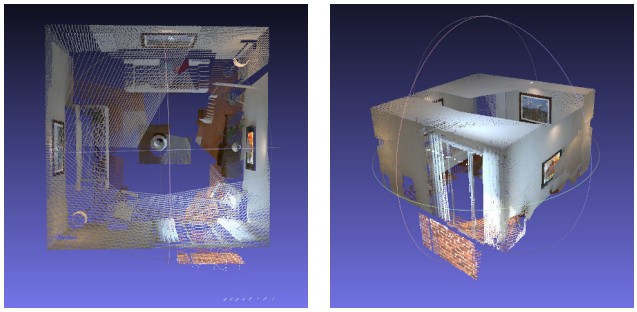


(a) lvr_traj0 (b) lvr_traj1

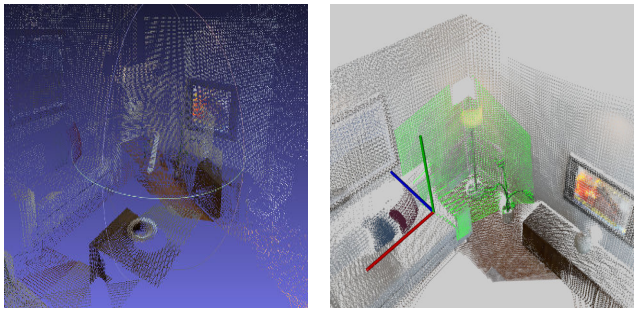
(c) lvr_traj2 (d) lvr_traj3

Figure 5.12: Examples of the Absolute Trajectory Error evaluation obtained by the Point-to-Hyperplane method combined with the BnB method. The benchmarck datasets [29] were used and a keyframe was chosen at each 10th frame of the sequence. It can be observed that closer estimations w.r.t. the groundtruth can be obtained with a refinement with the Point-to-hyperplane as the local approach while avoiding a mistracking problem.

(a) RGB-D keyframes



(b) top view (c) perspective view



(d) error (e) fixed error

Figure 5.13: Example of a 3D reconstruction (b) and (c) by considering only pre-selected keyframes (a). The proposed method obtains close solutions w.r.t. groundtruth. (d) The last image (frame 870) generates an error (d) since it contains similar geometric and color features w.r.t. the previous frame (800) but it can be fixed if the entire generated point cloud is considered for global registration (e).

CHAPTER 6

CONCLUSION AND PERSPECTIVES

Conclusion

In this work, a novel approach named Point-to-hyperplane ICP has been proposed. This approach has been employed for performing 3D reconstruction, visual odometry and real-time visual SLAM. The method has been analysed and applied for local and global RGB-D registration. The performance of the method has been demonstrated by performing visual odometry in well known benchmark datasets for both, real and synthetic environments. Futhermore, results obtained during this research have provided a comparison w.r.t. classic hybrid approaches that minimize the geometric error and photometric error simultaneously [58, 45].

The main contribution of this thesis has been the proposal and analysis of a joint error function that is minimized by considering all the acquired measurements together while avoiding the estimation of a tuning parameter which is very important for dealing with the uncertainty between different metric information and for weighting the contribution of each during the pose estimation process. The proposed Point-to-hyperplane ICP method is based on the Point-to-plane ICP approach, but it is minimized in the n-space where different measurements generates an hyperplane. In this particular work, only color and depth were considered but it has been mathematically demonstrated that the proposed approach can be extended to consider more information (e.g. semantic, temperature, etc.) while increasing the accuracy and robustness of the method. This leads to perform pose estimation processes of a network of sensors, which is commonly the case for robotic platforms that need to be localized in an unknown environment.

The invariance to the scale factor for Point-to-hyperplane RGB-D pose estimation was experimentally observed in [40] and mathematically demonstrated in [41]. In this thesis it was established that the invariance is due to the fact that the scale factor has influence only in the magnitude of the normal estimation, but no in its direction. Therefore, when the normal is projected into the error function, it does not matter how long the magnitude of the normal could be, since all coordinates

of the normal are equally scaled. This property of the proposed method have leaded to investigate the performance of the method in well known local and global approaches.

An extra contribution made during this thesis has been presented in 4.3, where direct approaches based on direct image warping have been improved by considering pixel coordinates for both, the searching of the closest points and the minimization of the photometric error. This has demonstrated that if pixel coordinates are minimized in an hybrid manner along with intensities, it can extend the convergence of classic direct approaches. As well as the Point-to-hyperplane ICP method, the searching of the closest points was handled by the FLANN library [63], which creates a balanced kd-tree. It was observed during the experimentation, that a better accuracy can be achieved if the searching of the closest points is done at each iteration of the minimization process, however it increases the computational cost. Therefore, it was concluded that the closest points can be estimated in the first iteration only when frame-to-keyframe visual odometry is performed. This increases the convergence rate and it speeds up the alignment. However, the proposed method demonstrated a similar performance of classic direct approaches when frame-to-frame visual odometry was performed.

Local Point-to-hyperplane ICP allowed to localize an RGB-D sensor in real-time. It was observed, however, that the computation of the normals in the hyperplane is computationally expensive. The method has been implemented in a real-time visual SLAM algorithm based on CPU only. The neighbouring n-points have been reduced in order to accelerate the convergence, but it was observed a reduced accuracy.

On the other hand, global Point-to-hyperplane ICP improved classic global approaches based on Branch-and-Bound algorithms. The analysis and evaluation of the bounds in this n-space were done. It was observed that if more dimensions are considered for the upper and lower bounds, then more accurate results are obtained while accelerating and guaranteeing convergence. This is due to the fact that a smaller number of generated subspaces are explored since the bounds are better constrained. In other words, the speed of the Branch-and-Bound algorithm depends on the number of subspaces that need to be tested. Therefore, it is critical to avoid subdividing unnecessarily. By better constraining the lower bounds, the fewer subdivisions will be necessary. In the experiments performed in this part of the thesis, the normals and kd-trees were pre-computed.

Perspectives

It can be mentioned here that the concern about how the tuning parameter $\boldsymbol{\lambda}$ deals with the uncertainty will be investigated in detail since the analysis of this thesis have provided a proof about how the Point-to-hyperplane approach can deal with the different scale between measurements when minimizing the hybrid error function. This research can lead to find out if there is any optimal estimation for the uncertainty value.

During the experimentation of this thesis, it was observed that one of the most demanding parts of the Point-to-hyperplane ICP approach were the estimation of the normals and the matching stage. Therefore, estimation of surface normals is one of the fundamental problems in the analysis of the Point-to-hyperplane ICP. As a future work, better strategies to estimate the normals will be implemented and the method will be accelerated with GPU as well as similar approaches in

the literature [50, 33, 4]. This will require a detailed analysis of how to estimate the normals in higher dimensions. On the other hand, the multi-dimensional kd-tree has allowed to find the nearest neighbours in n-dimensions in this thesis. For the global approach, however, the distance transform strategy was employed as well as in [90] during the Branch-and-Bound searching. The strategy can obtain true correspondences, but it is computationally expensive and its implementation in higher dimensions requires a further analysis. As a future work, kd-trees will be implemented for the global searching and the bounds will be better constrained by considering more information of the scene.

The main objective in the future, is to improve and exploit the proposed method here to be implemented in visual SLAM approaches while providing globally consistent estimations. This can be implemented in projects for autonomous driving and 3D reconstruction.

APPENDIX A

Apendix

A.1 RGB-D sensors. Comparative Table

Table A.1: RGB-D sensors. Hardware specifications, resolutions at 30 fps and type of technology are shown.

Model (Released year)	**Sensing technology**	**Dimensions** $l \times w \times h$ **(in)**	**Distance of use (m)**	**Resolution RGB** (30Hz)	**Resolution Depth** (30Hz)	**Field of view** (H,V)
 Kinect 1 (2010)	SL	$11 \times 2.5 \times 1.5$	0.4 - 3.0	640×480	320×240	57°, 43°
 Asus Xtion 1 (2011)	SL	$18 \times 3.5 \times 5$	0.8 - 3.5	1280×1024	640×480	58°, 45°
 Kinect 2 (2013)	ToF	$24.9 \times 6.6 \times 6.7$	0.5 - 4.5	1920×1080	512×424	70°, 60°
 Asus Xtion 2 (2017)	ToF	$11 \times 3.5 \times 3.5$	0.8 - 3.5	2592×1944	640×480	74°, 52°
 Intel R200 (2015)	SL	$4 \times 0.15 \times 0.4$	0.5 - 10.0	1980×1080	640×480	70°, 43°
 Intel F200 (2015)	SL	$4.3 \times 0.15 \times 0.5$	0.1 - 1.2	1980×1080	640×480	70°, 43°

Table A.2: RGB-D sensors (cont). Hardware specifications, resolutions at 30 fps and type of technology are shown.

Model (Released year)	**Sensing technology**	**Dimensions** $l \times w \times h$ **(in)**	**Distance of use (m)**	**Resolution RGB** (30Hz)	**Resolution Depth** (30Hz)	**Field of view** (H,V)
 Intel SR300 (2016)	ToF	$4.3 \times 0.15 \times 0.5$	0.2 - 1.5	1980×1080	640×480	68°, 41.5°
 Intel ZR300 (2017)	ToF	$6.1 \times 1.25 \times 0.35$	0.6 - 3.5	1980×1080	628×468	68°, 41.5°
 Intel Euclid (2017)	ToF		0.6 - 3.5	1980×1080	628×468	68°, 41.5°
 Orbbec Persee (2015)	SL	$6.7 \times 2.4 \times 2.2$	0.6 - 8.0	1280×720	640×480	60°, 49.5°
 Orbbec Astra mini (2015)	SL	$3.15 \times 0.8 \times 0.8$	0.6 - 5.0	640×480	640×480	60°, 49.5°
 Orbbec Astra (2015)	SL	$6.5 \times 1.57 \times 1.2$	0.4 - 2.0	1280×720	640×480	60°, 49.5°

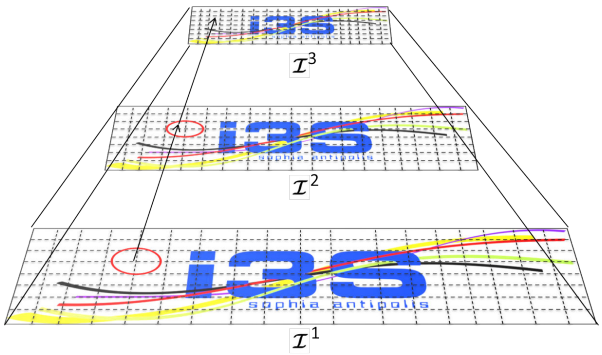


Figure A.1: Multi-resolution pyramid. The warping function is employed to generate a new image with lower resolution at each level. The tracking is performed at the top of the pyramid and the result is used as initialization for the level below. The process is repeated until achieve the base image.

A.2 Multi-resolution pyramid

A multiresolution pyramid is a set images representing the same image in multiple resolutions, where the original image is at the base of the pyramid and downsampled by a scale factor to form the levels of the pyramid [1]. In order to improve computational efficiency and convergence while estimating the pose, a multi-resolution pyramid can be constructed by using the warping function defined here as:

$$\mathbf{I}_i^{l+1} = \mathbf{I}_i^l \left(\omega(\mathbf{H}^l, \mathbf{P}_i^l)\right) \tag{A.1}$$

where $\mathbf{I}_i^{l+1}$ is the image at the next level ($l = 1$ correspond to the base of the pyramid) and $\mathbf{H}^l$ is a scale matrix that modify the calibration matrix of the RGB-D camera as:

$$\mathbf{H}^l = \begin{bmatrix} 1/\xi & 0 & 0 \\ 0 & 1/\xi & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{K} \in \mathbb{R}^{3\times3} \tag{A.2}$$

where $\xi = 2^l$ is the scale parameter.

The tracking is performed at the top level of the pyramid where the resolution of the RGB-D frame is smaller, which simplifies the computations. The obtained result at the top level is used to initialize the pose at the next level of the pyramid and the tracking is once again performed until reach the base of the pyramid (Figure A.1) for refining the alignment. This has the advantage of preserving large-scale features that can be identified in all the levels of the pyramid.

A.3 Efficient Second order Minimization

The ESM (Efficient Second order Minimization) method is an efficient algorithm that minimize the error function for matching images [**?**], it permit a quadratic convergence to the solution, but it needs the calculation of the photometric gradients of the warped image at each iteration, which takes more computational time but it could obtain more robust convergence. The function computes the difference between two images searching the closest intensity in a limited area around the warped coordinates. Therefore, the image-based pose estimation can be computed as follows:

$$\mathbf{x} = -2(\mathbf{W}\mathbf{J}_{\mathbf{ESM}})^{+}\mathbf{W}\mathbf{e}_{I} \tag{A.3}$$

where $(\cdot)^{+}$ is the pseudo-inverse operator and $\mathbf{J}_{ESM} = (\mathbf{J}_{\mathcal{I}^{*}} + \mathbf{J}_{\mathcal{I}^{w}})\mathbf{J}_{w}\mathbf{J}_{T}$. The Jacobian $\mathbf{J}_{\mathcal{I}^{*}}$ and $\mathbf{J}_{\mathcal{I}^{w}} \in \mathbb{R}^{mn\times 2mn}$ are the photometric gradients ($\nabla\mathcal{I}^{*}$, $\nabla\mathcal{I}^{w}$) of the reference and the warped image respectively, $\mathbf{J}_{w} \in \mathbb{R}^{2mn\times 3}$ is the stacked Jacobian matrix 3.23 from the warping function and $\mathbf{J}_{T} \in \mathbb{R}^{3\times 6}$ is the stacked Jacobian matrix 3.14 to parametrize $\mathbf{x}$.

A.4 Transformation space parametrization for the Branch and Bound algorithm

The relative position and orientation of an RGB-D frame with respect to another is known as a rigid transformation. A rigid motion "is an ordered pair $(\mathbf{R}(\mathbf{x}), \mathbf{t}(\mathbf{x}))$ where $\mathbf{t}(\mathbf{x}) \in \mathbb{R}^3$ and $\mathbf{R}(\mathbf{x}) \in SO(3)$. This group is known as the Special Euclidean Group $SE(3)$" [81]. Introduced in Section 2.2, a set of basic homogeneous transformations generating $SE(3)$ is given by:

$$\mathbf{t}(\mathbf{x})_x = \begin{bmatrix} 1 & 0 & 0 & X \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}(\mathbf{x})_\alpha = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\alpha) & -sin(\alpha) & 0 \\ 0 & sin(\alpha) & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{t}(\mathbf{x})_y = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & Y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}(\mathbf{x})_\beta = \begin{bmatrix} cos(\beta) & 0 & sin(\beta) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\beta) & 0 & cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.4}$$

$$\mathbf{t}(\mathbf{x})_z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{R}(\mathbf{x})_\gamma = \begin{bmatrix} cos(\gamma) & -sin(\gamma) & 0 & 0 \\ sin(\gamma) & cos(\gamma) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where any orientation or position can be expressed as rotation or translation about some axis, respectively, and they can be all represented in the following homogeneous transformation matrix:

$$\mathbf{T}(\mathbf{x}) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_{11} \\ r_{21} & r_{22} & r_{23} & t_{21} \\ r_{31} & r_{32} & r_{33} & t_{31} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{A.5}$$

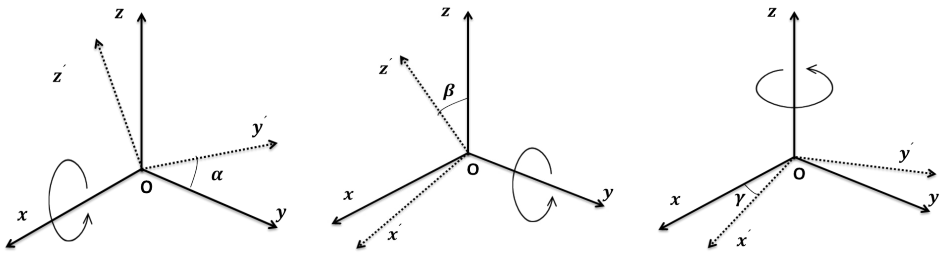
where:

$$\begin{array}{ccc} r_{11} = cos(\beta)cos(\gamma) & r_{12} = -sin(\gamma)cos(\alpha) + cos(\gamma)sin(\beta)sin(\alpha) & r_{13} = sin(\gamma)sin(\alpha) + cos(\gamma)sin(\beta)cos(\alpha) \\ r_{21} = sin(\alpha)cos(\beta) & r_{22} = cos(\gamma)cos(\alpha) + sin(\gamma)sin(\beta)sin(\alpha) & r_{23} = -cos(\gamma)sin(\alpha) + sin(\gamma)sin(\beta)cos(\alpha) \\ r_{31} = -sin(\beta) & r_{32} = cos(\beta)sin(\alpha) & r_{33} = cos(\beta)cos(\alpha) \\ t_{11} = X & t_{21} = Y & t_{31} = Z \end{array}$$

Translation can be easily represented as a sequential displacement along the axes and it can be mentioned here that computing translation is computationally less expensive than computing rotation. Whilst various strategies consider only pure rotation for the BnB and estimate translation with a local approach, the cited strategies in Table 5.1 have parametrized the translational space and have estimated the translation after finding an optimal rotation [90, 13] in a so-called nested BnB manner or by estimating rotation and translation in parallel [83]. All methods have enclosed

the translational space in a bounded 3D cube of length $[-l, l]^3$ where it is assumed that the optimal translation will be found. By separating the rotation and translation estimation, the bounding evaluation is less memory consuming.

The group of rotations has a metric structure, defined by the angle metric and a parametrization of the transformation space can be useful for representing the rotational and translational space as a geometric object for the BnB approach. The rotational space $SO(3)$ can provide the relative orientation of rigid object respect to another and it can be parametrized in its Euler angle, quaternion or angle-axis representation.



(a) Roll (b) Pitch (c) Yaw

Figure A.2: Euler angles.

The rotation matrices presented in (A.4) are referred in the literature as the Euler angle representation and summarized in the 3 DOF rotational term of (A.5) and commonly named roll, pitch and yaw angles. This representation is the most widely used technique, but for global convergence can be inefficient due to the fact that Euler angles are a redundant representation of rotations due to the fact that there are infinity number of combinations of Euler angles and they can reach a singularity (referred as Gimbal lock problem). Therefore, a minimization method that considers the entire transformation space can be computationally expensive since more non-neccessary explorations of subspaces can take place.

An alternative for the BnB application is to parametrize 3D rotations via angle-axis representation [30]. The advantage of the angle-axis representation is that it allows to describe a rotation about an arbitrary axis in space, which provides a convenient way to describe rotations. Therefore, any rotation matrix $SO(3)$ can be represented by a single unit vector $\hat{\mathbf{r}}$ defining the axis of rotation, and an angle θ as the angle of rotation about $\hat{\mathbf{r}}$ (as is shown in (**??**)). This representation can enclose the rotation space in a 3D solid sphere of radius equal to π.

$$\mathbf{r} = \mathbf{R}(\mathbf{x}) = \theta\hat{\mathbf{r}} \tag{A.6}$$

The relation with the Euler angles are given by the following expressions:

$$\theta = cos^{-1}\left(\frac{r_{11} + r_{22} + r_{33} - 1}{2}\right) \qquad \hat{\mathbf{r}} = \frac{1}{2sin(\theta)}\begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \tag{A.7}$$

The angle-axis representation has not, however, an unique solution for representing rotations about its negative angle and direction. A rotation of $-\theta$ about the $-\hat{\mathbf{r}} \in \mathbb{R}^3$ axis, represent the

same rotation of θ about $\hat{\mathbf{r}} \in \mathbb{R}^3$. Therefore, $\hat{\mathbf{r}} = -\hat{\mathbf{r}}$ which means that rotations with angles equal to π can have two corresponding angle axis representation on the surface of the sphere. If $\theta = 0$ then $\mathbf{R}(\mathbf{x})$ is the identity matrix and the axis of rotation is undefined, therefore any rotation can be represented by this form with $0 \leq \theta \leq \pi$.

For computer vision, a unit quaternion is a useful representation of orientations. Quaternions are 4D vectors that can represent 3D orientations in a more compact representation than matrices (Euler angles), which is useful while performing mathematical operations as multiplication, conjugation and interpolation between arbitrary orientations. Similarly to angle-axis representation, a unit quaternion represent a rotation by an angle around an arbitrary unit axis vector in 4 dimensions as $\mathbf{q} = \left\langle cos\frac{\theta}{2}, sin\frac{\theta}{2}\hat{\mathbf{r}} \right\rangle \in \mathbb{R}^4$ and its conversion to Euler angles is as follows:

$$\mathbf{q} = \left\langle cos\frac{\theta}{2}, sin\frac{\theta}{2}\hat{\mathbf{r}} \right\rangle = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{bmatrix} = \begin{bmatrix} cos(\alpha/2)cos(\beta/2)cos(\gamma/2) + sin(\alpha/2)sin(\beta/2)sin(\gamma/2) \\ sin(\alpha/2)cos(\beta/2)cos(\gamma/2) - cos(\alpha/2)sin(\beta/2)sin(\gamma/2) \\ cos(\alpha/2)sin(\beta/2)cos(\gamma/2) + sin(\alpha/2)cos(\beta/2)sin(\gamma/2) \\ cos(\alpha/2)cos(\beta/2)sin(\gamma/2) - sin(\alpha/2)sin(\beta/2)cos(\gamma/2) \end{bmatrix} \tag{A.8}$$

where $|\mathbf{q}| = \sqrt{q_1 + q_2 + q_3 + q_4} = 1$, which correspond to a set of vectors that form a sphere in a 4 dimensional space.

Bibliography

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. 1984, Pyramid methods in image processing. *RCA Engineer*, 29(6):33–41, 1984.

[2] Pablo Fernández Alcantarilla, Jesús Nuevo, and Adrien Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. In *BMVC*, 2013.

[3] Comport Andrew. *Robust real-time 3D tracking of rigid and articulated objects for augmented reality and robotics*. 2005.

[4] H. Badino, D. Huber, Y. Park, and T. Kanade. Fast and accurate computation of surface normals from range images. In *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[5] A. Ercil Batu Akan, M. Cetin. 3D Head tracking using normal flow constraints in a vehicle environment. In *Biennial on DSP for in-Vehicle and Mobile Systems*, Istanbul, Turkey, June 2007.

[6] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[7] Selim Benhimane and E. Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *IEEE International Conference on Intelligent Robots and Systems*, Sendai, Japan, Sept 2004.

[8] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975.

[9] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 14(2):239–256, Feb 1992.

[10] A. Parra Bustos, T. J. Chin, A. Eriksson, H. Li, and D. Suter. Fast rotation search with stereographic projections for 3d registration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(11):2227–2240, Nov 2016.

[11] P. Buyssens, M. Daisy, D. TschumperlÃ©, and O. LÃ©zoray. Superpixel-based depth map inpainting for rgb-d view synthesis. In *2015 IEEE International Conference on Image Processing (ICIP)*, pages 4332–4336, Sept 2015.

[12] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. *BRIEF: Binary Robust Independent Elementary Features*, pages 778–792. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[13] Dylan Campbell and Lars Petersson. GOGMA: globally-optimal gaussian mixture alignment. *CoRR*, abs/1603.00150, 2016.

[14] Dylan Campbell, Lars Petersson, Laurent Kneip, and Hongdong Li. Globally-optimal inlier set maximisation for simultaneous camera pose and feature correspondence. *CoRR*, abs/1709.09384, 2017.

[15] Massimo Camplani and Luis Salgado. Efficient spatio-temporal hole filling strategy for kinect depth maps. volume 8290, pages 82900E–82900E–10, 2012.

[16] Y. Chen and G. Medioni. Object modeling by registration of multiple range images. In *IEEE International Conference on Robotics and Automation*, Sacramento, CA, USA, Apr 1991.

[17] Dmitry Chetverikov, Dmitry Stepanov, and Pavel Krsek. Robust euclidean alignment of 3d point sets: the trimmed iterative closest point algorithm. *Image and Vision Computing*, 23(3):299 – 309, 2005.

[18] H. J. Chien, C. C. Chuang, C. Y. Chen, and R. Klette. When to use what feature? sift, surf, orb, or a-kaze features for monocular visual odometry. In *2016 International Conference on Image and Vision Computing New Zealand (IVCNZ)*, pages 1–6, Nov 2016.

[19] Andrew I Comport, Ezio Malis, and Patrick Rives. Accurate quadrifocal tracking for robust 3d visual odometry. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 40–45. IEEE, 2007.

[20] Dartmouth. Interpolation methods, 2017. Online; accessed 5-DEC-2017.

[21] L. Douadi, M.-J. Aldon, and A. Crosnier. Pair-wise registration of 3D/Color data sets with ICP. In *IEEE International Conference on Intelligent Robots and Systems*, Beijing, China, Oct 2006.

[22] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[23] Andrew Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21/14:1145–1153, January 2003.

[24] Natasha Gelfand, Niloy J. Mitra, Leonidas J. Guibas, and Helmut Pottmann. Robust global registration. In *Proceedings of the Third Eurographics Symposium on Geometry Processing*, SGP '05, Aire-la-Ville, Switzerland, Switzerland, 2005. Eurographics Association.

[25] Pierre Georgel, Selim Benhimane, and Nassir Navab. A unified approach combining photometric and geometric information for pose estimation. In *Proceedings of the British Machine Vision Conference*, Leeds, United Kingdom, September 2008.

[26] C. A. Glasbey and K. V. Mardia. A review of image-warping methods. *Journal of Applied Statistics*, 25(2):155–171, 1998.

[27] Maciej Halber and Thomas A. Funkhouser. Structured global registration of RGB-D scans in indoor environments. *CoRR*, abs/1607.08539, 2016.

[28] T. Han, C. Xu, R. Loxton, and L. Xie. Bi-objective optimization for robust rgb-d visual odometry. In *The 27th Chinese Control and Decision Conference (2015 CCDC)*, pages 1837–1844, May 2015.

[29] A. Handa, T. Whelan, J. McDonald, and A.J. Davison. A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. In *IEEE International Conference on Robotics and Automation*, Hong Kong, China, May 2014.

[30] Richard I. Hartley and Fredrik Kahl. Global optimization through rotation space search. *International Journal of Computer Vision*, 82(1):64–79, Apr 2009.

[31] Peter Henry, Michael Krainin, Evan Herbst, Xiaofeng Ren, and Dieter Fox. *The 12th International Symposium on Experimental Robotics*, chapter RGB-D Mapping: Using Depth Cameras for Dense 3D Modeling of Indoor Environments. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[32] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke. Registration with the point cloud library: A modular framework for aligning in 3-d. *IEEE Robotics Automation Magazine*, 22(4):110–124, Dec 2015.

[33] Stefan Holzer, Radu Bogdan Rusu, M. Dixon, Suat Gedikli, and Nassir Navab. Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images. *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2684–2689, 2012.

[34] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *J. Opt. Soc. Am. A*, 4(4):629–642, Apr 1987.

[35] Daniel Huber and Martial Hebert. Fully automatic registration of multiple 3d data sets. 21(7):637–650, July 2003.

[36] P.J. Huber, J. Wiley, and W. InterScience. *Robust statistics*. Wiley New York, 1981.

[37] R. Hulik, V. Beran, M. Spanel, P. Krsek, and P. Smrz. Fast and accurate plane segmentation in depth maps for indoor scenes. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1665–1670, Oct 2012.

[38] Michal Irani and P. Anandan. About direct methods. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, pages 267–277, London, UK, 1999. Springer-Verlag.

[39] Fernando I. Ireta Muñoz and Andrew I. Comport. Direct matching for improving image-based registration. In *IEEE International Conference on Intelligent Robots and Systems*, Hamburg, Germany, 2015.

[40] Fernando I. Ireta Muñoz and Andrew I. Comport. Point-to-hyperplane RGB-D pose estimation: Fusing photometric and geometric measurements. In *IEEE International Conference on Intelligent Robots and Systems*, Deajeon, South Korea, 2016.

[41] Fernando I. Ireta Muñoz and Andrew I. Comport. A proof that fusing measurements using point-to-hyperplane registration is invariant to relative scale. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Baden-Baden, Germany, 2016.

[42] Fernando I. Ireta Muñoz and Andrew I. Comport. Global point-to-hyperplane icp: Local and global pose estimation by fusing color and depth. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, Daegu, South Korea, 2017.

[43] Fernando I. Ireta Muñoz and Andrew I. Comport. Point-to-hyperplane icp: Fusing different metric measurements for pose estimation. *Advanced Robotics Journal*, September 2017.

[44] Andrew Edie Johnson and Sing Bing Kang. "registration and integration of textured 3d data ". *Image and Vision Computing*, 17(2):135 – 147, 1999.

[45] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In *IEEE International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 2013.

[46] K. Khoshelham. Accuracy Analysis of Kinect Depth Data. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3812:133–138, September 2011.

[47] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[48] Kenneth L. Lange, Roderick J. A. Little, and Jeremy M. G. Taylor. Robust statistical modeling using the t distribution. *Journal of the American Statistical Association*, 84(408):881–896, 1989.

[49] E. L. Lawler and D. E. Wood. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[50] M. Liu, F. Pomerleau, F. Colas, and R. Siegwart. Normal estimation for pointcloud using gpu based sparse tensor voting. In *2012 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 91–96, Dec 2012.

[51] Cristian Lorenz, Tobias Klinder, and Jens von Berg. *Feature-Based Registration Techniques*, pages 85–102. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[52] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, Nov 2004.

[53] H. L. Luo, C. T. Shen, Y. C. Chen, R. H. Wu, and Y. P. Hung. Automatic multi-resolution joint image smoothing for depth map refinement. In *2013 2nd IAPR Asian Conference on Pattern Recognition*, pages 284–287, Nov 2013.

[54] A. Makadia, A. Patterson, and K. Daniilidis. Fully automatic registration of 3d point clouds. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 1297–1304, June 2006.

[55] Renato Martins, Eduardo Fernandez-Moral, and Patrick Rives. Adaptive Direct RGB-D Registration and Mapping for Large Motions. In *Asian Conference on Computer Vision, ACCV 2016*, Taipei, Taiwan, November 2016.

[56] Desire L. Massart, Leonard Kaufman, Peter J. Rousseeuw, and Annick Leroy. Least median of squares: a robust method for outlier and model error detection in regression and calibration. *Analytica Chimica Acta*, 187:171 – 179, 1986.

[57] G. J. McLachlan and D. Peel. *Finite mixture models*. Wiley Series in Probability and Statistics, New York, 2000.

[58] M. Meilland and A.I. Comport. On unifying key-frame and voxel-based dense visual SLAM at large scales. In *International Conference on Intelligent Robots and Systems*, Tokyo, Japan, November 2013. IEEE/RSJ.

[59] Nicolas Mellado, Dror Aiger, and Niloy J. Mitra. Super 4pcs fast global pointcloud registration via smart indexing. *Computer Graphics Forum*, 33(5):205–215, 2014.

[60] H. Men, B. Gebre, and K. Pochiraju. Color point cloud registration with 4D ICP algorithm. In *IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011.

[61] J. Pauli Michael Korn, M. Holzkothen. Color supported generalized-ICP. In *International Conference on Computer Vision Theory and Applications*, Lisbon, Portugal, 2014.

[62] L. Morency and T. Darrell. Stereo tracking using ICP and normal flow constraint. In *16th International Conference on Pattern Recognition*, Quebec, Canada, 2002.

[63] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009.

[64] Marius Muja and David G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014.

[65] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE ISMAR*. IEEE, October 2011.

[66] A. Nuchter, K. Lingemann, and J. Hertzberg. Cached k-d tree search for icp algorithms. In *3-D Digital Imaging and Modeling, 2007. 3DIM '07. Sixth International Conference on*, pages 419–426, Aug 2007.

[67] C. Olsson, F. Kahl, and M. Oskarsson. Branch-and-bound methods for euclidean registration problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5):783–794, May 2009.

[68] Sergio Orts-Escolano, Christoph Rhemann, Sean Fanello, Wayne Chang, Adarsh Kowdle, Yury Degtyarev, David Kim, Philip L. Davidson, Sameh Khamis, Mingsong Dou, Vladimir Tankovich, Charles Loop, Qin Cai, Philip A. Chou, Sarah Mennicken, Julien Valentin, Vivek Pradeep, Shenlong Wang, Sing Bing Kang, Pushmeet Kohli, Yuliya Lutchyn, Cem Keskin, and Shahram Izadi. Holoportation: Virtual 3d teleportation in real-time. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 741–754, New York, NY, USA, 2016. ACM.

[69] Chavdar Papazov and Darius Burschka. *Stochastic Optimization for Rigid Point Set Registration*, pages 1043–1054. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[70] Chavdar Papazov and Darius Burschka. Stochastic global optimization for robust point set registration. *Comput. Vis. Image Underst.*, 115(12):1598–1609, December 2011.

[71] Sylvain Paris, Pierre Kornprobst, and Jack Tumblin. *Bilateral Filtering*. Now Publishers Inc., Hanover, MA, USA, 2009.

[72] Harold R. Parks. The volume of the unit n-ball. *Mathematics Magazine*, 86(4):270–274, 2013.

[73] Emanuele RodolÃ , Andrea Albarelli, Daniel Cremers, and Andrea Torsello. A simple and effective relevance-based point sampling for 3d shapes. *Pattern Recognition Letters*, 59:41 – 47, 2015.

[74] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[75] D. Ruprecht and H. Muller. Image warping with scattered data interpolation. *Computer Graphics and Applications, IEEE*, 15(2):37–43, Mar 1995.

[76] Hamed Sarbolandi, Damien Lefloch, and Andreas Kolb. Kinect range sensing: Structured-light versus time-of-flight kinect. *CoRR*, abs/1505.05459, 2015.

[77] Michael Schmeing and Xiaoyi Jiang. *Color Segmentation Based Depth Image Filtering*, pages 68–77. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[78] A. Segal, D. Haehnel, and S. Thrun. Generalized-ICP. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.

[79] Roland Siegwart and Illah R. Nourbakhsh. *Introduction to Autonomous Mobile Robots*. Bradford Company, Scituate, MA, USA, 2004.

[80] N. Silberman and R. Fergus. Indoor scene segmentation using a structured light sensor. In *Proceedings of the International Conference on Computer Vision - Workshop on 3D Representation and Recognition*, 2011.

[81] M.W. Spong, S. Hutchinson, and M. Vidyasagar. *Robot Modeling and Control*. Wiley, 2005.

[82] F. Steinbruecker, C. Kerl, J. Sturm, and D. Cremers. Large-scale multi-resolution surface reconstruction from RGB-D sequences. In *International Conference on Computer Vision*, Sydney, Australia, 2013.

[83] Julian Straub, Trevor Campbell, Jonathan P. How, and John W. Fisher III. Efficient globally optimal point cloud alignment using bayesian nonparametric mixtures. *CoRR*, abs/1603.04868, 2016.

[84] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *IEEE International Conference on Intelligent Robot and Systems*, Vilamoura, Algarve, Portugal, Oct. 2012.

[85] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. Visual slam algorithms: a survey from 2010 to 2016. *IPSJ Transactions on Computer Vision and Applications*, 9(1):16, Jun 2017.

[86] T.M. Tykkälä, C. Audras, and A.I Comport. Direct Iterative Closest Point for Real-time Visual Odometry. In *The Second international Workshop on Computer Vision in Vehicle Technology: From Earth to Mars in conjunction with the International Conference on Computer Vision*, Barcelona, Spain, November 2011.

[87] T. Whelan, H. Johannsson, M. Kaess, J.J. Leonard, and J. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 2013.

[88] Thomas Whelan, Michael Kaess, Hordur Johannsson, Maurice Fallon, John J. Leonard, and John Mcdonald. Real-time large-scale dense RGB-D SLAM with volumetric fusion. *International Journal of Robotics Research*, 34(4-5):598–626, April 2015.

[89] Thomas Whelan, Stefan Leutenegger, Renato Salas Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. In *Proceedings of Robotics: Science and Systems*, Rome, Italy, July 2015.

[90] Jiaolong Yang, Hongdong Li, and Yunde Jia. Go-icp: Solving 3d registration efficiently and globally optimally. In *Proceedings of the 2013 IEEE International Conference on Computer Vision*, ICCV '13, pages 1457–1464, Washington, DC, USA, 2013. IEEE Computer Society.

[91] Qian-Yi Zhou and Vladlen Koltun. Color map optimization for 3D reconstruction with consumer depth cameras. *ACM Trans. Graph.*, July 2014.

[92] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. *Fast Global Registration*, pages 766–782. Springer International Publishing, Cham, 2016.